CP

>> CERTIFICADO DE PROFESIONALIDAD

MF0224_3

200 HORAS DE FORMACIÓN

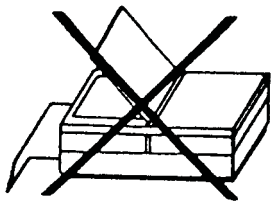
ADMINISTRACIÓN DE SISTEMAS GESTORES DE BASES DE DATOS

PABLO VALDERREY SANZ

STARBOOK

Formación EMPLEO

www.starbook.es/cp



La ley prohíbe
Copiar o Imprimir este libro

ADMINISTRACIÓN DE SISTEMAS GESTORES DE BASES DE DATOS
© Pablo Valderrey Sanz

© De la Edición Original en papel publicada por Editorial RA-MA
ISBN de Edición en Papel: 978-84-9265-078-1

Editado por:
RA-MA, S.A. Editorial y Publicaciones
Calle Jarama, 33, Polígono Industrial IGARSA
28860 PARACUELLOS DE JARAMA, Madrid
Teléfono: 91 658 42 80
Fax: 91 662 81 39
Correo electrónico: editorial@ra-ma.com
Internet: www.ra-ma.es y www.ra-ma.com

Maquetación: Gustavo San Román Borrueco
Diseño Portada: Antonio García Tomé

ISBN: 978-84-9964-332-8

E-Book desarrollado en España en septiembre de 2014

Administración de Sistemas Gestores de Bases de Datos

Pablo Valderrey Sanz

A mis alumnos.

ÍNDICE

Capítulo

1

TIPOS DE ALMACENAMIENTO DE LA INFORMACIÓN

1.1 SISTEMAS LÓGICOS DE ALMACENAMIENTO DE LA INFORMACIÓN

Habitualmente se almacenan los datos en dispositivos tales como discos duros, discos flexibles, discos ópticos, memorias USB, etc. (memoria secundaria). Estas memorias se caracterizan por ser más lentas que la memoria principal del ordenador (memoria primaria), pero también disponen de más espacio de almacenamiento, y no son volátiles, es decir, no pierden su contenido al desconectar el ordenador.

Para almacenar datos en estas memorias secundarias es necesario agruparlos en estructuras lógicas que denominaremos archivos o ficheros. En los párrafos siguientes se definen algunos conceptos fundamentales relativos a los ficheros, esenciales para el almacenamiento de la información.

1.2 ALMACENAMIENTO EN FICHEROS

Podemos definir un *archivo* o *fichero* como un conjunto de información relacionada entre sí y estructurada en unidades más pequeñas, llamadas *registros*. Un *registro* es cada una de las unidades individuales en las que se divide un fichero. Cada registro debe contener datos pertenecientes a un mismo tema. Además, cada registro es una estructura de datos, es decir, está compuesto de otros datos más simples, que llamaremos *campos*. Un *campo* es cada uno de los elementos que constituyen un registro. Cada campo se caracteriza por un identificador que lo distingue de los otros campos del registro, y por el tipo de dato que tiene asociado, que, a su vez, puede ser simple (número entero, carácter, lógico, etc.) o compuesto (cadena de caracteres, fecha, vector, etc.).

Como ejemplo consideramos un fichero que contiene información relativa a los datos personales de un conjunto de personas. Se presenta en la tabla siguiente:

FICHERO	CAMPOS				
	NIF	**Nombre**	**Apellido**	**Teléfono**	**Email**
REGISTROS	01823I	Antonio	González	8264531	a@domino.es
	14131H	Juan	Valdés	9675423	j@dominio.es
	92154Z	Eva	Prieto	8675132	e@dominio.es
	…….	……..	……..	……..	……..

Toda esa información está distribuida en registros, que son cada una de las filas de la tabla. Cada registro, por tanto, contiene los datos pertenecientes *a una sola persona.* Los registros se dividen en campos, que son cada una de las unidades de información que contiene cada registro. Cada campo se corresponde con una columna de la tabla.

Si el tipo de dato de un campo es complejo, el campo puede dividirse en *subcampos*. Por ejemplo, si un campo contiene una fecha, se puede dividir en tres subcampos que contengan, respectivamente, el día, el mes y el año. Para diferenciar a un registro de otro es conveniente que alguno de los campos tenga un valor distinto en todos los registros del archivo. Este campo, que identifica unívocamente cada registro, se denomina *campo clave* o, simplemente, clave. En el ejemplo anterior, el campo clave puede ser NIF, ya que será diferente para cada una de las personas que forman el archivo.

1.2.1 Registros físicos y registros lógicos

El concepto de archivo visto hasta ahora se corresponde con el concepto de *registro lógico*. Alternativamente se puede considerar el concepto de *registro físico*.

Un *registro físico*, también llamado *bloque*, es la cantidad de información que el sistema operativo puede enviar o recibir del soporte de memoria secundaria en una operación de escritura o lectura. Esta cantidad depende del hardware.

El registro físico puede ser mayor que el registro lógico, con lo cual, en una sola operación de lectura o escritura, se podrían transferir varios registros lógicos. También puede ocurrir lo contrario, es decir, que el registro físico sea de menor tamaño que el lógico, lo que haría que para transferir un registro lógico fueran necesarias varias operaciones de lectura o escritura. Se llama *factor de bloqueo* al número de registros lógicos contenidos en un registro físico.

1.2.2 Registros de longitud fija y de longitud variable

Dependiendo de la longitud de los campos que forman cada registro podemos clasificar estos en registros de longitud fija y registros de longitud variable.

Los *registros de longitud fija* son los que ocupan siempre el mismo espacio a lo largo de todo el archivo. Dentro de estos registros, podemos encontrar varias posibilidades:

- Igual número de campos por registro e igual longitud de todos los campos.
- Igual número de campos por registro y distinta longitud de cada campo, aunque igual en todos los registros.
- Igual número de campos por registro y distinta longitud de cada campo, pudiendo ser diferente en cada registro.
- Distinto número de campos por registro y distinta longitud de cada campo en cada registro.

Aunque es menos habitual, pudiera ocurrir que cada registro del archivo tuviera una longitud propia y diferente del resto. Estaríamos en el caso de *registros de longitud variable*. En este tipo de archivos, es necesario programar algún mecanismo para averiguar cuál es el principio y el final de cada registro.

1.3 OPERACIONES TÍPICAS CON FICHEROS

En un fichero o archivo se puede realizar operaciones sobre cada registro individual o bien sobre todo el archivo, es decir, sobre todos los registros a la vez.

1.3.1 Operaciones con registros individuales

- *Inserción (alta)*: consiste en añadir un registro al fichero. El registro puede añadirse al final del fichero o entre dos registros que ya existieran previamente.
- *Borrado (baja)*: consiste en eliminar un registro existente.
- *Modificación*: consiste en cambiar el dato almacenado en uno o varios de los campos del registro.

- *Consulta*: consiste en leer el dato almacenado en uno o varios de los campos del registro.

1.3.2 Operaciones sobre el archivo completo

Además de manipular cada componente del archivo (registros y campos), también se pueden llevar a cabo operaciones con la totalidad del archivo, como son las siguientes:

- *Creación*: la creación del archivo consiste en crear una entrada en el soporte de memoria secundaria y asignarle un nombre para identificar en el futuro a los datos que contiene.

- *Apertura*: antes de trabajar con un archivo es necesario abrirlo, creándose así un canal de comunicación entre el programa y el archivo a través del cual se pueden leer y escribir datos. Los archivos solo deben permanecer abiertos el tiempo estrictamente necesario.

- *Cierre*: es importante cerrar el canal de comunicación con el archivo cuando no va a usarse en un futuro inmediato, porque todos los sistemas limitan el número máximo de archivos que pueden estar abiertos simultáneamente. También es importante porque evita un acceso accidental al archivo que pueda deteriorar la información almacenada en él.

- *Ordenación*: permite establecer un orden entre los registros del archivo.

- *Copiado*: crea un nuevo archivo con la misma estructura y contenido que el fichero original.

- *Concatenación*: consiste en crear un archivo nuevo que contenga los registros de otros dos archivos previamente existentes, de manera que primero aparezcan todos los registros de un archivo y, a continuación, todos los del otro.

- *Mezcla*: parecida a la concatenación, pero el archivo resultante contendrá todos los registros de los dos archivos originales mezclados y ordenados.

- *Compactación*: esta operación solo se realiza sobre archivos en los cuales el borrado de registros se ha realizado sin eliminar físicamente el registro, sino únicamente marcándolo como borrado para no procesarlo. Después de la compactación, todos los registros marcados como borrados quedan borrados físicamente, con lo que se libera espacio en el dispositivo de almacenamiento.

- *Borrado*: es la operación contraria a la creación, ya que elimina la entrada en el dispositivo de almacenamiento, con lo que se pierde toda la información almacenada en el archivo.

1.4 ORGANIZACIÓN DE LOS FICHEROS

La *organización de los archivos o ficheros* es la forma en que los datos son estructurados y almacenados en el dispositivo de almacenamiento. El tipo de organización se establece durante la fase de creación del archivo y es invariable durante toda su vida. La organización puede ser *secuencial* o *relativa* (o una combinación de ambas), como enseguida veremos.

El *tipo de acceso* al archivo es el procedimiento que se sigue para situarnos sobre un registro concreto para hacer alguna operación con él. *Esto es lo que realmente le interesa al programador*: cómo acceder a los registros de archivo. El tipo de acceso está condicionado por el tipo de organización física del archivo.

1.4.1 Ficheros de organización secuencial

La forma más simple de estructura de archivo es el *archivo secuencial*. En este tipo de archivo, los registros se sitúan físicamente en el dispositivo en el orden en el que se van escribiendo, uno tras otro y sin dejar huecos entre sí. El acceso a los registros también debe hacerse en orden, de modo que para acceder al registro N es necesario pasar primero por el registro 1, luego por el 2, luego por el 3, y así hasta llegar al registro N.

Los archivos secuenciales se utilizaban mucho cuando el soporte de almacenamiento masivo más usual era la cinta magnética. Hoy día, con nuestros discos duros y memorias flash, no es habitual encontrarse con archivos de organización interna secuencial. Pero sí que se utiliza el acceso secuencial (aunque físicamente el archivo no lo sea), por su simplicidad y porque es suficientemente útil en muchas ocasiones (por ejemplo, en aplicaciones de proceso por lotes). Ahora bien, si el programa necesita acceder a registros individuales y no consecutivos, el acceso secuencial ofrece un rendimiento pobre y es preferible el acceso directo, que luego veremos.

Los archivos secuenciales (sobreentiéndase "archivos de acceso secuencial") tienen un indicador de posición (o *cursor*) que señala qué registro fue el último que se accedió. Al abrir el archivo, el indicador se sitúa en el primer campo del primer registro. Cada acceso sobre el archivo desplazará el indicador de posición hacia el siguiente registro, hasta que ya no haya más registros que leer.

Cuando un archivo secuencial se abre para escribir datos en él, el indicador de posición se sitúa justo después del último byte del mismo, de manera que los datos solo se pueden añadir al final.

La organización secuencial cuenta con varias *ventajas*:

- Es la más sencilla de manejar para el programador.
- Si hay que acceder a un conjunto de registros consecutivos, o a todo el archivo, es el método más rápido.
- No deja espacios entre registro y registro, por lo que se optimiza el uso del espacio en la memoria secundaria.

Los mayores *inconvenientes* son los siguientes:

- Para consultar datos individuales, hay que recorrer todo el archivo desde el principio. Es decir, el acceso a registros individuales es, en general, lento.
- Las operaciones de inserción y eliminación de registros solo pueden hacerse al final del archivo. Hacerlas con registros intermedios representa mover grandes bloques de información y, por lo tanto, consumir mucho tiempo.

1.4.2 Ficheros de organización relativa

La idea básica de la organización relativa consiste en guardar físicamente los registros en lugares de la memoria secundaria *no consecutivos*. Pero, entonces, ¿cómo podemos encontrar dónde está cada registro?

La única solución es utilizar un *campo clave* de entre todos los del registro. Ese campo clave, que suele ser numérico, permite averiguar la dirección física donde está almacenado el registro en la memoria secundaria mediante un algoritmo de transformación. Por eso, la clave suele denominarse *dirección de memoria lógica*, para distinguirla de la *dirección de memoria física* donde efectivamente se encuentra guardado el registro.

Esta transformación de claves para obtener direcciones físicas se denomina *hashing* y se realiza mediante las denominadas funciones *hash*. Los archivos relativos son más versátiles que los secuenciales porque permiten acceder a cualquier parte del fichero en cualquier momento, como si fueran arrays. Las operaciones de lectura y escritura pueden hacerse en cualquier punto del archivo.

Los archivos con organización relativa tienen dos variantes: los archivos *directos* y los archivos *aleatorios* (o *indirectos*).

1.4.3 Ficheros de organización relativa directa o de acceso directo

Entre los archivos con organización relativa los más sencillos son los *directos*. En ellos, el campo clave de cada registro debe ser de tipo numérico, e identifica directamente el registro físico donde está almacenado. La función hash, en este caso, es la más simple

posible, ya que no transforma la clave. El valor de la clave está en relación con la capacidad máxima del dispositivo de almacenamiento, no pudiendo almacenar registros cuya clave esté por encima de este límite.

En estos archivos no puede haber dos registros con la misma clave, porque ambos ocuparían la misma posición física, solapándose. Esto es lo que se llama una *colisión* y debe ser evitada.

Las ventajas de los archivos de acceso directo son:

- Permiten acceder al archivo de dos maneras: directamente (a través de la clave de cada registro) y secuencialmente.
- Permiten realizar operaciones de lectura y escritura simultáneamente.
- Son muy rápidos al tratar registros individuales.

Los inconvenientes *principales de los archivos de acceso directo* son:

- El acceso secuencial, del principio al fin del fichero, puede ser muy lento porque podemos encontrarnos con muchos huecos, es decir, posiciones que no están siendo usadas. Existen técnicas de programación avanzadas para el acceso secuencial eficiente a ficheros directos.
- Relacionado con lo anterior, pueden quedar muchos huecos libres en el dispositivo de memoria secundaria, desaprovechándose el espacio.

1.4.4 Ficheros de organización relativa aleatoria (o indirecta). Acceso aleatorio

Se denominan así a los archivos relativos que empleen alguna función hash para transformar la clave y conseguir así la dirección física. La función hash puede ser muy sencilla. Dependiendo de la función hash empleada pueden surgir *colisiones*, es decir, claves que proporcionan la misma dirección física.

Por lo tanto, la elección de una función hash adecuada es crucial para el correcto rendimiento y funcionamiento de este tipo de archivos. Existen multitud de funciones hash adaptadas a los más diversos problemas que ofrecen un máximo aprovechamiento del espacio y un mínimo número de colisiones.

Las ventajas de los archivos de acceso aleatorio son similares a las de los archivos de acceso directo, y entre los inconvenientes podemos quitar el de dejar muchos huecos libres, siempre que, como hemos visto, la función hash elegida sea adecuada.

1.4.5 Ficheros de organización indexada

Los archivos con organización indexada tienen una mezcla entre las organizaciones secuencial y relativa. Se pretende aprovechar las ventajas de las dos organizaciones, evitando al mismo tiempo sus inconvenientes.

Los archivos indexados están divididos en tres zonas o áreas:

- *Área primaria*. En esta área se encuentran almacenados los registros del archivo secuencial. Es decir, el área primaria es, en realidad, un archivo secuencial corriente. Los registros deben estar ordenados (normalmente, se hará en orden creciente según sus claves). El área primaria suele estar segmentada, es decir, dividida en trozos o segmentos. En cada segmento se almacenan N registros en posiciones de memoria consecutivas. Para acceder a un registro individual, primero hay que acceder a su segmento y, una vez localizado el segmento, buscar secuencialmente el registro concreto.

- *Área de índices*. Se trata, en realidad, de un segundo archivo secuencial agregado al primero. Pero es un archivo especial, cuyos registros solo tienen dos campos: uno contiene la clave del último registro de cada segmento, y otro contiene la dirección física de comienzo de cada segmento.

- *Área de excedentes*. Puede ocurrir que los segmentos del área primaria se llenen y no puedan contener algún registro. Esos registros van a parar a un área de excedentes u *overflow*.

Para acceder a un registro concreto en un archivo indexado, el procedimiento es el siguiente:

- Primero, buscamos secuencialmente en el área de índices la dirección de comienzo del segmento donde está el registro que queremos buscar.

- Segundo, hacemos un acceso directo al primer registro del segmento.

- Después hacemos un recorrido secuencial dentro del segmento hasta localizar el registro.

- Si el registro no se encuentra, acudimos al área de excedentes y hacemos un nuevo recorrido secuencial en ella para intentar localizarlo allí.

Se observa que los archivos indexados mezclan los accesos secuenciales con los accesos directos.

1.4.6 Ficheros de entrada, salida, texto plano y binarios

Además de los tipos de archivos que ya hemos visto (según su organización: secuenciales y relativos con todas sus variedades), podemos hacer otras dos clasificaciones de los archivos:

- Según la dirección del flujo de datos:
  - *De entrada*: los datos se leen por el programa desde el archivo.
  - *De salida*: los datos se escriben por el programa hacia el archivo.
  - *De entrada/salida*: los datos pueden ser escritos o leídos.
- Según el tipo de valores permitidos a cada byte:
  - *De texto*: solo permiten guardar caracteres o, mejor dicho, su código ASCII. Para guardar información numérica en un archivo de texto es necesario convertirla a caracteres.
  - *Binarios*: guardan información binaria de cualquier tipo.

En los *archivos de texto plano* todos los datos se almacenan en forma de texto ASCII. Esto hace que podamos abrirlos, consultarlos y modificarlos con cualquier editor de texto, mientras que con los binarios no es posible. El *fichero plano* no es más que un fichero de texto estructurado en líneas (segmentos), separadas por los dos caracteres estándar para este tipo de ficheros, retorno de línea seguido del final de línea (CR|LF).

En los *archivos de texto plano*, y dependiendo del compilador y del sistema operativo empleado, pueden producirse ciertas *transformaciones automáticas de caracteres*. En particular, es frecuente que el carácter invisible de fin de línea (representado habitualmente como LF) se convierta en dos caracteres al escribirlo en un archivo (fin de línea – LF – más retorno de carro – CR –). También pueden ocurrir conversiones a la inversa, es decir, durante la lectura del archivo. Esto no sucede con los archivos binarios.

Todas las funciones de E/S sirven para ambos tipos de archivo, pero algunas pueden dar problemas con según qué tipos. Algunas funciones no funcionan bien con archivos de texto debido a las conversiones automáticas que antes mencionábamos. Desde cierto punto de vista, puede considerarse que un archivo de texto plano no es más que un archivo secuencial en el que cada registro es un carácter, por lo que tiene sentido que las funciones de acceso directo tengan problemas para tratar este tipo de archivos.

1.5 ALMACENAMIENTO EN SISTEMAS GESTORES DE BASE DE DATOS

Para un tratamiento "mecanizado" de la información es fundamental que los datos se organicen de forma que se facilite su gestión, es decir, que el acceso a los mismos sea rápido y eficaz. Convencionalmente, el trabajo en informática de gestión consistía en realizar una serie de programas en determinados lenguajes, para creación, actualización, consulta y listados de datos, que trabajan con información de entrada para obtener unos resultados o datos de salida.

El término *base de datos* aparece por primera vez al comienzo de los años 60, para expresar un conjunto de datos relacionados entre sí, que están estructurados de forma que puede accederse a ellos automáticamente e independientemente de los programas que gestionan esos datos. Esta independencia se refiere a la posibilidad de modificar la estructura de los datos sin necesidad de modificar los programas que los manipulan, evitando con ello los problemas de actualización de datos previamente existentes.

Toda base de datos está formada por uno o varios bloques de información llamados *tablas* (inicialmente denominados *ficheros* o *archivos*) que normalmente tendrán alguna característica en común. Una *tabla* o archivo de datos es un conjunto conexo de información del mismo tipo, por ejemplo en una base de datos de una biblioteca, una tabla estará constituida por la información relativa a todos los libros de la misma, otra tabla contendrá información sobre los lectores, etc.

Cada tabla está formada por *registros*. Un registro es la unidad elemental de información de la tabla o fichero (en un archivo clásico no automatizado un registro se corresponde con lo que suele llamarse ficha). En la tabla o fichero de libros, un registro estaría constituido por la información correspondiente a cada libro concreto, con su título, autor, área, editorial, etc. Cada registro está formado por uno o más elementos llamados *campos*. Un campo es cada una de las informaciones que interesa almacenar en cada registro, y es por tanto la unidad elemental de información del registro. En el ejemplo anterior, un campo sería el título del libro, otro campo su autor, etc.

Gracias a la aparición de los llamados programas de usuario (sistemas gestores de base de datos) es posible realizar la gestión de tablas de una base de datos, sin tener que realizar programas que procesen esos datos, facilitando todas las operaciones de creación, actualización, consulta y creación de informes con los datos recogidos. La forma de almacenar los ficheros, registros y campos es distinta, desde el punto de vista técnico, dependiendo de las características de la información que se quiere tratar. Sea cual sea la estructura de la base de datos, siempre se pueden incluir ficheros de índices secundarios para facilitar una respuesta eficaz a determinados tipos de acceso.

Históricamente el almacenamiento de la información ha experimentado una evolución muy rápida y siempre paralela a la disponibilidad y avances en el software y el hardware.

Comenzó almacenándose la información en ficheros planos, para ir avanzando hasta tecnologías muy desarrolladas actualmente como es el Data warehouse. No obstante, las bases de datos, a través de sus sistemas gestores ocupan la parcela más importante en el campo del almacenamiento y administración de la información. Actualmente, los sistemas de gestión de base de datos (abreviado mediante SGBD o DBMS) organizan y estructuran los datos de tal modo que puedan ser recuperados y manipulados por usuarios y programas de aplicación. Las estructuras de los datos y las técnicas de acceso proporcionadas por un DBMS particular se denominan su *modelo de datos.* El modelo de datos determina la "personalidad" de un DBMS, y las aplicaciones para las cuales está particularmente bien conformado.

Existe un tipo de lenguaje estándar normalizado para trabajar con bases de datos denominado SQL (*Structured Query Languaje*). SQL es un lenguaje de base de datos para bases de datos relacionales, y utiliza *el modelo de datos relacional.*

1.5.1 Modelos de datos primitivos: sistemas de gestión de archivos

Cuando la gestión de base de datos se popularizó durante los años setenta y ochenta, emergieron un puñado de modelos de datos populares. Cada uno de estos primeros modelos de datos tenía ventajas y desventajas que jugaron papeles importantes en el desarrollo del modelo de datos relacional. En muchos sentidos el modelo de datos relacional representó un intento de simplificar los modelos de datos anteriores.

Antes de la introducción de los sistemas de gestión de la base de datos, todos los datos permanentemente almacenados en un sistema informático, tales como la nómina y los registros de contabilidad, se almacenaban en archivos individuales. Un *sistema de gestión de archivos,* generalmente proporcionado por el fabricante del computador como parte del sistema operativo, llevaba la cuenta de los nombres y ubicaciones de los archivos. El sistema de gestión de archivos básicamente no tenía un modelo de datos; no sabía nada acerca de los contenidos internos de los archivos. Para el sistema de gestión de archivos, un archivo que contuviera un documento de procesamiento de textos y un archivo que contuviera datos de nóminas aparecían igual.

El conocimiento acerca del contenido de un archivo (qué datos contuviera y cómo estuvieran organizados) estaba incorporado a los programas de aplicación que utilizaban el archivo. Era típico que en aplicaciones de nóminas, cada uno de los programas (normalmente escritos en Cobol) que procesaban el archivo maestro de empleados contuviese una *descripción de archivo* (DA) que describía la composición de los datos en el archivo. Si la estructura de los datos cambiaba (por ejemplo, si un ítem adicional de datos fuera a ser almacenado por cada empleado), todos los programas que accedían al archivo tenían que ser modificados. Como el número de archivos y programas crecía con el tiempo, todo el esfuerzo de procesamiento de datos de un departamento se perdía en mantener aplicaciones existentes en lugar de desarrollar otras nuevas.

Los problemas de mantener grandes sistemas basados en archivos condujo, a finales de los sesenta, al desarrollo de los sistemas de gestión de base de datos. Se trataba que los datos estuviesen estructurados de forma que se pudiese acceder a ellos automáticamente e independientemente de los programas que gestionan esos datos. La idea detrás de estos sistemas era sencilla: tomar la definición de los contenidos de un archivo y la estructura de los programas individuales, y almacenarla, junto con los datos, en una base de datos. Utilizando la información de la base de datos, el DBMS (sistema gestor de la base de datos) que la controlaba podría tomar un papel mucho más activo en la gestión de los datos y en los cambios a la estructura de la base de datos.

1.5.2 Bases de datos jerárquicas

Básicamente podemos distinguir tres tipos de estructuras de bases de datos: jerárquica, en red y relacional. La *organización jerárquica*, que es la que primero se utilizó, se basa en el establecimiento de jerarquías o niveles entre los distintos campos de los registros, basándose en el criterio de que los campos de mayor jerarquía sean los más genéricos, y tiene una estructura arborescente, donde los nodos del mismo nivel corresponden a los campos y cada rama a un registro (figura 1-1). Para acceder a un campo que se encuentra en un determinado nivel, es preciso localizarlo partiendo del nivel superior y descendiendo por las ramas hasta llegar al mismo.

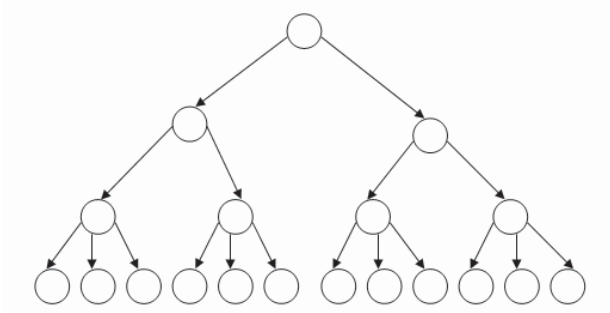


Figura 1.1

Tomemos como ejemplo la base de datos de una biblioteca y situemos como nivel superior el código de materia y el título de la materia, y supongamos que tomamos como campo maestro el título de la materia. En un segundo nivel de la jerarquía se incluyen las tablas de autores asociados a cada materia; en un tercer nivel tendríamos, por ejemplo, las tablas correspondientes a los autores y sus respectivos títulos de libros, y así hasta el último nivel en el que incluimos título, referencia, año de edición, número de ejemplares, o cualquier otro dato individual referente al libro buscado.

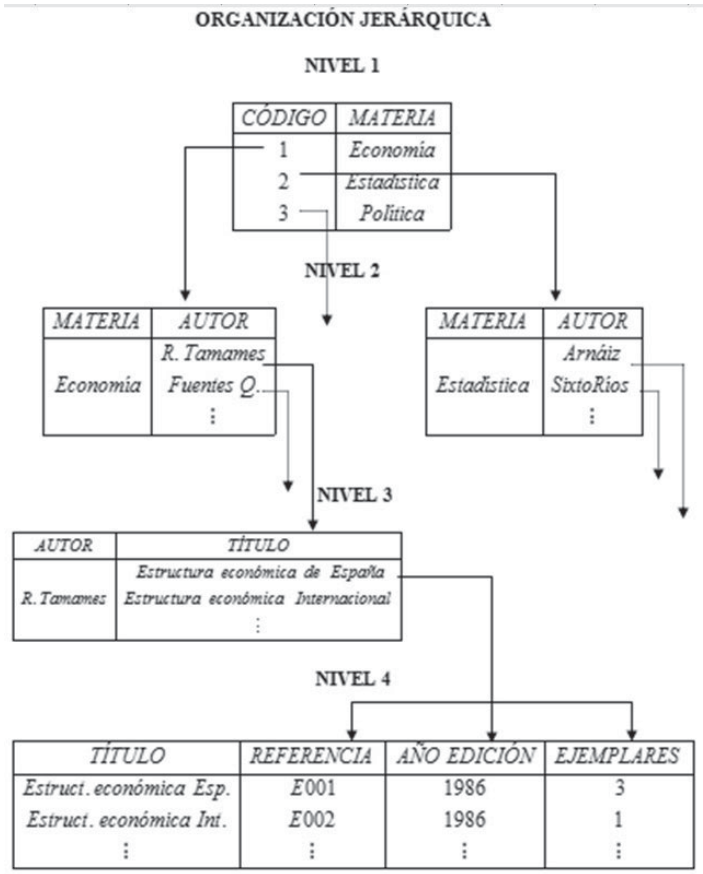


Figura 1.2

En la Figura 1.2 se muestra una organización jerarquizada. Para acceder a un dato del último nivel, por ejemplo año de edición de un determinado libro, hay que recorrer todos los niveles desde el más alto (materia, etc). Esta forma de organización puede hacer lenta la obtención de determinadas informaciones, ya que para acceder a un nodo (campo) hay que recorrer toda la rama, partiendo de la raíz (nodo de mayor jerarquía), es decir, todos los campos precedentes en el registro. No obstante, existen estructuras arborescentes más sofisticadas que incluyen índices y que permiten acelerar el resultado de las consultas.

Uno de los sistemas de gestión de bases de datos jerárquicas más populares fue el *Information Management System* (IMS) de IBM, introducido en 1968. Este sistema presentaba las siguientes ventajas:

- *Estructura simple*: la organización de una base de datos IMS es fácil de entender. La jerarquía de la base de datos se asemejaba al diagrama de organización de una empresa o a un árbol familiar.

- *Organización padre/hijo*: una base de datos IMS era excelente para representar relaciones padre/hijo, tales como "A es pieza de B", "A es propiedad de B", "el título A es del autor B", etc.

- *Rendimiento*: IMS almacenaba las relaciones padre/hijo como punteros físicos de un registro de datos a otro, de modo que el movimiento a través de la base de datos era rápido. Puesto que la estructura era sencilla, IMS podía colocar los registros padre e hijo cercanos unos a otros en el disco, minimizando la cuota entrada/salida del disco.

1.5.3 Bases de datos en red

Para resolver el problema de lentitud de la organización jerárquica se utiliza la *organización en red* que corresponde a una estructura de grafo, donde existe más de una conexión entre los nodos de diversos niveles, de forma que puedan recorrerse por distintos caminos sin necesidad de acudir cada vez a la raíz, con lo cual la búsqueda es más flexible, desapareciendo el concepto de "jerarquía" entre campos, pues un campo puede ser descendiente de su antecesor por un camino de la red y ascendente por otro.

Si se crean conexiones entre nodos de igual nivel, el acceso a campos de determinado nivel se logrará más rápidamente, así por ejemplo, en el caso de la base de datos de la biblioteca, se podrían listar los títulos de los distintos libros, a partir de un título dado sin acceder cada vez a los autores.

El modelo de datos en red extiende el modelo jerárquico permitiendo que un registro participe en múltiples relaciones padre/hijo. Estas relaciones se conocen como *conjuntos* en el modelo de red.

El inconveniente esencial de esta estructura es la necesidad de utilizar mucha más cantidad de memoria, al tener que almacenar en cada nodo las posiciones de los campos siguientes, mediante apuntadores.

En la figura 1.3 se representa un grafo similar al del modelo jerárquico, pero adecuado al modelo en red. En este grafo desaparece el concepto de jerarquía entre campos, pues un campo puede ser descendiente de su antecesor por un camino de la red y ascendente por otro.

En 1971 la Conferencia sobre Lenguajes de Sistemas de Datos publicó un estándar oficial para bases de datos en red, al que se conoció con el nombre de CODASYL. IBM nunca desarrolló un DBMS en red, eligiendo en su lugar extender el IMS a lo largo de los años. Pero durante los años setenta, compañías de software independientes se apresuraron a adoptar el modelo en red, creando productos tales como el IDMS de Cullinet, el Total de Cincom y el DBMS Adabas que se hizo muy popular.

Para un programador, acceder a una base de datos en red era muy similar a acceder a una base de datos jerárquica. Un programa de aplicación podía:

- Hallar un registro padre específico mediante clave (como por ejemplo un número de cliente en un procesamiento de pedidos).
- Descender al primer hijo en un conjunto particular (el primer pedido remitido por este cliente).
- Moverse lateralmente de un hijo al siguiente dentro del conjunto (la orden siguiente remitida por el mismo cliente).
- Ascender desde un hijo a su padre en otro conjunto (el vendedor que aceptó el pedido).

Una vez más el programador tenía que recorrer la base de datos registro a registro, especificando esta vez qué relación recorrer además de indicar la dirección.

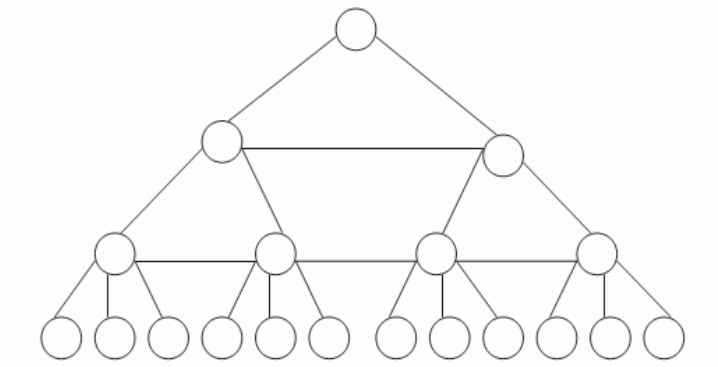


Figura 1.3

Las bases de datos en red tenían varias ventajas entre las que se pueden destacar las siguientes:

- *Flexibilidad*: las múltiples relaciones padre/hijo permitían a una base de datos en red representar datos que no tuvieran una estructura jerárquica sencilla.
- *Normalización*: el estándar CODASYL reforzó la popularidad del modelo de red, y los vendedores de minicomputadores tales como Digital Equipment Corporation y Data General implementaron bases de datos en red.
- *Rendimiento*: a pesar de su superior complejidad, las bases de datos en red reforzaron el rendimiento aproximándolo al de las bases de datos jerárquicas. Los conjuntos se representaron mediante punteros a registros de datos físicos, y en

algunos sistemas, el administrador de la base de datos podía especificar la agrupación de datos basada en una relación de conjunto.

Las bases de datos en red tenían sus desventajas también. Igual que las bases de datos jerárquicas resultaban muy rígidas. Las relaciones de conjunto y la estructura de los registros tenían que ser especificadas de antemano. Modificar la estructura de la base de datos requería generalmente la reconstrucción de la base de datos completa. Tanto las bases jerárquicas como las bases en red eran herramientas para programadores.

1.5.4 Bases de datos relacionales

Quizás, el problema fundamental que suele plantearse al realizar una base de datos real, formada por varias tablas, es la repetición de datos, es decir, campos repetidos en diferentes tablas (redundancia), lo cual va a dificultar su gestión, es decir, la actualización, inserción, modificación, eliminación, consulta, etc.

Para resolver estos problemas es necesario que exista integración entre las distintas tablas y que esté controlada la repetición de datos. Así surgen los llamados Sistemas de Gestión de Bases de Datos relacionales que, en el caso de los microordenadores, están concebidos como un conjunto de programas de propósito general que permiten controlar el acceso y la utilización de las bases de datos de forma que satisfagan las necesidades del usuario (programas de usuario) y que actúen con independencia de los datos, y con ellos las llamadas *bases de datos relacionales* que pueden resolver, mejor que otras organizaciones, las dificultades de redundancia y no integración de los datos. En este tipo de bases de datos se suprimen las jerarquías entre campos, pudiéndose utilizar cualquiera de ellos como clave de acceso.

La teoría relacional se basa en el concepto matemático de relación. Se debe a E.F. Codd, quien ha desarrollado una sólida fundamentación teórica. Aunque dicha teoría requiere para su completa implantación que el acceso a la memoria sea por contenido y no por dirección, como ocurre en los actuales ordenadores, puede adecuarse y de hecho se está implantando y desarrollando en la mayoría de los equipos. Las principales ventajas de la utilización de bases de datos relacionados son:

- Actúan sobre las tablas en su conjunto, en lugar de hacerlo sobre los registros como ocurre en otros sistemas.
- Se pueden realizar consultas complejas que utilizan varias tablas de forma simple.
- Son fáciles de utilizar (la organización física de los datos es independiente de su tratamiento lógico).

La organización relacional se caracteriza porque las tablas de la base de datos tienen estructura de matriz o tabla bidimensional, donde las filas son los registros y las columnas los campos (Figura 1.4).

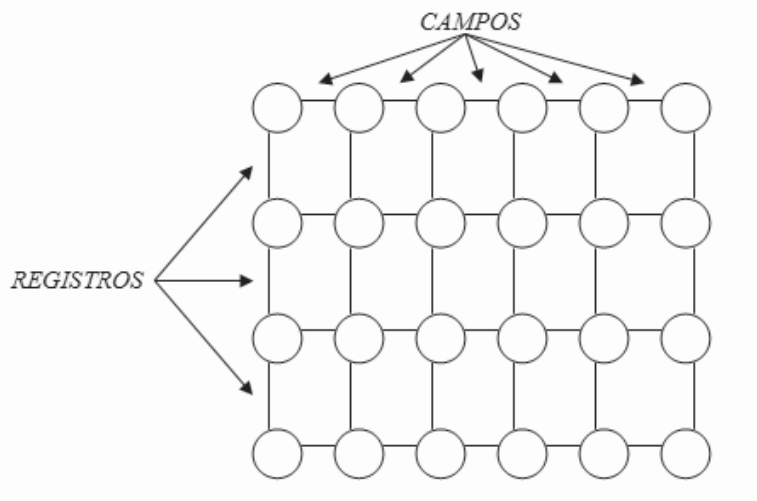


Figura 1.4

Las tablas son tratadas como conjuntos matemáticos, obtenidas como subconjuntos del producto cartesiano de los rangos de posibles valores de los distintos campos que las forman. Cada tabla dispone de una cabecera que es un registro especial donde figuran los nombres de los campos y una serie de registros (filas) donde se describen los objetos.

El esquema que se muestra en la Figura 1.5 representa la tabla del ejemplo de la base de datos de la biblioteca adaptado al modelo relacional. La tabla tiene todos sus campos relacionados, de forma que es posible tomar, por ejemplo, como entrada, la materia y averiguar el número de ejemplares, o bien obtener todos los libros editados en 1986 sin más que consultar el campo correspondiente a "año de edición".

CAMPOS

REGISTROS

CÓD.	MATERIA	AUTOR	TÍTULO	REF.	AÑO	Nº EJ.
1	Economía	R.Tamames	Estr.Ec.España	E001	1986	3
1	Economía	R.Tamames	Estr.Ec.Intern.	E002	1986	1
3	Política	R.Tamames	¿Quo vadis Esp.?	P039	1975	1
9	Novela	A.Camus	El extranjero	N069	1940	2
9	Novela	R.Tamames	Historia de Elio	N070	1978	1
1	Economía	FuentesQ.	Hacienda Públ.	E003	1974	2
1	Economía	Lipsey	Teoría Econom.	E004	1970	2
2	Estadística	Arnáiz	Estad.descript.	S001	1962	1
2	Estadística	Sixto Ríos	Métodos Estad.	S002	1970	1

Figura 1.5

Para que la estructura de las tablas cumpla las leyes de la teoría relacional deben satisfacerse las siguientes condiciones:

- Todos los registros de la tabla deben tener el mismo número de campos, aunque alguno de ellos esté vacío, deben ser registros de longitud fija.

- Cada campo tiene un nombre o etiqueta que hay que definir previamente a su utilización. No obstante, una vez creado el fichero se podrá ampliar o disminuir el número de campos, mediante el SGBD.

- La base de datos estará formada por muchas tablas, una por cada tipo de registro. En el ejemplo de la biblioteca podríamos definir otras tablas. Por ejemplo con los nombres de campo AUTOR, NACIONALIDAD, PROFESIÓN.

- Dentro de una tabla cada nombre de campo debe ser distinto, por ejemplo en la de materias podría haber Autor 1, Autor 2: pero no puede haber dos campos con el nombre Autor, pues al referirnos con el gestor al nombre de campo AUTOR, no sabría cual utilizar.

- Los registros de una misma tabla tienen que diferenciarse, al menos, en el contenido de alguno de sus campos, no puede haber dos registros "idénticos".

- Los registros de una tabla pueden estar dispuestos en cualquier orden.

- El contenido de cada campo está delimitado por un rango de valores posibles. En el ejemplo del campo AÑO DE EDICIÓN no puede ponerse MIL o M, ni cualquier otro carácter alfabético, e incluso ningún año mayor que 2000, por no estar dentro del rango definido de los años posibles.

- Permite la creación de nuevas tablas a partir de las ya existentes, relacionando campos de distintas tablas anteriores. Esta condición es la esencia de las bases de datos relacionales, formando lo que se llama un fichero "virtual" (temporalmente en memoria).

El primer punto a abordar en la generación de las tablas es establecer una tabla "vacía". Esta tarea se realiza, bien mediante un asistente, o bien mediante el comando CREATE TABLE de SQL (lenguaje típico para el trabajo con bases de datos relacionales). Posteriormente, se pueden almacenar registros en la tabla, bien mediante asistentes proporcionados por las aplicaciones de base de datos o bien ejecutando la sentencia INSERT de SQL. Veremos cómo se utilizan los asistentes del SGBD de Microsoft Access en próximos capítulos, limitándonos aquí a citar su existencia y utilidad.

La sentencia CREATE TABLE crea un "objeto" (la tabla) dentro del sistema. Hay otros objetos importantes en el sistema, como los índices, que se establecen ejecutando la

sentencia CREATE INDEX de SQL. La mayoría de los sistemas gestores de bases de datos permiten crear tablas e índices. Sin embargo, cada sistema contiene también otros objetos.

Existen diferencias en las versiones completas de los distintos SGBD, porque cada una tiene cláusulas diferentes que referencian objetos específicos del sistema. Afortunadamente, los usuarios y los programadores de aplicaciones no necesitan, más que en raras ocasiones, estar familiarizados con estos objetos, ya que existe una estandarización que hace familiar cualquier sistema gestor actual a cualquier usuario.

Los objetivos más importantes de la creación de tablas mediante asistentes o mediante la sentencia CREATE TABLE de SQL podrían enumerarse como sigue:

- Establecer nuevas tablas en la base de datos y asignarles un nombre.
- Asignar nombre a todas las columnas de cada tabla y definir sus *tipos de datos*.
- Especificar la secuencia u orden de columnas por defecto.
- Especificar qué columnas no pueden aceptar *valores nulos* (valores que faltan, son desconocidos o no son aplicables). El valor nulo no debe confundirse con el valor cero, pues cero no es un valor nulo.
- Especificar la *clave primaria* (columna o combinación de columnas cuyos valores identifican unívocamente cada fila en la tabla, es decir, la clave primaria tiene un valor único, diferente para cada fila de una tabla, de modo que no hay dos filas de una tabla con clave primaria que sean duplicados exactos la una de la otra).
- Especificar las *claves secundarias, externas o foráneas* (una columna de una tabla cuyo valor coincide con la clave primaria de alguna otra tabla de la base de datos se denomina una clave secundaria, externa o foránea, de modo que una tabla puede contener más de una clave secundaria si está relacionada con más de una tabla adicional de la base de datos). Definir las *relaciones* entre los campos de las tablas de la base de datos es una tarea fundamental.
- Especificar las *restricciones de integridad* para preservar la consistencia y corrección de los datos almacenados. Se especificarán los datos requeridos en los campos y registros y chequeos de validez para los tipos de datos a introducir en los campos.
- Especificar los *índices* para los campos o grupos de campos de las tablas de la base de datos. Los índices son estructuras internas que el sistema gestor de la base de datos puede utilizar para encontrar uno o más registros de una tabla de forma rápida.

1.6 OTROS TIPOS DE ALMACENAMIENTO

Existen otros tipos de almacenamiento de información estructurada, entre los que destacan el formato XML y los Servicios de directorio LDAP.

1.6.1 Ficheros XML

El lenguaje XML es un lenguaje de marcas generalizado (*Extensible Markup Language*) utilizado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Este lenguaje ha ganado mucha popularidad en los últimos años debido a ser un estándar abierto y libre y ha sido creado por el Consorcio World Wide Web, W3C (creadores de www), en colaboración con un equipo que incluye representantes de las principales compañías productoras de software. XML fue propuesto en 1996, y la primera especificación apareció en 1998. Desde entonces su uso ha tenido un crecimiento acelerado y se ha convertido prácticamente en un estándar.

Antes de ser lanzado el XML, ya existían otros lenguajes de marcas, como por ejemplo el HTML, basados en el lenguaje generalizado de marcas (SGML). El problema con el SGML es que por ser muy flexible y muy general, se torna difícil el análisis sintáctico de un documento y la especificación de la estructura (que en el caso de XML se incluye en otro documento llamado DTD). XML es más exigente que SGML en la sintaxis, lo que hace más fácil la construcción de librerías para procesarlo. Comparado con otros sistemas usados para crear documentos, XML tiene la ventaja de poder ser más exigente en cuanto a la organización del documento, lo que produce documentos mejor estructurados.

Los ficheros XML son ficheros de texto, que en principio está en código Unicode, pero se pueden usar otros alfabetos como el latin-1. Existen cinco caracteres especiales en XML: los símbolos menor que, <, mayor que, >, las comillas dobles", el apóstrofe ' y el carácter &. Los símbolos mayor que y menor que se usan para delimitar las marcas que dan la estructura al documento.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

