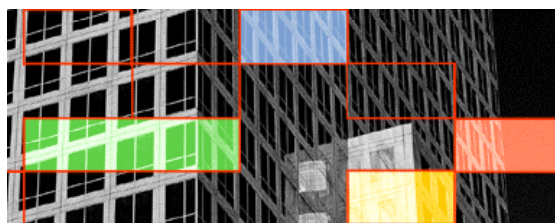




Universidad Nacional del Nordeste  
Facultad de Ciencias Exactas, Naturales y Agrimensura

Trabajo de Adscripción

SQL Server 2000



Microsoft  
**SQL Server 2000**  
Analysis Services

Erica Isabel Enriquez - L.U.: 29063

Prof. Director: Mgter. David Luis la Red Martínez

Licenciatura en Sistemas de Información  
Corrientes - Argentina

2007



A mis padres



# Índice General

<b>1</b>	<b>SQL Server 2000</b>	<b>1</b>
1.1	Introducción . . . . .	1
1.2	Bases de Datos de SQL Server . . . . .	5
1.3	Objetos de una Base de Datos . . . . .	8
1.4	Creación de Base de Datos . . . . .	11
1.5	Páginas y Extensiones . . . . .	13
1.5.1	Archivos y Grupos de Archivos Físicos de la Base de Datos	15
1.5.2	Archivos de Registro (LOG de Transacciones) . . . . .	17
1.5.3	Creación de Base de Datos . . . . .	17
1.6	Creación de Tablas . . . . .	20
1.6.1	Tipos de Datos de SQL Server 2000 . . . . .	20
1.6.2	Utilizar Datos Binarios . . . . .	21
1.6.3	Tipos de Datos Definidos por el Usuario . . . . .	22
1.6.4	Empleo de Comandos DDLL (Data Definition Language)	22
1.6.5	Implementar Restricciones . . . . .	27
1.6.6	Recuperar Información . . . . .	31
1.7	Requisitos de SQL Server 2005 Enterprise System . . . . .	35
	<b>Bibliografía</b>	<b>37</b>



# Índice de Figuras

1.1	Esquema cliente / servidor. . . . .	4
1.2	Componentes de SQL Server 2000. . . . .	6
1.3	Objetos de una base de datos. . . . .	9
1.4	Creación de tablas. . . . .	11
1.5	Almacenamiento de datos. . . . .	12
1.6	Página de datos. . . . .	14
1.7	Tipos de extensiones. . . . .	15
1.8	Funcionamiento del Log de transacciones. . . . .	18
1.9	Categorías y tipos de datos - 1. . . . .	23
1.10	Categorías y tipos de datos - 2. . . . .	24
1.11	Eliminación de Tablas. . . . .	28
1.12	Quitar Objetos. . . . .	29
1.13	Requisitos de SQL Server. . . . .	36



# Índice de Tablas

1.1	Tipos de páginas en SQL Server 2000. . . . .	13
1.2	Tipos de Constraint. . . . .	31



# Capítulo 1

## SQL Server 2000

### 1.1 Introducción

*SQL Server 2000* es un sistema de gestión de bases de datos relacionales (*SGDBR* o *RDBMS*: Relational Database Management System) diseñado para trabajar con grandes cantidades de información y con la capacidad de cumplir con los requerimientos de proceso de información para aplicaciones comerciales y sitios *Web*. [2] [1] [3]

Ofrece el soporte de información para las tradicionales aplicaciones *Cliente/Servidor*, las cuales están conformadas por una interfaz a través de la cual los clientes acceden a los datos por medio de una *LAN*.

La hoy emergente plataforma *.NET* exige un gran porcentaje de distribución de recursos, desconexión a los servidores de datos y un entorno descentralizado, para ello sus clientes deben ser *livianos*, tales como los navegadores de Internet, los cuales accederán a los datos por medio de servicios como el *Internet Information Services (IIS)*.

*SQL Server 2000* está diseñado para trabajar con dos tipos de bases de datos:

- *OLTP (OnLine Transaction Processing)*: Son bases de datos caracterizadas por mantener una gran cantidad de usuarios conectados concurrentemente realizando ingreso y/o modificación de datos. Por ejemplo: entrada de pedidos en línea, inventario, contabilidad o facturación.

- *OLAP (OnLine Analytical Processing)*: Son bases de datos que almacenan grandes cantidades de datos que sirven para la toma de decisiones, como por ejemplo las aplicaciones de análisis de ventas.

*SQL Server* puede ejecutarse sobre redes basadas en *Windows Server* así como sistema de base de datos de escritorio en máquinas *Windows NT Workstation*, *Windows Millenium* y *Windows 98*.

Los entornos *Cliente / Servidor* están implementados de tal forma que la información se guarde de forma centralizada en un computador central (*servidor*), siendo el servidor responsable del mantenimiento de la relación entre los datos, asegurarse del correcto almacenamiento de los datos, establecer restricciones que controlen la integridad de datos, etc.

Del lado cliente, este corre típicamente en distintas computadoras las cuales acceden al servidor a través de una aplicación, para realizar la solicitud de datos los clientes emplean el *Structured Query Language (SQL)*, este lenguaje tiene un conjunto de comandos que permiten especificar la información que se desea recuperar o modificar.

Existen muchas formas de organizar la información pero una de las formas más efectivas de hacerlo está representada por las bases de datos relacionales, las cuales están basadas en la aplicación de la teoría matemática de los conjuntos al problema de la organización de los datos.

En una base de datos relacional, los datos están organizados en *tablas* (llamadas *relaciones* en la teoría relacional).

Una *tabla* representa una clase de objeto que tiene importancia para una organización.

*Por ejemplo*, se puede tener una base de datos con una tabla para empleados, otra para clientes y otra para productos del almacén. Las tablas están compuestas de *columnas* y *filas* (*atributos* y *tuplas* en la teoría relacional).

La tabla *Empleados* tendría columnas para el nombre, el apellido, código del empleado, departamento, categoría laboral y cargo. Cada fila representa una instancia del objeto representado por la tabla. Por ejemplo, una fila de la tabla Empleados representa el empleado cuyo Id. de empleado es 12345.

Al organizar los datos en tablas, se pueden encontrar varias formas de definirlos. La teoría de las bases de datos relacionales define un proceso, la

*normalización*, que asegura que el conjunto de tablas definido organizará los datos de manera eficaz (ver fig. 1.1 de la pág. 4).

*SQL Server* incluye un conjunto de herramientas que facilitan la instalación y administración del servidor así como un conjunto de herramientas que facilitan el diseño e implementación de base de datos, entre ellos podemos mencionar:

- *SQL Server 2000 Database Engine*, diseñado para almacenar detalladamente los registros de las operaciones transaccionales (*OLTP*); este motor es responsable de mantener la seguridad de los datos, proveer un adecuado nivel de tolerancia a fallos, optimizar las consultas, emplear adecuadamente los bloqueos de recursos para optimizar la concurrencia, etc.
- *SQL Server 2000 Analysis Services*, provee herramientas para consultar información almacenada en *data warehouses* y *data marts*, como por ejemplo cuando se desea obtener información totalizada acerca de los niveles de ventas mensuales por regiones de ventas, etc.
- *Soporte para aplicaciones*, *SQL Server* brinda a las aplicaciones clientes la posibilidad de acceder a los datos a través de un lenguaje denominado *Transact-SQL*, asimismo es importante mencionar que ahora existe un soporte para devolver la información en formato *XML*.

Como *soporte para las aplicaciones clientes* se tiene:

1. *SQL Distributed Management Objects (SQL-DMO)*, *API* que brinda un conjunto de objetos *COM* que encapsulan toda la funcionalidad administrativa del motor de datos.
2. *Decision Support Objects (DSO)*, *API* que brinda un conjunto de objetos *COM* que encapsulan las funcionalidades de los *SQL Server 2000 Analysis Services*.
3. *Management Instrumentation (WMI)*, es una *API* orientada a objetos que permite administrar aplicaciones *scripts* para monitorear, configurar y controlar los servicios, recursos y aplicaciones de *Windows*. *SQL Server* ofrece una *API* que devuelve la información del motor de datos y de todas sus instancias, esta *API* se denomina *SQL Server 2000 WMI*.

