

Una vez establecido los valores y luego de pulsar Aceptar, en el Administrador Empresarial se observara la nueva base de datos creada.

Desde el Analizador de Consultas

Otra de las formas de crear una base de datos es a través del Analizador de Consultas, donde emplearemos la sentencia CREATE DATABASE, cuya sintaxis reducida es la siguiente:

```
CREATE DATABASE NombreBaseDatos
[ ON [PRIMARY
  NAME = nombreArchivoLógico,
  FILENAME = 'nombreArchivoSO',
  SIZE = tamaño,
  MAXSIZE = { tamañoMáximo | UNLIMITED },
  FILEGROWTH = incrementoCrecimiento) [...n]
]
[ LOG ON
  NAME = nombreArchivoLógico,
  FILENAME = 'nombreArchivoSO',
  SIZE = tamaño,
  MAXSIZE = { tamañoMáximo | UNLIMITED },
  FILEGROWTH = incrementoCrecimiento) [...n]
```

## **[COLLATE nombre\_collation] [ FOR LOAD | FOR ATTACH ]**

### **Argumentos**

#### **nombreBaseDatos**

Es el nombre de la nueva base de datos, deben ser únicos en un servidor y pueden tener hasta 128 caracteres, a menos que no se especifique ningún nombre lógico para el registro. Si no se especifica ningún nombre lógico de archivo de registro, SQL Server genera un nombre lógico al anexar un sufijo a nombreBaseDatos.

#### **ON**

Especifica que los archivos de disco utilizados para almacenar la parte de datos (archivos de datos) se han definido explícitamente. La palabra clave va seguida de una lista delimitada por comas de elementos que definen los archivos de datos del grupo de archivos principal.

#### **PRIMARY**

Especifica que la lista de archivos está asociada al grupo principal. Este grupo contiene todas las tablas del sistema de base de datos. También contiene todos los objetos no asignados a los grupos de archivos de usuario. El primer archivo especificado pasa a ser el archivo principal, el cual contiene el inicio lógico de la base de datos y de las tablas del sistema. Una base de datos sólo puede tener un archivo principal. Si no se especifica PRIMARY, el primer archivo enumerado en la instrucción CREATE DATABASE se convierte en el archivo principal.

#### **LOG ON**

Especifica que los archivos de registro de la base de datos (archivos de registro) se han definido explícitamente. La palabra clave va seguida de una lista delimitada por comas la cual define las características de los archivos de registro. Si no se especifica LOG ON, se crea automáticamente un único archivo de registro con un nombre generado por el sistema y un tamaño que es el 25% de la suma de los tamaños de todos los archivos de datos de la base de datos.

#### **FOR LOAD**

Cláusula que se mantiene por compatibilidad con versiones anteriores de SQL Server. La base de datos se crea con la opción de base de datos **dbo use only** activada y el estado se establece en "cargando". En realidad esto no es necesario en SQL Server 7.0 porque la instrucción RESTORE puede volver a crear la base de datos como parte de la operación de restauración.

#### **FOR ATTACH**

Crea la base de datos desde un conjunto existente de archivos del sistema operativo. Debe existir una entrada de archivos que determine cual es el archivo principal, las otras entradas son necesarias si existen archivos creados en una ruta de acceso distinta de cuando se creó la base de datos por primera vez o se adjuntó por última vez.

Utilice el procedimiento almacenado del sistema **sp\_attach\_db** en lugar de emplear **CREATE DATABASE FOR ATTACH** directamente, esto deberá emplearlo si debe especificar más de 16 archivos.

**COLLATE**

Especifica el conjunto de caracteres que se empleará para almacenar información en la base de datos, se puede emplear un conjunto de caracteres especificado por Windows o por SQL Server. De no especificarse se empleará el conjunto de caracteres seleccionado en el momento de la instalación

**NAME**

Especifica el nombre lógico del archivo.

No se requiere este parámetro cuando se especifica FOR ATTACH.

Este nombre es el utilizado para referenciar al archivo en las sentencias del Transact-SQL que se ejecuten después.

**FILENAME**

Especifica el nombre de archivo del sistema (archivo físico).

Se debe especificar la ruta de acceso y nombre de archivo que el sistema operativo utiliza cuando crea la base de datos. La ruta de acceso debe especificar un directorio en el servidor sobre el que se instaló SQL Server.

No se puede especificar un directorio en un sistema comprimido de archivos.

**SIZE**

Especifica el tamaño para el archivo. De no hacerlo SQL Server utiliza el tamaño del archivo principal de la base de datos model.

Cuando este parámetro no es especificado para un archivo secundario o de registro SQL Server automáticamente le asigna 1 MB.

El valor mínimo a asignar es de 512 KB. Si no se especifica tamaño, el valor predeterminado es 1 MB. El tamaño especificado para el archivo principal debe tener al menos el tamaño del archivo principal de la base de datos model.

**MAXSIZE**

Especifica el tamaño máximo de crecimiento del archivo. Se pueden utilizar los sufijos KB y MB, el valor predeterminado es MB. Especifique un número entero; no incluya decimales. Si no se especifica, el archivo aumenta hasta que el disco esté lleno.

**UNLIMITED**

Especifica que el archivo aumenta de tamaño hasta que el disco esté lleno.

**FILEGROWTH**

Especifica el incremento de crecimiento del archivo, este valor no puede exceder el valor MAXSIZE. Emplee un número entero. Un valor 0 indica que no hay crecimiento.

El valor se puede especificar en MB, KB o %, el valor predeterminado es MB. Cuando se especifica %, el tamaño de incremento de crecimiento es el porcentaje especificado del tamaño del archivo en el momento en que tiene lugar el incremento. De no emplear FILEGROWTH, el valor predeterminado es 10% y el valor mínimo es 64 KB. El tamaño especificado se redondea al múltiplo de 64 KB más cercano.

**Observaciones**

Emplee CREATE DATABASE para crear una base de datos y los archivos que almacenan 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 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**.

Para mostrar un informe de una base de datos o de todas las bases de datos de un servidor con SQL Server, ejecute **sp\_helpdb**. Para obtener un informe acerca del espacio utilizado en una base de datos, emplee **sp\_spaceused**. Para obtener un informe de los grupos de archivos de una base de datos, utilice **sp\_helpfilegroup**, y utilice **sp\_helpfile** para obtener el informe de los archivos de la base de datos.

¿Quiénes pueden crear bases de datos?

En forma predeterminada podrán hacerlos 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.

Ejemplos de creación de base de datos empleando el Analizador de Consultas

Primero ingrese al Analizador de Consultas para ello primero debe especificar el tipo de autenticación a realizar del sistema o estándar, vea la siguiente figura:



### Ejemplo 1

Crear la base de datos Prueba1 con los parámetros En forma predeterminada.

**Use Master**

**GO**

**Create Database Prueba1**

**GO**

Verifique la creación de la base de datos y note que automáticamente SQL Server asignó tamaños y nombres lógicos para los archivos. Para ello emplee el siguiente procedimiento almacenado del sistema:

**Sp\_HelpDB Prueba1**

**GO**

Debe obtener el siguiente resultado:

name	db_size	owner	dbid	created	status	Compatibility Level
Prueba1	1.12 MB	sa	8	Feb 28 2002	Status=ONLINE,	

**Updateability=READ\_WRITE,**  
**UsuarioAccess=MULTI\_USUARIO,**  
**Recovery=FULL,**  
**Version=539,**  
**Collation=SQL\_Latin1\_General\_CP1\_CI\_AS,**  
**SQLSortOrder=52,**  
**IsTornPageDetectionEnabled,**  
**IsAutoCreateStatistics,**  
 IsAutoUpdateStatistics 8.0

Además se mostrará un informe con los archivos que se crearon automáticamente:

Columnas	Archivo de datos	Archivo de Log
name	prueba1	prueba1_log
fileid	1	2
	C:\Program Files\Microsoft SQL Server\MSSQL\data\Prueba1.	C:\Program Files\Microsoft SQL Server\MSSQL\data\Prueba1_Log.
filename	mdf	ldf
filegroup	PRIMARY	NULL
size	640Kb	504Kb
maxsize	Unlimited	Unlimited
growth	10%	10%
usage	data only	log only

### **Ejemplo 2**

Crear la base de datos Prueba2 con un archivo de datos de 10Mb, un tamaño máximo de 20Mb y un crecimiento de 1Mb., el archivo de registro debe asumir los valores por default.