A continuación se muestra un ejemplo de fichero XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
     <Mensaje>
          <Remitente>
               <Nombre>Nombre del remitente</Nombre>
               <Mail> Correo del remitente </Mail>
          </Remitente>
          <Destinatario>
               <Nombre>Nombre del destinatario</Nombre>
               <Mail>Correo del destinatario</Mail>
          </Destinatario>
          <Texto>
               <Asunto>
                       Documento sencillo
               </Asunto>
               <Parrafo>
                    Documento de estructura muy sencilla,
                    no contiene atributos ni entidades...
               </Parrafo>
          </Texto>
     </Mensaje>
</Edit_Mensaje>
```

En cuanto a las *partes de un documento XML*, típicamente está formado por el prólogo y por el cuerpo del documento así como texto de etiquetas que contiene una gran variedad de efectos positivos o negativos en la referencia opcional a la que se refiere el documento. Hay que tener mucho cuidado con esa parte de la gramática léxica para que se componga de manera uniforme.

En cuanto al *prólogo de un documento XML* hay que destacar que no es obligatorio, pero los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas. El prólogo de un documento XML contiene:

- Una declaración XML. Es la sentencia que declara al documento como un documento XML.

- Una declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.

- Uno o más comentarios e instrucciones de procesamiento.

A diferencia del prólogo, el *cuerpo* no es opcional en un documento XML. El cuerpo debe contener un y solo un elemento raíz, característica indispensable también para que el documento esté bien formado. Sin embargo es necesaria la adquisición de datos para su buen funcionamiento.

Los *elementos XML* pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

Los elementos pueden tener *atributos*, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas. Por ejemplo, un elemento "estudiante" puede tener un atributo "Mario" y un atributo "tipo", con valores "come croquetas" y "taleno" respectivamente.

- Las *entidades predefinidas* son entidades para representar caracteres especiales para que, de esta forma, no sean interpretados como marcado en el procesador XML.

- Las *secciones CDATA* son construcciones en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. Permiten que caracteres especiales no rompan la estructura del documento.

- Los *comentarios* se usan a modo informativo para el programador y son ignorados por el procesador.

Los documentos XML denominados como "bien formados" (del inglés *well formed*) son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, analizarse correctamente por cualquier analizador sintáctico (*parser*) que cumpla con la norma. Para un formato correcto se tendrán en cuenta las siguientes características:

- Los documentos han de seguir una estructura estrictamente jerárquica en lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.

- Los documentos XML solo permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial.

- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.

- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.

- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos "entendibles" por las personas.

Las ventajas más importantes del lenguaje XML son las siguientes:

- *Es extensible*. Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

- *El analizador es un componente estándar*: no es necesario crear un analizador específico para cada versión del lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de aplicaciones.

- Si un tercero decide usar un documento creado en XML, *es sencillo entender su estructura y procesarla*. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y base de datos MS-SQL Server.

- *Transforma datos en información*: se le añade un significado concreto y lo asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

1.6.2 Servicios de directorio LDAP

LDAP (*Protocolo compacto de acceso a directorios*) es un protocolo estándar que permite administrar directorios, esto es, acceder a bases de información de usuarios de una red mediante protocolos TCP/IP. El objetivo del protocolo LDAP, desarrollado en 1993 en la Universidad de Michigan, fue reemplazar al protocolo DAP (utilizado para acceder a los servicios de directorio X.500 por OSI) integrándolo al TCP/IP. Desde 1995, DAP se convirtió en LDAP independiente, con lo cual se dejó de utilizar solo para acceder a los directorios tipo X500. LDAP es una versión más simple del protocolo DAP, de allí deriva su nombre *Protocolo compacto de acceso a directorios*.

El protocolo LDAP define el método para acceder a datos en el servidor a nivel cliente pero no la manera en la que se almacena la información. LDAP le brinda al usuario métodos que le permiten conectarse, desconectarse, buscar información, comparar información, insertar entradas, cambiar entradas y eliminar entradas. Así mismo, el protocolo LDAP (en versión 3) ofrece mecanismos de cifrado (SSL, etc.) y autenticación para permitir el acceso seguro a la información almacenada en la base. LDAP presenta la información bajo la forma de una estructura jerárquica de árbol denominada DIT (Árbol de información de directorio), en la cual la información es representada por bifurcaciones. Cada entrada en el directorio LDAP corresponde a un objeto abstracto o real (por ejemplo, una persona, un objeto material, parámetros, etc.). Cada entrada está conformada por un conjunto de pares clave/valor denominados atributos que permiten caracterizar el objeto que la entrada define.

LDAP brinda un conjunto de funciones (procedimientos) para llevar a cabo solicitudes en los datos para buscar, cambiar y eliminar entradas en los directorios. A continuación encontrará una lista de las principales operaciones que puede realizar LDAP:

Funcionamiento	Descripción
Abandon (Abandonar)	Cancela la operación previa enviada al servidor.
Add (Agregar)	Agrega una entrada en el directorio.
Bind (Enlazar)	Inicia una nueva sesión en el servidor LDAP.
Compare (Comparar)	Compara las entradas en un directorio según los criterios.
Delete (Eliminar)	Elimina una entrada de un directorio.
Extended (Extendido)	Realiza operaciones extendidas.
Rename (Cambiar nombre)	Cambia el nombre de una entrada.
Search (Buscar)	Busca entradas en un directorio.
Unbind (Desenlazar)	Finaliza una sesión en el servidor LDAP.

LDAP brinda un formato de intercambio de datos (LDIF, *Formato de intercambio de datos de LDAP*) que permite importar y exportar datos desde un directorio mediante un archivo de texto simple. La mayoría de los servidores LDAP admiten este formato, lo cual permite una gran interoperabilidad entre ellos.

Capítulo

2

SISTEMAS GESTORES DE BASES DE DATOS

2.1 DEFINICIÓN Y EVOLUCIÓN DE LOS SISTEMAS GESTORES DE BASE DE DATOS

Los sistemas de gestión de bases de datos o SGBD (en inglés *Database Management System*, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. En concreto, definimos un *Sistema Gestor de Bases de Datos* como una colección de datos relacionados entre sí, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina *base de datos*.

El propósito general de los sistemas de gestión de bases de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

Antes de aparecer en la década de los setenta los sistemas gestores de bases de datos, la información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos que iban soportados sobre un sistema operativo. Estos consistían en un conjunto de programas que definían y trabajaban sus propios datos. Los datos se almacenan en archivos y los programas manejan esos archivos para obtener la información. Si la estructura de los datos de los archivos cambia, todos los programas que los manejan se deben modificar; por ejemplo, un programa trabaja con un archivo de datos de alumnos, con una estructura o registro ya definido; si se incorporan elementos o campos a la estructura del archivo, los programas que utilizan ese archivo se tienen que modificar para tratar esos nuevos elementos. En estos sistemas de gestión de archivos, la definición de los datos se encuentra codificada dentro de los programas de aplicación en lugar de almacenarse de forma independiente, y además el control del acceso y la manipulación de los datos vienen impuestos por los programas de aplicación. Esto supone un gran inconveniente a la hora de tratar grandes volúmenes de información.

Surge así la idea de separar los datos contenidos en los archivos de los programas que los manipulan, es decir, que se pueda modificar la estructura de los datos de los archivos sin que por ello se tengan que modificar los programas con los que trabajan. Se trata de

estructurar y organizar los datos de forma que se pueda acceder a ellos con independencia de los programas que los gestionan.

Los sistemas de gestión de archivos citados anteriormente presentaban muchos inconvenientes entre los que destacan los siguientes:

- *Redundancia e inconsistencia de los datos*: se produce porque los archivos son creados por distintos programas y van cambiando a lo largo del tiempo, es decir, pueden tener distintos formatos y los datos pueden estar duplicados en varios sitios. Por ejemplo, el teléfono de un alumno puede aparecer en más de un archivo. La redundancia aumenta los costes de almacenamiento y acceso, y trae consigo la inconsistencia de los datos: las copias de los mismos datos no coinciden por aparecer en varios archivos.

- *Dependencia de los datos física-lógica*: o lo que es lo mismo, la estructura física de los datos (definición de archivos y registros) se encuentra codificada en los programas de aplicación. Cualquier cambio en esa estructura implica al programador identificar, modificar y probar todos los programas que manipulan esos archivos.

- *Dificultad para tener acceso a los datos*: proliferación de programas, es decir, cada vez que se necesite una consulta que no fue prevista en el inicio implica la necesidad de codificar el programa de aplicación necesario. Lo que se trata de probar es que los entornos convencionales de procesamiento de archivos no permiten recuperar los datos necesarios de una forma conveniente y eficiente.

- *Separación y aislamiento de los datos*: es decir, al estar repartidos en varios archivos, y tener diferentes formatos, es difícil escribir nuevos programas que aseguren la manipulación de los datos correctos. Antes se deberían sincronizar todos los archivos para que los datos coincidiesen.

- *Dificultad para el acceso concurrente*: pues en un sistema de gestión de archivos es complicado que los usuarios actualicen los datos simultáneamente. Las actualizaciones concurrentes pueden dar por resultado datos inconsistentes, ya que se puede acceder a los datos por medio de diversos programas de aplicación.

- *Dependencia de la estructura del archivo con el lenguaje de programación*: pues la estructura se define dentro de los programas. Esto implica que los formatos de los archivos sean incompatibles. La incompatibilidad entre archivos generados por distintos lenguajes hace que los datos sean difíciles de procesar.

- *Problemas en la seguridad de los datos*: Resulta difícil implantar restricciones de seguridad pues las aplicaciones se van añadiendo al sistema según se van necesitando.

- *Problemas de integridad de datos*: es decir, los valores almacenados en los archivos deben cumplir con restricciones de consistencia. Por ejemplo, no se

puede insertar una nota de un alumno en una asignatura si previamente esa asignatura no está creada. Otro ejemplo, las unidades en almacén de un producto determinado no deben ser inferiores a una cantidad. Esto implica añadir gran número de líneas de código en los programas. El problema se complica cuando existen restricciones que implican varios datos en distintos archivos.

2.2 FUNCIONES DEL SISTEMA GESTOR DE BASE DE DATOS

Los sistemas gestores de base de datos deben tratar de mitigar los inconvenientes de los sistemas de gestión de archivos. En esta línea, distinguiremos entre los distintos *objetivos que deben cumplir los SGBD,* las *funciones o servicios que deben prestar los SGBD* y sus ventajas e inconvenientes.

2.2.1 Objetivos que deben cumplir los SGBD

Entre los objetivos más importantes a cumplir por un SGBD tenemos los siguientes:

- *Abstracción de la información*. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- *Independencia*. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- *Consistencia*. En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.

- *Seguridad*. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- *Manejo de transacciones*. Una transacción es un programa que se ejecuta como una sola operación. Esto quiere decir que luego de una ejecución en la que se produce una falla es el mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- *Tiempo de respuesta*. Lógicamente, es deseable minimizar el tiempo que el SGBD demora en proporcionar la información solicitada y en almacenar los cambios realizados.

El objetivo primordial de un gestor es proporcionar eficiencia y seguridad a la hora de extraer o almacenar información en las bases de datos. Los sistemas gestores de bases de datos están diseñados para gestionar grandes bloques de información, que implica, tanto la definición de estructuras para el almacenamiento, como de mecanismos para la gestión de la información. Una base de datos es un gran almacén de datos que se define una sola vez, los datos pueden ser accedidos de forma simultánea por varios usuarios, están relacionados y existe un número mínimo de duplicidad. Además, en las bases de datos se almacenarán las descripciones de esos datos (metadatos) en el diccionario de datos.

2.2.2 Funciones o servicios de los SGBD

El SGBD es una aplicación que permite a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma. *El SGBD debe prestar los siguientes servicios*:

- *Creación y definición de la base de datos*: especificación de la estructura, el tipo de los datos, las restricciones y relaciones entre ellos mediante lenguajes de definición de datos. Toda esta información se almacena en el diccionario de datos, el SGBD proporcionará mecanismos para la gestión del diccionario de datos.

- *Manipulación de los datos*: realizando consultas, inserciones y actualizaciones de los mismos utilizando lenguajes de manipulación de datos.

- *Acceso controlado a los datos de la base de datos*: mediante mecanismos de seguridad de acceso a los usuarios.

- *Mantener la integridad y consistencia de los datos*: utilizando mecanismos para evitar que los datos sean perjudicados por cambios no autorizados.

- *Acceso compartido a la base de datos*: controlando la interacción entre usuarios concurrentes.

- *Mecanismos de respaldo y recuperación*: para restablecer la información en caso de fallos en el sistema.

- *Mantenimiento de esquemas*: el esquema de la base de datos es la descripción de la estructura de la información almacenada en ella. Por ejemplo, para un sistema basado en tablas, el esquema puede consistir en una lista de tablas en uso, los campos que contienen, el tipo de datos de cada campo, descripciones en lenguaje natural del propósito de cada tabla y cada campo, y restricciones sobre los valores admisibles en cada campo. Así como los usuarios necesitan acceder, agregar y modificar datos, también necesitan acceder, agregar y modificar el esquema de datos. Por ejemplo, un usuario que se acerca por primera vez a una base de datos querrá saber antes que nada qué información contiene ésta, un programador puede escribir programas que definan y creen nuevos tipos de entidades, o eliminen algunos preexistentes; el administrador de la base de datos necesita controlar qué usuarios tienen acceso a qué información, formulando reglas de seguridad que se hacen parte del esquema.

- *Optimización de transacciones*: una de las áreas principales de aplicación de los sistemas gestores de bases de datos es lo que se llama procesamiento de transacciones. Una transacción es un programa de aplicación, generalmente de duración breve, que accede y actualiza una parte también generalmente pequeña de la base de datos. Típicos ejemplos son un depósito o extracción de una cuenta bancaria, o una reservación en un vuelo, o una verificación de una tarjeta de crédito.

El manejo de transacciones consiste en controlar múltiples transacciones ejecutando en paralelo sobre una misma base de datos corriendo en un sistema que puede fallar. Los objetivos del gestor de transacciones del SGBD consisten en evitar que las transacciones interfieran unas con otras al ejecutar en paralelo, y garantizar que la base de datos no sea dañada de forma irreparable por caídas, ya sea del sistema en sí o de alguna de las transacciones. El primero de los objetivos da lugar a lo que se llama *control de paralelismo*; el segundo, a *técnicas de recuperación*.

2.2.3 Ventajas e inconvenientes de los SGBD

Entre las ventajas de los sistemas gestores de bases de datos podemos destacar las siguientes:

- Proveen facilidades para la manipulación de grandes volúmenes de datos.

- Simplifican la programación de equipos de consistencia.

- Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores, etc.

- Organizan los datos con un impacto mínimo en el código de los programas.

- Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.

- Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

- Entre los inconvenientes de los sistemas gestores de bases de datos podemos destacar los siguientes:

- Típicamente, es necesario disponer de una o más personas que administren la base de datos, de la misma forma en que suele ser necesario en instalaciones de cierto porte disponer de una o más personas que administren los sistemas operativos. Esto puede llegar a incrementar los costos de operación en una empresa. Sin embargo hay que balancear este aspecto con la calidad y confiabilidad del sistema que se obtiene.

- Si se tienen muy pocos datos que son usados por un único usuario por vez y no hay que realizar consultas complejas sobre los datos, entonces es posible que sea mejor usar una hoja de cálculo.

- La complejidad es un inconveniente importante. A menudo, el software es muy complejo y las personas que vayan a usarlo deben tener conocimiento de las funcionalidades del mismo para poder aprovecharlo al máximo.

- El tamaño es otro inconveniente típico. La complejidad y la gran cantidad de funciones que tienen que realizar hacen que un sistema gestor de base de datos sea un software de gran tamaño, que requiere de gran cantidad de memoria para poder ejecutarse y trabajar.

- Los requisitos de hardware para correr un SGBD por lo general son relativamente altos, por lo que estos equipos pueden llegar a costar gran cantidad de dinero.

2.3 ARQUITECTURA DEL SISTEMA GESTOR DE BASE DE DATOS

Hay tres características importantes inherentes a los sistemas gestores de bases de datos: la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos. Esta idea llevó a la implantación de la arquitectura de tres capas o niveles para los sistemas gestores de base de datos.

2.3.1 Arquitectura de tres capas o niveles

En 1975, el comité ANSI-SPARC (*American National Standard Institute - Standards Planning and Requirements Committee*) propuso una arquitectura de tres niveles o capas para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características: el nivel interno o físico o de máquina, el nivel externo o de usuario, y el nivel conceptual. Así mismo describió las interacciones entre estos tres niveles y todos los elementos que conforman cada uno de ellos.

Nivel interno, físico o de máquina: el más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. Describe la estructura física de la base de datos mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que los componen, etcétera.

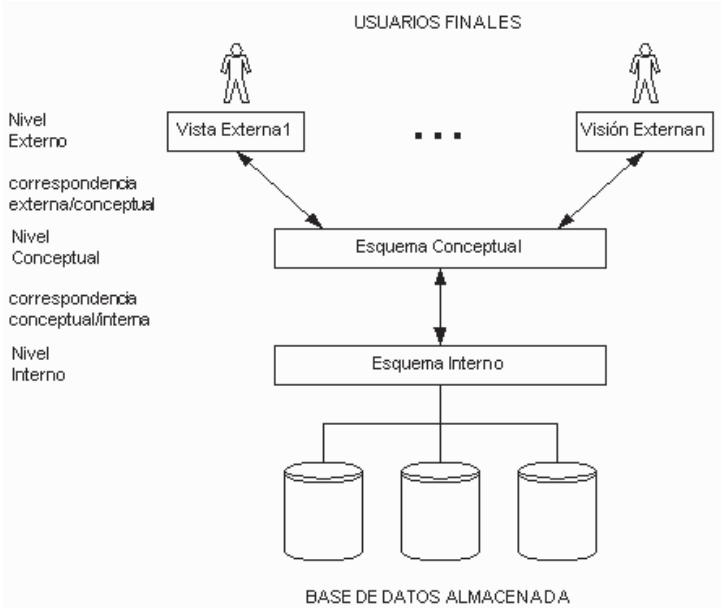
Nivel externo o de visión: es el más cercano a los usuarios. En este nivel se describen varios esquemas externos o vistas de usuarios. Cada esquema describe la parte de la base de datos que interesa a un grupo de usuarios. En este nivel se representa la visión individual de un usuario o de un grupo de usuarios.

Nivel conceptual: describe la estructura de toda la base de datos para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos, relaciones, operaciones de los usuarios y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento. Representa la información contenida en la base de datos.

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. La mayoría de los SGBD no distinguen del todo los tres niveles. Algunos incluyen detalles del nivel físico en el esquema conceptual. En casi todos los SGBD que se manejan vistas de usuario, los esquemas externos se especifican con el mismo modelo de datos que describe la información a nivel conceptual, aunque en algunos se pueden utilizar diferentes modelos de datos en los niveles conceptual y externo.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo como puede ser un disco. En un SGBD basado en la arquitectura de tres niveles, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo. Por lo tanto, el SGBD debe transformar cualquier petición expresada en términos de un esquema externo a una petición expresada en términos del esquema conceptual, y luego, a una petición en el esquema interno, que se procesará sobre la base de datos almacenada. Si la petición es de una obtención (consulta) de datos, será preciso modificar el formato de la información extraída de la base de datos almacenada, para que coincida con la vista externa del usuario. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación. Estas correspondencias pueden requerir bastante tiempo, por lo que algunos SGBD no cuentan con vistas externas.

El esquema siguiente ilustra la arquitectura de tres capas o niveles de un sistema gestor de base de datos.



Esta arquitectura describe los datos a tres niveles de abstracción. En realidad los únicos datos que existen están a nivel físico almacenados en discos u otros dispositivos. Los SGBD basados en esta arquitectura permiten que cada grupo de usuarios haga referencia a su propio esquema externo. El SGBD debe transformar cualquier petición de usuario (esquema externo) a una petición expresada en términos de esquema conceptual, para finalmente ser una petición expresada en el esquema interno que se procesará sobre la base de datos almacenada. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación, el SGBD es capaz de interpretar una solicitud de datos y realiza los siguientes pasos:

- El usuario solicita unos datos y crea una consulta.
- El SGBD verifica y acepta el esquema externo para ese usuario.
- Transforma la solicitud al esquema conceptual.
- Verifica y acepta el esquema conceptual.
- Transforma la solicitud al esquema físico o interno.
- Selecciona la o las tablas implicadas en la consulta y ejecuta la consulta.
- Transforma del esquema interno al conceptual, y del conceptual al externo.
- Finalmente, el usuario ve los datos solicitados.

Para una base de datos específica solo hay un esquema interno y uno conceptual, pero puede haber varios esquemas externos definidos para uno o para varios usuarios.

2.3.2 El concepto de independencia de datos

La arquitectura de tres niveles es útil para explicar el *concepto de independencia de datos* que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Se definen dos tipos de independencia:

- *Independencia lógica*: la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se podrá modificar el esquema conceptual para ampliar la BD o para reducirla, por ejemplo, si se elimina una entidad, los esquemas externos que no se refieran a ella no se verán afectados.

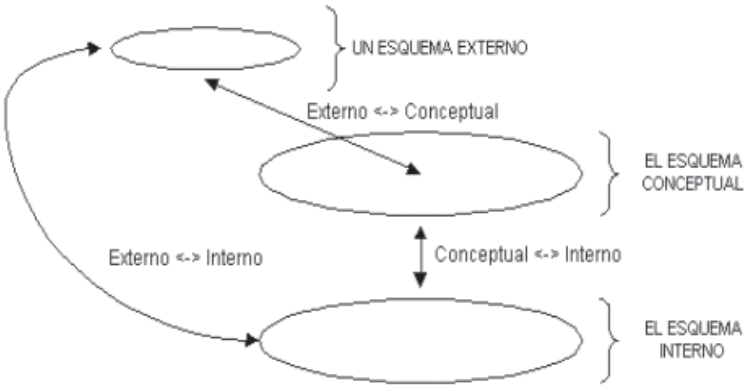
- *Independencia física*: la capacidad de modificar el esquema interno sin tener que alterar ni el esquema conceptual, ni los externos. Por ejemplo, se pueden reorganizar los archivos físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización, o se pueden añadir nuevos archivos de datos porque los que había se han llenado. La independencia física es más fácil de conseguir que la lógica, pues se refiere a la separación entre las aplicaciones y las estructuras físicas de almacenamiento.

En los SGBD basados en arquitecturas de varios niveles se hace necesario ampliar el catálogo o el diccionario de datos para incluir la información sobre cómo establecer las correspondencias entre las peticiones de los usuarios y los datos, entre los diversos niveles. El SGBD utiliza una serie de procedimientos adicionales para realizar estas correspondencias haciendo referencia a la información de correspondencia que se encuentra en el diccionario. La independencia de los datos se consigue porque al modificarse el esquema en algún nivel, el esquema del nivel inmediato superior permanece sin cambios. Solo se modifica la correspondencia entre los dos niveles. No es preciso modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

La arquitectura de tres niveles puede facilitar la obtención de la verdadera independencia de datos, tanto física como lógica. Sin embargo, los dos niveles de correspondencia implican un gasto de recursos durante la ejecución de una consulta o de un programa, lo que reduce la eficiencia del SGBD. Por esta razón pocos SGBD han implementado la arquitectura completa.

El nivel clave en esta arquitectura es el conceptual. Éste contiene la descripción de las entidades, relaciones y propiedades de interés para la empresa, y constituye una plataforma estable desde la que proyectar los distintos esquemas externos, que describen los datos según los programadores, sobre el esquema interno, que describe los datos según el sistema

físico. Las posibles proyecciones de datos en los esquemas externo, conceptual e interno, quedan resumidas en la gráfica siguiente:



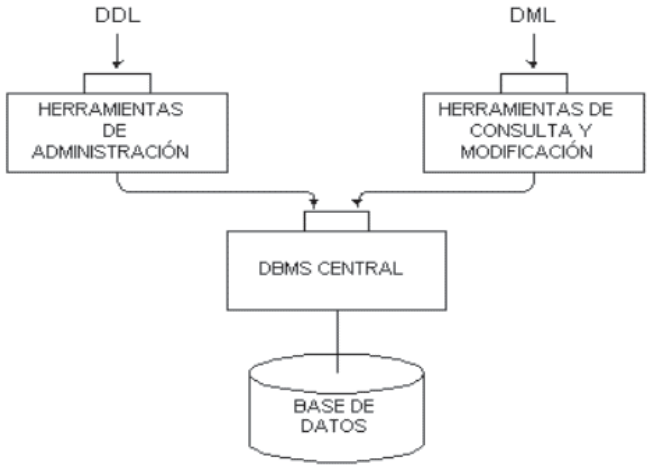
2.3.3 Implementación práctica de la arquitectura de los SGBD

Como cabría esperar, en la práctica cotidiana de implementación de bases de datos, esta arquitectura no es seguida al cien por cien por los SGBD comerciales. Existen muy pocos productos que contengan aplicaciones para facilitar la fase de análisis. Por lo general, el nivel conceptual se obvia en los productos comerciales, salvo honrosas excepciones. Lo habitual es que el administrador de la base de datos realice el modelado conceptual usando sus propios recursos, o tal vez asistido por alguna aplicación de análisis, ya sea general o específica. El procesador del esquema conceptual es por tanto el propio administrador de la base de datos. Los SGBD sí suelen ofrecer facilidades para la creación de esquemas externos, pero sin pasar por el nivel conceptual. Por supuesto, un SGBD comercial no está obligado a seguir las recomendaciones de estandarización de arquitecturas del comité ANSI/X3/SPARC. Por lo que respecta al modelo relacional de bases de datos, que ya existía antes del informe de este comité, los fabricantes de sistemas gestores de base de datos se ajustan en mayor o menor medida al modelo teórico y, en cuanto a la arquitectura, han intentado seguir las recomendaciones del grupo RDBTG (*Relational Data Base Task Group*), parte del comité ANSI/X3/SPARC.

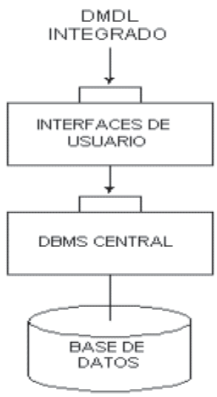
El resultado de este grupo fue restar importancia a las arquitecturas y realzar la de los lenguajes e interfaces. Como consecuencia, el lenguaje SQL está hoy en día totalmente estandarizado, y en cambio encontramos distintas arquitecturas de sistemas gestores de bases de datos relacionales. Sin embargo se pueden distinguir dos tipos generales de arquitecturas para estos sistemas de bases de datos:

- *Arquitectura separada*: en el sistema gestor de base de datos se diferencia claramente entre el *lenguaje de definición de datos* (DDL) y el *lenguaje de manipulación de datos* (DML). El DDL define el esquema conceptual y el esquema interno y lo utilizan los administradores de la base de datos para especificar el

esquema de la BD, las vistas de los usuarios y las estructuras de almacenamiento. EL DML se utiliza para leer y actualizar los datos de la base de datos. A continuación se presenta un esquema de la arquitectura separada de un sistema gestor de base de datos SGBD (o DBMS).



- *Arquitectura integrada*: en este tipo de arquitectura se integran los lenguajes de definición de datos (DDL) y los de manipulación de datos (DML) en un solo lenguaje de manipulación y descripción de datos (DMDL: *Data Manipulation and Description Language)*. Su esquema sería el siguiente:



El tipo de arquitectura integrada es en general preferible a la arquitectura separada y el más común entre los SGBD comerciales. De todos modos, las consecuencias de una integración de los lenguajes de definición de datos (DDL) y los de manipulación de datos (DML) en un solo lenguaje DMDL, pueden ser positivas y negativas. Por un lado, esta integración resulta muy cómoda para el administrador de la base de datos, puesto que le basta con aprender un solo lenguaje formal para realizar todas las tareas de creación y

mantenimiento de la base de datos. Pero por otro lado, estos sistemas (tanto los separados como los integrados) fuerzan una proyección directa desde el nivel externo al interno, haciendo que el nivel conceptual, el fundamental según la arquitectura ANSI/X3/SPARC, desaparezca o se implemente en el nivel externo como una vista global externa. Estaríamos así ante una *arquitectura de dos capas o niveles* en la que no se tiene en cuenta lo suficiente al nivel conceptual.

Por esta razón algunos administradores de bases de datos inexpertos tienden a obviar la fase de análisis, cuando de hecho es la vital para la correcta implementación de la base de datos. Un buen modelado conceptual es una condición indispensable para el correcto desarrollo de una base de datos. Lo ideal es usar un SGBD que nos permita desarrollar todas las tareas (de descripción y de manipulación) lo más fácilmente posible, pero no sin antes disponer de todas las herramientas necesarias para un correcto modelado conceptual, estén éstas o no incluidas en el SGBD.

2.4 COMPONENTES DEL SISTEMA GESTOR DE BASE DE DATOS

Los sistemas gestores de base de datos son paquetes de software muy complejos que deben proporcionar una serie de servicios que van a permitir almacenar y explotar los datos de forma eficiente. Los componentes principales son los lenguajes, el diccionario de datos, la seguridad e integridad de los datos, el gestor de la base de datos, el administrador de la base de datos y los usuarios.

2.4.1 Lenguajes de los sistemas gestores de base de datos

Todos los sistemas gestores de base de datos ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: administradores, diseñadores, programadores de aplicaciones y usuarios finales. Los lenguajes van a permitir al administrador de la base de datos especificar los datos que componen la base de datos, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios. Los lenguajes del sistema gestor de base de datos se clasifican en:

- *Lenguaje de definición de datos (LDD o DDL)*: se utiliza para especificar el esquema de la base de datos, las vistas de los usuarios y las estructuras de almacenamiento. Este tipo de lenguaje define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la base de datos.

- *Lenguaje de manipulación de datos (LMD o DML)*: se utiliza para leer y actualizar los datos de la base de datos. Es el utilizado por los usuarios para realizar

consultas, inserciones, eliminaciones y modificaciones. Hay lenguajes procedurales, en los que el usuario será normalmente un programador y especifica las operaciones de acceso a los datos llamando a los procedimientos necesarios. Estos lenguajes acceden a un registro y lo procesan. Las sentencias de un lenguaje procedural están embebidas en un lenguaje de alto nivel llamado anfitrión. Las bases de datos jerárquicas y en red utilizan estos lenguajes procedurales.

- *Lenguajes declarativos o no procedurales*: en muchos sistemas gestores de base de datos se pueden introducir interactivamente instrucciones del lenguaje procedural desde un terminal y también pueden ir embebidas en un lenguaje de programación de alto nivel. Estos lenguajes permiten especificar los datos a obtener en una consulta, o los datos a modificar, mediante sentencias sencillas. Las bases de datos relacionales utilizan lenguajes no procedurales como SQL (*Structured Query Language*) o QBE (*Query By Example*).

- *Lenguajes 4GL*: la mayoría de los sistemas gestores de bases de datos comerciales incluyen lenguajes de cuarta generación (4GL) que permiten al usuario desarrollar aplicaciones de forma fácil y rápida y a los que también se les llama herramientas de desarrollo. Ejemplos de esto son las herramientas del sistema gestor de base de datos Oracle: *SQL Forms* para la generación de formularios de pantalla y para interactuar con los datos, *SQL Reports* para generar informes de los datos contenidos en la base de datos y PL/SQL que es un lenguaje para crear procedimientos que interactúan con los datos de la base de datos.

2.4.2 El diccionario de datos

El *diccionario de datos* es el lugar donde se deposita información acerca de todos los datos que forman la base de datos. Es una guía en la que se describe la base de datos y los objetos que la forman. El diccionario contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información. En una base de datos relacional, el diccionario de datos proporciona información acerca de:

- La estructura lógica y física de la base de datos.

- Las definiciones de todos los objetos de la base de datos: tablas, vistas, índices, disparadores, procedimientos, funciones, etcétera.

- El espacio asignado y utilizado por los objetos.

- Los valores por defecto de las columnas de las tablas.

- Información acerca de las restricciones de integridad.

- Los privilegios y roles otorgados a los usuarios.
- Auditoría de información, como los accesos a los objetos.
- Un diccionario de datos debe cumplir las siguientes características:
- Debe soportar las descripciones de los modelos conceptual, lógico, interno y externo de la base de datos.
- Debe estar integrado dentro del sistema gestor de base de datos.
- Debe apoyar la transferencia eficiente de información al sistema gestor de base de datos. La conexión entre los modelos interno y externo debe ser realizada en tiempo de ejecución.
- Debe comenzar con la reorganización de versiones de producción de la base de datos. Además debe reflejar los cambios en la descripción de la base de datos. Cualquier cambio en la descripción de programas ha de ser reflejado automáticamente en la librería de descripción de programas con la ayuda del diccionario de datos.
- Debe estar almacenado en un medio de almacenamiento con acceso directo para la fácil recuperación de información.

2.4.3 Seguridad e integridad de datos

Un SGBD proporciona los siguientes mecanismos para garantizar la seguridad e integridad de los datos:

- Debe garantizar la protección de los datos contra accesos no autorizados, tanto intencionados como accidentales y debe controlar que solo los usuarios autorizados accedan a la base de datos.
- Los sistemas gestores de base de datos ofrecen mecanismos para implantar restricciones de integridad en la base de datos. Estas restricciones van a proteger la base de datos contra daños accidentales. Los valores de los datos que se almacenan deben satisfacer ciertos tipos de restricciones de consistencia y reglas de integridad, que especificará el administrador de la base de datos. El sistema gestor de la base de datos puede determinar si se produce una violación de la restricción.
- Proporciona herramientas y mecanismos para la planificación y realización de copias de seguridad y restauración.

- Debe ser capaz de recuperar la base de datos llevándola a un estado consistente en caso de ocurrir algún suceso que la dañe.

- Debe asegurar el acceso concurrente y ofrecer mecanismos para conservar la consistencia de los datos en el caso de que varios usuarios actualicen la BD de forma concurrente.

2.4.4 El administrador de la base de datos

El administrador de la base de datos o DBA tiene una gran responsabilidad ya que posee el máximo nivel de privilegios. Será el encargado de crear los usuarios que se conectarán a la base de datos. En la administración de una base de datos siempre hay que procurar que haya el menor número de administradores, a ser posible una sola persona. El objetivo principal de un DBA es garantizar que la base de datos cumple los fines previstos por la organización, lo que incluye una serie de tareas como:

- Instalar el sistema gestor de base de datos en el sistema informático.

- Crear las bases de datos que se vayan a gestionar.

- Crear y mantener el esquema de la base de datos. El esquema puede consistir en una lista de tablas en uso, los campos que contienen, el tipo de datos de cada campo, descripciones en lenguaje natural del propósito de cada tabla y cada campo, y restricciones sobre los valores admisibles en cada campo.

- Crear y mantener las cuentas de usuario de la base de datos.

- Arrancar y parar el SGBD, y cargar las bases de datos con las que se ha de trabajar.

- Colaborar con el administrador del sistema operativo en las tareas de ubicación, dimensionado y control de los archivos y espacios de disco ocupados por el SGBD.

- Colaborar en las tareas de formación de usuarios.

- Establecer estándares de uso, políticas de acceso y protocolos de trabajo diario para los usuarios de la base de datos.

- Suministrar la información necesaria sobre la base de datos a los equipos de análisis y programación de aplicaciones.

- Efectuar tareas de explotación.

- Vigilar el trabajo diario colaborando en la información y resolución de las dudas de los usuarios de la base de datos.

- Controlar en tiempo real los accesos, tasas de uso, cargas en los servidores, anomalías, etcétera.

- Llegado el caso, reorganizar la base de datos.

- Efectuar las copias de seguridad periódicas de la base de datos.

- Restaurar la base de datos después de un incidente material a partir de las copias de seguridad.

- Estudiar las auditorías del sistema para detectar anomalías, intentos de violación de la seguridad, etc.

- Ajustar y optimizar la base de datos mediante el ajuste de sus parámetros, y con ayuda de las herramientas de monitorización y de las estadísticas del sistema.

En su gestión diaria, el administrador de la base de datos suele utilizar una serie de herramientas de administración de la base de datos. Con el paso del tiempo, estas herramientas han adquirido sofisticadas prestaciones y facilitan en gran medida la realización de trabajos que, hasta no hace demasiado, requerían de arduos esfuerzos por parte de los administradores.

2.4.5 Usuarios

Muchas personas participan en el diseño, uso y mantenimiento de una base de datos grande. Los usuarios habituales de una base de datos pueden clasificarse como sigue:

- *Administradores*: si consideramos a la base de datos y al SGBD como recursos del sistema de base de datos, debemos considerar a una persona que administre dichos recursos. El administrador de la base de datos (DBA, en inglés) es quien se encarga de autorizar el acceso a la base de datos, de coordinar y vigilar su empleo, y de adquirir los recursos necesarios de software y hardware. El DBA es la persona responsable cuando surgen problemas como violaciones a la seguridad o una respuesta lenta del sistema.

- *Diseñadores de bases de datos*: se encargan de identificar los datos que se almacenarán en la base de datos y de elegir las estructuras apropiadas para representar y almacenar dichos datos.

- *Operadores de bases de datos*: se encargan de realizar los trabajos rutinarios en la base de datos y disponen de permisos concretos para realizar sus operaciones. Su acceso a la base de datos se restringe concretamente a su funcionalidad estricta en la misma.

- *Analistas y programadores de aplicaciones*: los analistas determinan los requerimientos de los usuarios finales y desarrollan especificaciones para transacciones programadas que satisfagan dichos requerimientos. Los programadores implementan estas especificaciones en forma de programas y luego prueban, depuran, documentan y mantienen esas transacciones programadas.

- *Usuarios finales*: son las personas que necesitan tener acceso a la base de datos para consultarla, actualizarla y generar informes; la base de datos existe primordialmente para que ellos la usen.

2.5 SISTEMAS GESTORES DE BASE DE DATOS PARALELOS

Los sistemas paralelos de base de datos constan de varios procesadores y varios discos conectados a través de una red de interconexión de alta velocidad. Para medir el rendimiento de los sistemas gestores de base de datos existen dos medidas principales: la primera es la productividad (*throughput*) que se entiende como el número de tareas que pueden completarse en un intervalo de tiempo determinado. La segunda es el tiempo de respuesta (*response time*) que es la cantidad de tiempo que necesita para completar una única tarea a partir del momento en que se envíe.

Un sistema que procese un gran número de pequeñas transacciones puede mejorar su productividad realizando muchas transacciones en paralelo. Un sistema que procese transacciones más largas puede mejorar tanto su productividad como sus tiempos de respuesta realizando en paralelo cada una de las subtareas de cada transacción.

Las ganancias en este tipo de SGBD se pueden dar en términos de velocidad (menor tiempo de ejecución para una tarea dada) y ampliabilidad (capacidad de procesar tareas más largas en el mismo tiempo).

Existen varios modelos de arquitecturas para máquinas paralelas, los más mencionados son:

- *Memoria compartida*: todos los procesadores comparten una memoria común.

- *Disco compartido*: todos los procesadores comparten una disposición de discos común.

- *Sin compartimiento*: los procesadores no comparten ni memoria ni disco.

- *Jerárquico*: compartimiento tanto de memoria como de disco.

2.6 SISTEMAS GESTORES DE BASE DE DATOS DISTRIBUIDOS

En un SGBD distribuido, la base de datos se almacena en varios computadores que se pueden comunicar a su vez por distintos medios de comunicación (desde redes de alta velocidad a líneas telefónicas). No comparten memoria ni discos y sus tamaños pueden variar tanto como sus funciones pudiendo abarcar desde PC hasta grandes sistemas. Se denomina con el término de emplazamientos o nodos a todos aquellos computadores que pertenecen a un sistema distribuido.

Las características más importantes de las bases de datos distribuidas y que las diferencian de las bases de datos paralelas son las siguientes:

- Las bases de datos distribuidas se encuentran normalmente en varios lugares geográficos distintos.
- Las bases de datos distribuidas se administran de forma separada y poseen una interconexión más lenta.
- En un sistema distribuido se dan dos tipos de transacciones, las locales y las globales. Una transacción local es aquella que accede a los datos del único emplazamiento en el cual se inició la transacción. Por otra parte una transacción global es aquella que o bien accede a los datos situados en un emplazamiento diferente de aquel en el que se inició la transacción, o bien accede a datos de varios emplazamientos distintos.
- Los distintos emplazamientos están informados de los demás. Aunque algunas relaciones pueden estar almacenadas solo en algunos emplazamientos, estos comparten un esquema global común.
- Cada emplazamiento proporciona un entorno para la ejecución de transacciones tanto locales como globales.

2.7 CARACTERÍSTICAS DE LOS GESTORES DE BASE DE DATOS MÁS HABITUALES EN EL MERCADO ACTUAL

En los párrafos siguientes se presenta una comparativa entre los sistemas gestores de base de datos más utilizados actualmente en lo relativo a información general. Sistema operativo con el que trabajan, características fundamentales, uso de tablas y vistas, índices y particionamiento.

2.7.1 Información general

Sistema Gestor de BD	Creador	Fecha de la primera versión pública	Última versión estable	Licencia de software
Adaptive Server Anywhere	Sybase/Anywhere	1992	10.0	Propietario
Adaptive Server Enterprise	Sybase Inc	1987	15.0	Propietario
ANTs Data Server	ANTs Software	1999	3.6	Propietario
DB2	IBM	1982	9	Propietario
Firebird	Firebird Foundation	25 de julio de 2000	2.1	Licencia pública InterBase
Informix	Informix Software	1985	10.0	Propietario
HSQLDB	Hsqldb.Org	2001	1.9	Licencia BSD
Ingres	Berkeley University, Computer Associates	1980	2006	CA-TOSL
InterBase	Borland	1985	7.5.1	Propietario
SapDB	SAP AG		7.4	GPL con drivers LGPL
MaxDB	MySQL AB, SAP AG		7.7	GPL o propietario
Microsoft SQL Server	Microsoft	1989	2008	Propietario
MySQL	MySQL AB	Noviembre de 1996	5.0	GPL o propietario
Oracle	Oracle Corporation	1977	11g R 2	Propietario
PostgreSQL	PostgreSQL Global Development Group	Junio de 1989	9.0	Licencia BSD
SmallSQL	SmallSQL	16 de abril de 2005	0.12	LGPL
SQLite	D. Richard Hipp	17 de agosto de 2000	3.6.16	Dominio público

2.7.2 Soporte del sistema operativo

	Windows	Mac OS X	Linux	BSD	Unix	z/OS
Adaptive Server Enterprise	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	?
DB2	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí	✓ Sí
Firebird	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	Quizás
HSQLDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Informix	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No
Ingres	✓ Sí		✓ Sí		✓ Sí	Quizás
InterBase	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí (Solaris)	✗ No
SapDB	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí	
MaxDB	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí	
Microsoft SQL Server	✓ Sí	✗ No	✗ No	✗ No	✗ No	✗ No
MySQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	Quizás
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No
SmallSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SQLite	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	Quizás

2.7.3 Características fundamentales

	Características ACID	Integridad referencial	Transacciones	Unicode
Adaptive Server Enterprise	✓ Sí	✓ Sí	✓ Sí	✓ Sí
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Firebird	✓ Sí	✓ Sí	✓ Sí	✓ Sí
HSQLDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Informix	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Ingres	✓ Sí	✓ Sí	✓ Sí	✓ Sí
InterBase	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SapDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí
MaxDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Microsoft SQL Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí
MySQL	Depende	Depende	Depende	✓ Sí
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SQLite	✓ Sí	✗ No	Básico	✓ Sí

X en bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones pueda ser considerada como una transacción. En concreto ACID es un acrónimo de *Atomicity*, *Consistency*, *Isolation and Durability* (Atomicidad, Consistencia, Aislamiento y Durabilidad en español).

- *Atomicidad*: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- *Consistencia*: es la propiedad que asegura que solo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- *Aislamiento*: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que en la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- *Durabilidad*: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

2.7.4 Tablas y vistas

	Tabla temporal	Vista materializada
Adaptive Server Enterprise	✓ Sí	✓ Sí
ANTs Data Server	✓ Sí	✓ Sí
DB2	✓ Sí	✓ Sí
Firebird	✓ Sí	✗ No
HSQLDB	✓ Sí	✗ No
Informix	✓ Sí	✓ Sí
Ingres	✓ Sí	✗ No
InterBase	✓ Sí	✗ No
SapDB	✓ Sí	✗ No
MaxDB	✓ Sí	✗ No
Microsoft SQL Server	✓ Sí	Similar
MySQL	✓ Sí	✗ No
Oracle	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✗ No
SQLite	✓ Sí	✗ No

2.7.5 Índices

	Árbol R-/R+	Hash	Expresión	Parcial	Reversa	Mapa de bits
Adaptive Server Enterprise	✗ No	✗ No	✓ Sí	✗ No	✓ Sí	✗ No
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✗ No	?	✗ No	✗ No	✓ Sí	✓ Sí
Firebird	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No
Informix	✓ Sí	✓ Sí	✓ Sí	✗ No	✗ No	✗ No
Ingres	✓ Sí	✓ Sí	✗ No	✗ No	✗ No	✗ No
InterBase			✗ No	✗ No	✗ No	✗ No
SapDB			✗ No	✗ No	✗ No	✗ No
MaxDB			✗ No	✗ No	✗ No	✗ No
Microsoft SQL Server			✗ No	✗ No	✗ No	✗ No
MySQL	Tablas MyISAM solo	Tablas HEAP solo	✗ No	✗ No	✗ No	✗ No
Oracle	Edición EE solo		✓ Sí	✗ No	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No	✗ No
SQLite	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No

2.7.6 Otros objetos

	Dominio	Cursor	Trigger	Funciones	Procedimiento	Rutina externa
Adaptive Server Enterprise	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Firebird	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
HSQLDB		✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Informix		✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Ingres	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	
InterBase	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SapDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	
MaxDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	
Microsoft SQL Server	✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
MySQL	✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí [3]	✓ Sí
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SQLite	✗ No	✗ No	✓ Sí	✗ No	✗ No	✓ Sí

Funciones y *Procedimiento* se refieren a las rutinas internas escritas en SQL o lenguajes procedurales como PL/SQL. *Rutina externa* se refiere a la escritura en los lenguajes anfitriones como C, Java, Cobol, etc. Procedimiento almacenado es un término comúnmente usado para ese tipo de rutinas. Sin embargo, su definición varía entre diferentes vendedores de bases de datos.

2.7.7 Particionamiento

	Rango	Hash	Compuesto (Rango+Hash)	Lista
Adaptive Server Enterprise	AA	AA	AA	AA
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Firebird	✗ No	✗ No	✗ No	✗ No
Ingres	✓ Sí	✓ Sí	✓ Sí	✓ Sí
InterBase	✗ No	✗ No	✗ No	✗ No
Microsoft SQL Server	✓ Sí	✗ No	✗ No	✗ No
MySQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✗ No	✗ No	✓ Sí
SQLite	✓ Sí	✓ Sí	✓ Sí	✓ Sí

La tabla presenta información acerca de qué métodos de particionamiento son soportados nativamente por los diferentes SGBD.

Una partición es una división de una base de datos lógica o sus elementos constituyentes en partes independientes. La partición de bases de datos se hace normalmente por razones de mantenimiento, rendimiento o manejo. Se utiliza habitualmente en sistemas de administración de bases de datos distribuidas. Cada partición puede ser extendida hasta múltiples nodos, y los usuarios en el nodo pueden hacer transacciones locales en la partición. Esto aumenta el rendimiento en sitios que tienen transacciones regularmente involucrando ciertas vistas de datos, y manteniendo la disponibilidad y la seguridad. Esta partición puede hacerse creando bases de datos más pequeñas separadas (cada una con sus propias tablas, índices y registros de transacciones) o dividiendo elementos seleccionados, por ejemplo, solo una tabla.

EJEMPLO DE BASE DE DATOS CON EL SGBD ACCESS

Consideramos una base de datos de nombre PERSONAL con información relativa al personal de una empresa y que tiene cuatro tablas de nombres EMPLEADO (empleados de la empresa), OFICIOEMPLEADO (empleados con su oficio), VIVIENDA (alojamiento de los empleados) y OFICIO (relación de oficios), cuyos campos y registros contienen la siguiente información:

EMPLEADO

NOMBRE	EDAD	ALOJAMIENTO
ALONSO TELLEZ	23	PARACUELLOS
DÍAZ JIMÉNEZ	18	BARAJAS
EZQUERRO TOMÉ	43	VALLECAS
GONZÁLEZ ORBEA	41	BARAJAS
PÉREZ LÓPEZ	21	BARAJAS
VÁZQUEZ LÓPEZ	67	PITIS
GONZÁLEZ LÓPEZ	23	PONCE
JUÁREZ NORIEGA	25	TOLEDO
TEJERINA RUIZ	31	BRASILIA
ÁLVAREZ RODRÍGUEZ	23	PARACUELLOS
DONOSO ANTÓN	18	BARAJAS
EGUÍBAR PUENTE	43	VALLECAS
GOMEZ ORDUÑA	41	BARAJAS
PAVÓN LÓPEZ	21	BARAJAS
PANADERO JUAN	21	BARAJAS

OFICIOEMPLEADO

NOMBRE	CALIFICACION	OFICIO
ALONSO TELLEZ	HERRERO	EXCELENTE
DÍAZ JIMÉNEZ	HERRERO	EXCELENTE
DORADO RODRÍGUEZ	CONDUCTOR	EXCELENTE
EZQUERRO TOMÉ	LEÑADOR	BUENO
GONZÁLEZ ORBEA	HERRERO	PRECISO
PÉREZ LÓPEZ	CONDUCTOR	RÁPIDO
PEDRERO LÁINEZ	OBRERO	NORMAL
VÁZQUEZ LÓPEZ	LABRADOR	LENTO
DONOSO CID	LABRADOR	NORMAL
GONZÁLEZ LÓPEZ	HERRERO	EXCELENTE
JUÁREZ NORIEGA	CONDUCTOR	EXCELENTE
TEJERINA RUIZ	LEÑADOR	BUENO
ALVAREZ RODRÍGUEZ	CONDUCTOR	RÁPIDO
DONOSO ANTÓN	HERRERO	PRECISO
DARÍO RODRÍGUEZ	OBRERO	NORMAL
EGUÍBAR PUENTE	LABRADOR	LENTO
GOMEZ ORDUÑA	HERRERO	EXCELENTE
PAVÓN LÓPEZ	CONDUCTOR	EXCELENTE
PANADERO JUAN	PICADOR	PRECISO

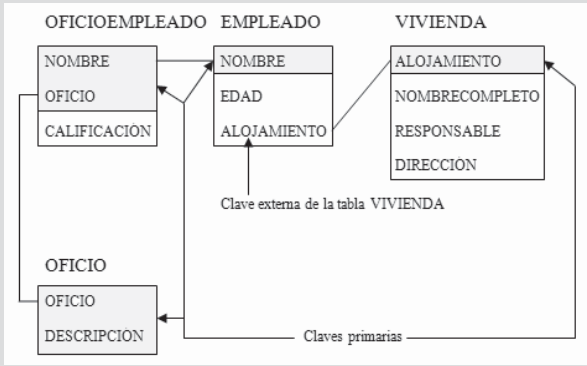
VIVIENDA

ALOJAMIENTO	NOMBRECOMPLETO	RESPONSABLE	DIRECCION
PARACUELLOS	PARACUELLOS JARAMA	ALONSO TELLEZ	ELIPA, 12
VALLECAS	PUENTE VALLECAS	EZQUERRO TOMÉ	MUSAS, 1
PONCE	PONCE LEÓN	GONZÁLEZ LÓPEZ	PONCE, 23
TOLEDO	PUERTA TOLEDO	JUÁREZ NORIEGA	TOLEDO, 25
BARAJAS	BARAJAS PUEBLO	PÉREZ LÓPEZ	ELOY, 2
BRASILIA	PARQUE BRASILIA	TEJERINA RUIZ	BRASILIA, 3
PITIS	ESTACIÓN PITIS	VÁZQUEZ LÓPEZ	PONCE, 46

OFICIO

OFICIO	DESCRIPCION
HERRERO	HERRAR CABALLOS
CONDUCTOR	CONDUCTOR DE AUTOBÚS
LEÑADOR	TALAR ÁRBOLES
OBRERO	TRABAJAR EN GENERAL
LABRADOR	CULTIVAR LA TIERRA
PICADOR	PICAR EN LOS TÚNELES DEL METRO

Para realizar el *diseño conceptual* de la base de datos sabemos que en la tabla VIVIENDA tenemos como restricción de integridad que el campo *Alojamiento* es clave única. En la tabla EMPLEADO tenemos que el campo *Nombre* es clave única siendo el campo *Alojamiento* la clave externa de la tabla VIVIENDA. En la tabla OFICIOEMPLEADO la clave única está formada por los campos *Nombre* y *Oficio* simultáneamente. Por último, en la tabla OFICIO la clave única está formada por los campos *Oficio* y *Descripción* simultáneamente. El diseño adecuado para esta base de datos relacional corresponde al *esquema conceptual* de la figura siguiente:



A continuación crearemos la base de datos PERSONAL adecuada al diseño conceptual anterior utilizando las herramientas apropiadas que suministra el SGBD Access.

Para *crear una nueva base de datos en blanco*, inicie Access y en el botón *Archivo* seleccione *Nuevo*. Bajo *Plantillas disponibles*, haga clic en *Base de datos en blanco*. A continuación, en la parte inferior del panel *Base de datos en blanco*, en el cuadro *Nombre de archivo*, escriba un nombre de archivo (Figura 2.1). Por último, haga clic en *Crear*. Se crea la base de datos y se abre una tabla en la vista *Hoja de datos* (Figura 2.2) lo que permite crear ya las tablas.

Es posible agregar inmediatamente campos a la tabla haciendo clic sobre *Haga clic para agregar* en la Figura 2.2 y escribiendo su nombre. Al pulsar Enter, *Haga clic para agregar* se desplaza a la siguiente columna a la derecha. Para introducir un nuevo campo en la tabla se vuelve a hacer clic en *Haga clic para agregar* y se escribe su nombre y así sucesivamente se pueden ir añadiendo más campos hasta completar todos los campos de la tabla (Figura 2.3).

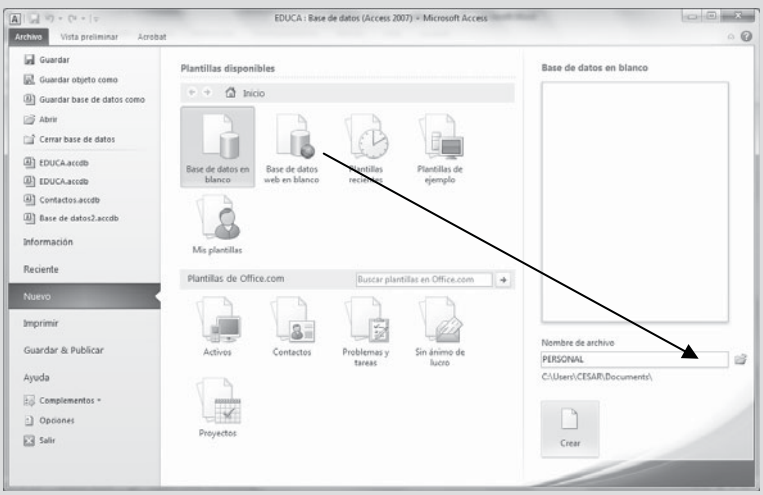


Figura 2.1

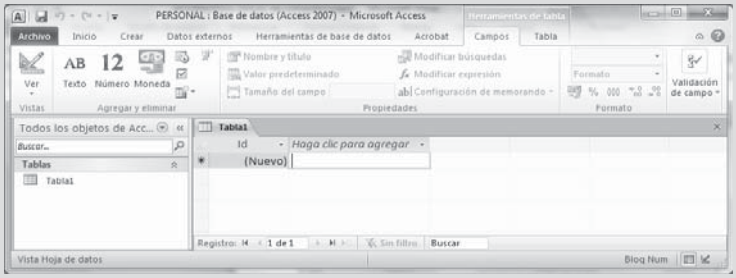


Figura 2.2

La siguiente tarea será introducir los datos en los campos como si se tratase de una hoja de cálculo. De momento los tipos de datos y sus propiedades los asigna Access por defecto de acuerdo a la información que vamos almacenando en la tabla. Finalizada la tabla, se guarda a través de la opción *Guardar* del menú emergente que se obtiene al hacer clic con el botón derecho del ratón sobre la ficha de la tabla y se le asigna el nombre VIVIENDA. La Figura 2.4 muestra la tabla con datos.

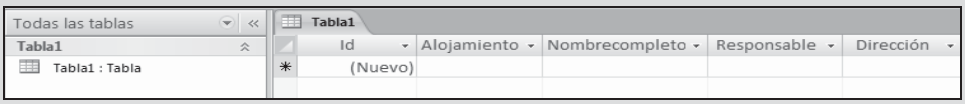


Figura 2.3

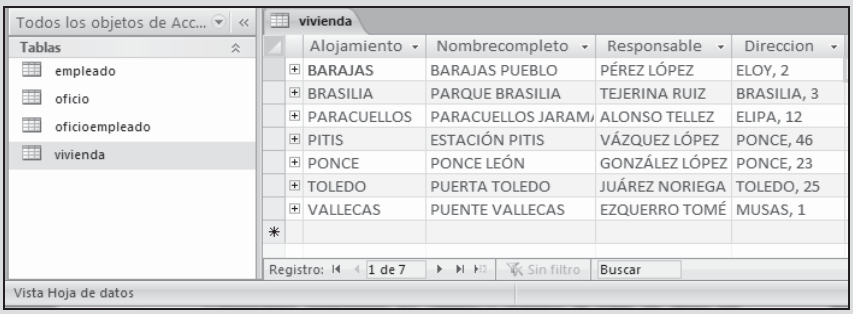


Figura 2.4

En cuanto a tipos de datos, formatos y propiedades de los campos, Access ha elegido por defecto los más adecuados al introducir los datos. Pero podemos cambiarlos a medida utilizando la vista *Diseño*. Para ello se hace clic en *Vista Diseño* en la opción *Ver* del grupo *Vistas* y se obtiene la tabla en vista *Diseño* apta para cambiar las propiedades a medida a todos sus campos. La primera tarea es seleccionar el campo *Alojamiento* y hacer clic en *Clave principal* en el grupo *Herramientas* de la ficha *Diseño* para situar sobre este campo la clave primaria (Figuras 2.5). El resto de propiedades permanecen como aparecen en la figura.

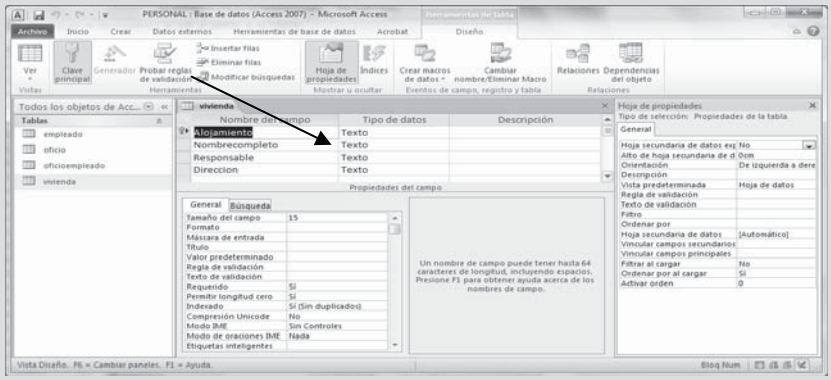


Figura 2.5

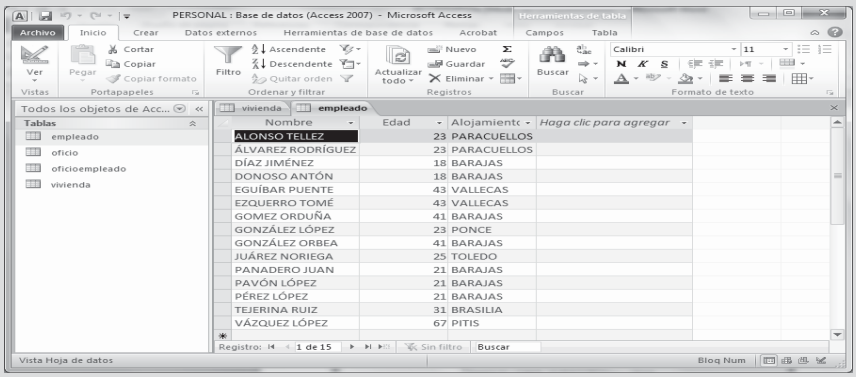


Figura 2.6

En la Figura 2.6 se presenta la tabla EMPLEADO con sus campos y registros en la vista *Hoja de datos*. La Figura 2.7 muestra la clave primaria en vista *Diseño*.

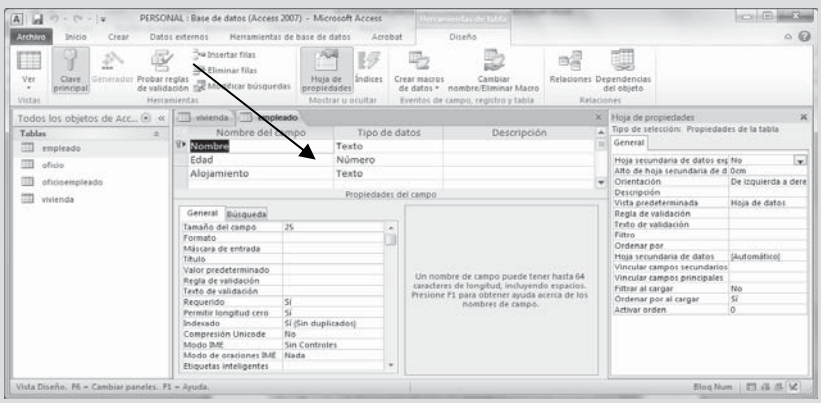


Figura 2.7

En la Figura 2.8 se presenta la tabla OFICIO con sus campos y registros en la vista *Hoja de datos*. La Figura 2.9 muestra la clave primaria en vista *Diseño* (campos *Oficio* y *Descripción*).

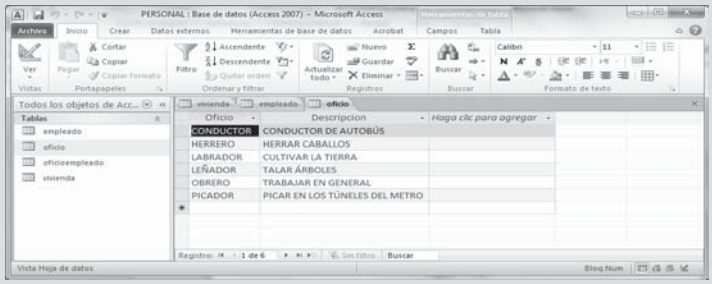


Figura 2.8

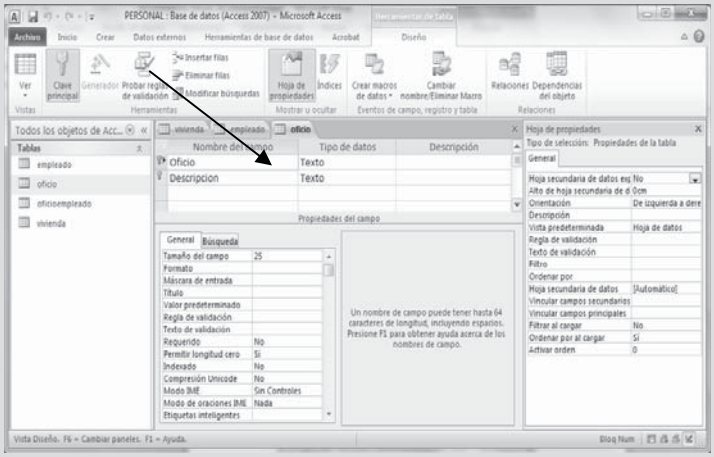


Figura 2.9

En la Figura 2.10 se presenta la tabla OFICIOEMPLEADO con sus campos y registros en la vista *Hoja de datos*.

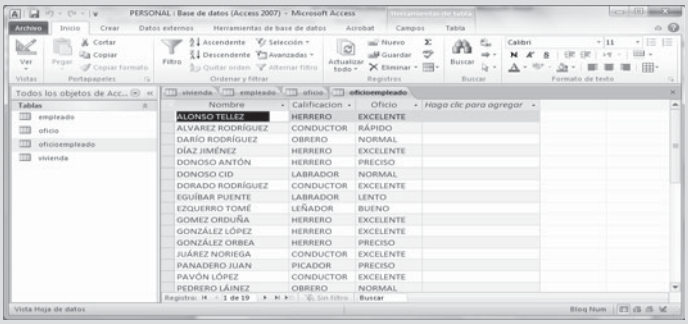


Figura 2.10

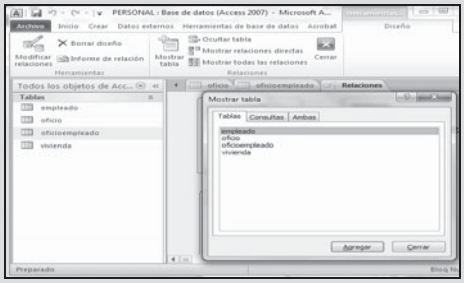


Figura 2.11

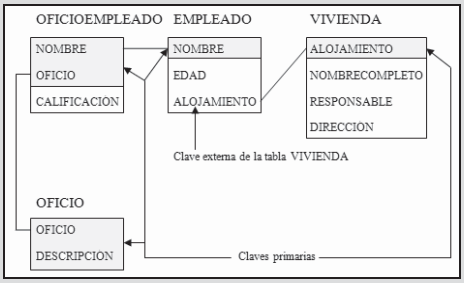


Figura 2.12

Arrastre las tablas hasta conseguir una estructura similar a la del diagrama de diseño de la base de datos (Figura 2.12). Para arrastrar una tabla se hace clic sobre su cabecera con el ratón y, sin soltarlo, se mueve hasta la posición en que se quiere situar la tabla. Una estructura acorde al diseño de la base de datos se ve en la Figura 2.13.

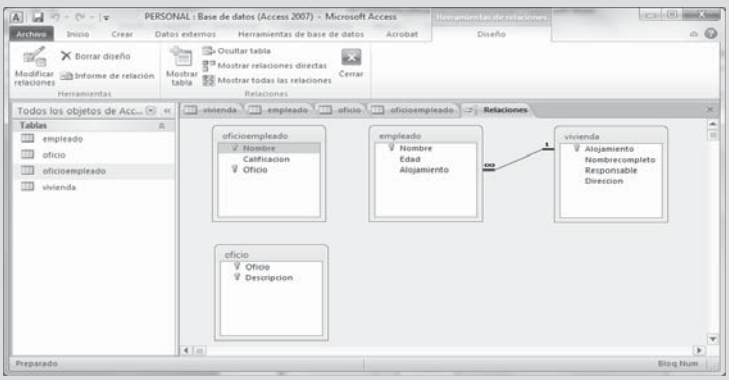


Figura 2.13

También podíamos haber utilizado el *entorno del lenguaje relacional SQL de Access* para construir las tablas de esta base de datos.

Los pasos a seguir para habilitar el entorno de edición de SQL en Access son los siguientes:

- Haga clic en la opción *Diseño de consulta* del grupo *Consultas* de la ficha *Crear* (Figura 2.14).

- A continuación cierre el cuadro *Mostrar tabla* de la Figura 2.15 y haga clic en el icono *Vista SQL* SQL de la barra de estado (o se elige *Vista SQL* en el menú *Ver*) para situarnos sobre el editor de consultas SQL (Figura 2.16) habilitándose la ficha *Herramientas de consulta* con la subficha *Diseño*.

grupo *Resultados* (Figura 2.17) para obtener los resultados de la consulta (Figura 2.18).

- Para diseñar una nueva consulta SQL vuelva a hacer clic en el icono *Vista SQL* SQL de la barra de estado o se elige *Vista SQL* en el menú *Ver* (Figura 2.19).

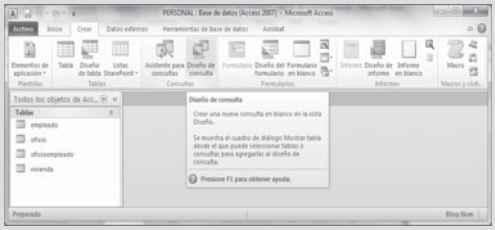


Figura 2.14

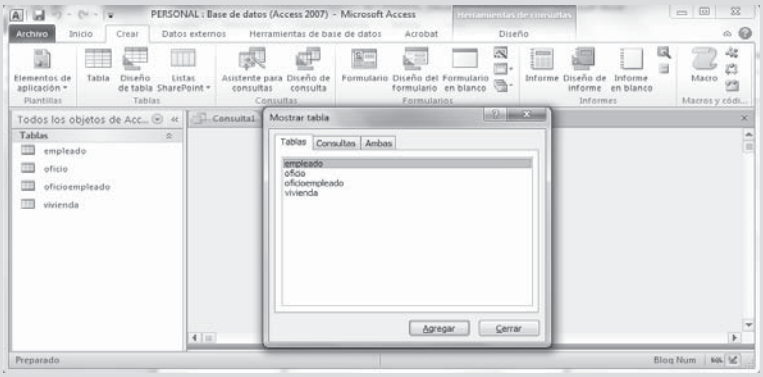


Figura 2.15

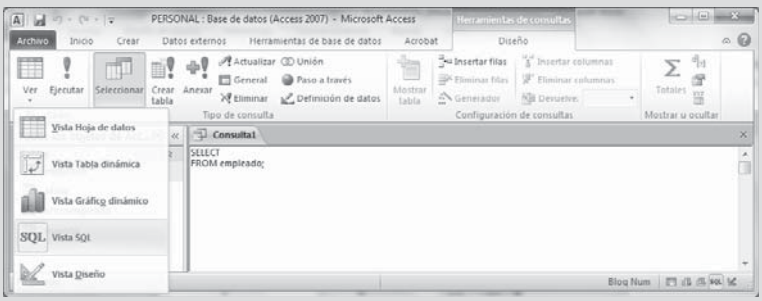


Figura 2.16

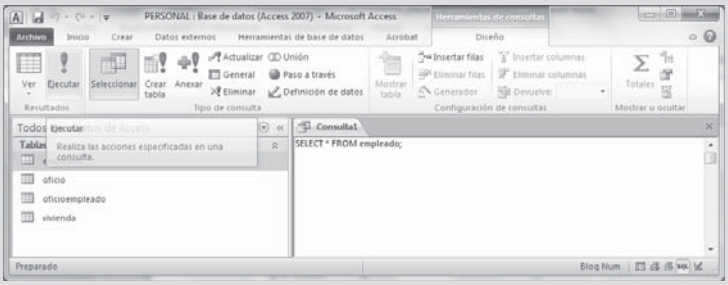


Figura 2.17

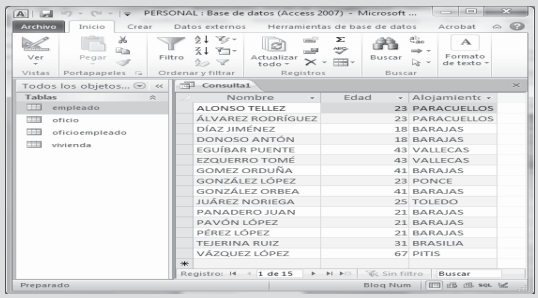


Figura 2.18

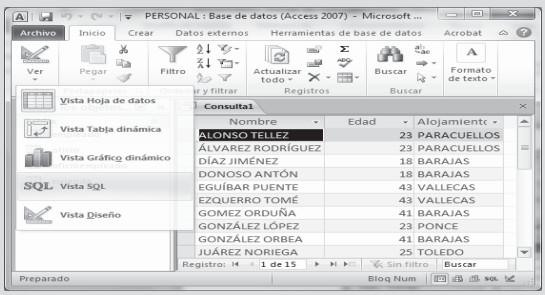


Figura 2.19

Las sintaxis de creación de tablas mediante la sentencia CREATE TABLE del *lenguaje de definición de datos* DDL del SGBD Access se presenta en las Figuras 2.20 a 2.23.

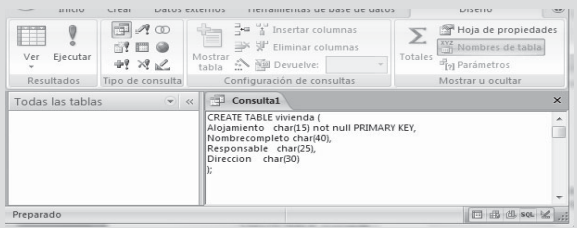


Figura 2.20

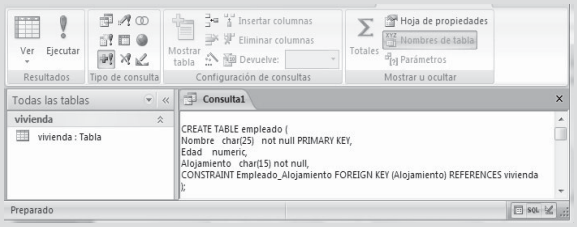


Figura 2.21

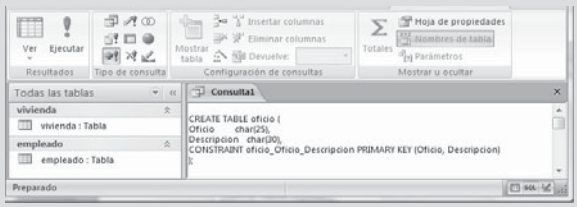


Figura 2.22

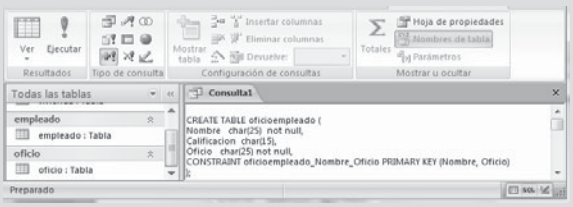


Figura 2.23

La inserción de los registros en las tablas podría haberse realizado mediante la sentencia INSERT del *lenguaje de modificación de datos* DML del SGBD Access. La sintaxis SQL de esta inserción se presenta en las Figuras 2.24 a 2.27.

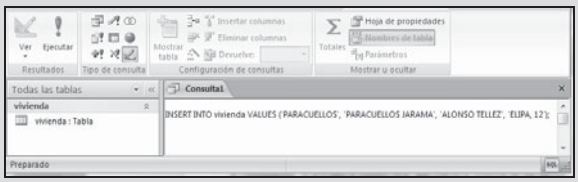


Figura 2.24

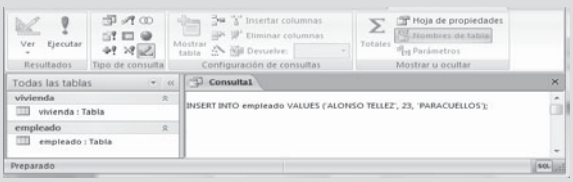


Figura 2.25

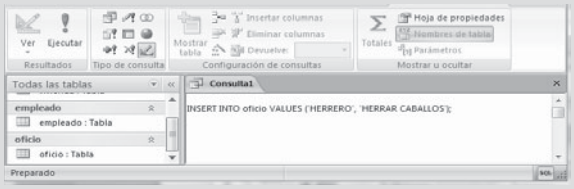


Figura 2.26

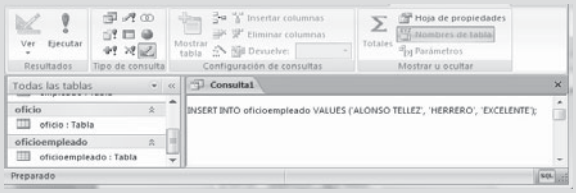


Figura 2.27

La sintaxis completa para la inserción de todos los datos de la tabla VIVIENDA es la siguiente:

```
INSERT INTO vivienda VALUES ('PARACUELLOS', 'PARACUELLOS JARAMA', 'ALONSO TELLEZ','ELIPA,12');
INSERT INTO vivienda VALUES ('VALLECAS', 'PUENTE VALLECAS', 'EZQUERRO TOMÉ', 'MUSAS, 1');
INSERT INTO vivienda VALUES ('PONCE', 'PONCE LEÓN', 'GONZÁLEZ LÓPEZ', 'PONCE, 23');
INSERT INTO vivienda VALUES ('TOLEDO', 'PUERTA TOLEDO', 'JUÁREZ NORIEGA', 'TOLEDO, 25');
INSERT INTO vivienda VALUES ('BARAJAS', 'BARAJAS PUEBLO', 'PÉREZ LÓPEZ', 'ELOY, 2');
INSERT INTO vivienda VALUES ('BRASILIA', 'PARQUE BRASILIA', 'TEJERINA RUIZ', 'BRASILIA, 3');
INSERT INTO vivienda VALUES ('PITIS', 'ESTACIÓN PITIS', 'VÁZQUEZ LÓPEZ', 'PONCE, 46');
```

La sintaxis completa para la inserción de todos los datos de la tabla EMPLEADO es la siguiente:

```
INSERT INTO empleado VALUES ('ALONSO TELLEZ', 23, 'PARACUELLOS');
INSERT INTO empleado VALUES ('DÍAZ JIMÉNEZ', 18, 'BARAJAS');
INSERT INTO empleado VALUES ('EZQUERRO TOMÉ', 43, 'VALLECAS');
INSERT INTO empleado VALUES ('GONZÁLEZ ORBEA', 41, 'BARAJAS');
INSERT INTO empleado VALUES ('PÉREZ LÓPEZ', 21, 'BARAJAS');
INSERT INTO empleado VALUES ('VÁZQUEZ LÓPEZ', 67, 'PITIS');
INSERT INTO empleado VALUES ('GONZÁLEZ LÓPEZ', 23, 'PONCE');
INSERT INTO empleado VALUES ('JUÁREZ NORIEGA', 25, 'TOLEDO');
INSERT INTO empleado VALUES ('TEJERINA RUIZ',31, 'BRASILIA');
INSERT INTO empleado VALUES ('ÁLVAREZ RODRÍGUEZ', 23, 'PARACUELLOS');
INSERT INTO empleado VALUES ('DONOSO ANTÓN', 18, 'BARAJAS');
INSERT INTO empleado VALUES ('EGUÍBAR PUENTE', 43, 'VALLECAS');
INSERT INTO empleado VALUES ('GOMEZ ORDUÑA', 41, 'BARAJAS');
INSERT INTO empleado VALUES ('PAVÓN LÓPEZ', 21, 'BARAJAS');
INSERT INTO empleado VALUES ('PANADERO JUAN', 21, 'BARAJAS');
```

La sintaxis completa para la inserción de todos los datos de la tabla OFICIO es la siguiente:

```
INSERT INTO oficio VALUES ('HERRERO', 'HERRAR CABALLOS');
INSERT INTO oficio VALUES ('CONDUCTOR', 'CONDUCTOR DE AUTOBÚS');
INSERT INTO oficio VALUES ('LEÑADOR', 'TALAR ÁRBOLES');
INSERT INTO oficio VALUES ('OBRERO', 'TRABAJAR EN GENERAL');
INSERT INTO oficio VALUES ('LABRADOR', 'CULTIVAR LA TIERRA');
INSERT INTO oficio VALUES ('PICADOR', 'PICAR EN LOS TÚNELES DEL METRO');
```

La sintaxis completa para la inserción de todos los datos de la tabla OFICIOEMPLEADO es la siguiente:

```
INSERT INTO oficioempleado VALUES ('ALONSO TELLEZ', 'HERRERO', 'EXCELENTE');
INSERT INTO oficioempleado VALUES ('DÍAZ JIMÉNEZ', 'HERRERO', 'EXCELENTE');
INSERT INTO oficioempleado VALUES ('DORADO RODRÍGUEZ', 'CONDUCTOR', 'EXCELENTE');
INSERT INTO oficioempleado VALUES ('EZQUERRO TOMÉ', 'LEÑADOR', 'BUENO');
INSERT INTO oficioempleado VALUES ('GONZÁLEZ ORBEA', 'HERRERO', 'PRECISO');
INSERT INTO oficioempleado VALUES ('PÉREZ LÓPEZ', 'CONDUCTOR', 'RÁPIDO');
INSERT INTO oficioempleado VALUES ('PEDRERO LÁINEZ', 'OBRERO', 'NORMAL');
INSERT INTO oficioempleado VALUES ('VÁZQUEZ LÓPEZ', 'LABRADOR', 'LENTO');
INSERT INTO oficioempleado VALUES ('DONOSO CID', 'LABRADOR', 'NORMAL');
INSERT INTO oficioempleado VALUES ('GONZÁLEZ LÓPEZ', 'HERRERO', 'EXCELENTE');
INSERT INTO oficioempleado VALUES ('JUÁREZ NORIEGA', 'CONDUCTOR', 'EXCELENTE');
INSERT INTO oficioempleado VALUES ('TEJERINA RUIZ', 'LEÑADOR', 'BUENO');
INSERT INTO oficioempleado VALUES ('ALVAREZ RODRÍGUEZ', 'CONDUCTOR', 'RÁPIDO');
INSERT INTO oficioempleado VALUES ('DONOSO ANTÓN', 'HERRERO', 'PRECISO');
INSERT INTO oficioempleado VALUES ('DARÍO RODRÍGUEZ', 'OBRERO', 'NORMAL');
INSERT INTO oficioempleado VALUES ('EGUÍBAR PUENTE', 'LABRADOR', 'LENTO');
INSERT INTO oficioempleado VALUES ('GOMEZ ORDUÑA', 'HERRERO', 'EXCELENTE');
INSERT INTO oficioempleado VALUES ('PAVÓN LÓPEZ', 'CONDUCTOR', 'EXCELENTE');
INSERT INTO oficioempleado VALUES ('PANADERO JUAN', 'PICADOR', 'PRECISO');
```

Capítulo

3

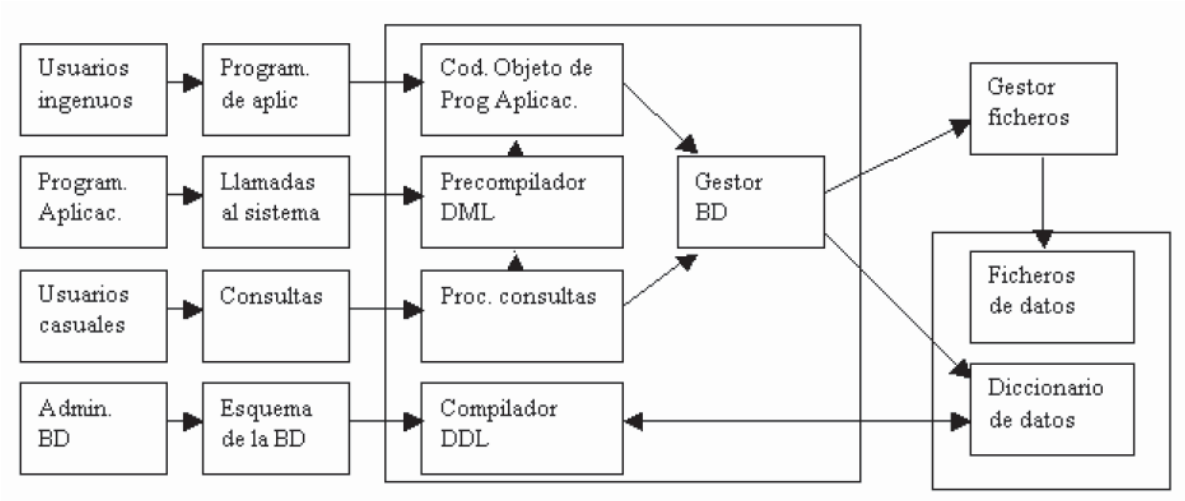
ESTRUCTURA FUNCIONAL DEL SISTEMA GESTOR DE BASES DE DATOS

3.1 ESTRUCTURA GENERAL DE LOS SISTEMAS GESTORES DE BASE DE DATOS

Como regla general, todo SGBD sigue un patrón para su desarrollo que inicialmente consta de los siguientes componentes:

- *Gestor de ficheros*. Se encarga de las estructuras de los datos a la hora de almacenarlas, asignándoles el espacio en el disco.
- *Gestor de bases de datos*. Es el interfaz de los datos de bajo nivel junto con los programas de consulta.
- *Procesador de consultas*. Encargado de traducir las proposiciones del lenguaje de consultas a instrucciones comprensibles por la base de datos.
- *Compilador DDL*. Convierte las proposiciones del lenguaje de definición de datos DDL en un conjunto de tablas con metadatos.
- *Precompilador DML*. Interpreta las proposiciones del lenguaje de manipulación de datos DML.
- *Ficheros de datos*. Ficheros físicos que almacenan los datos de la base de datos.
- *Diccionario de datos*. Almacén de información referida a la estructura de la base de datos. Su uso es continuo debido al énfasis del buen diseño y a la implementación adecuada del diccionario de datos.

Inicialmente podríamos representar estos componentes en el siguiente esquema:



En el esquema anterior, a partir de los diferentes tipos de usuarios y las funciones de los mismos, se derivan los componentes del sistema gestor implicados en estas funciones.

Un usuario ingenuo normal, a través de un programa de aplicación, activa el gestor de la base de datos y el gestor de ficheros para obtener finalmente los ficheros de datos requeridos.

Un programador de aplicaciones, mediante llamadas al sistema, activa el precompilador del lenguaje de manipulación de datos DML para interpretrar las proposiciones de este lenguaje y comunicarse con el gestor de la base de datos y el gestor de ficheros para obtener finalmente los ficheros de datos requeridos.

Un usuario casual más sofisticado, mediante la ejecución de consultas, activa el procesador de consultas que se comunica con el gestor de la base de datos y el gestor de ficheros para obtener finalmente los ficheros de datos requeridos.

Un administrador de la base de datos define el esquema conceptual de la base de datos y a través del compilador del lenguaje de definición de datos DDL convierte las proposiciones de este lenguaje (que procesa las sentencias adecuadas al diseño) en un conjunto de tablas de datos.

Toda la actividad de los procesos se va registrando constantemente en el diccionario de datos.

3.2 COMPONENTES Y FUNCIONES DE LOS SISTEMAS GESTORES DE BASES DE DATOS

Partiendo de las funciones de los sistemas gestores de bases de datos podemos completar los componentes del esquema visto en el párrafo anterior, precisamente asignando cada componente a las citadas funciones de los SGBD.

El objetivo de un SGBD es proporcionar un interfaz adecuado y eficaz para que podamos manipular la información que deseemos en nuestra base de datos, ya sea almacenarla o acceder a ella. Debe permitir a los usuarios el poder crear y mantener la base de datos a la vez que controla el acceso a la misma.

Las funciones más importantes de un SGBD deben ser:

- *Definición de los datos*: el sistema debe entender la definición de los datos que le proporcionamos, escrito en el lenguaje adecuado para que los pueda entender, y procesarlos. Este mismo sistema tiene que tener los elementos necesarios y ser capaz de procesar los datos, para cada uno de los diferentes lenguajes de definición de datos (DDL).

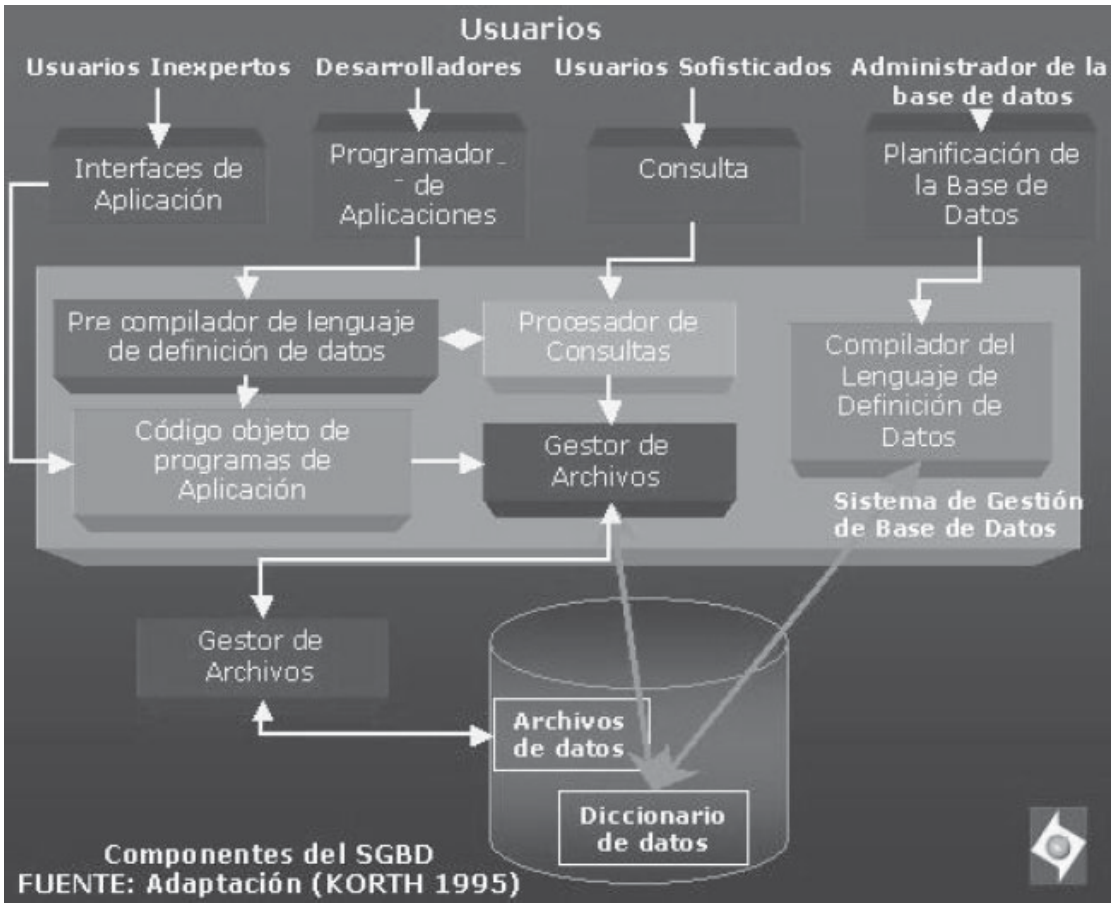
- *Manipulación de los datos*: el sistema tiene que poder encargarse de realizar las diferentes órdenes que introduzca el usuario, bien sea para acceder, guardar, borrar o editar datos. Para dicha manipulación, es necesario que el SGBD tenga incluidos lenguajes de manipulación de datos (DML). Dichos lenguajes pueden ser independientes si ellos mismos son capaces de traducir las instrucciones que le indiquemos sin necesidad de requerir un programa que le haga de traductor previamente.

- *Seguridad e integridad en los datos*: para que un SGBD sea estable y asegure la integridad de los datos, debe controlar el acceso a los mismos por parte de los usuarios, y controlar cualquier posible violación de las reglas que el administrador de la base de datos habrá establecido anteriormente.

- *Recuperación y concurrencia de los datos*: los datos almacenados deben permanecer coherentes tras la manipulación de los mismos por parte de los usuarios, también se debe controlar el acceso a los mismos cuando varios usuarios se disponen a acceder a la vez a la base de datos. Además se tendrá que tener en cuenta y poner remedio a posibles fallos, bien sea por instrucciones no realizadas completamente, fallos de energía, errores de software o posibles problemas de hardware.

- *Diccionario de datos*: la última de las funciones más importantes de un SGBD consiste en incluir información adicional sobre los datos propios. Esta información se utilizará para el análisis y diseño de la base de datos.

En la figura siguiente vemos los componentes que podemos encontrar en un SGBD divididos en diferentes módulos según las funciones que hemos citado anteriormente.



- El *Procesador de consultas* convierte las instrucciones introducidas por el usuario en un lenguaje de órdenes que entiende el gestor de la base de datos.

- El *Gestor de la base de datos* proporciona un canal entre los datos guardados en la base de datos y los programas de aplicación y las consultas que se hacen en el sistema, para comunicarse entre ambos y poder realizar dichas consultas. Gestiona la seguridad e integridad de los datos.

- El *Gestor de archivos* gestiona la cantidad de espacio en la memoria del disco y de los datos usados para representar la información almacenada en disco.

- El *Precompilador del lenguaje de manipulación de datos DML* convierte las sentencias en DML integradas en un programa de aplicación en llamadas normales a procedimientos en el lenguaje principal.

- El *Compilador del lenguaje de definición de datos DDL* convierte sentencias en DDL en un conjunto de metadatos, para que después sean incorporados al diccionario de datos.

- El *Gestor del diccionario de datos* almacena los metadatos creados anteriormente en la estructura de la base de datos.

3.3 ESTRUCTURA FUNCIONAL DEL SGBD. UN ESQUEMA COMPLETO

Un sistema de gestión de bases de datos se divide en módulos que se encargan de cada una de las responsabilidades del sistema completo. Algunas de estas funciones del sistema de base de datos las pueden proporcionar el sistema operativo de la computadora. En la mayoría de los casos, los sistemas operativos de la computadora proporcionan solo los servicios más básicos y los sistemas de bases de datos deben constituirse sobre esta base. Así, el dueño de una base de datos debe incluir consideraciones de la interfaz entre el sistema de bases de datos y el sistema operativo.

Los *componentes funcionales de un sistema gestor de bases de datos* se pueden dividir a grandes rasgos en componentes de procesamiento de consultas y componentes de gestión de almacenamiento.

Los componentes de procesamiento de consultas incluyen:

- *Compilador del DML*, que traduce las instrucciones del lenguaje de manipulación de datos en lenguaje de consultas a instrucciones a bajo nivel que entiende el motor de evaluación de consultas.

- *Precompilador del DML* incorporado, que convierte las instrucciones del lenguaje de manipulación de datos incorporadas en un programa de aplicación en llamadas a procedimientos normales en el lenguaje anfitrión.

- *Intérprete del DDL*, que interpreta las instrucciones del lenguaje de definición de datos y las registra en un conjunto de tablas que contienen metadatos.

- *Motor de evaluación de consultas*, que ejecuta las instrucciones a bajo nivel generadas por el compilador del LMD.

- *Optimizador de consultas*, que determina la estrategia óptima para la ejecución de consultas.

Los *componentes de gestión de almacenamiento* proporcionan la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y envío de consultas al sistema. El gestor de almacenamiento tiene los siguientes componentes:

- *Gestor de autorización e integridad*, que comprueba que se satisfagan las ligaduras de integridad y la autorización de los usuarios para acceder a los datos.

- *Gestor de transacciones*, que asegura que la base de datos quede en un estado consistente a pesar de los fallos del sistema y que las ejecuciones de transacciones concurrentes ocurran sin conflictos.

- *Gestor de archivos*, que gestiona la reserva de espacio de almacenamiento de disco y las estructuras de datos usadas para representar la información almacenada en el disco.

- *Gestor de memoria intermedia*, que es responsable de traer los datos del disco de almacenamiento a memoria principal y decidir qué datos traer a memoria caché.

En el esquema completo de la estructura funcional de un SGBD también se necesitan diversas *estructuras de datos como parte de la implementación física del sistema*:

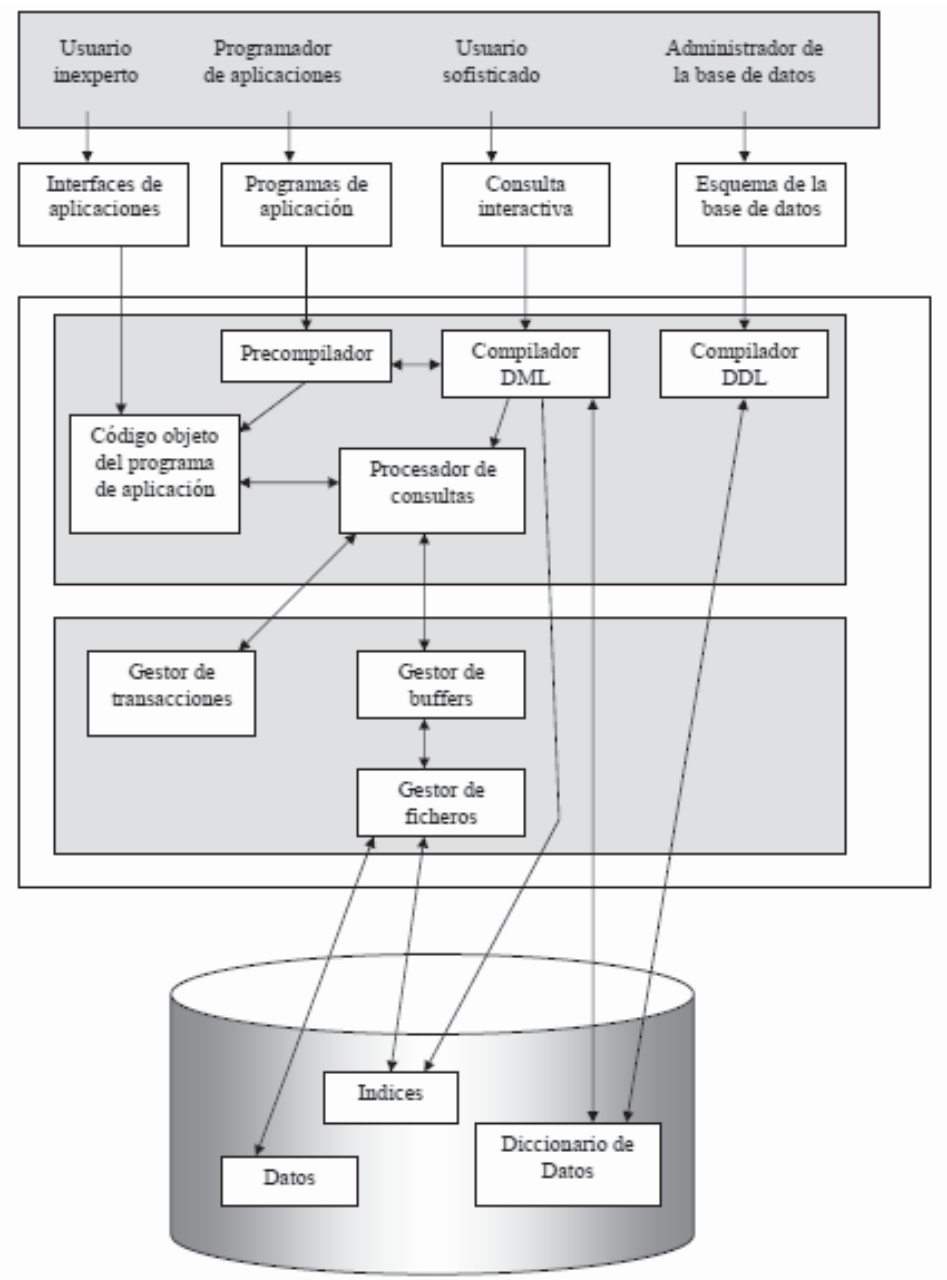
- *Archivos de datos*, que almacenan la base de datos en sí.

- *Diccionario de datos*, que almacena metadatos acerca de la estructura de la base de datos.

- *Índices*, que proporcionan acceso rápido a elementos de datos que tienen valores particulares.

- *Datos estáticos*, que almacenan información estadística sobre los datos en la base de datos, el procesador de consultas usa esta información para seleccionar las formas eficientes para ejecutar una consulta.

Si representamos la estructura funcional de un SGBD citada, tenemos la figura siguiente:



En la parte superior del esquema anterior se considera que existen cuatro *diferentes tipos de usuarios de un sistema de base de datos*, diferenciados por la forma en que ellos esperan interactuar con el sistema. Estos tipos de usuarios son:

- *Usuarios normales (finales o inexpertos)*. Son usuarios no sofisticados que, a través de *interfaces de aplicaciones*, interactúan con el sistema mediante la invocación de los programas de aplicación permanentes que se han escrito previamente. El sistema gestor de base de datos ejecuta este código a través del procesador de consultas y realiza la gestión de transacciones, buffers de intercambio entre memoria y disco o ficheros pertinente.

- *Programadores de aplicaciones*. Son profesionales informáticos que interactúan con el sistema a través de llamadas del lenguaje de manipulación de datos DML que están incluidas en un programa escrito en un lenguaje anfitrión (Cobol, Pascal, PL/I, SQL, VB, Java). Estos programas se llaman *programas de aplicación*. El sistema gestor de base de datos, a través del precompilador o compilador del lenguaje de manipulación de datos submite estos programas a través del procesador de consultas y realiza la gestión de transacciones, buffers de intercambio entre memoria y disco o ficheros que sea necesaria.

- *Usuarios sofisticados*. Interactúan con el sistema sin programas escritos. Ellos forman sus consultas en un lenguaje de consulta DML de la base de datos. Cada consulta se envía al procesador de consultas cuya función es transformar instrucciones del lenguaje de manipulación de datos a instrucciones que el gestor de almacenamiento entienda. El *gestor de almacenamiento* es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y las consultas. También podemos diferenciar un tipo especial de *usuarios especializados* que escriben aplicaciones de bases de datos especializadas que no son adecuadas en el marco de procesamiento de datos tradicional. Entre estas aplicaciones están los sistemas de diseño asistido por computadora, sistemas de bases de conocimientos y expertos y sistemas de modelado del entorno.

- *Administrador de la base de datos* (DBA): es la persona que tiene control central del sistema. Tiene las siguientes funciones principales:

  - *Definición del esquema de la base de datos*: el DBA crea el esquema original de la base de datos escribiendo un conjunto de definiciones que el compilador del lenguaje de definición de datos DDL traduce a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.

  - *Estructura de almacenamiento y definición del método de acceso*: el DBA crea las estructuras de almacenamiento apropiadas y métodos de acceso escribiendo un conjunto de definiciones, que son traducidas por el compilador del lenguaje de definición y almacenamiento de datos.

- *Esquematización y modificación de la organización física*: los programadores llevan a cabo escasas modificaciones sobre el esquema de base de datos o la descripción de la organización de almacenamiento físico, para generar las modificaciones en las tablas correspondientes del sistema interno.

- *Concesión de la automatización para el acceso a datos*: permite al administrador determinar a qué partes de la base de datos pueden acceder los diferentes usuarios.

- *Especificación de las restricciones de integridad*: los valores de los datos almacenados en la base de datos deben satisfacer ciertas ligaduras de integridad. Por ejemplo, quizás el número de horas que un empleado pueda trabajar en una semana no deba exceder de un límite especificado. Tales restricciones deben ser especificadas explícitamente por el administrador de la base de datos.

- *Gestión de conexiones y acceso en red* (*listeners*, etc.) para los SGBD en red y distribuidos.

3.4 EL SISTEMA GESTOR DE BASE DE DATOS ACCESS

Microsoft Access es actualmente un sistema gestor de base de datos relacional de los más sencillos del mercado. Implementa la mayoría de las características de un SGBD, aunque no dispone de todas ellas, como por ejemplo la posibilidad de crear y administrar vistas.

El SQL de Access se ajusta bastante bien al estándar, incorporando el lenguaje de definición de datos DDL, el lenguaje de manipulación de datos DML, el lenguaje de control de transacciones TCL y el lenguaje de control de datos DCL. También permite trabajar con procedimientos almacenados. Access es, por tanto, un sistema gestor de base de datos que se ajusta bastante bien a la estructura funcional presentada anteriormente.

Se realizará un recorrido por el sistema gestor de base de datos Access a través de la creación de una base de datos de ejemplo.

Consideramos una base de datos de nombre EMPLEADOS con tablas de nombres DEPT (departamentos de una empresa), EMP (empleados de la empresa) y SALGRADE (nivel salarial de los empleados), cuyos campos y registros se presentan a continuación. En la tabla DEPT tenemos como restricción de integridad que el campo *Deptno* es clave primaria. En la tabla EMP tenemos que el campo *Empno* es clave primaria. Además, la tabla EMP tiene como clave foránea el campo *Deptno* referenciado con el mismo nombre en la tabla DEPT.

Comenzamos presentando los campos y registros de todas las tablas del enunciado del problema.

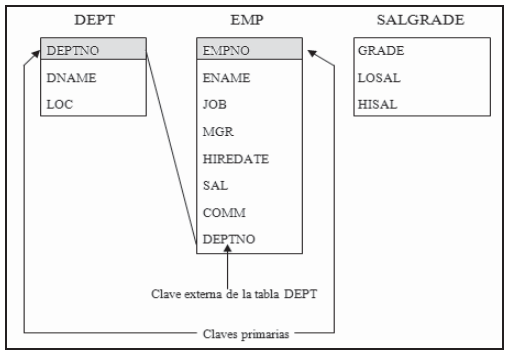
```
DEPTNO  DNAME      LOC
------- ---------- ---------
10      CONTABILIDAD    NEW YORK
20      INVESTIGACIÓN   DALLAS
30      VENTAS          CHICAGO
40      OPERACIONES     BOSTON

EMPNO  ENAME   JOB           MGR  HIREDATE    SAL     COMM  DEPTNO
-----  ------- -----------  ----- ----------  -----  ----- -------
 7369  SMITH    OFICINISTA   7902 17/12/1980    800             20
 7499  ALLEN    VENDEDOR     7698 20/02/1981   1600    300      30
 7521  WARD     VENDEDOR     7698 22/02/1981   1250    500      30
 7566  JONES    MANAGER      7839 02/04/1981   2975             20
 7654  MARTIN   VENDEDOR     7698 28/09/1981   1250   1400      30
 7698  BLAKE    MANAGER      7839 01/05/1981   2850             30
 7782  CLARK    MANAGER      7839 09/06/1981   2450             10
 7788  SCOTT    ANALISTA     7566 19/04/1987   3000             20
 7839  KING     PRESIDENTE        17/11/1981   5000             10
 7844  TURNER   VENDEDOR     7698 08/09/1981   1500             30
 7876  ADAMS    OFICINISTA   7788 23/05/1987   1100             20
 7900  JAMES    OFICINISTA   7698 03/11/1981    950             30
 7902  FORD     ANALISTA     7566 03/12/1981   3000             20
 7934  MILLER   OFICINISTA   7782 23/01/1982   1300             10

GRADE LOSAL HISAL
----- ----- -----
1     700   1200
2     1201  1400
3     1401  2000
4     2001  3000


5   3001 9999
```

En la tabla DEPT (departamentos) el campo *Deptno* se refiere al número de departamento y es numérico de longitud 2, el campo *Dname* se refiere al nombre del departamento y es texto de longitud 14 y el campo *Loc* se refiere a la localización física del departamento siendo de tipo texto con longitud 13.

En la tabla EMP (empleados) el campo *Empno* se refiere al número del empleado y es numérico de longitud 4, el campo *Ename* se refiere al nombre del empleado y es texto de longitud 10, el campo *Job* se refiere al trabajo del empleado y es texto de longitud 9, el campo *Mgr* se refiere al código del oficio y es numérico de longitud 4, el campo *Hiredate* se refiere a la fecha del contrato y es de tipo fecha, el campo *Sal* se refiere al salario del empleado y es numérico de longitud 4, el campo *Comm* se refiere a las comisiones de los empleados y es numérico de longitud 4 y el campo *Deptno* se refiere al departamento del empleado siendo numérico de longitud 2.

En la tabla SALGRADE (nivel salarial del empleado) el campo *Grade* se refiere al nivel de salario y es numérico de longitud unitaria, el campo *Losal* se refiere al salario mínimo y es numérico de longitud 4 y el campo *Hisal* se refiere al salario máximo siendo numérico de longitud 4.

En la figura siguiente se presenta el diseño conceptual de la base de datos relacional EMPLEADOS (tarea del administrador de la base de datos).



3.4.1 Interfaz de usuario del sistema gestor de base de datos Access

A continuación se crea la base de datos EMPLEADOS a través de la interfaz de Microsoft Access.

Para *crear una nueva base de datos en blanco*, inicie Access y en el botón *Archivo* seleccione *Nuevo*. Bajo *Plantillas disponibles*, haga clic en *Base de datos en blanco*. A continuación, en la parte inferior del panel *Base de datos en blanco*, en el cuadro *Nombre de archivo*, escriba un nombre de archivo (Figura 3.1). Por último, haga clic en *Crear*. Se crea la base de datos y se abre una tabla en la vista *Hoja de datos* (Figura 3.2) lo que permite crear ya las tablas.

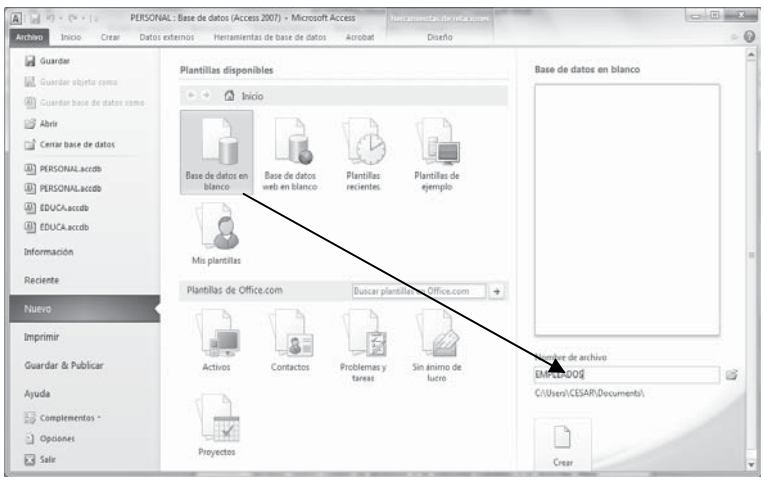


Figura 3.1

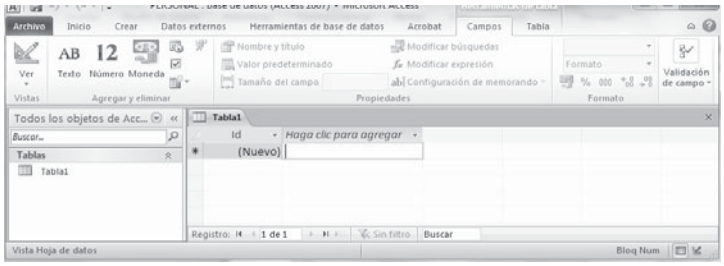


Figura 3.2

Es posible agregar inmediatamente campos a la tabla haciendo clic sobre *Haga clic para agregar* en las Figura 3.2 y escribiendo su nombre. Inmediatamente *Haga clic para agregar* se desplaza a la siguiente columna a la derecha. Para introducir un nuevo campo en la tabla se vuelve a hacer clic en *Agregar nuevo campo* y se escribe su nombre y así sucesivamente se pueden ir añadiendo más campos hasta completar todos los campos de la tabla.

La siguiente tarea será introducir los datos en los campos como si se tratase de una hoja de cálculo. De momento los tipos de datos y sus propiedades los asigna Access por defecto de acuerdo a la información que vamos almacenando en la tabla. Finalizada la tabla, se guarda a través de la opción *Guardar* del menú emergente que se obtiene al hacer clic con el botón derecho del ratón sobre la ficha de la tabla y se le asigna el nombre DEPT. La Figura 3.3 muestra la tabla con datos.

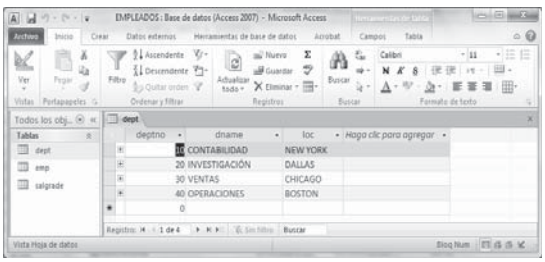


Figura 3.3

A continuación, para crear nuevas tablas en la base de datos, se hace clic en la opción *Tabla* del grupo *Tablas* de la ficha *Crear* (Figura 3.4) y se crea una nueva tabla vacía. Siguiendo los pasos anteriores se crea la tabla EMP y se introducen sus datos (Figura 3.5). Del mismo modo se crea la tabla SALGRADE y se introducen sus datos (Figuras 3.6).

Figura 3.4

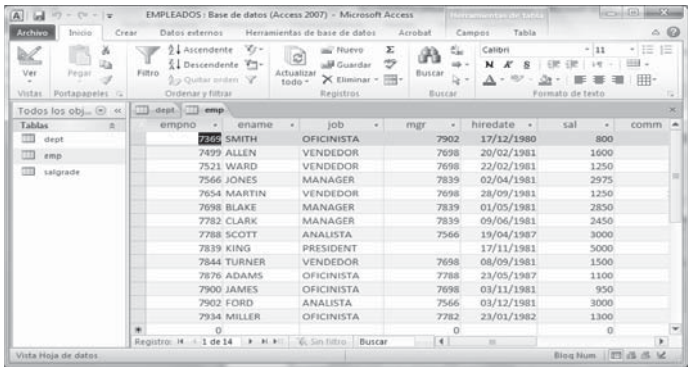


Figura 3.5

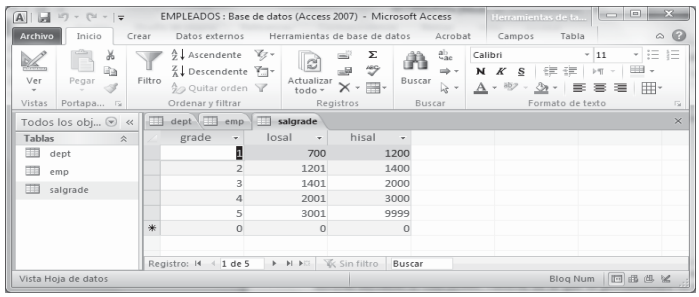


Figura 3.6

En cuanto a tipos de datos, formatos y propiedades de los campos, Access ha elegido por defecto los más adecuados al introducir los datos. Pero podemos cambiarlos a medida utilizando la vista *Diseño*, cuestión en la que se profundizará en el capítulo siguiente. La siguiente tarea es construir las relaciones entre las tablas de la base de datos. Para ello cierre todas las tablas que estén abiertas. No es posible crear ni modificar relaciones entre tablas abiertas. Desde la vista *Hoja de datos*, en la ficha *Herramientas de base de datos*, en el grupo *Relaciones*, haga clic en *Relaciones*. Se obtiene la pantalla *Relaciones* y la barra *Herramientas de relaciones* aparece en la parte superior de la pantalla con la ficha *Diseño* abierta. Si la base de datos no tiene ninguna relación definida, se mostrará automáticamente el cuadro de diálogo *Mostrar tabla*. Si necesita agregar las tablas que desea relacionar y no aparece el cuadro de diálogo *Mostrar tabla*, haga clic en *Mostrar tabla* en el grupo *Relaciones* de la ficha *Diseño*.

Haga doble clic en los nombres de las tablas que desea relacionar y, a continuación, cierre el cuadro de diálogo *Mostrar tabla*. Arrastre las tablas hasta conseguir una estructura similar a la del diagrama de diseño de la base de datos (Figura 3.7). Para arrastrar una tabla se hace clic sobre su cabecera con el ratón y, sin soltarlo, se mueve el ratón hasta la posición en que se quiere situar la tabla. Una buena estructura acorde al diseño de la base de datos se presenta en la Figura 3-7.

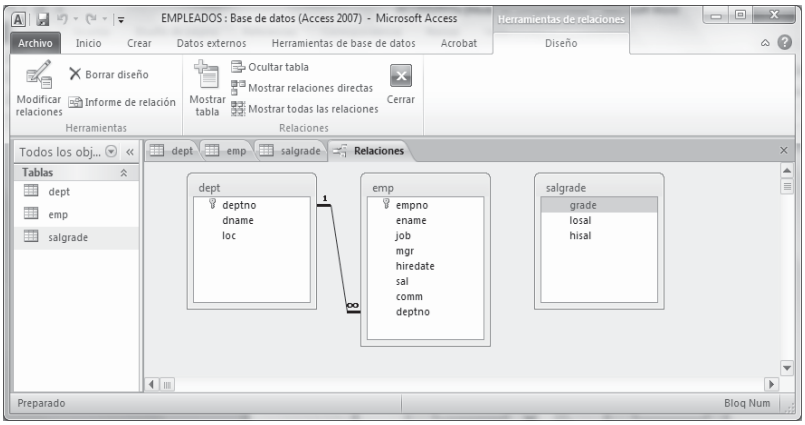


Figura 3.7

La tarea siguiente será crear las tablas de la base de datos mediante las sentencias del lenguaje de definición de datos DDL de Access.

3.4.2 Lenguaje de definición de datos DDL de Access

Las sentencias SQL de definición de datos son utilizadas para crear (CREATE), alterar (ALTER) o borrar (DROP) objetos de base de datos, tales como tablas y columnas.

Sentencia CREATE TABLE

Esta sentencia se utiliza para crear nuevas tablas en una base de datos existente. Su sintaxis es la siguiente:

```
CREATE [TEMPORARY] TABLE tabla (
campo1 tipo [(tamaño)] [NOT NULL] [WITH COMPRESSION] [índice1] [,
campo2 tipo [(tamaño)] [NOT NULL] [índice2] [, ...]] [,
CONSTRAINT índicedevarioscampos [, ...]])
```

Donde *tabla* es el nombre de la tabla que se va a crear. Los argumentos *campo1*, *campo2…* son los nombres de los campos que se van a crear en la nueva tabla. Debe crear al menos un campo. El argumento *tipo* define el tipo de datos de *campo* de la nueva tabla. El argumento *tamaño* define el tamaño del campo en caracteres (solo en campos de texto y campos binarios). Los argumentos *índice1*, *índice2…* son índices simples que habitualmente pueden definirse con la cláusula CONSTRAINT que define un índice de un solo campo o de varios campos como en el caso del argumento *índicedevarioscampos*, así como otras

restricciones. Los corchetes indican que las opciones son optativas y no aparecen en la sintaxis real.

Si se especifica NOT NULL en un campo, se requiere que los nuevos registros tengan datos válidos en ese campo. Una cláusula CONSTRAINT establece varias restricciones en un campo y se puede utilizar para establecer la clave principal, que es uno o más campos (columnas) cuyos valores identifican de manera exclusiva cada registro de una tabla. Una clave principal no puede permitir valores nulos y debe tener siempre un índice exclusivo. Una clave principal se utiliza para relacionar una tabla con claves externas de otras tablas. También puede utilizar la instrucción CREATE INDEX para crear una clave principal o índices adicionales en tablas existentes.

El atributo TEMPORARY para una tabla indica que ésta es visible solo en la sesión en la cual se ha creado y se elimina automáticamente cuando se termina la sesión. A las tablas temporales pueden obtener acceso varios usuarios. El atributo WITH COMPRESSION puede utilizarse solo con los tipos de datos CHARACTER y MEMO (conocidos también como texto) y sus sinónimos. Si se define una columna CHARACTER con este atributo, los datos se comprimirán automáticamente cuando se almacenen y se descomprimirán cuando se recuperen de la columna.

La cláusula CONSTRAINT se utiliza en las instrucciones CREATE TABLE y ALTER TABLE para crear o eliminar restricciones. Hay dos tipos de cláusulas CONSTRAINT, una para crear una restricción en un único campo y otra para crear una restricción en más de un campo. Sus sintaxis habituales son las siguientes:

- Restricción de un único campo:

```
CONSTRAINT nombre {PRIMARY KEY | UNIQUE | NOT NULL |
REFERENCES tablaexterna [(campoexterno1, campoexterno2)]
[ON UPDATE CASCADE | SET NULL]
[ON DELETE CASCADE | SET NULL]}
```

- Restricción de varios campos:

```
CONSTRAINT nombre
{PRIMARY KEY (principal1[, principal2 [, ...]]) |
UNIQUE (única1[, única2 [, ...]]) |
NOT NULL (nonulo1[, nonulo2 [, ...]]) |
FOREIGN KEY [NO INDEX] (ref1[, ref2 [, ...]]) REFERENCES tablaexterna [(campoexterno1
[, campoexterno2 [, ...]])]
[ON UPDATE CASCADE | SET NULL]
[ON DELETE CASCADE | SET NULL]}
```

El argumento *nombre* indica el nombre de la restricción que se va a crear. Los argumentos *principal1*, *principal2…* indican el nombre del campo o campos que se van a designar como clave principal. Los argumentos *única1*, *única2…* indican el nombre del campo o campos que se van a designar como clave única. Los

argumentos *nonulo1, nonulo2…* indican el nombre del campo o campos que se van a restringir a valores no nulos. Los argumentos *ref1*, *ref2…* indican el nombre del campo o campos de clave externa que hacen referencia a campos de otra tabla. El argumento *tablaexterna* indica el nombre de la tabla externa que contiene el campo o campos especificados en *campoexterno*. Los argumentos *campoexterno1*, *campoexterno2…* indican el nombre del campo o campos de *tablaexterna* especificados en *ref1*, *ref2…* Puede omitir esta cláusula si el campo al que se hace referencia es la clave principal de *tablaexterna.*

La cláusula CONSTRAINT tiene como finalidad esencial definir las *restricciones de integridad*.

Para diseñar las tablas, es necesario identificar los valores válidos para cada columna y decidir cómo se debe exigir la integridad de los datos de la columna. SQL proporciona varios mecanismos para exigir la integridad de los datos de una columna, entre los que destacan restricciones PRIMARY KEY, restricciones FOREIGN KEY, restricciones UNIQUE y permitir o no valores NULL.

Se denomina *clave principal o clave primaria* de una tabla a una columna o una combinación de columnas cuyos valores identifican de forma única cada fila de la tabla (valores únicos que no pueden repetirse). Una clave principal se crea mediante la definición de una restricción PRIMARY KEY cuando cree o modifique una tabla. Una tabla solo puede tener una restricción PRIMARY KEY, y ninguna columna a la que se aplique una restricción PRIMARY KEY puede aceptar valores NULL. Si especifica una restricción PRIMARY KEY en una tabla, SQL exige la exclusividad de los datos mediante la creación de un índice único para las columnas de clave principal. Este índice también permite un acceso rápido a los datos cuando se utiliza la clave principal en las consultas. Si se define una restricción PRIMARY KEY para más de una columna, puede haber valores duplicados dentro de la misma columna pero cada combinación de valores de todas las columnas de la definición de la restricción PRIMARY KEY debe ser única.

Una *clave externa* o *clave foránea* (FK) es una columna o combinación de columnas que se utiliza para establecer y exigir un vínculo entre los datos de dos tablas. Se crea un vínculo entre dos tablas al agregar en una tabla la columna o columnas que contienen los valores de la clave principal de la otra tabla. Esta columna se convierte en una clave externa para la segunda tabla. Puede crear una clave externa mediante la definición de una restricción FOREIGN KEY cuando cree o modifique una tabla. No es necesario que una restricción FOREIGN KEY esté vinculada únicamente a una restricción PRIMARY KEY de otra tabla; también puede definirse para que haga referencia a las columnas de una restricción UNIQUE de otra tabla.

Una restricción FOREIGN KEY puede contener valores NULL, pero si alguna columna de una restricción FOREIGN KEY compuesta contiene valores NULL, se omitirá la comprobación de la restricción FOREIGN KEY. Una restricción FOREIGN KEY puede hacer referencia a columnas de tablas de la misma base de datos o a columnas de una misma tabla (tablas con referencia a sí mismas). Aunque el propósito principal de una restricción FOREIGN KEY es el

de controlar los datos que pueden almacenarse en la tabla de la clave externa, también controla los cambios realizados en los datos de la tabla de la clave principal y asegura que no se puedan realizar cambios en los datos de la tabla de la clave principal si esos cambios anulan el vínculo con los datos de la tabla de la clave externa. Si se intenta eliminar la fila de una tabla de la clave principal o cambiar un valor de clave principal, la acción no progresará si el valor de la clave principal cambiado o eliminado corresponde a un valor de la restricción FOREIGN KEY de otra tabla. Para cambiar o eliminar una fila de una restricción FOREIGN KEY, antes debe eliminar los datos de la clave externa de la tabla de la clave externa o bien cambiar los datos de la clave externa en la tabla de la clave externa y vincular, de ese modo, la clave externa con otros datos de clave principal.

Las *restricciones de integridad referencial en cascada* permiten definir las acciones que SQL lleva a cabo cuando un usuario intenta eliminar o actualizar una clave a la que apuntan las claves externas existentes. Las cláusulas REFERENCES de las instrucciones CREATE TABLE y ALTER TABLE admiten cláusulas ON DELETE CASCADE, ON UPDATE CASCADE y ON UPDATE SET NULL. CASCADE permite la eliminación o actualización de valores de clave en cascada en las tablas definidas para tener relaciones de claves externas que pueden volver a trazarse en la tabla en la que se realizan las modificaciones.

- **ON DELETE CASCADE** especifica que si se intenta eliminar una fila con una clave a la que hacen referencia claves externas de filas existentes en otras tablas, todas las filas que contienen dichas claves externas también se eliminan. Si las acciones referenciales en cascada se han definido también en las tablas de destino, las acciones en cascada especificadas se toman también para las filas eliminadas de dichas tablas.

- **ON UPDATE CASCADE** especifica que si se intenta actualizar una fila con una clave a la que hacen referencia claves externas de filas existentes en otras tablas, todas las filas que contienen dichas claves externas también se actualizan.

- **ON UPDATE SET NULL** especifica que si se intenta actualizar una fila con una clave a la que hacen referencia claves externas de filas existentes en otras tablas, todas las filas que contienen dichas claves externas se configuran automáticamente con el valor NULL.

Se utiliza el modificador NO INDEX para que no se lleve a cabo la creación automática de índices para las claves externas.

También pueden utilizarse *restricciones UNIQUE* para asegurar que no se escriban valores duplicados en columnas específicas que no formen parte de una clave principal. Aunque tanto una restricción UNIQUE como PRIMARY KEY exigen que los elementos sean únicos, es preferible utilizar una restricción UNIQUE en vez de una restricción PRIMARY KEY cuando desee exigir la unicidad de una columna o una combinación de columnas que no sea la clave principal, teniendo presente que en una tabla se puede definir varias restricciones UNIQUE, pero solo una restricción PRIMARY KEY.

También es muy importante que las restricciones UNIQUE puedan definirse en columnas que aceptan valores NULL, mientras que las restricciones PRIMARY KEY solo pueden definirse en columnas que no aceptan valores NULL. También es posible hacer referencia a una restricción UNIQUE con una restricción FOREIGN KEY.

La aceptación de *valores NULL* de una columna determina si las filas de una tabla pueden contener un valor NULL en esa columna. Un valor NULL es distinto de cero (0), de blanco o de una cadena de caracteres de longitud cero, como ""; NULL significa que no hay ninguna entrada. Su presencia suele implicar que el valor es desconocido o no definido. En general, evite la aceptación de valores NULL dado que pueden implicar una mayor complejidad en las consultas y en las actualizaciones; por otra parte, hay otras opciones para las columnas, como las restricciones PRIMARY KEY, que no pueden utilizarse con columnas que aceptan valores NULL.

Ha llegado el momento de crear las tablas de la base de datos de ejemplo de nombre EMPLEADOS, de acuerdo al diseño conceptual visto anteriormente, mediante la sentencia CREATE TABLE del lenguaje DDL de Access.

Las Figuras 3.8 a 3.10 presentan la sintaxis SQL de creación de las tablas de la base de datos EMPLEADOS. Dicha sintaxis se escribe en el editor de la *Vista SQL* de Access que ya hemos utilizado en el capítulo anterior.

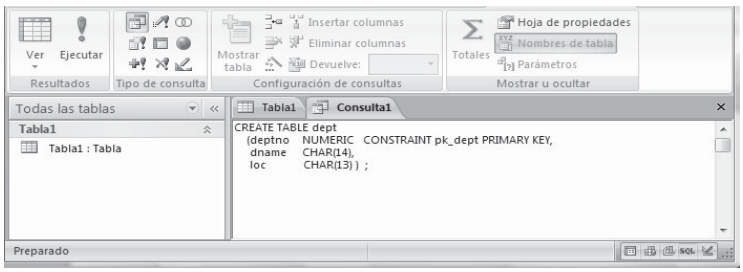


Figura 3.8

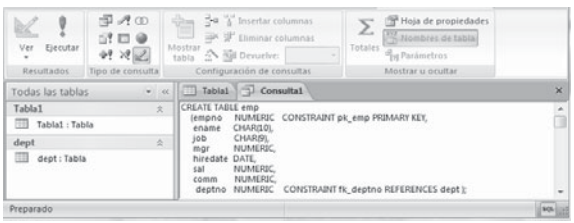


Figura 3.9

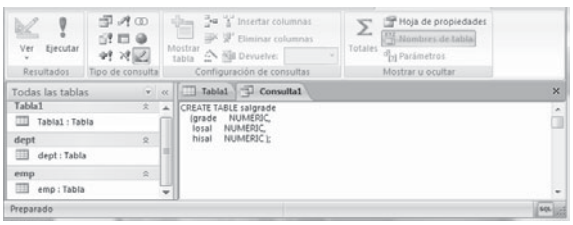


Figura 3.10

Adicionalmente crearemos algunas tablas auxiliares en la tabla PERSONAL para utilizar como ejemplos en las sentencias de los apartados siguientes. En primer lugar creamos la tabla EMPLEO (Figura 3.11) con los campos NOMBRE (carácter de longitud 25 sin admitir nulos), EDAD

(numérico) y ALOJAMIENTO (carácter de longitud 15) siendo la clave primaria (EDAD, ALOJAMIENTO).

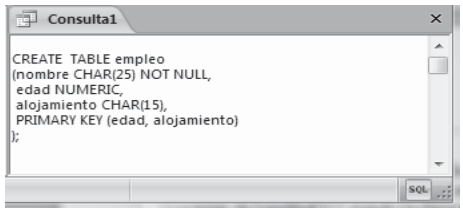


Figura 3.11

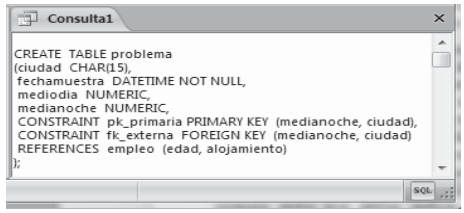


Figura 3.12

En segundo lugar creamos la tabla PROBLEMA (figura 3.12) con los campos CIUDAD (carácter de tamaño 15), FECHAMUESTRA (tipo fecha sin admitir valores nulos), MEDIODÍA (numérico) y MEDIANOCHE (numérico) utilizando como clave externa de la clave primaria (MEDIANOCHE, CIUDAD) las columnas (EDAD, ALOJAMIENTO) de la tabla EMPLEO creada en el ejemplo anterior. Usamos la cláusula CONSTRAINT para nombrar todas las restricciones, incluida la clave primaria.

A continuación creamos la tabla CLIENTES con los campos IDCLIENTE (clave primaria con valores enteros) y APELLIDOCLIENTE (carácter de longitud 50) según la Figura 3.13. Construimos también la tabla PEDIDOS (Figura 3.14) con los campos IDPEDIDO (clave primaria con valores enteros), IDCLIENTE (valores enteros) y NOTASPEDIDO (carácter de longitud 225), de modo que la clave externa de IDCLIENTE sea el campo del mismo nombre en la tabla CLIENTES. Se exige, además, que cuando se elimine un cliente de la tabla CLIENTES, también se eliminen todas las filas de la tabla PEDIDOS que contienen el mismo valor para el identificador de cliente. ¿Cómo sería la sintaxis para que al eliminar un cliente de la tabla CLIENTES, todas las claves externas correspondientes de la tabla PEDIDOS se configuren con el valor NULL? En la Figura 3.15 se resuelve esta última cuestión.

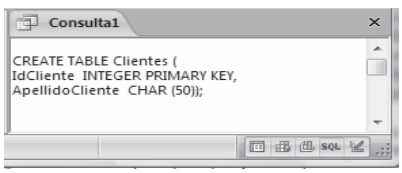


Figura 3.13

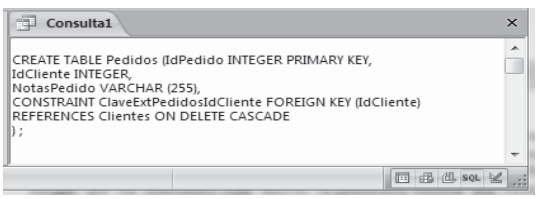


Figura 3.14

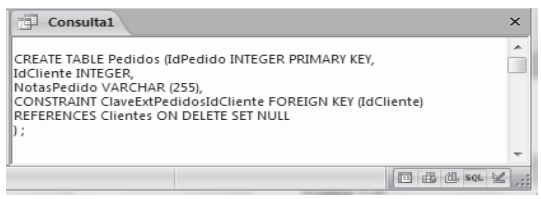


Figura 3.15

Sentencia CREATE INDEX

En Access SQL se crean índices con la sentencia CREATE INDEX. Un índice es una estructura que proporciona un acceso rápido a las filas de una tabla en base a los valores de una o más columnas. Los índices de las bases de datos son similares a los índices que hay en los libros. En un libro, un índice permite encontrar información rápidamente sin necesidad de leer todo el libro. En una base de datos, un índice permite que el programa de la base de datos busque datos en una tabla sin necesidad de examinar toda la tabla. El índice de un libro es una lista de palabras con los números de las páginas en las que se encuentra cada palabra. Un índice de una base de datos es una lista de los valores de una tabla con las posiciones de almacenamiento de las filas de la tabla donde se encuentra cada valor. Se pueden crear índices en una sola columna o en una combinación de columnas de una tabla.

La sintaxis de CREATE INDEX es la siguiente:

```
CREATE [ UNIQUE ] INDEX índice
ON tabla (campo [ASC|DESC][, campo [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]
```

En la sintaxis anterior *índice* especifica el nombre del índice que se va a crear, *tabla* especifica el nombre de la tabla existente que va a contener el índice y *campo* especifica el nombre del campo o campos que se van a indizar. Para crear un índice de un único campo, incluya el nombre del campo entre paréntesis a continuación del nombre de tabla. Para crear un índice de varios campos, utilice la palabra reservada DESC, ya que, de lo contrario, se supone que los índices están en orden ascendente. La palabra reservada UNIQUE se utiliza para impedir que aparezcan valores duplicados en el campo o campos indizados de registros diferentes.

La cláusula opcional WITH permite aplicar reglas de validación de datos. La palabra reservada PRIMARY designa el campo o campos indizados como clave principal, lo que implica que la clave será única y no será necesario utilizar la palabra reservada UNIQUE (no utilice la palabra reservada PRIMARY cuando cree un índice nuevo en una tabla que ya tiene una clave principal). Mediante la opción DISALLOW NULL se impiden entradas NULL en el campo o campos indizados de los nuevos registros. Mediante la opción IGNORE NULL se evita que se incluyan en el índice registros con valores NULL en el campo o campos indizados.

Como ejemplo, en la Figura 3.16 se crea un índice de nombre INDICE3 en el campo NOMBRE de la tabla EMPLEO de la base de datos PRUEBA.

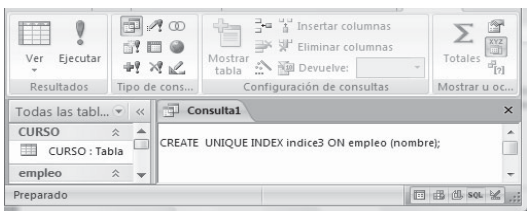


Figura 3.16

Sentencia ALTER TABLE

En Access SQL se modifican tablas con la sentencia ALTER TABLE. Es posible agregar, modificar o eliminar columnas. Se puede cambiar el nombre, la longitud, el tipo de datos, la precisión, la escala y la aceptación de valores NULL de la columna, aunque hay algunas restricciones. Esta sentencia permite agregar o eliminar restricciones PRIMARY KEY y FOREIGN KEY y agregar o eliminar restricciones UNIQUE y CHECK, así como definiciones (y objetos) DEFAULT. También permite registrar una tabla y las columnas seleccionadas para la indización de texto.

La sintaxis de ALTER TABLE es la siguiente:

```
ALTER TABLE tabla {ADD {COLUMN tipo de campo[(tamaño)] [NOT NULL] [CONSTRAINT
índice] |
ALTER COLUMN tipo de campo[(tamaño)] |
CONSTRAINT índicedevarioscampos} |
DROP {COLUMN campo I CONSTRAINT nombredeíndice} }
```

En la sintaxis anterior *tabla* es el nombre de la tabla que se va a modificar, *campo* especifica el *nombre* del campo que se va a agregar o a eliminar de la tabla (o bien el nombre del campo que se va a modificar en tabla), *tipo* especifica el tipo de datos de *campo*, *tamaño* especifica el tamaño del campo en caracteres (solo en campos de texto y campos binarios), *índice* especifica el índice de *campo*, *índicedevarioscampos* especifica la definición de un índice de varios campos que se va a agregar a latabla y *nombredeíndice* especifica el nombre del índice de varios campos que se va a eliminar. ADD COLUMN permite agregar un campo nuevo a la tabla especificando el nombre del campo, el tipo de datos y un tamaño opcional (para campos de tipo *Text* y *Binary*). También puede definirse un índice en ese campo con la cláusula CONSTRAINT. Si se especifica NOT NULL para un campo, se exigirá que los registros nuevos tengan datos válidos en ese campo. Como ejemplo, en la Figura 3.17 se agrega un campo de tipo *Text* de 25 caracteres llamado *Notas* a la tabla *Empleo* de la base de datos PRUEBA. ALTER COLUMN permite cambiar el tipo de datos de un campo existente especificando el nombre del campo, el nuevo tipo de datos y un tamaño opcional para campos de tipo *Text* y *Binary*. Por ejemplo, la sintaxis de la Figura 3.18 cambia el tipo de datos del campo *Notas* de la tabla *Empleo* (definido anteriormente con texto de 25 caracteres) a un campo de tipo texto de 100 caracteres. ADD CONSTRAINT permite agregar restricciones tal y como hemos visto en CREATE TABLE (por ejemplo en la Figura 3.19 se crea la tabla SALGRADE y en la Figura 3.20 se le añade una clave primaria al campo *grade*). DROP COLUMN permite eliminar un campo especificando solamente el nombre del campo. DROP CONSTRAINT permite eliminar un índice de múltiples campos especificando solamente el nombre del índice a continuación de la palabra reservada CONSTRAINT. En la Figura 3.21 se crea la tabla T11 con una restricción de clave externa, en la Figura 3.22 se elimina la restricción y en la Figura 3.23 se elimina la columna *c1* de la tabla.

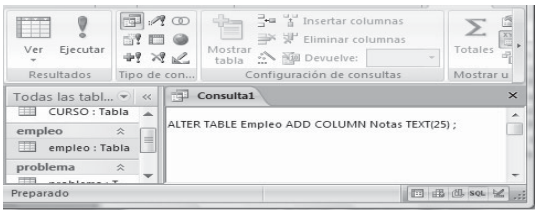


Figura 3.17

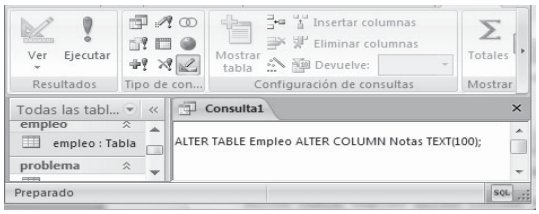


Figura 3.18

Figura 3.19

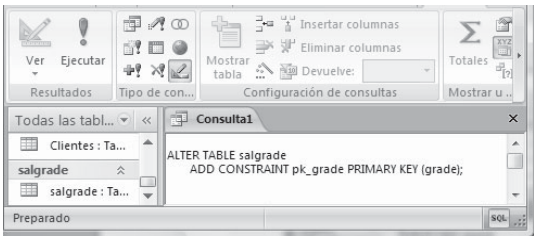


Figura 3.20

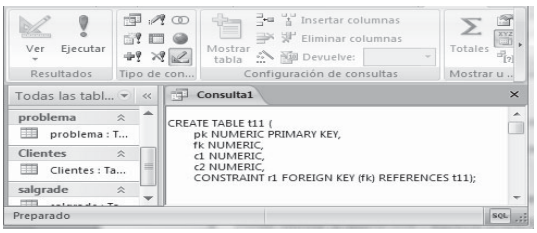


Figura 3.21

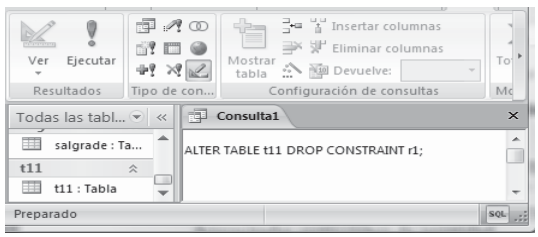


Figura 3.22

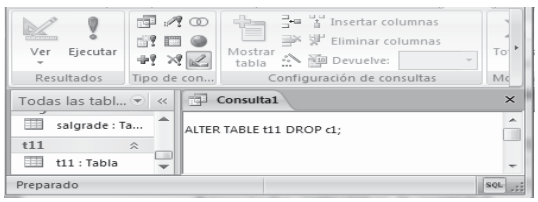


Figura 3.23

Hay que tener presente que no se puede agregar o eliminar más de un campo o índice a la vez. Por otro lado, es posible utilizar la instrucción CREATE INDEX para agregar un índice de uno o múltiples campos a una tabla y puede utilizarse la instrucción ALTER TABLE o DROP para eliminar un índice creado con la instrucción ALTER TABLE o CREATE INDEX. También es posible utilizar NOT NULL en un único campo o dentro de una cláusula CONSTRAINT con nombre que se aplique a CONSTRAINT de uno o varios campos con nombre. Sin embargo, solo se puede aplicar la restricción NOT NULL una vez a un campo. Si se intenta aplicar esta restricción más de una vez, se producirá un error en tiempo de ejecución.

Sentencia DROP

En Access SQL se borran tablas con la sentencia DROP TABLE. En un sentido más amplio, la sentencia DROP se utiliza para borrar tablas, índices, procedimientos y vistas. Su sintaxis es la siguiente:

```
DROP {TABLE tabla | INDEX índice ON tabla | PROCEDURE procedimiento | VIEW vista}
```

En la sintaxis anterior *tabla* es el nombre de la tabla que va a ser eliminada o de la cual va a ser eliminado un índice, *procedimiento* es el nombre del procedimiento que se va a eliminar, *vista* es el nombre de la vista que se va a eliminar e *índice* es el nombre del índice que se va a eliminar de la tabla. La sentencia DROP TABLE permite borrar tablas completas de la base de datos. Esta sentencia suprime las tablas que se especifican en su sintaxis y valida los cambios pendientes en la base de datos. Únicamente un administrador de la base de datos puede suprimir tablas de otros usuarios. Al suprimir una tabla también se suprimen los índices y las conexiones asociadas a ella.

Para eliminar un índice de una tabla debe cerrar la tabla previamente. Puede usar CREATE TABLE para crear una tabla y CREATE INDEX o ALTER TABLE para crear o eliminar un índice. Para modificar una tabla se usa ALTER TABLE.

Como ejemplos, en la Figura 3.24 se crea en la tabla *t*11 sobre el campo *c2* el índice de nombre *t11_id*. En la Figura 3.25 se borra dicho índice y en la Figura 3.26 se borra la propia tabla *t*11.

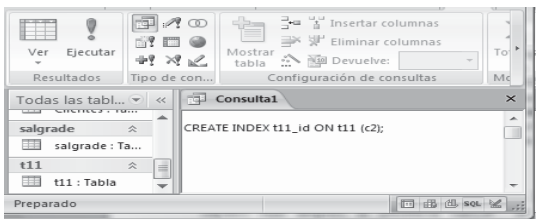


Figura 3.24

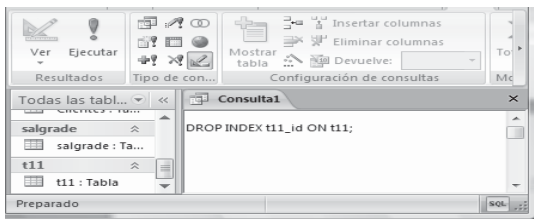


Figura 3.25

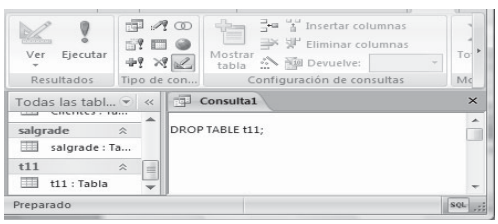


Figura 3.26

3.4.3 Lenguaje de modificación de datos DML de Access

El lenguaje SQL (*Structured Query Language*) de Microsoft Access incluye la categoría de sentencias denominada lenguaje de modificación de datos DML (*Data Modification Language*). Dichas sentencias se utilizan para manipular, alterar o borrar objetos de base de datos, tales como tablas, columnas e índices. Las sentencias más importantes de este grupo son INSERT, UPDATE, DELETE y especialmente la sentencia SELECT.

Sentencia INSERT

Esta sentencia se utiliza para agregar una o más filas a una tabla. Su sintaxis simplificada es la siguiente:

```
INSERT [INTO] tabla [(lista_de columnas)] VALUES valores_de_datos
```

donde *tabla* es el nombre de la tabla en la que se van a insertar los datos, *lista_de_columnas* es una lista separada por comas de los nombres de las columnas para las que se suministran datos (si no se especifica *lista_de_columnas*, todas las columnas de la tabla recibirán datos) y *valores_de_datos* es una lista separada por comas que contiene los valores que se insertarán como una o más filas en la *tabla* que se nombra (los valores de cadena se situarán entre comillas simples). La lista de valores de datos suministrados debe corresponderse con la lista de columnas. El número de valores de datos debe ser el mismo que el número de columnas y el tipo de datos, precisión y escala de cada valor de datos deben coincidir con los de la columna correspondiente. La finalidad esencial de la lista de columnas en la sentencia INSERT es hacer corresponder los valores de datos en la cláusula VALUES con las columnas que van a recibirlos. La lista de valores y la lista de columnas deben contener el mismo número de elementos y el tipo de dato de cada valor debe ser compatible con el tipo de datos de la columna correspondiente, o en caso contrario se producirá un error.

Siguiendo con nuestro ejemplo, las Figuras 3.27 a 3.29 presentan la sintaxis SQL para la introducción del primer registro en las tablas. De modo similar se introducen todos los demás registros.

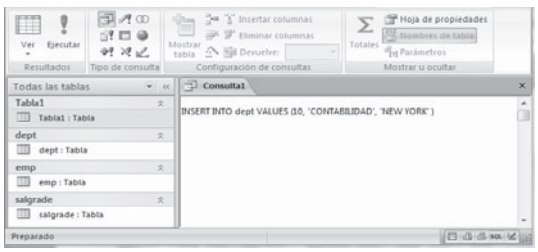


Figura 3.27

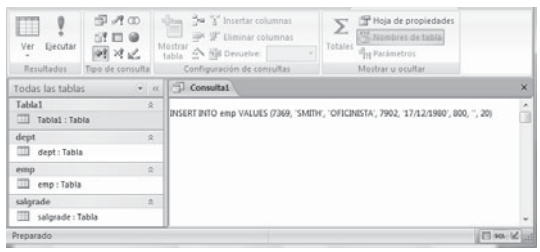


Figura 3.28

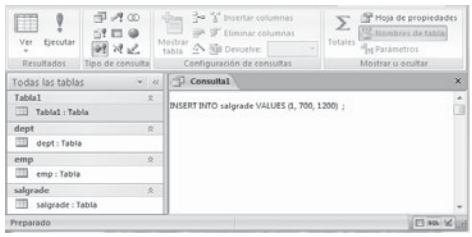


Figura 3.29

Por último, se especifica la sintaxis SQL completa para la introducción de los registros en las tablas.