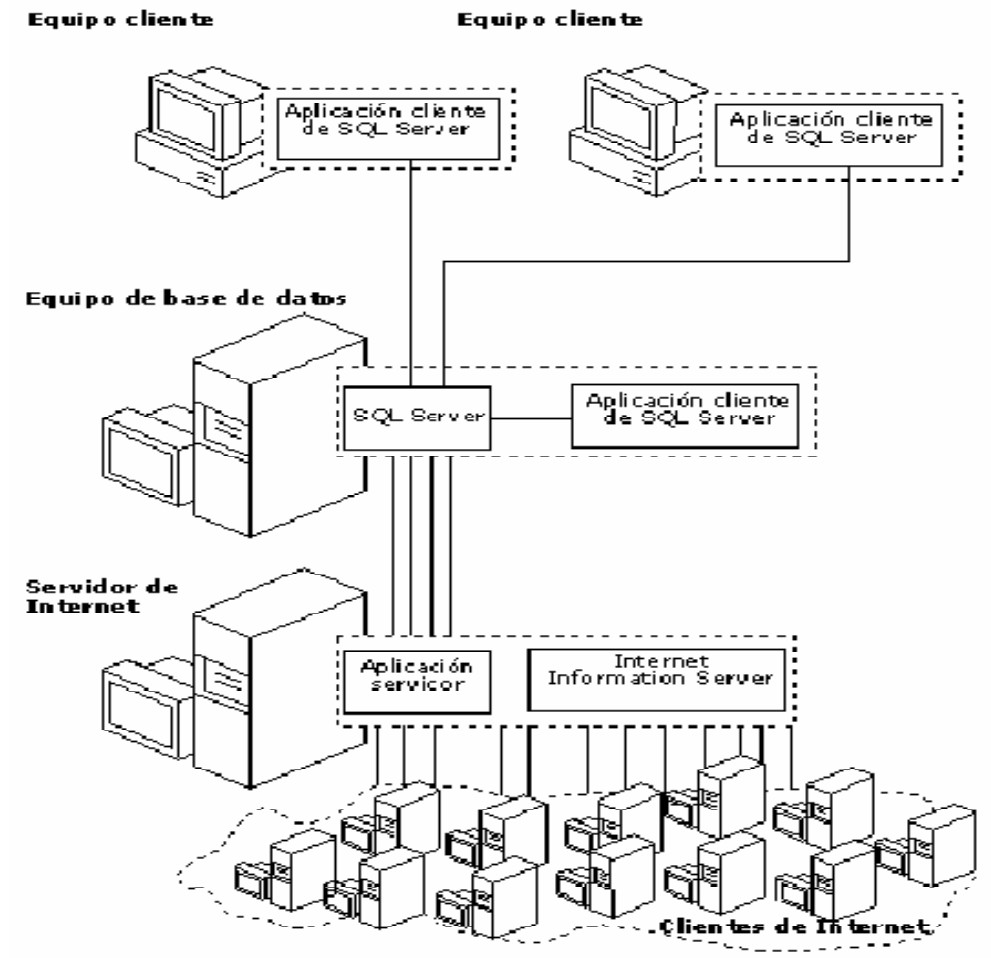


Figura 1.1: Esquema cliente / servidor.

Entre los *componentes adicionales* de *SQL Server 2000*, se puede mencionar:

- *Data Transformation Services*, permite recuperar información de un origen de datos, realizar transformaciones sencillas o complejas (como totalización de datos) y almacenarlos en otro origen de datos, como una base de datos *SQL* o un cubo multidimensional.
- *Replicación*, se puede distribuir la información a través de un mecanismo de replicación con la finalidad de optimizar el rendimiento o de mantener autonomía, mientras una copia de la información almacenada en diferentes computadoras mantengan sincronización.
- *English Query*, provee de un sistema que permite a los usuarios plantear una pregunta en lenguaje natural en lugar de emplear un formato *Transact-SQL*. Por ejemplo: “List all customers”, “How many blue dress were sold in 2001?”, etc.
- *Meta Data Services*, son un conjunto de servicios que permiten almacenar información acerca de las bases de datos y aplicaciones clientes que las emplean; esta información es aprovechada cuando se requiere intercambiar con otras aplicaciones. Los *Meta Data Services* proveen tres estándares: *Meta Data Coalition Open Information Model (MDC OIM)*, *Interfaces COM* y *XML Encoding*.
- Además de ello cuenta con la documentación apropiada para poder obtener información detallada de cada uno de los tópicos de *SQL Server* (ver fig. 1.2 de la pág. 6).

## 1.2 Bases de Datos de SQL Server

*SQL Server* soporta bases de datos del sistema y bases de datos del usuario.

Las bases de datos del sistema almacenan información que permite operar y administrar el sistema, mientras que las de usuario almacenan los datos requeridos por las operaciones del cliente.

Las *bases de datos del sistema* son:

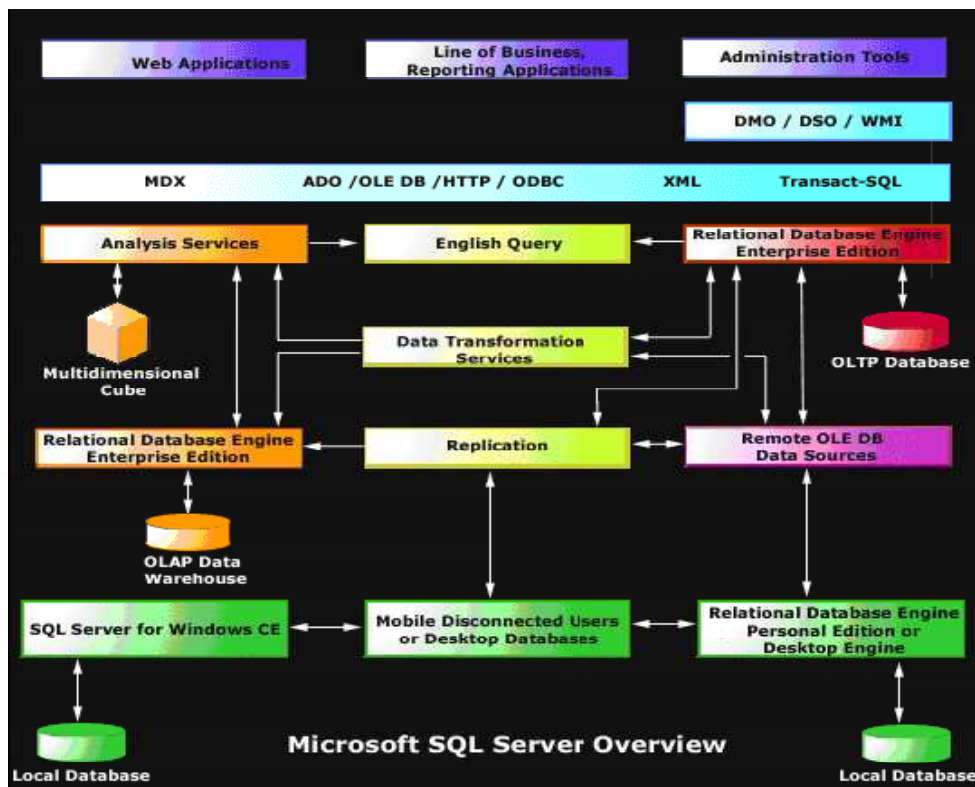


Figura 1.2: Componentes de SQL Server 2000.

- *master*

La base de datos *master* se compone de las tablas de sistema que realizan el seguimiento de la instalación del servidor y de todas las bases de datos que se creen posteriormente. Asimismo controla las asignaciones de archivos, los parámetros de configuración que afectan al sistema, las cuentas de inicio de sesión. Esta base de datos es crítica para el sistema, así que es bueno tener siempre una copia de seguridad actualizada.

- *tempdb*

Es una base de datos temporal, fundamentalmente un espacio de trabajo, es diferente a las demás bases de datos, puesto que se regenera cada vez que arranca *SQL Server*. Se emplea para las tablas temporales creadas explícitamente por los usuarios, para las tablas de trabajo intermedias de *SQL Server* durante el procesamiento y la ordenación de las consultas.

- *model*

Se utiliza como plantilla para todas las bases de datos creadas en un sistema.

Cuando se emite una instrucción *CREATE DATABASE*, la primera parte de la base de datos se crea copiando el contenido de la base de datos *model*, el resto de la nueva base de datos se llena con páginas vacías.

- *msdb*

Es empleada por el servicio *SQL Server Agent* para guardar información con respecto a tareas de automatización como por ejemplo copias de seguridad y tareas de duplicación, asimismo solución a problemas.

La información contenida en las tablas que contiene esta base de datos, es fácilmente accedida desde el Administrador Empresarial, así que se debe tener cuidado de modificar esta información directamente a menos que se conozca muy bien lo que se está haciendo.

- *distribution*

Almacena toda la información referente a la distribución de datos basada en un proceso de replicación.

### 1.3 Objetos de una Base de Datos

Las *Tablas* son *objetos* de la base de datos que contienen la información de los usuarios; estos datos están organizados en filas y columnas, de manera similar a la de una hoja de cálculo. Cada columna representa un dato aislado y en bruto que por sí solo no brinda información, por lo tanto estas columnas se deben agrupar y formar una fila para obtener conocimiento acerca del objeto tratado en la tabla.

*Por ejemplo*, se puede definir una tabla que contenga los datos de los productos ofertados por una tienda, cada producto estaría representado por una fila mientras que las columnas podrían identificar los detalles como el código del producto, la descripción, el precio, las unidades en stock, etc. (ver fig. 9 de la pág. 9).