**Use Master**

**GO**

**Create Database Prueba2**

**On Primary**

**(NAME = 'Prueba2\_Data',**

**FILENAME = 'C:\Program Files\Microsoft SQL**

**Server\MSSQL\data\Prueba2\_Data.Mdf,**

**SIZE = 10Mb,**

**MAXSIZE = 20Mb,**

**FILEGROWTH= 1Mb)**

**GO**

Verifique la creación de la base de datos anterior:

**Sp\_HelpDB Prueba2**

**GO**

Puede notar como SQL Server aprovecha los valores predeterminados en la base de datos model para completar la información que corresponde al log de transacciones, la cual no se especificó en la sentencia CREATE DATABASE.

### **Ejemplo 3**

Crear la base de datos Prueba3 especificando un archivo de datos con un tamaño inicial de 15Mb, un tamaño máximo de 30Mb y un crecimiento de 5Mb., el archivo de registro debe tener un tamaño inicial de 5MB y uno máximo de 10MB, el crecimiento debe ser de 1MB.

**Use Master**

**GO**

**Create Database Prueba3**

**On Primary**

**(NAME = 'Prueba3\_Data',**

```

        FILENAME      = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\Prueba3_Data.Mdf'
        SIZE          = 15Mb,
        MAXSIZE       = 30Mb,
        FILEGROWTH    = 5Mb)
Log On
    (NAME            = 'Prueba3_Log',
     FILENAME        = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\Prueba3_Log.Ldf'
     SIZE            = 5Mb,
     MAXSIZE         = 10Mb,
     FILEGROWTH      = 1Mb)
GO
-- Verifique la información con :
Sp_HelpDB Prueba3
GO

```

Otra de las formas de comprobar la creación de las bases de datos es mostrando las filas de la tabla del sistema SysDatabases.

```

Use Master
GO
Select DbID, Name From SysDatabases
GO

```

Revise los resultados.

### 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 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 mas gigabytes de almacenamiento, mientras que si distribuyó la base de datos en múltiples archivos pequeños será mas 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 puede definir nuevos grupos de archivos los cuales permitirán tratar múltiples archivos como si se tratará de uno solo.

Para poder realizar esta tarea emplee la sentencia ALTER DATABASE

**Sintaxis**  
**ALTER DATABASE NombreBD**

```
{ ADD FILE <Especificación del archivo> [,...n] [TO FILEGROUP
nombreGrupoArchivos]
| ADD LOG FILE <<Especificación del archivo> [,...n]
| REMOVE FILE nombreArchivoLógico
| ADD FILEGROUP nombreGrupoArchivos
| REMOVE FILEGROUP nombreGrupoArchivos
| MODIFY FILE <<Especificación del archivo>
| MODIFY FILEGROUP nombreGrupoArchivos propiedadGrupoArchivos
| SET < optionspec > [ ,...n ] [ WITH < termination > ]
| COLLATE < collation_name > }
```

## Argumentos

### ADD FILE

Especifica que se está agregando un archivo.

### TO FILEGROUP

Especifica el grupo de archivos al que debe agregarse el archivo especificado.

### ADD LOG FILE

Especifica que se agregue un archivo de registro a la base de datos indicada.

### REMOVE FILE

Elimina el archivo de la base de datos y retira su referencia en las tablas del sistema además de eliminar el archivo físico. Este archivo no podrá eliminarse si no está vacío.

### ADD FILEGROUP

Especifica que se va a agregar un grupo de archivos.

### REMOVE FILEGROUP

Quita el grupo de archivos de la base de datos, no se puede realizar si el grupo de archivos no está vacío.

### MODIFY FILE

Especifica que el archivo dado se debe modificar, incluidas las opciones *FILENAME*, *SIZE*, *FILEGROWTH* y *MAXSIZE*. Sólo se puede cambiar una de estas propiedades a la vez.

### MODIFY FILEGROUP nombreGrupoArchivos propiedadGrupoArchivos

Especifica la propiedad que se aplicará a los archivos que pertenecen al grupo de archivos.



Los valores de **propiedadGrupoArchivos** son:

#### **READONLY**

Especifica que el grupo de archivos es de sólo lectura. De tal manera que no se podrán realizar modificaciones sobre los archivos pertenecientes a este grupo.

#### **READWRITE**

Invierte la propiedad READONLY. Están habilitadas las actualizaciones para los objetos del grupo de archivos.

#### **DEFAULT**

Especifica que el grupo de archivos es el predeterminado de la base de datos. Sólo un grupo puede ser el predeterminado, esta propiedad se quita del grupo de archivos que había sido anteriormente el predeterminado. CREATE DATABASE hace que el grupo de archivos principal sea el grupo predeterminado inicialmente. Si no se especifica ningún grupo de archivos en las instrucciones CREATE TABLE, ALTER TABLE o CREATE INDEX, se crean nuevas tablas e índices en el grupo predeterminado.

#### **SET**

Permite establecer valores para algunas de las características de trabajo en una base de datos, por ejemplo el tipo de recovery que se empleará para los backups.

#### **COLLATE**

Especifica el conjunto de caracteres a emplear, ya sean de Windows o de SQL Server 2000.

#### **Observaciones**

No se puede agregar o quitar un archivo mientras se está ejecutando una instrucción BACKUP.

#### **Ejemplo 1**

Modificar la base de datos Prueba2, de tal manera que le debe agregar un archivo de datos secundario de 5MB y un tamaño máximo de 10 MB. con un crecimiento de 1MB. Antes de ejecutar el siguiente comando utilice Sp\_HelpDB Prueba2, para comparar luego con los resultados después de ejecutar la sentencia.

**USE master**

**GO**

**ALTER DATABASE Prueba2**

**ADD FILE**

**(**

**NAME = Prueba2Sec\_Data,**

**FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL\data\**

**Prue2Data.ndf',**

**SIZE = 5MB,**

**MAXSIZE = 10MB,**

**FILEGROWTH = 1MB**

**)**

**GO**

**Sp\_HelpDB Prueba2**

**GO**

**-- Compare los resultados con los anteriores**

Si desea información de los archivos, emplee la siguiente sentencia:

**Use Prueba2**

**GO**

**Sp\_HelpFile**

**GO**

Si desea ver las características sólo del archivo que agrego utilice la siguiente sentencia:

**Sp\_HelpFile Prueba2Sec\_Data**

**GO**

**/\* El resultado le mostrará información del archivo que acaba de agregar \*/**

### ***Ejemplo 2***

Crear dos grupos de archivos en la base de datos Prueba2, el primer grupo se llamará CONSULTORES y el otro se llamará OPERACIONES.

**ALTER DATABASE Prueba2**

**ADD FILEGROUP Consultores**

**GO**

**ALTER DATABASE Prueba2**

**ADD FILEGROUP Operaciones**

**GO**

**-- Verifique la información con las siguientes instrucciones:**

**Use Prueba2**

**GO**

**Sp\_HelpFileGroup**

**GO**

Se mostrará el siguiente resultado:

groupname	groupid	filecount
Consultores	2	0
Operaciones	3	0
PRIMARY	1	2

### ***Ejemplo 3***

A cada uno de los grupos creados anteriormente añadale dos archivos de datos, para ello considere lo siguiente: los archivos del grupo CONSULTORES deben tener un tamaño de 10 MB. cada uno, con un tamaño máximo de 20 MB y un crecimiento de

2 MB., mientras que los del grupo OPERACIONES tendrán un tamaño inicial de 5 MB y un máximo de 30 MB. con un crecimiento de 5 Mb.

```
Use Master
GO
ALTER DATABASE Prueba2
ADD FILE
(NAME          = 'DatCons01',
 FILENAME      = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\DatCons1.ndf',
 SIZE          = 10MB,
 MAXSIZE       = 20MB,
 FILEGROWTH    = 2MB),
(NAME          = 'DatCons02',
 FILENAME      = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\DatCons2.ndf',
 SIZE          = 10MB,
 MAXSIZE       = 20MB,
 FILEGROWTH    = 2MB),
TO FILEGROUP CONSULTORES
GO
ALTER DATABASE Prueba2
ADD FILE
(NAME          = 'DatOper01',
 FILENAME      = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\DatOper1.ndf',
 SIZE          = 5MB,
 MAXSIZE       = 30MB,
 FILEGROWTH    = 5MB),
(NAME          = 'DatOper02',
 FILENAME      = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\DatOper2.ndf',
 SIZE          = 5MB,
 MAXSIZE       = 30MB,
 FILEGROWTH    = 5MB),
TO FILEGROUP OPERACIONES
GO
```