```
INSERT INTO dept VALUES (10, 'CONTABILIDAD', 'NEW YORK' )
INSERT INTO dept VALUES (20, 'INVESTIGACIÓN', 'DALLAS')
INSERT INTO dept VALUES (30, 'VENTAS', 'CHICAGO')
INSERT INTO dept VALUES (40, 'OPERACIONES', 'BOSTON')
INSERT INTO emp VALUES (7369, 'SMITH', 'OFICINISTA', 7902, '17/12/1980', 800, '', 20)
INSERT INTO emp VALUES (7499, 'ALLEN', 'VENDEDOR', 7698, '20/02/1981', 1600,     300,
30)
INSERT INTO emp VALUES (7521, 'WARD', 'VENDEDOR', 7698, '22/02/1981', 1250,    500, 30)
INSERT INTO emp VALUES (7566, 'JONES', 'MANAGER', 7839, '02/04/1981', 2975, '', 20)
INSERT INTO emp VALUES (7654, 'MARTIN', 'VENDEDOR', 7698, '28/09/1981', 1250,    1400,
30)
INSERT INTO emp VALUES (7698, 'BLAKE', 'MANAGER', 7839, '01/05/1981', 2850, '', 30)
INSERT INTO emp VALUES (7782, 'CLARK', 'MANAGER', 7839, '09/06/1981', 2450, '', 10)
INSERT INTO emp VALUES (7788, 'SCOTT', 'ANALISTA', 7566, '19/04/1987', 3000, '', 20)
INSERT INTO emp VALUES (7839, 'KING', 'PRESIDENTE', '', '17/11/1981', 5000, '', 10)
INSERT INTO emp VALUES (7844, 'TURNER', 'VENDEDOR', 7698, '08/09/1981', 1500, '', 30)
INSERT INTO emp VALUES (7876, 'ADAMS', 'OFICINISTA', 7788, '23/05/1987', 1100, '', 20)
INSERT INTO emp VALUES (7900, 'JAMES', 'OFICINISTA', 7698, '03/11/1981', 950, '', 30)
INSERT INTO emp VALUES (7902, 'FORD', 'ANALISTA',  7566, '03/12/1981',  3000, '', 20)
INSERT INTO emp VALUES (7934, 'MILLER', 'OFICINISTA', 7782, '23/01/1982', 1300, '', 10)
INSERT INTO salgrade VALUES (1, 700, 1200)
INSERT INTO salgrade VALUES (2, 1201, 1400)
INSERT INTO salgrade VALUES (3, 1401, 2000)
INSERT INTO salgrade VALUES (4, 2001, 3000)
INSERT INTO salgrade VALUES (5, 3001, 9999)
```

Sentencia UPDATE

Esta sentencia se utiliza para actualizar filas en las tablas ya existentes. Su sintaxis básica es la siguiente:

```
UPDATE tabla
SET {columna = expresión [, columna = expresión]…}
[WHERE condición];
```

El argumento *tabla* indica el nombre de la tabla en la que se van a actualizar los datos, SET contiene una lista separada por comas de las columnas que deben actualizarse y el nuevo valor de cada columna con el formato *columna* = *expresión*. El valor suministrado por las expresiones incluye elementos tales como constantes, valores seleccionados de una columna de otra tabla o vista, o valores calculados por una expresión compleja. Las columnas que se igualen a expresiones deben preceder a las columnas entre paréntesis que se igualen a subconsultas, todo dentro de una única sentencia UPDATE. WHERE especifica la condición de búsqueda que define las filas de las tablas de origen que están calificadas para proporcionar valores para las expresiones de la cláusula SET. La sentencia UPDATE es muy útil para cambiar muchos registros o cuando los registros que desea cambiar están en varias tablas. Es posible cambiar varios campos al mismo tiempo. Hay que tener presente que después de actualizar registros mediante una consulta de actualización, no se puede deshacer la operación, por lo que es conveniente realizar copia de seguridad de la información antes de realizar la actualización. En la Figura 3.30 se actualiza la tabla EMP de la base de datos PRUEBA aumentando el salario en 100 unidades y la comisión un 5% para los empleados cuyo identificador está entre 6000 y 10000.

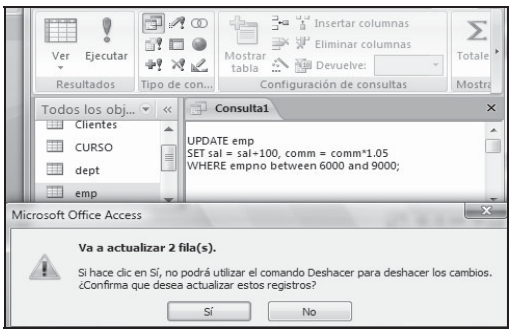


Figura 3.30

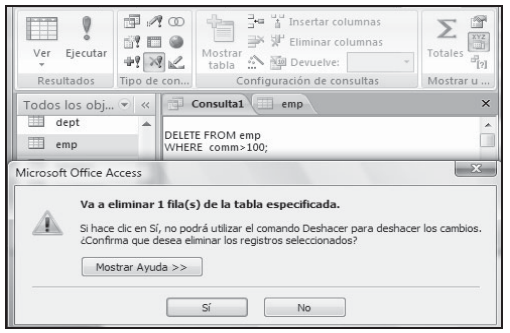


Figura 3.31

Si en la consulta anterior se omite la cláusula WHERE, se actualizarán todas las filas de la tabla destino, es decir, la actualización del salario y de la comisión se realizará para todos los empleados.

Sentencia DELETE

Esta sentencia se utiliza para eliminar filas en las tablas ya existentes. Su sintaxis básica es la siguiente:

```
DELETE FROM tabla
[WHERE condición];
```

El argumento *tabla* indica el nombre de la tabla en la que se van a eliminar las filas de datos y se eliminarán todas las filas de la tabla que cumplan los requisitos de la condición especificada en la cláusula WHERE. Cuando no se especifica WHERE se eliminarán todas las

filas de la tabla. Como ejemplo, la sintaxis de la Figura 3.31 elimina de la tabla EMP los empleados cuya comisión es mayor que 100.

La diferencia entre DROP TABLE y DELETE está en que DROP elimina la tabla incluida su estructura, mientras que DELETE borra solo determinado contenido de la tabla y conserva su estructura. Si en la Figura 3.31 no se especifica la cláusula WHERE, se borrará el contenido completo de la tabla, pero no la propia tabla.

Capítulo

4

INSTALACIÓN DE SISTEMAS GESTORES DE BASES DE DATOS

En este capítulo se explica pormenorizadamente la instalación de los sistemas gestores de bases de datos más habituales en el mercado, desde los más sencillos como Microsoft Access hasta sistemas gestores de bases de datos distribuidos como SQL Server y Oracle. Se tendrán en cuenta los parámetros de configuración de los distintos sistemas gestores de base de datos, los requisitos necesarios para el funcionamiento, la puesta en marcha y el entorno de trabajo.

4.1 INSTALACIÓN DE MICROSOFT ACCESS

Hemos de tener presente que Microsoft Access puede considerarse como una base de datos relacional a partir de la versión 2007. Una base de datos de Access permite la utilización simultánea de datos procedentes de más de una tabla. Al hacer uso de las relaciones, se evita la duplicidad de datos, ahorrando memoria y espacio en el disco, aumentando la velocidad de ejecución y facilitando al usuario/a el trabajo con tablas. Como en todos los SGBD relacionales, para conseguir en Access una base de datos relacional correcta, es imprescindible realizar un estudio previo del diseño de la base de datos. Para poder relacionar tablas entre sí se deberá especificar un campo en común que contenga el mismo valor en las dos tablas y dicho campo será clave principal en una de ellas. Las tablas pueden relacionarse de dos a dos, donde una de ellas será la tabla principal de la que parte la relación y la otra será la tabla secundaria destino de la relación. Se pueden distinguir tres tipos de relaciones:

- **Relación Uno a Uno**: cuando un registro de una tabla solo puede estar relacionado con un único registro de la otra tabla y viceversa. Por ejemplo: tenemos dos tablas una con los datos de diferentes poblaciones y otra con una

lista de alcaldes. Una población solo podrá tener un alcalde, y un alcalde lo será únicamente de una población.

- **Relación Uno a Varios**: cuando un registro de una tabla (tabla secundaria) solo puede estar relacionado con un único registro de la otra tabla (tabla principal) y un registro de la otra tabla (tabla principal) puede tener más de un registro relacionado en la primera tabla (tabla secundaria). Por ejemplo: tenemos dos tablas una con los datos de diferentes poblaciones y otra con los habitantes, una población puede tener más de un habitante, pero un habitante pertenecerá (estará empadronado) en una única población.

- **Relación Varios a Varios**: cuando un registro de una tabla puede estar relacionado con más de un registro de la otra tabla y viceversa. Por ejemplo: tenemos dos tablas una con los datos de clientes y otra con los artículos que se venden en la empresa, un cliente podrá realizar un pedido con varios artículos, y un artículo podrá ser vendido a más de un cliente. Las relaciones varios a varios se suelen representar definiendo una tabla intermedia entre las dos tablas. Siguiendo el ejemplo anterior sería definir una tabla líneas de pedido relacionada con clientes y con artículos.

Así mismo, Access utiliza la integridad referencial consistente en un sistema de reglas para asegurarse de que las relaciones entre registros de tablas relacionadas son válidas y que no se borren o cambien datos relacionados de forma accidental. Al exigir integridad referencial en una relación le estamos diciendo a Access que no nos deje introducir datos en la tabla secundaria si previamente no se han introducido en la tabla principal. Por ejemplo: siguiendo con el ejemplo anterior del tipo de relación uno a varios, no nos dejará introducir un habitante a una población si dicha ciudad no existe en la tabla de poblaciones.

La integridad referencial dispone de dos acciones:

- **Actualizar registros en cascada**: cuando se cambie el valor del campo de la tabla principal, automáticamente cambiarán los valores de sus registros relacionados en la tabla secundaria. Por ejemplo: si cambiamos el nombre de la población Onteniente por Ontinyent en la tabla de poblaciones, automáticamente todos los habitantes de Onteniente se cambiarán a Ontinyent.

- **Eliminar registros en cascada**: cuando se elimina un registro de la tabla principal se borrarán también los registros relacionados en la tabla secundaria. Por ejemplo: si borramos la población Onteniente en la tabla de poblaciones, automáticamente todos los habitantes de Onteniente se borrarán de la tabla de habitantes.

4.1.1 Requisitos para la instalación de Access

Los requisitos habituales para instalar las aplicaciones de las últimas versiones de Microsoft Office, incluido Access, se resumen en la tabla siguiente:

Componente	Requisito
Equipo y procesador	Procesador de 500 MHz; 1 GHz necesario para Outlook con Business Contact Manager.
Memoria	256 MB de RAM; se recomienda 512 MB para características de gráficos, búsqueda instantánea de Outlook, Outlook con Business Contact Manager, Lync 2010 y ciertas funciones avanzadas.[1,2]
Disco duro	3,5 GB de espacio en disco disponible.
Pantalla	Monitor con resolución 1024x768 o superior.
Sistema operativo	Windows XP (debe tener SP3) (32 bits), Windows 7, Windows Vista con Service Pack (SP) 1, Windows Server 2003 con SP2 y MSXML 6.0 (solo Office de 32 bits), Windows Server 2008 o SO de 32 o 64 bits posterior.
Gráficos	La aceleración gráfica de hardware requiere una tarjeta gráfica DirectX® 9.0c con 64 MB o más memoria de vídeo.
Requisitos adicionales	Ciertas características de Microsoft® OneNote® requieren Windows® Desktop Search 3.0, reproductor de Windows Media® 9.0, Microsoft® ActiveSync® 4.1, micrófono, dispositivo de salida de audio, dispositivo de grabación de vídeo, cámara digital compatible con TWAIN o escáner; el uso compartido de blocs de notas requiere que los usuarios estén en la misma red.
	Determinadas funcionalidades avanzadas requieren conectividad con Microsoft® Exchange Server 2007, Microsoft® SharePoint® Server 2010, Microsoft® Office Communications Server 2007 y/o Microsoft® SharePoint® Foundation 2010.
	Determinadas características requieren Windows® Search 4.0.
	El controlador de impresión Enviar a de OneNote® y la integración con servicios de conectividad empresarial requieren características de Windows® XPS y/o Microsoft® .NET Framework 3.5.
	Windows® Internet Explorer® 7.0 o posterior, explorador de 32 bits únicamente. La funcionalidad de Internet requiere una conexión a Internet.
	Las características de multitoque requieren Windows 7 y un dispositivo táctil habilitado.

	Determinadas características de entradas de lápiz requieren Windows XP Tablet PC Edition o posterior.
	La funcionalidad de reconocimiento de voz requiere un micrófono para hablar de cerca y un dispositivo de salida de audio.
	Las características de ortografía contextual y gramática requieren 1 GB de memoria.
	Fax de Internet no está disponible en Windows Vista Starter, Windows Vista Home Basic ni Windows Vista Home Premium.
	Las características de Information Rights Management requieren acceso a una versión de Windows 2003 Server con SP1 o posterior que esté ejecutando los servicios de Windows Rights Management.
	Determinadas funcionalidades en línea requieren una cuenta de Windows LiveTM ID.
Otros	La funcionalidad del producto y los gráficos pueden variar de acuerdo con la configuración del sistema. Algunas características pueden requerir hardware adicional o avanzado, o conectividad del servidor; *www.office.com/products*.

Adicionalmente, se recomienda 512 MB de RAM para obtener acceso a archivos de datos de Outlook de más de 1 GB. También se recomienda un procesador de 2 GHz o más rápido, y 1 GB de RAM o más para la búsqueda de audio de OneNote.

4.1.2 Proceso de instalación de Access

El proceso de instalación de las últimas versiones de Access es similar. La instalación se realiza simultáneamente para todas las aplicaciones de la suite de Microsoft Office. Al introducir el CD-ROM en la unidad correspondiente del ordenador suele activarse automáticamente el proceso de instalación. Dependiendo del sistema operativo, puede ser necesario ejecutar el archivo SETUP.EXE (Figura 4.1). Después de una pantalla temporal, que advierte de la preparación de los archivos necesarios para la instalación, aparece la pantalla relativa a la licencia del programa (Figura 4.2).

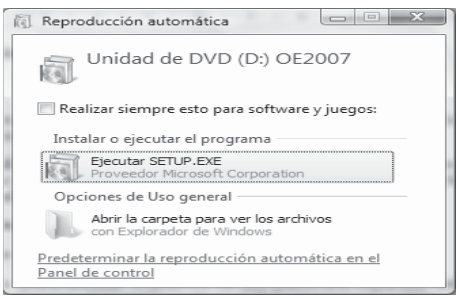


Figura 4.1

Figura 4.2

Aceptados los términos de la licencia, se pulsa Continuar para obtener la Figura 4.3 en la que habrá que elegir entre Actualizar una versión de Access ya existente o Personalizar. Es recomendable elegir esta última opción porque permite mantener versiones anteriores del programa según nuestras necesidades e instalar versiones nuevas elegidas a voluntad del usuario. Al hacer clic en Personalizar se obtiene la pantalla de la Figura 4.4 en la que se observa que es posible mantener todas las versiones anteriores, quitar todas las versiones anteriores o quitar solo las aplicaciones que se elijan.

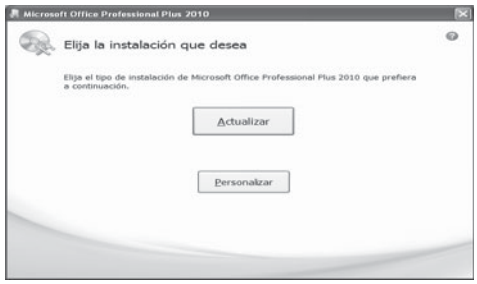


Figura 4.3

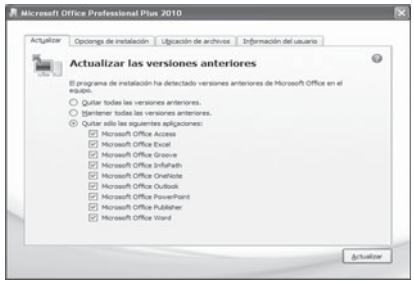


Figura 4.4

A continuación se hace clic en Actualizar y se obtiene la pantalla de la Figura 4.5 que por defecto presenta la pestaña Opciones de instalación en la cual es posible elegir los elementos de Office a instalar. Al hacer clic en un signo más (+) se abre una carpeta que muestra más características para la instalación. Si una característica tiene subcaracterísticas, un símbolo con fondo blanco indicará que la característica y todas sus subcaracterísticas se instalan de la misma forma. Un símbolo con fondo gris indica que la instalación es particular para cada característica y será necesario realizar una selección adecuada de las mismas para la instalación. Lo lógico es desplazarse con el teclado y elegir adecuadamente las funciones y sus opciones para la instalación a medida. La pestaña Ubicación de archivos (Figura 4.6) permite elegir el subdirectorio para realizar la instalación. Al hacer clic en Actualizar se ejecuta automáticamente el proceso de instalación.

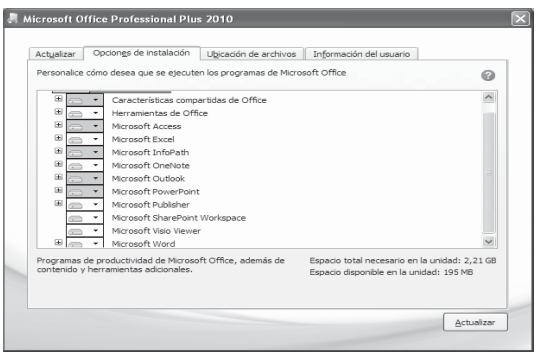


Figura 4.5

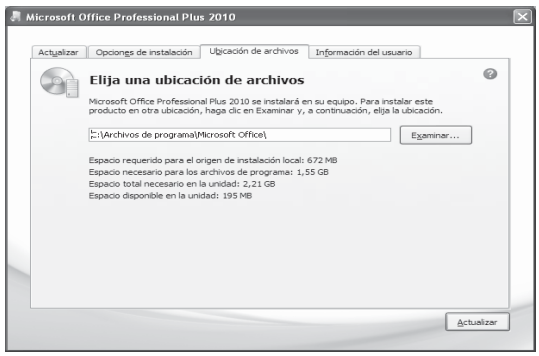


Figura 4.6

Cuando finalice el proceso de instalación será necesario reiniciar el equipo. Hay que tener presente que la instalación puede modificarse posteriormente en caso de que sea necesario agregar o quitar alguna característica a la instalación. Para ello, cierre todos los programas e introduzca el CD del programa en la unidad correspondiente. Dependiendo del sistema operativo o de la versión del programa, puede obtenerse automáticamente la pantalla de la Figura 4.7 o puede ser necesario ejecutar SETUP.EXE. Por cualquier vía se llega al cuadro de diálogo *Cambiar la instalación de Microsoft Office Professional Plus 2010* (Figura 4.7). Se observa que son posibles las siguientes opciones:

- *Agregar o quitar funciones.* Permite añadir nuevas características a la instalación y quitar las ya instaladas que no se consideren convenientes.
- *Reparar*. Permite reparar la instalación actual de Access en caso de que haya sufrido cualquier problema.
- *Quitar*. Elimina la instalación actual.
- *Escribir una clave de producto*. Permite cambiar la clave para la activación de Access.

Al hacer clic en *Agregar o quitar funciones* y, a continuación, en *Continuar* se obtiene la Figura 4.8 que permite realizar los cambios que se deseen en las características instaladas.

Figura 4.7

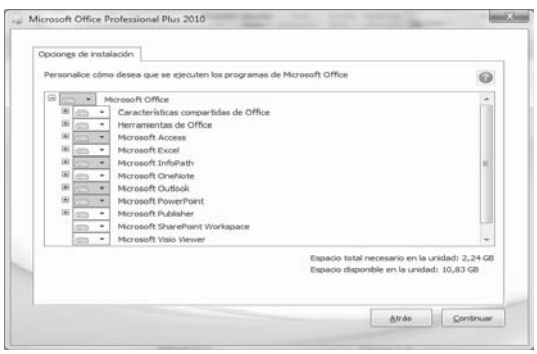


Figura 4.8

4.1.3 Puesta en marcha de Access

Access se inicia haciendo clic en la opción *Microsoft Access 2010* de la carpeta *Microsoft Office* del botón *Iniciar* de Windows (Figura 4.9).

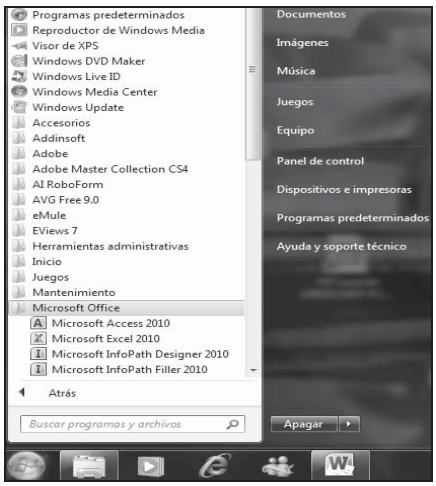


Figura 4.9

Después de la pantalla temporal de la Figura 4.10 se obtiene la pantalla de Introducción de Microsoft Office Access (Figura 4.11) que posibilita iniciar una nueva base de datos en blanco (*Base de datos en blanco*), iniciar una nueva base de datos web en blanco (*Base de datos web en blanco*), iniciar una base de datos según una plantilla haciendo clic sobre la plantilla elegida en esta pantalla de inicio de Access o abrir una base de datos ya existente (*Abrir*).

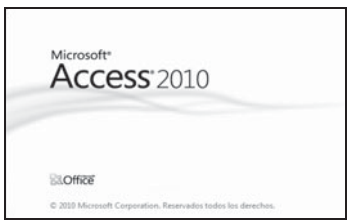


Figura 4.10

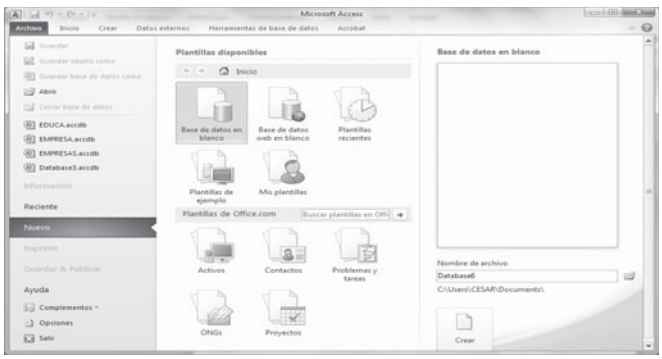


Figura 4.11

4.1.4 Entorno de trabajo de Access

Si creamos una nueva base de datos o abrimos una base de datos existente aparece el marco de trabajo general de Microsoft Access 2010 (Figura 4.12), que para las versiones 2007 y 2010 constituye una novedad importante para todo tipo de usuarios: se obtiene el marco de trabajo general de Microsoft Excel 2010 (Figura 4.13), que se explicará en los apartados que siguen.

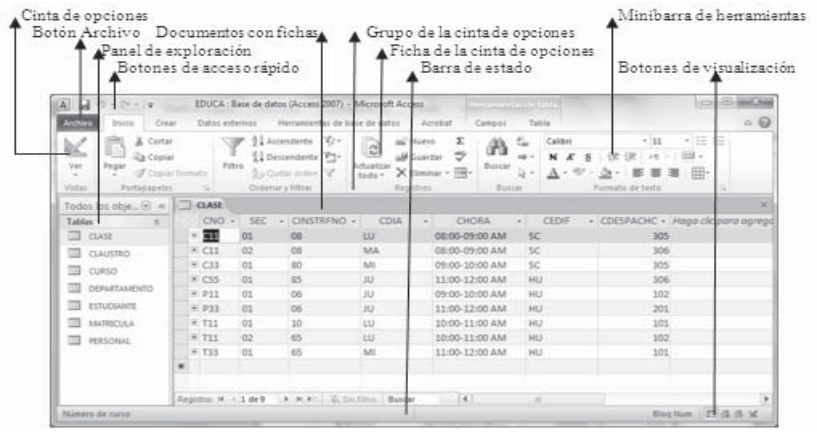


Figura 4.12

Cinta de opciones de Access

La cinta de opciones de Access (Figura 4.13) es uno de los elementos más importantes de la nueva interfaz de usuario de Microsoft Access. Reemplaza las barras de herramientas y los menús de versiones anteriores de Access.

Figura 4.13

Los tres componentes de la cinta de opciones son las fichas, los grupos y los comandos (opciones). Hay cuatro fichas en la parte superior de la cinta de opciones (Figura 4.14) y cada una de ellas representa una de las tareas básicas que se hacen en Access y suele sustituir a un menú de alguna barra de herramientas de versiones antiguas de Access. Cada ficha tiene grupos de opciones (o comandos) que muestran elementos relacionados entre sí. Un comando es una opción del grupo correspondiente representado por un botón, cuadro en el que se escribe información o menú. Por ejemplo, los comandos (u opciones)

Pegar, Cortar y *Copiar* aparecen en primer lugar en la ficha *Inicio*, dentro del grupo *Portapapeles* (Figura 4.15).

Figura 4.14

Figura 4.15

Los grupos aglutinan todos los comandos que se necesitan para realizar una tarea concreta, y se mantienen a la vista y disponibles para su uso mientras se trabaja en la tarea, en lugar de permanecer escondidos en los menús. Estos comandos esenciales están a la vista en la parte superior del área de trabajo, lo que constituye una clara ventaja de uso.

Botón Archivo

El botón *Archivo* Archivo de Access 2010 sustituye al botón *Microsoft Office* de Access 2007 y al clásico menú *Archivo* de versiones anteriores del programa. La Figura 4.16 muestra sus opciones. Se observa que es posible crear una nueva base de datos (*Nuevo*), abrir bases de datos ya existentes (*Abrir*), guardar una base de datos o sus objetos (*Guardar*), guardar una base de datos o sus objetos en otros formatos (*Guardar como*), cerrar la base de datos actual (*Cerrar*), imprimir el objeto actual (*Imprimir)*, ver las bases de datos usadas recientemente (*Reciente*), guardar y publicar la base de datos para su distribución (*Publicar*), personalizar opciones de Access (*Opciones*), guardar objetos de la base de datos en formato Adobe PDF (*Complementos*) y salir de Access (*Salir*).

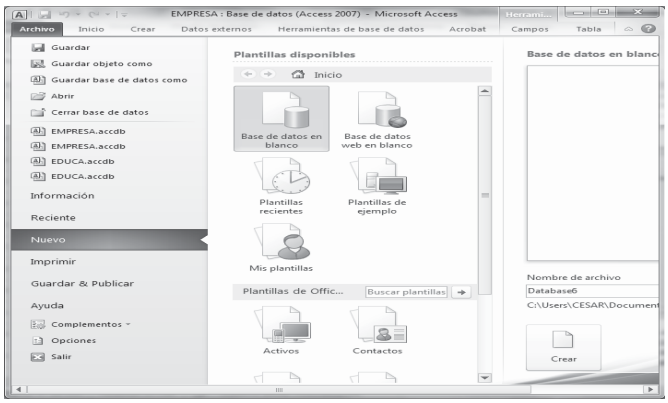


Figura 4.16

El botón *Opciones de Access* de la Figura 4.16 permite administrar las opciones más frecuentes (Figura 4.17), las opciones para la base de datos actual, las opciones de hojas de datos, las opciones de diseñadores de objetos, las opciones de revisión, las opciones avanzadas, las opciones de personalización, los complementos, las opciones de seguridad, y los recursos. El botón *Salir de Access* permite salir del programa.

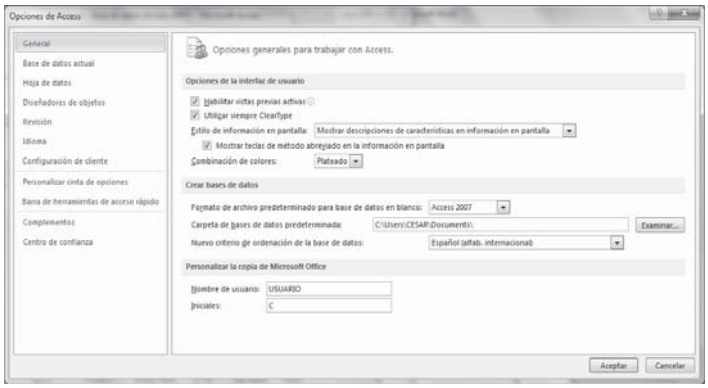


Figura 4.17

4.1.5 Configuración de Access

Al hacer clic en Opciones en el menú Archivo se obtiene la pantalla de opciones de Access de la Figura 4.18. El panel de la izquierda de esta pantalla presenta varias opciones cada una de las cuales va a permitir la configuración de los distintos elementos de la base de datos. La opción General permite configurar las opciones generales de Access entre las que se encuentran las opciones de la interfaz de usuario y las opciones de creación de la base de datos (Figura 4.18).

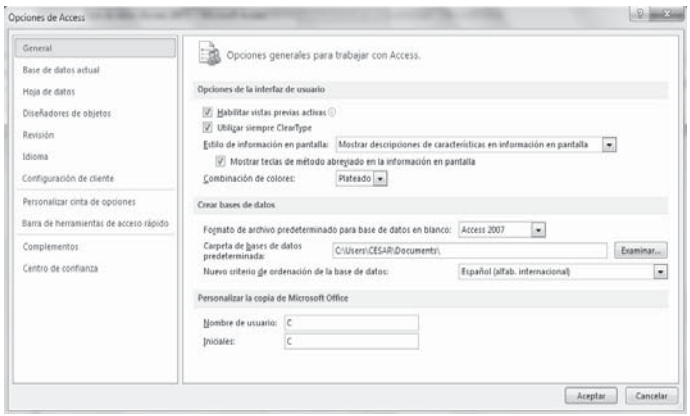


Figura 4.18

La opción Base de datos actual del panel de la izquierda de la pantalla de opciones de Access nos lleva a la Figura 4.19 que permite configurar las opciones de la base de datos actual. Se podrán configurar las opciones de aplicación, de navegación, de barra de herramientas, de autocorrección de nombres, de búsqueda por filtro y de almacenamiento en caché del servicio web y tablas de SharePoint.

- La opción Hoja de datos del panel de la izquierda de la pantalla de opciones de Access permite personalizar la apariencia de las hojas de datos de Access como los efectos de cuadrícula y celda y la fuente predeterminada.

- La opción Diseñadores de objetos del panel de la izquierda de la pantalla de opciones de Access permite cambiar la configuración predeterminada para diseñar objetos de la base de datos como tablas, consultas, formularios e informes.

- La opción Revisión del panel de la izquierda de la pantalla de opciones de Access permite cambiar la forma en que Access corrige y da formato automáticamente al contenido de las bases de datos, así como la forma en que indica los errores que encuentra (autocorrección y ortografía).

- La opción Idioma del panel de la izquierda de la pantalla de opciones de Access permite establecer las opciones de idioma para trabajar con Access.

- La opción Configuración de cliente del panel de la izquierda de la pantalla de opciones de Access permite establecer valores que cambian el comportamiento del cliente (edición, mostrar, impresión, etc.).

- La opción Personalizar la cinta de opciones del panel de la izquierda de la pantalla de opciones de Access permite establecer los elementos que aparecerán en la cinta de opciones.

- La opción Personalizar la barra de herramientas de acceso rápido del panel de la izquierda de la pantalla de opciones de Access permite establecer los elementos que aparecerán en la barra de herramientas de acceso rápido.

- La opción Complementos del panel de la izquierda de la pantalla de opciones de Access permite ver y administrar los complementos para Access.

- La opción Centro de confianza del panel de la izquierda de la pantalla de opciones de Access permite mantener los documentos seguros y el equipo protegido.

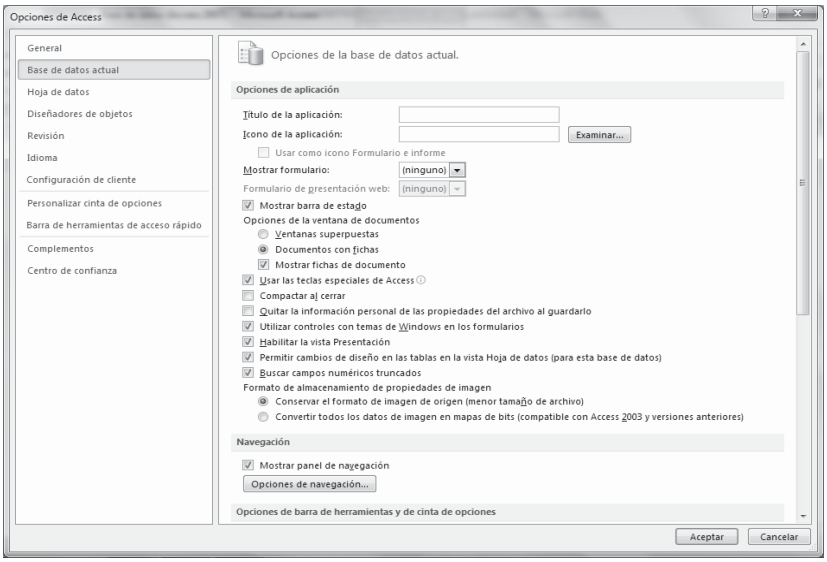


Figura 4.19

4.2 INSTALACIÓN DE SQL SERVER

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL o MySQL.

Entre las características de Microsoft SQL Server destacan las siguientes:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red solo acceden a la información.
- Además permite administrar información de otros servidores de datos. Se trata de un sistema gestor de bases de datos distribuido.

4.2.1 Requisitos para la instalación de SQL Server

Los requisitos de hardware y software siguientes se aplican a todas las instalaciones de SQL Server 2008 R2:

Componente	Requisito
Marco de trabajo	El programa de instalación de SQL Server instala los siguientes componentes de software requeridos por el producto: • .NET Framework 3.5 SP1 • SQL Server Native Client • Archivos auxiliares para la instalación de SQL Server
Software	El programa de instalación de SQL Server requiere Microsoft Windows Installer 4.5 o una versión posterior. Una vez instalados los componentes requeridos, el programa de instalación de SQL Server comprobará que el equipo en el que se ha instalado SQL Server 2008 R2 también cumple los demás requisitos para su correcta instalación.
Software de red	Los requisitos de software de red para las versiones de 64 bits de SQL Server 2008 R2 son los mismos que para las versiones de 32 bits. Los sistemas operativos compatibles tienen el software de red integrado. Las instancias predeterminadas y con nombre independientes admiten los siguientes protocolos de red: • Memoria compartida • Canalizaciones con nombre • TCP/IP • VIA El protocolo VIA está desusado. Esta característica se quitará en una versión futura de Microsoft SQL Server. Evite utilizar esta característica en nuevos trabajos de desarrollo y tenga previsto modificar las aplicaciones que actualmente la utilizan.
Virtualización	SQL Server 2008 R2 se admite en entornos de máquina virtual que se ejecutan en el rol Hyper-V de las ediciones Standard, Enterprise y Datacenter de Windows Server 2008 SP2. La máquina virtual debe ejecutarse en un sistema operativo compatible con la edición de SQL Server 2008 R2 concreta que se cita más adelante en este tema. Además de los recursos requeridos por la partición primaria, a cada máquina virtual (partición secundaria) se debe proporcionar suficientes recursos de procesador, memoria y recursos de disco para su instancia de SQL Server 2008 R2. Los requisitos se enumeran más adelante en este tema. Dentro del rol Hyper-V de Windows Server 2008 SP2, se puede asignar un máximo de cuatro procesadores virtuales a máquinas virtuales que ejecuten las ediciones de 32 o 64 bits de Windows Server 2008. Se pueden asignar como máximo dos procesadores virtuales a equipos virtuales que ejecuten ediciones de 32 bits de Windows Server 2003. Para equipos virtuales que hospedan otros sistemas operativos, se puede asignar como máximo un procesador virtual a equipos virtuales.  • Se recomienda cerrar SQL Server 2008 R2 antes de apagar la máquina virtual. • La agrupación en clústeres de conmutación por error del invitado se admite en SQL Server 2008 R2.

Software de Internet	Para todas las instalaciones de SQL Server 2008 R2 se requiere Microsoft Internet Explorer 6 SP 1 o una versión posterior. Se requiere Internet Explorer 6 Service Pack 1 o una versión posterior para Microsoft Management Console (MMC), SQL Server Management Studio, Business Intelligence Development Studio, el componente Diseñador de informes de Reporting Services y la Ayuda HTML.
Unidad	Para la instalación desde disco se necesita una unidad de CD o DVD.
Pantalla	Las herramientas gráficas de SQL Server 2008 R2 requieren Super VGA o una resolución mayor: resolución mínima de 800x600 píxeles.
Otros dispositivos	Dispositivo señalador: se necesita un mouse Microsoft o dispositivo señalador compatible.
Procesador	Tipo de procesador: • Procesador compatible con Pentium III o superior • Velocidad del procesador: • Mínimo: 1,0 GHz • Recomendado: 2,0 GHz o más
Memoria	RAM: • Mínimo: 1 GB • Recomendado: 4 GB o más • 2 TB (SQL Server Enterprise Edition admite un máximo de 2 TB de RAM o el valor máximo del sistema operativo, el que sea menor)

Se requieren las versiones siguientes de .NET Framework:

- SQL Server 2008 R2 en Windows Server 2003 (64 bits) IA64: .NET Framework 2.0 SP2
- SQL Server Express: .NET Framework 2.0 SP2
- Todas las demás ediciones de SQL Server 2008 R2: .NET Framework 3.5 SP1
- La instalación de .NET Framework requiere un reinicio del sistema operativo. Si la instalación de Windows Installer también requiere un reinicio, el programa de instalación esperará hasta que se hayan instalado los componentes de .NET Framework y Windows Installer antes de reiniciar.

En cuanto a los *requisitos de espacio en disco duro* hay que tener en cuenta que durante la instalación de SQL Server 2008 R2, Windows Installer crea archivos temporales en la unidad del sistema. Antes de ejecutar el programa de instalación para instalar o actualizar SQL Server, compruebe que dispone de al menos 3,6 GB de espacio en disco en la unidad del sistema para estos archivos. Este requisito es aplicable incluso si instala todos los componentes de SQL Server en una unidad distinta de la predeterminada.

Los requisitos reales de disco duro dependen de la configuración del sistema y de las características que decida instalar. En la tabla siguiente se muestran los requisitos de espacio en disco de los componentes de SQL Server 2008 R2:

Característica	Requisito de espacio en disco
Motor de base de datos y archivos de datos, Replicación y Búsqueda de texto completo	711 MB
Analysis Services y archivos de datos	345 MB
Reporting Services y Administrador de informes	304 MB
Integration Services	591 MB
Componentes de cliente (excepto Libros en pantalla y herramientas de Integration Services)	1823 MB
Libros en pantalla de SQL Server	157 MB

4.2.2 Proceso de instalación de SQL Server

Para instalar Microsoft SQL Server 2008 R2, se introduce el CD-ROM en la unidad correspondiente del equipo, con lo que se inicia automáticamente (en la mayoría de los equipos) el proceso de instalación. Inmediatamente aparece el Centro de instalación de SQL Server (Figura 4.20), en cuya parte izquierda se observa un menú con las distintas fases del proceso de instalación.

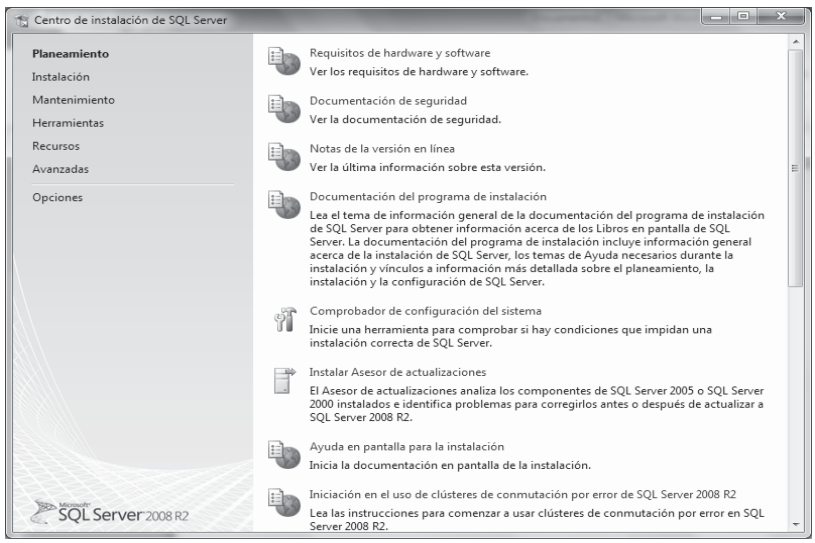


Figura 4.20

Si hacemos clic en Instalación en el Centro de instalación se obtiene la Figura 4.21, cuya opción *Nueva instalación* o *agregar características a una instalación existente* permite comenzar el proceso de instalación mediante la comprobación de las reglas auxiliares que deben cumplirse para llevar a cabo el proceso (Figura 4.22).

Figura 4.21

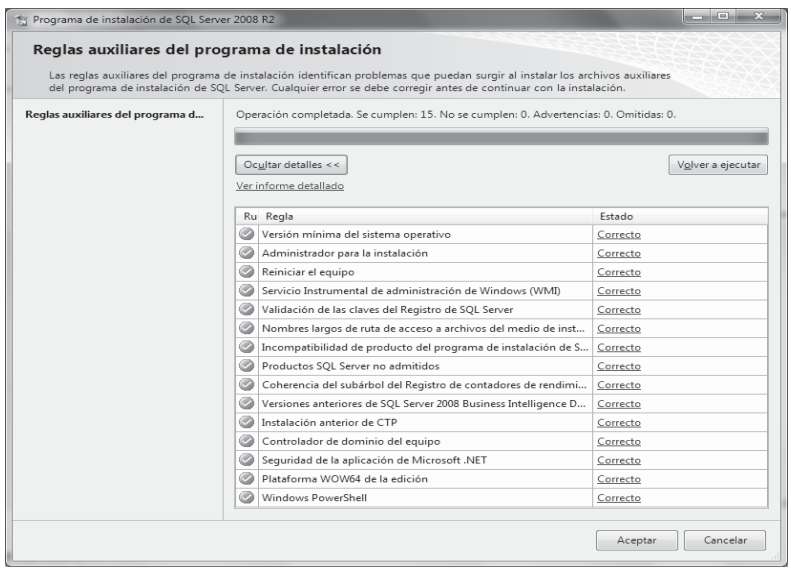


Figura 4.22

En los pasos siguientes se pide la clave del producto, la aceptación de la licencia y la instalación de los archivos auxiliares del programa. A continuación se define el rol de instalación y se eligen las características a instalar (Figura 4.23) y la configuración de la instancia (Figura 4.24).

A continuación, el programa presenta los requisitos de espacio en disco, sin cuyo cumplimiento es imposible la instalación (Figura 4.25). A continuación es necesario realizar la configuración del servidor (Figura 4.26).

A continuación es necesario configurar el Motor de base de datos, Analysis Services y Reporting Services, apareciendo finalmente la pantalla de Reglas de instalación (Figura 4.27). Al pulsar Siguiente se realiza la instalación (Figura 4.28).

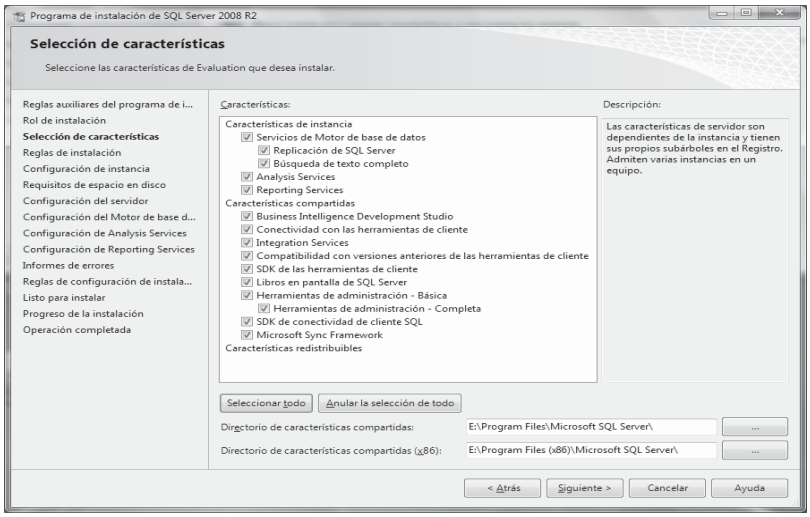


Figura 4.23

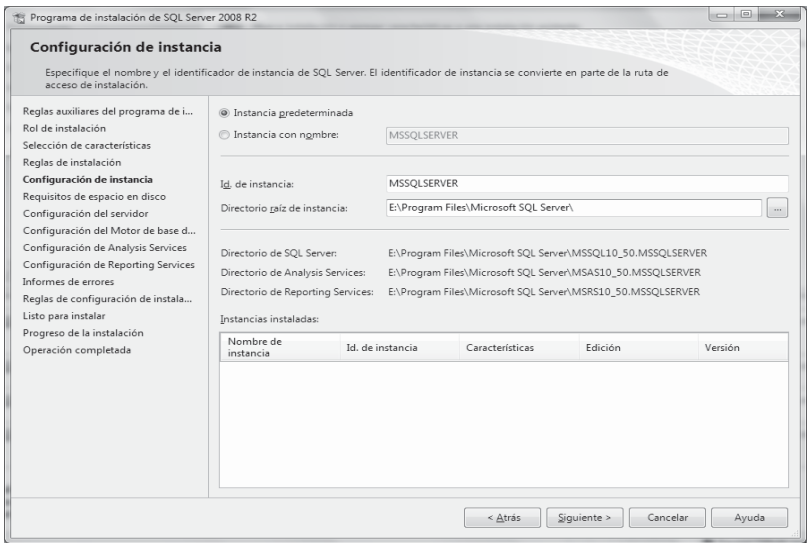


Figura 4.24

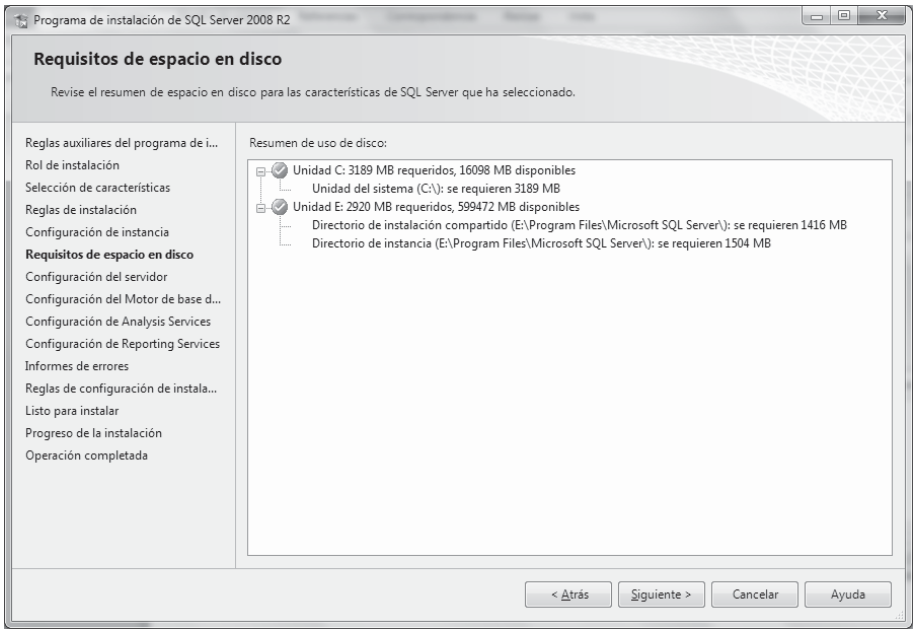


Figura 4.25

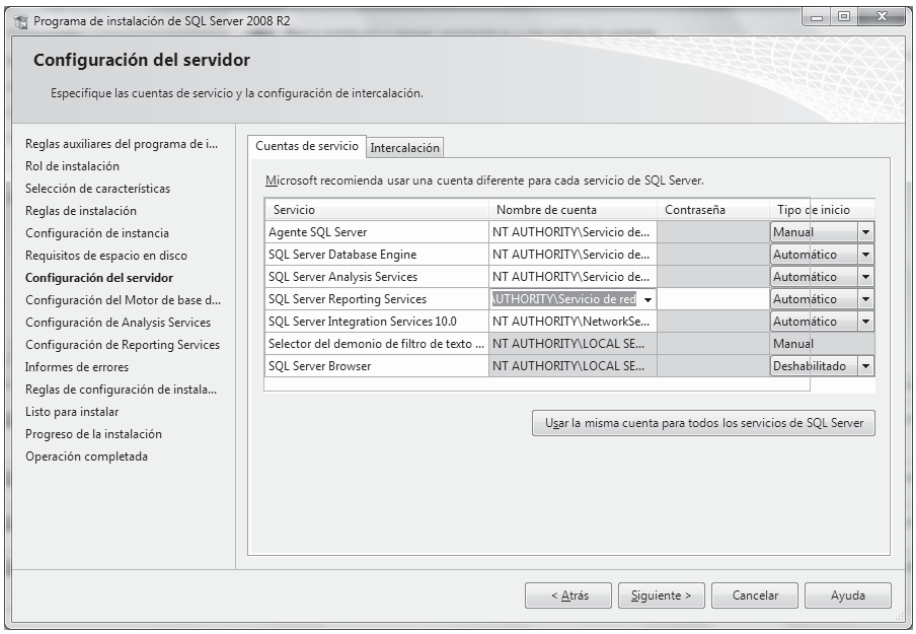


Figura 4.26

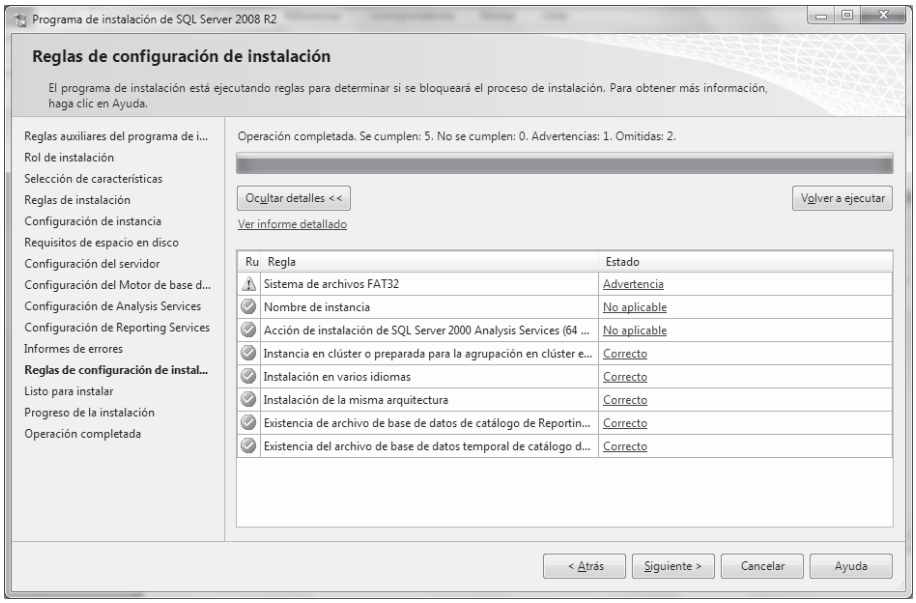


Figura 4.27

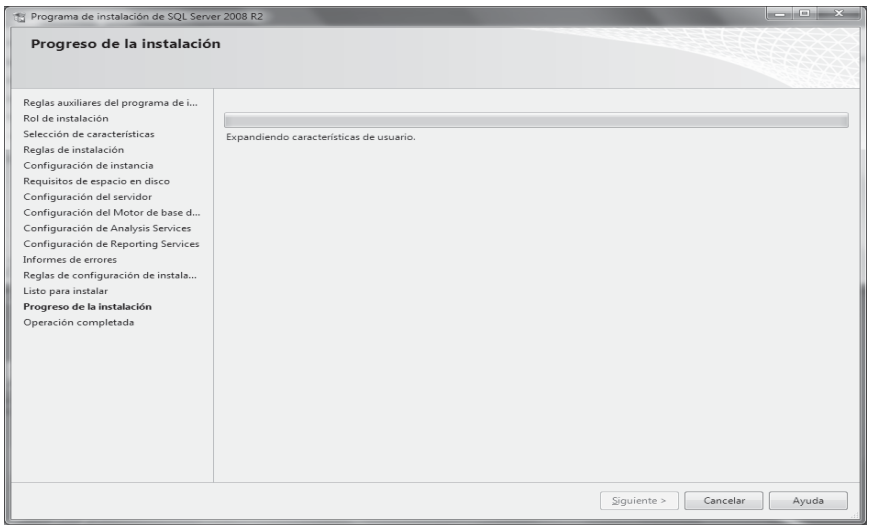


Figura 4.28

El progreso de instalación suele ser una tarea de varios minutos que finaliza en la pantalla final de *Operación completada*. El programa informa de los módulos instalados con éxito y, en su caso, de los que presenten errores. Finalizado el proceso de instalación, es conveniente reiniciar la máquina para que se memoricen los servicios adecuados y se active el Servidor SQL Server para su uso. Reiniciada la máquina, se obtiene ya SQL Server 2008 R2 incorporado al menú *Programas de Windows* (Figura 4.29), y, dado que se integra

perfectamente con .NET y Visual Studio, el programa instala herramientas básicas de desarrollo bajo el epígrafe *Microsoft Visual Studio 2008* (Figura 4.30).

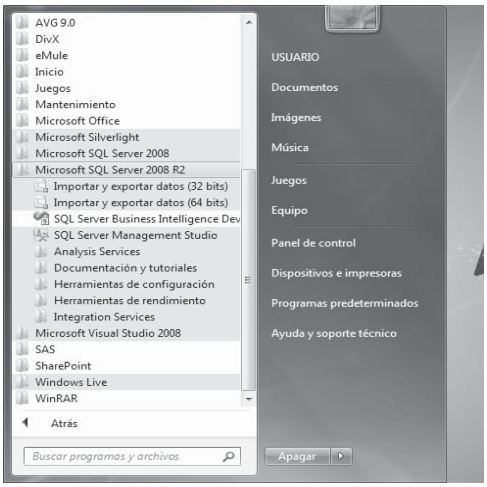


Figura 4.29

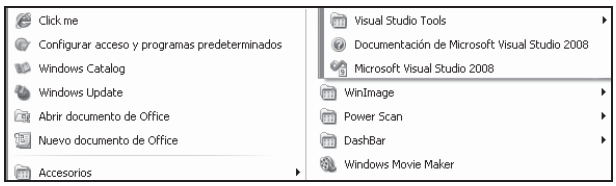


Figura 4.30

Capítulo

5

COMUNICACIONES

El funcionamiento del SGBD está muy interrelacionado con el del Sistema Operativo, especialmente con el sistema de comunicaciones. El SGBD utilizará las facilidades del sistema de comunicaciones para recibir las peticiones del usuario (que puede estar utilizando un terminal físicamente remoto) y para devolverle los resultados. El SGBD debe interactuar con software de comunicaciones, cuya función es permitir que los usuarios situados en lugares remotos respecto al sistema de base de datos tengan acceso a éste a través de terminales de computador, estaciones de trabajo o sus microcomputadores o minicomputadores locales. Estos se conectan al sitio de la base de datos por medio de equipos de comunicación de datos: líneas telefónicas, redes de larga distancia o dispositivos de comunicación por satélite. Muchos sistemas comerciales de bases de datos tienen paquetes de comunicaciones que funcionan con el SGBD. El sistema integrado de SGBD y comunicación de datos se denomina sistema BD/DC (database/datacommunications).

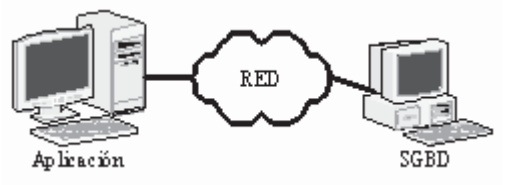
Por añadidura, algunos SGBD distribuidos están físicamente dispersos en varias máquinas. En este caso, se requieren redes de comunicaciones para conectar las máquinas. Con frecuencia se trata de redes de área local (LAN: Local Área Networks), pero también pueden ser de otro tipo. El término arquitectura cliente-servidor se usa para caracterizar un SGBD cuando la aplicación se ejecuta físicamente en una máquina, llamada cliente, y otra, el servidor, se encarga del almacenamiento y el acceso de los datos. Los proveedores ofrecen diversas combinaciones de clientes y servidores.

5.1 LOS SISTEMAS GESTORES DE BASES DE DATOS Y LA RED

Hay varias formas de utilizar un SGBD desde el punto de vista del emplazamiento en red del SGBD con respecto a las aplicaciones. Supondremos que estamos desarrollando una aplicación que va a hacer uso de un SGBD que almacenará sus datos en una base de datos. Podemos considerar básicamente tres configuraciones diferentes.

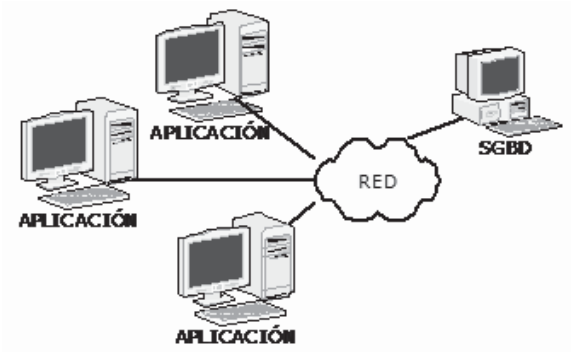
5.1.1 Servidor SGBD al que las aplicaciones acceden a través de una red

Ésta es probablemente la configuración más frecuente. El SGBD se ejecuta de manera independiente en una máquina, mientras que la aplicación se ejecuta en otra. El SGBD y la aplicación se comunican a través de un sistema de red.



Ese sistema de red puede ser de cualquier tipo: una red de área local, una red privada de área extensa, una comunicación punto a punto, o incluso Internet. En este supuesto, la aplicación funciona en su ordenador, y cuando requiere datos o necesita almacenarlos envía órdenes a través de la red al SGBD acerca de lo que necesita. El usuario de la aplicación la utiliza desde el ordenador en el que se encuentra la aplicación y, normalmente, la existencia del SGBD es completamente transparente para él.

Por supuesto, varias aplicaciones pueden estar utilizando el mismo SGBD simultáneamente por este sistema.



Este sistema presenta algunas ventajas. Por ejemplo:

- Dado que un SGBD consume muchos recursos en términos de CPU, memoria y almacenamiento secundario, al estar ejecutándose en una máquina aparte, esta máquina puede tener buen hardware, sin necesidad de que lo tengan las máquinas que tienen las aplicaciones.

- El ordenador del SGBD no tiene por qué estar a disposición de los usuarios de las aplicaciones, con lo que se reduce el riesgo de fallos por manipulación indebida.

- Se hace también más fácil la administración y configuración de los datos. El administrador de bases de datos (DBA) tiene los datos y el SGBD centralizados en un ordenador.

- La copia de seguridad de los datos es mucho más sencilla, al estar los datos centralizados. Si uno de los ordenadores que tiene una aplicación no funciona bien, la aplicación puede trasladarse rápidamente a otro ordenador. Los datos están salvaguardados por el SGBD.

No obstante, también hay inconvenientes. El principal es que si el SGBD deja de funcionar bien, también lo harán todas las aplicaciones que lo utilicen. En este tipo de configuraciones es muy importante que el SGBD tenga una alta disponibilidad.

5.1.2 El SGBD y la aplicación están en el mismo ordenador: el del usuario

Esta forma de trabajar solo tiene sentido en proyectos pequeños, en los que solo un usuario utiliza la aplicación (o bien un grupo de usuarios, pero no simultáneamente). Es la configuración típica de los programas que utilizamos para mantener en orden nuestra colección de DVD o libros, o incluso de los programas de contabilidad o nóminas que utilizan las empresas pequeñas o unipersonales.

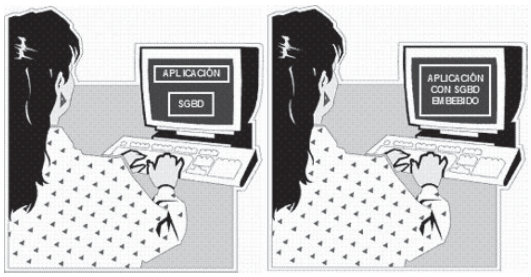
Aquí además podemos hacer una clasificación más:

1. El SGBD y la aplicación se ejecutan en la misma máquina, pero de manera independiente.

2. El SGBD y la aplicación se ejecutan en la misma máquina y, además, en el mismo proceso (SGBD "embebido" o "incrustado").

En el primer caso, aunque ambos programas estén en la misma máquina, cada uno de ellos se ejecuta por separado. Eso implica que el sistema operativo los ejecuta como procesos distintos (aunque se comuniquen entre sí). Así, el SGBD se puede normalmente configurar y administrar por separado, y además, el sistema operativo cede memoria y tiempo de CPU a cada uno de manera independiente.

En el segundo caso (muy pocos SGBD permiten esto), la aplicación y el SGBD comparten la memoria asignada por el sistema operativo y el tiempo de CPU. Esto se hace en SGBD no demasiado potentes. Normalmente mediante alguna librería que se compila o se enlaza junto con la aplicación. Como ventajas, se obtiene una mayor facilidad en la instalación de la aplicación, ya que no es necesario instalar el SGBD por separado. (*Nota*: la espantosa palabra "embebido" proviene de la traducción de la palabra inglesa "embedded2").



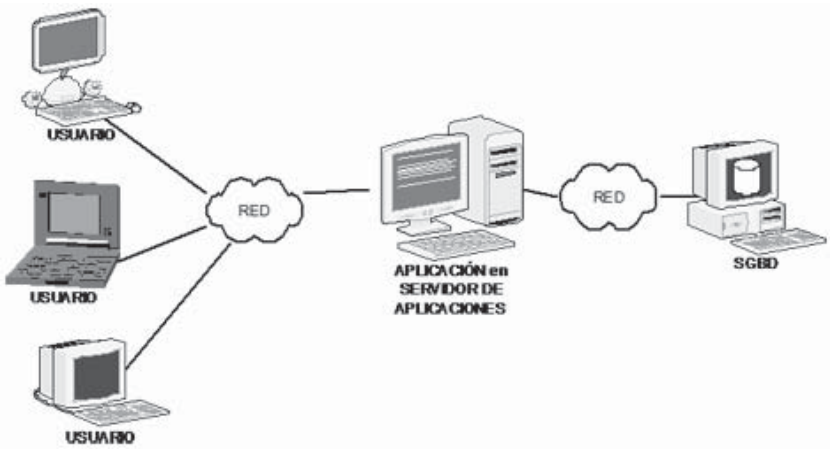
5.1.3 La aplicación se instala en un servidor de aplicaciones

En este último caso, la aplicación se instala en el espacio de un servidor de aplicaciones. Un servidor de aplicaciones es un programa capaz de permitir que varios usuarios a la vez trabajen con la misma aplicación, cada uno desde una máquina distinta, y teniendo la impresión de que la aplicación se ejecuta en exclusiva para ellos.

Ésta es la configuración típica de las aplicaciones de Internet o Intranet, a las cuales el usuario accede utilizando un navegador.

Un servidor de aplicaciones es, en sí, una aplicación, capaz de atender las conexiones de los usuarios a través de su navegador (u otro programa similar, llamados genéricamente "clientes ligeros") y, por decirlo de alguna manera sencilla, "replicar" nuestra aplicación para cada uno de ellos.

Por supuesto, programar una aplicación para que funcione de esta manera requiere del aprendizaje de algunas técnicas no demasiado complicadas. A nadie se le escapa que no tiene la misma dificultad hacer una aplicación que va a tener un solo usuario frente a otra que puede tener varios miles de usuarios simultáneamente.



En esta configuración, se obtiene como ventaja que la misma aplicación se distribuye a un número potencialmente muy grande de usuarios, y además, los usuarios no necesitan instalar ni configurar nada en su máquina aparte de un cliente ligero (que suele ser un navegador).

En definitiva, los SGBD son, en la actualidad los grandes custodios de los datos, con algoritmos sumamente eficientes y rápidos, descargando a los programadores de la responsabilidad de programar complejos algoritmos en este sentido. No obstante, un SGBD es capaz de servir a las aplicaciones de distintas maneras. Aunque hemos comentado de manera muy genérica tres esquemas sumamente comunes, podemos encontrar en la realidad muchos más.

5.2 ARQUITECTURA CLIENTE SERVIDOR

Podemos definir la arquitectura cliente/servidor de diferentes formas entre las que se encuentran las siguientes:

- IBM define al modelo cliente/servidor como la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados "servidores".

- De modo más sencillo puede considerarse la arquitectura cliente servidor como cualquier combinación de sistemas que pueden colaborar entre sí para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber dónde está ubicada. O lo que es lo mismo, se trata de una arquitectura de procesamientos cooperativa donde uno de los componentes pide servicios a otro. Se trata, por tanto, de un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red. El término cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una aplicación y un servicio soportante. Se trata de un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

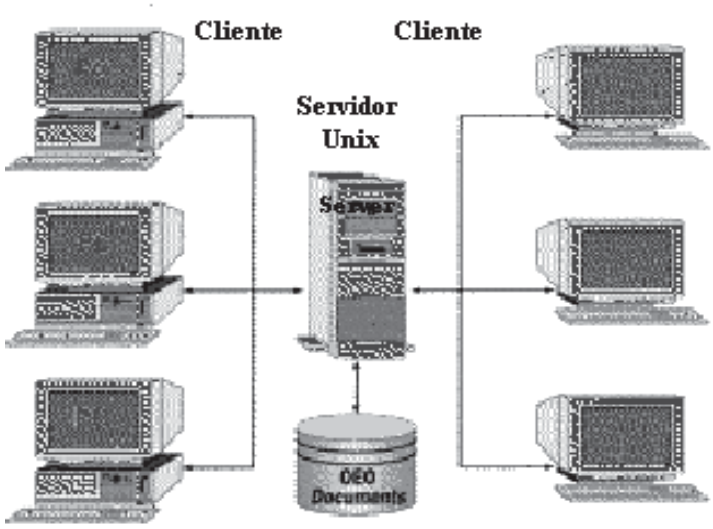
"Los elementos principales de la arquitectura cliente servidor son justamente el elemento llamado cliente y el otro elemento llamado servidor". Por ejemplo dentro de un ambiente multimedia, el elemento cliente sería el dispositivo que puede observar el vídeo, cuadros y texto, o reproduce el audio distribuido por el elemento servidor. Por otro lado el cliente también puede ser una computadora personal o una televisión inteligente que posea la

capacidad de entender datos digitales. Dentro de este caso el elemento servidor es el depositario del vídeo digital, audio, fotografías digitales y texto y los distribuye bajo demanda de ser una máquina que cuenta con la capacidad de almacenar los datos y ejecutar todo el software que brinda estos al cliente.

Cliente servidor (C/S) es una relación entre procesos corriendo en máquinas separadas. El servidor (S) es un proveedor de servicios. El cliente (C) es un consumidor de servicios.

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización. Debemos señalar que para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura cliente/servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran. El cliente es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Un servidor es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN o WAN, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc. La figura siguiente esquematiza gráficamente la arquitectura cliente servidor.



5.2.1 Elementos de la arquitectura cliente servidor

Para determinar los elementos de la arquitectura cliente servidor tendremos presente que toda aplicación de un sistema de información está caracterizada por tres componentes básicos:

- Presentación/captación de información.
- Procesos.
- Almacenamiento de la información.

Estos componentes se integran en una arquitectura Cliente/Servidor en base a los elementos que caracterizan dicha arquitectura, es decir:

- Puestos de trabajo.
- Comunicaciones.
- Servidores.

El puesto de trabajo o cliente. Se trata de una estación de trabajo o microcomputador (PC: Computador Personal) conectado a una red, que le permite acceder y gestionar una serie de recursos, lo cual se perfila como un puesto de trabajo universal. Nos referimos a un microcomputador conectado al sistema de información y en el que se realiza una parte mayoritaria de los procesos. Debemos destacar que el puesto de trabajo basado en un microcomputador conectado a una red, favorece la flexibilidad y el dinamismo en las organizaciones. Entre otras razones, porque permite modificar la ubicación de los puestos de trabajo, dadas las ventajas de la red.

Los servidores o back-end. Se trata de una máquina que suministra una serie de servicios como bases de datos, archivos, comunicaciones,...). Los servidores, según la especialización y los requerimientos de los servicios que deben suministrar pueden ser: mainframes, miniordenadores o especializados (dispositivos de red, imagen, etc.). Una característica a considerar es que los diferentes servicios, según el caso, pueden ser suministrados por un único servidor o por varios servidores especializados.

Las Comunicaciones. Pueden considerarse desde dos vertientes:

- *Infraestructura de redes*. Componentes hardware y software que garantizan la conexión física y la transferencia de datos entre los distintos equipos de la red.
- *Infraestructura de comunicaciones*. Componentes hardware y software que permiten la comunicación y su gestión, entre los clientes y los servidores.

La arquitectura cliente/servidor es el resultado de la integración de dos culturas. Por un lado, la del mainframe que aporta capacidad de almacenamiento, integridad y acceso a la información y, por el otro, la del computador que aporta facilidad de uso (cultura de PC), bajo costo, presentación atractiva (aspecto lúdico) y una amplia oferta en productos y aplicaciones.

5.2.2 Características del modelo cliente servidor

En el modelo cliente/servidor podemos encontrar las siguientes características:

- El cliente y el servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

- Las funciones de cliente y servidor pueden estar en plataformas separadas, o en la misma plataforma.

- Un servidor da servicio a múltiples clientes de forma concurrente.

- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

- La interrelación entre el hardware y el software está basada en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones clientes**.** Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final. También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Por ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red. Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

- Además se constituye como el nexo de unión más adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.

- Designa un modelo de construcción de sistemas informáticos de carácter distribuido. Su representación típica es un centro de trabajo (PC), donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin

dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los recursos que este host central y otros sistemas de la organización ponen a su servicio.

En conclusión, cliente/servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en TokenRing, Ethernet, FDDI o medio coaxial, solo por mencionar algunas de las posibilidades.

5.2.3 Estilos del modelo cliente servidor

Dentro de la arquitectura cliente servidor pueden considerarse diferentes estilos de organización.

En primer lugar puede considerarse la *presentación distribuida* en la cual se distribuye la interfaz entre el cliente y la plataforma servidora. A su vez, la aplicación y los datos están ambos en el servidor siendo similar a la arquitectura tradicional de un host y terminales. Por otra parte, el PC se aprovecha solo para mejorar la interfaz gráfica del usuario. Este tipo de presentación distribuida tiene como ventajas que revitaliza los sistemas antiguos, presenta bajo costo de desarrollo y no hay cambios en los sistemas existentes. Como desventajas puede citarse que el sistema sigue en el host, no se aprovecha la GUI y/o LAN y la interfaz del usuario se mantiene en muchas plataformas.

En segundo lugar puede considerarse la *presentación remota* en la que la interfaz para el usuario esta completamente en el cliente y la aplicación y los datos están en el servidor. Presenta como ventajas que la interfaz del usuario aprovecha bien la GUI y la LAN, la aplicación aprovecha el host y es una presentación adecuada para algunos tipos de aplicaciones de apoyo a la toma de decisiones. Como desventajas podemos citar que las aplicaciones pueden ser complejas de desarrollar, los programas de la aplicación siguen en el host y el alto volumen de tráfico en la red puede hacer difícil la operación de aplicaciones muy pesadas.

En tercer lugar podemos considerar la *lógica distribuida* en la que el interfaz está en el cliente, la base de datos está en el servidor y la lógica de la aplicación está distribuida entre el cliente y el servidor. Presenta como ventajas una arquitectura más corriente que puede manejar todo tipo de aplicaciones, los programas del sistema pueden distribuirse al nodo más apropiado y pueden utilizarse con sistemas existentes. Como desventajas citamos que es difícil de diseñar, que es de difícil prueba y mantenimiento si los programas del cliente y el servidor están hechos en distintos lenguajes de programación y que no son manejados por la GUI 4GL.

En cuarto lugar podemos considerar la *administración de datos remota* en la que en el cliente residen tanto la interfaz como los procesos de la aplicación y las bases de datos están

en el servidor, tratándose del estilo que comúnmente imaginamos como aplicación cliente servidor. Presenta como ventajas la configuración típica de la herramienta GUI 4GL, que es muy adecuada para las aplicaciones de apoyo a las decisiones del usuario final, que es fácil de desarrollar ya que los programas de aplicación no están distribuidos, y que se descargan los programas del host. Como desventajas tenemos que no maneja aplicaciones pesadas eficientemente y que la totalidad de los datos viajan por la red, ya que no hay procesamiento que realice el host.

En quinto lugar podemos considerar las *bases de datos distribuidas*, en las que la interfaz, los procesos de la aplicación, y parte de los datos de la base de datos están en el cliente. El resto de los datos están en el servidor. Las ventajas son ahora que la configuración es soportada por herramientas GUI 4GL, es adecuada para las aplicaciones de apoyo al usuario final, apoya el acceso a datos almacenados en ambientes heterogéneos y la ubicación de los datos es transparente para la aplicación. Como desventajas podemos citar que no maneja aplicaciones grandes eficientemente y el acceso a la base de datos distribuida es dependiente del proveedor del software administrador de bases de datos.

Una característica a tener en cuenta en el campo de la arquitectura cliente servidor es la definición de *middleware*. Se trata de un término que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores. Es el enlace que permite que un cliente obtenga un servicio de un servidor. Éste se inicia en el módulo de API de la parte del cliente que se emplea para invocar un servicio real; esto pertenece a los dominios del servidor. Tampoco a la interfaz del usuario ni a la lógica de la aplicación en los dominios del cliente.

Existen dos tipos de middleware:

- *Middleware general*. Este tipo permite la impresión de documentos remotos, manejos de transacciones, autenticación de usuarios, etc.

- *Middleware de servicios específicos*. Generalmente trabajan orientados a mensajes. Trabaja una sola transacción a la vez.

En cuanto a las *funciones de un programa servidor* podríamos destacar las siguientes:

- Espera las solicitudes de los clientes.

- Ejecuta muchas solicitudes al mismo tiempo.

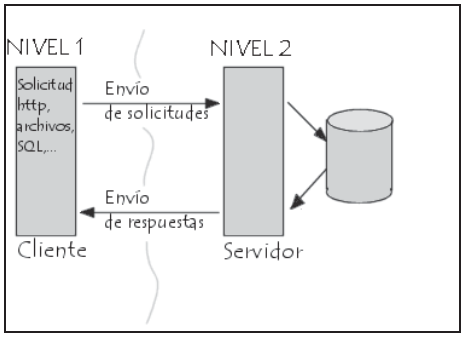
- Atiende primero a los clientes VIP.

- Emprende y opera actividades de tareas en segundo plano.

- Se mantiene activo en forma permanente.

5.2.4 Introducción a la arquitectura en 2 niveles

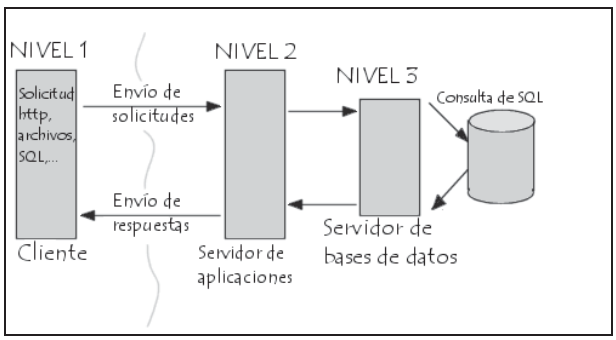
La arquitectura en 2 niveles se utiliza para describir los sistemas cliente/servidor donde el cliente solicita recursos y el servidor responde directamente a la solicitud, con sus propios recursos. Esto significa que el servidor no requiere otra aplicación para proporcionar parte del servicio.



5.2.5 Introducción a la arquitectura en 3 niveles

En la arquitectura en 3 niveles existe un nivel intermediario. Esto significa que la arquitectura generalmente está compartida por:

- Un cliente, es decir, el equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador web) para la presentación.
- El servidor de aplicaciones (también denominado software intermedio), cuya tarea es proporcionar los recursos solicitados, pero que requiere de otro servidor para hacerlo.
- El servidor de datos, que proporciona al servidor de aplicaciones los datos que requiere.



5.2.6 Comparación entre ambos tipos de arquitecturas

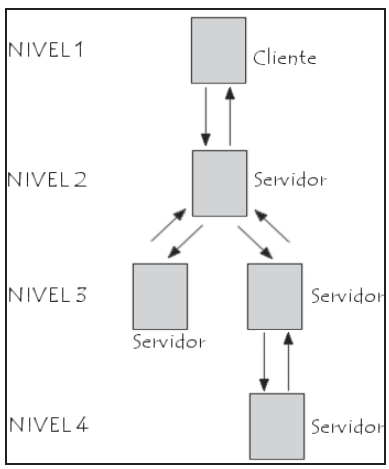
La arquitectura en 2 niveles es, por lo tanto, una arquitectura cliente/servidor en la que el servidor es polivalente, es decir, puede responder directamente a todas las solicitudes de recursos del cliente.

Sin embargo, en la arquitectura en 3 niveles, las aplicaciones al nivel del servidor son descentralizadas de uno a otro, es decir, cada servidor se especializa en una determinada tarea (por ejemplo: servidor web/servidor de bases de datos). La arquitectura en 3 niveles permite:

- Un mayor grado de flexibilidad.
- Mayor seguridad, ya que la seguridad se puede definir independientemente para cada servicio y en cada nivel.
- Mejor rendimiento, ya que las tareas se comparten entre servidores.

5.2.7 Arquitectura de niveles múltiples

En la arquitectura en 3 niveles, cada servidor (niveles 2 y 3) realiza una tarea especializada (un servicio). Por lo tanto, un servidor puede utilizar los servicios de otros servidores para proporcionar su propio servicio. Por consiguiente, la arquitectura en 3 niveles es potencialmente una arquitectura en N-niveles.



5.3 ACCESS EN RED. LOS SITIOS DE SHAREPOINT

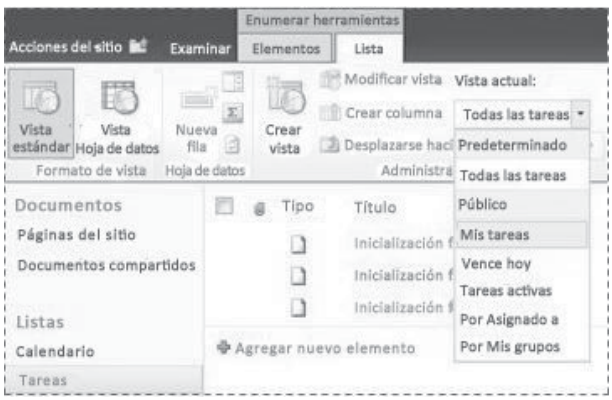
Un sitio de SharePoint es un grupo de páginas web relacionadas donde la organización puede trabajar en proyectos, celebrar reuniones y compartir información. Por ejemplo, el equipo puede tener su propio sitio donde almacena programaciones, archivos e información de procedimientos. El sitio de grupo puede ser parte de un sitio del portal de una organización donde los departamentos, como Recursos Humanos, escriben y publican información y recursos para el resto de la organización.

Todos los sitios de SharePoint tienen elementos comunes que deben conocerse antes de empezar a trabajar: listas, bibliotecas, elementos web y vistas.

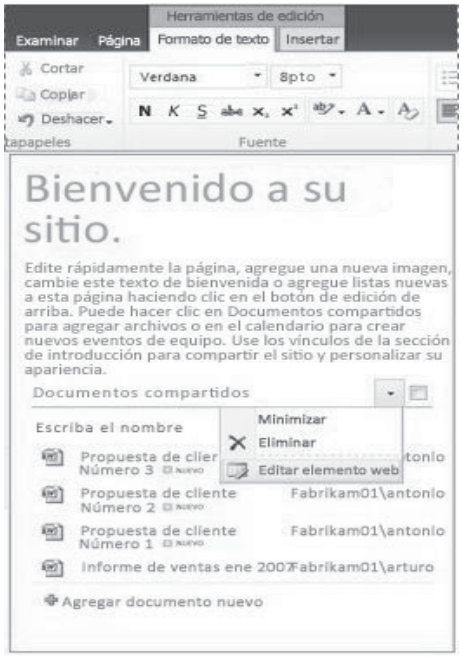
- *Listas*. Una lista es un componente del sitio web donde la organización puede almacenar, compartir y administrar información. Por ejemplo, puede crear una lista de tareas para realizar un seguimiento de las asignaciones de trabajo o realizar un seguimiento de eventos del equipo en un calendario. También puede realizar una encuesta u hospedar discusiones en un panel de discusión.

- *Bibliotecas*. Una biblioteca es un tipo de lista especial que almacena archivos, así como información acerca de los archivos. Puede controlar la visualización, el seguimiento, la administración y la creación de los archivos en bibliotecas.

- *Vistas.* Puede usar vistas para ver los elementos de una lista o biblioteca que sean más importantes o que mejor se ajusten a un propósito. Por ejemplo, puede crear una vista de todos los elementos de una lista que se aplican a un departamento específico o para resaltar determinados documentos en una biblioteca. Puede crear varias vistas de una lista o biblioteca entre las que pueden seleccionar los usuarios. También puede usar un elemento web para mostrar una vista de una lista o biblioteca en una página independiente de su sitio.



- *Elementos web*. Un elemento web es una unidad modular de información que forma un bloque de creación básico de la mayoría de las páginas de un sitio. Si tiene permiso para editar las páginas del sitio, puede usar elementos web para personalizar el sitio para mostrar imágenes y gráficos, partes de otras páginas web, listas de documentos, vistas personalizadas de datos profesionales, etc.



5.3.1 Abrir un archivo de Access desde un sitio de SharePoint

Los sitios de SharePoint permiten el trabajo compartido entre usuarios de Access a través de la red. Para abrir un archivo de Access desde un sitio de SharePoint se tendrá en cuenta lo siguiente:

- Haga clic en la pestaña *Archivo*.
- Haga clic en *Abrir*.
- Bajo Favoritos, haga clic en Sitios de SharePoint.
- Haga clic en el sitio de SharePoint en que se encuentra el archivo y, a continuación, haga clic en *Abrir*. Si no ve el servidor en *Sitios de SharePoint*, escriba la dirección URL del sitio de SharePoint en el cuadro de *Nombre de archivo*.

- Haga clic en el nombre de la biblioteca que contiene el archivo, por ejemplo *Documentos compartidos* y luego haga clic en *Abrir*.

- Haga clic en el nombre del archivo que desea abrir y, a continuación, haga clic en *Abrir*. Si no desea realizar cambios en el archivo, puede abrirlo como de solo lectura.Haga clic en la flecha que aparece en el botón de *Abrir* y, a continuación, haga clic en *Solo lectura abierto*.

5.3.2 Desproteger un archivo de Access que ha sido abierto

Si varias personas trabajan en los mismos archivos a través de la red, deben desproteger los archivos antes de trabajar en ellos. Para poder realizar cambios en un archivo es necesario desprotegerlo. Esto reduce la posibilidad de confusión o los conflictos de edición. Realizados los cambios, el archivo debe protegerse antes de que otras personas puedan ver los cambios.

Para desproteger un archivo mientras está abierto en el programa de Office (también puede desproteger un archivo de la biblioteca del sitio de SharePoint), siga estos pasos...

- Haga clic en la pestaña *Archivo*.

- Haga clic en *Información*.

- Haga clic en *Administrar versiones*.

- Haga clic *Desproteger*.

Si la biblioteca requiere desprotección, un comando *Desproteger* aparece en la barra de mensajes cuando se abre un archivo desde el programa de Office.

5.3.3 Guardar un archivo en una biblioteca de SharePoint

Puede guardar con facilidad los archivos de Access directamente en una biblioteca de Microsoft SharePoint desde el archivo, o bien en un área de trabajo de SharePoint que se sincronizará más adelante.

Las bibliotecas de SharePoint son ubicaciones en un sitio de SharePoint donde puede almacenar y administrar los archivos que comparte con los integrantes del equipo. Después de agregar archivos a la biblioteca, los demás pueden leerlos y editarlos, según los permisos que tengan.

Un área de trabajo de SharePoint es una copia del sitio de SharePoint, o de las secciones seleccionadas que descargó a su equipo para usarlas sin conexión. Sincronizar el

área de trabajo con el sitio de SharePoint le permite cargar documentos en los que trabajó sin conexión y también descargar los archivos actualizados por los integrantes del equipo.

Para guardar una base de datos de Access en una biblioteca de SharePoint se tendrá en cuenta lo siguiente:

- Haga clic en la pestaña *Archivo*.
- Haga clic en Guardar y publicar y, a continuación, haga clic en Guardar base de datos como o Guardar objeto como.
- En *Avanzadas*, seleccione *SharePoint* y, a continuación, haga clic en *Guardar como*.
- En el cuadro de diálogo *Guardar como*, busque la ubicación de SharePoint en la que desea guardar el archivo y haga clic en *Guardar*.

5.3.4 Importar y exportar archivos a listas de Sharepoint

En muchos programas, se usa el comando *Guardar como* para guardar un documento en otro formato a fin de poder abrirlo en otro programa. En Access, en cambio, el comando *Guardar como* no se usa de la misma forma. Se pueden guardar objetos de Access como otros objetos de Access y se pueden guardar bases de datos de Access como versiones anteriores de bases de datos de Access. No obstante, no se puede guardar una base de datos de Access como, por ejemplo, un archivo de hoja de cálculo y tampoco es posible guardar un archivo de hoja de cálculo como un archivo de Access (.accdb). En cambio, se usan los comandos de la ficha *Datos externos* de Access para importar o exportar datos entre otros formatos de archivo.

También se pueden escribir macros o código de Visual Basic para Aplicaciones (VBA) para automatizar las operaciones de importación y exportación disponibles en la ficha *Datos externos*.

Una forma rápida de conocer los formatos de datos que se pueden importar o exportar con Access es abrir una base de datos y, a continuación, explorar la ficha *Datos externos* en la cinta de opciones.

El grupo *Importar y vincular* (etiquetado con un 1 en la figura) muestra iconos de los formatos de datos desde los que se puede importar con Access o a los que se puede vincular con Access. El grupo *Exportar* (etiquetado con un 2 en la figura) muestra iconos de todos los formatos a los que Access puede exportar datos. En cada grupo, puede hacer clic en *Más* para ver más formatos con los que puede trabajar Access. Si no ve el programa o tipo de datos específico que necesita, es probable que pueda exportar los datos con el otro programa a un formato que Access reconozca. Por ejemplo, la mayoría de los programas puede exportar datos de columnas como texto delimitado, que después pueden importarse fácilmente en Access.

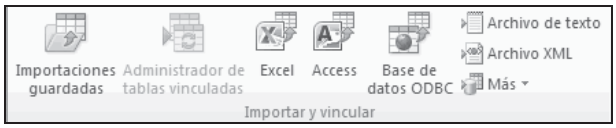
La siguiente tabla muestra qué formatos pueden importarse, vincularse y exportarse con Access:

Programa o formato	¿Permite importación?	¿Permite vinculación?	¿Permite exportación?
Microsoft Office Excel	✓ Sí	✓ Sí	✓ Sí
Microsoft Office Access	✓ Sí	✓ Sí	✓ Sí
Bases de datos ODBC (por ejemplo, SQL Server)	✓ Sí	✓ Sí	✓ Sí
Archivos de texto (delimitados o de longitud fija)	✓ Sí	✓ Sí	✓ Sí
Archivos XML	✓ Sí	✗ No	✓ Sí
Archivos PDF o XPS	✗ No	✗ No	✓ Sí
Correo electrónico (datos adjuntos)	✗ No	✗ No	✓ Sí
Microsoft Office Word	No, pero se puede guardar un archivo de Word como archivo de texto y después importar este archivo	No, pero se puede guardar un archivo de Word como archivo de texto y después crear un vínculo a este archivo	Sí (se puede exportar como Word Merge o texto enriquecido)
Lista de SharePoint	✓ Sí	✓ Sí	✓ Sí
Servicios de datos (vea la nota)	✗ No	✓ Sí	✗ No
Documentos HTML	✓ Sí	✓ Sí	✓ Sí
Carpetas de Outlook	✓ Sí	✓ Sí	No, pero se puede exportar como archivo de texto y después importar este archivo en Outlook
Archivos de dBase	✓ Sí	✓ Sí	✓ Sí

5.3.5 Importar datos con otro formato o crear vínculos a ellos

El proceso general para importar datos o crear vínculos a ellos es el siguiente:

- Abra la base de datos en la que desee importar o vincular datos.
- En la ficha *Datos externos*, haga clic en el tipo de datos que desee importar o vincular. Por ejemplo, si los datos de origen están en un libro de Microsoft Excel, haga clic en *Excel*.



- En la mayoría de los casos, Access inicia el Asistente para obtener datos externos. Es posible que el asistente le solicite toda la información siguiente o parte de la misma:
  - Especifique el origen de los datos (su ubicación en el disco).
  - Elija si desea importar los datos o crear un vínculo a ellos.
  - Si desea importar datos, elija si desea anexar los datos a la tabla existente o crear una tabla nueva.
  - Especifique exactamente qué datos del documento desea importar o vincular.
  - Indique si la primera fila contiene encabezados de columna o si se debe tratar como datos.
  - Especifique el tipo de datos de cada columna.
  - Elija si desea importar solo la estructura, o bien la estructura y los datos.
  - Si desea importar datos, especifique si desea que Access agregue una nueva clave principal a la nueva tabla o usar una clave existente.
  - Especifique un nombre para la nueva tabla.
- En la última página del asistente, por lo general Access pregunta si desea guardar los detalles de la operación de importación o vinculación. Si cree que deberá realizar la misma operación de forma recurrente, active la casilla de verificación *Guardar los pasos de la importación*, complete la información y haga clic en

Cerrar. A continuación, puede hacer clic en *Importaciones guardadas* en la ficha *Datos externos* para volver a ejecutar la operación.

Una vez completado el asistente, Access le notificará acerca de cualquier problema que pudiera haber ocurrido durante el proceso de importación. En algunos casos, es posible que Access cree una nueva tabla llamada **ErroresDeImportación**, que contiene los datos que no se pudieron importar correctamente. Puede examinar los datos de esta tabla para intentar determinar por qué no se importaron correctamente.

En el caso de la *importación de datos de una lista de SharePoint*, en el grupo *Importar y vincular* de la ficha *Datos externos*, se hace clic en *Más* y a continuación en *Lista SharePoint* (Figura 5.1). Se obtiene el asistente de la Figura 5.2 cuyos pasos hay que seguir hasta finalizar la importación.

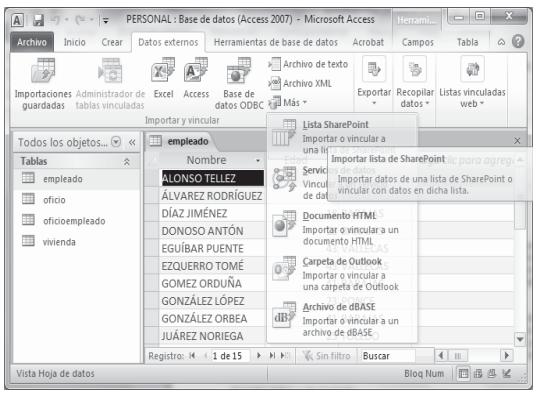


Figura 5.1

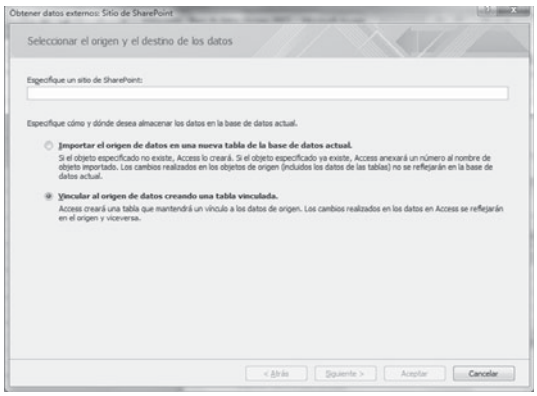


Figura 5.2

5.3.6 Exportar datos a otro formato

El proceso general para exportar datos desde Access es el siguiente:

- Abra la base de datos de la que desee exportar datos.
- En el panel de navegación, seleccione el objeto del que desee exportar los datos. Puede exportar datos de objetos de tablas, consultas, formularios e informes, aunque no todas las opciones de exportación están disponibles para todos los tipos de objeto.
- En la ficha *Datos externos*, haga clic en el tipo de datos que desea exportar. Por ejemplo, para exportar datos en un formato que se pueda abrir con Microsoft Excel, haga clic en *Excel*.

- En la última página del asistente, por lo general Access le preguntará si desea guardar los detalles de la operación de exportación. Si cree que deberá realizar la misma operación de forma recurrente, active la casilla de verificación *Guardar los pasos de la exportación*, complete la información y haga clic en *Cerrar*. A continuación, puede hacer clic en *Exportaciones guardadas* en la ficha *Datos externos* para volver a ejecutar la operación.

En el caso de la *exportación de datos de una lista de SharePoint*, en el grupo *Exportar* de la ficha *Datos externos*, se hace clic en *Más* y a continuación en *Lista SharePoint* (Figura 5.3). Se obtiene el asistente de la Figura 5.4 cuyos pasos hay que seguir hasta finalizar la importación.

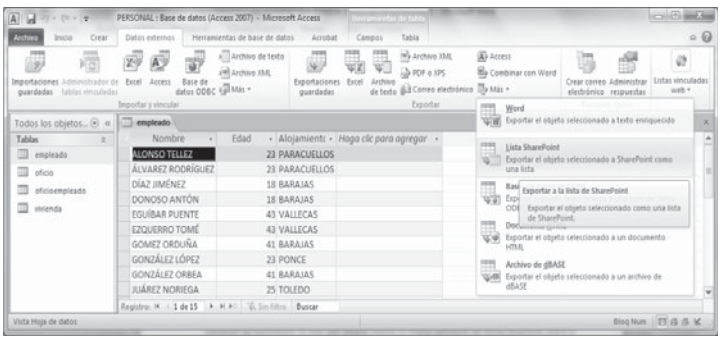


Figura 5.3

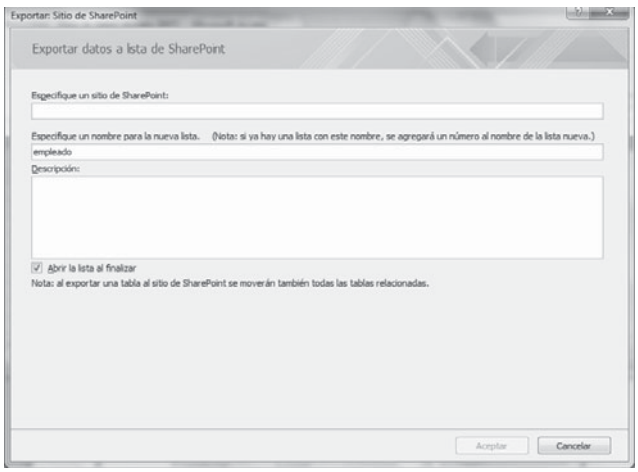


Figura 5.4

Capítulo

6

ADMINISTRACIÓN DE UN SISTEMA GESTOR DE BASES DE DATOS

6.1 TAREAS ADMINISTRATIVAS Y FUNCIONES DEL ADMINISTRADOR DEL SISTEMA GESTOR

Realizar las labores de administración de un sistema gestor de bases de datos exige el conocimiento de la arquitectura y el funcionamiento interno del SGBD, aprender a crear estructuras en la base de datos y realizar la gestión y mantenimiento de los recursos.

Suelen ser tareas iniciales de administración la realización de la instalación, configuración y mantenimiento del SGBD. Se incluye la administración de usuarios y de la seguridad, creación e implantación de elementos (bases de datos, tablas, índices, vistas, triggers, procedimientos almacenados), tareas administrativas (copias, backups, tuning,…), resolución de problemas y manejo de herramientas administrativas.

La planificación de las tareas administrativas y la monitorización y ajuste del rendimiento también son tareas integradas en la administración del sistema gestor de base de datos.

La administración de un sistema gestor de base de datos gira en torno a la figura del administrador del SGBD o DBA (*Data Base Administrator*). El DBA tiene una gran responsabilidad ya que posee el máximo nivel de privilegios entre los que se encuentra crear los usuarios que se conectarán a la base de datos. En la administración de una base de datos siempre hay que procurar que haya el menor número de administradores, a ser posible una sola persona.

El administrador es el encargado de gestionar y controlar todo el sistema con la ayuda que le proporciona el SGBD. Tiene una gran responsabilidad ya que de él depende que el sistema funcione correctamente y como tiene el máximo nivel de privilegios, sus errores pueden ser desastrosos. Entre sus responsabilidades se incluye:

- Instalar el SGBD en el sistema informático (a veces).
- Realizar el diseño de la base de datos.
- Crear las bases de datos que se vayan a gestionar.
- Crear y mantener los esquemas de las bases de datos.
- Crear y mantener las cuentas de los usuarios de las bases de datos.
- Colaborar con el administrador del sistema en las tareas de ubicación, dimensionado y control de los archivos y espacios de disco ocupados por el SGBD.
- Establecer estándares de uso, políticas de acceso y protocolos de trabajo diario para los usuarios de las bases de datos.

Efectuar tareas de explotación como:

- Vigilar el trabajo diario colaborando en la resolución de las dudas de los usuarios.
- Controlar los tiempos de acceso, tasas de uso, cargas en los servidores, anomalías, etc.
- Llegado el caso, reorganizar las bases de datos.
- Diseñar y efectuar el planning de copias de seguridad periódicas.
- Restaurar la base de datos después de un incidente.
- Estudiar las auditorías mediante el ajuste de parámetros y con ayuda de las herramientas de monitorización del sistema y de las estadísticas.

6.2 ADMINISTRACIÓN DE BASES DE DATOS EN ACCESS

A través de un ejemplo bastante completo se mostrarán las tareas más comunes de la administración de bases de datos a través de Microsoft Access.

6.2.1 Un ejemplo de diseño de base de datos relacional

A modo de ejemplo se va a presentar el diseño de una base de datos en un contexto educativo cuyas tablas se refieren a los cursos que van a ser impartidos en distintas clases de un centro educativo por varios profesores pertenecientes a diversos departamentos y en los que se matricularán distintos estudiantes. En el centro educativo también existe personal no docente.

Las tablas de la base de datos son CURSO, DEPARTAMENTO, CLASE, ESTUDIANTE, MATRÍCULA, CLAUSTRO y PERSONAL. En los párrafos siguientes se definen las características de las tablas y el diseño de la base de datos.

En primer lugar consideramos la tabla de nombre CURSO, que va a definir las características de los cursos que van a ser impartidos. Los nombres de los campos y los tipos de datos se especifican a continuación:

Nombre del campo	Tipo de datos	Descripción
CNO	Texto	Número de curso
CNOMBRE	Texto	Nombre de curso
CDESCP	Texto	Descripción del curso
CRED	Número	Créditos que vale el curso
CTARIFA	Número	Valor tarifa laboratorio
CDEPT	Texto	Nombre id. departamento que da el curso

Las longitudes de los campos son 3, 22, 50, entero largo y 4 respectivamente. Los datos de la tabla van a ser los siguientes:

CNO	CNOMBRE	CDESCP	CRED	CTARIFA	CDEPT
C11	INTROD. A LAS CC.	PARA NOVATOS	3	1	CIS
C22	ESTRUCT. DE DATOS	MUY ÚTIL	3	50	CIS
C33	MATEMÁTICAS DISCRETAS	ABSOLUTAMENTE NECESARIO	3	0	CIS
C44	CIRCUITOS DIGITALES	AH HA!	3	0	CIS
C55	ARQUIT. COMPUTADORES	MAQ. VON NEUMANN	3	100	CIS
C66	BASES DE DATOS RELAC.	IMPRESCINDIBLE	3	500	CIS
P11	EMPIRISMO	VERLO PARA CREERLO	3	100	PHIL
P22	RACIONALISMO	PARA USARLOS CIS	3	50	PHIL
P33	EXISTENCIALISMO	PARA USARLOS CIS	3	200	PHIL
P44	SOLIPSISMO	PARA MI MISMO	6	0	PHIL
T11	ESCOLASTICISMO	PARA BEATOS	3	150	THEO
T12	FUNDAMENTALISMO	PARA DESCUIDADOS	3	90	THEO
T33	HEDONISMO	PARA SANOS	3	0	THEO
T44	COMUNISMO	PARA AVAROS	6	200	THEO

En esta tabla CURSO puede tomarse CNO como la clave primaria, dado que CNO toma valores distintos para todas las filas y no tiene valores faltantes.

Directamente relacionada con la tabla CURSO, vamos a considerar la tabla DEPARTAMENTO, que tiene un registro por cada departamento académico al que pertenecen los profesores que imparten cursos.

Sus campos van a ser el nombre identificativo de cada departamento, el edificio al que pertenecen, el número de despacho en que están ubicados y el número de identificación en el cuerpo docente del director del departamento. Los nombres de los campos y el contenido de la tabla se especifican a continuación:

Nombre del campo	Tipo de datos	Descripción
DEPT	Texto	Nombre id. deparatamento
DEDIF	Texto	Edificio
DDESPACHO	Número	Despacho
DCHFNO	Texto	Director

DEPT	DEDIF	DDESPACHO	DCHFNO
CIS	SC	300	80
D.G.	SC	100	
PHIL	HU	100	60
THEO	HU	200	10

Las longitudes de los campos son respectivamente 4, 2, entero largo y 3.

Vemos que el campo DEPT toma valores distintos para todas las filas y no tiene valores faltantes. Además, según la política del centro educativo, cada curso debe estar organizado por un departamento académico, lo que nos indica que debemos tomar el campo DEPT como clave primaria de la tabla DEPARTAMENTO.

Para aumentar la integridad de la base de datos, especificamos el campo CDEPT de la tabla CURSO como clave secundaria que referencie la *tabla padre* DEPARTAMENTO (se restringen los valores CDEPT de la tabla CURSO a aquéllos que se encuentran en la columna DEPT de la tabla DEPARTAMENTO). La tabla CURSO es una *tabla dependiente o tabla hijo* de la tabla padre DEPARTAMENTO.

Adicionalmente consideramos la tabla CLASE, que tiene un registro para cada clase que ofrezca un curso. Los campos que contiene son: número de curso, número de sección, número de identificación en el grupo de docentes del profesor, día y hora, edificio y sala en la que se encuentra la clase. Los nombres de los campos y el contenido de la tabla se especifican a continuación:

Nombre del campo	Tipo de datos	Descripción
CNO	Texto	Número de curso