Una *Vista* es un *objeto* definido por una consulta. Similar a tabla, la vista muestra un conjunto de columnas y filas de datos con un nombre, sin embargo, en la vista no existen datos, estos son obtenidos desde las tablas subyacentes a la consulta. De esta forma si la información cambia en las tablas, estos cambios también serán observados desde la vista. Fundamental emplean para mostrar la información relevante para el usuario y ocultar la complejidad de las consultas.

Los *Tipos de datos* especifican qué tipo de valores son permitidos en cada una de las columnas que conforman la estructura de la fila. Por ejemplo, si se desea almacenar precios de productos en una columna se debería especificar que el tipo de datos sea money, si se desea almacenar nombres se debe escoger un tipo de dato que permita almacenar información de tipo carácter.

*SQL Server* nos ofrece un conjunto de tipos de datos predefinidos, pero también existe la posibilidad de definir tipos de datos de usuario.

Un *Procedimiento almacenado* es una serie de instrucciones *SQL* precompiladas las cuales organizadas lógicamente permiten llevar a cabo una operación transaccional o de control. Un *Procedimiento almacenado* siempre se ejecuta en el lado del *Servidor* y no en la máquina *Cliente* desde la cual se hace el

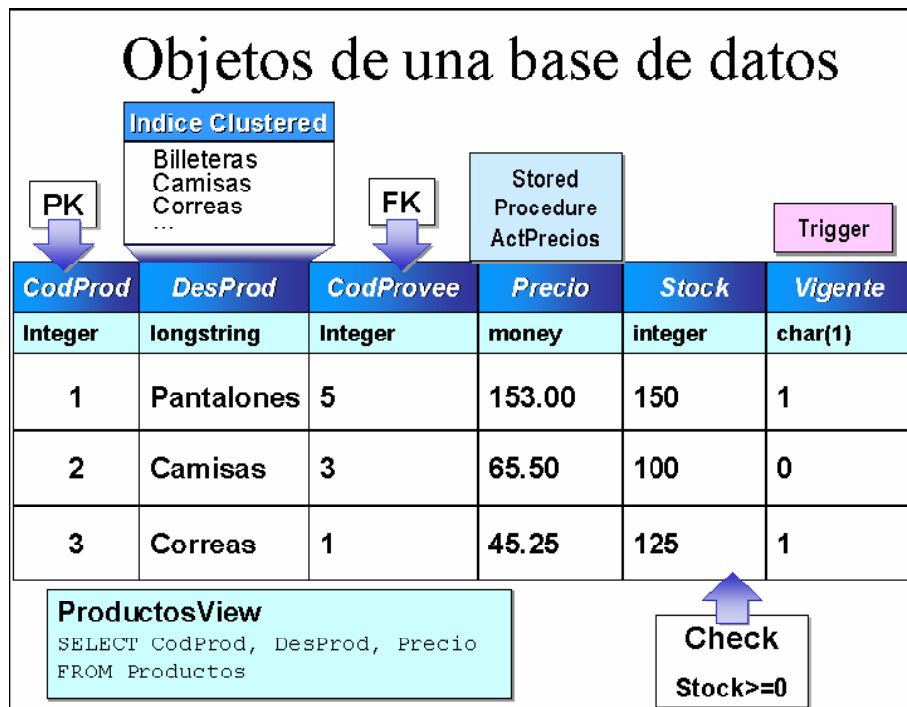


Figura 1.3: Objetos de una base de datos.

requerimiento. Para ejecutarlos deben ser invocados explícitamente por los usuarios.

Un *Desencadenador* es un *Procedimiento almacenado* especial, el cual se invoca automáticamente ante una operación de *Insert*, *Update* o *Delete* sobre una tabla. Un *Desencadenador* puede consultar otras tablas y puede incluir complejas instrucciones *SQL*; se emplean para mantener la integridad referencial, preservando las relaciones definidas entre las tablas cuando se ingresan o borran registros de aquellas tablas.

Los *Valores predeterminados* especifican el valor que *SQL Server* insertará en una columna cuando el usuario no ingresa un dato específico. Por ejemplo, si se desconoce el apellido materno de un empleado, *SQL Server* podría incluir automáticamente la cadena *NN* para identificar este campo.

Las *Reglas* son *objetos* que especifican los valores aceptables que pueden ser ingresados dentro de una columna particular. Las *Reglas* son asociadas a una columna o a un tipo de dato definido por el usuario. Una columna o un *Tipo de dato* puede tener solamente una *Regla* asociada con ellos.

Las *Restricciones* son restricciones que se asignan a las columnas de una tabla y son controladas automáticamente por *SQL Server*.

Esto nos provee las siguientes *ventajas*:

- Se pueden asociar múltiples *constraints* a una columna, y también se puede asociar un *constraints* a múltiples columnas.
- Se pueden crear las *Restricciones* al momento de crear la tabla (*create-table*).

Las *Restricciones* conforman el standards *ANSI* para la creación y alteración de tablas, estos no son extensiones del *Transact SQL*.

Se puede usar un *constraints* para forzar la integridad referencial, el cual es el proceso de mantener relaciones definidas entre tablas cuando se ingresan o eliminan registros en aquellas tablas.

Los *índices* de *SQL Server* son similares a los índices de un libro que nos permiten llegar rápidamente a las páginas deseadas sin necesidad de pasar hoja por hoja; de forma similar los índices de una tabla nos permitirán buscar

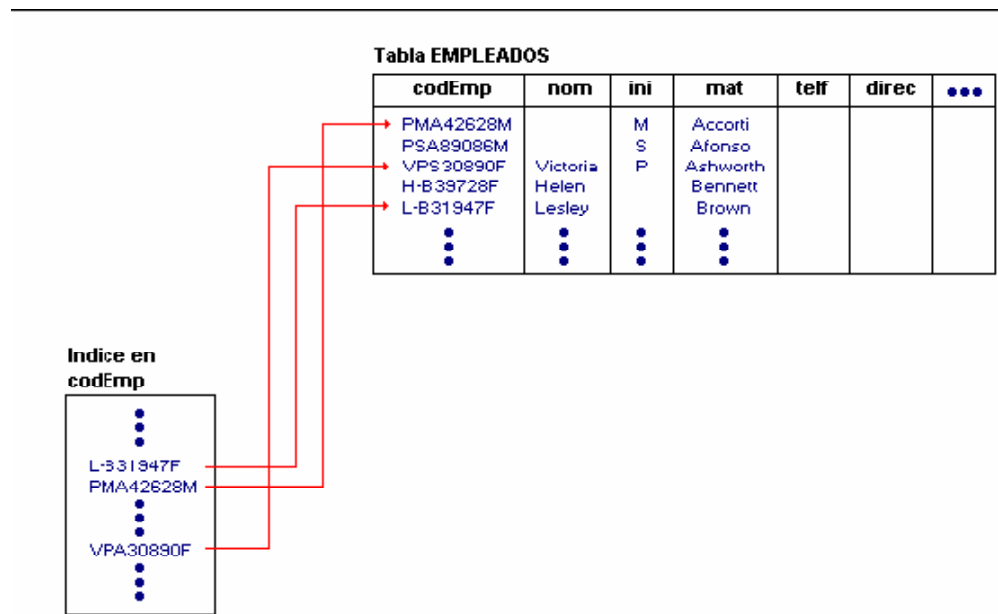


Figura 1.4: Creación de tablas.

información rápidamente sin necesidad de recorrer registro por registro toda la tabla.

Un *índice* contiene *valores* y *punteros* a las filas donde estos valores se encuentran (ver fig. 1.4 de la pág. 11).

## 1.4 Creación de Base de Datos

En términos sencillos una base de datos de *SQL Server* es una *colección de objetos* que contiene y administra *datos*.

Antes de crear una base de datos es importante entender cómo es que almacena la información (ver fig. 1.5 de la pág. 12).

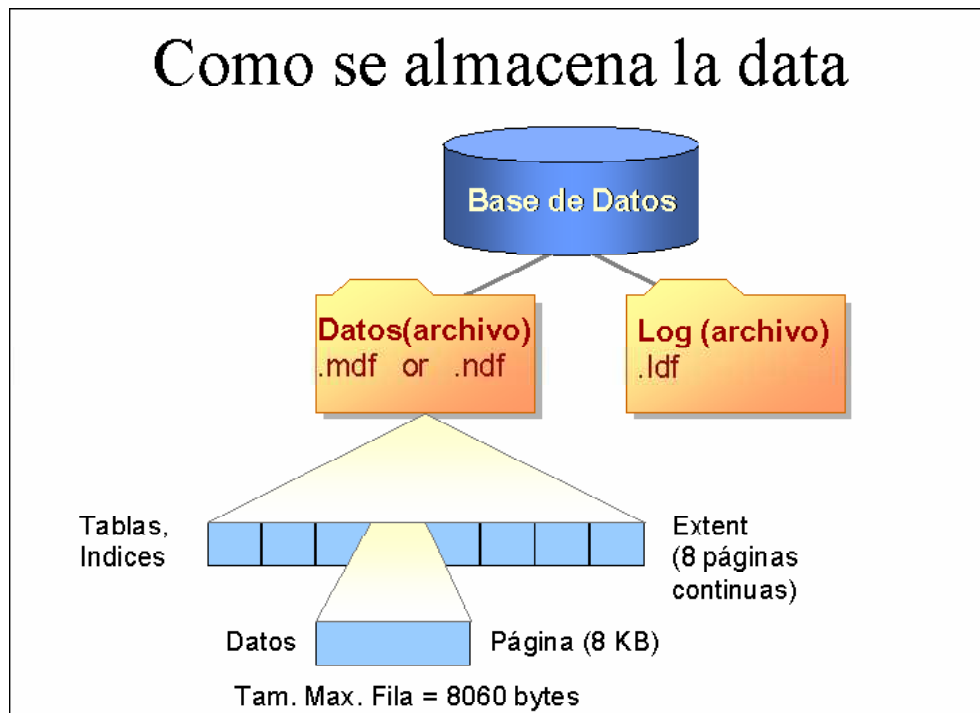


Figura 1.5: Almacenamiento de datos.