Una vez que ejecuta estas sentencias, verifique la información con la siguiente instrucción:

```
Use Prueba2
GO
Sp_HelpFileGroup
GO
```

groupname	groupid	filecount
Consultores	2	2
Operaciones	3	2
PRIMARY	1	2

Si desea información de un grupo de archivos en particular, utilice:

```
Sp_HelpFileGroup Operaciones  
GO
```

#### ***Ejemplo 4***

Modificar el tamaño del DatOper01 asignándole como nuevo tamaño máximo 40 Mb.

```
Use Master  
GO  
ALTER DATABASE Prueba2  
MODIFY FILE  
( NAME      = 'DatOper01',  
  MAXSIZE   = 40Mb)  
GO
```

-- Para verificar el cambio emplee la siguiente instrucción:

```
Sp_HelpFile DatOper01  
GO
```

#### ***Ejemplo 5***

Eliminar el archivo DatOper01.

```
Use Master  
GO  
ALTER DATABASE Prueba2  
REMOVE FILE 'DatOper01'  
GO
```

#### ***Ejemplo 6***

Hacer que el grupo de archivos Operaciones sea el grupo En forma predeterminada.

```
Use Master  
GO  
ALTER DATABASE Prueba2  
MODIFY FILEGROUP Operaciones DEFAULT  
GO
```

Cuando lo que se requiere es eliminar una base de datos, debe emplear la sentencia DROP DATABASE, cuya sintaxis es la siguiente:

```
DROP DATABASE database_name [,...n]
```

#### ***Ejemplo 7***

Eliminar la base de datos Prueba2.

```
Use Master
```

```
GO  
DROP DATABASE Prueba2  
GO
```

Revisar la tabla SysDatabases, verifique que se elimino la entrada de Prueba2

### ***Ejemplo 8***

Eliminar la base de datos Prueba1 y NuevoNombre

```
Use Master  
GO  
DROP DATABASE Prueba1, NuevoNombre  
GO
```

Revisar la tabla SysDatabases, verifique que se elimino la entrada de Prueba2

```
Use Master  
GO  
Select Name From SysDatabases  
GO
```

Renombrando Base de Datos

Para quitar una base de datos, utilice DROP DATABASE. Para cambiar el nombre de una base de datos, utilice **sp\_renamedb**, pero recuerde que para esto la base de datos a renombrar tiene que tener activa la opción **'single user'**, si desea comprobar el empleo de esta sentencia realice la siguiente secuencia de instrucciones desde el Analizador de Consultas, verifique la base de datos Prueba3 (creada en el Ejemplo3) no este seleccionada en el Administrador Empresarial, para asegurarse haga clic izquierdo en Databases, luego ingrese al Analizador de Consultas con una cuenta de administrador (Windows authentication) o con la cuenta sa :

```
/* Comprueba la presencia de Prueba3 */  
Use Master  
GO  
Select name From SysDatabases  
GO
```

El resultado sería:

<u>Name</u>
master
tempdb
model
msdb
pubs
Northwind
Prueba3
Prueba1
Prueba2

**/\* Para renombrar active la opción Single User en la Base de datos a renombrar \*/**

**Sp\_DBOption 'Prueba3', 'Single User', 'True'**  
**GO**

El resultado sería:

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

The database is now single usuario.

**/\* En este instante la base de datos puede ser renombrada \*/**

**Sp\_RenameDB 'Prueba3', 'NuevoNombre'**  
**GO**

El resultado sería:

(1 row(s) affected)

DBCC execution completed. If DBCC printed error messages, contact your system administrator.

The database is renamed and in single usuario mode.

A member of the sysadmin rol must reset the database to multiusuario mode with sp\_dboption.

**/\* Compruebe el correcto renombrado de la base de datos y luego retire la opción single usuario de la base de datos \*/**

**Select Name From SysDatabases**  
**GO**

**Sp\_DBOption 'NuevoNombre', 'Single Usuario', 'True'**  
**GO**

## **Creación de Tablas**

### **Objetivos:**

- Determinar Tipos de datos de SQL Server a utilizar en las tablas
- Implementación de tablas, establecer restricciones
- Asignar permisos

### ***Temas:***

- Tipos de datos de SQL Server – Tipos de datos de usuario
- Empleo de comando DDL (Data Definition Language)
- Implementar Restricciones
- Asignar roles y/o permisos – Comandos DCL (Data Control Language)

## **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 que 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.  
Si un objeto se define como **money**, puede contener hasta 19 dígitos y 4 de ellos pueden estar a la derecha del decimal. El objeto usa 8 bytes para almacenar los datos. Por tanto, el tipo de datos **money** tiene una precisión de 19, una escala de 4 y una longitud de 8.

## **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. Los datos **binary** y **varbinary** 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.

La siguiente es una tabla que describe los tipos de datos provistos por SQL Server:





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*&(%B99nh 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 <sup>63</sup> – 9223372036854775808) hasta 2 <sup>63</sup> –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.

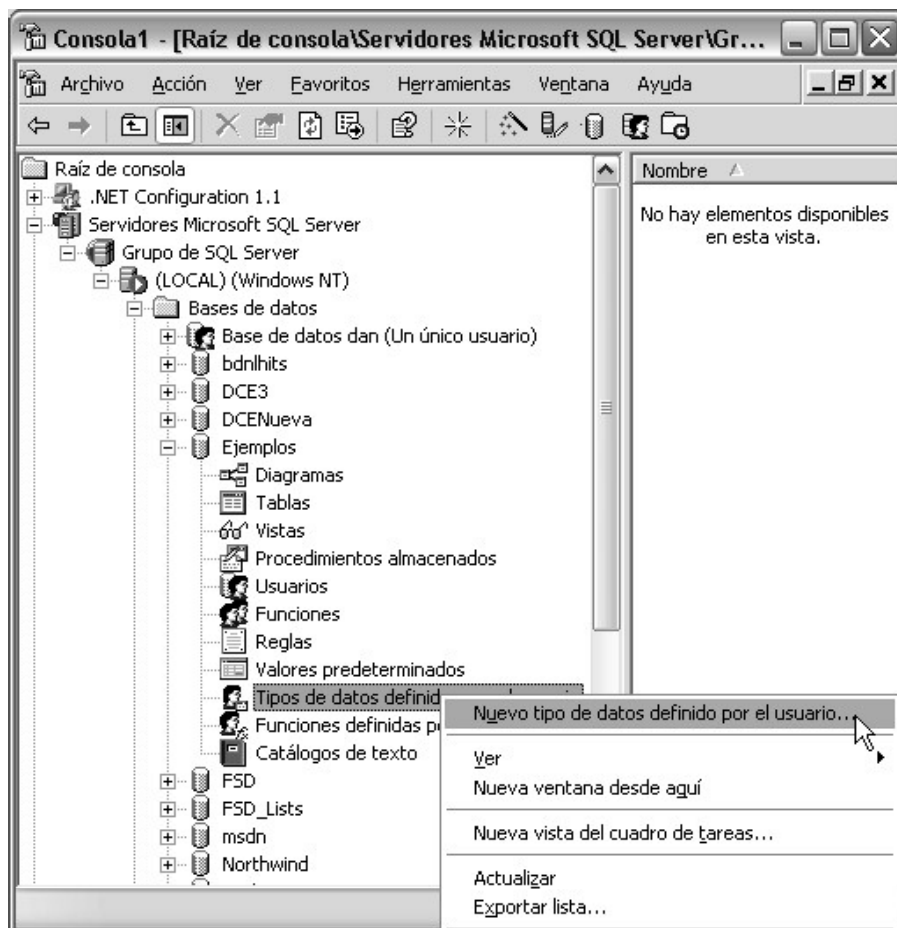
Categoría	Descripción	Tipo de Dato	Descripción
		smallint	Datos en el rango de –32,768 hasta 32,767. Se requieren 2 bytes por cada valor de este tipo.
		tinyint	Datos entre 0 y 255, se requiere de 1 byte.
Monetario	Cantidades monetarias positivas o negativas.	money	Datos monetarios entre –922,337,203,685,477.5808 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.

Categoría	Descripción	Tipo de Dato	Descripción
	sobre una columna valores que incluyan este conjunto de caracteres. Hay que recordar que los datos Unicode emplean dos bytes por cada carácter a representar.	nvarchar	Datos que pueden variar, hasta 4000 caracteres Unicode.
		ntext	Datos que exceden los 4000 caracteres Unicode.