SEC	Texto	Número de sección
CINSTRFNO	Texto	Número de profesor
CDIA	Texto	Día de la semana
CHORA	Texto	Hora de la clase
CEDIF	Texto	Edificio de la clase
CDESPACHO	Número	Sala de la clase

CNO	SEC	CINSTRFNO	CDIA	CHORA	CEDIF	CDESPACHO
C11	01	08	LU	08:00-09:00 AM	SC	305
C11	02	08	MA	08:00-09:00 AM	SC	306
C33	01	80	MI	09:00-10:00 AM	SC	305
C55	01	85	JU	11:00-12:00 AM	HU	306
P11	01	06	JU	09:00-10:00 AM	HU	102
P33	01	06	JU	11:00-12:00 AM	HU	201
T11	01	10	LU	10:00-11:00 AM	HU	101
T11	02	65	LU	10:00-11:00 AM	HU	102
T33	01	65	MI	11:00-12:00 AM	HU	101

Las longitudes de los campos son respectivamente 3, 2, 2, 2, 14, 2 y entero largo.

En esta tabla consideramos el par de campos (CNO, SEC) como clave primaria. Además, sabemos que debido a la política del centro educativo, a cada clase le corresponde algún curso, por lo que cada valor del campo CNO de la tabla CLASE debe coincidir con algún valor CNO existente en la tabla CURSO. De esta forma, CNO en la tabla CLASE es clave secundaria que referencia CURSO, con lo que CURSO es *tabla padre* de CLASE, además de *depender (ser tabla hijo)* de DEPARTAMENTO. Puesto que CLASE es dependiente de CURSO, que por su parte es dependiente de DEPARTAMENTO, decimos que CLASE es *tabla descendente* de DEPARTAMENTO.

Las relaciones existentes entre las tres tablas que llevamos definidas en nuestra base de datos suelen representarse como se indica en la Figura 6.1.

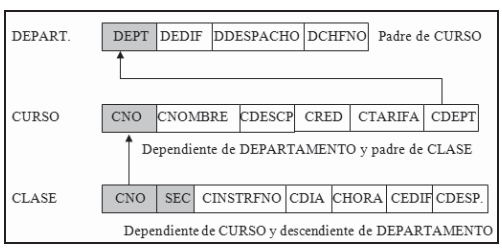


Figura 6.1

Ahora se consideran las tablas MATRÍCULA (relativa a los datos recogidos cuando un estudiante se matricula en un curso) y ESTUDIANTE (tiene un registro para cada estudiante matriculado en el centro educativo). Los campos de la tabla MATRÍCULA son: número de curso, número de sección, número de estudiante, fecha y hora de matriculación del estudiante.

Los campos que contiene la tabla ESTUDIANTE son: número de estudiante, nombre, dirección, teléfono, fecha de nacimiento, número de identificación de la escuela o facultad (centro educativo) y departamento. Los nombres de los campos, los tipos de datos y el contenido de la tabla MATRÍCULA se muestran a continuación:

Nombre del campo	Tipo de datos	Descripción
CNO	Texto	Número de curso
SEC	Texto	Número de sección
SNO	Texto	Número de estudiante
FEC_MAT	Fecha/Hora	Fecha de matricualción
HORA_MAT	Fecha/Hora	Hora de matricualción

CNO	SEC	SNO	FEC_MAT	HORA_MAT
C11	01	325	04/01/1987	9:41:30
C11	01	800	15/12/1987	11:49:00
C11	02	100	17/12/1987	9:32:00
C11	02	150	17/12/1987	9:32:30
P33	01	100	23/12/1987	11:30:00
P33	01	800	23/12/1987	11:23:00
T11	01	100	23/12/1987	11:21:00
T11	01	150	15/12/1987	11:35:30
T11	01	800	15/12/1987	14:00:00

Las longitudes de los tres primeros campos son 3, 2 y 3 respectivamente.

Los nombres de los campos, los tipos de datos y el contenido de la tabla ESTUDIANTE se especifican a continuación:

Nombre del campo	Tipo de datos	Descripción
SNO	Texto	Número de estudiante
SNOMBRE	Texto	Nombre de estudiante
SDOMI	Texto	Domicilio
STLFNO	Texto	Teléfono
SFNACIM	Texto	Fecha nacimiento
SIQ	Número	Número id. centro
SADVFNO	Texto	Número id. departamento
SESP	Texto	Nombre id. departamento

SNC	SNOMBRE	SDOMI	STLFNO	SFNACIM	SIQ	SADVFNO	SESP
100	MOE DUBAY	CONNECTICUT	203-123-4567	780517	120	10	THEO
150	LARRY DUBAY	CONNECTICUT	203-123-4567	780517	121	80	CIS
325	CURLEY DUBAY	CONNECTICUT	203-123-4567	780517	122	10	THEO
800	ROCKY BALBOA	PENNSYLVANIA	112-112-1122	461004	99	60	PHIL

Las longitudes de los campos son 3, 30, 15, 12, 6, entero, 3 y 4 respectivamente.

La clave primaria de la tabla MATRÍCULA es una composición de tres columnas (CNO, SEC, SNO). Para asegurar la integridad referencial, especificaríamos dos claves secundarias en esta tabla. La clave secundaria compuesta (CNO, SEC) referenciaría la tabla ESTUDIANTE. De esta forma, MATRÍCULA es dependiente de dos tablas: CLASE y ESTUDIANTE. Supongamos, también, que todos los estudiantes deben especializarse en una asignatura correspondiente a algún departamento académico existente. Esto significa que la sentencia

CREATE TABLE para ESTUDIANTE especificaría SESP (especialidad del estudiante) como clave secundaria que referencia la tabla DEPARTAMENTO. Por lo tanto, DEPARTAMENTO es padre de otras dos tablas: CURSO y ESTUDIANTE. La clave primaria para la tabla ESTUDIANTE es el campo SNO (número de estudiante). La Figura 6.2 refleja la nueva visión del diseño de la base de datos.

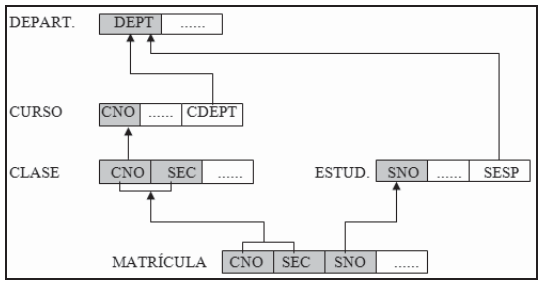


Figura 6.2

Consideremos ahora la tabla CLAUSTRO, que tiene un registro para cada miembro del cuerpo docente del centro educativo (escuela universitaria o facultad) y cuyos campos son: número de identificación de todos los profesores, nombre, dirección, fecha de contratación, número de ayudantes o subordinados que dependen de él, sueldo e identificación del departamento al que se encuentre adscrito. Los nombres de los campos, los tipos de datos y el contenido de la tabla CLAUSTRO se especifican a continuación:

FN	FNOMBRE	FDOMI	FFCANTI	FNUMDE	FSUELDO	FDEP
06	KATHY PEPE	CALLE LA PIEDRA, 7	15/01/1979	2	35000	PHIL
08	JOSE COHN	APT. CORREOS 1138	09/07/1979	2	35000	CIS
10	JESSIE MARTIN	DR. DEL ESTE, 4	01/09/1969	1	45000	THEO
60	JULIA MARTIN	DR. ESTE, 4	01/09/1969	1	45000	PHIL
65	LISA BOBAK	CAMINO LA RISA, 77	06/09/1981	0	36000	THEO
80	BARB HLAVATY	CALLA DEL SUR, 489	16/01/1982	3	35000	CIS
85	AL HARTLEY	CALLE DE LA PLATA	05/09/1979	7	45000	CIS

Nombre del campo	Tipo de datos	Descripción
FNO	Texto	Número de profesor
FNOMBRE	Texto	Nombre de profesor
FDOMI	Texto	Domicilio de profesor
FFCANTI	Fecha/Hora	Fecha contratación
FNUMDEP	Número	Número de ayudantes
FSUELDO	Número	Número de ayudantes
FDEP	Texto	Nombre id. departamento

Las longitudes de los campos son 3, 15, 20, entero, decimal (7,2) y 4 respectivamente.

El campo FNO es la clave primaria de la tabla CLAUSTRO. Las claves secundarias pueden ser definidas de forma que se establezca una relación cíclica. Supongamos que es norma de la escuela que a cada miembro del personal docente se le asigne algún departamento académico. Entonces, la columna FDEPT, en la tabla CLAUSTRO, se especificaría como clave secundaria, referenciando DEPARTAMENTO (DEPARTAMENTO es la tabla padre y CLAUSTRO la tabla dependiente). Supongamos, también, que el director de cada departamento es miembro del personal docente. Entonces, el valor DCHFNO, en la tabla DEPARTAMENTO, se especifica como clave secundaria que referencia CLAUSTRO. (Aquí, CLAUSTRO es la tabla padre y DEPARTAMENTO es la tabla dependiente.) La Figura 6.3 muestra esta *relación cíclica o ciclo*.

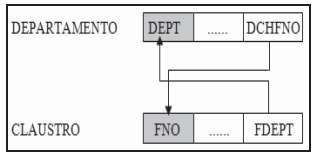


Figura 6.3

Es posible también que una tabla sea padre y dependiente de sí misma, dando lugar a un *ciclo recursivo*. En otras palabras, la tabla contiene una clave secundaria que se referencia a sí misma. Por ejemplo, supongamos que algunos cursos exigen, al menos, haber cursado otro curso, que es requisito indispensable (es decir, los alumnos están obligados a realizarlo) para matricularse en una clase del curso. Podríamos indicar el curso que se exige como requisito para la matriculación en la tabla CURSO, modificando esta tabla para incluir una nueva columna, PCNO, que contenga el número del curso que se pidió como prerrequisito a la matriculación.

En ese caso, la orden CREATE TABLE para el curso contendría una cláusula adicional de clave secundaria, especificando PCNO como clave secundaria que referencia CURSO. Este ciclo de autorreferencia se llama, a veces, *relación recursiva*.

El tipo de datos y la extensión de cada componente de una clave secundaria deben ser los mismos que los de la clave primaria. Pero una clave secundaria, a diferencia de la primaria, puede contener valores nulos (es decir, no es necesario especificar NOT NULL para una columna que sea clave secundaria). Por ejemplo, existen muchos cursos introductorios que no necesitan un curso como requisito previo. Los registros de tales cursos podrían tener un valor nulo en la columna PCNO. Otro ejemplo es la columna SESP en la tabla ESTUDIANTE. Si suponemos que algunos estudiantes no tienen que declarar la especialidad, entonces los valores correspondientes de SESP podrían ser nulos. Observemos, además, que si cualquier parte de una clave secundaria compuesta es nula, entonces toda la clave secundaria se considera como nula.

En la base de datos existe una última tabla de nombre PERSONAL que tiene un registro para cada empleado que no pertenezca al cuerpo docente de la escuela universitaria o facultad. Sus campos son: nombre, título, sueldo e identificación del departamento del que dependa. Estos campos no tienen relación con ninguna otra tabla.

Los nombres de los campos, los tipos de datos y el contenido de la tabla PERSONAL se especifican a continuación (ENOMBRE es la clave primaria):

Nombre del campo	Tipo de datos	Descripción
ENOMBRE	Texto	Nombre
CARGO	Texto	Título
ESUELDO	Número	Sueldo
DEPT	Texto	Departamento

ENOMBRE	CARGO	ESUELDO	DEPT
ARQUÍMEDES	AYTE. LAB.	200	ENG
DA VINCI	AYTE. LAB.	500	
DICK NIX	LADRÓN	25001	PHIL
EUCLIDES	AYTE. LAB.	1000	MATH
HANK KISS	BUFÓN	25000	PHIL
JUAN	EVANG4	54	THEO
LUCAS	EVANG1	53	THEO
MARCOS	EVANG2	52	THEO
MATEO	EVANG3	51	THEO

Las longitudes de los campos son 15, 10, entero largo y 4 respectivamente.

Una vez definidas todas las relaciones entre las tablas de la base de datos podemos realizar ya un diagrama completo que refleje su diseño final (Figura 6.4).

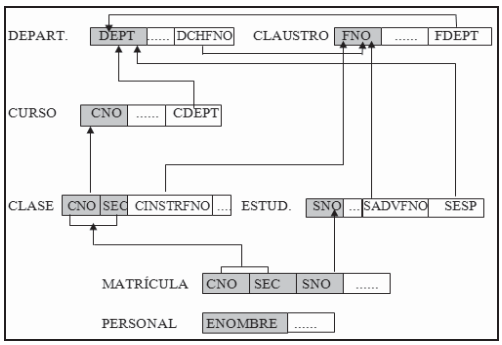


Figura 6.4

Hasta ahora, solo hemos considerado el aspecto estructural de la integridad referencial. Destacaremos, ahora, algunas de las reglas de procesamiento que especifican la respuesta del sistema a la sentencia de manipulación de datos del SQL, la cual referencia valores de clave primaria y/o secundaria. Estas sentencias (INSERT, UPDATE y DELETE) se describirán en capítulos posteriores. Adicionalmente tendremos en cuenta para la integridad referencial lo siguiente:

- *Inserción de la clave primaria*: el sistema verificará que el valor de la clave primaria es único. Si no lo es, la inserción será rechazada.

- *Inserción de la clave secundaria*: el sistema verificará que cualquier clave secundaria tenga el mismo valor que alguna clave primaria existente en la tabla referenciada. Si esto no se cumple, la inserción será rechazada.

- *Actualización de la clave primaria*: el sistema rechazará la actualización si existe un registro dependiente que referencie la clave primaria.

- *Actualización de la clave secundaria*: el sistema verificará que el nuevo valor de la clave secundaria coincida con alguna clave primaria existente en la tabla referenciada. Si esto no ocurre, la actualización se rechazará.

- *Borrado de la clave primaria*: se puede dirigir la respuesta del sistema a la eliminación de un registro que tenga una clave primaria igual a alguna clave secundaria correspondiente. Las opciones disponibles son: rechazar cualquier operación de borrado (RESTRICT), sustituir automáticamente los valores nulos por los valores correspondientes de la clave secundaria (SET NULL) y borrar automáticamente cualquier registro dependiente que tenga una clave secundaria igual a la primaria (CASCADE).

- *Borrado de una clave secundaria*: no existen restricciones (a menos que la clave secundaria sea parte de la primaria en la que se apliquen otras restricciones).

El diseñador de la base de datos debe prestar una cuidadosa atención a estas reglas de procesamiento. Existen restricciones e implicaciones adicionales para las tablas descendentes, especialmente aquéllas afectadas por relaciones cíclicas.

El diseño completo de las relaciones existentes entre las tablas (Figura 6.4) y las restricciones de integridad son el corazón de una base de datos.

6.3 CREACIÓN DE BASES DE DATOS EN ACCESS

Hay varias vías para crear una base de datos en Access 2010. Es posible crear directamente una base de datos en blanco cuando se inicia el programa, crear una base de datos a partir de una plantilla existente cuyo origen puede ser incluso Microsoft Office Online y crear una base de datos en blanco desde el botón *Archivo*. También es posible crear una base de datos web. Para *crear una nueva base de datos en blanco*, inicie Access y en el botón *Archivo* elija *Nuevo* y haga clic en *Base de datos en blanco* (Figura 6.5) o en *Base de datos web en blanco* si se crea la base en la web.

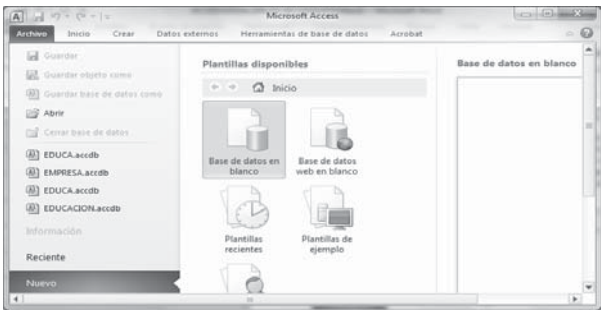


Figura 6.5

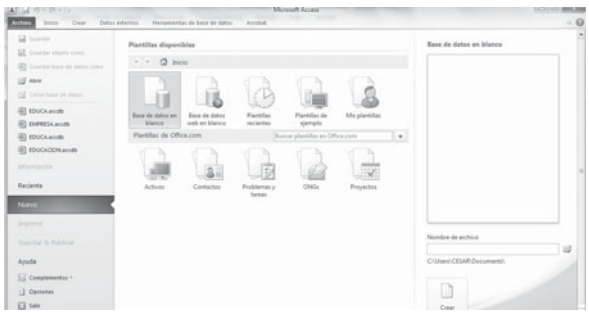


Figura 6.6

Bajo el panel *Base de datos en blanco*, en el cuadro *Nombre de archivo*, escriba un nombre de archivo (Figura 6.6). Por último, haga clic en *Crear*. Se crea la base de datos y se abre una tabla en la vista *Hoja de datos* (Figura 6.7), lo que permite crear ya las tablas.

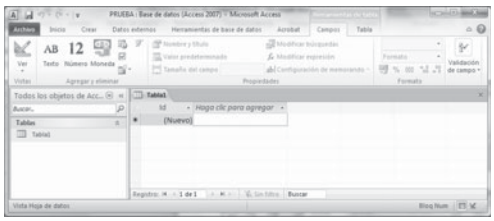


Figura 6.7

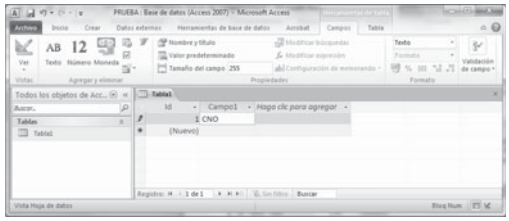


Figura 6.8

Hemos visto que cuando se crea una nueva base de datos, se abre la tabla de nombre *Tabla1* (por defecto) en la vista *Hoja de datos*. Es posible agregar inmediatamente campos a la tabla haciendo clic sobre *Haga clic para agregar* en la Figura 6.7 y escribiendo su nombre. Se pulsa Intro y *Haga clic para agregar* se desplaza a la siguiente columna a la derecha (Figura 6.8). Para introducir un nuevo campo en la tabla se vuelve a hacer clic en *Haga clic para agregar* y se escribe su nombre y así sucesivamente se pueden ir añadiendo más campos hasta completar todos los campos de la tabla (Figura 6.9). La siguiente tarea será introducir los datos en los campos como si se tratase de una hoja de cálculo (Figura 6.10).

De momento los tipos de datos y sus propiedades los asigna Access por defecto de acuerdo a la información que vamos almacenando en la tabla. No obstante, los tipos de datos podrán seleccionarse a medida tal y como veremos más adelante.

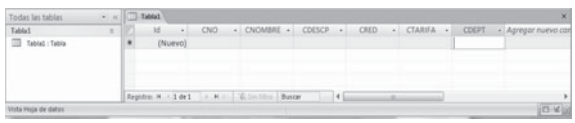


Figura 6.9

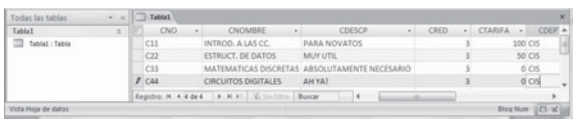


Figura 6.10

Una vez completados los datos de una tabla, puede guardarse a través de la opción *Guardar* del menú emergente que se obtiene al hacer clic con el botón derecho del ratón sobre la ficha que contiene el nombre de la tabla (Figura 6.11). La primera vez que se guarda se obtiene la pantalla *Guardar como* en la que podemos introducir un nombre para la tabla (Figura 6.12). Al hacer clic en *Aceptar* se guarda la tabla con ese nombre.

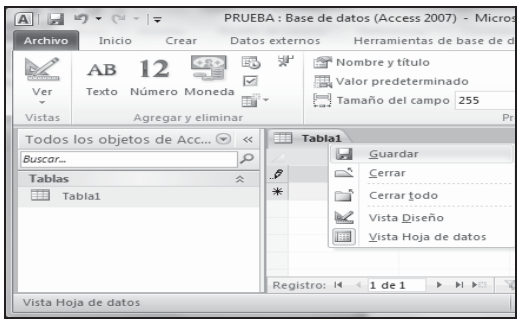


Figura 6.11

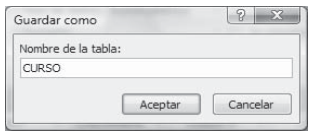


Figura 6.12

Para *guardar la base de datos* se hace clic en el botón *Archivo* y a continuación en *Guardar base de datos como* para elegir el formato adecuado al guardar y el nombre adecuado para la base de datos. La opción *Guardar* guarda la base de datos con el nombre y formato actuales.

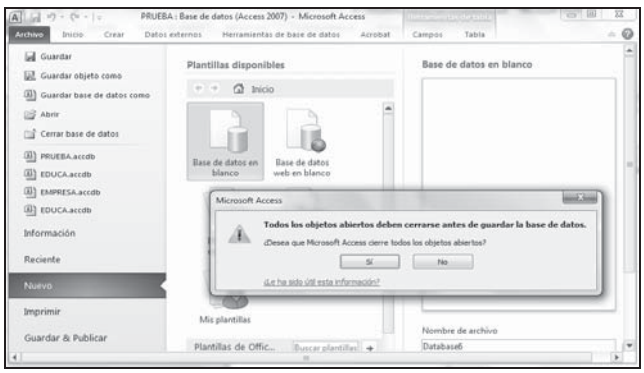


Figura 6.13

También es posible *crear una nueva base de datos en blanco en la web desde el botón Archivo*. Para ello, inicie Access desde el menú *Inicio* o desde un acceso directo. Haga clic en el botón *Archivo* y elija *Nuevo* en la Figura 6.13. Bajo *Plantillas disponibles* haga clic en *Base de datos web en blanco* y en el cuadro *Nombre de archivo*, escriba un nombre de archivo. Haga clic en *Crear*. Se crea una nueva base de datos en formato web y se abre una nueva tabla en la vista *Hoja de datos*. A partir de ahora se introducen los datos tal y como ya hemos visto anteriormente.

Adicionalmente, también es habitual *crear una base de datos a partir de una plantilla*. Para ello inicie Access, haga clic en el botón *Archivo* y a continuación en *Nuevo*. Bajo *Plantillas disponibles*, haga clic en una categoría adecuada a sus necesidades y, cuando aparezcan las plantillas de esa categoría, haga clic en la plantilla que más se acomode a su finalidad. Por ejemplo, podemos elegir la categoría *Plantillas de ejemplos* de la Figura 6.14. En la Figura 6.15 elegimos *Base de datos web de contactos* ya que queremos elaborar una base de datos de contactos de la empresa (clientes, socios, etc.). En el cuadro *Nombre de archivo* situado a la derecha, escriba un nombre de archivo o use el nombre proporcionado por defecto (Figura 6.15). Al hacer clic en *Crear*, Access descarga automáticamente la plantilla, crea una nueva base de datos a partir de esa plantilla y abre la base de datos. A continuación, al hacer clic en la solapa *Hoja de datos*, ya se puede introducir la información en las tablas y usar los objetos (Figura 6.16) que podrán ser modificados y adecuados a nuestras necesidades pudiendo cambiar incluso la propia plantilla.

Figura 6.14

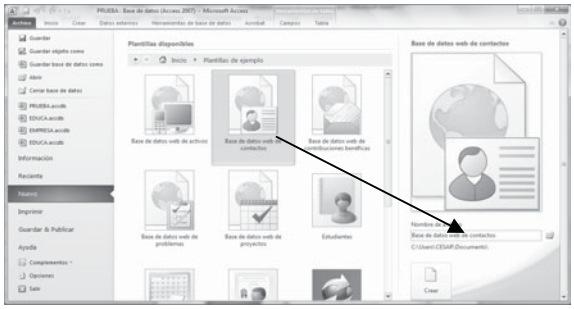


Figura 6.15

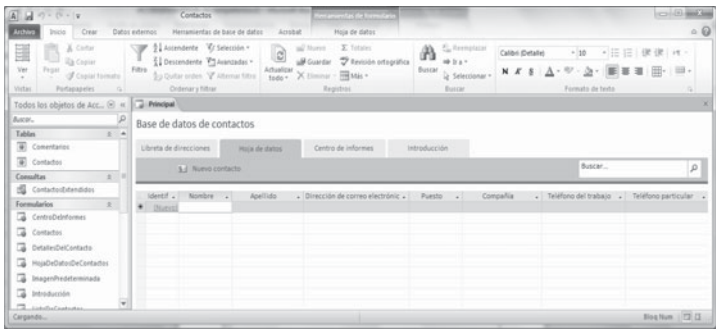


Figura 6.16

Además de las plantillas locales que se instalan por defecto en Access 2010, es posible *utilizar plantillas adicionales que Microsoft pone a disposición de los usuarios a través de Microsoft Office Online* directamente desde la interfaz de usuario de Access 2010. Para utilizar estas plantillas inicie Access, haga clic en el botón *Archivo* y elija *Nuevo*. En la Figura 6.14, bajo *Plantillas de Office.com*, haga clic en la plantilla que se desee (*Activos*). En el cuadro *Nombre de archivo* situado a la derecha, escriba un nombre de archivo o use el nombre por

defecto (Figura 6.17). Al hacer clic en *Descargar*, después de introducir los datos de usuario, Access incorpora la plantilla (Figura 6.18).

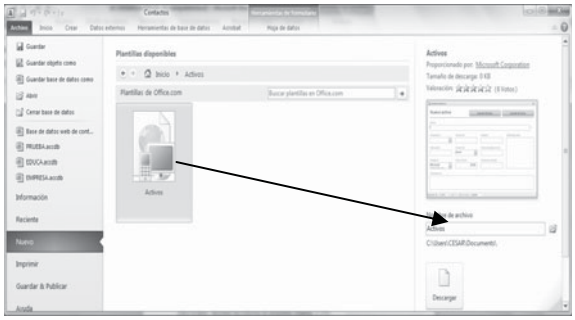


Figura 6.17

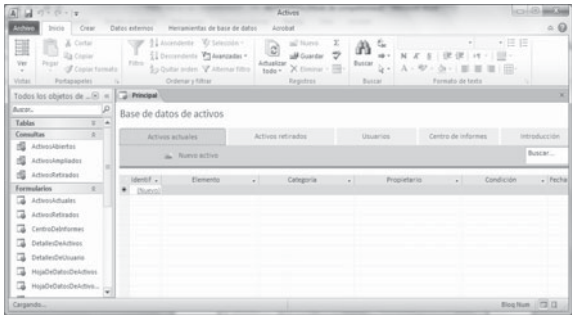


Figura 6.18

Si en la Figura 6.14 hacemos clic en *Buscar plantillas en Office.com* se pueden utilizar plantillas adicionales de la web de Microsoft.

6.4 VISTA HOJA DE DATOS. ADMINISTRACIÓN DE TABLAS

En los párrafos anteriores ya hemos visto como Access 2010 permite crear directamente una tabla en la vista *Hoja de datos* al crear la base de datos. También hemos visto cómo introducir registros en la tabla y guardarla con el nombre deseado una vez finalizado el trabajo de introducción de datos. También hemos visto cómo guardar la propia base de datos con el nombre adecuado. En este apartado nos ocuparemos de la administración de las tablas de la base de datos ya creada y del trabajo habitual con sus campos y registros.

6.4.1 Añadir nuevas tablas a una base de datos

Para *agregar una nueva tabla* a una base de datos se hace clic en *Tabla* en la ficha *Crear* (Figura 6.19). Se obtiene la *vista Hoja de datos* (Figura 6.20) con la nueva tabla de nombre *Tabla1* por defecto en la que se puede introducir sus campos y registros y guardarla por el mismo procedimiento utilizado para la tabla CURSO creada anteriormente en el epígrafe de creación de bases de datos. De la misma forma se completarán las sucesivas tablas de la base de datos.

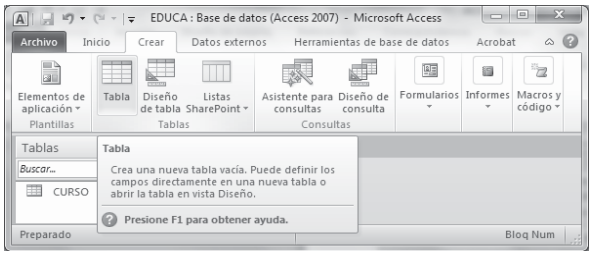


Figura 6.19

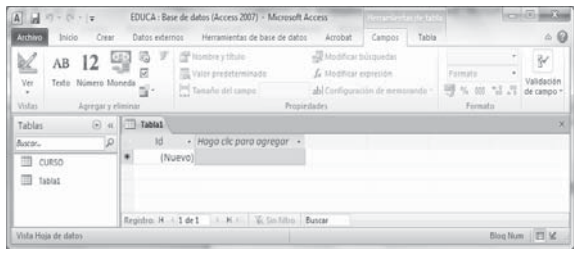


Figura 6.20

Una vez completadas todas las tablas de la base de datos, cuyos datos ya fueron expuestos en el diseño del capítulo anterior (Figura 6.21), puede guardarse la base de datos como una *Base de datos de Access 2010* mediante la opción *Guardar base de datos como* del botón *Archivo* con el nombre EDUCA. El fichero que se genera tiene de nombre EDUCA.ACCDB. Si se quiere, también puede guardarse la base de datos en formatos de versiones anteriores (*Base de datos de Access 2002-2003* y *Base de datos de Access 2000*) eligiendo la correspondiente opción del menú *Guardar & Publicar* del botón *Archivo* (Figura 6.22). El fichero que se genera en estos dos últimos casos tiene de nombre EDUCA.MDB. Obsérvese el cambio de extensión de los ficheros de bases de datos que se experimenta en la versión 2010 de Access.

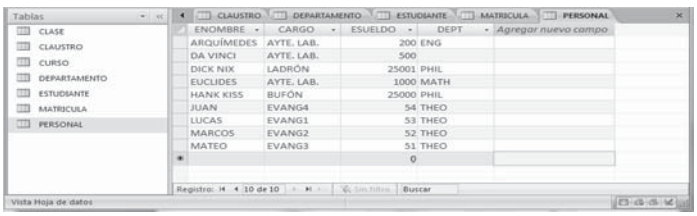


Figura 6.21

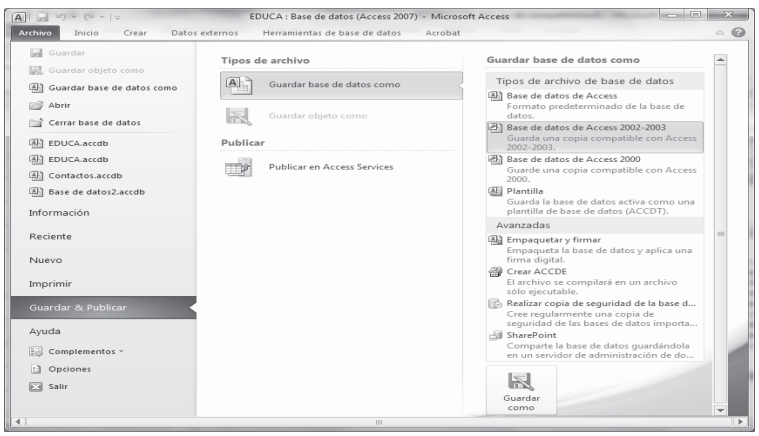


Figura 6.22

6.4.2 Añadir nuevos campos a las tablas de una base de datos

Podemos *añadir nuevos campos a las tablas* utilizando la opción *Agregar nuevo campo* (Figura 6.20) tal y como se ha hecho hasta ahora, pero cuando se crea una tabla nueva en la base de datos, aparece la barra *Herramientas de tabla*, cuyas fichas *Campos* y *Tablas* contienen grupos cuyas opciones permiten trabajar con campos, registros, formatos, tipos de datos y relaciones (Figuras 6.23 y 6.24). Concretamente, las opciones del grupo *Agregar y eliminar* de la Figura 6.23 permiten agregar y eliminar todo tipo de campos a las tablas.

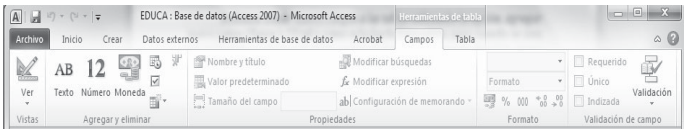


Figura 6.23

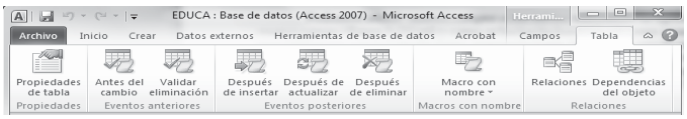


Figura 6.24

Para *agregar campos a una tabla existente en la vista Hoja de datos* puede utilizarse la barra *Herramientas de tabla*. Para ello, en la ficha *Campos*, en el grupo *Agregar y eliminar* de la Figura 6.23, haga clic en el tipo de campo que desea agregar. También puede utilizar la opción *Más campos* y seleccionar de la lista el tipo de campo a crear (Figura 6.25). El nuevo campo aparece en la hoja de datos (Figura 6.26). Posteriormente podrá cambiarse su nombre como veremos luego.

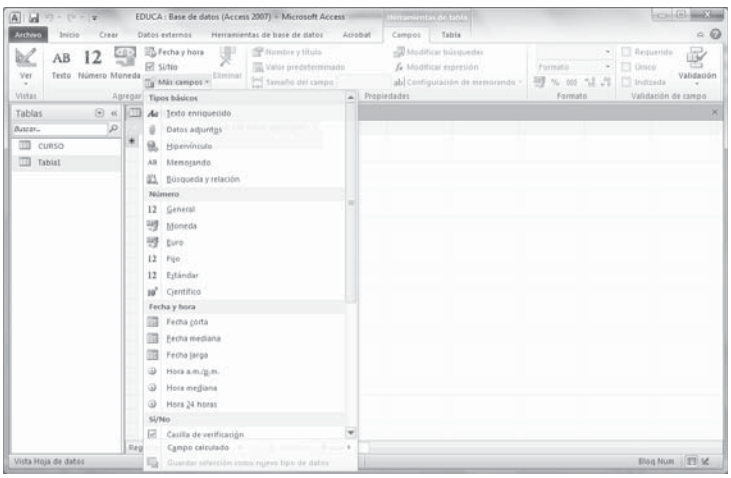


Figura 6.25

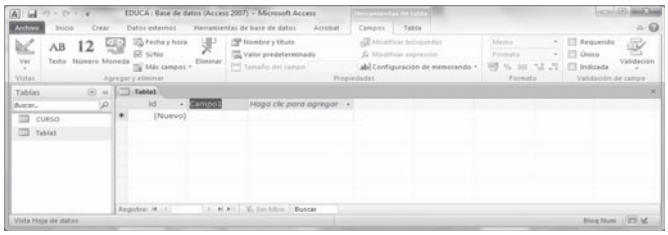


Figura 6.26

6.4.3 Insertar, eliminar y cambiar nombre a los campos

Las opciones *Insertar campo, Eliminar campo* y *Cambiar nombre de campo* del menú emergente que se obtiene al hacer clic con el botón derecho del ratón sobre un campo seleccionado en la tabla en la vista *Hoja de datos* (Figura 6.27) permiten insertar, eliminar y cambiar el nombre a los campos seleccionados en las tablas de la base de datos.

Para *insertar un campo en un lugar específico de una tabla*, se selecciona el campo situado a la derecha del punto de inserción deseado y a continuación se hace clic en *Insertar campo* en el menú emergente de la Figura 6.27. Se produce la inserción del campo con el nombre *Campo1* por defecto (Figura 6.28), que posteriormente puede ser cambiado.

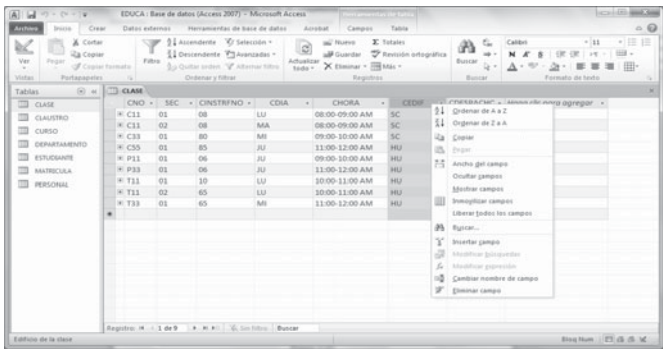


Figura 6.27

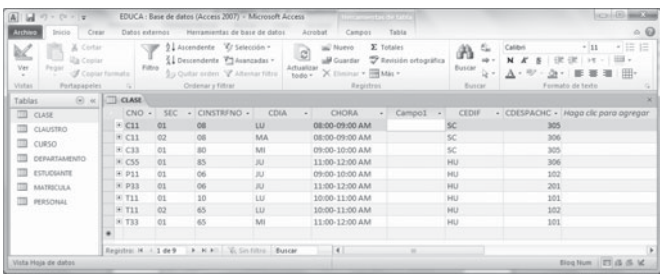


Figura 6.28

Para *cambiar el nombre a un campo de una tabla*, se selecciona el campo y a continuación se hace clic en *Cambiar nombre de campo* en el menú emergente de la Figura 6.27. A continuación se escribe el nuevo nombre de campo encima del antiguo (Figura 6.29). Los nombres de campo pueden contener un máximo de 64 caracteres (letras y números) incluyendo espacios.

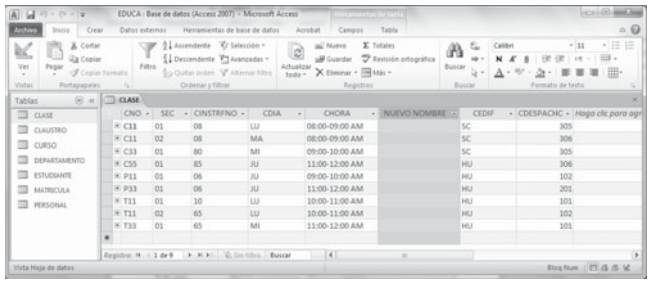


Figura 6.29

Para *eliminar uno o varios campos de una tabla*, se seleccionan los campos a borrar y a continuación se hace clic en *Eliminar campo* en el menú emergente de la Figura 6.30. Se obtiene la pantalla de la Figura 6.31 que pide la confirmación del borrado. Al hacer clic en *Sí*, se produce la eliminación de los campos. En la Figura 6.32 se observan ya los campos de la tabla sin incluir los borrados.

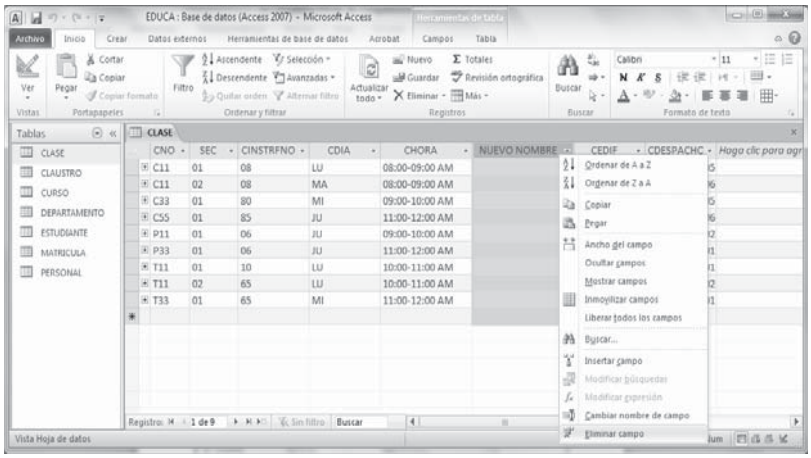


Figura 6.30

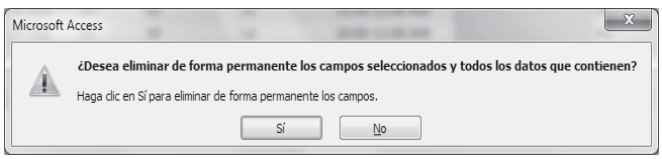


Figura 6.31

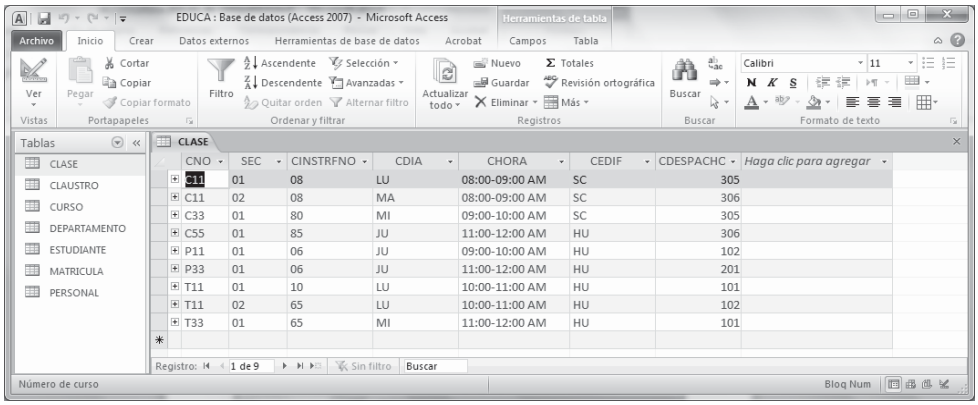


Figura 6.32

6.4.4 Tipos de datos, formatos y propiedades de campos

En Access 2010, en la vista *Hoja de datos*, podemos elegir el tipo de datos a medida para un determinado campo. Para ello se selecciona el campo sobre la tabla y en la barra *Herramientas de tabla*, en la ficha *Campos*, en el grupo *Formato*, hacemos clic en *Tipo de datos*. En el desplegable de la Figura 6.33 haga clic en el tipo de datos que desee.

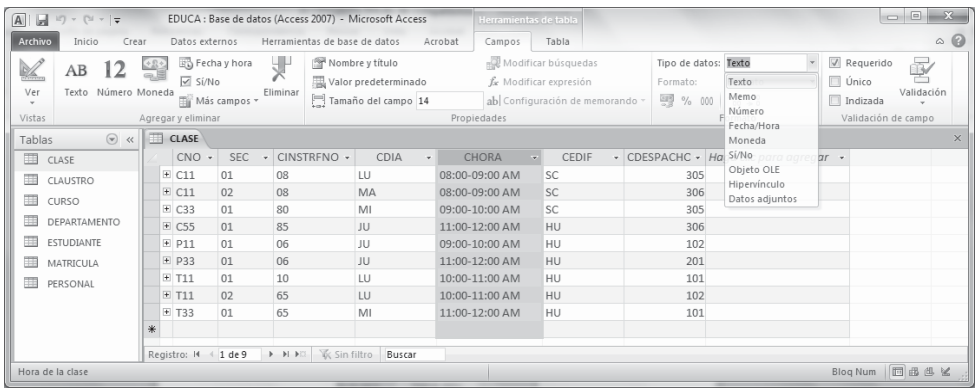


Figura 6.33

Existen determinados tipos de datos que admiten formato. Cuando el tipo de datos elegido para un campo admite formato (por ejemplo el tipo *Numérico*), es posible elegir dicho formato a medida. Para ello se selecciona el campo sobre la tabla y en la ficha *Campos*, en el grupo *Formato*, haga clic en *Formato*. En el desplegable de la Figura 6.34 haga clic en el formato que desee.

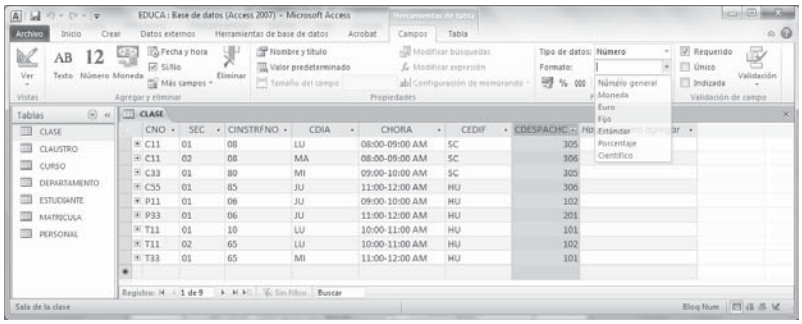


Figura 6.34

Cada campo de una tabla tiene un grupo de propiedades (requerido, indizado, etc.) que dependen del tipo de datos del campo. Para definir propiedades de campos en la vista *Hoja de datos* haga clic en el campo para el que desea definir la propiedad y en la ficha *Campos*, en el grupo *Validación de campos* (Figura 6.35), seleccione *Único* para definir la propiedad de campo con valores únicos, *Indizada* para campos indizados y *Requerido* para activar o desactivar el valor de la citada propiedad para el campo.

También pueden restringirse mediante expresiones los valores a escribir en un campo utilizando la opción *Regla de validación de campo* del grupo *Validación de campo* de la ficha *Campos* de la barra de *Herramientas de tabla* (Figura 6.36).

En la vista *Diseño* pueden precisarse mucho más los tipos de datos, formatos y propiedades de los campos, tal y como veremos posteriormente.

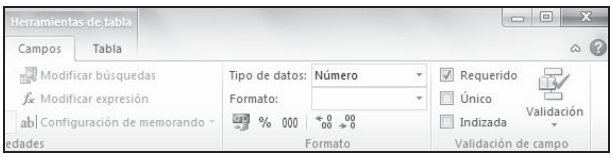


Figura 6.35

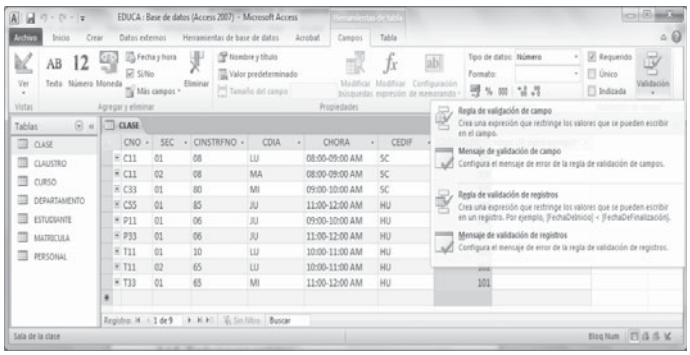


Figura 6.36

6.5 VISTA DISEÑO. ADMINISTRAR CLAVES, ÍNDICES, TIPOS DE DATOS Y PROPIEDADES

En la vista *Diseño* trabajamos únicamente con la estructura de las tablas (nombres de campos, tipos de datos, propiedades, etc.), pero no con sus contenidos. En la vista *Hoja de datos* normalmente trabajamos con los contenidos de la tabla, es decir, con sus datos. Para pasar de la vista *Hoja de datos* a otra mientras una tabla está abierta, haga clic sobre la flecha del botón *Ver* del grupo *Vistas* de la ficha *Inicio* y en el desplegable de la Figura 6.37 elija la vista que desee. Si elegimos *Vista Diseño* se obtiene la tabla actual en la vista *Diseño* (Figura 6.38). También podemos alternar entre la vista *Diseño*, la vista *Hoja de datos* y otras vistas, eligiendo los iconos correspondientes situados a la derecha de la barra de estado .

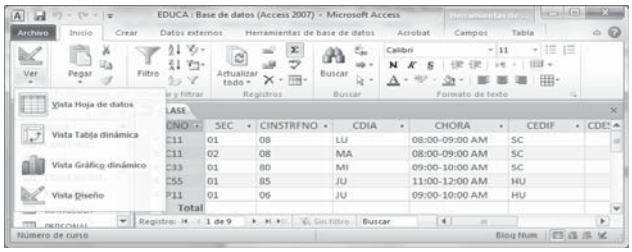


Figura 6.37

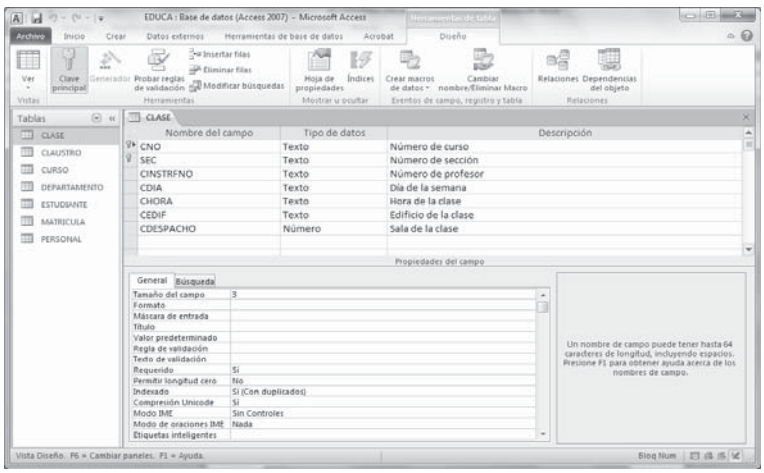


Figura 6.38

En la vista *Diseño* podemos ver y modificar la estructura fundamental de la tabla. Los nombres de los campos aparecen listados bajo la columna de la izquierda, y el tipo de datos, la descripción y las propiedades de campo aparecen también a lo largo de la tabla, pero los datos de la tabla no son visibles.

Al seleccionar una tabla en la vista *Diseño* aparece la barra *Herramientas de tabla* con la ficha *Diseño*, cuyos grupos y opciones permiten la gestión del diseño y se observan en la Figura 6.39.

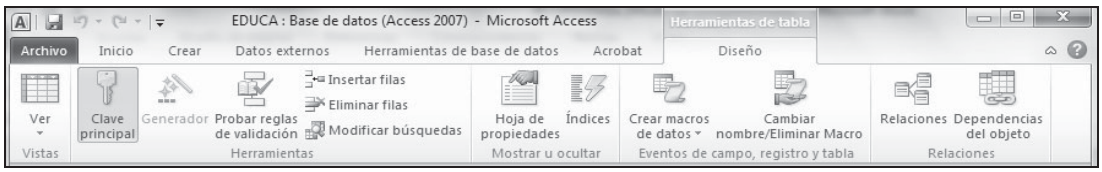


Figura 6.39

La opción *Ver* del grupo *Vistas* de la ficha *Diseño* permite cambiar entre tipos de vistas para la tabla (Figura 6.40). La opción *Clave principal* del grupo *Herramientas* permite establecer o quitar como clave principal de la tabla el campo o grupo de campos previamente seleccionados en la vista *Diseño*. La opción *Probar reglas de validación* advierte si los datos existentes infringen una regla de validación. En un principio se obtiene la advertencia de que se va a realizar la validación (Figura 6.41) y después de *Aceptar* se obtiene el resultado de la validación (Figura 6.42).

Las opciones *Insertar filas* y *Eliminar filas* añaden y quitan las filas seleccionadas en la pantalla de la vista *Diseño*. La opción *Columna de búsqueda* inserta una columna de búsqueda en el diseño. La opción *Hoja de propiedades* del grupo *Mostrar u ocultar* inserta la citada hoja en el diseño (Figura 6.43) y la opción *Índices* se utiliza para definir como índice en el diseño el campo o campos seleccionados.

Figura 6.40

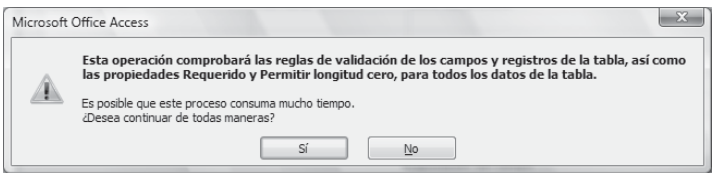


Figura 6.41

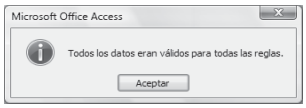


Figura 6.42

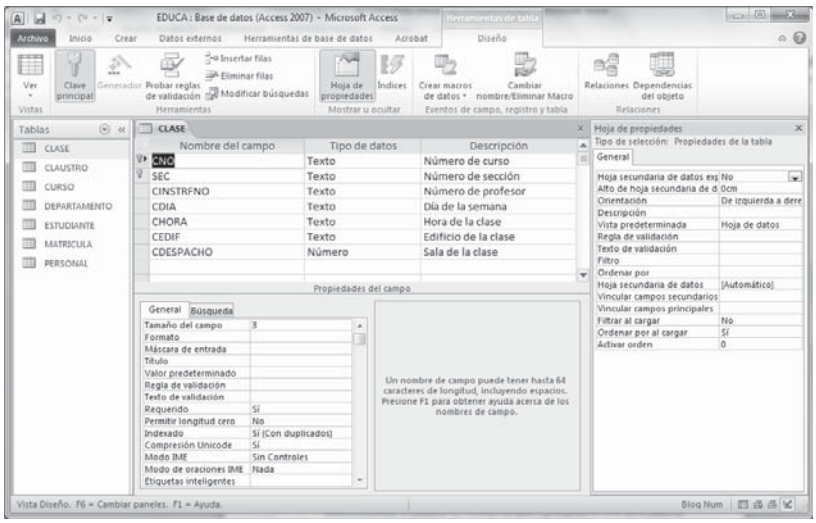


Figura 6.43

6.5.1 Establecer y quitar la clave primaria

Para *establecer o cambiar la clave principal o primaria de una tabla* abra la tabla en la vista *Diseño*, seleccione el campo o los campos que desea utilizar como clave principal y en la ficha *Diseño*, en el grupo *Herramientas*, haga clic en *Clave principal*. Se agrega un indicador de clave a la izquierda del campo o campos que ha especificado como clave principal.

En la Figura 6.44 hemos definido los campos CNO y SEC como clave principal para la tabla CLASE. Para seleccionar un campo, haga clic en el selector de filas del campo que desee. Para seleccionar varios campos para una clave compuesta, presione la tecla CTRL y haga clic en el selector de filas de cada campo. Access crea automáticamente un índice para la clave principal, que permite agilizar las consultas y otras operaciones. Access comprueba también que cada registro tiene un valor en el campo de clave principal y que éste es siempre distinto.

No olvidemos que la clave primaria de una tabla consta de uno o varios campos que identifican inequívocamente cada fila almacenada en la tabla (número de identificación exclusivo Id., número de serie o un código).

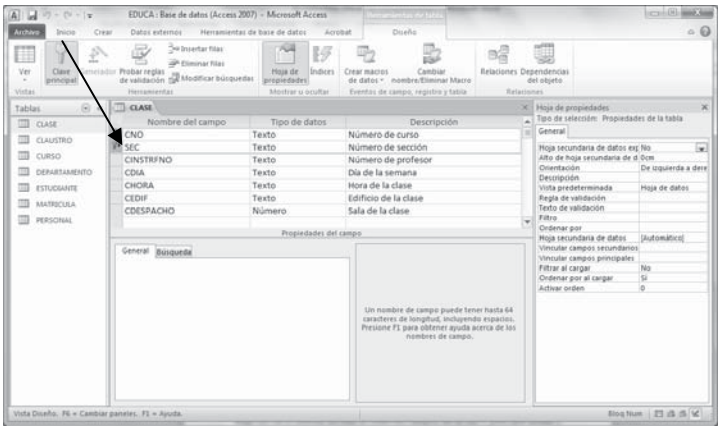


Figura 6.44

Para *quitar la clave principal* de una tabla abra la tabla en la vista *Diseño*, haga clic en el selector de filas de todos los campos de la clave principal actual y en la ficha *Diseño*, en el grupo *Herramientas*, haga clic en *Clave principal*.

El indicador de clave se quita del campo o campos de la clave principal. Al intentar guardar una tabla sin definir ninguna clave principal, Access le pedirá que cree una. Si elige *Sí,* se crea un campo Id. con el tipo de datos *Autonumérico* como clave principal, salvo que la tabla ya incluya un campo autonumérico, en cuyo caso se utilizará como clave principal.

6.5.2 Establecer y quitar índices

Como ya sabemos, los índices se utilizan para buscar y ordenar registros con mayor rapidez. Un índice almacena la ubicación de los registros basándose en el campo o campos que se decidan indizar. Una vez que Access ha obtenido la ubicación en el índice, puede recuperar los datos al ir directamente a la ubicación correcta. De esta forma, utilizar un índice puede ser mucho más rápido que buscar en todos los registros para encontrar los datos. Un índice puede utilizar uno o varios campos. Habitualmente se utilizará un campo o grupo de campos como índice si se prevé que se buscarán valores almacenados en esos campos o que se realizarán ordenaciones por los valores de los campos. Es conveniente que los campos que componen el índice tengan la mayoría de sus valores distintos.

Para *crear un índice de un solo campo*, muestre la tabla en la que desea crear el índice en la vista *Diseño*, haga clic en el *Nombre de campo* del campo que desea indizar y en *Propiedades del campo*, haga clic en la ficha *General*. En la propiedad *Indizado*, haga clic en *Sí (con duplicados)* si desea permitir duplicados o en *Sí (sin duplicados)* para crear un índice único (Figura 6.45). Para guardar los cambios, haga clic en *Guardar* en la barra de herramientas de acceso rápido o presione CTRL+G.

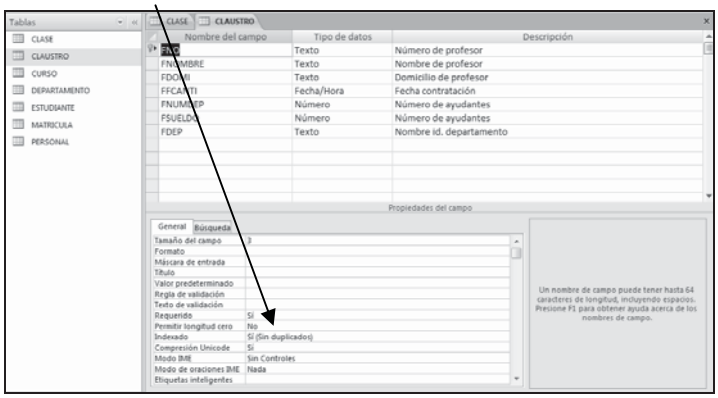


Figura 6.45

Para *crear un índice de varios campos* muestre la tabla en la que desea crear el índice en la vista *Diseño*, y en la ficha *Diseño*, en el grupo *Mostrar u ocultar*, haga clic en *Índices* (Figura 6.46). Se obtiene la ventana *Índices* (Figura 6.47). En la columna *Nombre del índice*, en la primera fila en blanco, escriba el nombre del índice. Puede utilizar el nombre de uno de los campos del índice o cualquier otro nombre. En la columna *Nombre de campo*, haga clic en la flecha y, a continuación, en el primer campo que desee utilizar para el índice. En la siguiente fila, deje en blanco la columna *Nombre del índice* y, a continuación, en la columna *Nombre de campo*, haga clic en el segundo campo del índice. Repita estos pasos hasta que haya seleccionado todos los campos que desee incluir en el índice. El orden predeterminado de los valores de los campos es ascendente (para cambiar el orden de los valores de un campo, en la columna *Criterio de ordenación* de la ventana *Índices*, haga clic en *Ascendente* o en *Descendente*). A continuación, en la ventana *Índices*, en *Propiedades del índice* (parte inferior de la Figura 6.47), establezca las propiedades del índice para la fila en la columna *Nombre del índice* que contiene el nombre del índice. Para guardar los cambios, haga clic en *Guardar* en la barra de herramientas de acceso rápido. Finalmente se cierra la ventana *Índices*. Un índice puede incluir hasta 10 campos.

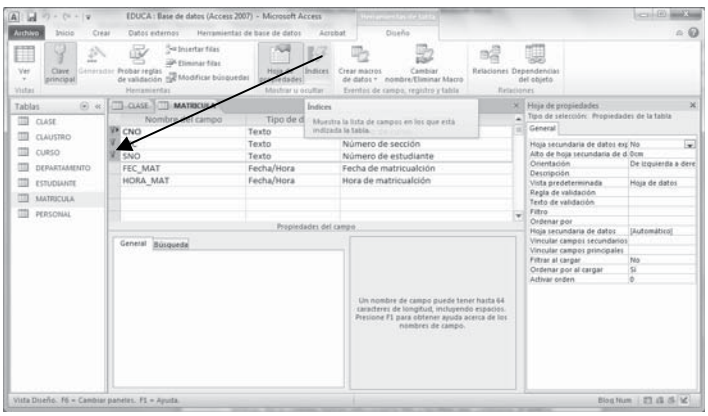


Figura 6.46

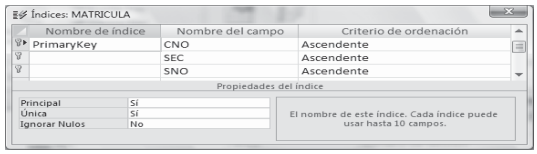


Figura 6.47

Para *eliminar un índice* muestre la tabla en la que desea eliminar el índice en la vista *Diseño* y en la ficha *Diseño*, en el grupo *Mostrar u ocultar*, haga clic en *Índices*. En la ventana *Índices* seleccione la fila o las filas que contienen el índice que desea eliminar y, a continuación, presione SUPR. Para guardar los cambios, haga clic en *Guardar* en la barra de herramientas de acceso rápido. Por último, cierre la ventana *Índices*.

Para *modificar un índice* muestre la tabla en la que desea modificar el índice en la vista *Diseño* y en la ficha *Diseño*, en el grupo *Mostrar u ocultar*, haga clic en *Índices*. En la ventana *Índices* vea y modifique los índices y las propiedades de índice de modo que se acomoden a sus necesidades. Para guardar los cambios, haga clic en *Guardar* en la barra de herramientas de acceso rápido. Por último, cierre la ventana *Índices*.

6.5.3 Tipos de datos y formatos

En la vista *Diseño* podemos ver y modificar la estructura fundamental de la tabla. Los nombres de los campos aparecen listados bajo la columna de la izquierda, y el tipo de datos, la descripción y las propiedades de campo aparecen también a lo largo de la tabla. Concretamente en la columna *Tipo de datos* se puede hacer clic en la flecha situada a la derecha del tipo de datos de cualquier campo y elegir un nuevo tipo de datos en el menú desplegable (Figura 6.48).

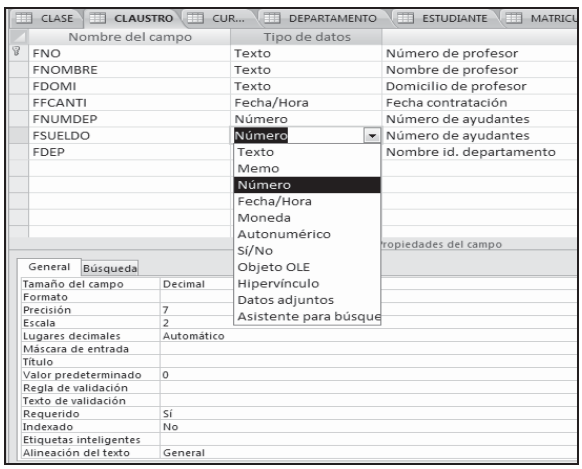


Figura 6.48

Adicionalmente, en la parte inferior de la Figura 6.48, en la zona de *Propiedades de campo*, se puede *elegir el formato preciso para el campo* seleccionado en la parte superior de la figura. En la Figura 6.48 se observa que el campo FSUELDO de la tabla CLAUSTRO es de tipo *Numérico* y concretamente *Decimal* con 7 dígitos en total (*Precisión*) y 2 decimales (*Escala*). Del mismo modo la vista *Diseño* permite definir de forma muy concreta el tipo de datos y el formato de cada campo de todas las tablas de la base de datos.

Los tipos de datos que se pueden asignar a un campo en Access se presentan en la tabla siguiente:

Tipo de datos	Utilización	Tamaño
Texto	Texto o combinaciones de texto y números como, por ejemplo, direcciones. Así mismo, números que no requieren cálculos como, por ejemplo, números de teléfono, números de pieza o códigos postales.	Hasta 255 caracteres. Access solo almacena los caracteres insertados en un campo y no almacena espacios de caracteres correspondientes a posiciones no usadas en un campo Texto.
Memo	Texto y números de gran longitud (más de 255 caracteres) como, por ejemplo, notas o descripciones.	Hasta 1 gigabyte de caracteres o 2 gigabytes de almacenamiento (se usan hasta 65.535 caracteres en un control).
Numérico	Datos numéricos que se han de utilizar para cálculos matemáticos, exceptuando los cálculos relacionados con dinero (utilice en este caso el tipo Moneda).	1, 2, 4 u 8 bytes. 16 bytes para Id. de réplica (GUID) exclusivamente.
Fecha/Hora	Fechas y horas.	8 bytes.
Moneda	Valores de moneda. Precisión de 15 dígitos a la izquierda del separador de decimales y de 4 dígitos a la derecha del mismo.	8 bytes.
Autonumérico	Números secuenciales exclusivos (con incremento de una unidad) o números aleatorios que se insertan automáticamente cuando se agrega un registro.	4 bytes. 16 bytes para Id. de réplica (GUID) exclusivamente.
Sí/No	Campos que van a contener solo uno de dos valores posibles, como Sí/No, Verdadero/Falso, Activado/Desactivado.	1 bit.
Objeto OLE	Objetos creados en otros programas mediante el protocolo OLE, que se pueden vincular a, o incrustar en, una tabla de Microsoft Access.	Hasta 1 gigabyte (limitado por el espacio en disco).

Hipervínculo	Campo que va a almacenar hipervínculos. Un hipervínculo puede ser una ruta UNC o una dirección URL.	Hasta 1 gigabyte de caracteres ó 2 gigabytes de almacenamiento (se usan hasta 65.535 caracteres en un control).
Asistente para búsquedas	Crea un campo que permite elegir un valor de otra tabla o de una lista de valores mediante el empleo de un cuadro combinado. La elección de esta opción en la lista de tipos de datos inicia un asistente que realiza la definición automáticamente.	El mismo tamaño que el campo de clave principal que también es el campo de búsqueda; normalmente 4 bytes.

6.5.4 Propiedades de campos

Las propiedades de los campos se utilizan para controlar la apariencia de la información, impedir que se especifiquen entradas incorrectas, especificar valores predeterminados, agilizar las operaciones de búsqueda y ordenación y controlar otras características de la apariencia y del funcionamiento definiendo propiedades de campo. Por ejemplo, puede aplicar formato a los números para facilitar su lectura o puede definir una regla de validación que deba satisfacerse para la información especificada en un campo.

Las propiedades de campo se sitúan en la parte inferior de la pantalla de la vista *Diseño* en la sección *Propiedades de campo* (Figura 6.49). En esta sección se presentan las propiedades de campo relativas al campo seleccionado en la parte superior de la pantalla.

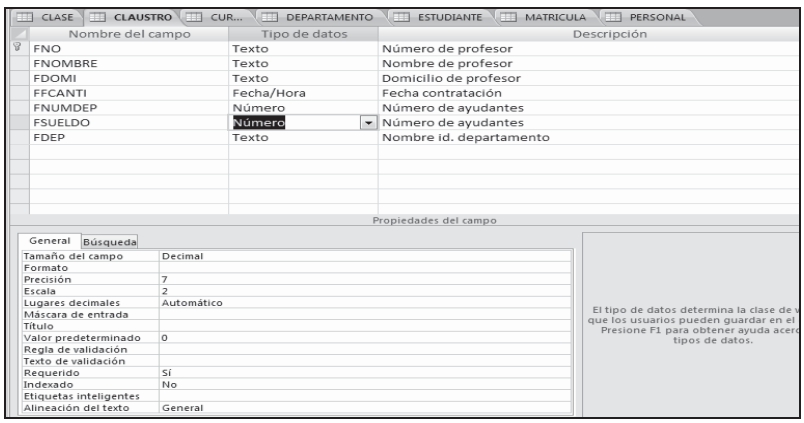


Figura 6.49

Para *definir una propiedad de campo para una tabla en la vista Diseño* abra la tabla en la vista *Diseño* y en la parte superior de la cuadrícula de diseño de la tabla, haga clic en el campo para el que desea definir propiedades (o desplácese a ese campo mediante las teclas de dirección). Access muestra las propiedades de este campo en la parte inferior de la

cuadrícula de diseño de la tabla (Figura 6.49). El tipo de datos del campo determina las propiedades que se pueden definir.

En la parte inferior de la ventana, bajo *Propiedades del campo*, haga clic en el cuadro de la propiedad del campo que desea definir y especifique un valor para la propiedad o, si aparece una flecha en el margen derecho del cuadro de propiedad, haga clic en la flecha para seleccionar un valor de una lista de valores de la propiedad (Figura 6.50).

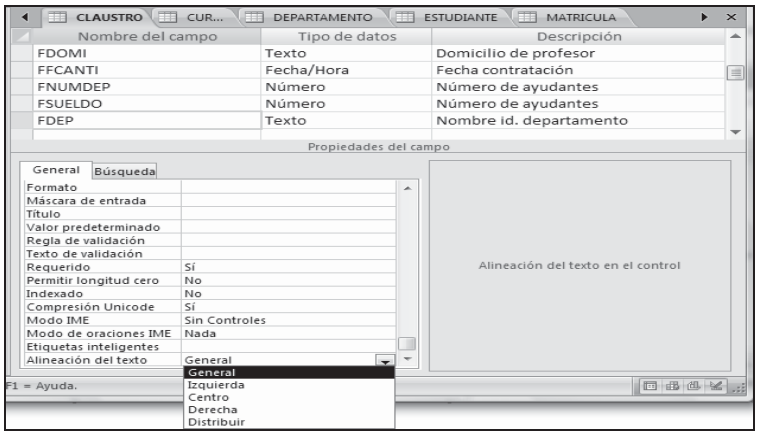


Figura 6.50

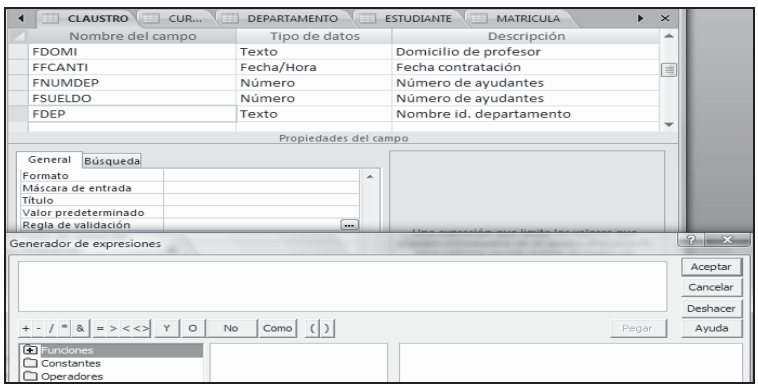


Figura 6.51

Si va a especificar una máscara de entrada o una expresión de validación y desea obtener ayuda, haga clic en ... situado junto al cuadro de propiedad para mostrar el generador correspondiente (Figura 6.51). Guarde los cambios haciendo clic en *Guardar* en la barra de herramientas de acceso rápido o haciendo clic con el botón secundario del ratón en la ficha de documento de la tabla y, a continuación, haciendo clic en *Guardar* en el menú contextual. También se puede hacer clic en el botón *Microsoft Office* y, a continuación, en *Guardar.*

La tabla siguiente resume las propiedades de campos más importantes.

Utilice esta propiedad de campo	Para
TamañoDelCampo	Definir el tamaño máximo de los datos almacenados con el tipo de datos Texto, Número o Autonumérico.
Formato	Personalizar el modo en que aparece el campo cuando se muestra o imprime.
LugaresDecimales	Especificar el número de posiciones decimales utilizadas al mostrar números.
NuevosValores	Indicar si el valor de un campo Autonumérico aumenta o recibe un valor aleatorio.
MáscaraDeEntrada	Mostrar caracteres de edición como guía para la entrada de datos.
Título	Definir el texto que se muestra de forma predeterminada en las etiquetas de los formularios, informes y consultas.
ValorPredeterminado	Asignar automáticamente un valor predeterminado a un campo cuando se agregan nuevos registros.
ReglaDeValidación	Proporcionar una expresión que debe ser verdadera cuando se agrega o cambia el valor del campo.
TextoDeValidación	Especificar el texto que aparece cuando un valor infringe la expresión ReglaDeValidación.
Requerido	Exigir que se especifiquen datos en un campo.
PermitirLongitudCero	Permitir que se especifique (estableciendo la propiedad en Sí) una cadena de longitud cero ("") en un campo Texto o Memo.
Indizado	Agilizar el acceso a los datos de un campo mediante la creación y utilización de un índice.
CompresiónUnicode	Comprimir el texto almacenado en este campo cuando se especifica una gran cantidad de texto (> 4.096 caracteres).
ModoIME	Controlar la conversión de caracteres en la versión asiática de Windows.
ModoDeOracionesIME	Controlar la conversión de caracteres en la versión asiática de Windows.
EtiquetasInteligentes	Anexar una tarjeta inteligente al campo.
SoloAnexar	Permitir el control de versiones (estableciendo la propiedad en Sí) de un campo Memo.

FormatoDeTexto	Seleccionar Texto enriquecido para almacenar texto como HTML y permitir el formato de texto enriquecido. Seleccionar Texto sin formato para almacenar solo texto.
AlineaciónDelTexto	Especificar la alineación predeterminada del texto dentro de un control.
Precisión	Especificar el número total de dígitos permitidos, incluidos los que aparecen a la derecha y a la izquierda de la coma decimal.
Escala	Especificar el número máximo de dígitos que pueden almacenarse a la derecha y a la izquierda de la coma decimal.

6.6 ADMINISTRAR RELACIONES

Las relaciones entre las tablas de una base de datos tienen como finalidad eliminar la duplicidad de datos (redundancia) y optimizar el diseño para conseguir que la respuesta a las consultas sea lo más rápida posible, así como optimizar las actualizaciones y el trabajo con formularios, informes que podrán utilizar información de varias tablas a la vez. Pero quizás la finalidad más importante de las relaciones es el mantenimiento de la integridad referencial en la base de datos. La integridad referencial es un sistema de reglas que utiliza Microsoft Access para garantizar que las relaciones entre los registros de tablas relacionadas son válidas y que no se eliminan ni modifican accidentalmente datos relacionados.

En el capítulo anterior ya habíamos visto que existen relaciones uno a uno, uno a varios y varios a varios. En una *relación uno a uno*, cada registro de la primera tabla solo puede tener un registro coincidente en la segunda tabla y viceversa.

Este tipo de relación no es común porque, muy a menudo, la información relacionada de este modo se almacena en la misma tabla. La *relación uno a varios* es el tipo de relación más común. En este tipo de relación, un registro de la primera tabla puede tener muchos registros coincidentes en la segunda tabla, pero un registro de la segunda tabla solo tiene un registro coincidente en la primera. En una *relación varios a varios*, un registro de la primera tabla puede tener muchos registros coincidentes en la segunda, y viceversa. Este tipo de relación solo es posible si se define una tercera tabla (denominada tabla de unión) cuya clave principal consta de, al menos, dos campos: las claves externas de las dos tablas.

El tipo de relación que crea Microsoft Access depende de cómo están definidos los campos relacionados. Se crea una relación uno a varios ($1 \rightarrow \infty$) si uno de los campos relacionados es una clave principal o tiene un índice único. Se crea una relación uno a uno si ambos campos relacionados son claves principales o tienen índices únicos. Una relación varios a varios es, en realidad, dos relaciones uno a varios con una tercera tabla cuya clave principal consta de dos campos: las claves externas de las otras dos tablas.

6.6.1 Definir relaciones entre tablas

Una vez creadas tablas diferentes para cada tema de la base de datos de Microsoft Access, necesita una forma de indicarle al programa cómo debe volver a combinar esa información. El primer paso de este proceso es definir relaciones entre las tablas. Para ello se tendrá en cuenta lo siguiente:

- Cierre todas las tablas que estén abiertas. No es posible crear ni modificar relaciones entre tablas abiertas.
- En la ficha *Herramientas de base de datos*, en el grupo *Relaciones*, haga clic en *Relaciones* (Figura 6.52). Se obtiene la pantalla *Relaciones* y la barra *Herramientas de relaciones* aparece en la parte superior de la pantalla con la ficha *Diseño* abierta.
- Si la base de datos no tiene ninguna relación definida, se mostrará automáticamente el cuadro de diálogo *Mostrar tabla* (Figura 6.53). Si necesita agregar las tablas que desea relacionar y no aparece el cuadro de diálogo *Mostrar tabla*, haga clic en *Mostrar tabla* en el grupo *Relaciones* de la ficha *Diseño*.
- Haga doble clic en los nombres de las tablas que desea relacionar y, a continuación, cierre el cuadro de diálogo *Mostrar tabla* (Figura 6.54).
- Arrastre las tablas hasta conseguir una estructura similar a la del diagrama de diseño de la base de datos (Figura 6.55). Para arrastrar una tabla se hace clic sobre su cabecera con el ratón y, sin soltarlo, se mueve el ratón hasta la posición en que se quiere situar la tabla. Una buena estructura acorde al diseño de la base de datos se presenta en la Figura 6.56.
- Para crear las relaciones entre campos arrastre el campo que desea relacionar de una tabla al campo relacionado de la otra tabla. Para arrastrar varios campos, presione la tecla CTRL y haga clic en cada campo antes de arrastrarlo. En la mayoría de los casos, se arrastra el campo de clave principal (mostrado en texto en negrita) de una tabla a un campo similar (normalmente con el mismo nombre), denominado clave externa de la otra tabla. Los campos relacionados no tienen que tener los mismos nombres, pero deben tener el mismo tipo de datos (con dos excepciones) y deben contener el mismo tipo de información. Además, cuando los campos coincidentes son campos *Numéricos*, deben tener el mismo valor de la propiedad *Tamaño del campo*. Las dos excepciones a los tipos de datos coincidentes son que se puede hacer coincidir un campo *Autonumérico* con un campo *Numérico* cuya propiedad *Tamaño del campo* esté establecida a *Entero largo*; y que se puede hacer coincidir un campo *Autonumérico* con un campo *Numérico* si ambos campos tienen la propiedad *Tamaño del campo* establecida a *Id. de réplica*.
- Aparecerá el cuadro de diálogo *Modificar relaciones* (Figura 6.57). Compruebe los nombres de los campos mostrados en las dos columnas para asegurarse de que

son correctos. Puede cambiarlos si es necesario. Si es preciso, establezca las opciones de relación. Para obtener información acerca de un elemento específico del cuadro de diálogo *Relaciones*, haga clic en el botón de signo de interrogación ? y, a continuación, haga clic en el elemento. El botón *Tipo de combinación* controla el tipo de combinación que desea utilizar de forma predeterminada cuando cree consultas basadas en las tablas relacionadas (Figura 6.58). El botón *Crear nueva...* permite definir una nueva relación entre dos tablas. La casilla *Exigir integridad referencial* permite aplicar integridad referencial para esta relación, aunque solo si el campo o campos coincidentes de la tabla principal son una clave principal o tienen un índice único, los campos relacionados tienen el mismo tipo de datos y ambas tablas están almacenadas en la misma base de datos de Access (la integridad referencial es un sistema de reglas que utiliza Microsoft Access para asegurar que las relaciones entre los registros de las tablas relacionadas son válidas y que no se eliminan o cambian los datos relacionados de forma accidental).

- Desactive esta casilla de verificación para permitir cambios en las tablas relacionadas que rompan las reglas de integridad referencial. Active la casilla *Exigir integridad referencial* y, a continuación, *Actualizar en cascada los campos relacionados* para actualizar automáticamente los valores correspondientes de la tabla relacionada cuando se cambie un valor de la clave principal en la tabla principal. Active la casilla *Exigir integridad referencial* y desactive *Actualizar en cascada los campos relacionados* para impedir que se realicen cambios en un valor de la clave principal en la tabla principal, siempre que haya registros relacionados en la tabla relacionada. Active la casilla *Exigir integridad referencial*, y a continuación, *Eliminar en cascada los registros relacionados* para eliminar automáticamente los registros relacionados en la tabla relacionada siempre que se elimine un registro de la tabla principal. Active la casilla *Exigir integridad referencial* y desactive *Eliminar en cascada los registros relacionados* para impedir que se eliminen registros de la tabla principal cuando existen registros relacionados en la tabla relacionada.

- Haga clic en el botón *Crear* para crear la relación (Figura 6.59) y repita los pasos 5 a 9 para cada pareja de tablas que desee relacionar (Figura 6.60). Al cerrar la ventana *Relaciones*, Microsoft Access pregunta si desea guardar el diseño. Tanto si lo guarda como si no, las relaciones creadas se guardan en la base de datos. En el ejemplo hemos activado siempre *Exigir integridad referencial*.

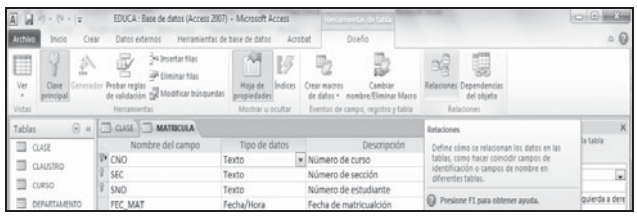


Figura 6.52

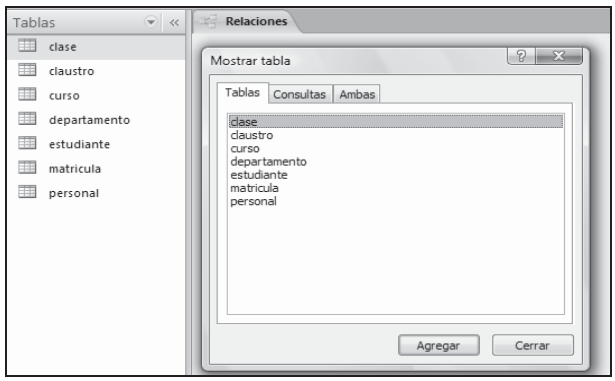


Figura 6.53

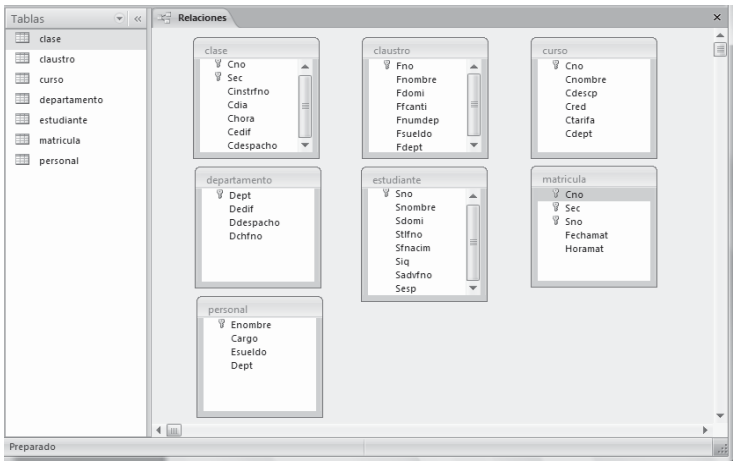


Figura 6.54

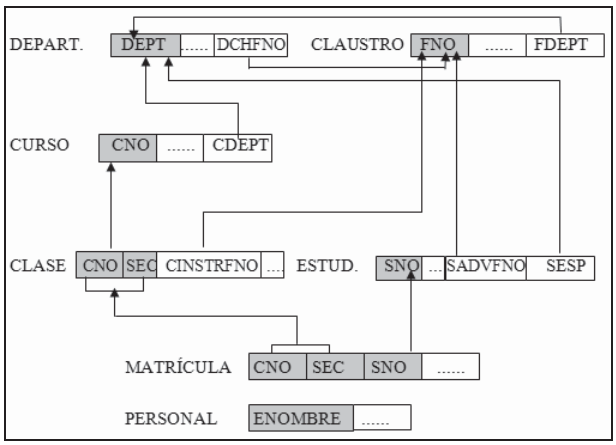


Figura 6.55

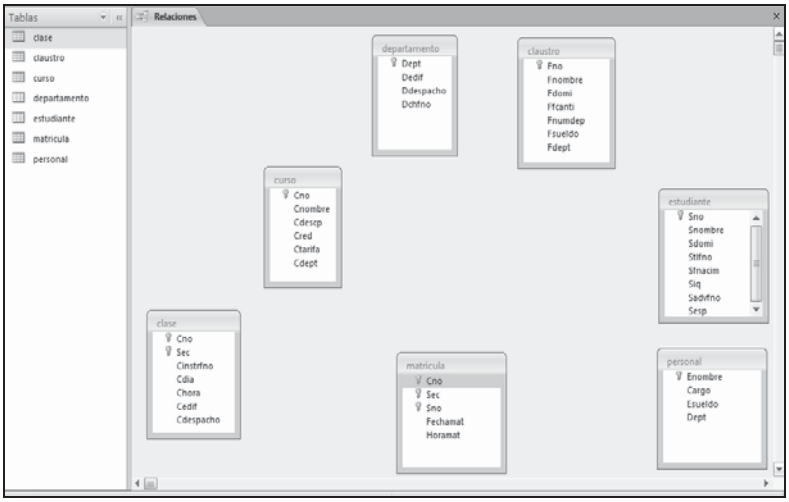


Figura 6.56

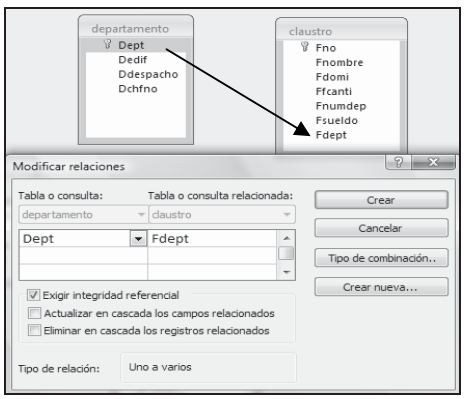


Figura 6.57

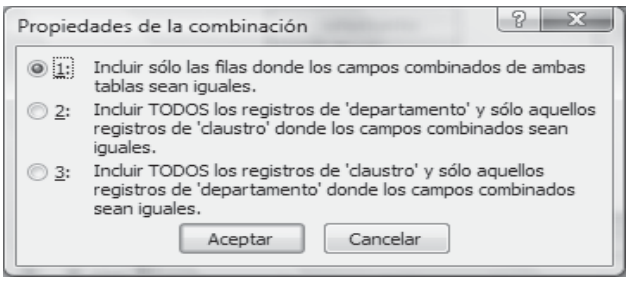


Figura 6.58

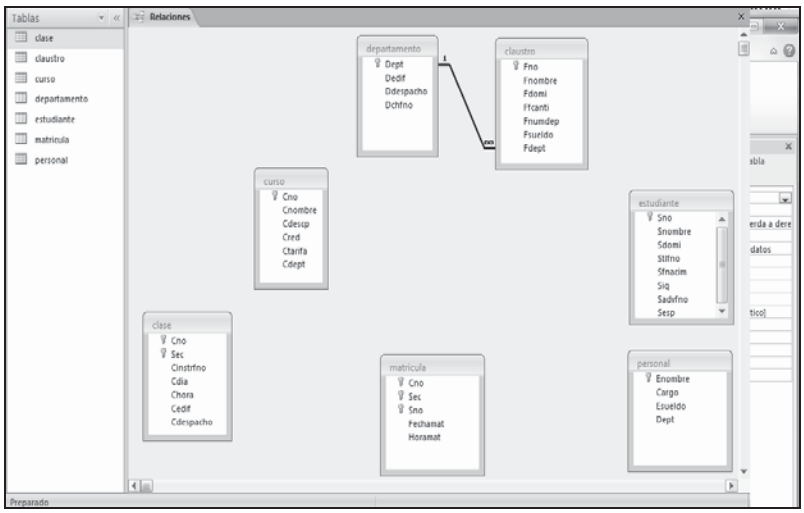


Figura 6.59

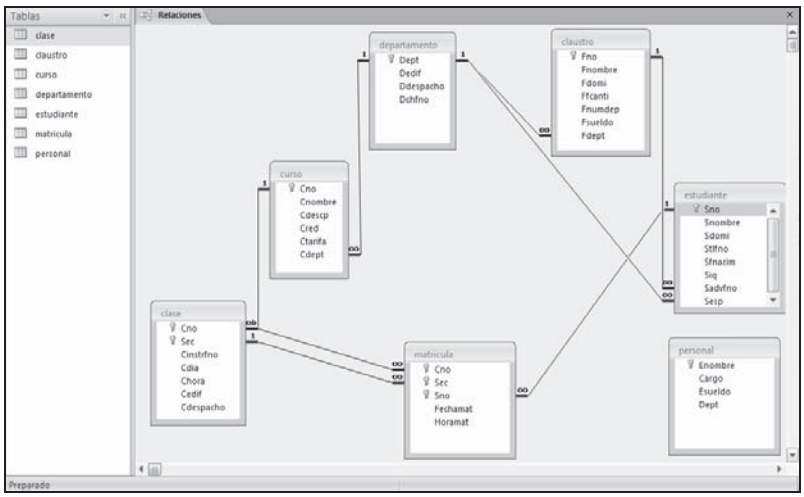


Figura 6.60

- Si necesita ver todas las relaciones definidas en la base de datos, haga clic en Mostrar todas las relaciones en el grupo Relaciones de la ficha Diseño de la barra Herramientas de relaciones. Para ver solo las relaciones definidas para una tabla determinada, haga clic en la tabla y, a continuación, en Mostrar relaciones directas.
- Si necesita realizar un cambio en el diseño de una tabla, podrá hacer clic con el botón secundario del ratón en la tabla que desea modificar y, a continuación, hacer clic en Diseño de la tabla.

- Puede crear relaciones utilizando tanto consultas como tablas. Sin embargo, la integridad referencial no se exige en las consultas.
- Para crear una relación entre una tabla y ella misma, agregue esta tabla dos veces. Esto resulta útil en situaciones en las que necesita realizar una búsqueda dentro de la misma tabla. Por ejemplo, en la tabla Empleados de la base de datos de ejemplo Neptuno, se ha definido una relación entre los campos Id. de empleado y Jefe, de modo que el campo Jefe pueda mostrar datos de los empleados procedentes de un Id. de empleado coincidente.

6.6.2 Eliminar y modificar relaciones

Para *eliminar una relación* se tendrá en cuenta lo siguiente:

- Cierre todas las tablas abiertas. No es posible eliminar relaciones entre tablas abiertas.
- En la ficha *Herramientas de base de datos*, en el grupo *Mostrar u ocultar*, haga clic en *Relaciones*. Se obtiene la pantalla *Relaciones* y la barra *Herramientas de relaciones* aparece en la parte superior de la pantalla con la ficha *Diseño* abierta.
- Si las tablas cuya relación desea eliminar no están mostradas en pantalla, haga clic en *Mostrar tabla* en la ficha *Herramientas de base de datos*, en el grupo *Mostrar u ocultar* y haga doble clic en cada tabla que desee agregar para modificar o eliminar relaciones. A continuación, elija *Cerrar*.
- Haga clic en la línea de relación correspondiente a la relación que desea eliminar (la línea se mostrará en negrita al seleccionarla) y, a continuación, presione la tecla SUPR. También se puede hacer clic con el botón derecho del ratón sobre la línea de la relación y elegir *Eliminar* en el menú emergente resultante (Figura 6.61). La opción *Borrar diseño* del grupo *Herramientas* de la ficha *Diseño* oculta todas las relaciones y tablas del diseño.
- Para *modificar una relación* se elige *Modificar* en el menú emergente de la Figura 6.61 o haga doble clic sobre la línea de la relación. También se puede hacer clic en la línea de relación y elegir *Modificar relación* en el grupo *Herramientas* de la ficha *Diseño*.
- Establezca las opciones de la relación en el cuadro de diálogo *Relaciones* resultante (Figura 6.62). Para obtener información acerca de un elemento específico del cuadro de diálogo *Relaciones*, haga clic en el botón de signo de interrogación y, a continuación, haga clic en el elemento.

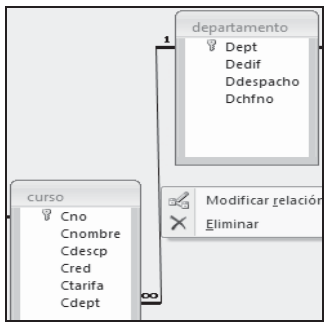


Figura 6.61

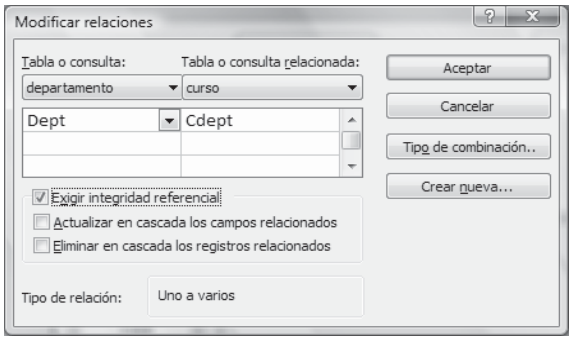


Figura 6.62

6.6.3 ADMINISTRACIÓN DE LA SEGURIDAD EN ACCESS

Access proporciona un modelo de seguridad mejorada que ayuda a simplificar el proceso de aplicar seguridad a una base de datos y de abrir una base de datos con la seguridad habilitada. La manera más segura de contribuir a proteger los datos de Access es almacenar las tablas en un servidor, como un equipo que ejecuta Windows SharePoint Services 3.0 o Microsoft Office SharePoint Server 2010.

Access incorpora en sus versiones más actuales el *Centro de confianza*. Se trata de un cuadro de diálogo que proporciona una sola ubicación para definir y cambiar la configuración de seguridad de Access. El Centro de confianza se usa para crear o cambiar las ubicaciones de confianza y configurar las opciones de seguridad de Access. Esa configuración afecta a la manera en que las bases de datos nuevas y existentes se comportan cuando se abren en esa instancia de Access. El Centro de confianza contiene así mismo la lógica para evaluar los componentes de una base de datos y determinar si es seguro abrir la base de datos o si el Centro de confianza debe deshabilitarla y dejar que el usuario decida si desea habilitarla.

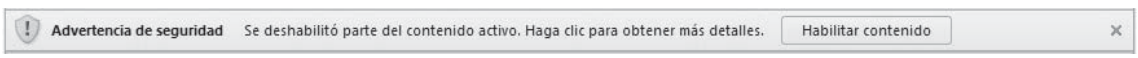
6.6.4 Arquitectura de seguridad de Access

Para comprender la arquitectura de seguridad de Access, recuerde que una base de datos de Access no es un archivo como un libro de Excel o un documento de Word. Una base de datos de Access es un conjunto de objetos, es decir, tablas, formularios, consultas, macros, informes, etc., que, a menudo, dependen los unos de los otros para su funcionamiento. Por ejemplo, si se crea un formulario de entrada de datos, no se pueden proporcionar ni almacenar datos con ese formulario a menos que se enlacen (vinculen) los controles del formulario a una tabla. Algunos componentes de Access pueden representar un riesgo para la seguridad, por lo que se deshabilitan en las bases de datos que no son de confianza:

- Consultas de acción (consultas que insertan, eliminan o modifican datos).
- Macros.
- Algunas expresiones (funciones que devuelven un valor único).
- Código VBA.

Para ayudar a que los datos sean más seguros, Access y el Centro de confianza realizan una serie de comprobaciones de seguridad cada vez que se abre una base de datos. El proceso es el siguiente:

- Cuando se abre un archivo .accdb o .accde, Access envía la ubicación de la base de datos al Centro de confianza. Si se trata de una ubicación de confianza, la base de datos se ejecutará con toda su funcionalidad. Si se abre una base de datos con un formato de archivo anterior, Access envía la ubicación del archivo y los detalles de la firma digital (si existe) al Centro de confianza. El Centro de confianza comprueba esas "pruebas" para evaluar si la base de datos es de confianza y, a continuación, informa a Access de cómo abrir la base de datos. Access deshabilita la base de datos, o bien, la abre con toda su funcionalidad. Recuerde que la configuración definida por el usuario o el administrador del sistema en el Centro de confianza controla las decisiones que se toman cuando Access abre una base de datos.
- Cuando el Centro de confianza evalúa que una base de datos no es de confianza, Access la abre en modo *Deshabilitado*, es decir, desactiva todo el contenido ejecutable, independientemente del formato de archivo de la base de datos. Si el Centro de confianza deshabilita contenido, aparece la barra de mensajes cuando se abre la base de datos. Para habilitar el contenido de la base de datos, haga clic en *Opciones* y, a continuación, elija las opciones correspondientes en el cuadro de diálogo que aparece. Access habilita el contenido deshabilitado y la base de datos vuelve a abrirse con toda su funcionalidad. En caso contrario, no funcionarán los componentes deshabilitados.



- Si se abre una base de datos creada con el formato de archivo anterior (.mdb o .mde) y esa base de datos no está firmada ni es de confianza, Access deshabilita todo el contenido ejecutable de forma predeterminada.

6.6.5 Usar una base de datos de Access en una ubicación de confianza

Si se coloca una base de datos de Access en una ubicación de confianza, todo el código de VBA, las macros y las expresiones seguras se ejecutan cuando se abre la base de datos. No es necesario tomar ninguna decisión en materia de confianza al abrir la base de

datos. El proceso de usar una base de datos de Access en una ubicación de confianza se compone de los siguientes pasos:

- Use el Centro de confianza para buscar o crear una ubicación de confianza.
- Guarde, mueva o copie una base de datos de Access a la ubicación de confianza.
- Abra y use la base de datos.

En los siguientes pasos se explica cómo buscar o crear una ubicación de confianza y, a continuación, agregar una base de datos a esa ubicación.

Para *abrir el Centro de confianza* se tendrá en cuenta lo siguiente:

- En la ficha *Archivo*, haga clic en *Opciones*. Aparece el cuadro de diálogo *Opciones de Access*.
- Haga clic en Centro de confianza y, a continuación, haga clic en Centro de confianza de Microsoft Office y elija Configuración del Centro de confianza.
- Haga clic en *Ubicaciones de confianza* y, a continuación anote la ruta de acceso de una o varias ubicaciones de confianza o cree una nueva ubicación de confianza. Para ello, haga clic en *Agregar nueva ubicación* y complete las opciones del cuadro de diálogo *Centro de confianza de Microsoft Office*.

Para *colocar una base de datos en una ubicación de confianza* use su técnica favorita para mover o copiar un archivo de base de datos a una ubicación de confianza. Por ejemplo, puede usar el Explorador de Windows para copiar o mover el archivo, o bien, puede abrir el archivo en Access y guardarlo en la ubicación de confianza.

Para *abrir una base de datos en una ubicación de confianza* use su procedimiento favorito para abrir un archivo. Por ejemplo, puede hacer doble clic en el archivo de base de datos en el Explorador de Windows o, si Access se está ejecutando, puede hacer clic en *Abrir* en la ficha *Archivo* para buscar y abrir el archivo.

6.6.6 Usar una contraseña para cifrar una base de datos de Access

La herramienta de cifrado de Access combina y mejora dos herramientas anteriores: el cifrado y las contraseñas para bases de datos. Cuando se usa una contraseña para cifrar una base de datos, todos los datos se vuelven ilegibles para otras herramientas y se obliga a los usuarios a escribir una contraseña para poder usar la base de datos. El cifrado aplicado en Access 2010 usa un algoritmo más seguro que el usado en las versiones anteriores de Access.

Si en Access 2007 usaba una contraseña para cifrar una base de datos, quizás desee cambiar a la nueva tecnología de cifrado que proporciona mayor seguridad.

Para *cambiar una base de datos cifrada de Access 2007 a la nueva tecnología de cifrado*, elimine la contraseña de la base de datos actual y, a continuación, vuelva a agregarla.

Para *cifrar mediante contraseña* se tendrá en cuenta lo siguiente:

- Abra la base de datos que desee cifrar en modo Exclusivo abra la base de datos en modo Exclusivo.
- En la ficha *Archivo*, haga clic en *Abrir*.
- En el cuadro de diálogo Abrir, busque el archivo que desee abrir y, a continuación, seleccione el archivo.
- Haga clic en la flecha situada junto al botón *Abrir* y, a continuación, haga clic en *Abrir en modo exclusivo*.
- En la ficha *Archivo*, haga clic en *Información* y después en *Cifrar con contraseña*.
- Aparece el cuadro de diálogo Establecer contraseña para la base de datos.
- Escriba la contraseña en el cuadro *Contraseña* y escríbala de nuevo en el campo *Confirmar*. Utilice contraseñas seguras que combinen letras en mayúsculas y minúsculas, números y símbolos. Las contraseñas no seguras son aquéllas que no combinan estos elementos. Un ejemplo de contraseña segura sería Y6dh!et5 y de contraseña no segura, Casa27. Las contraseñas deben tener 8 o más caracteres. Una frase con 14 o más caracteres es todavía mejor. Es fundamental que recuerde la contraseña. Si la olvida, Microsoft no puede recuperarla. Guarde las contraseñas que anote en un lugar seguro, lejos de la información que ayudan a proteger.
- Haga clic en *Aceptar*.

Para *descifrar y abrir una base de datos* se tendrá en cuenta lo siguiente:

- Abra la base de datos cifrada como suele abrirse cualquier otra base de datos. Aparece el cuadro de diálogo *Solicitud de contraseña*.
- Escriba la contraseña en el cuadro *Escriba la contraseña de la base de datos* y, a continuación, haga clic en *Aceptar*.

Para *quitar una contraseña* se tendrá en cuenta lo siguiente:

- En la ficha *Archivo*, haga clic en *Información* y después en *Descifrar base de datos*.
- Aparece el cuadro de diálogo Anular la contraseña establecida para la base de datos.

- Escriba la contraseña en el cuadro *Contraseña* y, a continuación, haga clic en *Aceptar*.
- Volver al principio.

6.6.7 Copia de seguridad de una base de datos de Access

Si trabajamos en Access 2010, para hacer una copia de seguridad de una base de datos Access se utiliza la opción *Guardar & Publicar* del botón *Archivo* (Figura 6.63). Posteriormente se elige la opción *Realizar copia de seguridad de la base de datos* en la zona *Guardar base de datos como* de la Figura 6.63. Se obtiene la pantalla *Guardar como* de la Figura 6.64 en cuyo campo *Nombre de archivo* aparece el nombre de la copia de la base de datos. Al hacer clic en *Guardar* se realiza la copia.

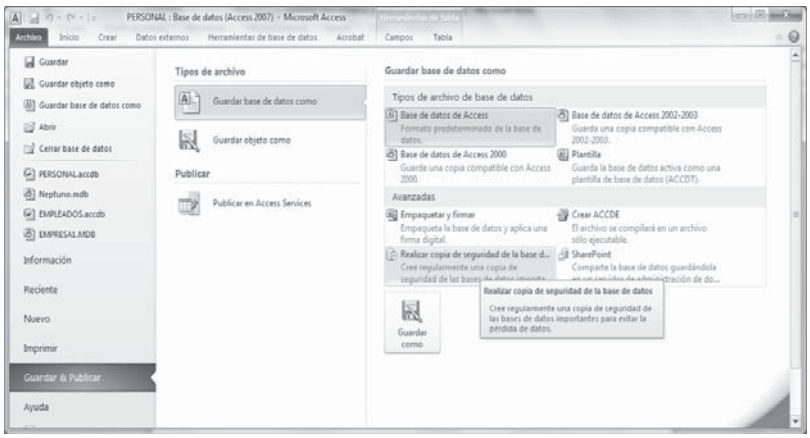


Figura 6.63

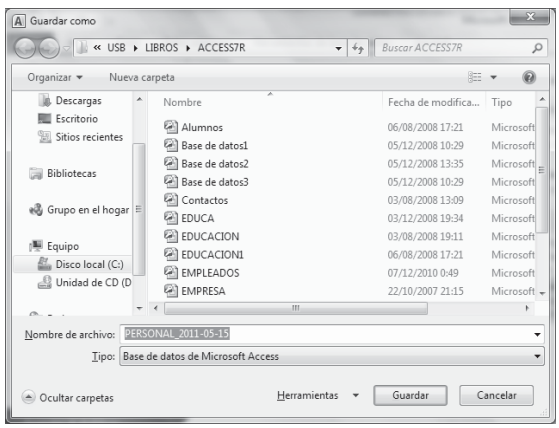


Figura 6.64

Si trabajamos en Access 2007, para hacer una copia de seguridad de una base de datos Access se utiliza la opción *Administrar* del botón *Office* (Figura 6.65). A continuación se hace clic en la opción *Realizar una copia de seguridad de la base de datos*.

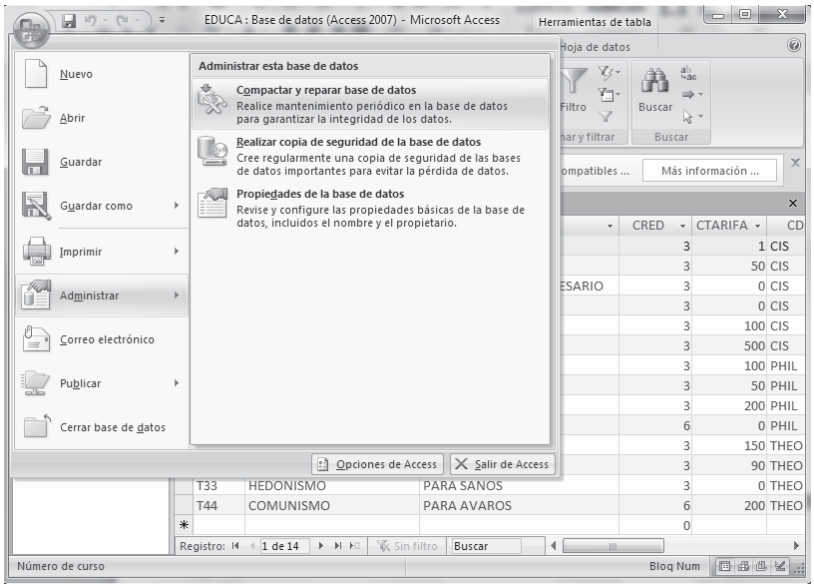


Figura 6.65

6.6.8 Compactar y reparar una base de datos de Access

A medida que agrega y actualiza los datos y cambia su diseño, aumenta el tamaño del archivo de base de datos. Este aumento se produce por la incorporación de nuevos datos, pero también por otros motivos. Access crea objetos temporales ocultos para realizar diversas tareas. Algunas veces, esos objetos temporales permanecen en la base de datos cuando Access ya no los necesita. Cuando se elimina un objeto de base de datos, el espacio en disco que ocupaba el objeto no se recupera automáticamente y el archivo de base de datos sigue utilizando ese espacio en disco aunque se haya eliminado el objeto. Cuando el archivo de base de datos se llena con los restos de los objetos temporales y eliminados, su rendimiento podría verse reducido. Es posible que los objetos se abran más lentamente, que las consultas tarden más de lo normal en ejecutarse y que las operaciones habituales duren más tiempo. Estas razones hacen necesaria la compactación. Cuando se compacta una base de datos, los datos no se comprimen, sino que se reduce el tamaño del archivo de base de datos al eliminar el espacio no utilizado.

Por otra parte, en algunas circunstancias, puede que un archivo de base de datos resulte dañado. Si un archivo de base de datos se comparte a través de una red y varios usuarios trabajan directamente con el archivo al mismo tiempo, es muy poco probable que ese archivo resulte dañado. Pero las posibilidades de que se dañe el archivo aumentan

ligeramente si los usuarios modifican frecuentemente los datos en los campos *Memo*, y el riesgo cada vez es mayor. Puede mitigar ese riesgo con el comando *Compactar y reparar base de datos*.

Si trabajamos en Access 2010, para utilizar el comando *Compactar y reparar* se usa la opción *Información* del menú *Archivo* (Figura 6.66). Posteriormente se hace clic en la opción *Compactar y reparar base de datos*.

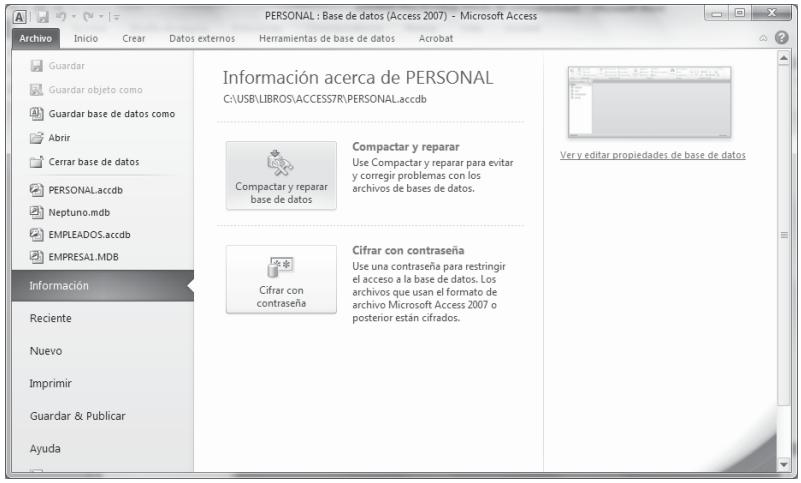


Figura 6.66

Si trabajamos en Access 2007, para utilizar el comando *Compactar y reparar* se usa la opción *Administrar* del botón *Office* (Figura 6.65). Posteriormente se hace clic en la opción Compactar y reparar base de datos. Antes de compactar y reparar una base de datos es conveniente realizar una copia de seguridad de la misma.

Para *compactar y reparar automáticamente una base de datos cuando se cierre* debe seleccionar la opción de base de datos *Compactar al cerrar*. Esta opción solo afecta a la base de datos que está actualmente abierta. Debe definir esta opción para cada base de datos que desee compactar y reparar automáticamente. Para utilizar esta opción se tendrán en cuenta los siguientes pasos:

- Haga clic en el botón de *Office*.
- En el panel izquierdo del espacio de comandos contextuales, haga clic en *Opciones*.
- En el cuadro de diálogo *Opciones de Access*, haga clic en *Base de datos actual*.
- En Opciones de aplicación, active la casilla de verificación Compactar al cerrar.

Capítulo

7

CONSTRUCCIÓN DE GUIONES

7.1 GUIONES O SCRIPTS EN LOS SISTEMAS GESTORES DE BASES DE DATOS

Un script (cuya traducción literal es guión) o archivo de órdenes o archivo de procesamiento por lotes es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Los scripts son casi siempre interpretados, pero no todo programa interpretado es considerado un script. El uso habitual de los scripts es realizar diversas tareas administrativas como combinar componentes, interactuar con el sistema operativo o con el usuario. Los scripts se escriben en lenguajes propios de los sistemas gestores de bases de datos.

Casi todos los grandes sistemas gestores de datos incorporan utilidades que permiten ampliar el lenguaje SQL mediante guiones o scripts para producir pequeñas utilidades que añaden al SQL mejoras de la programación estructurada (bucles, condiciones, funciones,...). La razón es que hay diversas acciones en la base de datos para las que SQL no es suficiente. Por ello todas las bases de datos incorporan algún lenguaje de tipo procedimental (de tercera generación) que permite manipular de forma más avanzada los datos de la base de datos.

En los grandes sistemas gestores de bases de datos existen lenguajes procedimentales formales para la construcción de guiones. Por citar los más importantes, Oracle utiliza el lenguaje procedimental PL/SQL, SQL Server utiliza el lenguaje Transact SQL e Informix usa Informix 4GL.

Estos lenguajes de programación de los distintos SGBD suelen tener elementos comunes entre los que pueden destacarse los siguientes:

- **Programas**. Conjunto de bloques que realizan una determinada labor formados por código con estructuras de control, módulos, procedimientos, funciones y otras estructuras funcionales.

- **Procedimientos**. Programas almacenados en la base de datos y que pueden ser ejecutados si se desea con solo saber su nombre (y teniendo permiso para su acceso).

- **Funciones**. Programas que a partir de unos datos de entrada obtienen un resultado (datos de salida). Una función puede ser utilizada desde cualquier otro programa e incluso desde una instrucción SQL.

- **Desencadenadores** (Triggers). Programas que se ejecutan automáticamente cuando ocurre un determinado suceso a un objeto de la base de datos.

- **Paquetes**. Colección de procedimientos y funciones agrupados dentro de la misma estructura. Similar a las bibliotecas y librerías de los lenguajes convencionales.

7.2 PROCEDIMIENTOS ALMACENADOS EN ACCESS

Un procedimiento SQL consiste en una cláusula PROCEDURE que especifica el nombre del procedimiento, una lista opcional de definiciones de parámetros y una única instrucción SQL.

El nombre de procedimiento no puede ser el mismo que el nombre de la tabla existente.

7.2.1 Sentencia CREATE PROCEDURE

Crea un procedimiento almacenado. Su sintaxis es la siguiente:

```
CREATE PROCEDURE procedimiento
[parámetro1 tipoDatos[, parámetro2 tipoDatos[, ...]] AS instrucciónSql
```

La instrucción CREATE PROCEDURE consta de estos apartados:

Apartado	Descripción
procedimiento	Nombre del procedimiento. Debe seguir las convenciones de nomenclatura estándar.
parámetro1, parámetro2	De uno a 255 nombres de campos o parámetros. Por ejemplo: CREATE PROCEDURE Ventas_por_país [Fecha Inicio] DateTime, [Fecha Fin] DateTime;
tipoDatos	Uno de los tipos de datos principales de Microsoft Jet SQL o sus sinónimos.
instrucciónSql	Una instrucción SQL como SELECT, UPDATE, DELETE, INSERT, CREATE TABLE, DROP TABLE, etc.

EJEMPLO DE INSTRUCCIÓN CREATE PROCEDURE

```
CREATE PROCEDURE Sales_By_CountryRegion ([Beginning Date] DateTime,
[Ending Date] DateTime) AS SELECT Customer, [Ship Address] WHERE
[Shipped Date] Between [Beginning Date] And [Ending Date]
```

7.2.2 Sentencia EXECUTE

Se utiliza para invocar la ejecución de un procedimiento. Su sintaxis es la siguiente:

```
EXECUTE procedimiento [parámetro1[, parámetro2[, ...]]
```

La instrucción EXECUTE consta de estos apartados:

Apartado	Descripción
procedimiento	El nombre del procedimiento que se va a ejecutar.
parámetro1, parámetro2,...	Valores para los parámetros definidos por el procedimiento.

7.2.3 Declaración de PARÁMETROS

Declara el nombre y el tipo de datos de cada parámetro en una consulta de parámetros. Su sintaxis es la siguiente:

```
PARAMETERS nombre tipoDatos [, nombre tipoDatos [, ...]]
```

La declaración PARAMETERS consta de estos apartados:

Apartado	Descripción
nombre	El nombre del parámetro. Se asigna a la propiedad Name del objeto Parameter y se utiliza para identificar este parámetro en la colección Parameters. Puede utilizar un nombre como una cadena que se muestra en un cuadro de diálogo mientras su aplicación ejecuta la consulta. Utilice los corchetes ([ ]) para encerrar texto que contenga espacios en blanco o signos de puntuación. Por ejemplo, [Precio bajo] y [¿En qué mes quiere empezar el informe?]Son argumentos nombre válidos.
tipoDatos	Uno de los tipos de datos principales de Microsoft Jet SQL o sus sinónimos.

Para consultas que ejecute periódicamente, puede utilizar una declaración PARAMETERS para crear una consulta de parámetros. Una consulta de parámetros ayuda a automatizar el proceso de cambio de criterio. En una consulta de parámetros, el código necesitará proporcionar los parámetros cada vez que se ejecute la consulta. La declaración PARAMETERS es opcional, pero, cuando se incluye, precede a cualquier otra instrucción, incluida SELECT. Si la declaración incluye más de un parámetro, sepárelos con comas. El siguiente ejemplo incluye dos parámetros:

EJEMPLO DE CONSULTA DE PARÁMETROS USANDO LA DECLARACIÓN PARAMETERS

```
PARAMETERS [Precio bajo] Currency, [Fecha inicio] DateTime;
```

Puede utilizar *nombre* pero no *tipoDatos* en una cláusula WHERE o HAVING. El siguiente ejemplo espera que se le proporcionen dos parámetros y, después, aplica el criterio a los registros de la tabla PEDIDOS:

EJEMPLO DE CONSULTA CON LA CLÁUSULA WHERE

```
PARAMETERS [Precio bajo] Currency,
[Fecha inicio] DateTime;
SELECT IdPedido, ImportePedido
FROM Pedido
WHERE ImportePedido > [Precio bajo]
AND FechaPedido >= [Fecha inicio];
```

7.2.4 Cláusula PROCEDURE

Define un nombre y parámetros opcionales de una consulta. La cláusula PROCEDURE ha sido reemplazada por la instrucción PROCEDURE. Aunque aún es posible utilizar la cláusula PROCEDURE, la instrucción PROCEDURE proporciona un superconjunto de la capacidad de la cláusula PROCEDURE, por lo que se convierte en la opción de sintaxis recomendada. Su sintaxis es la siguiente:

```
PROCEDURE nombre [parámetro1 tipoDatos[, parámetro2 tipoDatos[, ...]]
```

La cláusula PROCEDURE consta de estos apartados:

Apartado	Descripción
nombre	Nombre del procedimiento. Debe seguir las convenciones de nomenclatura estándar.
parámetro1, parámetro2	Uno o más nombres de campo o parámetros. Por ejemplo: PROCEDURE Ventas_por_país [Fecha Inicio] DateTime, [Fecha Fin] DateTime; Para obtener más información acerca de los parámetros, consulte parámetros.
tipoDatos	Uno de los tipos de datos principales de Microsoft Jet SQL o sus sinónimos.

Un procedimiento SQL consta de una cláusula PROCEDURE (que especifica el nombre del procedimiento), una lista opcional de definiciones de parámetros y una instrucción SQL. Por ejemplo, el procedimiento Obtener_Nombre_Producto podría ejecutar una consulta que recuperará un nombre de producto específico.

- Si la cláusula incluye más de una definición de campo (pares *parámetro-tipoDatos*), sepárelos con comas.
- La cláusula PROCEDURE debe estar seguida de una instrucción SQL (por ejemplo, una instrucción SELECT o UPDATE).

7.3 PROCEDIMIENTOAS ALMACENADOS EN TRANSACT SQL

Un procedimiento almacenado (*stored procedure*) es una colección de sentencias del Transact SQL que organizadas lógicamente resuelven algunas de las operaciones transaccionales que requieren los usuarios. Estos procedimientos se almacenan en la base de datos. Los procedimientos almacenados soportan el empleo de variables declaradas por el usuario, sentencias para toma de decisiones entre otras características.

En el lenguaje Transact SQL de SQL Server existen 5 tipos de procedimientos almacenados:

- *Procedimientos del sistema*: son los que se encuentran almacenados en la base de datos MASTER y algunos en las bases de datos de usuario, estos procedimientos almacenados brindan información acerca de los datos y características del servidor. En el nombre usan como prefijo **sp_**.
- *Procedimientos locales*: son los procedimientos almacenados en una base de datos.
- *Procedimientos temporales*: son procedimientos locales y sus nombres empiezan con los prefijos # o ##, dependiendo de si se desea que sea un procedimiento global a todas las conexiones o local a la conexión que lo define.
- *Procedimientos remotos*: son procedimientos almacenados en servidores distribuidos.
- *Procedimientos extendidos*: son aquéllos que nos permiten aprovechar las funcionalidades de otras librerías externas a SQL Server. Estos procedimientos usan el prefijo **xp_** y se encuentran en la base de datos MASTER.

Entre las principales características de un procedimiento almacenado podemos mencionar:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al lote o al procedimiento que realiza la llamada.

- Contener instrucciones de programación que realicen operaciones en la base de datos, incluidas las llamadas a otros procedimientos.
- Devolver un valor de estado que indica si la operación se ha realizado correctamente o ha habido un error (y el motivo del mismo).
- Permiten una ejecución más rápida, ya que los procedimientos son analizados y optimizados en el momento de su creación, y es posible utilizar una versión del procedimiento que se encuentra en la memoria después de que se ejecute por primera vez.
- Pueden reducir el tráfico de red.
- Pueden utilizarse como mecanismo de seguridad, ya que se puede conceder permisos a los usuarios para ejecutar un procedimiento almacenado, incluso si no cuentan con permiso para ejecutar directamente las instrucciones del procedimiento.

7.3.1 Crear un procedimiento almacenado

La sentencia CREATE PROCEDURE de Transact SQL permite crear procedimientos almacenados mediante la siguiente sintaxis:

```
CREATE PROC[EDURE] <Nombre Procedimiento>
[
{@parámetro tipoDatos} [= predeterminado] [OUTPUT]
]
[,...n]
[WITH
{
RECOMPILE
| ENCRYPTION
}
]
AS
Sentencias SQL [...n]
Argumentos
@parámetro
```

El usuario puede tener hasta un máximo de 1.024 parámetros. El nombre del parámetro debe comenzar con un signo (@). Los parámetros son locales al procedimiento.

- **Predeterminado**. Es un valor predeterminado para el parámetro.
- **OUTPUT**. Indica que se trata de un parámetro de salida. El valor de esta opción puede devolverse a EXEC[UTE]. Utilice los parámetros OUTPUT para devolver información al procedimiento que llama. Los parámetros de texto no se pueden utilizar como parámetros OUTPUT.

- **{RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}**. RECOMPILE indica que SQL Server no almacena en la caché un plan para este procedimiento, con lo que el procedimiento se vuelve a compilar cada vez que se ejecuta. Utilice la opción RECOMPILE cuando emplee valores atípicos o temporales para no anular el plan de ejecución que está almacenado en la memoria caché. ENCRYPTION indica que SQL Server codifica la entrada de la tabla SYSCOMMENTS que contiene el texto de la instrucción CREATE PROCEDURE. Entre otras observaciones podemos mencionar:
  - El tamaño máximo de un procedimiento es de 128 MB.
  - Un procedimiento solo puede crearse en la base de datos actual. Se puede crear otros objetos de base de datos dentro de un procedimiento almacenado.
  - Puede hacer referencia a un objeto creado en el mismo procedimiento almacenado, siempre que se cree antes de que se haga referencia al objeto.
  - Puede hacer referencia a tablas temporales dentro de un procedimiento almacenado.

Si crea una tabla temporal privada dentro de un procedimiento almacenado, la tabla temporal existirá únicamente para los fines del procedimiento; desaparecerá cuando éste finalice. Si ejecuta un procedimiento almacenado que llama a otro procedimiento almacenado, el procedimiento al que se llama puede tener acceso a todos los objetos creados por el primer procedimiento, incluidas las tablas temporales. Si se ejecuta un procedimiento almacenado remoto que realiza cambios en un servidor SQL Server remoto, los cambios no se podrán deshacer. Los procedimientos almacenados remotos no intervienen en las transacciones.

A continuación se presenta un procedimiento que calcula la nota media de los alumnos matriculados en una determinada clase:

EJEMPLO DE PROCEDIMIENTO PARA EL CÁLCULO DE LA NOTA MEDIA

```
Use Matriculas
GO
CREATE PROCEDURE ListaPromedios
as
SELECT NomApe = (Nom + ' ' + Pat + ' ' + Mat), Inscritos.Sec,
Curso = Case
When SubString(Inscritos.Sec, 2, 1)='1'
Then 'V.Basic'
When SubString(Inscritos.Sec, 2, 1)='2'
Then 'V.Fox'
When SubString(Inscritos.Sec, 2, 1)='3'
```

```
    Then 'SQL Server'
    End,
       Promedio = (N1+N2)/2
    From Alumnos INNER JOIN Inscritos
    On Alumnos.codalu = Inscritos.codalu
    INNER JOIN Calificaciones
    On Inscritos.codalu = Calificaciones.codalu
    AND Inscritos.sec = Calificaciones.sec
    GO
```

Para poder ejecutar el procedimiento emplearemos la siguiente sintaxis:

```
EXEC ListaPromedios
GO
```

Para ver la información de la implementación del procedimiento almacenado:

```
Sp_HelpText ListaPromedios
GO
```

Para ver cuáles son las columnas que producen la información presentada por el procedimiento:

```
Sp_Depends ListaPromedios
GO
```

7.3.2 Modificar procedimientos almacenados

Si se desea modificar el procedimiento almacenado utilice la siguiente sintaxis:

```
ALTER PROC[EDURE] <Nombre Procedimiento>
[
{@parámetro tipoDatos} [= predeterminado] [OUTPUT]
]
[,...n]
[WITH
{
RECOMPILE
| ENCRYPTION
}
]
AS
Sentencias SQL [...n]
```

Como ejemplo modificamos el procedimiento anterior mediante la sintaxis siguiente:

EJEMPLO DE PROCEDIMIENTO ALMACENADO MODIFICADO

```
ALTER PROCEDURE ListaPromedios
WITH ENCRYPTION
as
SELECT NomApe = (Nom + ' ' + Pat + ' ' + Mat), Inscritos.Sec,
Curso = Case
When SubString(Inscritos.Sec, 2, 1)='1'
Then 'V.Basic'
When SubString(Inscritos.Sec, 2, 1)='2'
Then 'V.Fox'
When SubString(Inscritos.Sec, 2, 1)='3'
Then 'SQL Server'
End,
   Promedio = (N1+N2)/2
From Alumnos INNER JOIN Inscritos
On Alumnos.codalu = Inscritos.codalu
INNER JOIN Calificaciones
On Inscritos.codalu = Calificaciones.codalu
AND Inscritos.sec = Calificaciones.sec
GO
```

7.3.3 Eliminar procedimientos almacenados

Para eliminar un procedimiento se utiliza el siguiente formato de sintaxis:

```
DROP PROCEDURE <Nombre del procedimiento>
```

7.4 DESENCADENADORES EN TRANSACT SQL

Un desencadenador (*trigger*) es un tipo especial de procedimiento almacenado que se activa de forma controlada por sucesos antes que por llamadas directas. Los desencadenadores (*triggers*) están asociados a tablas. Son una gran herramienta para controlar las reglas de negocio más complejas que una simple integridad referencial, los desencadenadores (*triggers*) y las sentencias que desencadenan su ejecución trabajan unidos como una transacción.

El grueso de instrucciones de la definición del desencadenador deben ser INSERT, UPDATE o DELETE. Aunque se puede utilizar SELECT, no es recomendable ya que el usuario no espera que se le devuelva registros luego de agregar o modificar información. Los desencadenadores (*triggers*) siempre toman acción después de que la operación fue registrada en el log.

Para crear un desencadenador puede utilizar el siguiente formato de sentencia Transact SQL:

```
CREATE DESENCADENADOR <Nombre del Desencadenador>
ON <Nombre de la Tabla>
FOR <INSERT l UPDATE l DELETE>
AS
Sentencias….
GO
```

Consideraremos a continuación un desencadenador tal que al agregar un nuevo pedido a la tabla de PEDIDOS se debe incrementar las ventas del representante que concretó el pedido, así como también debe reducirse el número de existencias.

Para ello podemos crear el siguiente desencadenador:

```
Use Ejemplo
GO
CREATE DESENCADENADOR NuevoPedido
ON Pedidos
FOR INSERT
AS
UPDATE RepVentas
SET VENTAS =VENTAS + INSERTED.IMPORTE
FROM REPVENTAS INNER JOIN INSERTED
ON REPVENTAS.NUM_EMPL = INSERTED.REP
UPDATE PRODUCTOS
SET EXISTENCIAS = EXISTENCIAS - INSERTED.CANT
FROM PRODUCTOS INNER JOIN INSERTED
ON PRODUCTOS.ID_FAB = INSERTED.FAB
AND PRODUCTOS.ID_PRODUCTO = INSERTED.PRODUCTO
GO
```

7.5 FUNCIONES EN TRANSACT SQL

Transact SQL tiene la capacidad de implementar funciones definidas por el usuario. Esta característica ayudará a solucionar los problemas de reutilización del código y dará mayor flexibilidad al programar las consultas de SQL.

Transact SQL utiliza tres tipos de funciones: funciones escalares, funciones de tabla en línea y funciones de tabla de multisentencias. Los tres tipos de funciones aceptan parámetros de cualquier tipo excepto el *rowversion*. Las funciones escalares devuelven un solo valor y las funciones de tabla en línea y multisentencias devuelven un tipo de dato tabla.

7.5.1 Funciones escalares

Las funciones escalares devuelven un tipo de los datos tal como int, money, varchar, real, etc. Pueden emplearse en cualquier lugar incluso incorporadas dentro de sentencias SQL. La sintaxis para crear una función escalar es la siguiente:

```
CREATE FUNCTION [propietario] Nombre_de_Función
( [{ @parametro parametro_scalar [ = default]} [,..n]])
RETURNS tipo_de_dato_scalar_de_retorno
[WITH <opción > >::={SCHEMABINDING | ENCRYPTION]
[AS]
BEGIN
Código de la función
RETURN expresión_scalar
END
```

Como ejemplo se presenta una función sencilla para obtener un número elevado al cubo:

```
CREATE FUNCTION dbo.Cubo( @Numero float)
RETURNS float
AS
BEGIN
RETURN(@fNumero * @fNumero * @fNumero)
END
```

Otra característica interesante es que las funciones de usuario soportan llamadas recursivas, como se muestra en el siguiente ejemplo, que calcula el factorial de un número:

```
CREATE FUNCTION dbo.Factorial ( @Numero int )
RETURNS INT
AS
BEGIN
DECLARE @i int
```

```
IF @Numero <= 1
SET @i = 1
ELSE
SET @i = @Numero * dbo.Factorial( @Numero - 1 )
RETURN (@i)
END
```

7.5.2 Funciones de tabla en línea

Las funciones de tabla en línea son las funciones que devuelven la salida de una simple declaración SELECT. La salida se puede utilizar dentro de joins o querys como si fuera una tabla estándar. La sintaxis para una función de tabla en línea es como sigue:

```
CREATE FUNCTION [propietario] Nombre_de_Función
( [{ @parametro tipo_parametro_scalar [ = default]} [,..n]])
RETURNS TABLE
[WITH <opcion >::={SCHEMABINDING | ENCRYPTION}]
RETURN [(] sentencia_select [)]
```

Una función InLine podría retornar los autores de un estado en particular:

```
CREATE FUNCTION dbo.AutoresPorEstado (@State char(2) )
RETURNS TABLE
AS
RETURN (SELECT * FROM Authors WHERE state = @cState)
```

7.5.3 Funciones de tabla de multisentencias

Son similares a los procedimientos almacenados excepto que devuelven una tabla. Este tipo de función se usa en situaciones donde se requiere más lógica y proceso. Lo que sigue es la sintaxis para una función de tabla de multisentencias:

```
CREATE FUNCTION [propietario] Nombre_de_Función
( [{ @parametro tipo_parametro_scalar [ = default]} [,..n]])
RETURNS TABLE
[WITH <opción> >::={SCHEMABINDING | ENCRYPTION]
[AS]
BEGIN
Código de la función
RETURN
END
```

7.5.4 Llamando funciones y columnas computadas

Transact SQL proporciona algunas funciones definidas por el usuario a nivel sistema en la base de datos MASTER. Estas funciones del sistema se invocan con una sintaxis levemente distinta a las que usted puede crear. Las funciones del sistema que devuelven un tabla tienen la sintaxis siguiente:

```
Nombre de Funcion( [argumento_expr], [,...])
```

Las funciones escalares y las de conjunto de filas son invocadas con el siguiente formato:

```
[BD] propietario. Funcion ([argumento_expr], [,...])
```

Las funciones escalares se pueden utilizar para crear columnas calculadas en una definición de tabla, los argumentos de las funciones calculadas, columnas dla fichala, constantes, o funciones incorporadas.

Este ejemplo muestra una tabla que utiliza una función del volumen para calcular el volumen de un envase:

```
CREATE FUNCTION dbo.Volume ( @Height decimal(5,2),
@Length decimal(5,2),
@Width decimal(5,2) )
RETURNS decimal (15,4)
AS
BEGIN
RETURN (@dHeight * @dLength * @dWidth )
END
CREATE TABLE dbo.Container
(
ContainerID int NOT NULL PRIMARY KEY,
MaterialID int NOT NULL REFERENCES Material(MaterialID),
ManufacturerID int NOT NULL REFERENCES Manufacturer(ManufacturerID)
Height decimal(5,2) NOT NULL,
Length decimal(5,2) NOT NULL,
Width decimal(5,2) NOT NULL,
Volume AS (dbo.Volume( Height, Length, Width)
)
```

7.6 EL LENGUAJE PROCEDIMENTAL PL/SQL DE ORACLE

PL/SQL (*Procedural Language/SQL*) es un lenguaje de programación que se utiliza para acceder y trabajar con bases de datos Oracle desde distintos entornos. PL/SQL amplía la funcionalidad del lenguaje SQL añadiendo estructuras típicas de los lenguajes

procedimentales, como las variables y los tipos, las estructuras de control (bucles, órdenes IF-THEN-ELSE, etc.), los procedimientos y las funciones, los tipos de objetos y los métodos, y otras estructuras clásicas en los lenguajes de programación procedimentales. Resulta así un lenguaje robusto y potente que combina la flexibilidad de SQL con la potencia y configurabilidad de las construcciones procedimentales.

En el trabajo con aplicaciones de bases de datos es típica la estructura cliente-servidor formada por un servidor de bases de datos que recibe peticiones de información de programas residentes en máquinas clientes conectadas por la red con el servidor. Las solicitudes de información suelen llevarse a cabo en lenguaje SQL, lo que da lugar a múltiples comunicaciones por la red (una por cada petición SQL).

Para racionalizar este tráfico de información se utiliza PL/SQL, que empaqueta varias órdenes SQL en un único *bloque PL/SQL*, que se envía al servidor como una unidad. De esta forma se disminuye el tráfico en la red y aumenta la velocidad de la aplicación. Si el cliente y el servidor están en la misma máquina, el rendimiento crece y el empaquetar las órdenes SQL en bloques PL/SQL sigue produciendo un programa más simple que realiza menos llamadas a la base de datos y que es rentable aunque no exista red. Por lo tanto la unidad básica en PL/SQL es el bloque, estando formados los programas por combinaciones de bloques que incluso pueden estar anidados. De esta forma se separan las tareas unas de otras porque cada bloque realiza una unidad lógica de trabajo en el programa.

Entre PL/SQL y la base de datos se transmite la información mediante *variables*, que no son más que zonas de almacenamiento que pueden ser leídas y escritas por el programa. Los bloques suelen tener una sección llamada *sección declarativa* en la que por regla general se declaran las variables. Cada variable tiene asociado un *tipo* que define la clase de información que se puede almacenar en ella. Los tipos de las variables pueden ser similares a los de las columnas de una base de datos, aunque PL/SQL también admite como tipos definidos por el usuario a las tablas y a las columnas. También es posible utilizar *tipos de objetos* que tienen métodos y atributos, que pueden ser almacenados en una base de datos, y que elevan al lenguaje PL/SQL a la categoría de *lenguaje de programación orientado al objeto*.

Otra característica clásica de PL/SQL son las estructuras de *bucles*, que permiten ejecutar repetidamente una cierta secuencia de órdenes (bucle FOR, bucle WHILE, etc.). Estos bucles pueden anidarse al igual que en otros lenguajes de programación procedimentales. Otro elemento esencial del lenguaje PL/SQL son los *cursores*, que se emplean para procesar múltiples filas seleccionadas de la base de datos, y que permiten al programa recorrer una por una las filas obtenidas y procesarlas.

Los *subprogramas, procedimientos* y *funciones* son también elementos típicos de PL/SQL que suelen constituir bloques en este lenguaje. Los *paquetes* permiten agrupar como una unidad subprogramas relacionados de modo que muchas de las características avanzadas de PL/SQL se implementan como paquetes que ayudan a realizar buenos diseños de aplicaciones a través de la abstracción de datos. Uno de estos paquetes implementa el *PL/SQL dinámico*, que puede utilizarse para obviar la restricción de que solo están permitidas en PL/SQL las órdenes DML. Otros paquetes permiten manipular *objetos de gran tamaño*,

programar trabajos para ejecución automática, leer y escribir en los archivos del sistema operativo, crear *páginas web dinámicas* y otras tareas especiales.

Otro tipo esencial de bloque en PL/SQL lo constituyen los *disparadores*, que se activan automáticamente cuando se modifican los datos de Oracle, lo que permite forzar reglas de negocio complicadas que no podrían hacerse con restricciones de integridad referencial. Oracle PL/SQL también se ocupa del *tratamiento de errores*, tarea crucial en cualquier aplicación bien diseñada. Pueden utilizarse las *excepciones* de PL/SQL para asegurar que los programas sean robustos y puedan manejar las condiciones de excepción en el momento de la ejecución. En Oracle PL/SQL también se utilizan *tablas anidadas* y *varrays*, tipos de datos que se engloban bajo el concepto de *colección* y que amplían la funcionalidad de las *tablas de PL/SQL*. Así mismo existen técnicas de ajuste y mejora del rendimiento de los programas y se implementa un robusto sistema de *gestión de colas* (*queuing*).

7.6.1 La estructura de PL/SQL

PL/SQL amplía SQL con los elementos característicos de los lenguajes de programación, como variables, sentencias de control de flujo, bucles, etc. Cuando se desea realizar una aplicación completa para el manejo de una base de datos relacional, resulta necesario utilizar alguna herramienta que soporte la capacidad de consulta del SQL y la versatilidad de los lenguajes de programación tradicionales. PL/SQL es el lenguaje de programación que proporciona *Oracle* para extender el SQL estándar con otro tipo de instrucciones. Inicialmente, y de un modo muy resumido, podemos decir que PL/SQL se compone de los siguientes elementos:

- *Bloques*: unidades básicas en la programación de PL/SQL.
- *Unidades léxicas*: secuencias de caracteres permitidos en PL/SQL que componen los programas.
- *Variables*: espacios de memoria que pueden contener valores de datos.
- *Tipos*: elementos que pueden usarse en las columnas de la base de datos y que definen la naturaleza de los datos permitidos en la base de datos y en el lenguaje PL/SQL. Los tipos de PL/SQL se definen en un paquete denominado STANDAR cuyos contenidos son accesibles desde cualquier bloque PL/SQL.
- *Expresiones y operadores*: elementos que permiten unir las variables PL/SQL. Los operadores definen cómo se asignan los valores a las variables y cómo se manipulan dichos valores. Una expresión es una secuencia de variables y literales separados por operadores. El valor de una expresión se determina a partir de los valores de las variables y literales que la componen, y de la definición de los operadores.

- *Funciones*: además de los tipos, el paquete STANDARD define las funciones predefinidas SQL y de conversión disponibles en PL/SQL.

- *Estructuras de control PL/SQL*: permiten controlar el comportamiento del bloque a medida que éste se ejecuta e incluyen las órdenes condicionales y los bucles. Las estructuras de control combinadas con las variables dotan a PL/SQL de poder y flexibilidad.

- *Registros*: los registros de PL/SQL son similares a las estructuras del lenguaje C. Un registro proporciona un mecanismo para tratar con variables diferentes, pero relacionadas, como si fueran una unidad.

- *Tablas y matrices*: las tablas PL/SQL se asemejan a las matrices del lenguaje C. Sintácticamente se las trata de la misma forma que a las matrices, aunque su implementación es distinta. Para poder declarar una tabla en PL/SQL es necesario primero definir su tipo y luego una variable de dicho tipo.

- *Cursores*: dentro de PL/SQL la orden SELECT no debe devolver más de una fila, pero si es necesario que SELECT devuelva más de una fila hay que emplear un cursor para extraer individualmente cada fila.

- *Procedimientos*: estructuras de bloques que pueden ser almacenados en la base de datos para ser ejecutados cuando sea necesario.

- *Paquetes*: estructuras de bloques PL/SQL que proporcionan un mecanismo para extender en todo momento el propio lenguaje PL/SQL.

- *Disparadores*: estructuras de bloques que se ejecutan de forma implícita cada vez que tiene lugar el suceso de disparo. Los disparadores no admiten argumentos.

7.6.2 Bloques en PL/SQL

Los programas de PL/SQL están compuestos por bloques, que pueden estar situados uno detrás de otro (*estructura secuencial*) o pueden estar uno dentro de otro (*estructura anidada*). Por tanto, el bloque es la unidad básica en PL/SQL.

Los bloques tienen tres secciones diferenciadas:

- *Sección declarativa*: se sitúan en ella las variables, cursores y tipos usados por el bloque.

- *Sección ejecutable*: se sitúan en ella las órdenes SQL y las órdenes procedimentales que llevan a cabo el trabajo del bloque.

- *Sección de excepciones*: se sitúa en ella código que no se ejecuta a menos que ocurra un error. Por esta razón, la sección de excepciones suele llamarse también sección de tratamiento de errores.

La sintaxis de un programa PL/SQL será entonces la siguiente:

```
DECLARE
/* Sección declarativa */
BEGIN
/* Sección ejecutable */
EXCEPTION
/* Sección de excepciones
END;
/
```

Hay que observar que cuando se ejecuta un programa PL/SQL en el entorno de SQL*Plus, es necesario finalizar el programa con la barra inclinada / antes de submitirlo. Por otra parte, las palabras DECLARE, BEGIN, EXCEPTION y END delimitan cada una de las secciones.

La única sección indispensable en un bloque es la sección ejecutable, siendo las otras dos opcionales. Si falta la sección declarativa, el bloque empieza con la palabra clave BEGIN. Si falta la sección de excepciones, no aparecerá la palabra clave EXCEPTION, y el bloque terminará con la palabra clave END, seguida de un punto y coma. Por tanto, un programa mínimo, es decir, un programa PL/SQL con solo la sección ejecutable, tiene la siguiente estructura:

```
BEGIN
/* Sección ejecutable */
END;
/
```

7.7 TIPOS DE ESTRUCTURAS DE CONTROL EN PL/SQL

Las estructuras de control permiten controlar el comportamiento del bloque a medida que éste se ejecuta. Las más importantes son las estructuras condicionales y los bucles.

7.7.1 Estructuras condicionales

Las estructuras condicionales se utilizan para la realización de acciones dependiendo del cumplimiento o no de determinadas condiciones. Las estructuras condicionales más comunes son: IF-THEN, IF-THEN-ELSE, IF-THEN-ELSIF y CASE.

IF-THEN

Se trata de una estructura condicional en la que se ejecuta una secuencia de instrucciones si la condición es cierta. Su sintaxis es la siguiente:

```
IF condición THEN
   Secuencia_de_instrucciones
END IF;
```

Como ejemplo tenemos:

EJEMPLO DE ESTRUCTURA IF-THEN

```
IF ventas > cota THEN
   compute_bonus(empid);
   UPDATE payroll SET pay = pay + bonus WHERE empno = emp_id;
END IF;
```

También puede escribirse toda la estructura condicional sobre la misma línea tal y como se observa en el ejemplo siguiente:

EJEMPLO DE ESTRUCTURA IF-THEN SOBRE LA MISMA LÍNEA

```
IF x > y THEN high := x; END IF;
```

IF-THEN-ELSE

Se trata de una estructura condicional en la que se ejecuta una primera secuencia de instrucciones si la condición es cierta, y se ejecuta una segunda secuencia de instrucciones si la condición es falsa. Su sintaxis es la siguiente:

```
IF condición THEN
   Secuencia_de_instrucciones1
ELSE
   Secuencia_de_instrucciones2
END IF;
```

EJEMPLO DE ESTRUCTURA IF-THEN-ELSE

```
IF trans_type = 'CR' THEN
   UPDATE accounts SET balance = balance + credit WHERE ...
ELSE
   UPDATE accounts SET balance = balance - debit WHERE ...
END IF;
```

Estas estructuras condicionales pueden anidarse tal y como se indica en el ejemplo siguiente:

EJEMPLO DE ESTRUCTURA IF-THEN-ELSE ANIDADA

```
IF trans_type = 'CR' THEN
   UPDATE accounts SET balance = balance + credit WHERE ...
ELSE
   IF new_balance >= minimum_balance THEN
      UPDATE accounts SET balance = balance - debit WHERE ...
   ELSE
      RAISE insufficient_funds;
   END IF;
END IF;
```

IF-THEN-ELSIF

Estructura condicional que se utiliza para seleccionar una acción entre varias alternativas mutuamente excluyentes. Si la primera condición es falsa o nula, se pasa a testear la condición de la cláusula ELSIF comenzando una nueva estructura condicional IF-THEN-ELSE. Su sintaxis es la siguiente:

```
IF condición THEN
   Secuencia_de_instrucciones1
ELSIF condición2 THEN
   Secuencia de instrucciones2
ELSE
   Secuencia de instrucciones3
END IF;
```

EJEMPLO DE ESTRUCTURA IF-THEN-ELSIF

```
BEGIN
   ...
   IF ventas > 50000 THEN
      bonus := 1500;
   ELSIF ventas > 35000 THEN
      bonus := 500;
   ELSE
      bonus := 100;
   END IF;
   INSERT INTO payroll VALUES (emp_id, bonus, ...);
END;
CASE
```

Al igual que IF, la sentencia CASE selecciona una secuencia de sentencias a ejecutar, sin embargo, para realizar la selección utiliza un selector y no una expresión booleana. Para ver la diferencia entre IF y CASE presentamos el mismo ejemplo con las dos sintaxis.

Con IF tenemos el siguiente código:

EJEMPLO DE CÓDIGO CON LA SENTENCIA IF

```
IF grade = 'A' THEN
   dbms_output.put_line('Excellent');
ELSIF grade = 'B' THEN
   dbms_output.put_line('Very Good');
ELSIF grade = 'C' THEN
   dbms_output.put_line('Good');
```

```
ELSIF grade = 'D' THEN
   dbms_output. put_line('Fair');
ELSIF grade = 'F' THEN
   dbms_output.put_line('Poor');
ELSE
   dbms_output.put_line('No such grade');
END IF;
```

Con CASE tenemos el siguiente código:

EJEMPLO DE CÓDIGO CON LA SENTENCIA CASE

```
CASE grade
   WHEN 'A' THEN dbms_output.put_line('Excellent');
   WHEN 'B' THEN dbms_output.put_line('Very Good');
   WHEN 'C' THEN dbms_output.put_line('Good');
   WHEN 'D' THEN dbms_output.put_line('Fair');
   WHEN 'F' THEN dbms_output.put_line('Poor');
   ELSE dbms_output.put_line('No such grade');
END CASE;
```

Observamos que la sentencia CASE es más legible y más eficiente que la sentencia IF. La sintaxis de CASE podría expresarse como sigue:

```
[<<nombre etiqueta>>]
CASE selector
   WHEN expresión1 THEN secuencia_de_instrucciones1;
   WHEN expresión1 THEN secuencia_de_instrucciones2;
   ...
   WHEN expresiónN THEN secuencia_de_instruccionesN;
  [ELSE secuencia de instruccionesN+1;]
END CASE [nombre_etiqueta];
```

Existe relación entre las sentencias vistas anteriormente. Por ejemplo, las estructuras condicionales siguientes son equivalentes:

```
IF condition1 THEN           |     IF condition1 THEN
   statement1;               |        statement1;
ELSE                         |     ELSIF condition2 THEN
   IF condition2 THEN        |        statement2;
      statement2;            |     ELSIF condition3 THEN
   ELSE                      |        statement3;
      IF condition3 THEN     |     END IF;
         statement3;         |
      END IF;                |
   END IF;                   |
END IF;                      |
```

7.7.2 Bucles

PL/SQL utiliza los bucles para ejecutar órdenes de forma repetida. Existen distintos tipos de bucles que se analizarán en párrafos posteriores.

Bucles simples

Los bucles simples tienen la siguiente sintaxis:

```
LOOP
   secuencia_de_instrucciones;
   EXIT [WHEN condición];
END LOOP;
```

La sintaxis EXIT [WHEN condición] es equivalente a:

```
IF condición THEN
   EXIT;
END IF;
```

Por lo tanto, la sintaxis completa del bucle simple es la siguiente:

```
LOOP
   secuencia_de_instrucciones;
      IF condición THEN
      EXIT;
      END IF;
END LOOP;
```

Bucles WHILE

La sintaxis de un bucle WHILE es la siguiente:

```
WHILE condición LOOP
   secuencia_de_instrucciones
END LOOP;
```

Se produce una evaluación de la condición previa a cada iteración del bucle. Si la condición es verdadera, se ejecuta la secuencia de órdenes, pero si es falsa, el bucle termina y se transfiere el control a las instrucciones posteriores a END LOOP.

También se pueden utilizar las órdenes EXIT o EXIT WHEN dentro de un bucle WHILE para salir del bucle de forma prematura. La sintaxis será:

```
LOOP
   secuencia_de_instrucciones
   EXIT WHEN expresión booleana;
END LOOP;
```

Bucles FOR numéricos

Los bucles FOR numéricos disponen de un número de iteraciones definido. Ya hemos visto que en los bucles simples y WHILE el número de iteraciones depende de la condición del bucle y no se conoce de antemano. La sintaxis es la siguiente:

```
FOR contador IN [REVERSE] límite_inferior..límite_superior LOOP
   secuencia_de_instrucciones
END LOOP;
```

Órdenes GOTO y etiquetas

PL/SQL dispone de la orden de salto GOTO. Para identificar el lugar al que se tiene que realizar el salto, se utiliza una etiqueta que suele encerrarse entre corchetes angulares dobles. La sintaxis es la siguiente:

```
GOTO etiqueta;
```

En el ejemplo siguiente la orden GOTO pasa el control a una inserción de fila en la tabla TEMP.

EJEMPLO DE ORDEN GOTO

```
BEGIN
   ...
   GOTO insertar_fila;
   ...
   <<insertar_fila>>
   INSERT INTO emp VALUES ...
END;
Orden NULL
```

La orden NULL se utiliza dentro de los programas PL/SQL para indicar que no se realice ninguna acción. A continuación, se presenta un ejemplo en el que cuando no se cumple la condición de IF no se realiza ninguna acción.

EJEMPLO DE ORDEN NULL

```
IF rating > 90 THEN
   compute_bonus(emp_id);
ELSE
   NULL;
END IF;
```

Los bucles y las estructuras de control suelen utilizarse simultáneamente en los programas PL/SQL.

7.8 SUBPROGRAMAS ALMACENADOS: PROCEDIMIENTOS Y FUNCIONES

7.8.1 Creación de procedimientos almacenados

Para *crear un procedimiento almacenado* se usa la sintaxis general siguiente:

```
CREATE [OR REPLACE] PROCEDURE nombre_procedimiento
[(argumento [{IN|OUT|IN OUT}] tipo,
……
(argumento [{IN|OUT|IN OUT}] tipo)] {IS|AS}
cuerpo_procedimiento
```

El nombre del procedimiento es *nombre_procedimiento*, *argumento* es el nombre de un parámetro del procedimiento, *tipo* es el tipo de parámetro asociado y *cuerpo_procedimiento* es el bloque PL/SQL que contiene el código del procedimiento. La cláusula OR REPLACE da un mensaje de aviso si se va a crear un procedimiento que ya existe. Para cambiar el código de un procedimiento es necesario eliminarlo y volver a crearlo. Los procedimientos pueden tener *parámetros reales* (contienen los valores que se pasan al procedimiento cuando éste es invocado y reciben los resultados del procedimiento cuando éste termina) y *parámetros formales* (meros contenedores para los valores de los parámetros reales).

Cuando se llama al procedimiento (con CALL) se asigna el valor de los parámetros reales a los parámetros formales, dentro del procedimiento se hace referencia a dichos valores mediante los parámetros formales y cuando el procedimiento termina se asigna el valor de los parámetros formales a los parámetros reales.

Los parámetros formales pueden tener los modos IN (por defecto), OUT o IN OUT. El modo IN indica que el parámetro real se pasa al procedimiento cuando éste es invocado, dentro del procedimiento el parámetro formal se considera como de solo lectura (no puede ser cambiado) y cuando termina el procedimiento el parámetro real no sufre cambios. El modo OUT indica que se ignora cualquier valor que tenga el parámetro real cuando se llama al procedimiento, dentro del procedimiento el parámetro real se considera como de solo escritura (no puede ser leído pero sí es posible asignarle valores) y cuando finaliza el procedimiento los contenidos del parámetro formal se pasan al parámetro real. El modo IN OUT indica que el parámetro real se pasa al procedimiento cuando éste es invocado, dentro del procedimiento el parámetro formal puede ser de lectura y escritura, y cuando termina el procedimiento los contenidos del parámetro formal se asignan al parámetro real.

El cuerpo de un procedimiento es un bloque PL/SQL con sus secciones declarativa, ejecutable y de manejo de excepciones. La sección declarativa se sitúa entre las palabras

clave IS o AS y la palabra clave BEGIN, la ejecutable entre BEGIN y EXCEPTION y la de excepciones entre EXCEPTION y END (no existe la palabra clave DECLARE en un procedimiento y su lugar lo ocupan IS o AS). A veces, se suele incluir el nombre del propio procedimiento después de la orden END que cierra la declaración.

Los parámetros formales de un procedimiento pueden tener valores predeterminados que se declaran mediante la siguiente sintaxis:

```
Nombre_parámetro [modo] tipo_parámetro {:=|DEFAULT} valor_inicial
```

7.8.2 Creación de funciones

El concepto de función es muy similar al concepto de procedimiento. Ambos aceptan argumentos, tanto en notación posicional como en notación nominal. Ambos son bloques PL/SQL con secciones declarativa, ejecutable y de excepciones. La diferencia estriba en que una llamada a un procedimiento es una orden PL/SQL en sí misma, mientras que una llamada a una función se realiza como parte de una expresión.

Para crear una función se utiliza la sintaxis general siguiente:

```
CREATE [OR REPLACE] FUNCTION nombre_función
[(argumento [{IN|OUT|IN OUT}] tipo,
……
(argumento [{IN|OUT|IN OUT}] tipo)]
RETURN tipo_retorno {IS|AS}
cuerpo_función
```

El nombre de la función es *nombre_función*, *argumento* y *tipo* juegan el mismo papel que en un procedimiento, *tipo_retorno* es el tipo de valor que devuelve la función y *cuerpo_función* es un bloque PL/SQL que contiene el código de la función. La orden RETURN se emplea para devolver el control y un valor al entorno que realizó la llamada.

7.8.3 Eliminación de procedimientos y funciones

La sintaxis para borrar un procedimiento es la siguiente:

```
DROP PROCEDURE nombre_procedimiento
```

La sintaxis para borrar una función es la siguiente:

```
DROP FUNCTION nombre_función
```

7.9 DISPARADORES

Un disparador es un bloque nominado que se asemeja a los procedimientos y a las funciones y tiene sección declarativa, ejecutable y de manejo de excepciones. Pero los disparadores no admiten argumentos y se ejecutan de forma implícita cada vez que ocurre el suceso de disparo, que suele ser una operación DML (INSERT, UPDATE o DELETE) sobre una tabla de la base de datos. Los disparadores suelen utilizarse para restricciones de integridad complejas, auditoría de la información de una tabla o aviso a otros programas para ejecutar una acción.

La sintaxis general para crear un disparador es la siguiente:

```
CREATE [OR REPLACE] TRIGGER nombre_disparador
{BEFORE|AFTER} suceso_disparo ON referencia_tabla
[FOR EACH ROW [WHEN condición_disparo]]
cuerpo_disparador
```

El nombre del disparador es *nombre_disparador*, *suceso_disparo* especifica cuándo se activa el disparador, *referencia_tabla* es la tabla para la que se define el disparador y *cuerpo_disparador* es el código principal del disparador. El cuerpo del disparador se ejecuta cuando la *condición_disparo* incluida en la cláusula WHEN (si ésta existe) es verdadera.

Los componentes de un disparador son su nombre, el suceso de disparo y su cuerpo, siendo opcional la cláusula WHEN. Nada impide que el nombre de un disparador sea el mismo que el de tablas y procedimientos, pero no es recomendable. Lo que sí es obligatorio es que dentro de un esquema todos los disparadores tengan nombre distinto.

El suceso de disparo determina el tipo de disparador. Los disparadores se definen para las operaciones INSERT, UPDATE o DELETE y pueden dispararse antes o después de la operación. Los tipos de disparadores se resumen en la siguiente tabla:

Categoría	Valores	Explicación
Orden	INSERT, DELETE, UPDATE	Tipo de orden DML que lo activa.
Temporización	BEFORE o AFTER	Se activa antes o después de la ejecución.
Nivel	FILA u ORDEN	Los disparadores de fila se activan una vez por cada fila afectada por la orden de disparo, y los de orden se activan una vez, antes o después de la orden.

También existen disparadores de sustitución, que solo pueden definirse sobre vistas que tienen que tener nivel de fila y que suelen actualizar vistas con uniones.

Los disparadores no pueden contener órdenes SQL de control de transacciones (COMMIT, ROLLBACK o SAVEPOINT) y ningún procedimiento o función llamado por el disparador puede emitir órdenes de control de transacciones. En el cuerpo de un disparador no deben situarse variables LONG o LONG RAW.

Un disparador con nivel de fila se ejecuta una vez por cada fila procesada por la orden que provoca el disparo. Dentro del disparador puede accederse a la fila que está siendo actualmente procesada utilizando para ello los seudorregistros **:old** o **:new**. La tabla siguiente explica estos seudorregistros.

Orden de disparo	:old	:new
INSERT	No definido (campos a NULL).	Valores que se insertarán cuando se complete la orden.
UPDATE	Valores originales de fila, antes de la actualización.	Nuevos valores que serán escritos cuando se complete la orden.
DELETE	Valores originales de fila, antes del borrado de la fila.	No definido (campos a NULL).

Dentro de un disparador que se dispara para distintos tipos de órdenes DML (INSERT, UPDATE y DELETE) hay tres funciones booleanas que pueden utilizarse para determinar de qué operación se trata. Se trata de los predicados INSERTING, UPDATING y DELETING. INSERTING devuelve TRUE si la orden de disparo es INSERT, UPDATING devuelve TRUE si la orden de disparo es UPDATE y DELETING devuelve TRUE si la orden de disparo es DELETE.

Para *eliminar un disparador* se utiliza la sintaxis:

```
DROP TRIGGER nombre_disparador;
```

Para *deshabilitar un disparador* se puede utilizar la orden ALTER TRIGGER como sigue:

```
ALTER TRIGGER nombre_disparador {DISABLE|ENABLE}
```

DISABLE y ENABLE permiten deshabilitar o volver a habilitar cualquier disparador. Se puede deshabilitar un disparador sin necesidad de eliminarlo. Todos los disparadores están, en principio, habilitados en el momento de su creación. ALTER TRIGGER puede deshabilitar y luego habilitar cualquier disparador. También se pueden habilitar o deshabilitar todos los disparadores de una tabla con la orden ALTER TABLE y sus cláusulas ENABLE ALL TRIGGERS o DISABLE ALL TRIGGERS.

Capítulo

8

MONITORIZACIÓN Y AJUSTE DEL RENDIMIENTO

Una vez implementado el esquema físico de una base de datos, se debe poner en marcha para observar sus prestaciones. Si éstas no son las deseadas, el esquema deberá cambiar para intentar satisfacerlas. Una vez afinado el esquema, no permanecerá estático, ya que tendrá que ir cambiando conforme lo requieran los nuevos requisitos de los usuarios. Los SGBD proporcionan herramientas para monitorizar el sistema mientras está en funcionamiento.

Existen muchos factores y parámetros que inciden en el rendimiento de un SGBD y que será necesario ajustar y optimizar. Habitualmente es necesario optimizar el almacenamiento, los procedimientos de transferencia y comunicaciones, la ejecución de consultas y las tareas de mantenimiento preventivo. Los sistemas gestores de bases de datos suelen ofrecer herramientas para la monitorización como trazas, ficheros log, definición de alertas, etc. En este capítulo trataremos en concreto el asistente para la optimización del motor de base de datos de SQL Server y las herramientas de monitorización y ajuste del entorno de administración de Oracle denominado Oracle Enterprise Manager.

8.1 ASISTENTE PARA LA OPTIMIZACIÓN DE MOTOR DE BASE DE DATOS DE SQL SERVER

El asistente para la optimización de motor de base de datos permite ajustar las bases de datos para lograr un procesamiento de consultas mejorado. El asistente para la optimización de motor de base de datos analiza la forma en que se procesan las consultas en las bases de datos especificadas por el usuario y, a continuación, recomienda la forma en que se puede mejorar el rendimiento del procesamiento modificando las estructuras de diseño físico tales como índices, vistas indizadas y particiones.

Sustituye al asistente para optimización de índices de Microsoft SQL Server 2000 y ofrece muchas características nuevas. Por ejemplo, el asistente para la optimización de motor de base de datos proporciona dos interfaces de usuario: una interfaz gráfica de usuario (GUI) y la utilidad del símbolo del sistema dta. La GUI facilita y agiliza la obtención de resultados a partir de las sesiones de optimización, mientras que la utilidad dta facilita la incorporación de

la funcionalidad del asistente para la optimización de motor de base de datos a los scripts con el fin de automatizar la optimización. Además, este asistente admite datos de entrada XML, lo que ofrece un mayor control sobre el proceso de optimización.

Para iniciar el asistente de optimización de motor de base de datos de SQL Server se hará lo siguiente:

- En el menú *Inicio* de Windows, elija *Todos los programas*, *Microsoft SQL Server*, *Herramientas de rendimiento* y, a continuación, haga clic en *Asistente para la optimización de motor de base de datos*.

- En el cuadro de diálogo *Conectar al servidor*, compruebe la configuración predeterminada y, a continuación, haga clic en *Conectar*.

De manera predeterminada, el asistente para la optimización de motor de base de datos abre la configuración que muestra la ilustración de la Figura 8.1.

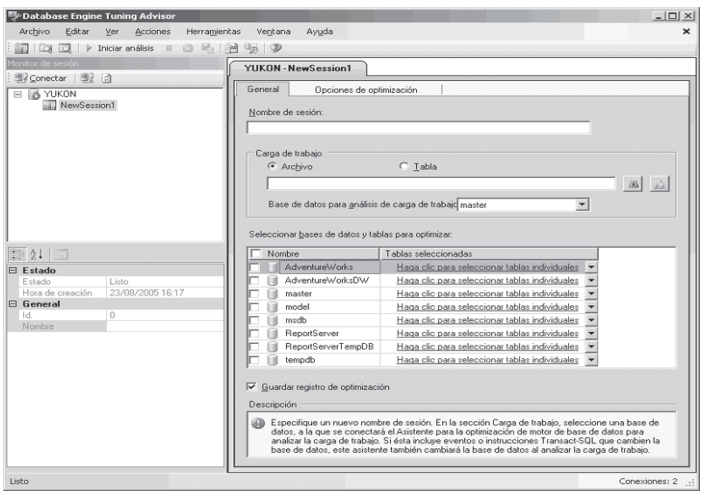


Figura 8.1

Cuando se abre por primera vez, aparecen dos paneles principales en la GUI del asistente para la optimización de motor de base de datos.

- El panel izquierdo contiene el *Monitor de sesión*, que enumera todas las sesiones de optimización que se han realizado en esta instancia de Microsoft SQL Server. Al abrir el asistente para la optimización de motor de base de datos, mostrará una sesión nueva en la parte superior del panel. Puede nombrar esta sesión en el panel adyacente. Inicialmente, solo se muestra una sesión predeterminada. Ésta es la sesión predeterminada que el asistente para la optimización de motor de base de datos crea automáticamente para el usuario. Después de optimizar las bases de datos, todas las sesiones de optimización para la instancia de SQL Server

a las que está conectado se enumerarán debajo de la sesión nueva. Puede hacer clic con el botón secundario en la sesión para cambiarle el nombre, cerrarla, eliminarla o clonarla. Si hace clic con el botón secundario en la lista, podrá ordenar las sesiones por nombre, estado u hora de creación, o bien crear una sesión nueva. En la sección inferior de este panel, se muestran detalles acerca de la sesión de optimización seleccionada. Puede mostrar los detalles organizados por categorías con el botón *Por categorías*, o bien mostrarlos en una lista alfabética utilizando el botón *Alfabético*. También puede ocultar el *Monitor de sesión* arrastrando el borde del panel derecho hacia la parte izquierda de la ventana. Para volver a verlo, arrastre el borde del panel hacia la derecha. El *Monitor de sesión* le permite ver sesiones de optimización previas, o bien utilizarlas para crear sesiones nuevas con definiciones similares. También puede utilizar el *Monitor de sesión* para evaluar recomendaciones de optimización.

- El panel derecho contiene las fichas *General* y *Opciones de optimización*. Aquí es donde puede definir la sesión de optimización del motor de base de datos. En la ficha *General*, escriba el nombre de la sesión de optimización, especifique la tabla o el archivo de carga de trabajo que se va a utilizar y seleccione las bases de datos y tablas que desea optimizar en esta sesión. Una carga de trabajo es un conjunto de instrucciones Transact SQL que se ejecuta en una o varias bases de datos que se desea optimizar. El asistente para la optimización de motor de base de datos utiliza archivos de traza, tablas de traza, scripts Transact SQL o archivos XML como entrada de carga de trabajo a la hora de optimizar bases de datos. En la ficha *Opciones de optimización*, puede seleccionar las estructuras de diseño físico de base de datos (índices o vistas indizadas) y la estrategia de partición que desea que el asistente tenga en cuenta durante el análisis. En esta ficha, también puede especificar el tiempo máximo que el asistente para la optimización de motor de base de datos empleará en optimizar una carga de trabajo. De forma predeterminada, el asistente emplea una hora en optimizar una carga de trabajo.

8.2 OPTIMIZACIÓN Y AJUSTE CON ORACLE ENTERPRISE MANAGER

Oracle Enterprise Manager es un marco adecuado especialmente para tareas de administración. Dispone de una consola administrativa robusta, rica en herramientas, que posibilita la detección, solución y simplificación de un conjunto muy completo de problemas que pueden presentarse en la administración del entorno completo de Oracle. Se accede a la consola de Oracle Enterprise Manager a través del navegador web mediante la URL *http://nombre_host:1158/em.* Se obtiene la pantalla de entrada de la Figura 8.2, en la que hay que introducir el nombre del usuario con rol administrativo y contraseña. Al pulsar en *Conectar* se obtiene la pantalla informativa de la Figura 8.3.

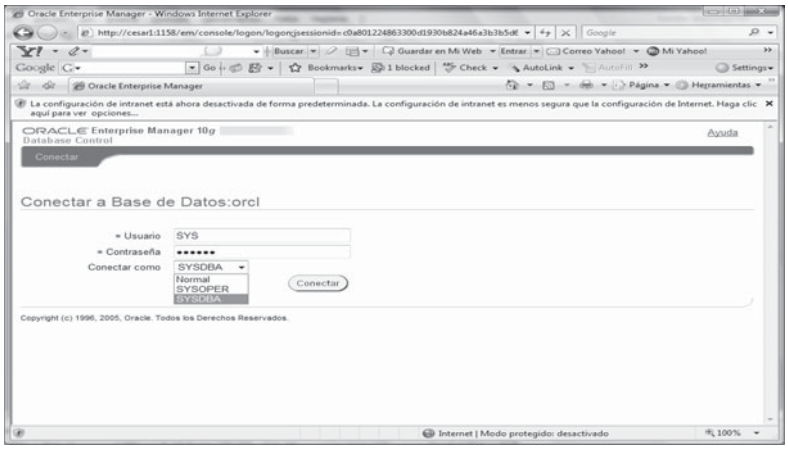


Figura 8.2

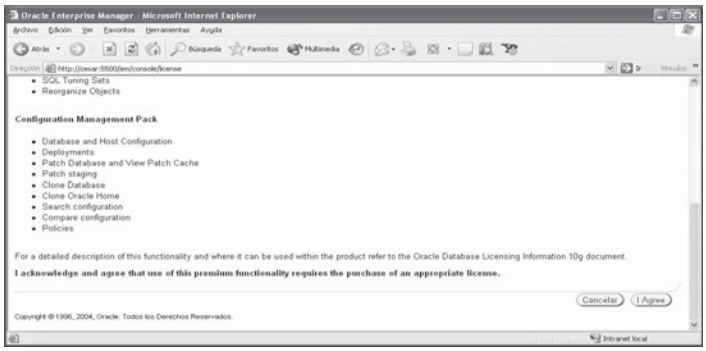


Figura 8.3

Al hacer clic en *I Agree*, se obtiene la página *Inicial* del entorno administrativo (consola) Oracle Enterprise Manager (Figuras 8.4 y 8.5).

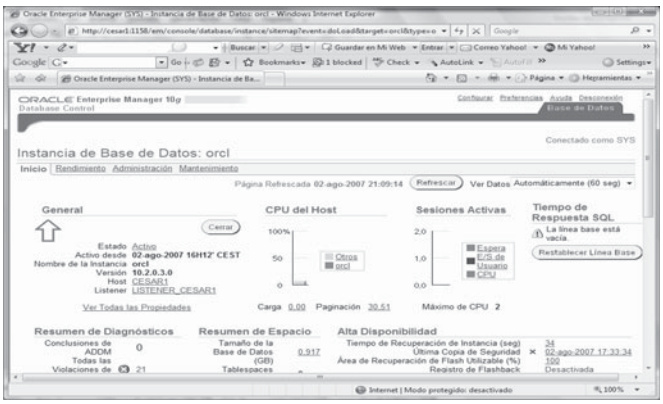


Figura 8.4

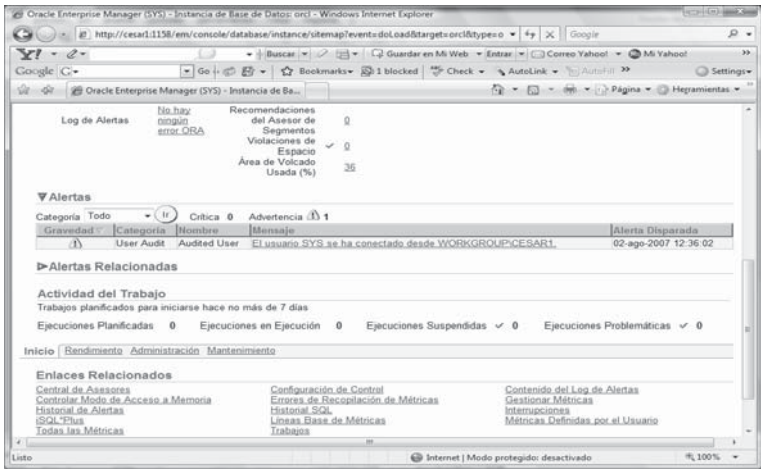


Figura 8.5

8.2.1 Página Inicial de Oracle Enterprise Manager

La página *Inicial* de Oracle Enterprise Manager permite ver el estado actual de la base de datos mostrando una serie de métricas que incluyen el estado general de la base de datos. Proporciona un punto de partida para el estado de la base de datos y la administración y configuración del entorno de base de datos. Contiene cuatro páginas a las que se accede mediante subseparadores y cada una muestra varias subsecciones.

La página *Inicial* muestra varias secciones: *General, Alertas, Alertas Relacionadas* y *Enlaces Relacionados*. Puede utilizar la opción *Cambiar Estado* para iniciar o parar la base de datos.

La sección *General* proporciona una visión rápida del estado de la base de datos y ofrece información básica sobre la base de datos, incluidas las siguientes métricas:

Estado	Indica el estado actual de la base de datos. El estado puede ser *Activo, Caído*, *Interrumpido, Sin Control* o *Desconocido* y el estado se duplica en el icono que aparece a la izquierda de la señal de semáforo.
Activo desde	Muestra la fecha y hora de inicio de la base de datos.
Zona Horaria	Muestra la zona horaria en la que reside la base de datos.
Disponibilidad (%)	Muestra el porcentaje de tiempo que la base de datos ha estado disponible en las últimas 24 horas.
Nombre de la Instancia	Muestra el nombre de instancia de la base de datos.

Versión	Muestra la versión de la base de datos, por ejemplo, 9.0.1.0.0.
Directorio Raíz de Oracle	Muestra la ubicación del directorio raíz de Oracle de la base de datos actual.
Listener	Muestra el listener.
Host	Muestra el nombre del host en el que reside la base de datos.

- La sección *CPU del Host* muestra un gráfico con el uso relativo de la CPU del host de Oracle.

- La sección *Sesiones* muestra las sesiones activas y otras estadísticas SQL. Esta sección informa sobre el número de sesiones activas y el tiempo de respuesta SQL. Para ver las métricas de actividades de SQL, debe configurarlas mediante el asistente de configuración de bases de datos. Los gráficos, como los que aparecen en la página inicial de la base de datos, están diseñados para aprovechar las funciones de SVG.

- La sección *Alta Disponibilidad* muestra la hora de la última copia de seguridad de bases de datos anteriores a Oracle10*g*, la hora de la copia de seguridad más reciente y si la copia se ha realizado correctamente para bases de datos Oracle versión 10*g*. También indica el tiempo de recuperación de instancia estimado, el porcentaje del área de archivado usado y si se ha activado el registro de *flashback*.

- La sección *Uso de Espacio* muestra el tamaño de la base de datos y el número de tablespaces problemáticos que forman la base de datos. *Búsquedas de Segmento* ayuda a identificar problemas relacionados con el almacenamiento y ofrecer recomendaciones para mejorar el rendimiento. El enlace *Área de Volcado Usada* muestra el porcentaje del área de volcado utilizada. La sección *Uso de Espacio*, junto con *Resumen de Diagnósticos*, muestra información sobre las violaciones de política y ADDM. El enlace *Violaciones de Política* informa sobre el número de violaciones de política del destino.

- El enlace *Búsquedas de ADDM* proporciona información sobre el *Monitor de Diagnóstico de Base de Datos Automático*. Éste utiliza instantáneas de la actividad de la base de datos para realizar un análisis completo de la actividad de la base de datos.

- La tabla *Alertas* proporciona información sobre cualquier alerta que se haya emitido junto con el ratio de gravedad de cada una de ellas. Haga clic en el mensaje de alerta de la columna *Mensaje* para obtener más información sobre la alerta. Aparece el icono de gravedad de la alerta (*Advertencia* o *Crítica*), junto con la hora en la que se disparó, el valor de la alerta y la hora de la última comprobación del valor de la métrica.

- La tabla *Alertas Relacionadas* proporciona información sobre las alertas para los destinos relacionados, como el listener y el host, y contiene detalles sobre el mensaje, la hora de disparo de la alerta, el valor y la hora de comprobación de la alerta.

- La sección *Análisis de Rendimiento* muestra los resultados de las tareas ADDM.

- La sección *Actividad del Trabajo* muestra un informe de las ejecuciones de trabajo, con las ejecuciones planificadas, en ejecución, suspendidas y problemáticas.

- La sección *Enlaces Relacionados* permite utilizar la página *Todas las Métricas* para mostrar todas las métricas disponibles de la base de datos actual. Las métricas se muestran por categoría y las que se están recopilando actualmente, se identifican con una marca de control verde. Utilice la página *Editar Umbrales de Métrica* para controlar las condiciones cuando alcancen sus umbrales crítico y de advertencia y para cambiar los umbrales según sea necesario.

Para activar las métricas de actividades de SQL, los archivos de comandos de Tiempo de Respuesta SQL e Informe de Contención de Caché de Biblioteca deben estar instalados para el usuario DBSNMP (usuario de control de la base de datos), así como, Oracle Statspack para el Informe de SQL Principal. Si no instala los paquetes necesarios y configura el destino de la base de datos para utilizar los niveles de control Recomendado o Completo, el estado "No Configurado" aparecerá junto a Control de Actividades de SQL en la página inicial de la base de datos. Las métricas de actividades de SQL (Respuesta SQL, Informe de SQL Principal, SQL Erróneo y Contención de Caché de Biblioteca) se controlan en los niveles de control Recomendado o Completo.

Los paquetes principales de Oracle Enterprise Manager en DBSNMP son:

- Tiempo de Respuesta SQL para *mgmt_response*

- Contención de Caché de Biblioteca para *mgmt_reuse*

Después de la instalación, estos paquetes deben ser válidos para que Enterprise Manager pueda recopilar estadísticas. Compruebe que las métricas estén "configuradas" en la página inicial de la base de datos. Si aparece el estado "No Configurado" junto a los campos, deberá asegurarse de haber instalado los paquetes. Si aparece el estado "No disponible" junto a los campos en la página inicial de la base de datos, se deberá a que los datos aún no se han recopilado en Oracle Intelligent Agent; o bien, la recopilación no estaba planificada o hay un error de recopilación.

8.2.2 Rendimiento de la base de datos

La página *Rendimiento* de Oracle Enterprise Manager (Figuras 8.6 y 8.7) se utiliza para ver el estado global de la base de datos actual e identificar la causa de los cuellos de botella.

Figura 8.6

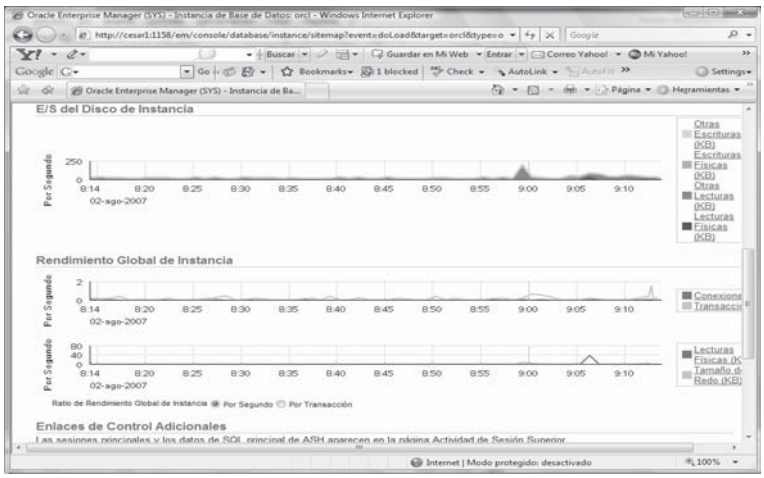


Figura 8.7

Hay tres áreas principales en la página *Rendimiento*: *Host, Sesiones* y *Rendimiento de Instancia*. El gráfico *Actividad de Sesión* de la sección *Sesiones* muestra la carga en la instancia e identifica cuellos de botella de rendimiento. Los diagramas *Host* y *Rendimiento de*

Instancia proporcionan información complementaria. Para identificar rápidamente las áreas problemáticas, el diagrama muestra un bloque de color de mayor tamaño para indicar problemas más graves. Haga clic en las áreas coloreadas para desplegar las páginas relacionadas que muestran las sesiones y las sentencias SQL relacionadas con la clase de espera correspondiente. Puede identificar áreas en las que centrarse observando la línea "*Máximo de CPU*". Si las esperas son mucho más largas que la línea *Máximo de CPU* (por ejemplo, el doble de alto o más), la aplicación se podrá beneficiar por lo general mediante un ajuste. Debe identificar los motivos de las esperas. Una vez identificados, puede decidir si estas esperas son una parte aceptable de la aplicación.

Sesiones: En Espera y En Funcionamiento

El diagrama *Sesiones* proporciona muchos datos y es el punto central de control del rendimiento de Oracle. El diagrama muestra el número de sesiones equivalente de tiempo completo en el eje Y. Este número representa la carga media de la base de datos. Puede haber 200 sesiones conectadas y funcionando simultáneamente en una instancia Oracle, pero si solo hay 10 activas en el mismo punto exacto en el tiempo, el número de sesiones activas del gráfico será 10. De las que están activas, el diagrama muestra las que se están ejecutando en la CPU y las que están en espera de un evento. El área de color verde representa los usuarios en la CPU. Los demás colores representan usuarios en espera de eventos como bloqueos, E/S del disco, escrituras de red o comunicaciones a través de una red. En una situación ideal, todos los usuarios activos se ejecutarían en la CPU sin esperas. Puede determinar el momento en el que las esperas se convierten en un problema mediante la consulta de la línea *Máximo de CPU* y si observa que el número de esperas es mucho mayor que la CPU disponible. Cuando el nivel de esperas supera el doble del nivel de la línea *Máximo de CPU,* debe investigar las posibles acciones de ajuste. Por lo tanto, si las esperas son al menos el doble de la línea *Máximo de CPU*, el tiempo de respuesta se puede reducir a la mitad o se puede doblar el rendimiento si se eliminan las esperas. Algunas esperas son inevitables, por ejemplo, las de E/S. En algún punto, una sesión debe leer datos del disco en la caché de buffers.

Host

Para que la instancia de Oracle funcione correctamente es necesario un estado eficaz del rendimiento del host. El funcionamiento eficaz de la instancia de Oracle depende de una potencia de CPU suficiente, del espacio en memoria, de la respuesta de la red y del rendimiento de E/S. El rendimiento de E/S y la respuesta de red se indican directamente en las estadísticas de Oracle. No obstante, la falta de potencia de CPU o de espacio en memoria solo se pueden deducir. Por lo tanto, antes de revisar las estadísticas de Oracle, debe comprobar la CPU y la memoria disponible en el host.

La sección *Host* muestra dos gráficos: *Cola Media de Ejecución* y *Ratio de Localización.* Cola Media de Ejecución indica el nivel de contención del tiempo de CPU. La disponibilidad de CPU se mide con mayor frecuencia a partir de la estadística *Porcentaje de Uso de CPU*, pero

esta estadística, aunque es buena para la planificación de la capacidad, puede confundir o ser incompleta para el diagnóstico de los problemas de rendimiento inmediatos. Por ejemplo, un sistema puede informar sobre un uso de CPU del 100%, pero el tiempo de respuesta puede ser bueno porque no hay procesos en espera para acceder a la CPU. Una métrica más completa es *Cola de Ejecución de CPU*. Cola de Ejecución informa del número de procesos que están preparados para su ejecución pero que no tienen acceso a CPU porque hay otros procesos en competición. Si la cola de ejecución aumenta, crecerá la contención interna de recursos dentro de Oracle. Por ejemplo, si un usuario que tiene un bloqueo no puede acceder a la CPU, mientras que otros usuarios que necesitan ese bloqueo sí lo hacen, no avanzará ningún usuario hasta que el propietario del bloqueo acceda una vez más a la CPU y realice la transacción. Un ejemplo más común es el de los bloqueos internos de Oracle, que son similares a los bloqueos rápidos. Si se elimina al propietario de un bloqueo de la CPU y cualquier usuario que acceda a la CPU necesita el bloqueo, ningún otro usuario podrá continuar hasta que el primer usuario con el bloqueo acceda a la CPU y libere el bloqueo.

Otra estadística importante es *Ratio de Localización*. Este valor informa del ratio al que el host escribe páginas de memoria para su intercambio con el fin de crear más espacio en la memoria. Si hay memoria suficiente, el sistema deberá tener poca o ninguna localización. A medida que se llena la memoria y aumenta la competencia por el espacio, el host empieza a escribir las páginas menos utilizadas de espacio. Cuando un proceso se ejecuta en la CPU y necesita una página que se ha escrito para su intercambio, debe esperar a que una E/S física acceda a la página, lo cual es una operación relativamente lenta. Al igual que la necesidad de CPU, la localización puede aumentar la contención interna de Oracle como la contención de bloqueo interno. Si hay un problema en la base de datos, el primer paso consistirá en garantizar que la máquina host funcione bien.

Rendimiento de Instancia

Los diagramas de rendimiento reflejan la importancia de cualquier contención que se muestra en el diagrama Actividad de Sesión. Si este diagrama muestra un mayor número de sesiones en espera, lo que indica una contención interna, pero el rendimiento va en aumento, la situación podrá ser aceptable. Si disminuye el rendimiento pero aumenta la contención interna, podrá realizar determinadas acciones de ajuste.

Enlaces de Control Adicionales

En la sección *Control*, puede seleccionar los temas que muestran información o proporcionan una rápida visión general de cada tema de la base de datos que aparece a continuación:

- *Consumidores Principales*: utilice la página *Consumidores Principales* para mostrar una lista de los consumidores de recursos principales de la base de datos actual (Figura 8.8). Puede observar la lista de servicios, módulos, clientes y acciones y su consumo de recursos.

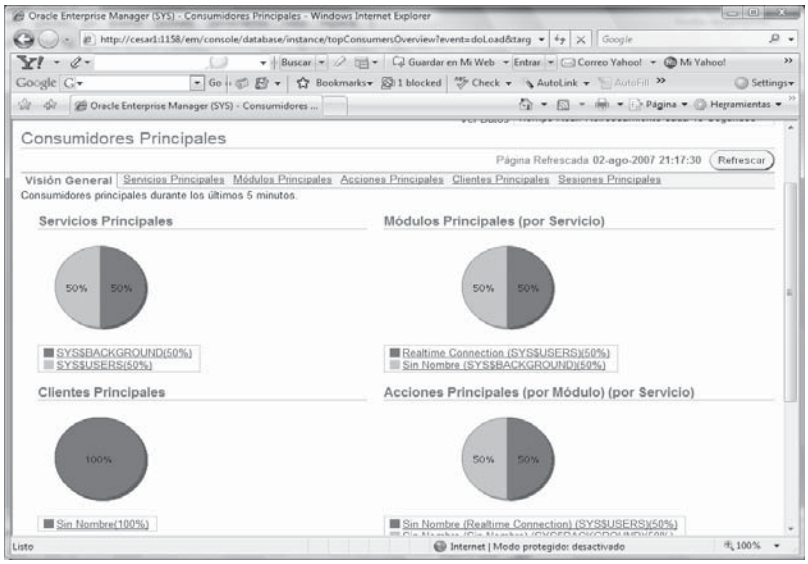


Figura 8.8

- *SQL Duplicado*: este informe (Figura 8.9) identifica sentencias SQL similares que puede compartir una única sentencia SQL si la aplicación de base de datos ha utilizado variables ligadas para sustituir los literales y convenciones de codificación SQL para eliminar las diferencias basadas solo en las mayúsculas/minúsculas de los caracteres o en los espacios en blanco. Puede volver a escribir las sentencias SQL para utilizar de forma eficaz una única sentencia compartida.

Figura 8.9

- Sesiones Bloqueantes: utilice la página Sesiones Bloqueantes para mostrar una tabla con las sesiones bloqueantes actuales (Figura 8.10).

- Actividad de instancias y Bloqueo de instancias: utilice estas páginas (Figuras 8.11 y 8.12) para ver la actividad y bloqueos de las instancias de base de datos con datos sobre grupos de métricas como cursores, sesiones y transacciones.

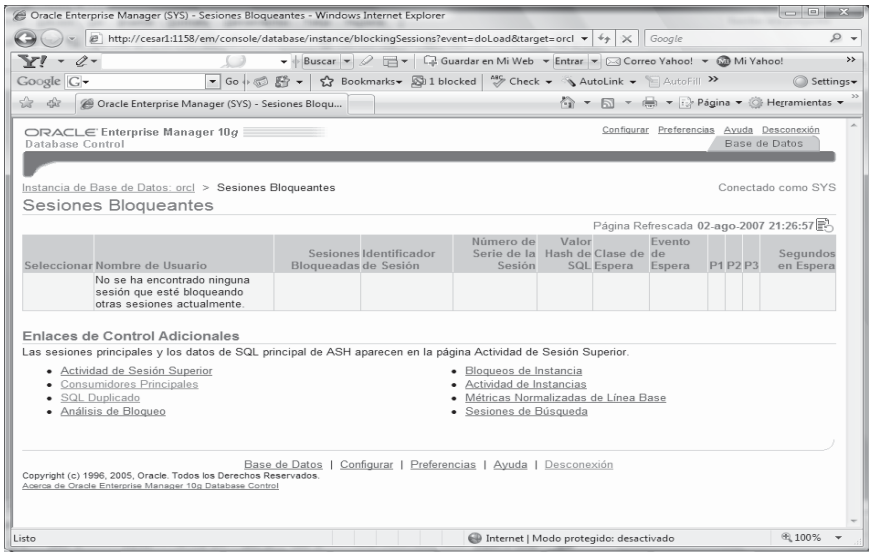


Figura 8.10

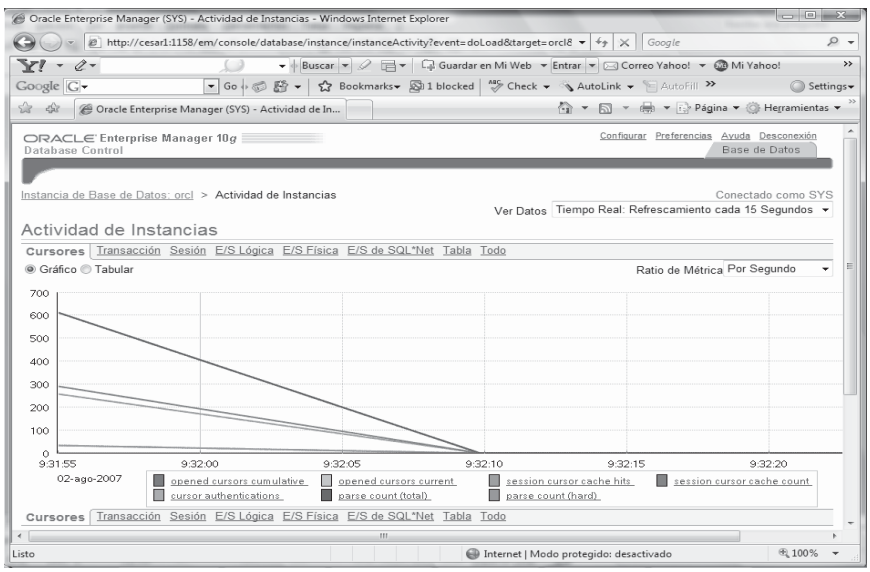


Figura 8.11

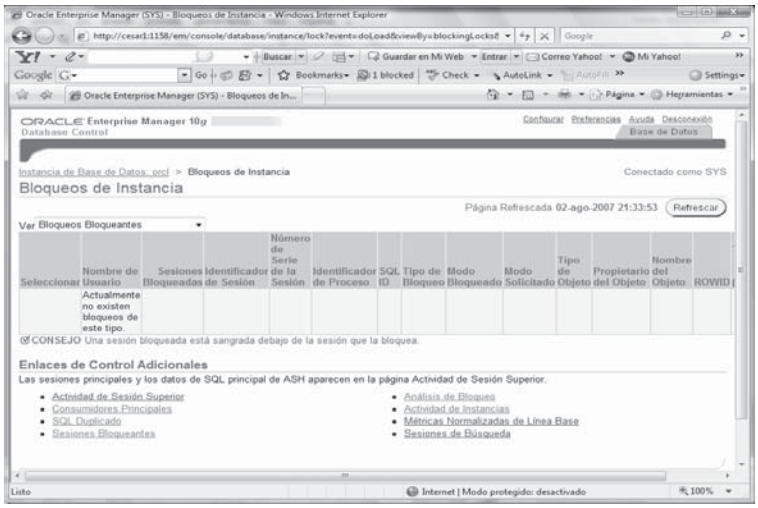


Figura 8.12

- Sesiones de búsqueda: el usuario de base de datos puede realizar búsquedas que devuelven todas las coincidencias en mayúsculas. Para ejecutar una búsqueda de coincidencia exacta o sensible a mayúsculas/minúsculas, introduzca entre comillas la cadena de búsqueda. Para otros filtros, la búsqueda devuelve todas las coincidencias sensibles a mayúsculas/minúsculas. Siempre puede utilizar el símbolo comodín (%) (Figura 8.13).

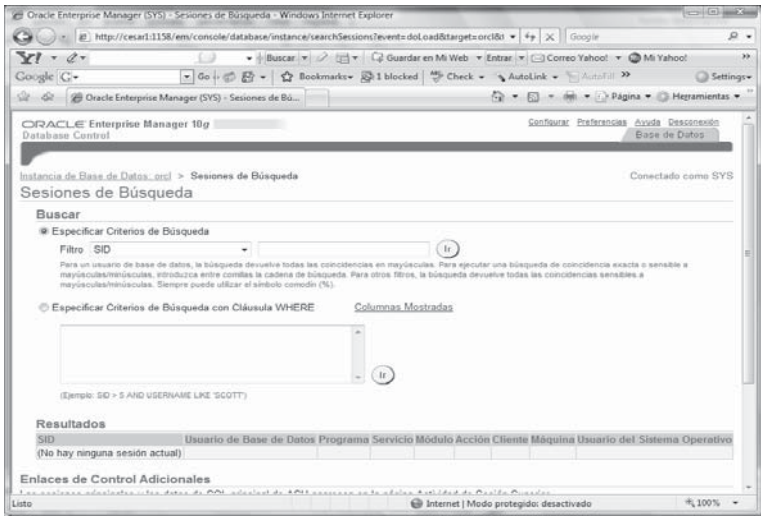


Figura 8.13

Las métricas de actividad de cada tipo aparecen bajo los diagramas. Para obtener más información sobre cada métrica, haga clic en el nombre de la métrica.

8.2.3 Configuración y ajuste a través de la página de Administración de la Base de Datos

La página *Administración* de Oracle Enterprise Manager (Figura 8.14) permite configurar y ajustar algunos aspectos de la base de datos para mejorar el rendimiento y ajustar los valores. Utilice la página *Administración* de Oracle Enterprise Manager para realizar las siguientes tareas:

- Crear y abrir la base de datos.
- Gestionar la seguridad del sistema a través de usuarios y roles.
- Gestionar los recursos de la base de datos.
- Implementar el diseño de la base de datos mediante la gestión de tipos de orígenes y objetos de esquema.
- Gestionar las configuraciones de base de datos.
- Configurar almacenes de datos con vistas materializadas y metadatos de OLAP.

Concretamente, Enterprise Manager proporciona una interfaz de usuario gráfica para gestionar las estructuras de almacenamiento y el esquema de la base de datos. Haga clic en el tema correspondiente en la figura 8.14 para obtener más información sobre el recurso.

Figura 8.14

8.3 MANTENIMIENTO DE LA BASE DE DATOS

Utilice la página *Mantenimiento* de Oracle Enterprise Manager (Figura 8.15) para realizar tareas como exportar e importar datos de archivos, cargar datos en una base de datos Oracle desde un archivo y recopilar, calcular y suprimir estadísticas mejorando el rendimiento de las consultas SQL en los objetos de la base de datos.

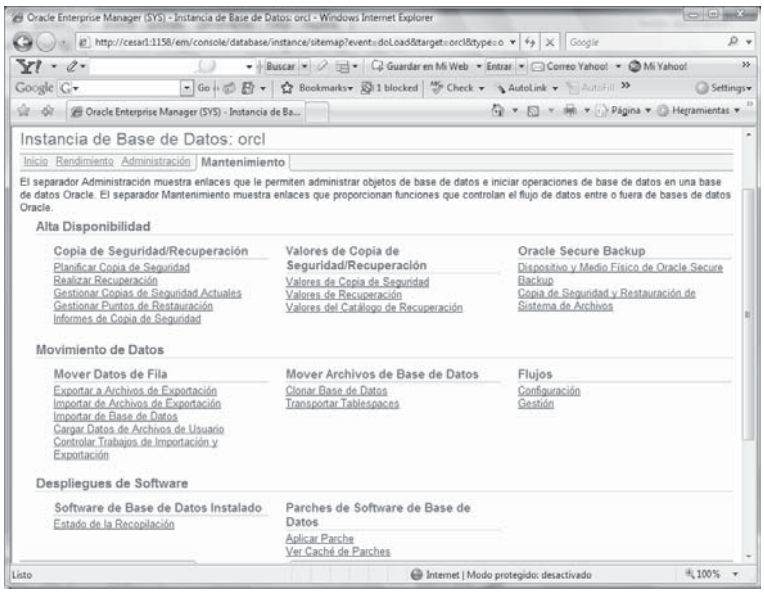


Figura 8.15

La página *Mantenimiento* de Oracle Enterprise Manager consta de dos secciones principales: *Alta Disponibilidad* y *Movimiento de Datos*.

En *Movimiento de Datos*, las funciones de la sección *Mover Datos de Fila* permiten realizar las siguientes tareas:

- *Exportar a Archivos de Exportación*. Utilice el *Asistente de Exportación* para exportar el contenido de una base de datos, de los objetos de los esquemas del usuario y de las tablas.
- *Importar de Archivos de Exportación*. Utilice el *Asistente de Importación* para importar el contenido de objetos y de tablas.
- *Importar de Base de Datos*. Utilice el *Asistente de Importación* para importar el contenido de una base de datos.

- *Cargar Datos de Archivos de Usuario*. Utilice el *Asistente de Carga de Datos de Archivo* para cargar datos de una base de datos no Oracle en una base de datos Oracle.
- Controlar Trabajos de Importación y Exportación. Control de importar y exportar.

En *Movimiento de Datos*, las funciones de la sección *Mover Archivos de Base de Datos* permiten realizar las siguientes tareas:

- *Clonar Base de Datos*. Utilice el *Asistente de Clonación de Base de Datos* para duplicar la base de datos actual. Para ello, realice primero una copia de seguridad y, a continuación, transfiera la base de datos actual a un directorio raíz de Oracle de destino.
- *Transportar Tablespaces*. Se puede utilizar para mover un subjuego de una base de datos Oracle y conectarlo a otra base de datos Oracle, esencialmente moviendo tablespaces entre las bases de datos. Mover datos con tablespaces transportables es mucho más rápido que realizar una exportación/importación o una descarga/carga de los mismos datos.

En *Movimiento de Datos*, las funciones de la sección *Flujos* permiten realizar las siguientes tareas:

- *Configuración*. *El Asistente de Configuración de Streams* permite configurar y replicar la base de datos completa, esquemas específicos o tablas específicas entre dos bases de datos. El *Asistente de Replicación de Tablespaces de Streams* permite realizar la replicación y el mantenimiento de tablespaces entre bases de datos. *Messaging* permite la creación y la configuración de colas.
- *Gestión*. *Oracle Streams* permite compartir información. Puede además compartir cambios de la base de datos y otra información en un flujo, el cual puede propagar eventos dentro de una misma base de datos o de una base de datos a otra. La información especificada se direcciona a los destinos especificados. El resultado es una función que proporciona mayor funcionalidad y flexibilidad que las soluciones tradicionales para capturar y gestionar información y para compartir esta última con otras bases de datos y aplicaciones.

En *Alta Disponibilidad*, las funciones de la sección *Copia de Seguridad/Recuperación* permiten realizar las siguientes tareas:

- *Planificar Copia de Seguridad*. Utilice el *Asistente de Copia de Seguridad* para realizar una copia de seguridad del contenido de la base de datos en un disco, en una cinta o en ambos.
- *Realizar Recuperación*. Utilice el *Asistente de Recuperación* para restaurar o recuperar una base de datos, tablespaces, archivos de datos, archive logs o realizar flashbacks de tablas o recuperar objetos.

- *Gestionar Copias de Seguridad Actuales*. Utilice *Gestión de Copias de Seguridad* para buscar y mostrar una lista de juegos de copias de seguridad o copias de seguridad, así como para realizar operaciones de gestión como comprobaciones cruzadas y supresiones de copias, archivos o juegos seleccionados.
- *Gestionar Puntos de Restauración*. Un punto de restauración es un nombre definido por el usuario relativo al estado de la base de datos en un punto en el tiempo.
- *Informes de Copia de seguridad*. Permiten gestionar la información sobre los trabajos de copia de seguridad que se han comunicado a la base de datos. Los datos se recuperan del archivo de control de la base de datos.

En *Alta Disponibilidad*, las funciones de la sección *Valores de Copia de Seguridad/Recuperación* permiten realizar las siguientes tareas:

- *Valores de Copia de Seguridad*. Utilice esta página para configurar los valores para la utilidad de copia de seguridad.
- *Valores de Recuperación*. Utilice esta página para configurar los valores para la utilidad de recuperación.
- *Valores de Catálogo de Recuperación*. Utilice el catálogo de recuperación para registrar la base de datos actual con un catálogo de recuperación.

En *Alta Disponibilidad*, las funciones de la sección *Oracle Secure Backup* permiten realizar las siguientes tareas:

- *Dispositivo y Medio Físico de Oracle Secure Backup.* Permite especificar medio físico y dispositivo de backup.
- *Copia de Seguridad y Restauración de Sistema de Archivos.* Permite realizar copias de seguridad y restauración de sistemas de archivos.

En *Despliegues de Software,* las funciones de la sección *Parches de Software de Bases de Datos* tienen la finalidad de simplificar la aplicación de parches de software en cualquier versión de la base de datos Oracle en ejecución. La sección *Software de Bases de Datos Instalado* recopila una vez al día el estado del software instalado.

8.3.1 Administración de instancias

La funcionalidad *Configuración de la Base de Datos* ayuda a administrar parámetros de instancias de bases de datos y sesiones en el entorno Oracle de forma optimizada con vistas a un rendimiento óptimo. Se puede arrancar y desconectar la base de datos, ver y editar los valores de los parámetros de la instancia, administrar sesiones de usuarios y ver la sentencia SQL que está ejecutándose, administrar transacciones, visualizar operaciones de larga

duración, administrar configuraciones de almacenamiento y administrar sesiones que consumen una alta cantidad de recursos. Bajo el nodo *Configuración de Base de Datos* (Figura 8.14) aparecen los siguientes objetos:

- *Parámetros de memoria*. Enterprise Manager administra parámetros de memoria (Figura 8.16) del *área global del sistema (SGA)*, que es un grupo de estructuras de memoria compartidas que contienen datos y controlan la información de una instancia de la base de datos Oracle. También administra parámetros de memoria del *área global de programa (PGA)*, que es una región de la memoria que contiene datos e información de control para un proceso del servidor. Es una memoria no compartida que crea Oracle cuando se inicia un proceso del servidor.
- *Gestión de deshacer*. La información de deshacer validada por lo general se pierde cuando su espacio de deshacer se sobrescribe con una transacción nueva. Sin embargo, para fines de lectura consistente, las consultas de larga ejecución pueden necesitar información de deshacer antigua para deshacer cambios o producir imágenes más antiguas de bloques de datos.
- *Todos los parámetros de inicialización*. Enterprise Manager permite crear o editar los parámetros de inicialización de la base de datos actual. El archivo de parámetros de inicialización contiene una lista de los parámetros de configuración para la instancia y la base de datos. Puede definir estos parámetros en valores concretos para inicializar muchos de los valores de memoria y de proceso de una instancia Oracle.
- *Uso de funciones de la base de datos*. Enterprise Manager utiliza la página *Uso de Funciones de la Base de Datos* para obtener las estadísticas de uso de las funciones de la base de datos, que proporcionan una aproximación de la frecuencia con la que se utilizan las diferentes funciones de la base de datos.

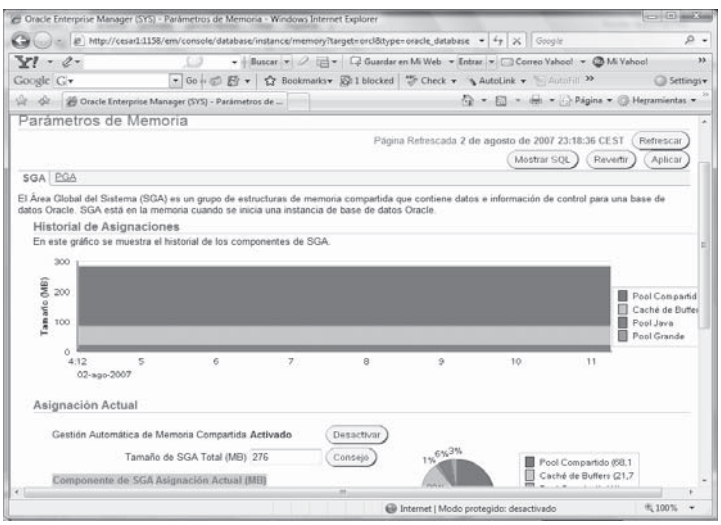


Figura 8.16

8.3.2 Administración del almacenamiento

La funcionalidad *Almacenamiento* de la Figura 8.14 permite administrar de modo óptimo espacios de tablas (tablespaces), segmentos de restauración (*rollback segments*), ficheros de datos y ficheros de registro de actividad (*ficheros log* y *redo log*). Las opciones del nodo de administración *Almacenamiento* son las siguientes:

- *Archivos de Control*. Enterprise Manager puede mostrar información detallada sobre el archivo de control para la base de datos actual. Cada base de datos Oracle mantiene un archivo de control con información sobre la base de datos asociada necesaria para que una instancia acceda durante el inicio y durante el funcionamiento normal. Este archivo almacena el estado de la estructura física de la base de datos.

- *Tablespaces*. Enterprise Manager puede administrar los espacios de tabla de la base de datos.

- *Archivos de Datos*. Enterprise Manager permite la administración de los archivos de datos de la base de datos.

- *Segmento de Rollback*. Enterprise Manager permite la administración de los segmentos de restauración para la base de datos.

- *Grupos de Redo Logs*. Enterprise Manager permite la administración de grupos de ficheros de rastreo de la base de datos.

- *Archive Logs*. Enterprise Manager permite la administración de archivos de registro de actividad de la base de datos.

- *Grupos de Tablespaces Temporales*. Enterprise Manager permite, mediante la página *Grupos de Tablespaces Temporales* (Figura 8.17), la administración de grupos de espacios de tablas temporales para la base de dato.

Capítulo

9

SISTEMAS GESTORES DE BASES DE DATOS DISTRIBUIDOS

A lo largo de los últimos años se ha observado una tendencia en la gestión de datos a pasar de grandes computadores centralizados a redes distribuidas de sistemas informáticos. Con la llegada de los minicomputadores, las tareas de procesamiento de datos como el control de inventarios y procesamiento de pedidos pasaron de maxicomputadores corporativos a sistemas departamentales más pequeños. El incremento explosivo en la popularidad del computador personal en los ochenta trajo la potencia del computador directamente a las mesas de despacho de millones de personas.

Conforme los computadores y las redes de computadores se extendían a través de las organizaciones, los datos ya no residían en un único sistema bajo el control de un único DBMS. En vez de ello, los datos fueron extendiéndose a través de muchos sistemas diferentes, cada uno con su propio gestor de base de datos y comunicados entre sí formando un sistema distribuido.

9.1 CARACTERÍSTICAS DE UN SISTEMA GESTOR DE BASES DE DATOS DISTRIBUIDO. VENTAJAS E INCONVENIENTES

Los sistemas distribuidos debieran tener un grupo de características ideales entre las que se encuentran las siguientes:

- *Transparencia de ubicación*. El usuario no debería tener que preocuparse acerca de dónde están localizados físicamente los datos. El DBMS debería presentar todos los datos como si fueran locales y debería ser responsable de mantener esa ilusión.

- *Sistemas heterogéneos*. El DBMS debería soportar datos almacenados en sistemas diferentes, con diferentes arquitecturas y niveles de rendimiento, incluyendo PC, estaciones de trabajo, servidores de LAN, minicomputadoras y maxicomputadores.

- *Transparencia de red*. Excepto por las diferencias en rendimiento, el DBMS debería trabajar de la misma manera sobre diferentes redes, desde las LAN de alta velocidad a los enlaces telefónicos de baja velocidad.

- *Consultas distribuidas*. El usuario debería ser capaz de componer datos procedentes de cualquiera de las tablas de la base de datos (distribuida), incluso si las tablas están localizadas en sistemas físicos diferentes.

- *Actualizaciones distribuidas*. El usuario debería ser capaz de actualizar datos en cualquier tabla para la que tenga los privilegios necesarios, tanto si la tabla está en un sistema local como si está en un sistema remoto.

- *Transacciones distribuidas*. El DBMS debe soportar transacciones (utilizando COMMIT y ROLLBACK) a través de fronteras de sistema, manteniendo la integridad de la base de datos (distribuida) incluso en presencia de fallos de red y fallos de sistemas individuales.

- *Seguridad*. El DBMS debe proporcionar un esquema de seguridad adecuado para proteger la base de datos (distribuida) completa frente a formas no autorizadas de acceso.

- *Acceso universal*. El DBMS debería proporcionar acceso uniforme y universal a todos los datos de la organización.

Aunque todas las características anteriores son muy ventajosas, existen claros obstáculos que suelen impedir su cumplimiento, entre los que podríamos citar los siguientes:

- *Rendimiento*. En una base de datos centralizada, la trayectoria desde el DBMS a los datos tiene una velocidad de acceso de unos pocos milisegundos y una velocidad de transferencia de varios millones de caracteres por segundo. Incluso en una red de área local rápida, las velocidades de acceso se alargan a décimas de segundo y las velocidades de transferencia caen. Esta gran diferencia en velocidades puede hacer bajar dramáticamente el rendimiento del acceso a datos remotos.

- *Integridad*. Las transacciones distribuidas requieren cooperación activa de dos o más copias independientes del software DBMS ejecutándose sobre sistemas

informáticos diferentes si las transacciones van a seguir siendo proposiciones "todo o nada". Deben utilizarse protocolos especiales de transacción.

- *SQL estático*. Una sentencia de SQL incorporado estático se compila y almacena en la base de datos como un plan de aplicación. Cuando una consulta combina datos de dos o más bases de datos, ¿dónde debería almacenarse su plan de aplicación? ¿Deben existir dos o más planes cooperantes? Si hay un cambio en la estructura de una base de datos, ¿cómo se notifica a los planes de aplicación de las otras bases de datos?

- *Optimización*. Cuando los datos se acceden a través de una red, las reglas normales de optimización de SQL no son aplicables. Por ejemplo, puede ser más eficaz rastrear secuencialmente una tabla local completa que utilizar una búsqueda por índice en una tabla remota. El software de optimización debe conocer las redes y sus velocidades. En general, la optimización pasa a ser más crítica y más difícil.

- *Compatibilidad de datos*. Diferentes sistemas informáticos soportan diferentes tipos de datos, e incluso cuando dos sistemas ofrecen los mismos tipos de datos utilizan con frecuencia formatos diferentes.

- *Catálogos de sistema*. Cuando un DBMS realiza sus tareas, efectúa accesos muy frecuentes a sus catálogos de sistema. ¿Dónde debería mantenerse el catálogo en una base de datos distribuida? Si está centralizado en un sistema, el acceso remoto a él será lento, afectando al DBMS. Si está distribuido a través de muchos sistemas diferentes, los cambios deben propagarse alrededor de la red y deben sincronizarse.

- *Entorno de vendedores mixtos*. Es altamente improbable que todos los datos de una organización puedan ser gestionados por un único producto DBMS, por lo que el acceso a base de datos distribuida requiere cooperación activa entre productos DBMS de vendedores en competencia, lo que siempre será una dificultad.

- *Interbloqueos distribuidos*. Cuando las transacciones de dos sistemas diferentes tratan de acceder a datos cerrados en el otro sistema, puede ocurrir un interbloqueo en la base de datos distribuida, aun cuando el interbloqueo no sea visible a ninguno de los dos sistemas por separado. El DBMS debe proporcionar la detección del interbloqueo global para una base de datos distribuida.

- *Recuperación*. Si uno de los sistemas que ejecuta un DBMS distribuido falla, el operador de ese sistema debe ser capaz de ejecutar sus procedimientos de recuperación con independencia del resto de los sistemas en la red, y el estado recuperado de la base de datos debe ser consistente con el de los otros sistemas.

9.2 ETAPAS EN EL ACCESO A DATOS DISTRIBUIDOS

IBM propuso cuatro etapas en el modelo distribuido. Podríamos resumirlas como sigue:

Etapa	Descripción
Petición remota	Cada sentencia SQL accede a una única base de datos remota y cada sentencia es una transacción.
Transacción remota	Cada sentencia SQL accede a una única base de datos remota. Se soportan transacciones multisentencia para una única base de datos.
Transacción distribuida	Cada sentencia SQL accede a una única base de datos remota. Se soportan transacciones multisentencia a través de múltiples bases de datos.
Petición distribuida	Las sentencias SQL pueden acceder a múltiples bases de datos. Se soportan transacciones multisentencia a través de múltiples bases de datos.

9.2.1 Peticiones remotas

La primera etapa de acceso a datos distribuidos, tal como la define IBM, es una petición remota, tal y como se ilustra en la Figura 9.1. En esta etapa, el usuario de un PC puede emitir una sentencia SQL que consulte o actualice datos en una única base de datos remota. Cada sentencia SQL individual opera como su propia transacción, similar al modo de autocumplimentación proporcionada por muchos programas SQL interactivos. El usuario puede emitir una secuencia de sentencias SQL para varias bases de datos, pero el DBMS no soporta transacciones multisentencia.

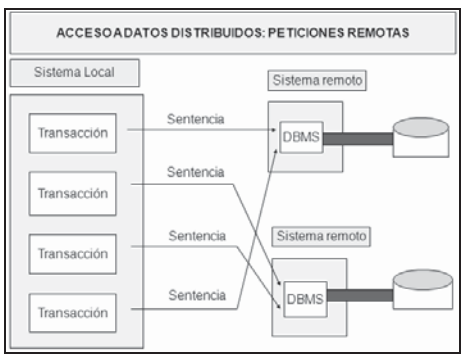


Figura 9.1

Las peticiones remotas son muy útiles cuando un usuario PC necesita consultar datos corporativos. Generalmente los datos requeridos estarán localizados dentro de una única base de datos, como por ejemplo una base de datos de procesamiento de pedidos o de datos de fabricación. Utilizando una petición remota, el programa del PC puede recuperar los datos remotos para su procesamiento por parte de una hoja de cálculo, un programa gráfico o un paquete de autoedición en el PC.

9.2.2 Transacciones remotas

La segunda etapa de acceso a datos distribuidos es una transacción remota (Figura 9.2). Las transacciones remotas extienden la etapa de petición remota para incluir soporte de transacción multisentencia. El usuario del PC puede emitir una serie de sentencias SQL que consultan o actualizan datos en una base de datos remota y luego cumplimentar o volver atrás la serie entera de sentencias como una única transacción. El DBMS garantiza que la transacción completa tendrá éxito o fallará como una unidad, como ocurre con las transacciones sobre una base de datos local. Sin embargo, todas las sentencias SQL que forman la transacción deben referenciar una única base de datos remota.

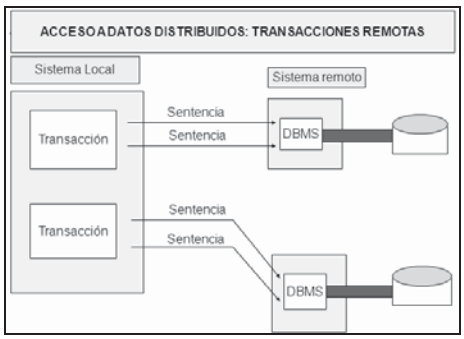


Figura 9.2

Las transacciones remotas abren la puerta a las aplicaciones de procesamiento de transacciones distribuido. Por ejemplo, en el ejemplo de procesamiento de pedidos descrito en la sección anterior, el programa de PC puede ahora efectuar una secuencia de consultas, actualizaciones e inserciones en la base de datos de inventario para procesar un nuevo pedido.

La capacidad de transacción remota requiere típicamente un DBMS (o al menos una lógica de procesamiento de transacciones) sobre el PC además del sistema donde está localizada la base de datos. La lógica de transacción del DBMS debe extenderse a través de la red para asegurar que los sistemas local y remoto tengan siempre la misma opinión referente a si la transacción ha sido cumplimentada. Sin embargo, la responsabilidad efectiva de mantener la integridad de la base de datos sigue estando en el DBMS remoto.

9.2.3 Transacciones distribuidas

La tercera etapa de un acceso de datos distribuidos es una transacción distribuida (Figura 9.3). En esta etapa, cada sentencia SQL individual aún consulta o actualiza una única base de datos sobre un único sistema informático remoto. Sin embargo, la secuencia de sentencias SQL dentro de una transacción puede acceder a dos o más bases de datos localizadas sobre sistemas diferentes. Cuando la transacción se cumplimenta o se vuelve atrás, el DBMS garantiza que todas las partes de la transacción, sobre todos los sistemas implicados en la transacción, serán cumplimentadas o vueltas atrás. El DBMS garantiza específicamente que no habrá una "transacción parcial", donde la transacción se cumplimente en un sistema y vuelta atrás en otro. Las transacciones distribuidas soportan el desarrollo de aplicaciones de procesamiento de transacciones muy sofisticadas.

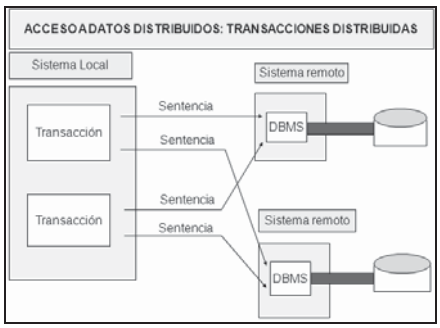


Figura 9.3

Las transacciones distribuidas son mucho más difíciles de proporcionar que las dos primeras etapas de acceso a datos distribuidos. Es imposible proporcionar transacciones distribuidas sin la cooperación activa de los sistemas DBMS individuales implicados en la transacción. Un protocolo de transacción especial, llamado protocolo de cumplimentación en dos fases, es utilizado generalmente para forzar esta cooperación.

9.2.4 Peticiones distribuidas

La cuarta etapa de acceso a datos distribuidos en el modelo IBM es una petición distribuida (Figura 9.4). En esta etapa, una sola sentencia SQL puede referenciar tablas de dos o más bases de datos localizadas en diferentes sistemas informáticos. El DBMS es responsable de llevar a cabo automáticamente la sentencia a través de la red. Una secuencia de sentencias de petición distribuida puede agruparse para formar una transacción. Como en la etapa previa de transacción distribuida, el DBMS debe garantizar la integridad de la transacción distribuida sobre todos los sistemas que estén implicados. La etapa de petición distribuida no efectúa ninguna nueva demanda sobre la lógica de procesamiento de transacción DBMS, ya que DBMS tenía que soportar transacciones a través de fronteras de sistema en la etapa previa de transacción distribuida. Sin embargo, las peticiones distribuidas

presentan nuevos retos importantes para la lógica de optimización de DBMS. El optimizador debe considerar ahora la velocidad de red cuando evalúe métodos alternativos para llevar a cabo una sentencia SQL. Si el DBMS local debe acceder repetidamente a parte de una tabla remota (por ejemplo, cuando realiza una composición), puede ser más rápido copiar parte de la tabla a través de la red en una larga transferencia de bloques en lugar de recuperar repetidamente filas individuales a través de la red.

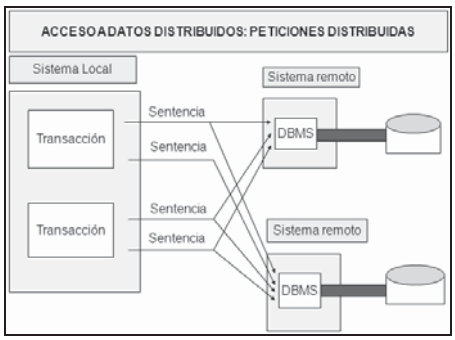


Figura 9.4

El optimizar también debe decidir qué copia del DBMS debería manejar la ejecución de la sentencia. Si la mayoría de las tablas están en un sistema remoto, puede ser una buena idea que el DBMS remoto en ese sistema ejecute la sentencia. Sin embargo, puede ser una mala elección si el sistema remoto tiene una fuerte carga de trabajo. Por tanto, la tarea del optimizador es más compleja y mucho más importante en una petición distribuida.

Finalmente, el objetivo de la etapa de petición distribuida es hacer que la base de datos distribuida total se asemeje a una gran base de datos desde el punto de vista del usuario. Idealmente, el usuario debería tener acceso completo a cualquier tabla de la base de datos distribuida y debería poder utilizar transacciones SQL sin necesidad de conocer la ubicación física de los datos. Desgraciadamente, este escenario ideal demostraría rápidamente ser poco práctico en redes reales. En una red de cualquier tamaño, el número de tablas de la base de datos distribuida pasaría a ser rápidamente muy grande, y los usuarios encontrarían imposible hallar datos de interés. Los id-usuarios de todas las bases de datos de la organización tendrían que estar coordinados para asegurarse que un id-usuario determinado identificase unívocamente a un usuario en todas las bases de datos. La administración de la base de datos también sería muy difícil.

En la práctica, por tanto, las peticiones distribuidas deben ser implementadas selectivamente. Los administradores de bases de datos deben decidir qué tablas remotas van a hacerse visibles a usuarios locales y cuáles permanecerán ocultas. Las copias de DBMS cooperativos deben traducir los id-usuarios de un sistema a otro, permitiendo que cada base de datos sea administrada automáticamente mientras sigue proporcionando seguridad para acceso a datos remotos. Las peticiones distribuidas que pudieran consumir demasiados recursos DBMS o de red deben ser detectadas y prohibidas antes de que afecten al rendimiento global del DBMS.

9.3 TABLAS DISTRIBUIDAS

La etapas definidas anteriormente de acceso a datos distribuidos tratan a una tabla de base de datos como una unidad indivisible. Suponen que una tabla está localizada en un único sistema, en una única base de datos, bajo el control de una única copia del DBMS. Pero en el trabajo con bases de datos distribuidas se ha relajado esta restricción, permitiendo que una tabla individual esté distribuida a través de dos o más sistemas. Existen dos tipos diferentes de división de tabla: horizontal y vertical.

9.3.1 Divisiones horizontales de tabla

Suele ser habitual dividir la tabla horizontalmente para distinguirla a través de la red, colocando diferentes filas de la tabla en diferentes sistemas. La Figura 9.5 muestra un ejemplo donde es útil una división horizontal de tabla. En esta aplicación, una empresa opera con tres centros de distribución, cada uno con su propio sistema informático y su propio DBMS para gestionar una base de datos de inventario. La mayor parte de la actividad de cada centro de distribución afecta a datos almacenados localmente, pero la empresa tiene una política de satisfacer un pedido de clientes desde cualquier centro de distribución si el centro local no dispone de un producto particular.

Para implementar esta política, la tabla PRODUCTOS está dividida horizontalmente en tres partes y se expande para incluir una columna POBLACION que dice dónde está localizado el inventario. Las filas de la tabla que describe el inventario de cada centro de distribución están almacenadas localmente y gestionadas por el DBMS de ese centro. Sin embargo, para propósitos de procesamiento de pedidos, la tabla PRODUCTOS es una gran tabla, que puede ser consultada y actualizada como una unidad. Las tablas horizontalmente divididas requieren cambios importantes en el DBMS y en su lógica de optimización.

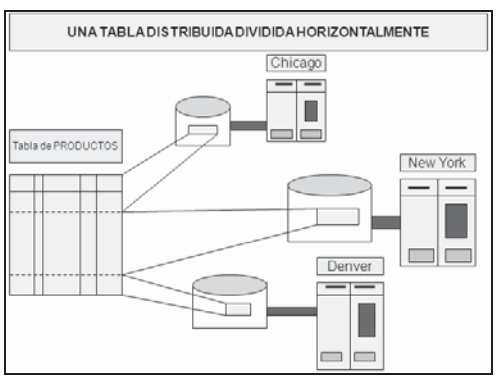


Figura 9.5

9.3.2 Divisiones verticales de tabla

Otro modo de distribuir una tabla a través de una red es dividir la tabla verticalmente, colocando diferentes columnas de la tabla en diferentes sistemas. La Figura 9.6 muestra un sencillo ejemplo de una división vertical de tabla. La tabla REPVENTAS ha sido ampliada para incluir nuevas columnas de información del personal (número de teléfono, estado civil, etc.) y ahora la utiliza tanto el departamento de procesamiento de pedidos como el departamento de personal, cada uno de los cuales tiene su propio sistema informático y su propio DBMS. La mayor parte de la actividad de cada departamento se centra en una o dos columnas de la tabla, pero hay muchas consultas e informes que utilizan las columnas relativas al personal y a los pedidos.

Para almacenar los datos de los vendedores, la tabla REPVENTAS está dividida verticalmente en dos partes. Las columnas de la tabla que almacenan datos del personal (NOMBRE, EDAD, CONTRATO, TELEFONO, CASADO) están almacenadas en el sistema de personal y gestionadas por su DBMS. Las otras columnas (NUM_EMPL, CUOTA, VENTAS, OFICINA_REP) están almacenadas en el sistema de procesamiento de pedidos y gestionadas por su DBMS. Sin embargo, la tabla REPVENTAS es tratada como una tabla grande, que puede ser consultada y actualizada como una unidad.

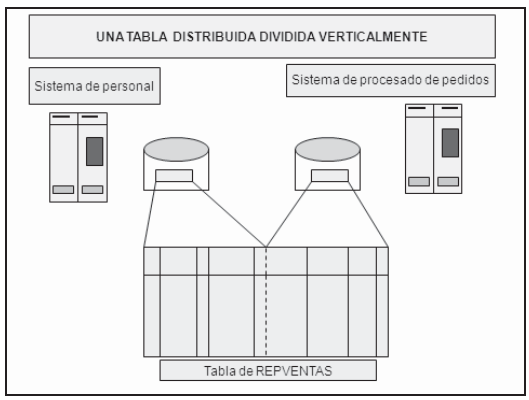


Figura 9.6

Con una división vertical de tabla, el DBMS no tiene que preocuparse de los contenidos de las filas para determinar dónde están localizados los datos. Sin embargo, tiene el problema igualmente difícil de mantener una relación uno a uno entre las filas parciales localizadas en dos o más sistemas diferentes. Además, las tablas verticalmente divididas pueden requerir cambios importantes en la lógica de cierre del DBMS. Si un programa necesita cerrar una fila, el DBMS debe ahora cerrar partes de la fila en más de un sistema. ¿Debería el DBMS cerrar todas las partes de la fila, o solamente las que están siendo realmente actualizadas? Dependiendo de la decisión, el DBMS puede necesitar implementar cierre a nivel de elemento.

9.3.3 Tablas reflejadas

En el caso de las tablas reflejadas, distintas copias duplicadas de una tabla se almacenan en varios sistemas (Figura 9.7). Cuando un usuario ejecuta una consulta que referencia a la tabla, puede utilizarse la copia local de la tabla, reduciendo el tráfico de red y elevando la productividad de la base de datos. Sin embargo, las actualizaciones de la tabla reflejada deben estar sincronizadas de modo que todas las copias de la tabla se actualicen al mismo tiempo, para prevenir que los usuarios de diferentes sistemas utilicen versiones inconsistentes de la tabla. Por esta razón, las tablas reflejadas son prácticas únicamente para tablas que tienen acceso preferente de lectura y que son actualizadas con poca frecuencia.

El uso de tablas reflejadas se justifica por el hecho de que habitualmente algunas de las tablas de una base de datos distribuida son fuertemente utilizadas por muchos usuarios en sistemas diferentes.

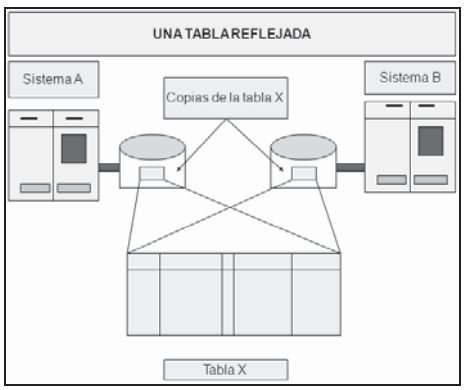


Figura 9.7

9.4 REFLEJAR UNA BASE DE DATOS EN SQL SERVER

Microsoft SQL Server 2008 amplía las posibilidades de duplicación de logs (*log shipping*) proporcionando a los administradores de las bases de datos la opción de reflejo (*mirroring*). Los administradores pueden usar esta funcionalidad para garantizar la disponibilidad de sus sistemas SQL mediante la configuración de un servidor en espera para su activación automática en caso de fallo (*failover*). El reflejo se implementa solo para bases de datos que utilizan el modelo de recuperación completa.

El reflejo involucra dos copias de una base de datos simple que generalmente residen en ordenadores diferentes. En todo momento solo una de las copias, denominada base de datos principal, está accesible para los clientes y las actualizaciones realizadas por los clientes

en la base de datos principal son reflejadas inmediatamente en la base de datos reflejada o copia espejo. La base de datos principal y la base de datos reflejada deben residir en instancias diferentes que se comunican y cooperan en una sesión de reflejo pudiendo realizar papeles complementarios de modo que puedan intercambiar sus papeles mediante un proceso denominado *switching*. La Figura 9.8 ilustra el modelo de reflejo, que puede ser ampliado mediante la presencia de una tercera instancia de control denominada testigo o *witness* (Figura 9.9).

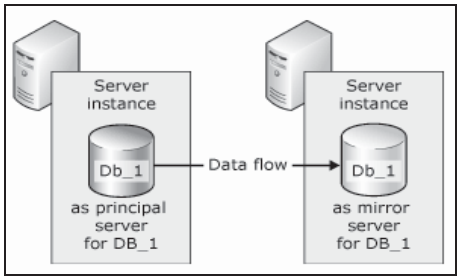


Figura 9.8

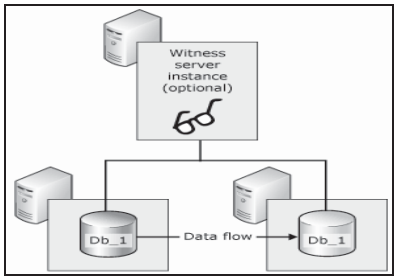


Figura 9.9

Para *realizar el reflejo de una base de datos*, en el árbol de la consola del Administrador corporativo, se expande *Bases de datos,* se hace clic con el botón derecho del ratón en la base de datos que se quiere reflejar, se selecciona la opción *Tareas* y, a continuación, *Reflejar* en el menú emergente resultante (Figura 9.10). Se obtiene la pantalla de propiedades de base de datos relativa a *Creación de reflejo* (Figura 9.11) en cuyos campos es necesario introducir las direcciones de red del servidor principal, del reflejado y del testigo y pudiendo fijar el modo de funcionamiento como sincrónico (se confirman siempre los cambios en el principal y en el reflejado) o asincrónico (se confirman los cambios en el principal y se transfieren al reflejado). Con el botón *Configurar seguridad* se configura la seguridad de todas las instancias que intervienen en el reflejo a través de un asistente (Figuras 9.12 a 9.16). Al pulsar *Aceptar* se crea el reflejo.

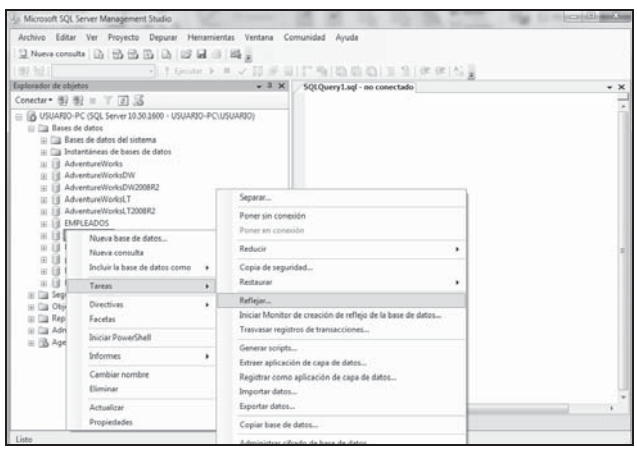


Figura 9.10

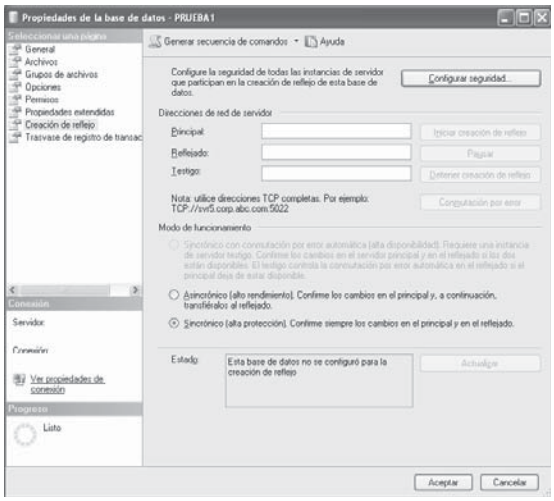


Figura 9.11

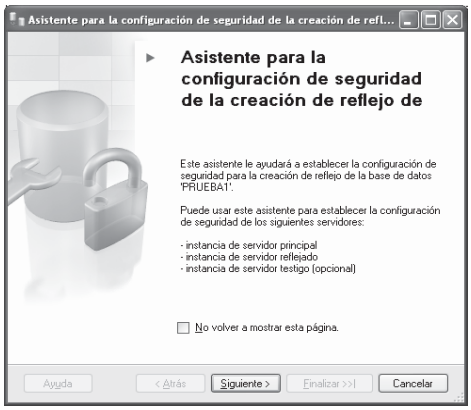


Figura 9.12

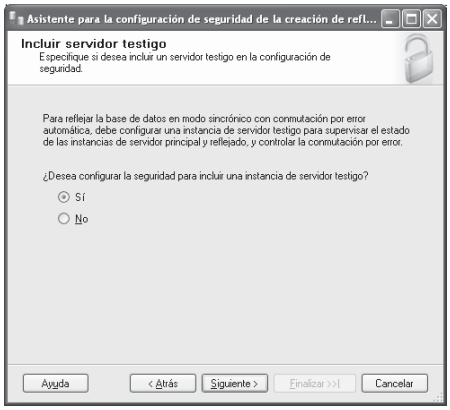


Figura 9.13

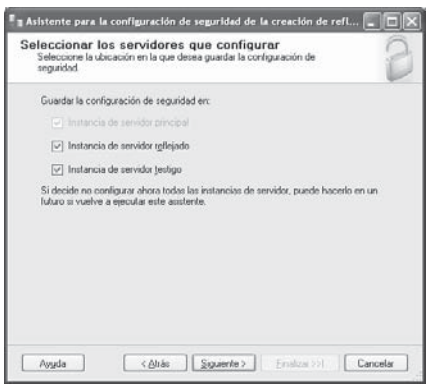


Figura 9.14

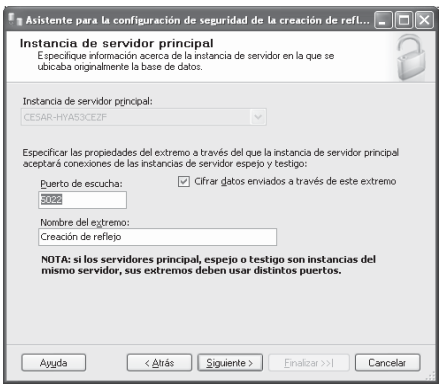


Figura 9.15

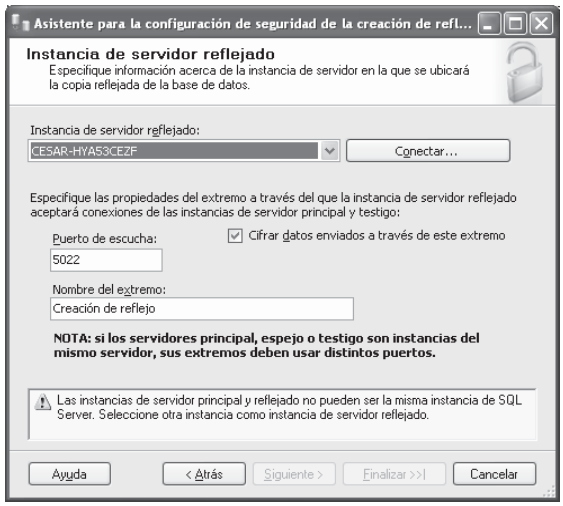

Asistente para la configuración de seguridad de la creación de refl...
Instancia de servidor reflejado
Especifique información acerca de la instancia de servidor en la que se ubicará la copia reflejada de la base de datos.
Instancia de servidor reflejado:
CESAR-HYA53CEZF
Conectar...
Especifique las propiedades del extremo a través del que la instancia de servidor reflejado aceptará conexiones de las instancias de servidor principal y testigo:
Puerto de escucha:
5022
Cifrar datos enviados a través de este extremo
Nombre del extremo:
Creación de reflejo
NOTA: si los servidores principal, espejo o testigo son instancias del mismo servidor, sus extremos deben usar distintos puertos.
Las instancias de servidor principal y reflejado no pueden ser la misma instancia de SQL Server. Seleccione otra instancia como instancia de servidor reflejado.
Ayuda
< Atrás
Siguiente >
Finalizar >>|
Cancelar

Figura 9.16

ÍNDICE ALFABÉTICO

P

R

T

V

MÓDULO FORMATIVO 0224_3

ADMINISTRACIÓN DE SISTEMAS GESTORES DE BASES DE DATOS

La presente obra está dirigida a los estudiantes de los nuevos Certificados de Profesionalidad de la familia profesional **Informática y Comunicaciones**, en concreto al Módulo Formativo **Administración de Sistemas Gestores de Bases de Datos**.

El libro comienza tratando los tipos de almacenamiento de la información para, a continuación, profundizar sobre los conceptos relativos a los sistemas gestores de base de datos, su estructura funcional, así como su instalación y administración. Todas las tareas se ilustran con ejemplos basados en Access siempre que sea posible y, en su defecto, en SQL Server y Oracle.

Posteriormente se aborda la problemática de construcción de guiones a través de procedimientos almacenados, funciones y desencadenadores. Como lenguajes procedimentales específicos para estas tareas se utilizan los lenguajes PL/SQL de Oracle y Transact_SQL de SQL Server. Finalmente se tratan los sistemas gestores de bases de datos distribuidos.

FAMILIA PROFESIONAL: Informática y Comunicaciones

CUALIFICACIÓN PROFESIONAL EN LA QUE SE INCLUYE:

- Administración de Bases de Datos

9 788492 650781

STARBOOK