## 1.5 Páginas y Extensiones

Antes de crear una base de datos con *SQL Server 2000*, se debe tener en cuenta que la *unidad básica de almacenamiento es la página (data page)*, el tamaño de cada page es de 8 KB, lo cual representa un total de 128 páginas por cada megabyte.

El comienzo de cada página es una cabecera de 96 bytes que se utiliza para almacenar información de cabecera tal como el tipo de página, la cantidad de espacio libre de la página y el Id. del objeto propietario de la página.

Existen ocho *tipos de páginas* en los archivos de datos de una base de datos *SQL Server 2000* (ver tabla 1.1 de la pág. 13).

Tipo de página	Contenido
Datos	Filas con todos los datos excepto los de tipo <b>text</b> , <b>ntext</b> e <b>image</b> .
Índice	Entradas de índices.
Texto o imagen	Datos de tipo <b>text</b> , <b>ntext</b> e <b>image</b> .
Mapa de asignación global / Mapa de asignación global secundario	Información acerca de las extensiones asignadas.
Espacio libre en la página	Información acerca del espacio libre disponible en las páginas.
Mapa de asignación de índices	Información acerca de las extensiones utilizadas por una tabla o un índice.
Bulk Changed Map	Información de los extens modificados por operaciones Bulk desde el último backup del log.
Differential Changed Map	Información de los extens modificados desde el último full database backup.

Tabla 1.1: Tipos de páginas en SQL Server 2000.

Los *archivos de registro (LOG)* no contienen páginas, contienen series de *registros*.

Las páginas de datos contienen todos los datos de las filas de datos excepto

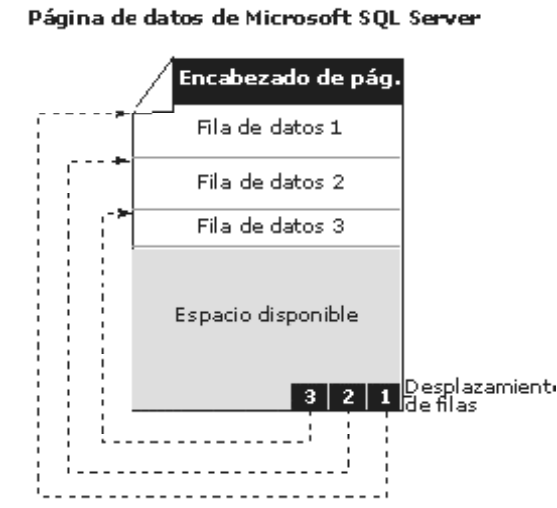


Figura 1.6: Página de datos.

los datos *text*, *ntext* e *image*, que están almacenados en páginas separadas. Las filas de datos se colocan en las páginas una a continuación de otra, empezando inmediatamente después de la cabecera, al final de cada página se encuentra una tabla de posiciones de filas que contiene una entrada por cada fila de la página y cada entrada registra la posición, desde el principio de la página, del primer byte de la fila.

Las entradas de la tabla de posiciones de filas están en orden inverso a la secuencia de las filas de la página (ver fig. 1.6 de la pág. 14).

En *SQL Server* las filas no pueden continuar en otras páginas.

Las *extensiones* son la *unidad básica de asignación de espacio* a las *tablas* e *índices*.

Constan de 8 páginas contiguas, es decir 64 KB. Lo cual representa 16 *extensiones* por MB.

Para hacer que la asignación de espacio sea eficiente, *SQL Server 2000* no asigna *extensiones* enteras a tablas con poca cantidad de datos.

Se tiene dos *tipos de extensiones*:

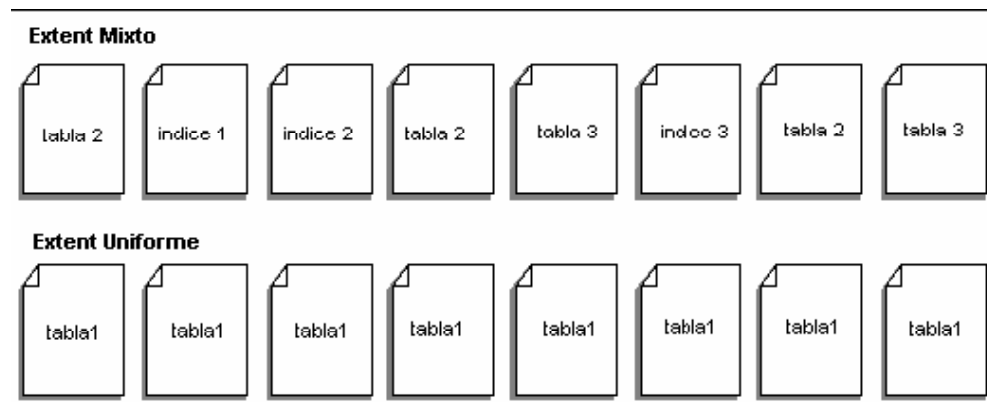


Figura 1.7: Tipos de extensiones.

- Las *extensiones uniformes* son propiedad de un único objeto; sólo el objeto propietario puede utilizar las ocho páginas de la extensión.
- *Extensiones mixtas*, pueden estar compartidas por hasta ocho objetos (ver fig. 1.7 de la pág. 15).

Las tablas o índices nuevos son asignados a páginas de extensiones mixtas. Cuando la tabla o el índice crecen hasta el punto de ocupar ocho páginas, se pasan a extensiones uniformes.

### 1.5.1 Archivos y Grupos de Archivos Físicos de la Base de Datos

Un *archivo* de base de datos no es más que un archivo del sistema operativo.

Una base de datos se distribuye en por lo menos dos archivos, aunque es muy probable que sean varios los archivos de base de datos que se especifican al crear o al modificar una base de datos.

Principalmente *SQL Server* divide su trabajo en un *archivo para datos* y otro para el *registro de las transacciones (log)*.

*SQL Server 2000* permite los tres siguientes *tipos de archivos*:

- *Archivos de datos primarios*

Toda base de datos tiene un archivo de datos *primario* que realiza el seguimiento de todos los demás archivos, además de almacenar datos. Por convenio este archivo tiene la extensión *MDF*.

- *Archivos de datos secundarios*

Una base de datos puede tener cero o varios archivos de datos *secundarios*. Por convenio la extensión recomendada para los archivos de datos secundarios es *NDF*.

- *Archivos de registro (LOG)*

Todas las bases de datos por lo menos tendrán un archivo de *registro* que contiene la información necesaria para recuperar todas las transacciones que suceden sobre la misma. Por convenio la extensión de este archivo es *LDF*.

Por lo tanto, al crear una base de datos, debemos considerar las siguientes *premisas y reglas para el almacenamiento de los datos*:

- Todas las Bases de Datos tienen un archivo de base de datos primario (.mdf) y uno para el Log de Transacciones (.ldf). Además puede tener archivos de datos secundarios (.ndf).
- Cuando se crea una Base de Datos, una copia de la Base de Datos Model, la cual incluye tablas del sistema, es copiada en la nueva Base de Datos.
- La Data (los datos) es almacenada en bloques de 8-kilobytes (KB) de espacio de disco contiguo llamado páginas.
- Las filas o registros no pueden atravesar páginas. Esto es, que la máxima cantidad de datos en una fila de datos simple es de 8060 bytes.
- Las tablas y los índices son almacenados en *Extents*. Un *Extents* consta de ocho páginas contiguas, o sea 64 KB.
- El Log de Transacciones lleva toda la información necesaria para la recuperación de la Base de Datos en una eventual caída del sistema. Por defecto, el tamaño del Log de Transacciones es del 25% del tamaño de los archivos de datos. Se usa esta configuración como punto de partida y se ajusta de acuerdo a las necesidades de la aplicación.