## 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.

Para crear un tipo de dato puede emplear el Administrador Empresarial expandiendo la base de datos donde desea crear el dato, luego deberá hacer un clic derecho sobre Tipos de datos definidos por el Usuario y seleccionar “*Nuevo tipo de datos definido por el usuario...*”, tal como lo muestra la siguiente representación:



Complete la caja de diálogo, tal como lo muestra la siguiente representación:



Desde el Analizador de Consultas puede emplear el stored procedure del sistema SP\_ADDTYPE cuya sintaxis es la siguiente:

**sp\_addtype [@typename =] tipo,  
[@phystype =] tipoDatosSistema  
, [@nulltype =] 'tipoNull'**

#### Argumentos

**[@typename =] tipo**

Es el nombre para el tipo de datos definido de usuario, deben ser únicos en cada base de datos.

**[@phystype =] tipoDatosSistema**

Es el tipo de datos proporcionado por SQL Server, (decimal, int, etc.) en el que se basa el tipo de datos del usuario.

**[@nulltype =] 'tipoNull'**

Indica la forma en que el tipo de datos del usuario trata los valores nulos. Este argumento puede tener como valor 'NULL', 'NOT NULL' o 'NONULL'. Si no se define explícitamente tipoNull, se establece de acuerdo al criterio predeterminado para valores nulos.

#### **Ejemplo 1**

En este ejemplo se creará la base de datos Ejemplo y en ella se definirá el tipo de datos RUC de tipo char(11) el cual no permitirá valores NULL.

**USE master**

**GO**

**CREATE DATABASE Ejemplo**

```

ON PRIMARY
    (NAME          = 'Ejem_Data',
      FILENAME     = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\EjemData.Mdf',
      SIZE         = 20Mb,
      MAXSIZE      = 40Mb,
      FILEGROWTH   = 2Mb)
LOG ON
    (NAME          = 'Ejem_Log',
      FILENAME     = 'C:\Program Files\Microsoft SQL
Server\MSSQL\data\EjemLog.Ldf',
      SIZE         = 5Mb,
      MAXSIZE      = 10Mb,
      FILEGROWTH   = 1Mb)

GO
Use Ejemplo
GO
Sp_Addtype RUC, 'CHAR(11)', 'NOT NULL'
GO

```

Para verificar que el tipo de dato se ha creado exitosamente, ejecute la siguiente consulta:

```

Select * From Systypes
GO

```

## ***Ejemplo 2***

En este ejemplo se creará el tipo de datos Onomástico de tipo datetime y que permitirá valores NULL.

```

EXEC sp_addtype Onomastico, datetime, 'NULL'
GO

```

Los tipos de datos que se agregan son inscritos en la tabla **systypes**, estos tipos de datos son eliminados con el procedimiento almacenado del sistema **sp\_droptype**.

```

sp_droptype [@typename =] 'tipode dato'

```

Ejemplo:

```

Use Ejemplo
GO
Sp_droptype 'Onomastico'
GO
Select * From Systypes
GO
Sp_droptype 'RUC'
GO
Select * From Systypes
GO

```

## **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, ruc, 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.

Podemos distinguir los siguientes tipos de tablas:

### **Tablas del Sistema**

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

Si desea obtener información almacenada en las tablas del sistema 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.

### **Ejemplo:**

```
SELECT * FROM SYSUSUARIOS
```

### **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 la referencian se desconectan del SQL Server.

## Creación de tablas

Cuando se crea una tabla debe asignarle 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 nos 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

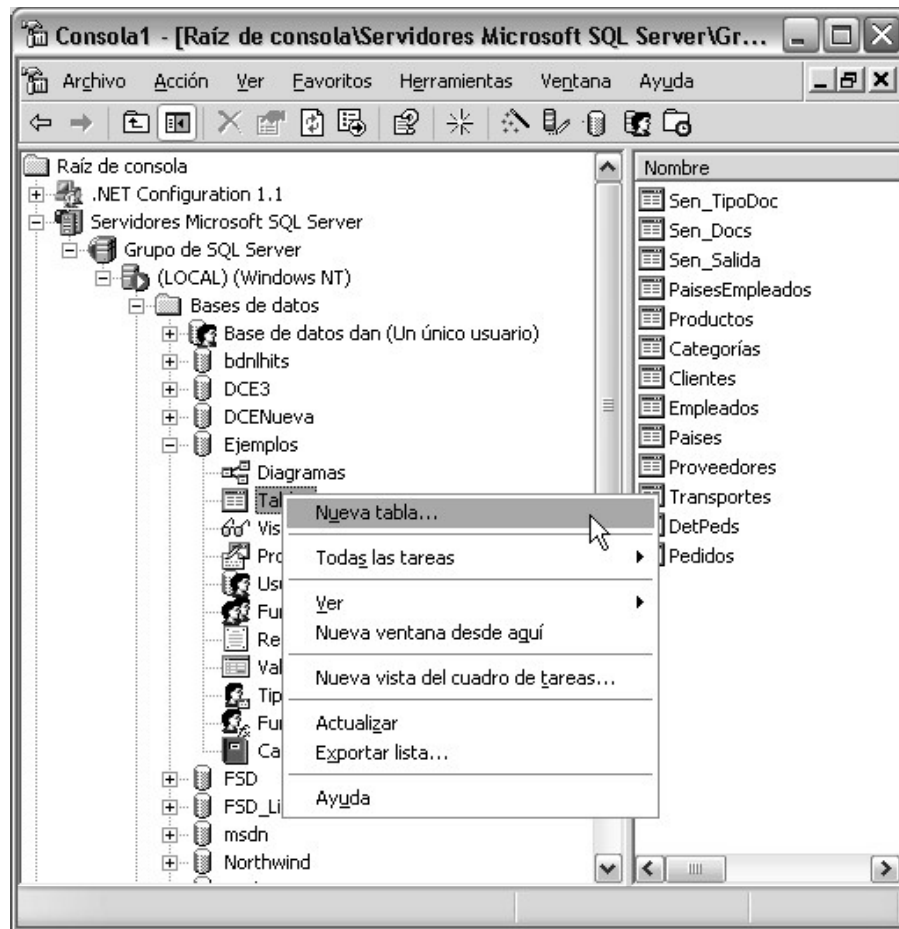
- billones de tablas por base de datos
- 1024 columnas por tabla
- 8060 es el tamaño máximo de registro (sin considerar datos image, text y ntext)
- Al momento de definir una columna se puede especificar si la columna soporta o no valores NULL.

Para crear tablas debe utilizar la sentencia CREATE TABLE, cuya sintaxis es la siguiente:

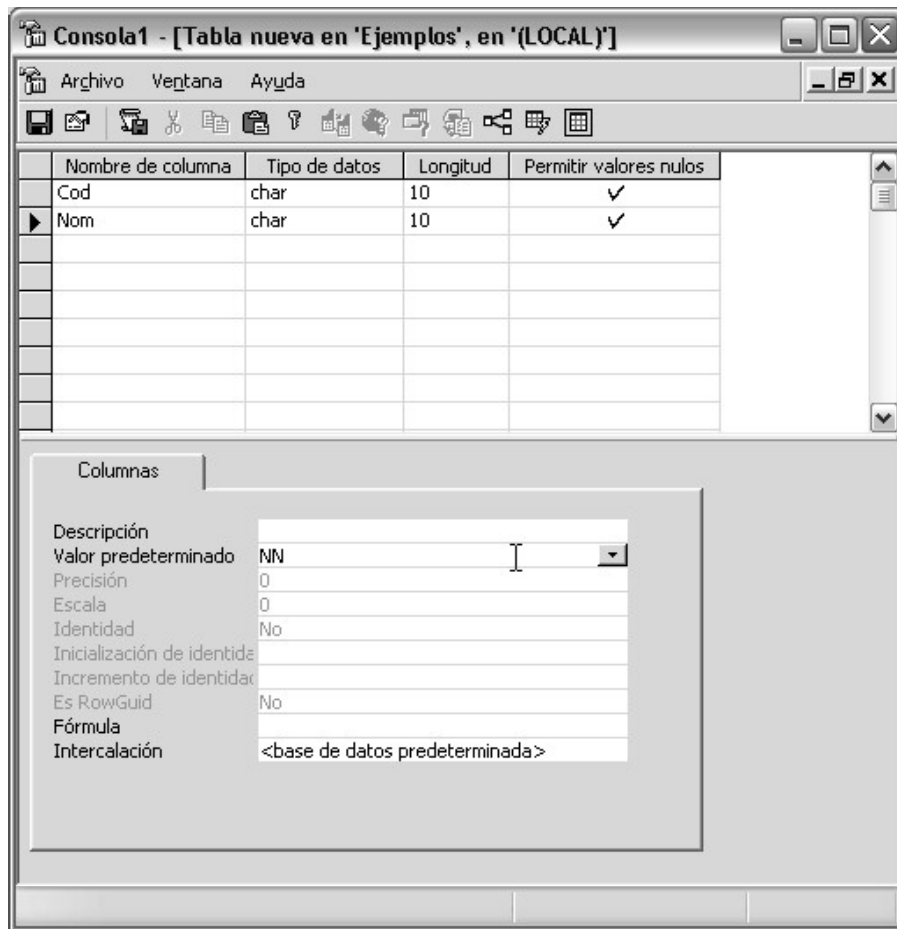
```
CREATE TABLE <Nombre de Tabla>  
    (Nom_Columna1    Tipo_de_Dato                [NULL I NOT NULL],  
     Nom_Columna2    Tipo_de_Dato                [NULL I NOT NULL],  
     Nom_Columna3 As formula ...)  
GO
```

También puede crear sus tablas desde el Administrador Empresarial, para ello extienda la carpeta Tablas de la base de datos donde creará la tabla, haga clic derecho y seleccione Nueva Tabla, tal como lo indica la siguiente representación:





Aparecerá la siguiente caja de diálogo, complete de acuerdo a la representación:

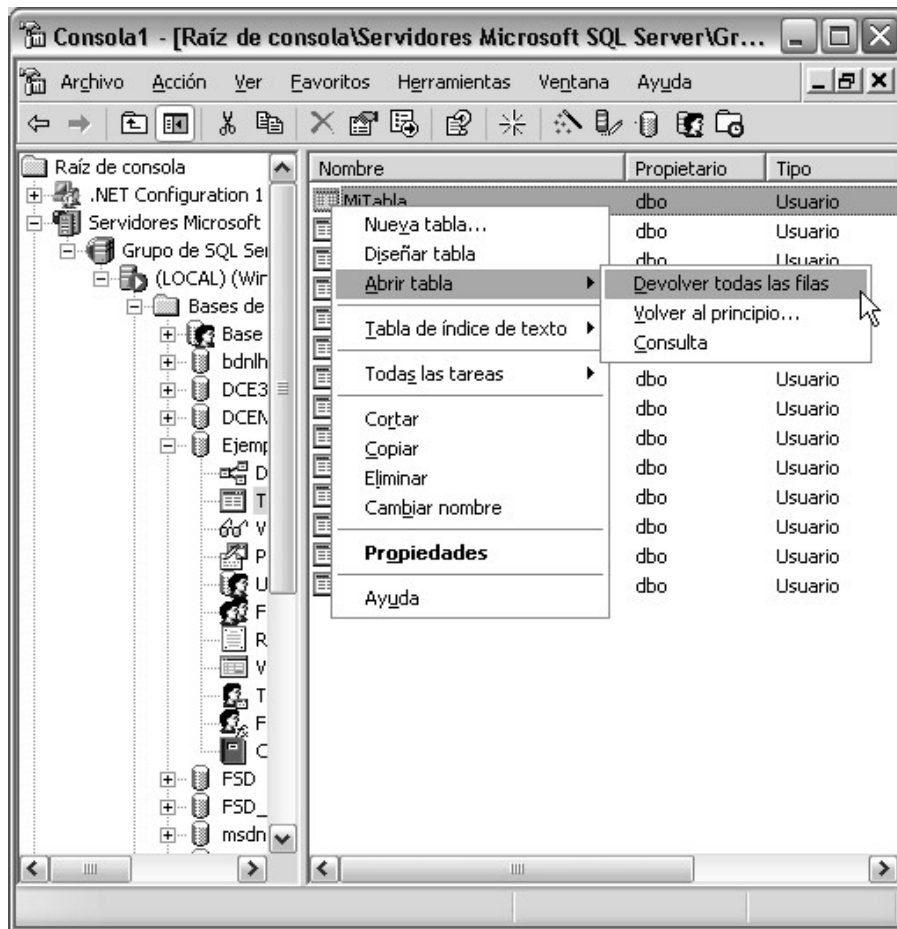


Cuando finalice pulse el icono de grabar y asigne el nombre de acuerdo a la representación:



Luego de pulsar Aceptar, pulse la combinación Ctrl-F4 y podrá observar que el icono correspondiente a esta nueva tabla aparece en el panel de la derecha.

Para agregar los registros de prueba de esta tabla, haga clic derecho sobre la tabla DemoTabla, seleccione la opción Open table y luego un clic en Return all rows, tal como lo muestra la siguiente figura:



Agregar unos registros al finalizar pulse Ctrl-F4.

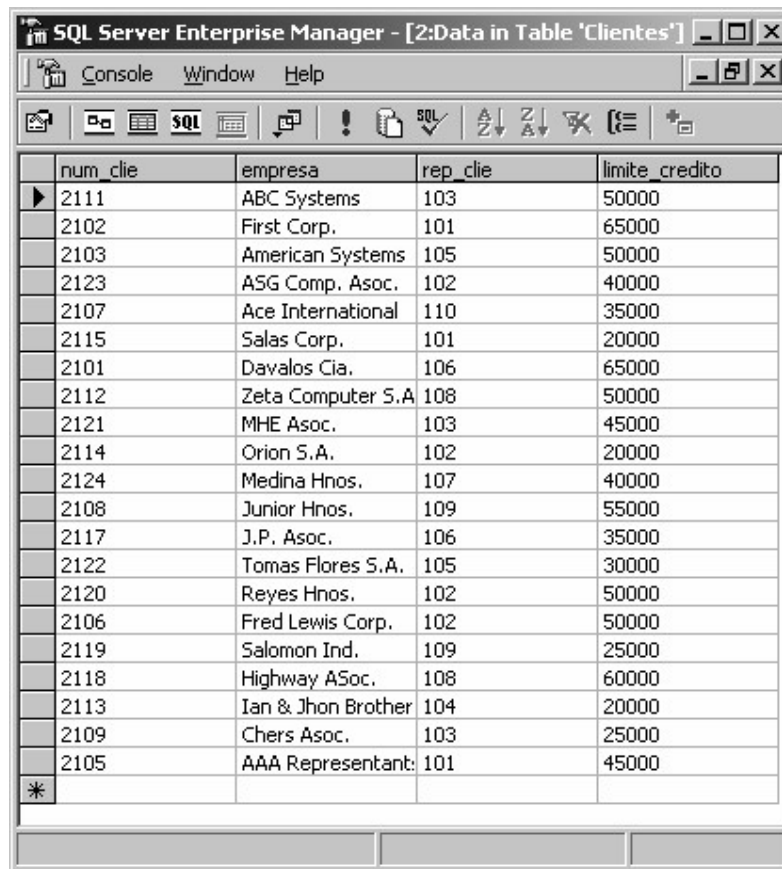
## Ejercicios:

En la base de datos Ejemplo, crear las siguientes tablas:

### **CLIENTES**

Nombre de Columna	Tipo de dato	Permite NULL
num_clie	integer	NOT NULL
empresa	varchar(20)	NOT NULL
rep_clie	Integer	NULL
limite_credito	Money	NULL

Agregar los siguientes registros a la tabla Clientes:



The screenshot shows the SQL Server Enterprise Manager interface with the 'Clientes' table open. The table contains 21 records, each with a unique 'num\_clie' and associated company name, representative ID, and credit limit. The records are as follows:

num_clie	empresa	rep_clie	limite_credito
2111	ABC Systems	103	50000
2102	First Corp.	101	65000
2103	American Systems	105	50000
2123	ASG Comp. Asoc.	102	40000
2107	Ace International	110	35000
2115	Salas Corp.	101	20000
2101	Davalos Cia.	106	65000
2112	Zeta Computer S.A	108	50000
2121	MHE Asoc.	103	45000
2114	Orion S.A.	102	20000
2124	Medina Hnos.	107	40000
2108	Junior Hnos.	109	55000
2117	J.P. Asoc.	106	35000
2122	Tomas Flores S.A.	105	30000
2120	Reyes Hnos.	102	50000
2106	Fred Lewis Corp.	102	50000
2119	Salomon Ind.	109	25000
2118	Highway ASoc.	108	60000
2113	Ian & Jhon Brother	104	20000
2109	Chers Asoc.	103	25000
2105	AAA Representant:	101	45000

Al terminar pulse CTRL-F4 con ello los registros permanecerán en la tabla.