### 1.5.2 Archivos de Registro (LOG de Transacciones)

El *LOG* de transacciones archiva todas las modificaciones de los datos tal cual son ejecutados. El proceso es como sigue:

1. Una modificación de datos es enviada por la aplicación cliente.
2. Cuando una modificación es ejecutada, las páginas afectadas son leídas del disco a memoria (*Buffer Cache*), provista de las páginas que no están todavía en la *DataCache* del *query* previo.
3. Cada comando de modificación de datos es archivado en el *LOG*. El cambio siempre es archivado en el *LOG* y es escrito en el disco antes que el cambio sea hecho en la Base de Datos. Este tipo de *LOG* es llamado *LOG* de tipo *write-ahead*.
4. Una vez que las páginas de datos residen en el *Buffer Cache*, y las páginas de *LOG* son archivadas sobre el disco en el archivo del *LOG*, el proceso de *CHECKPOINT*, escribe todas las transacciones completas a la Base de Datos en el disco.
5. Una modificación de datos es enviada por la aplicación cliente. Si el sistema falla, automáticamente el proceso de recuperación usa el *LOG* de Transacciones para llevar hacia delante todas las transacciones comprometidas (*COMMIT*) y llevar hacia atrás alguna transacción incompleta (*ROLLBACK*) (ver fig. 1.8 de la pág. 18).

Los *marcadores de transacción* en el *LOG* son usados durante la recuperación automática para determinar los puntos de inicio y el fin de una transacción.

Una *transacción* se considera completa cuando el marcador *BEGIN TRANSACTION* tiene un marcador asociado *COMMIT TRANSACTION*. Las páginas de datos son escritas al disco cuando ocurre el *CHECKPOINT*.

### 1.5.3 Creación de Base de Datos

Se puede crear una base de datos de distintas maneras, utilizando el *Wizard*, desde el *Administrador Empresarial* o a través del *Query Analyzer*.

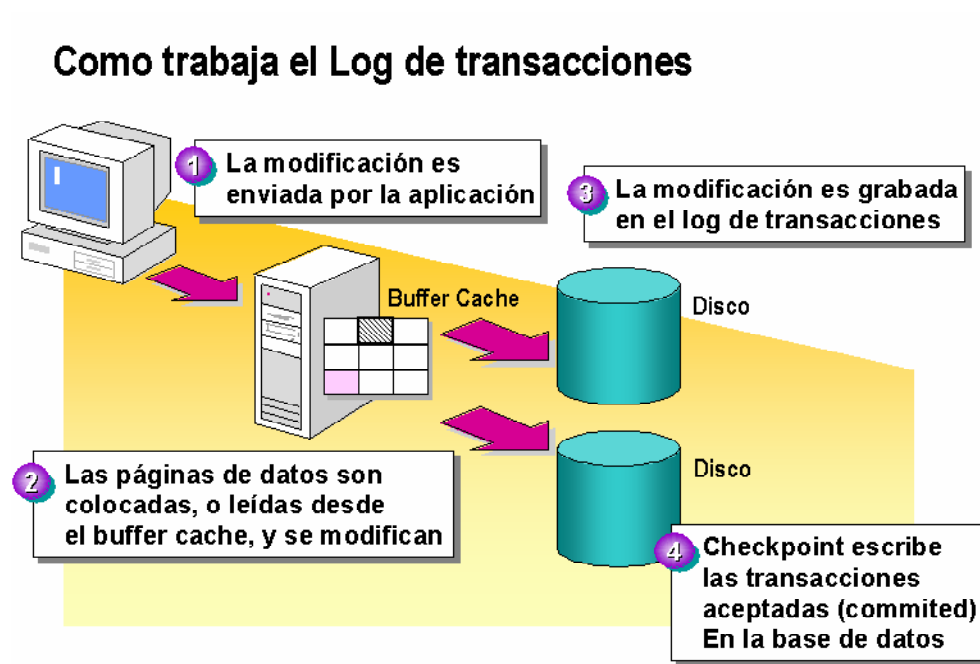


Figura 1.8: Funcionamiento del Log de transacciones.

- *Observaciones:*

Se emplea *CREATE DATABASE* para crear una base de datos y los archivos que almacena ésta. *SQL Server* implementa *CREATE DATABASE* en dos pasos:

*SQL Server* utiliza una copia de *model* para inicializar la base de datos y sus metadatos.

*SQL Server* rellena el resto de la base de datos con páginas vacías, excepto las páginas que tengan datos internos que registren cómo se emplea el espacio en la base de datos.

Cualquier objeto definido por el usuario en *model* se copiará a todas las bases de datos recién creadas.

Cada base de datos nueva hereda los valores opcionales de la base de datos *model* (a menos que se especifique *FOR ATTACH*).

En un servidor se puede especificar un máximo de 32.767 bases de datos.

Cuando se especifica una instrucción *CREATE DATABASE nombreBaseDatos* sin parámetros adicionales, la base de datos se crea con el mismo tamaño que *model*.

Cada base de datos tiene un propietario con capacidad para realizar actividades especiales. El propietario es el usuario que crea la base de datos, este propietario se puede cambiar mediante *sp\_changedbowner*.

### ¿Quiénes Pueden Crear Bases de Datos?

En forma predeterminada podrán hacerlo los usuarios que pertenecen al rol *sysadmin* y *dbcreator*. Los miembros de las funciones fijas de servidor *sysadmin* y *SecurityAdmin* pueden conceder permisos *CREATE DATABASE* a otros inicios de sesión.

Los miembros de las funciones fijas de servidor *sysadmin* y *dbcreator* pueden agregar otros inicios de sesión a la función *dbcreator*. El permiso *CREATE DATABASE* debe concederse explícitamente; no se concede mediante la instrucción *GRANT ALL*.

Estos permisos se limitan a unos cuantos inicios de sesión para mantener el control de la utilización de los discos del equipo que ejecuta *SQL Server*.

## Creando Múltiples Archivos

La ventaja de almacenar la base de datos en múltiples archivos radica en la flexibilidad de modificar en futuro la configuración del hardware sin que se vea afectada la base de datos, otro de los motivos es que si se emplea una base de datos de 15GB y por algún motivo ocurre una falla y desea recuperar desde el backup, necesitaría una unidad de 15 o más GB de almacenamiento, mientras que si se distribuyó la base de datos en múltiples archivos pequeños será más probable que tenga disponibles múltiples unidades de 4 GB que unidades de 15GB.

Con las sentencias del *Transact-SQL* es posible modificar la lista de archivos que conforman la base de datos, agregar o quitar archivos, incluso se puede definir nuevos grupos de archivos, los cuales permitirán tratar múltiples archivos como si se tratará de uno solo.

Para poder realizar esta tarea se emplea la sentencia *ALTER DATABASE*.

## Renombrando Base de Datos

Para quitar una base de datos, se utiliza *DROP DATABASE*. Para cambiar el nombre de una base de datos, se utiliza *sp\_renamedb*, pero se debe recordar que para esto la base de datos a renombrar tiene que tener activa la opción *'single user'*.

## 1.6 Creación de Tablas

### 1.6.1 Tipos de Datos de SQL Server 2000

*SQL Server* brinda una serie de tipos de datos para almacenar la información; la correcta selección del tipo de dato es simplemente una cuestión de determinar qué valores desea almacenar, como por ejemplo carácter, enteros, binario, fechas, etc.

Los siguientes objetos tienen tipos de datos:

- Columnas de tablas y vistas.

- Parámetros de procedimientos almacenados.
- Variables.
- Funciones de *Transact-SQL* que devuelven uno o más valores de datos de un tipo de datos específico.
- Procedimientos almacenados que devuelven un código, que siempre es de tipo integer.

Al asignar un tipo de datos a un objeto se definen cuatro atributos del objeto:

- La clase de datos que contiene el objeto, por ejemplo, carácter, entero o binario.
- La longitud del valor almacenado o su tamaño.
- La precisión del número (sólo tipos de datos numéricos). La precisión es el número de dígitos que puede contener el número. Por ejemplo, un objeto *smallint* puede contener hasta 5 dígitos, con lo que tiene una precisión de 5.
- La escala del número (sólo tipos de datos numéricos). La escala es el máximo número de dígitos a la derecha del separador decimal. Por ejemplo, un objeto *int* no puede aceptar un separador decimal y tiene una escala de 0. Un objeto *money* puede tener hasta 4 dígitos a la derecha del separador decimal y tiene una escala de 4.

### 1.6.2 Utilizar Datos Binarios

Los tipos de datos *binary* y *varbinary* almacenan cadenas de bits. Mientras que los datos de carácter se interpretan según la página de códigos de *SQL Server*, los datos *binary* y *varbinary* son, simplemente, un flujo de bits y pueden tener una longitud de hasta 8.000 bytes.

Las constantes binarias tienen un 0x (un cero y una letra x en minúsculas) a la izquierda, seguido de la representación hexadecimal del patrón de bits. Por ejemplo, 0x2A especifica el valor hexadecimal 2A, que es equivalente al valor decimal 42 o un patrón de bits de un byte de 00101010.

Las tablas 1.9 de la pág. 23 y 1.10 de la pág. 24 describen los tipos de datos provistos por *SQL Server*.

### 1.6.3 Tipos de Datos Definidos por el Usuario

Los tipos de datos definidos por el usuario están basados en los tipos de datos disponibles a través de *SQL Server 2000*. Los tipos de datos definidos por el usuario se pueden emplear para asegurar que un dato tenga las mismas características sobre múltiples tablas.

### 1.6.4 Empleo de Comandos DDL (Data Definition Language)

*SQL Server 2000* emplea las tablas como objetos de almacenamiento de datos que los usuarios manipulan a través de sus aplicaciones o vía web.

Las tablas son objetos compuestos por una estructura (conjunto de columnas) que almacenan información interrelacionada (filas) acerca de algún objeto en general.

Las tablas se definen para los objetos críticos de una base de datos, por ejemplo CLIENTES y su estructura estaría conformada por cada uno de los atributos que se requieran de los clientes para poder obtener información de ellos, como por ejemplo: nombres, direcciones, teléfonos, celular, etc.

Cada uno de estos atributos tiene un tipo de dato definido y además la tabla debe permitir asegurar que cada código de producto es único en la misma, para asegurarse de no almacenar la información del mismo cliente dos veces.

Las tablas suelen estar relacionadas entre sí, para facilitar el hecho de consultas entre múltiples tablas.

Se pueden distinguir los siguientes tipos de tablas.

#### Tablas del Sistema

La información usada por *SQL Server* y sus componentes es almacenada en tablas especiales denominadas como tablas del sistema. Estas tablas no deben alterarse directamente por el usuario.

Categoría	Descripción	Tipo de Dato	Descripción
Binario	Almacenan cadenas de bits. La data consiste de números hexadecimales. Por ejemplo el decimal 245 es F5 en hexadecimal.	binary	La data debe tener una longitud fija (hasta 8 KB).
		varbinary	Los datos pueden variar en el número de dígitos hexadecimales (hasta 8 KB).
		image	La data puede tener una longitud variable y exceder los 8Kb.
Caracter	Consisten de una combinación de letras, símbolos y números. Por ejemplo las combinaciones "John928" y "(0*&(%B09nh jkJ".	char	Los datos deben tener una longitud fija (Hasta 8 KB).
		varchar	La data puede variar en el número de caracteres (Hasta 8 KB.)
		text	Los datos pueden ser caracteres ASCII que excedan los 8 KB.
Fecha y Hora	Consisten en combinaciones válidas de estos datos. No puede separar en tipos distintos el almacenamiento de sólo fechas o sólo horas.	Datetime	Fechas en el rango 01 Ene 1753 hasta el 31 Dic 9999 (Se requiere 8 bytes por valor).
		smalldatetime	Fechas en el rango 01 Ene 1900 hasta 06 Jun 2079 (Se requiere requires 4 bytes por valor).
Decimal	Consisten en información que almacena información significativa después del punto decimal.	decimal	Los datos pueden tener hasta 38 dígitos, todos los cuales podrían estar a la derecha del punto decimal. Este tipo de dato guarda un valor exacto del número y no una aproximación.
		numeric	Para SQL Server, el tipo de dato numeric es equivalente al tipo de datos decimal.
Punto Flotante	Números aproximados (Punto flotante).	float	Datos en el rango de 1.79E + 308 hasta 1.79E + 308.
		real	Datos en el rango de 3.40E + 38 hasta 3.40E + 38.
Enteros	Consiste en información numérica positiva o negativa como por ejemplo -5, 0 y 25.	bigint	Datos en el rango de $2^{63}$ (-9223372036854775808) hasta $2^{63}-1$ (9223372036854775807). Se requieren de 8 bytes para almacenar estos valores.
		int	Datos en el rango de -2,147,483,648 hasta 2,147,483,647. Se requieren de 4 bytes para almacenar estos valores.

Figura 1.9: Categorías y tipos de datos - 1.

Categoría	Descripción	Tipo de Dato	Descripción
		smallint	Datos en el rango de -32,768 hasta 32,767. Se requieren 2 bytes por cada valor de este tipo.
		tinyint	Datos entre 0 y 255, se requiere de 1 byte.
Monetario	Cantidades monetarias positivas o negativas.	money	Datos monetarios entre -922,337,203,685,477.5808 y +922,337,203,685,477.5807 (Se requieren 8 bytes por valor).
		smallmoney	Datos monetarios entre -214,748,3648 y 214,748,3647 (Se requieren de 4 bytes por valor).
Especiales	Consisten en información que no recae en ninguna de las categorías anteriormente mencionadas.	bit	Datos que consisten de 1 o 0. Emplear este tipo de dato para representar TRUE o FALSE ó YES o NO.
		cursor	Este tipo de dato es empleado por variables o procedimientos almacenados que emplean parámetros OUTPUT referenciados a un cursor.
		timestamp	Este tipo de dato es empleado para indicar la actividad que ocurre sobre una fila. La secuencia de este número se incrementa en formato binario.
		uniqueidentifier	Consiste en un número hexadecimal que especifica un globally unique identifier (GUID), es útil cuando se desea asegurar la unicidad de una fila entre muchas otras.
		SQL_variant	Almacena varios tipos de datos, a excepción de text, ntext, timestamp, image y sql_variant.
		table	Almacena un resultado de una consulta para su posterior procesamiento. Se puede emplear para definir variables locales de tipo table o para retornar los valores devueltos por una función del usuario.
Unicode	Al emplear este tipo de datos se puede almacenar	nchar	Datos con longitud fija, hasta 4000 caracteres Unicode.

Figura 1.10: Categorías y tipos de datos - 2.

Si se desea obtener información almacenada en las tablas del sistema se debe usar:

- Información de la vista esquema (schema view).
- Procedimientos almacenados de sistema.
- Instrucciones *Transact-SQL* y funciones.
- *SQL-DMO*.
- Catálogo de funciones *API*.

Las tablas del sistema almacenan información, llamada *Metadata*, acerca del sistema y de los objetos de las bases de datos. Todas las tablas del sistema comienzan con el prefijo *SYS*.

### Tablas del Usuario

**Permanentes** Son las tablas donde se almacena la información que los usuarios utilizan para sus operaciones. Esta información existirá hasta que se elimine explícitamente.

**Temporales** Estas son tablas similares a las permanentes que se graban en *tempdb*, y son eliminadas automáticamente cuando ya no son usadas.

Hay dos tipos de tablas temporales, *locales* y *globales*, difieren una de la otra en sus nombres, su visibilidad y su ámbito de vida:

- *Tablas Temporales Locales*. El primer carácter del nombre de #, su visibilidad es solamente para la conexión actual del usuario y son eliminadas cuando el usuario se desconecta.
- *Tablas Temporales Globales*. Su nombre comienza con ##, su visibilidad es para cualquier usuario, y son eliminadas luego que todos los usuarios que las referencian se desconectan del *SQL Server*.

### Creación de Tablas

Cuando se crea una tabla se le debe asignar un nombre a la misma, un nombre a cada columna además de un tipo de datos y de ser necesaria, una longitud.

Adicional a las características antes mencionadas, *SQL Server 2000* brinda la posibilidad de implementar columnas calculadas, definiéndolas como fórmulas.

Los nombres de las columnas deben ser únicos en la tabla.

**Consideraciones al Crear Tablas** Se debe considerar lo siguiente:

- Billones de tablas por base de datos.
- 1024 columnas por tabla.
- 8060 como tamaño máximo de registro (sin considerar datos *image*, *text* y *ntext*).
- Al momento de definir una columna se puede especificar si la columna soporta o no valores NULL.

Para crear tablas se debe utilizar la sentencia *CREATE TABLE*.

**Valores Autogenerados Para las Columnas** En *SQL Server 2000* se puede definir columnas que obtengan valores generados por el sistema, para ello se puede hacer uso de las siguientes especificaciones.

**Propiedad Identity** Permite generar valores secuenciales del sistema, este tipo de valores pueden ser utilizados en columnas que serán empleadas como *primary key*.

Para emplear esta propiedad se debe especificar un valor de inicio y uno de incremento. Este tipo de columnas no son editables.

### Eliminación de tablas

Para eliminar una tabla, se debe hacer click derecho sobre la tabla y se selecciona la opción *Eliminar* (ver fig. 1.11 de la pág. 28).

Aparecerá la siguiente caja de diálogo (ver fig. 1.12 de la pág. 29).

Pulsar click sobre *Quitar Todos* y con ello la tabla será retirada de la base de datos.

Otra forma es utilizando la sentencia *DROP TABLE*.

### 1.6.5 Implementar Restricciones

Uno de los principales objetivos de una base de datos relacional es cuidar y controlar la integridad de los datos, la cual podría perderse ante operaciones que modifican la información como: *INSERT*, *UPDATE* y *DELETE*.

Por ejemplo se puede perder la integridad de datos ante alguna de las siguientes situaciones:

- Se puede registrar un pedido de un producto no existente.
- Se podría modificar los datos existentes son valores incorrectos.
- Los cambios a la base de datos se podrían aplicar parcialmente, por ejemplo si se registra un pedido sin actualizar el stock del producto requerido.