### **RepVentas**

Nombre de Columna	Tipo de dato	Permite NULL
num_empl	integer	NOT NULL
nombre	varchar(15)	NOT NULL
edad	integer	
oficina_rep	integer	

titulo	varchar(10)	
contrato	date	NOT NULL
director	integer	
cuota	money	
ventas	money	NOT NULL

También podemos crear tablas a partir de sentencias del Transact para ingrese al Analizador de Consultas y ejecute las siguientes instrucciones:

#### Use Ejemplo

**GO**

**CREATE TABLE Oficinas**

**( oficina integer not null,  
ciudad varchar(15) not null,  
region varchar(10) not null,  
dir integer,  
objetivo money,  
ventas money not null)**

**GO**

#### **PEDIDOS**

Nombre de Columna	Tipo de dato	Permite NULL
num pedido	integer	NOT NULL
fecha pedido	datetime	NOT NULL
clie	Integer	NOT NULL
rep	integer	
fab	char(3)	NOT NULL
producto	char(5)	NOT NULL
cant	integer	NOT NULL
importe	money	NOT NULL

#### **PRODUCTOS**

Nombre de Columna	Tipo de dato	Permite NULL
id fab	char(3)	NOT NULL
id producto	char(5)	NOT NULL
descripcion	varchar(20)	NOT NULL
precio	Money	NOT NULL
existencias	Integer	NOT NULL

Nota: Una vez que terminó de crear las tablas ejecute el script AgregaDatos.sql para poder poblar la información de las tablas.

#### **Modificación de la estructura de las tablas**

Con SQL Server 2000 se puede modificar la estructura de las tablas, se podrá agregar, eliminar o modificar las características de las columnas de la tabla.

Para demostrar el empleo de estas instrucciones emplearemos una tabla de prueba a la cual le daremos la siguiente estructura:

#### **Use Ejemplo**

**GO**

#### **Create Table Prueba**

```
( cod      char(1)      NOT NULL,  
  nom      char(20)     NULL,  
  pat      varchar(20)  NOT NULL,  
  mat      varchar(20)  NOT NULL)
```

**GO**

Verificar la creación de la tabla con la siguiente instrucción:

#### **Sp\_Help Prueba**

**GO**

Ahora modificaremos el campo nom para asignarle como tipo de datos varchar con la misma longitud y que no permita valores NULL

#### **ALTER TABLE Prueba**

**ALTER COLUMN nom varchar(20) NOT NULL**

**GO**

Verifique empleando:

#### **Sp\_Help Prueba**

**GO**

Luego agregaremos un campo llamado Sueldo de tipo money:

#### **ALTER TABLE Prueba**

**ADD Sueldo money**

**GO**

Una observación es que cuando agrega una columna con ALTER TABLE no puede utilizar NOT NULL, salvo que emplee la propiedad IDENTITY en una columna.

Verifique la modificación de la tabla:

#### **Sp\_Help Prueba**

**GO**

Agregar la columna fecha\_nac, de tipo datetime:

#### **ALTER TABLE Prueba**

**ADD fecha\_nac datetime**

**GO**

**Sp\_Help Prueba**

**GO**

Ahora eliminaremos la columna sueldo:

**ALTER TABLE Prueba**

**DROP COLUMN sueldo**

**GO**

**Sp\_Help Prueba**

**GO**

Una observación importante, antes de eliminar una columna deberá eliminar los índices basados en esa columna.

Ahora eliminaremos la columna cod:

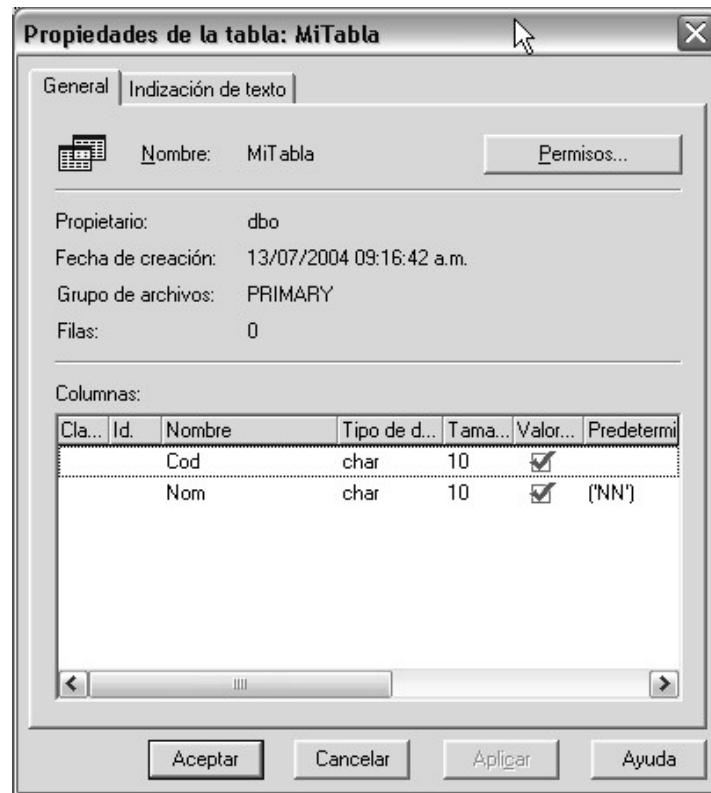
**ALTER TABLE Prueba**

**DROP COLUMN cod**

**GO**

Otra de las formas de modificar la estructura de una tabla es desde el Administrador Empresarial, para ello haga clic derecho sobre la tabla a modificar y seleccione la opción **Diseñar Tabla**, tendrá una presentación similar a la que utilizó al momento de crear la tabla.

Si tan sólo desea observar la estructura haga doble clic sobre la tabla y aparecerá la siguiente presentación:



## Valores autogenerados para las columnas

En SQL Server 2000 se puede definir columnas que obtengan valores generados por el sistema, para ello podemos hacer uso de:

### 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 debe especificar un valor de inicio y uno de incremento. Recuerde que este tipo de columnas no son editables.

Ejemplo:

**USE EJEMPLO**

**GO**

**ALTER TABLE Prueba**

**ADD COLUMN cod integer Identity(1,1) NOT NULL**

**GO**

Para comprobar la generación de los valores ejecute la siguiente secuencia de comandos:

**USE EJEMPLO**



```

GO
INSERT PRUEBA VALUES ('JOSE', 'ROJAS', 'CALDERON',1000)
GO
INSERT PRUEBA VALUES ('ANA MARIA', 'SALAS', 'GUILLEN',1000)
GO
SELECT COD, NOM, PAT, MAT, SUELDO FROM PRUEBA
GO

```

Para ver información sobre la columna IDENTITY puede utilizar las funciones:

```

Select Ident_Seed('Prueba')      /* Retorna el valor de inicio de la columna
identity */
GO
Select Ident_Incr('Prueba') /* Retorna el valor de incremento de la columna
identity */
GO

```

### **Función NEWID y Datos de tipo UNIQUEIDENTIFIER**

El tipo de dato uniqueidentifier y la función NEWID trabajan unidas para poder generar valores en el formato GUID.

Este tipo de datos no genera valores automáticamente, sino que por el contrario hay que definirle un valor En forma predeterminada que especifique el empleo de la función NEWID.