*SQL Server* provee de múltiples medios para controlar la integridad de datos, como por ejemplo:

- *Datos requeridos*, es una de las restricciones más sencillas que especifican qué columnas permiten valores nulos y qué columnas no. Al momento de definir las columnas de una tabla se podrá asignar a cada columna la especificación *NULL* o *NOT NULL*.
- *Control de validez*, permite controlar los valores que se le asignarán a una columna. Por ejemplo, en una columna que guarde promedios de estudiantes se podría controlar que el rango de valores se encuentre entre 0 y 10.

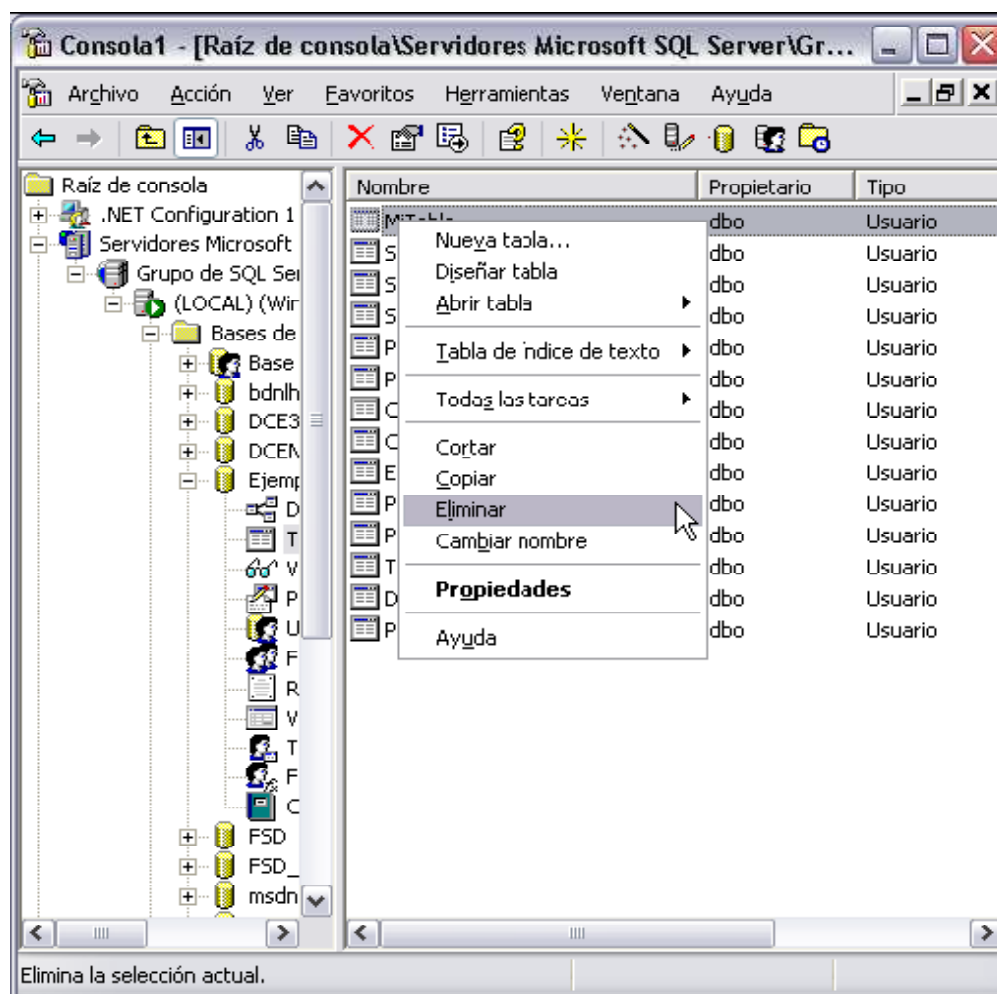


Figura 1.11: Eliminación de Tablas.

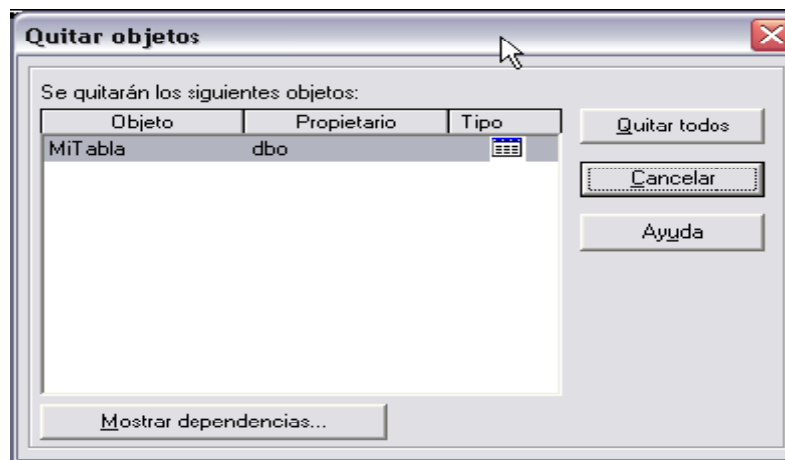


Figura 1.12: Quitar Objetos.

- *Integridad de entidad*, referido a que una clave principal asegura la unicidad de cada registro.
- *Integridad referencial*, asegura las relaciones entre las claves primarias y claves foráneas, por ejemplo al agregar un pedido, controlar que el código de producto que se especifica en el pedido exista en la tabla de productos.
- *Reglas comerciales*, las modificaciones en la base de datos deben cumplir con ciertos requisitos para poder mantener la información íntegra, por ejemplo en el caso anteriormente mencionado, el producto podría existir pero el stock encontrarse en 0, de tal manera que no debería registrarse el pedido.

*SQL Server* para poder forzar la integridad de datos propone dos modalidades:

- *Integridad Declarativa*

Se debe definir el criterio de consistencia como criterio de la definición del objeto.

Para utilizar integridad declarativa se puede emplear *constraints*, *defaults* y *rules*.

- *Integridad por Procedimientos*

Se pueden escribir procedimientos almacenados y desencadenadores (*Triggers*) para poder forzar la integridad de datos. Aunque las restricciones muy complejas podrían implementarse a través de lenguajes de programación y otras herramientas clientes.

Se revisará la integridad declarativa definiéndola a partir de los *CONSTRAINTS*.

Los *CONSTRAINTS* son un método estándar de forzar la integridad de datos, aseguran que los datos ingresados a las columnas sean válidos y que las relaciones entre las tablas se mantengan.

Los *CONSTRAINTS* se puede definir al momento de crear la tabla, aunque también es posible hacerlo después de que las tablas se han creado.

Los *CONSTRAINTS* se ejecutan antes que la información se registre en el log.

Tipo de Integridad	Tipo de Constraints
Controlar rango de valores sobre las columnas	Default Check
Unicidad de registros y valores unicos. Referencial	Primary Key Unique Foreign Key Check

Tabla 1.2: Tipos de Constraint.

## Diagrama de Base de Datos

Una vez que se ha terminado de implementar las restricciones especificadas anteriormente y luego que se tiene las restricciones funcionando se podría dar un vistazo al diagrama de la base de datos:

Los diagramas representan gráficamente la estructura de la *Base de Datos*, se puede ver sus tablas y diseño, además de las relaciones entre ellas. También se convierte en una herramienta gráfica para crear, modificar e implementar integridad y constancia de datos.

### 1.6.6 Recuperar Información

Uno de los principales motivos por el cual se guarda información, es por que posteriormente se va a consultar, una de las principales razones por las cuales las bases de datos relacionales lograron gran aceptación es, por la forma tan sencilla de lograr acceder a los datos.

Y como parte de estas facilidades para poder realizar consultas, se encuentra a la sentencia *SELECT*.

## Select

Recupera información de la Base de Datos y permite la selección de una o más filas o columnas de una o muchas tablas. La sintaxis completa de la instrucción *SELECT* es compleja.

## Insert

Se utiliza la sentencia *INSERT* para agregar registros a una tabla.

Si el valor que se intenta agregar a una de las columnas no cumple con alguno de los *constraints* establecidos la operación abortará inmediatamente.

## Update

Esta sentencia nos permite modificar la información de las tablas.

Si la actualización de una fila no cumple con una restricción o regla, infringe la configuración de valores *NULL* o si el nuevo valor es de un tipo de datos incompatible, se cancela la instrucción, se devuelve un error y no se actualiza ningún registro.

## Delete

Las instrucciones *DELETE* y *TRUNCATE TABLE* remueven filas de una tabla.

Se usa la instrucción *DELETE* para eliminar una o más filas de una tabla.

Se debe tener en cuenta las siguientes consideraciones:

- El *SQL Server* borra todas las filas de una tabla a menos que se use la cláusula

*WHERE*.

- Cada fila borrada genera historia en el *Log de Transacciones*.

Para eliminar registros se puede utilizar también la sentencia *TRUNCATE TABLE*, que resulta más rápida que *DELETE* puesto que no genera entradas en el log de transacciones.

### Recuperar Información de Dos o Más Tablas (Joins)

Para muchas de las consultas que los usuarios realizan sobre la data almacenada en nuestra base de datos es necesario extraer información de más de una tabla, para ello es necesario emplear los *JOINS* que representan una operación producir un conjunto de resultados que incorporen filas y columnas de las tablas referidas en la consulta, esto se hace basándose en columnas comunes a las tablas.

Cuando se ejecutan los *JOIN*, *SQL Server* compara los valores de las columnas especificadas fila por fila, entonces usa los resultados de la comparación para combinar los valores que califican como nuevas filas.

Cuando se implementa los *JOIN*, se debe tener en cuenta las siguientes consideraciones:

- Especificar las condiciones del *JOIN* en base a *Primary Key* y a *Foreign Key*.
- Si una tabla tiene un *Primary Key* compuesta, se debe referenciar a la clave entera en la cláusula *ON* del *JOIN* de tablas.
- Las columnas comunes a las tablas deben ser del mismo tipo de dato.
- Si dos o más columnas de las diferentes tablas que participan en el *JOIN*, tienen el mismo nombre, se deberá calificar dichas columnas usando el formato NombreTabla. Nombre. Columna.
- Limitar en lo posible el número de tablas en un *JOIN*, a más tablas, el *SQL Server* se tomará más tiempo para resolver la consulta.
- La cláusula *LEFT OUTER JOIN* permite observar todos los registros de la tabla que se referencia a la izquierda en una consulta, completa las filas con *NULL* en caso que no exista un valor almacenado en la tabla de la derecha.
- La cláusula *RIGHT OUTER JOIN* permite observar todos los registros de la tabla que se referencia a la derecha en una consulta, completa las

filas con *NULL* en caso que no exista un valor almacenado en la tabla de la izquierda.

- La cláusula *FULL OUTER JOIN* muestra la combinación de todos los registros de la tabla de la izquierda con los registros de la tabla de la derecha.

### Desencadenadores

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 unidas 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.

### Asignar Roles y/o Permisos - Comandos DCL (Data Control Language)

Los comandos *Data Control Language* permiten asignar o negar derechos a los usuarios sobre los distintos objetos de la base de datos. Para esto se deben haber definido los usuarios que tendrán acceso a la base de datos.

Los comandos *DCL* se pueden asignar por sentencias o por objetos.

Creados los usuarios se puede asignar, revocar o negar permisos sobre cada uno de los objetos de la base de datos.

Para ello se emplearan las siguientes instrucciones:

- **GRANT:** Permite asignar permisos sobre los objetos de una base de datos y/o sentencias a utilizar.
- **REVOKE:** Remueve la asignación o negación del recurso de base de datos.
- **DENY:** Permite negar permisos sobre los objetos de una base de datos y/o sentencias a utilizar.

## 1.7 Requisitos de SQL Server 2005 Enterprise System

Los requisitos de *SQL Server Enterprise Edition* en 32-bit y x64 plataformas, así como los sistemas basados en *Itanium* son los siguientes (ver fig.1.13 de la pág. 36).<sup>1</sup>

---

<sup>1</sup>*SQL Server* trabaja en hardware certificado para el uso con el sistema operativo de *Microsoft Windows*.

Requisitos Enterprise Edition System			
	32-bit	x64	Itanium
<b>Procesador</b>	600-megahertz (MHz) Pentium III-compatible o más rápido; 1 gigahertz (GHz) o el procesador más rápido recomendado.	1-GHz AMD Opteron, AMD Athlon 64, Intel Xeon con Intel EM64T support, Intel Pentium IV con EM64T support processor or faster	1-GHz Itanium o el procesador más rápido.
<b>Sistema Operativo</b>	Microsoft Windows 2000 Server con Service Pack (SP) 4 or later; Windows Server 2003 Standard Edition, Enterprise Edition, o Datacenter Edition con SP 1 or later; Windows Small Business Server 2003 con SP 1 or later	Microsoft Windows Server 2003 Standard x64 Edition, Enterprise x64 Edition, o Datacenter x64 Edition con SP 1 or later	Microsoft Windows Server 2003 Enterprise Edition o DataCenter Edition para los sistemas Itanium-based con SP 1 or later
<b>Memoria</b>	512 megabytes (MB) de RAM o más; 1 gigabyte (GB) o más recomendado.	512 MB de RAM o más; 1 GB o más recomendado.	512 MB de RAM o más; 1 GB o más recomendado
<b>Disco Duro</b>	<ul style="list-style-type: none"> <li>• Aproximadamente 350 MB de espacio del disco duro disponible para la instalación recomendada</li> <li>• Aproximadamente 425 MB de espacio del disco duro para SQL Server Books Online, SQL Server Mobile Books Online, y nuestra base de datos</li> </ul>	<ul style="list-style-type: none"> <li>• Aproximadamente 350 MB de espacio del disco duro disponible para la instalación recomendada</li> <li>• Aproximadamente 425 MB de espacio del disco duro disponible adicional para SQL Server Books Online, SQL Server Mobile Books Online, y nuestra base de datos</li> </ul>	<ul style="list-style-type: none"> <li>• Aproximadamente 350 MB de espacio del disco duro disponible para la instalación recomendada</li> <li>• Aproximadamente 425 MB de espacio del disco duro adicional para SQL Server Books Online, SQL Server Mobile Books Online, y nuestra base de datos</li> </ul>
<b>Drive</b>	CD-ROM o DVD-ROM drive	CD-ROM o DVD-ROM drive	CD-ROM o DVD-ROM drive
<b>Pantalla</b>	Super VGA (1,024x768) o superior-resolución del adaptador de video y monitor	Super VGA (1,024x768) o superior-resolución del adaptador de video y monitor	Super VGA (1,024x768) o superior-resolución del adaptador de video y monitor
<b>Otros Dispositivos</b>	Microsoft Mouse o el dispositivo señalado compatible	Microsoft Mouse o el dispositivo señalado compatible	Microsoft Mouse o el dispositivo señalado compatible
<b>Otros Requisitos</b>	<ul style="list-style-type: none"> <li>• Microsoft Internet Explorer 6.0 SP1 or later</li> <li>• Para informar los servicios, usted necesita Microsoft Internet Information Services (IIS) 5.0 or later, y ASP.NET 2.0 or later</li> </ul>	<ul style="list-style-type: none"> <li>• Microsoft Internet Explorer 6.0 SP1 or later</li> <li>• Para informar los servicios, usted necesita Microsoft Internet Information Services (IIS) 5.0 or later, and ASP.NET 2.0 or later</li> </ul>	<ul style="list-style-type: none"> <li>• Microsoft Internet Explorer 6.0 SP1 or later</li> <li>• Para informar los servicios, usted necesita Microsoft Internet Information Services (IIS) 5.0 or later, and ASP.NET 2.0 or later</li> </ul>

Figura 1.13: Requisitos de SQL Server.

# Bibliografía

- [1] Jason Gerend Charlie Russel, Sharon Crawford. *Microsoft Server 2003 Administrators Companion*. Microsoft Press, USA, 2003.
- [2] Mark Rouse Don Jones. *Microsoft Server 2003 Delta Guide*. Que Publishing, USA, 2003.
- [3] Microsoft. *SQL Server Architecture*. Microsoft Press, USA, 2005.



# Índice de Materias

- .NET, 1
- índices, 10
- ANSI, 10
- API, 25
- archivos físicos
  - de la base de datos, 15
- base de datos
  - creación, 11
  - objetos, 8
- cliente / servidor, 2
- COM, 5
- CONSTRAINTS, 30
- constraints, 10
- CREATE DATABASE, 19
- Data Control Language, 34
- data marts, 3
- data page, 13
- data warehouses, 3
- datos
  - tipos de, 8
- datos binarios
  - utilización, 21
- DDL, 22
- DELETE, 27
- Delete, 32
- desencadenador, 10, 34
- DSO, 3
- extensiones, 13, 14
- IIS, 1
- INSERT, 27
- Insert, 32
- Joins, 33
- joins, 33
- LAN, 1
- LDF, 16
- LOG, 13
  - de transacciones, 17
- MDC OIM, 5
- MDF, 16
- NDF, 16
- OLAP, 2
- OLTP, 1, 3
- páginas, 13
  - tipos de, 13
- procedimiento
  - almacenado, 8
- RDBMS, 1
- reglas, 10
- restricciones, 10
  - implementar, 27
- Select, 31
- SQL, 2
  - Server 2000
    - Analysis Services, 3

- Database Engine, 3
  - WMI, 3
- Server Agent, 7
- Server, 11
- SQL Server, 1, 5
- SQL-DMO, 3, 25
- tablas, 8
  - creación, 20, 26
  - del sistema, 22
  - del usuario, 25
  - temporales
    - globales, 25
    - locales, 25
- Transact-SQL, 3, 20
- Truncate Table, 32
- Udpate, 32
- UPDATE, 27
- valores
  - predeterminados, 10
- vista, 8
- Web, 1
- Windows
  - 98, 2
  - Millenium, 2
  - NT Workstation, 2
  - Server, 2
- WMI, 3
- XML, 3, 5