Para poder observar un ejemplo de lo antes explicado, ejecute la siguiente secuencia de comandos:

```

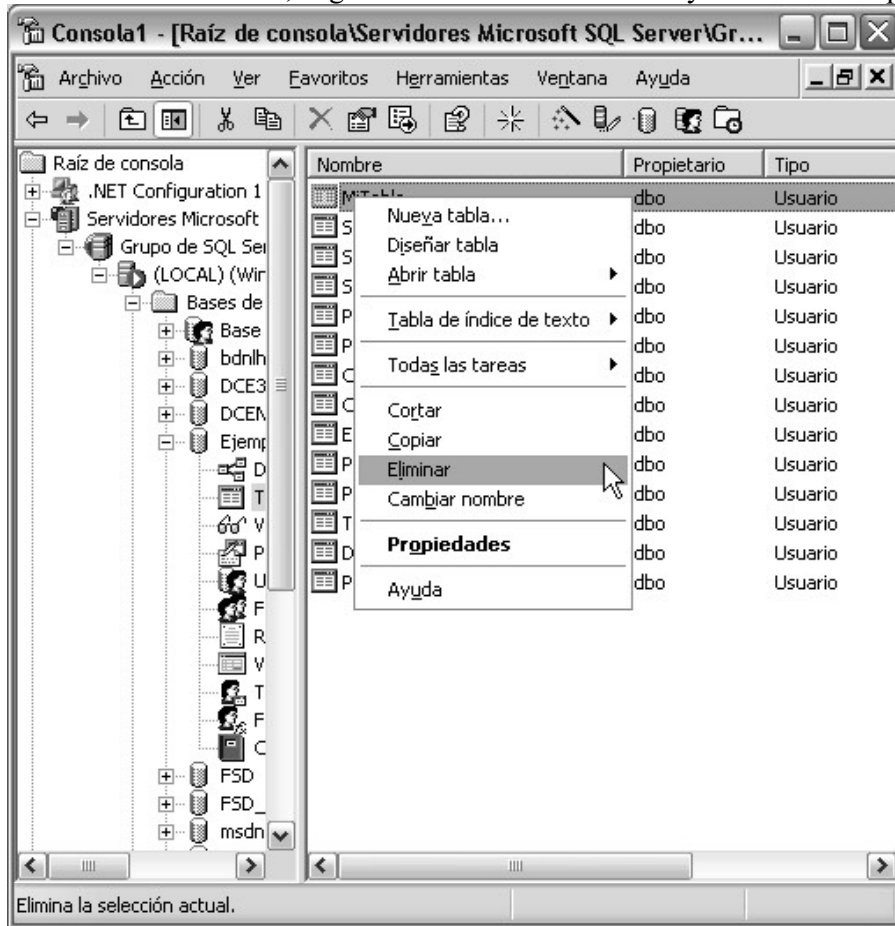
CREATE TABLE Prueba2
( código      uniqueidentifier NOT NULL DEFAULT NEWID(),
  nombre      char(20) NOT NULL)
GO

INSERT Prueba2 (nombre) VALUES ('Mauricio')
GO
INSERT Prueba2 (nombre) VALUES ('Gina')
GO
INSERT Prueba2 (nombre) VALUES ('Cristina')
GO
SELECT * FROM Prueba2
GO

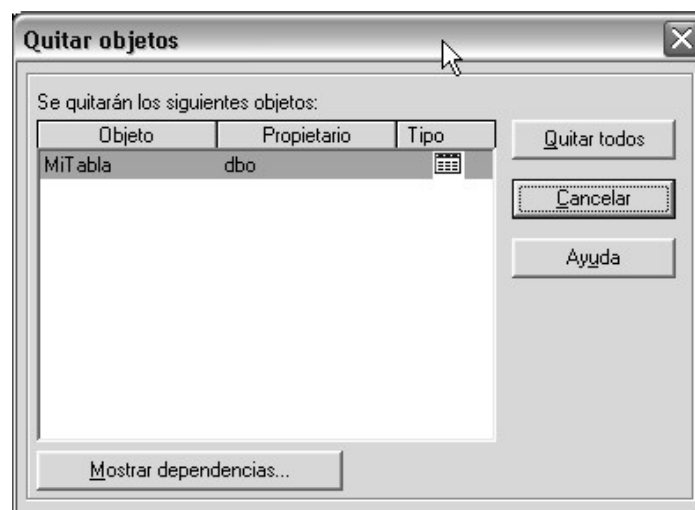
```

### **Eliminación de tablas**

Para eliminar una tabla, haga clic derecho sobre la tabla y seleccione la opción Eliminar



aparecerá la siguiente caja de diálogo:



Pulse clic sobre *Quitar Todos* y con ello la tabla será retirada de la base de datos.

Otra forma es utilizando la sentencia DROP TABLE cuya sintaxis es la siguiente:

**DROP TABLE <Nombre de la Tabla>**

Para probar el empleo de esta instrucción utilice la siguiente sentencia:

```
DROP TABLE Prueba2  
GO
```

Compruebe que las tablas Prueba y Prueba2 están eliminadas, con la siguiente instrucción:

```
SELECT NAME FROM SYSOBJECTS WHERE TYPE='U'  
GO
```

### **Implementar Restricciones**

Uno de los principales objetivos de una base de datos relacional es cuidar y controlar la integridad de 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
- Podría modificarse los datos existentes con valores incorrectos
- Los cambios a la base de datos podrían aplicarse 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 mas sencillas que especifican que columnas permiten valores nulos y que columnas no. Al momento de definir las columnas de una tabla 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.
- **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.

En esta parte del capítulo revisaremos 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 mantendrá.

Los constraints pueden definirse 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.

## Tipos de Constraint

Tipo de Integridad	Tipo de Constraint
Controlar rango de valores sobre las columnas.	DEFAULT
	CHECK
Unicidad de registros y valores únicos.	PRIMARY KEY
	UNIQUE
Referencial	FOREIGN KEY
	CHECK

### Definir restricción PRIMARY KEY

```
ALTER TABLE <Nombre de la Tabla>  
ADD CONSTRAINT <Nombre del Constraint>  
PRIMARY KEY (columna1, ...)  
GO
```

Un constraint de tipo PRIMARY KEY asegura la unicidad de cada fila en la tabla, sólo se podrá definir uno por tabla y debemos recordar que no permite valores NULL.

En forma predeterminada crea un índice CLUSTERED.

### *Ejemplos:*

Implementar un PRIMARY KEY Constraint que asegure la unicidad de cada cliente.

```
Use Ejemplo  
GO
```

```
Select * From Clientes /* Note el orden de los códigos de clientes */  
GO
```

```
ALTER TABLE Clientes  
ADD CONSTRAINT PK_Cli_numclie  
PRIMARY KEY (num_clie)  
GO
```

```
Select * From Clientes /* Note que las filas aparecen ordenadas */  
GO
```

Implementar un PRIMARY KEY Constraint que asegure la unicidad de cada representante de ventas.

```
Select * From RepVentas /* Note el orden de los códigos de empleados */  
GO
```

```
ALTER TABLE RepVentas  
ADD CONSTRAINT PK_num_clie  
PRIMARY KEY (num_empl)  
GO
```

```
Select * From RepVentas /* Note que las filas aparecen ordenadas */  
GO
```

Implementar un PRIMARY KEY Constraint que asegure la unicidad de cada oficina.

```
Select * From Oficinas /* Note el orden de los códigos de oficinas */  
GO
```

```
ALTER TABLE Oficinas  
ADD CONSTRAINT PK_Oficina  
PRIMARY KEY (Oficina)  
GO
```

```
Select * From Oficinas /* Note que las filas aparecen ordenadas */  
GO
```

Implementar un PRIMARY KEY Constraint que asegure la unicidad de cada pedido.

```
Select * From Pedidos /* Note el orden de los códigos de pedidos */  
GO
```

```
ALTER TABLE Pedidos  
ADD CONSTRAINT PK_num_pedido  
PRIMARY KEY (num_pedido)  
GO
```

```
Select * From Pedidos /* Note que las filas aparecen ordenadas */  
GO
```

Implementar un PRIMARY KEY Constraint que asegure la unicidad de cada producto.

```
Select * From Productos /* Note el orden de los códigos de producto */  
GO
```

```
ALTER TABLE Productos  
ADD CONSTRAINT PK_fab_prod  
PRIMARY KEY (id_fab, id_producto)  
GO
```

```
Select * From Productos /* Note que las filas aparecen ordenadas */
```

**GO**

### **Definir FOREIGN KEY Constraint**

```
ALTER TABLE <Nombre de la Tabla>  
ADD CONSTRAINT <Nombre del Constraint>  
FOREIGN KEY (columna1, ...)  
REFERENCES Tabla(columna, ...)  
GO
```

Un foreign key constraint permjite forzar la integridad de datos manteniendo la relación entre una llave primaria y una llave secundaria.

Para implementar este tipo de característica debemos recordar que el número de columnas y el tipo de datos referenciados en la cláusula FOREIGN KEY debe ser el mismo que el mencionado en la cláusula REFERENCES

### ***Ejemplos:***

Implementar un foreign key constraint que asegure que cada vez que asigne un representante de ventas a un cliente este exista.

**USE Ejemplo**  
**GO**

```
ALTER TABLE Clientes  
ADD CONSTRAINT FK_Cli_RepVentas  
FOREIGN KEY (Rep_Clie)  
REFERENCES RepVentas(Num_Empl)  
GO
```

Implementar un foreign key constraint que asegure que cada vez que a un representante de ventas se le asigne un director, esté se encuentre registrado.

```
ALTER TABLE RepVentas  
ADD CONSTRAINT FK_Dir_RepVentas  
FOREIGN KEY (Director)  
REFERENCES RepVentas(Num_Empl)  
GO
```

Implementar un foreign key constraint que asegure que la oficina asignada al representante de ventas se encuentre en la tabla oficinas.

```
ALTER TABLE RepVentas  
ADD CONSTRAINT FK_Ofi_Oficinas  
FOREIGN KEY (oficina_rep)  
REFERENCES Oficinas(Oficina)  
GO
```

Implementar un foreign key constraint que verifique el código de director de la oficina.

```
ALTER TABLE Oficinas  
ADD CONSTRAINT FK_Direc_RepVentas  
FOREIGN KEY (dir)  
REFERENCES RepVentas(num_empl)  
GO
```

Implementar un foreign key constraint que verifique la existencia del representante de ventas que toma un pedido.

```
ALTER TABLE Pedidos  
ADD CONSTRAINT FK_Rep_RepVentas  
FOREIGN KEY (rep)  
REFERENCES RepVentas(num_empl)  
GO
```

Implementar un foreign key constraint que verifique la existencia de los productos que se indican al momento de tomar un pedido.

```
ALTER TABLE Pedidos  
ADD CONSTRAINT FK_FabPro_Productos  
FOREIGN KEY (fab, producto)  
REFERENCES Productos(id_fab, id_producto)  
GO
```

Definir CHECK CONSTRAINT

```
ALTER TABLE <Nombre de la tabla>  
ADD CONSTRAINT <Nombre del Constraint>  
CHECK <Regla a validar>  
GO
```

Un Check Constraint restringe a los usuarios la posibilidad de ingresar valores inapropiados a una columna. Este constraint actúa cuando el usuario emplea una instrucción INSERT o UPDATE.

**Ejemplos:**

Implementar un check constraint que verifique que los códigos de los representantes de ventas sean mayores que 100.

```
ALTER TABLE RepVentas  
ADD CONSTRAINT CK_RV_100  
CHECK (Num_Empl > 100)  
GO
```

Implementar un check constraint que verifique que los códigos de los pedidos sean mayores que 100000.



```
ALTER TABLE Pedidos  
ADD CONSTRAINT CK_Pedidos  
CHECK (num_pedido > 100000)  
GO
```

**Implementar DEFAULT CONSTRAINTS**

```
ALTER TABLE <Nombre de la tabla>  
ADD CONSTRAINT <Nombre del constraint>  
DEFAULT <Valor En forma predeterminada>  
FOR <columna>  
GO
```

Estos constraints trabajan al momento de utilizar la función INSERT y asignan un valor automáticamente a la columna que no se le asignó.

### **Ejemplo:**

Asignar un valor en forma predeterminada a la columna DIRECTOR de la tabla que almacena los datos de los representantes de ventas haciendo que el código En forma predeterminada sea 106.

```
ALTER TABLE RepVentas  
ADD CONSTRAINT DF_RV_Director  
DEFAULT 106  
FOR Director  
GO
```

Como parte final de esta implementación emplearemos un conjunto de instrucciones para tratar de modificar la información de las distintas tablas y veremos como los constraints implementados realizan su trabajo.

Para ello ejecute las siguientes instrucciones desde el Analizador de Consultas

```
/* Intentemos agregar un cliente con el código 2113 */  
Insert Clientes Values (2113, 'Amago Sys.', 103, 15000)  
GO  
/* La sentencia falla debido a que el primary constraint 'PK_Cli_numclie'  
le impide incluir códigos duplicados */  
  
/* Ahora indicaremos un código de cliente apropiado pero el código de  
representante de ventas inexistente */  
Insert Clientes Values (3000, 'Amago Sys.', 250, 15000)  
GO  
  
/* En este caso el error se produce debido a que el foreign key constraint  
'FK_Cli_RepVentas' a detectado que el código del representante de ventas  
es inexistente */  
  
/* Ahora agregaremos un representante de ventas sin indicar el código de su  
director */  
Insert RepVentas Values (450, 'Karem Vigo', 33, 22, 'Rep.Ventas', '3/5/1991',  
DEFAULT, DEFAULT, 12000)  
GO  
  
Select num_empl, nombre, director From RepVentas Where num_empl = 450  
GO  
  
/* Luego de ejecutar estas instrucciones observará que el código de director  
asignado a Karem Vigo es el 106, código asignado por el Default Constraint  
*/  
  
/* Ahora intentaremos agregar un representante de ventas con código menor  
que 100*/
```

```
Insert RepVentas Values (50, 'Sofia Quevedo', 33, 22, 'Rep.Ventas', '3/5/1991',  
DEFAULT, DEFAULT, 12000)  
GO
```

```
/* En este caso el error se produce debido a que el check constraint  
a detectado que el código del representante de ventas es menor que  
100 */
```

## Diagrama de Base de Datos

Una vez que hemos terminado de implementar las restricciones especificadas anteriormente y luego que ya tenemos las restricciones funcionando podríamos dar un vistazo al diagrama de la base de datos:

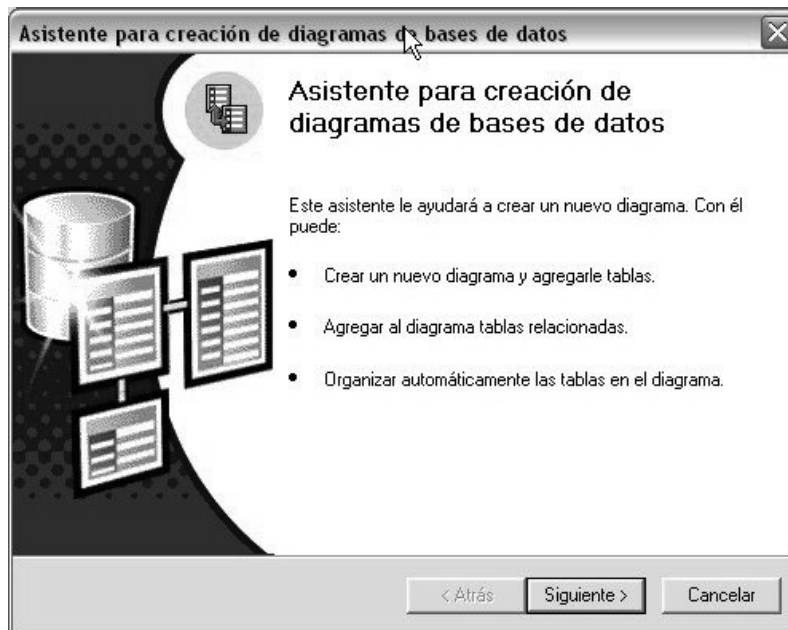
Los diagramas representan gráficamente la estructura de la Base de Datos, podemos ver sus tablas y diseño, además de las relaciones entre ellas. También se convierte en una herramienta gráfica para crear, modificar e implementar integridad y constancia de datos.

### **Ejemplo:**

1. Por lo tanto, haga clic derecho sobre Diagramas y seleccione la opción Nuevo Diagrama de base de datos, tal como lo muestra la figura:



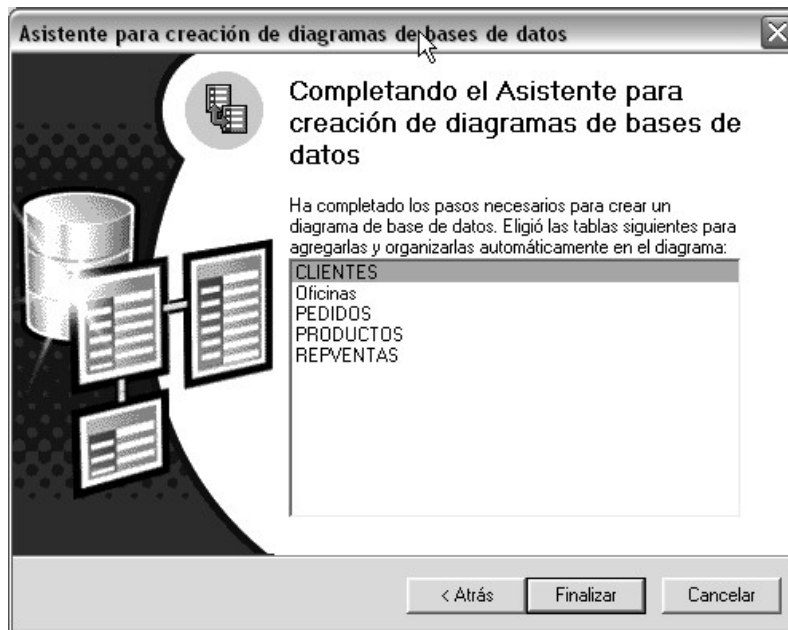
Luego de esto aparecerá un mensaje de bienvenida al asistente para definición del diagrama de la base de datos, tal como lo muestra la siguiente imagen:



2. Pulse Siguiente y se presentará una caja de diálogo, donde debe seleccionar las tablas que se muestran en la siguiente representación:



3. Luego de pulsar el botón Agregar y Siguiente, aparecerá la siguiente pantalla:



Luego de pulsar Finalizar tendrá la siguiente representación en pantalla:

