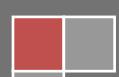


SQL Server 2008

Administración de
base de datos con
SQL Server 2008



Víctor José Vergel Rodríguez
victorvergel@verinsis.com





Víctor José Vergel Rodríguez

Módulo 1 ÍNDICE

Módulo 1	Índice	3
-----------------	---------------------	----------

Módulo 2	Introducción.....	7
-----------------	--------------------------	----------

2.1	Introducción.....	7
2.2	Características.....	7
2.3	Características de las diferentes versiones de sql server 2008.....	11
2.4	Instalación SQL Server 2008.....	12
2.5	Novedades.....	17
2.6	Componentes de SQL Server 2008.....	18

Módulo 3	Iniciación a la Administración.....	23
-----------------	--------------------------------------------	-----------

3.1	SQL Server Management Studio – Administrador corporativo	23
3.1.1	<i>Propiedades de la conexión actual</i>	25
3.1.2	<i>Registrar un servidor SQL Server vía IP pública:</i>	29
3.2	Creación de una BD.....	29
3.2.1	<i>BD del sistema.....</i>	29
3.2.2	<i>Creación de BD propias.....</i>	31
3.3	Crear un diagrama de base de datos.....	33
3.4	Adjuntar BDs	34
3.5	Separar	35
3.6	Copiar BD	35
3.7	Importar / Exportar datos	37
3.7.1	<i>Importación de una base de access</i>	41
3.7.2	<i>Importar datos desde oracle.....</i>	43
3.8	SQL Configuration Manager	44
3.9	SQLCMD.....	46
3.10	SQL Server Management Studio	49
3.10.1	<i>Registro de servidores.....</i>	49
3.10.2	<i>Explorador de objetos.....</i>	50
3.10.3	<i>Explorador de plantillas.....</i>	51
3.11	Instantáneas.....	51
3.11.1	<i>Asignar nombres a instantáneas de bases de datos</i>	52
3.11.2	<i>Conexiones de clientes con una instantánea de base de datos.....</i>	53
3.11.2.1	Restricciones a tener en cuenta sobre las instantáneas	55

Módulo 4	SQL Server 2008. Tablas, Índices y Vistas	56
-----------------	--------------------------------------------------------	-----------

4.1	Crear una nueva tabla /vista.....	56
4.1.1	<i>Creación de índices.....</i>	58
4.1.2	<i>Relaciones</i>	61
4.1.3	<i>Restricciones – CHECK</i>	62

Módulo 5	SQL.....	64
-----------------	-----------------	-----------

5.1	Introducción SQL (Structured Query Language)	64
5.2	DDL	64
5.2.1	<i>CREATE TABLE</i>	64



5.2.1.1	Restricciones:	67
5.2.1.2	Tipos de datos	67
5.2.2	<i>CREATE INDEX</i>	79
5.2.3	<i>DROP INDEX</i>	80
5.2.4	<i>ALTER TABLE</i>	80
5.2.5	<i>DROP TABLE</i>	80
5.2.6	<i>CREATE VIEW</i>	80
5.2.7	<i>ALTER VIEW</i>	81
5.2.8	<i>DROP VIEW</i>	81
5.2.9	<i>CREATE TRIGGER</i>	81
5.2.10	<i>ALTER TRIGGER</i>	81
5.2.11	<i>DROP TRIGGER</i>	82
5.3	DML	82
5.3.1	<i>Select</i>	82
5.3.1.1	Cláusula FROM	88
5.3.1.2	GROUP BY - HAVING	91
5.3.1.3	COMPUTE	93
5.3.2	<i>Funciones de agrupación: SUM, MAX, MIN, AVG, COUNT</i>	93
5.3.3	<i>Funciones: CONVERT, GETDATE, DATEDIFF, DATEPART, SOUNDEX, SUBSTRING, LEFT Y RIGHT, UPPER, CHARINDEX, RTRIM Y LTRIM, LEN, REPLICATE, SPACE, REPLACE STR, CHAR, ASCII</i>	96
5.3.4	<i>Funciones de sistema: ISNULL, COALESCE, USER_ID, USER_NAME, DATALENGTH, COL_LENGTH, CONVERT</i>	102
5.3.4.1	Query designer	105
5.3.5	<i>Consultas avanzadas:</i>	106
5.3.5.1	Selecciones con conjuntos. UNION; INTERSECT; EXCEPT	106
5.3.5.2	Selecciones con subconsultas	106
5.3.5.3	Creación de una tabla temporal con una SELECT	107
5.3.6	<i>INSERT</i>	111
5.3.7	<i>Bulk insert</i>	112
5.3.8	<i>UPDATE</i>	113
5.3.9	<i>DELETE</i>	113
5.3.9.1	TRUNCATE TABLE	114
Módulo 6	TRANSACT SQL. TRANSACCIONES	119
6.1	<i>TCL – Transaction Control Language</i>	119
6.2	<i>Elementos auxiliares: USE, variables, GO, EXECUTE, PRINT</i>	122
6.2.1	<i>Comando USE</i>	122
6.2.2	<i>Variables</i>	122
6.2.2.1	Variables definidas por el usuario	122
6.2.2.2	Variables de sistema	123
6.2.3	<i>GO</i>	126
6.2.4	<i>EXECUTE</i>	127
6.2.5	<i>PRINT</i>	128
6.3	<i>Estructuras de control</i>	128
6.3.1	<i>IF...ELSE</i>	128
6.3.2	<i>CASE</i>	128
6.3.3	<i>WHILE</i>	130
6.3.4	<i>GOTO</i>	131
6.4	<i>Funciones y Procedimientos almacenados</i>	132
6.4.1	<i>Función</i>	133
6.4.1.1	Funciones definidas en el sistema:	134
6.4.2	<i>Procedimiento</i>	135
6.4.2.1	Procedimientos definidos en el sistema:	138
6.5	<i>Cursos de Transact-SQL</i>	141
6.5.1	<i>Funciones FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE</i>	142



6.5.2	<i>Tipos de cursos</i>	145
6.6	Try/catch	148
6.7	Trigger/Disparador/Desencadenador	152
6.7.1	<i>Desencadenadores DML</i>	152
6.7.2	<i>Desencadenadores DDL</i>	158
6.7.3	<i>Desencadenadores logon</i>	161

Módulo 7 Seguridad en SQL Server 2008 164

7.1	Comprender los modos de seguridad	164
7.1.1.1	<i>Autentificación de Windows</i>	164
7.1.2	<i>Autentificación de SQL Server. Inicios de sesión</i>	165
7.2	Seguridad a nivel de base de datos. Usuarios de base de datos	167
7.3	Credenciales	174
7.4	Execute as	175

Módulo 8 Agente de SQL Server 176

8.1	Introducción	176
8.2	Propiedades del agente de SQL Server	177
8.3	Configuración de Operadores	179
8.4	Creación de Trabajos	179
8.4.1	<i>Categorías</i>	180
8.4.2	<i>Programación</i>	181
8.4.3	<i>Respuestas de trabajo</i>	181
8.5	Uso del Registro de Errores	187
8.6	Configuración de Alertas	188

Módulo 9 Copia de Seguridad, Restauración y Recuperación. Backup 191

9.1	Introducción	191
9.2	Tipos de copias de seguridad de datos	191
9.2.1	<i>Copias de seguridad de base de datos, parciales y de archivos</i>	192
9.2.2	<i>Copia de seguridad el registro de transacciones</i>	193
9.3	Restricciones de las operaciones de copia de seguridad en SQL Server	193
9.4	Modelos de recuperación de bases de datos	194
9.4.1	<i>Modelo de recuperación simple</i>	194
9.4.2	<i>Modelo de recuperación completo</i>	196
9.4.3	<i>Modelo de recuperación registro masivo</i>	197
9.4.4	<i>Opciones de recuperación</i>	198
9.5	Resumen de la teoría de copias de seguridad	198
9.6	Copias de seguridad en Management Studio	202
9.6.1	<i>Copia seguridad Completa con Management Studio</i>	203
9.6.2	<i>Cómo realizar copias de seguridad de archivos y grupos de archivos de la base de datos</i>	204
9.6.3	<i>Cómo hacer una copia de seguridad del registro de transacciones</i>	205
9.6.4	<i>Cómo crear una copia de seguridad diferencial de base de datos</i>	205
9.7	Restauración de una copia de seguridad de base de datos	206

Módulo 10 Mantenimiento De La Base de Datos 207

10.1	Reducir una base de datos/archivos	207
10.1.1	<i>Reducir automáticamente:</i>	207
10.1.2	<i>Reducir bajo solicitud:</i>	207
10.1.3	<i>Reducir el registro de transacciones</i>	208
10.2	Configuración de superficie	209



10.3	Plan de mantenimiento.....	211
10.4	Monitor de actividad.....	215
10.5	Registro de transacciones.....	217
10.5.1	<i>Cómo detener el crecimiento inesperado del registro de transacciones de una base de datos de SQL Server.....</i>	219
10.6	Asistente para la optimización de la BD.....	221
10.7	Replicación de BD	222
10.7.1	<i>Componentes de la réplica.....</i>	223
10.7.2	<i>Generación de una réplica.....</i>	225
Módulo 11	Soluciones Ejercicios	230
Módulo 12	Errores	278
Módulo 13	Bibliografía	281
13.1	Webs	282

Módulo 2 INTRODUCCIÓN

2.1 *Introducción*

SQL nace de un lenguaje de cómputo llamado SEQUEL creado por IBM, que fue diseñado específicamente para consulta de base de datos.

SQL SERVER es el producto emblema de motor de base de datos de Microsoft que está generando una enorme cantidad de interés en el mercado. Microsoft se ha comprometido a invertir grandes cantidades de dinero en apoyo a la comercialización del producto y cuenta con que SQL SERVER se convierta en el principal motor de base de datos en la industria de cómputo para la plataforma WINDOWS XP aunque también puede operar en Windows 95, Windows 98, Vista...etc.

Microsoft SQL SERVER 2005 es el penúltimo lanzamiento de los productos de base de datos de Microsoft que aprovecha la sólida base establecida por SQL Server 6.5, 7.0 y 2000. Actualmente contamos en el mercado con la nueva versión de SQL Server 2008 R2.

2.2 *Características*

PORATABILIDAD

Las bases de datos pueden desarrollarse fácilmente ya sea en un equipo mainframe o una mini computadora, sin importar su sistema operativo. Cuenta con soporte de SUN, lo que permite una mayor comunicación entre servidores.

COMPATIBILIDAD

Los DBMS o SGBD (DataBase Management System, Sistema de Gestión de Bases de Datos) se pueden ejecutar ya sea en computadoras personales, microcomputadoras, mainframes y computadoras con procesamiento paralelo masivo así como en distintas arquitecturas de hardware y software sin tener la necesidad de cambiar una sola línea de código. El optimizador de consultas soporta paralelismo entre consultas lo que implica la capacidad de procesar una sola consulta en múltiples CPU. Primera base de datos que ofrece una compatibilidad de código del 100%. Primera base de datos con los servicios de transformación de datos integrados (DTS: Data Transformation Service). Con DTS se pueden importar y exportar datos entre varias fuentes de datos heterogéneas y destinos de datos, es decir, transferir y transformar los datos automáticamente.

CONECTIVIDAD

Los SGBD pueden trabajar con información almacenada con otros sistemas de ba-



ses de datos así como también almacenar los datos y acceder a ellos desde otros paquetes de software.

Es la primera base de datos que ofrece la administración multiservidor para un gran número de servidores. Esto se conoce como acceso universal a los datos (Universal Data Access), la estrategia de Microsoft para permitir el acceso de alto rendimiento a una gran cantidad de fuentes de información.

SEGURIDAD

Permite verificaciones de usuarios, mantener clientes diferentes en una misma base de datos y a señalar que a ciertos datos solo podrán acceder determinados usuarios así como la codificación de información privada. El Administrador puede programar permisos por tabla, columna o fila. Por medio del sistema operativo se permite la restricción de los movimientos que pudieran hacerse con los archivos, así como controlar los accesos con cuentas, a esto se le conoce como autenticación. La seguridad comprende en protección y codificación de tablas de datos, columnas y filas, así como las transferencias de datos entre un cliente y un servidor como auditorias que identifican violaciones a la seguridad.

ADMINISTRACIÓN

Aporta funcionalidades de administración y tuning de la memoria, del CPU y de disco, de manera que se reduce el tiempo para la administración. Primera base de datos que soporta la configuración automática y la auto-optimización por medio del servicio Agente SQL Server.

RENDIMIENTO

Permiten una alta disponibilidad de aplicaciones sin necesidad de una reconfiguración de datos.

HERRAMIENTAS DE DESARROLLO

Funcionan con un amplio conjunto de herramientas de desarrollo, herramientas de consulta para el usuario final, aplicaciones comerciales y herramientas de gestión de la información del ámbito corporativo. La estrategia Microsoft consiste en reducir el costo y la complejidad de almacenamiento de datos al tiempo que pone la tecnología al alcance de un mayor número de personas.

Innovaciones que se incluyen:

- Generación de informes y análisis corporativos hasta el modelado de datos y el soporte de la toma de decisiones.
- Generación de Microsoft Repository (Depósito de Microsoft), una infraestructura común para compartir la información.

REQUERIMIENTOS DEL SISTEMA

<http://msdn.microsoft.com/es-es/library/ms143506.aspx>

Resumiendo podríamos decir lo siguiente:

Hardware:

Básicamente diferenciamos la express de el resto.

SQL Server 2008 R2 Datacenter (64 bits) IA64

En la tabla siguiente se muestran los requisitos del sistema para SQL Server 2008 R2 Datacenter (64 bits) IA64.

Componente	Requisito
Procesador	<p>Tipo de procesador:</p> <ul style="list-style-type: none"> • Procesador Itanium o más rápido <p>Velocidad del procesador:</p> <ul style="list-style-type: none"> • Recomendado: 1,0 GHz o más
Sistema operativo	<p>Windows Server 2008 R2 de 64 bits Itanium</p> <p>Windows Server 2008 SP2 de 64 bits Itanium</p> <p>Windows Server 2003 SP2 de 64 bits Itanium Datacenter</p> <p>Windows Server 2003 SP2 de 64 bits Itanium Enterprise</p> <p>Windows Server 2003 R2 SP2 de 64 bits Itanium Datacenter</p> <p>Windows Server 2003 R2 SP2 de 64 bits Itanium Enterprise</p>
Memoria	<p>RAM:</p> <ul style="list-style-type: none"> • Mínimo: 1 GB • Recomendado: 4 GB o más



SQL Server 2008 R2 Express with Tools (64 bits) x64

En la tabla siguiente se muestran los requisitos del sistema para SQL Server 2008 R2 Express con herramientas (64 bits) x64.

Componente	Requisito
Procesador	<p>Tipo de procesador:</p> <ul style="list-style-type: none"> ◦ Mínimo: AMD Opteron, AMD Athlon 64, Intel Xeon compatible con Intel EM64T, Intel Pentium IV compatible con EM64T <p>Velocidad del procesador:</p> <ul style="list-style-type: none"> ◦ Mínimo: 1,4 GHz ◦ Recomendado: 2,0 GHz o más
Sistema operativo	<p>Windows Server 2003 x64</p> <p>Windows Server 2003 SP2 de 64 bits x64 Datacenter</p> <p>Windows Server 2003 SP2 de 64 bits x64 Enterprise</p> <p>Windows Server 2003 SP2 de 64 bits x64 Standard</p> <p>Windows Server 2003 R2 SP2 de 64 bits x64 Datacenter</p> <p>Windows Server 2003 R2 SP2 de 64 bits x64 Enterprise</p> <p>Windows Server 2003 R2 SP2 de 64 bits x64 Standard</p> <p>Windows Vista SP2 Ultimate x64</p> <p>Windows Vista SP2 Home Premium x64</p>

Víctor José Vergel

	Windows Server 2008 SP2 de 64 bits x64 Standard sin Hyper-V Windows Server 2008 SP2 de 64 bits x64 Web Windows Server 2008 SP2 de 64 bits x64 Foundation Server Windows 7 x64 Ultimate Windows 7 x64 Home Premium Windows 7 x64 Home Basic Windows 7 x64 Enterprise Windows 7 x64 Professional Windows Server 2008 R2 de 64 bits x64 Datacenter Windows Server 2008 R2 de 64 bits x64 Enterprise Windows Server 2008 R2 de 64 bits x64 Standard Windows Server 2008 R2 de 64 bits x64 Web Windows Server 2008 R2 x64 para Windows Essential Server Solutions Windows Server 2008 R2 de 64 bits x64 Foundation Server
Memoria	RAM: <ul style="list-style-type: none"> • Mínimo: 512 MB • Recomendado: 1 GB • Máximo: 1 GB para Motor de base de datos.

Software:

- * .NET Framework 3.5 SP11
- * SQL Server Native Client (librerías SQL Server)
- * Archivos auxiliares para la instalación de SQL Server
- * Microsoft Windows Installer 4.5

2.3 Características de las diferentes versiones de sql server 2008

Término	Definición
SQL Server Developer (x86, x64 e IA64)	SQL Server Developer permite a los desarrolladores crear cualquier tipo de aplicación basada en SQL Server. Incluye toda la funcionalidad de SQL Server Datacenter, pero su uso está autorizado como sistema de desarrollo y pruebas, no como servidor de producción. SQL Server Developer es una opción ideal para las personas que crean y prueban aplicaciones. Puede actualizar SQL Server Developer para utilizarlo en producción.
SQL Server Workgroup (x86 y x64)	SQL Server Workgroup es ideal para ejecutar bases de datos ubicadas en sucursales y proporciona una administración de datos confiable y una plataforma de informes que incluye capacidades de sincronización y de administración seguras y remotas.
SQL Server Web (x86, x64)	SQL Server Web es una opción con un costo total de propiedad bajo para los hosts de web y los sitios web que proporciona capacidades de administración y escalabilidad para propiedades web, tanto de pequeña como de gran escala.
SQL Server Express (x86 y x64)	La plataforma de bases de datos de SQL Server Express se basa en SQL Server. Es también la sustitución de Microsoft Desktop Engine (MSDE). Gracias a su integración con Visual Studio, SQL Server Express facilita el desarrollo de aplicaciones controladas por datos que tienen una gran capacidad, ofrecen un almacenamiento seguro y se implementan con rapidez.
SQL Server Express with Tools (x86 y x64)	SQL Server Express es gratuito y los ISV pueden redistribuirlo (según su contrato). SQL Server Express es ideal para obtener información y crear pequeñas aplicaciones de servidor y de escritorio. Esta edición es la mejor opción para los fabricantes de software independientes, los desarrolladores no profesionales y los aficionados que crean aplicaciones cliente. Si necesita características de base de datos más avanzadas, SQL Server Express se puede actualizar sin problemas a versiones más sofisticadas de SQL Server.
SQL Server Express con Advanced Services (x86 y x64)	
Compact 3.5 SP1 (x86)	SQL Server Compact 3.5 es una base de datos gratuita e incrustada, ideal para crear aplicaciones independientes que se conectan ocasionalmente para dispositivos móviles, escritorios y clientes web en todas las plataformas de Windows.

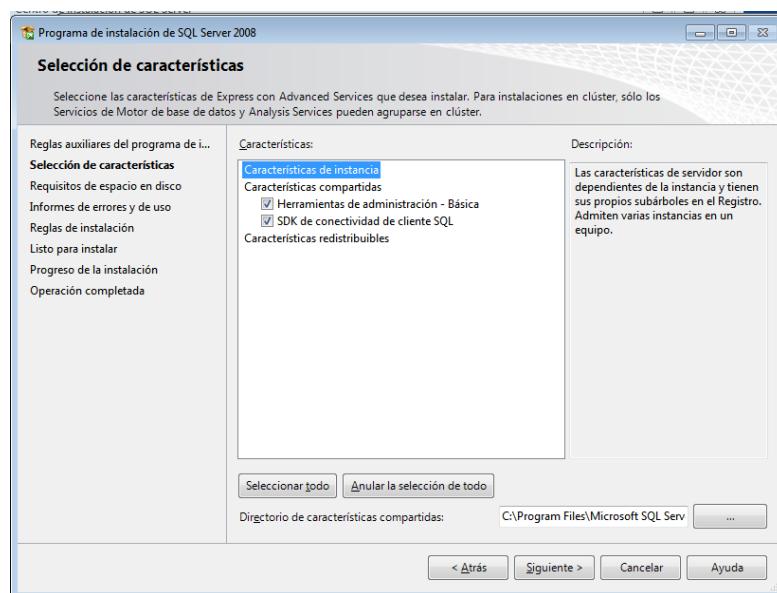
Versión	Precio
SQL Server 2008 Enterprise Edition	\$24,999.00
SQL Server 2008 Standard Edition	\$5,999.00
SQL Server 2008 Workgroup Edition	\$3,899.00
SQL Server 2008 Developer Edition	\$49.95

<http://msdn.microsoft.com/es-es/library/cc645993.aspx>

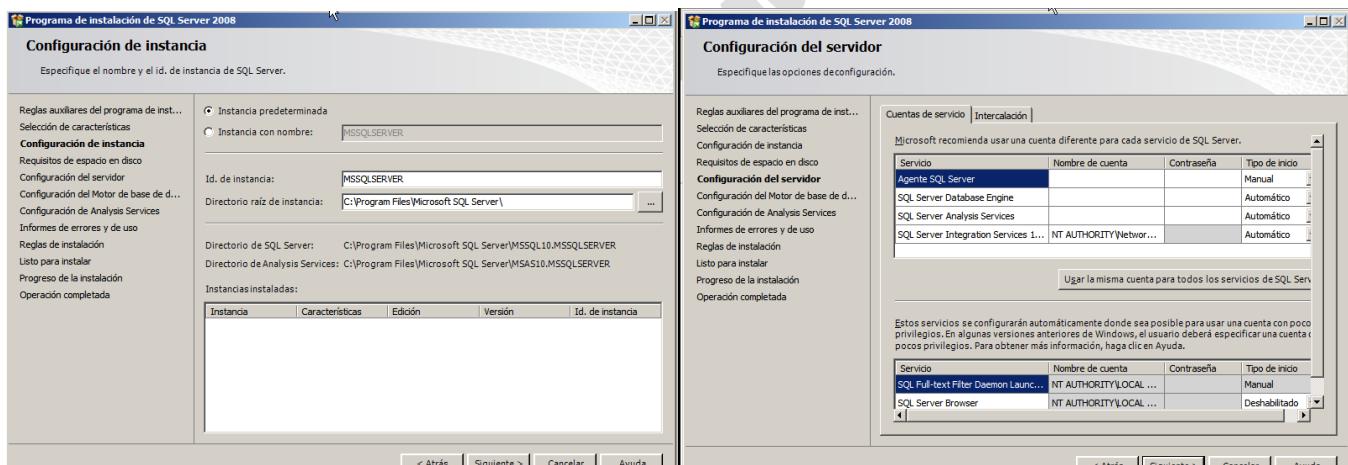
Nombre de la característica	Datacenter	Enterprise	Standard	Web	Workgroup	Express con Advanced Services	Express with Tools	Express
Número de CPU	Máximo sistema operativo	8	4	4	2	1	1	1
Máxima cantidad de memoria utilizada	Máximo sistema operativo	2 TB	64 GB	64 GB	4 GB	1 GB	1 GB	1 GB
Tamaño máximo de la base de datos	524 PB	524 PB	524 PB	524 PB	524 PB	10 GB	10 GB	10 GB

2.4 Instalación SQL Server 2008

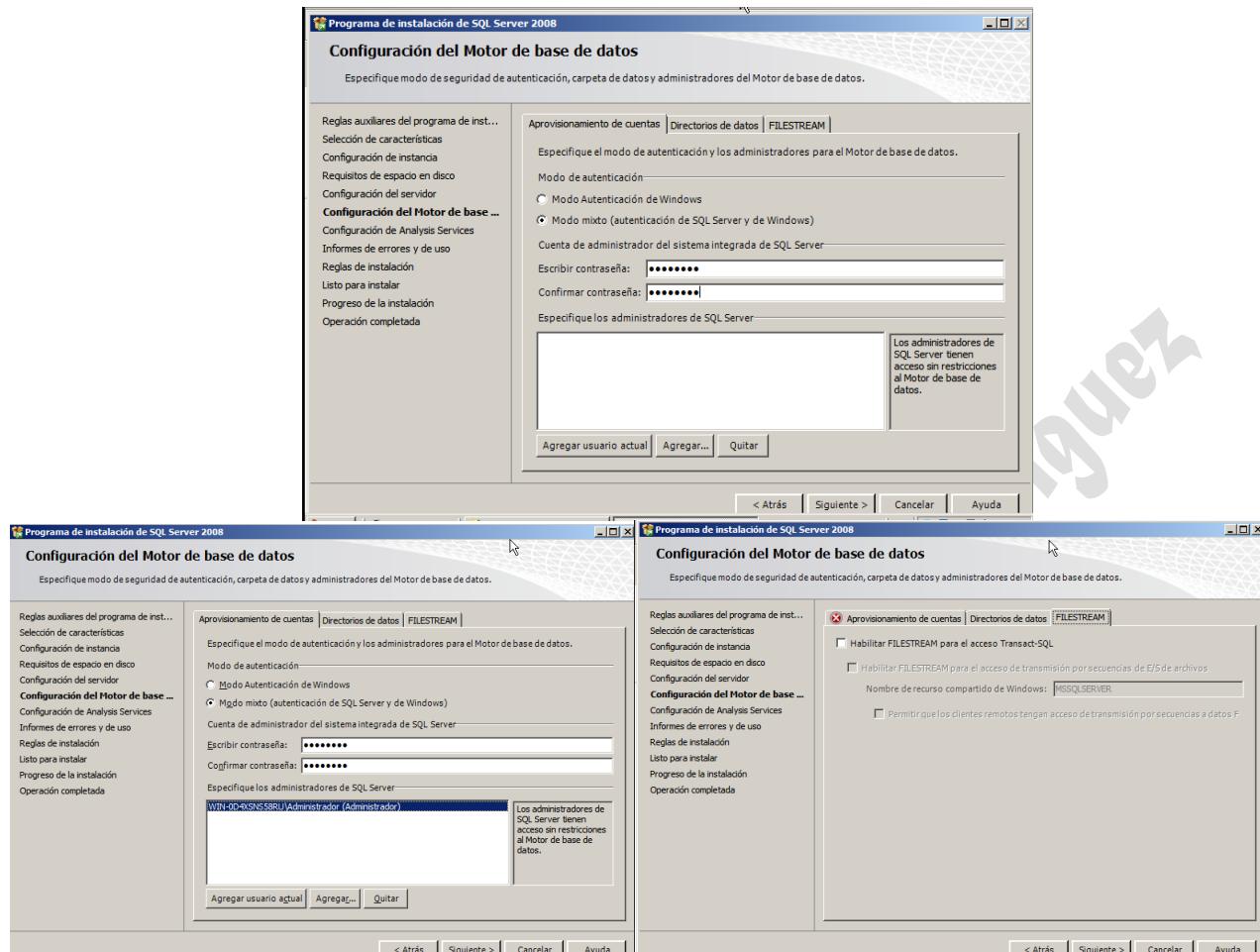
Básicamente con Windows 7 en el cliente necesitamos instalar .netframework 3.5 y el sqlserver management studio.



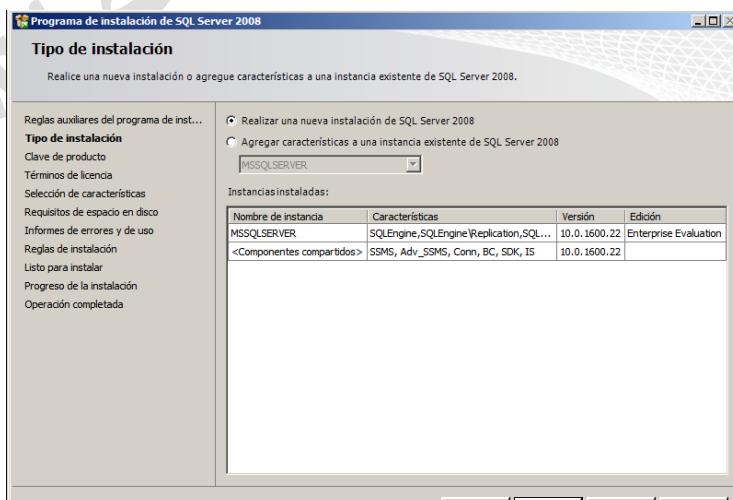
De lado del servidor:



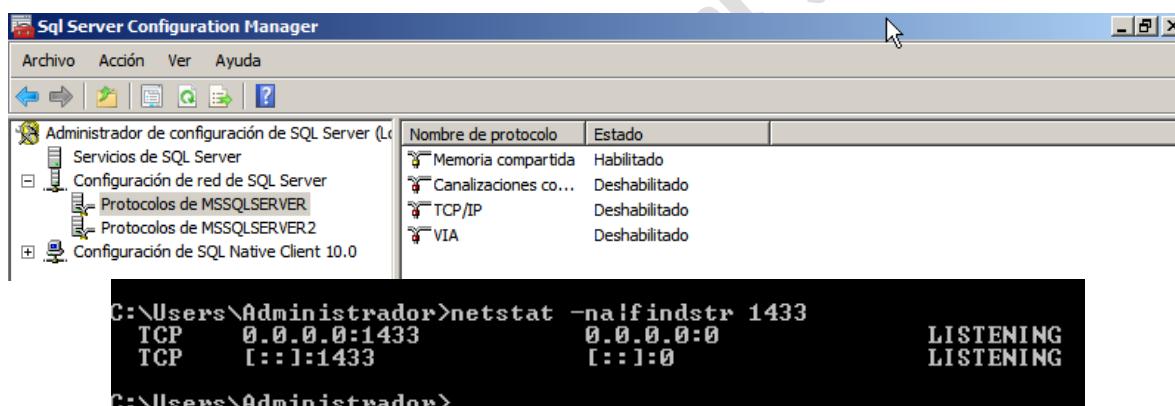
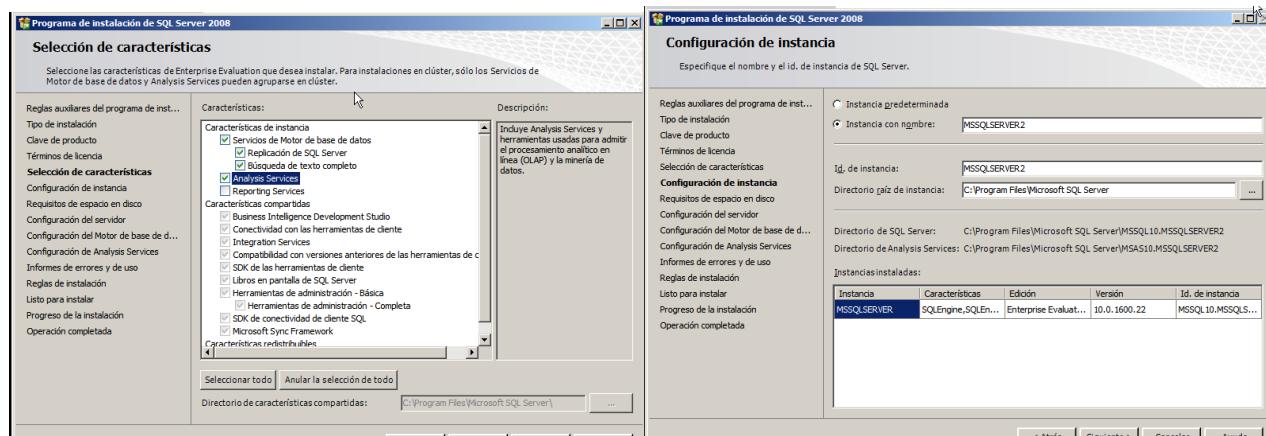
Diferencia entre ponerlo servicio de red y SYSTEM



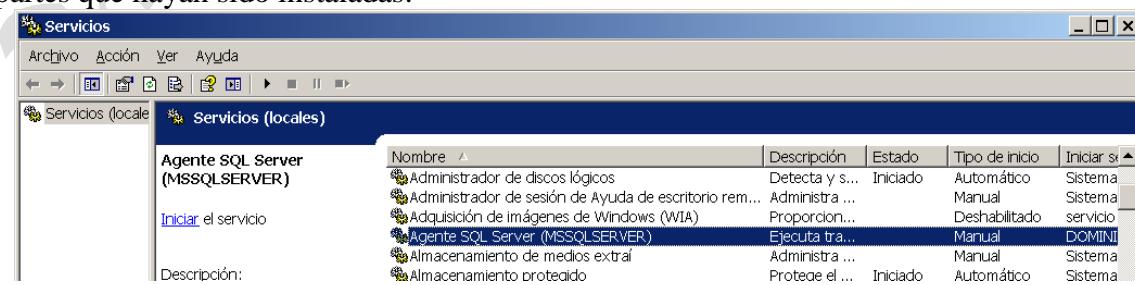
Nueva instalación:



Algunas opciones vienen ya activadas (de la instalación anterior, componentes compartidos)



Cuando finalice la instalación se iniciarán varios Servicios nuevos dependiendo las partes que hayan sido instaladas:



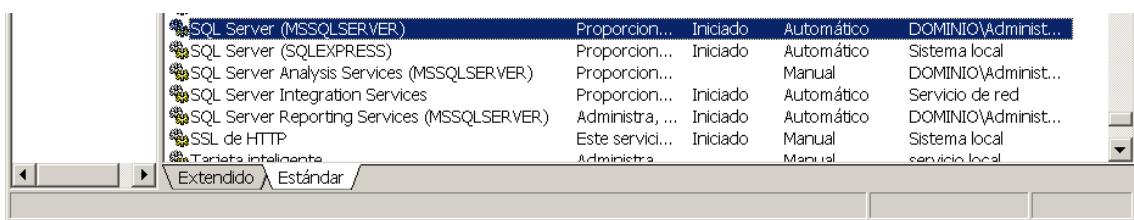


Imagen 1. Servicios instalados de SQL Server 2005 con múltiples instancias

SQL Server será el servicio de la BD en sí y *SQL Server Agent* será el servicio que se encargará entre otras cosas de ejecutar trabajos programados, supervisar la actividad y enviar mensajes de alerta a los operadores configurados sobre el estado del servidor. El resto de servicios son “secundarios”, añaden funcionalidad a nuestro SGBD con Reporting Services, que en coordinación con el servidor web permitirá visualizar informes en tiempo real sobre los datos de la BD.

Para instalar en segundo servidor en un mismo equipo, otra instancia, habrá que modificar el número de puerto por el que queremos comunicarnos con la BD, en este caso 1434 es el número de puerto por defecto.



Una vez en la consola Management Studio nos conectaremos a ella haciendo uso del nombre de instancia que hayamos colocado en la instalación:

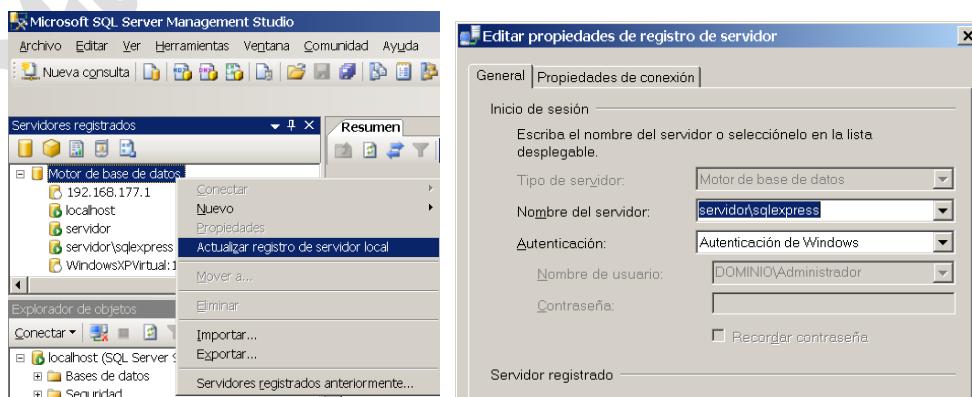


Imagen 2. Conexión a la segunda instancia

La herramienta de administración de SQL Server 2008 nos permitirá definir puertos de conexión con las diferentes instancias. Tener en cuenta que en la versión SQL Server Express los protocolos TCP/IP vienen deshabilitados, luego no se podrá acceder a ella hasta que no se habiliten. Tampoco viene habilitado las Canalizaciones con nombre. Esto último se trata de realizar una cadena de conexión diferente a un servidor de BD.

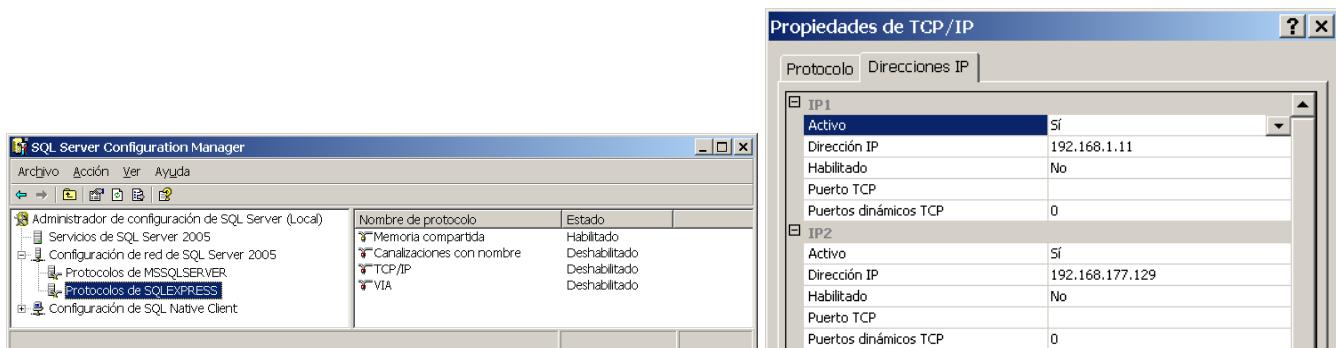


Imagen 3. Configuración de la segunda instancia instalada



Imagen 4. Propiedades de la canalización con nombre

2.5 Novedades

- Mejoras en la disponibilidad de datos.
- Mejoras de la capacidad de administración (motor de base de datos).
- Mejoras de programación (motor de base de datos). El almacenamiento **FILESTREAM** habilita a las aplicaciones de SQL Server para almacenar datos no estructurados, tales como documentos e imágenes, en el sistema de archivos. **Columnas dispersas** y conjuntos de columnas. Las columnas dispersas son columnas normales que disponen de un formato de almacenamiento optimizado para los valores NULL. Considere la posibilidad de usar este tipo de columnas si al menos del 20 al 40 por ciento de los valores de una columna serán NULL. Para obtener más información, vea Usar columnas dispersas. **Tablas anchas**. Las tablas anchas son tablas que contienen uno o más conjuntos de columnas. Una tabla ancha puede tener hasta 30000 columnas, 1000 índices y 30000 estadísticas. Para obtener más información, vea Tipos de tablas especiales.
- Mejoras de escalabilidad y rendimiento (motor de base de datos)
- Mejoras de seguridad (motor de base de datos). Nuevas funciones de cifrado. Tam-

bien, auditar. SQL Server Audit es una nueva característica de SQL Server 2008 que permite crear auditorías personalizadas de eventos Database Engine (Motor de base de datos). SQL Server Audit utiliza eventos extendidos para registrar la información para la auditoría y proporciona las herramientas y procesos necesarios para habilitar, almacenar y ver auditorías en diversos servidores y objetos de base de datos.

2.6 Componentes de SQL Server 2008

(reutilizado del 2005)

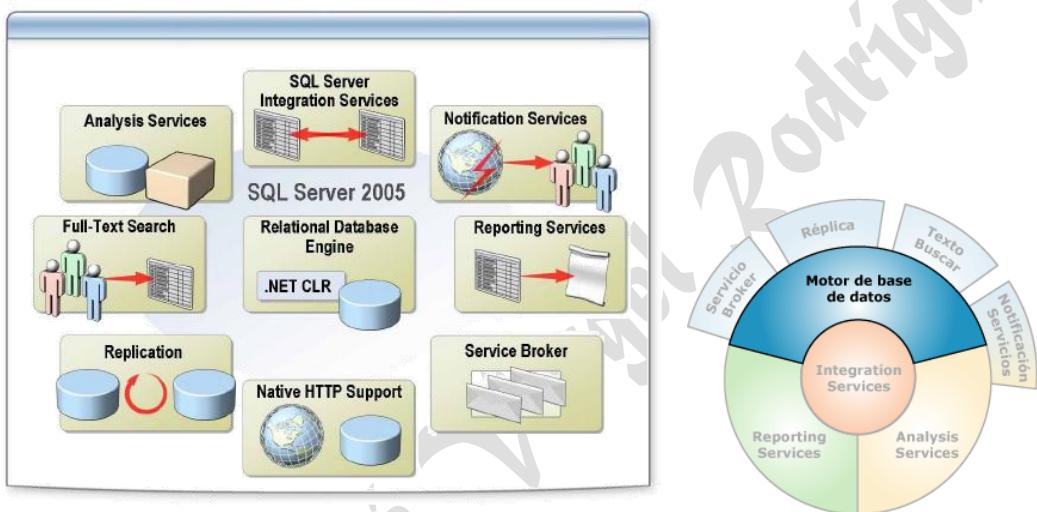


Imagen 5. Componentes de SQL Server 2008

- ✓ Relational database engine

El SQL Server 2008 Database Engine (Motor de base de datos de SQL Server 2008) de Microsoft es el servicio principal para almacenar, procesar y proteger datos. El Database Engine (Motor de base de datos) proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa. El Database Engine (Motor de base de datos) también proporciona compatibilidad completa para mantener una alta disponibilidad.

- ✓ Analysis Services

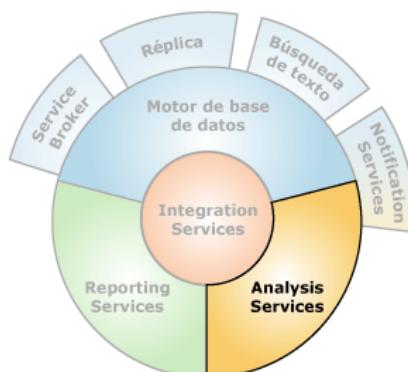
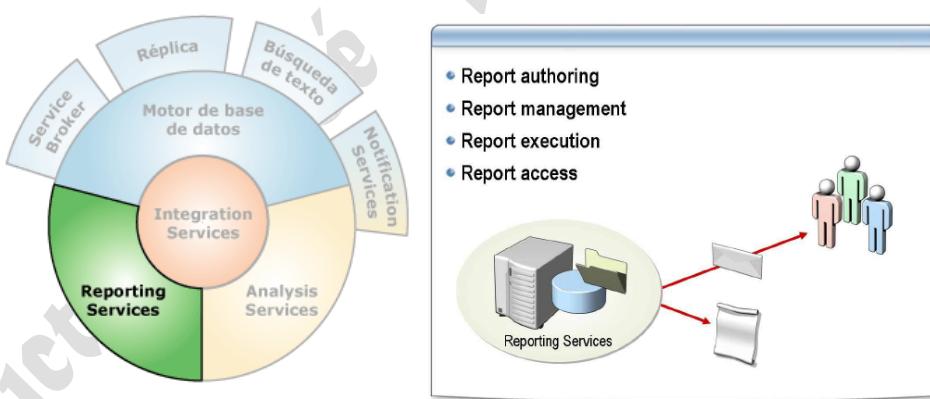


Imagen 6. Analysis Services

Microsoft SQL Server 2008 Analysis Services (SSAS) proporciona funciones de procesamiento analítico en línea (OLAP¹) y minería de datos² para aplicaciones de Business Intelligence. Analysis Services admite OLAP al permitirle diseñar, crear y administrar estructuras multidimensionales que contienen datos agregados desde otros orígenes de datos, por ejemplo bases de datos relacionales. Para las aplicaciones de minería de datos, Analysis Services le permite diseñar, crear y visualizar modelos de minería de datos, recopilados en otros orígenes de datos mediante una variedad de algoritmos de minería de datos estándar.

✓ **Reporting Services**



✓ **Imagen 7. Reporting Services**

¹ OLAP es una solución utilizada en el campo de la Inteligencia de Negocios (Business Intelligence), la cual consiste en consultas a estructuras multidimensionales (o Cubos OLAP) que contienen datos resumidos de grandes Bases de Datos o Sistemas Transaccionales (OLTP). Se usa en informes de negocios de ventas, marketing, informes de dirección, minería de datos y áreas similares. La razón de usar OLAP para las consultas es la velocidad de respuesta. Una base de datos relacional almacena entidades en tablas discretas si han sido normalizadas, esta estructura es buena en un sistema OLTP pero para las complejas consultas multitable es relativamente lenta. La principal característica que potencia OLAP, es que es lo más rápido a la hora de hacer selects, en contraposición con OLTP que es la mejor opción para INSERTS, UPDATES Y DELETES.

² Las bases de la minería de datos se encuentran en la inteligencia artificial y en el análisis estadístico y mediante los modelos extraídos utilizando técnicas de minería de datos se aborda la solución a problemas de predicción, clasificación y segmentación.

Microsoft SQL Server 2008 Reporting Services (SSRS) ofrece funcionalidad empresarial de informes habilitados para Web con el fin de poder crear informes que extraigan contenido a partir de una variedad de orígenes de datos, publicar informes con distintos formatos y administrar centralmente la seguridad y las suscripciones.

✓ **SQL Server Integration Services (SSIS)**

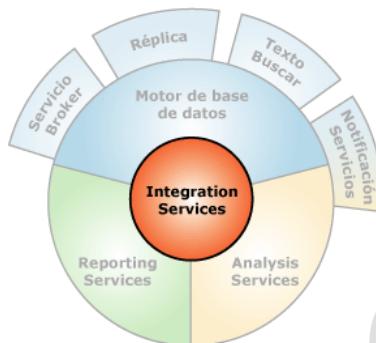


Imagen 8. Integration Services

Microsoft SQL Server 2008 Integration Services (SSIS) es una plataforma que permite generar soluciones de integración de datos de alto rendimiento, entre las que se incluyen paquetes de extracción, transformación y carga (ETL) para el almacenamiento de datos.

Integration Services incluye herramientas gráficas y asistentes para generar y depurar paquetes, tareas para realizar funciones de flujo de trabajo, como las operaciones de FTP, tareas para ejecutar instrucciones SQL o para enviar mensajes de correo electrónico, orígenes y destinos de datos para extraer y cargar datos, transformaciones para limpiar, agregar, mezclar y copiar datos, un servicio de administración, el servicio Integration Services para administrar Integration Services e interfaces de programación de aplicaciones (API) para programar el modelo de objetos de Integration Services.

✓ **Notification Services**

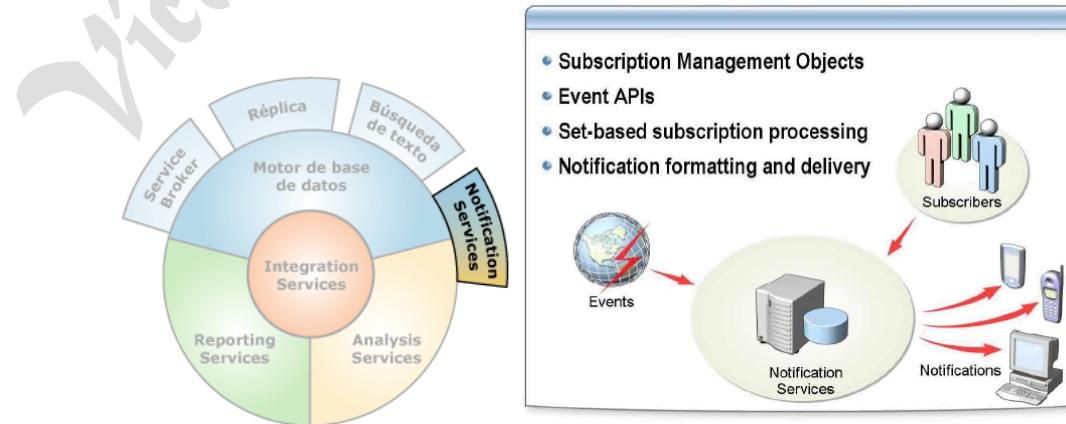


Imagen 9. Notification Services

Microsoft SQL Server Notification Services es una plataforma para desarrollar e implementar aplicaciones que generan y envían notificaciones a los suscriptores. Las notificaciones generadas son mensajes personalizados y oportunos que pueden enviarse a una amplia gama de dispositivos y que reflejan las preferencias del suscriptor.

Los suscriptores crean suscripciones para las aplicaciones de notificación. Una suscripción es un interés manifiesto en un tipo determinado de evento. Por ejemplo, las suscripciones pueden expresar preferencias como ésta: *"avisarme cuando el precio de los valores alcance los 70,00€"* o *"avisarme cuando se actualice el documento de estrategia que está redactando mi equipo"*.

Una notificación puede generarse y enviarse al suscriptor tan pronto como se produzca un evento que la desencadene. O bien, puede generarse y enviarse según una programación predeterminada especificada por el suscriptor.

Las notificaciones pueden enviarse a una gran variedad de dispositivos. Por ejemplo, una notificación puede enviarse al teléfono móvil de un suscriptor, a su asistente digital personal (PDA), a Microsoft Windows Messenger o a su cuenta de correo electrónico. Dado que el suscriptor suele llevar consigo estos dispositivos, las notificaciones son el medio ideal para enviar información importante.

- ✓ Service Broker

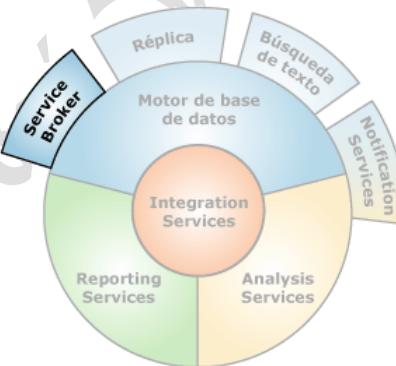


Imagen 10. Service Broker

Microsoft SQL Server 2008 Service Broker ayuda a los programadores a generar aplicaciones de base de datos seguras y escalables. Esta nueva tecnología, una parte de Database Engine (Motor de base de datos), proporciona una plataforma de comunicación basada en mensajes que permite a los componentes de aplicación independientes actuar como un conjunto de funcionamiento. Service Broker incluye infraestructura para la programación asíncrona que puede utilizarse para las aplicaciones de una base de datos única o de una sola instancia, así como para aplicaciones distribuidas.

- ✓ Réplica

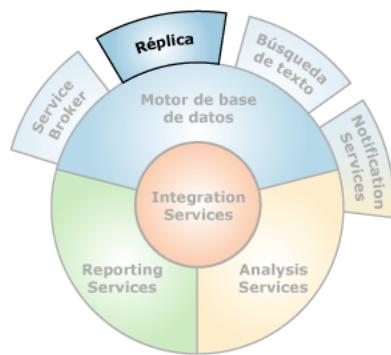


Imagen 11. Réplica

La réplica es un conjunto de tecnologías destinadas a la copia y distribución de datos y objetos de base de datos desde una base de datos a otra, para luego sincronizar ambas bases de datos y mantener su coherencia. La réplica permite distribuir datos entre diferentes ubicaciones y entre usuarios remotos o móviles mediante redes locales y de área extensa, conexiones de acceso telefónico, conexiones inalámbricas e Internet.

- ✓ Búsqueda de texto

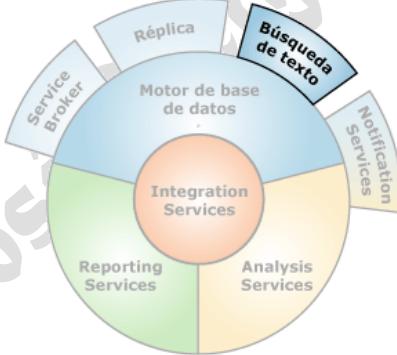


Imagen 12. Búsqueda de texto

Microsoft SQL Server 2008 contiene la funcionalidad necesaria para realizar consultas de texto en datos basados en caracteres sin formato contenidos en tablas de SQL Server. Las consultas de texto pueden contener palabras y frases, o formas diversas de una palabra o frase.

Módulo 3 INICIACIÓN A LA ADMINISTRACIÓN.

3.1 SQL Server Management Studio – Administrador corporativo

Esta herramienta es la más importante ya que agrupa toda la configuración del servidor y la funcionalidad de mantenimiento así como el diseño y mantenimiento de las bases de datos.



Imagen 13. Administrador corporativo

Inicialmente sólo administra un único servidor, que será el primero que se instaló en el ordenador local. Podremos también administrar servidores remotos y un servidor virtual, que corresponderán con servidores que estén en un cluster.

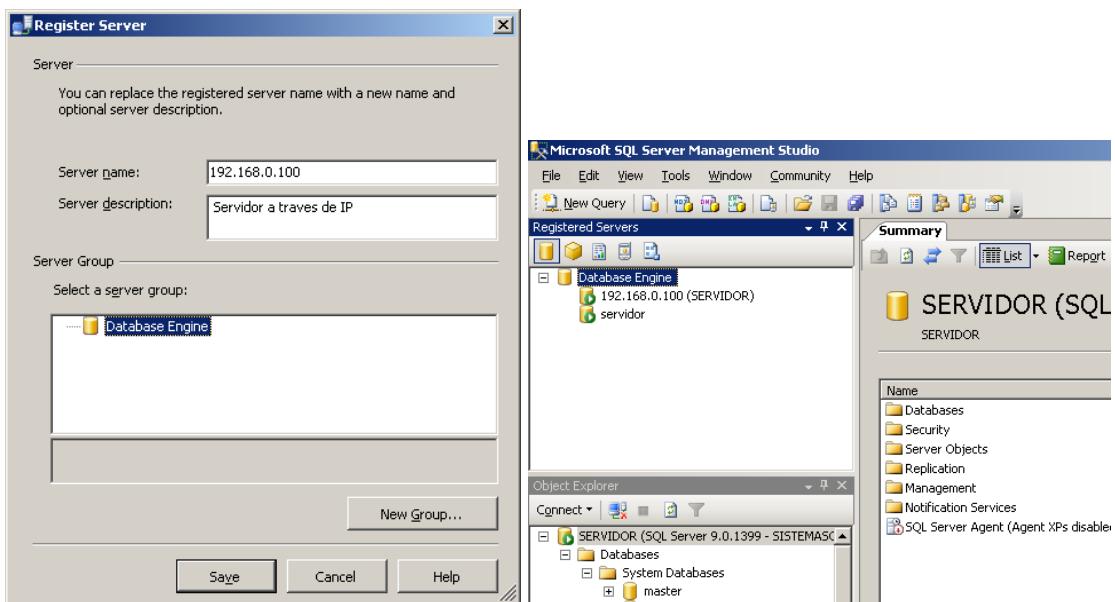
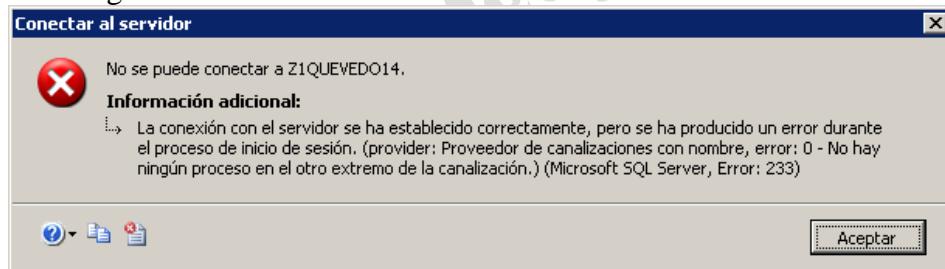


Imagen 14. Registro de un nuevo servidor y vista servidores registrados

Para conectarse a otro servidor de SQL Server 2008 debemos permitirlo previamente ya que por defecto viene denegado el acceso. El error que se produce si no se da permisos es el siguiente:



La manera de configurar será:

Inicio->Microsoft SQL Server 2005->Herramientas de configuración->Configuración de superficie

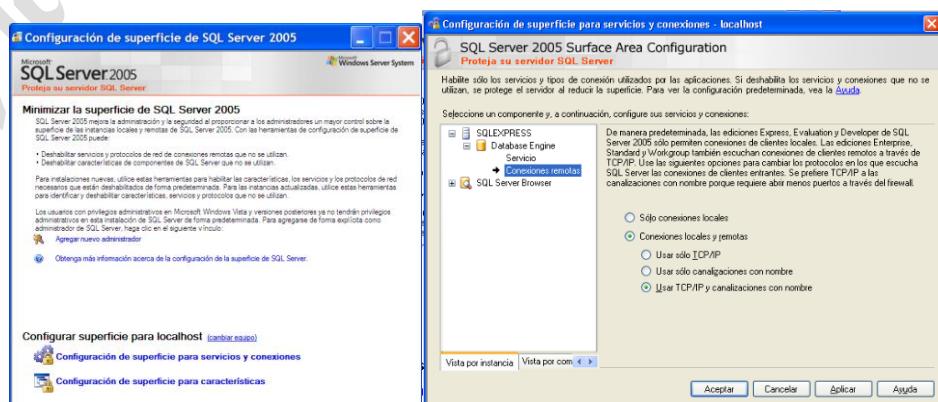


Imagen 15. Configurador de superficie de SQL Server 2005- TCP/IP y canalizaciones con nombre

También debemos en el SQL Configuration Manager > Configuración de red de SQLSERVER y habilitamos las conexiones TCP/IP y las canalizaciones con nombre y por último: Seguimos en el SQL Configuration Manager > Protocolos de SQL Server > TCP/IP y en las propiedades Direcciones IP > IPALL > Puerto TCP > poner el 1433 (o el puerto que deseas habilitar para SQL).

Por último únicamente tenemos que reiniciar el servicio de SQL Server configurado. A partir de ese momento aparecerá en el explorador de objetos este nuevo equipo. (Véase 0

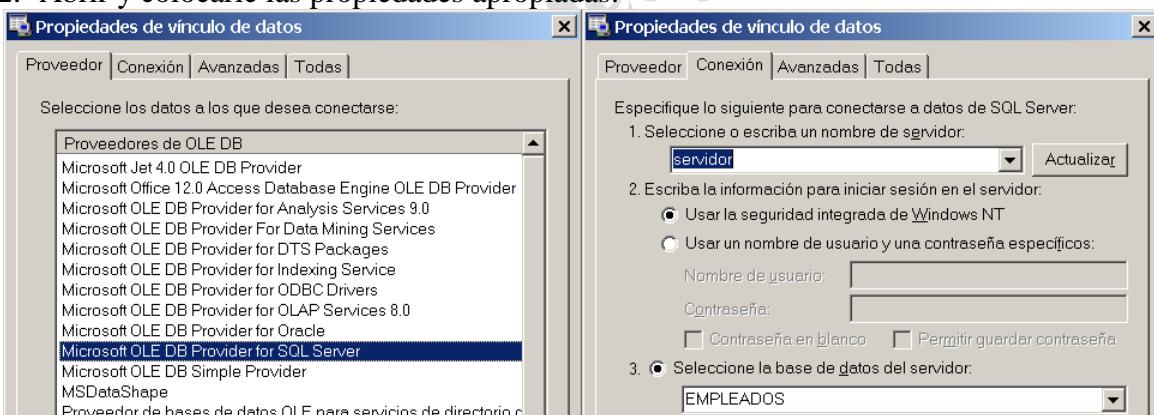
Configuración de superficie pág. 209)

3.1.1 Propiedades de la conexión actual

Las conexiones a la base de datos se realizarán con muchos fines, uno de ellos muy importante que será el acceso desde aplicaciones para la recuperación de datos. Para poder crear una cadena de conexión fácilmente y reutilizar en una aplicación seguiremos los siguientes pasos:

1.- Crear un fichero udl. 

2.- Abrir y colocarle las propiedades apropiadas:



3.- Posteriormente abrir el fichero udl con el bloc de notas:

```
Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=EMPLEADOS;Data Source=servidor
```

[Dalton et al, 2001c]

La conexión que se pueda establecer con el servidor de base de datos tendrá una serie de propiedades: En el menú de Herramientas – Opciones de la consola de SQL Server podemos ver las propiedades de conexión actuales. Las configuraciones disponibles son:

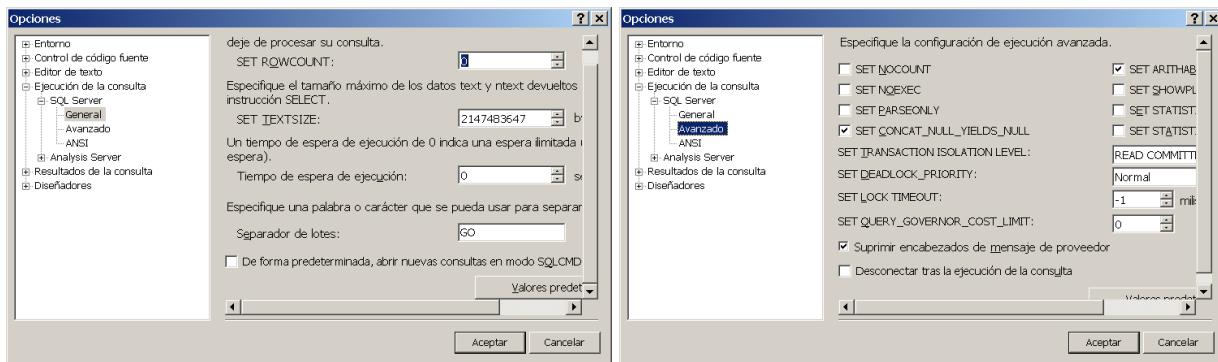


Imagen 16. Opciones de conexión

- **Set nocount.** Esta opción suprime los mensajes del contador de filas que aparecen al final de cada sentencia.

```
SELECT *
    FROM AUTHORS
    WHERE upper(au_lname) like 'W%'
```

Desactivado:

au_id	au_lname	au_fname	phone	ad
172-32-1176	White	Johnson	408 496-7223	109

(1 filas afectadas)

Activado:

au_id	au_lname	au_fname	phone	ad
172-32-1176	White	Johnson	408 496-7223	109

- **Set noexec.** Esta opción analiza y compila la consulta sin llegar a ejecutarla. Le permite validar la sintaxis y los objetos referenciados sin llegar a ejecutar la consulta.

```
SELECT *
    FROM AUTHORS
    WHERE upper(au_lname) like 'W%' – El campo au_lname no existe, sí au_lname
```

Activado:

Servidor: mensaje 207, nivel 16, estado 3, línea 1

El nombre de columna 'au_lname' no es válido.

Si el predicado estuviera bien escrito obtendríamos:

Comandos completados con éxito.

- **Set parseonly.** Esta opción analiza la consulta, pero ni la compila ni la ejecuta. No verifica los objetos referenciados en la consulta, es decir, que únicamente al ejecutar una consulta permitirá analizarla sintácticamente.

```
SELECT *
    FROM AUTHORS
    WHERE upper(au_lname) like 'W%' – El campo au_lname no existe, sí au_lname
```

Comandos completados con éxito.

No aparece ningún mensaje de error al no analizar los nombres de los campos.

- **Set concat_null_yields_null.** Esta opción hace que la consulta devuelva un valor NULL en una operación de concatenación si alguno de los operandos es un NULL.

```

update dbo.employee set minit=null
where lname='Brown'

Set concat_null_yields_null on

SELECT lname,minit,lname+minit  concatenados from dbo.employee
where lname like 'B%'

Set concat_null_yields_null off

SELECT lname,minit,lname+minit  concatenados from dbo.employee
where lname like 'B%'

SELECT lname,minit from dbo.employee
where minit is null

```

	Iname	minit	concatena...
1	Bennett		Bennett
2	Brown	NULL	NULL

Imagen 17. Resultados con concat_null_yields_null a on

- **Set arithabort.** Esta opción indica que la consulta debería finalizar si un desbordamiento aritmético o una división entre cero tiene lugar por error en la consulta.
- **Set showplan_text.** Esta opción proporciona una versión en texto del plan de consulta pero no ejecuta la consulta.

StmtText
1 SELECT * FROM titleauthor, authors WHERE titleauthor.au_Id=authors.au_id

StmtText
1 --Nested Loops(Inner Join, OUTER REFERENCES:([authors].[au_id]))
2 --Clustered Index Scan(OBJECT:([pubs].[dbo].[authors].[UPKCL_auidind]))
3 --Clustered Index Seek(OBJECT:([pubs].[dbo].[titleauthor].[UPKCL_taind]), SEEK:([titl

- **Set statistics time.** Esta opción proporciona un listado del tiempo en milisegundos necesario para analizar, compilar y ejecutar cada sentencia.

```

SELECT *
FROM titleauthor, authors
WHERE titleauthor.au_Id=authors.au_id

```

Tiempo de ejecución de SQL Server:

 Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

Tiempo de análisis y compilación de SQL Server:

 Tiempo de CPU = 0 ms., tiempo transcurrido = 0 ms.

- **Set statistics IO.** Esta opción proporciona una actividad de disco (lecturas, escritura y escaneados) por cada tabla referenciada en cada sentencia.

SELECT *

```
FROM titleauthor, authors
WHERE titleauthor.au_id=authors.au_id
```

Tabla 'titleauthor'. Número de exploraciones 23, lecturas lógicas 46, lecturas físicas 0, lecturas anticipadas 0.

Tabla 'authors'. Número de exploraciones 1, lecturas lógicas 2, lecturas físicas 0, lecturas anticipadas 0.

- **Set rowcount.** Esta opción especifica el número máximo de filas que puede devolver cada sentencia de la consulta.

Existe una instrucción *TOP n [PERCENT]* (Véase Ejemplo 17. Cláusula *TOP* y *WITH TIES* de la pág 85), que sirve para especificar el número de líneas devueltas por una consulta y sólo de una consulta, la diferencia con set rowcount es que éste último cambia el número de filas devueltas hasta que se vuelve a ejecutar la instrucción set rowcount con otro argumento.

- **Set ansi_defaults.** Esta opción activa o desactiva las opciones restantes en la lista como una única unidad. Si las otras opciones se configuran individualmente, esta operación aparece activada y sombreada.
- **Set ansi_nulls.** Esta opción hace que la consulta devuelva la respuesta UNKNOWN por cada valor NULL en una comparación. Si se desactiva esta opción, las comparaciones con valores NULL pasan a ser siempre TRUE.
- **Set ansi_null_dflt_on.** Esta opción anula los valores nulos predeterminados en las columnas nuevas.
- **Set ansi_padding.** Esta opción determina si los espacios en blanco en los extremos se eliminan o no automáticamente. Si se activa, los espacios en blanco en los extremos para datos VARCHAR y los números cero en los extremos para datos VARBINARY no se eliminan automáticamente.
- **Set ansi_warnings.** Esta opción hace que el servidor envíe mensajes de advertencia para aquellas condiciones que violan las reglas ANSI, pero no las de Transact-SQL.
- **Set cursor_close_on_commit.** Esta opción hace que se cierren los cursos cuando se realiza una transaction.
- **Set implicit_transactions.** Esta opción comienza transacciones sin que se usen explícitamente sentencias BEGIN TRANSACTIONS.
- **Set quoted_identifier.** Esta opción hace que los caracteres entre doble comilla sean interpretados como identificadores.

3.1.2 Registrar un servidor SQL Server vía IP pública:

Agregar un servidor de este tipo dependerá de la conexión que ese equipo tenga a internet. Inicialmente, si agregamos un servidor a través de la IP externa (IP pública) del servidor aparecerá el siguiente error:

Cuando no se conecta (no indicado en el router el puerto que deseamos abrir:



Imagen 18. Error de conexión con un servidor SQL Server remoto

Para que esto no suceda, necesitamos tener configurado el router para abrir los **puertos 1433 TCP y 1434 UDP**.

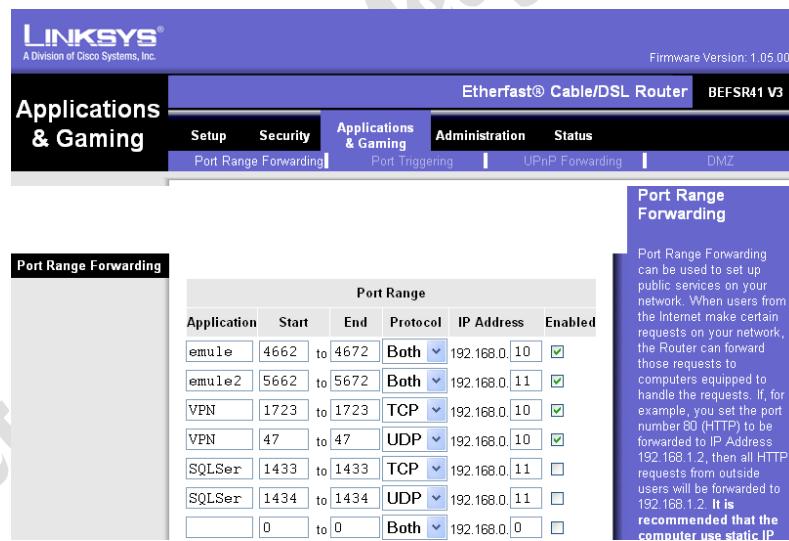
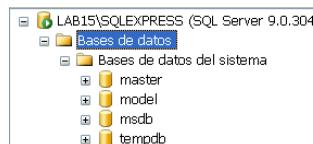


Imagen 19. Configuración del router

3.2 Creación de una BD.

Antes de comenzar a crear nuestras bases de datos aprenderemos cuales hay ya en el sistema.

3.2.1 BD del sistema



- **master**

Esta base de datos es fundamental para el funcionamiento de SQL Server. Almacena desde usuarios y contraseñas, servidores registrados hasta localización de determinados ficheros en el servidor.

```
select * from INFORMATION_SCHEMA.TABLES
select * from INFORMATION_SCHEMA.COLUMNS
SELECT * from sys.sysaltfiles
SELECT * from sys.sysdatabases
```

También posee procedimientos almacenados del sistema para saber por ejemplo quien está conectado al servidor, que columnas y de qué tipos tiene una determinada tabla, etc.

```
sys.sp_who
sys.sp_columns spt_values
```

Las bases de datos definidas por el usuario tienen más funciones particulares: como funciones del sistema-otras funciones, donde se encuentra SERVERPROPERTY()

```
SELECT SERVERPROPERTY('ServerName')
SELECT SERVERPROPERTY('Edition')
SELECT SERVERPROPERTY('ProductVersion')
SELECT SERVERPROPERTY('ProductLevel')
```

Resultados	
(Sin nombre de columna)	
1	SERVIDOR
(Sin nombre de columna)	
1	Enterprise Edition
(Sin nombre de columna)	
1	9.00.1399.06
(Sin nombre de columna)	
1	RTM

Similar a estos procedimientos están las funciones de sistema en cada base de datos: Funciones-Funciones del sistema-Funciones de configuración:

```
select @@servername
select @@version
```

Realmente estas son variables de sistema.

- **model**

En este caso se trata de una estructura básica que constituye el estado inicial de todas las BD que se vayan creando. Sería el concepto que tenemos de plantilla. También se podrá almacenar procedimientos/funciones que queremos que aparezcan en las nuevas BD a crear así como permisos a usuarios determinados.

- **msdb**

Posee un histórico de backups del servidor SQL Server 2005, datos de restauración así como información de administración necesaria para el servidor.

- Tempdb

Es utilizada por todos los usuarios del servidor para almacenar objetos temporalmente. tempdb se vuelve a crear cada vez que se inicia SQL Server, de forma que el sistema siempre se inicia con una copia limpia de la base de datos. Las tablas y los procedimientos almacenados temporales se quitan automáticamente en la desconexión y ninguna conexión permanece activa cuando se cierra el sistema. Por tanto, en la base de datos tempdb no hay nada que deba guardarse de una a otra sesión de SQL Server. (Véase Ejemplo 51. Creación de tablas temporales (#)de la pág. 107)

3.2.2 Creación de BD propias

Antes de crear una BD deberíamos por rutina hacer una copia de seguridad de la BD master puesto que el proceso modificará dicha BD (Véase Módulo 9 Copia de Seguridad, Restauración y Recuperación. Backup). Para crear la nueva base de datos necesitaremos un usuario con permisos para usar la base de datos *master* (ya que en esta BD se modificarán las vistas sys.sysaltfiles y sys.sysdatabases). Además debemos saber que al crear una nueva BD se creará a partir de una copia de la base de datos *model*.

	name	dbid	sid	mode	status	status2	crdate
1	BASE CENTROS	10	0x0105000000000005150000007CEB240DD...	0	1077936157	1090519040	2005-03-17 12
	name	filename					
2	BASE FINAL	9	0x0105000000000005150000007CEB240DD...	0	1077936157	1090519040	2005-03-16 12
3	Ejercicio 5 sp	7	0x0105000000000005150000007CEB240DD...	0	1077936157	1090519040	2005-03-12 18
4	Ejercicio 6 ciclismo	8	0x0105000000000005150000007CEB240DD...	0	1077936157	1090519040	2005-03-12 22
5	master	1	0x01	0	24	1090519040	2000-08-06 01
6	model	3	0x01	0	1073741840	1090519040	2000-08-06 01
7	msdb	4	0x01	0	24	1090519040	2000-08-06 01
8	Northwind	6	0x01	0	28	1090519040	2000-08-06 01
9	pubs	5	0x01	0	24	1090519040	2000-08-06 01
10	tempdb	2	0x01	0	8	1090519040	2005-03-24 14
	name	filename					
11	Northwind	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\northwnd.mdf					
12	Northwind_log	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\northwnd.ldf					
13	Ejercicio 5 sp_dat	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\Ejercicio 5 sp.mdf					
14	Ejercicio 5 sp_log	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\Ejercicio 5 sp.ldf					
15	Ejercicio 6 ciclismo_dat	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\Ejercicio 6 ciclismo					
16	Ejercicio 6 ciclismo_log	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\Ejercicio 6 ciclismo					
17	BASE FINAL_dat	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\BASE FINAL.mdf					
18	BASE FINAL_log	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\BASE FINAL.ldf					
19	BASE CENTROS_dat	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\BASE CENTROS.mdf					
20	BASE CENTROS_log	C:\Archivos de programa\Microsoft SQL Server\MSSQL\data\BASE CENTROS.ldf					

Imagen 20. Contenidos de la BD master, concretamente de sysdatabases y sysaltfiles

Si por ejemplo tenemos una tabla en la base de datos *model* dicha tabla aparecerá por defecto en todas las BD creadas:

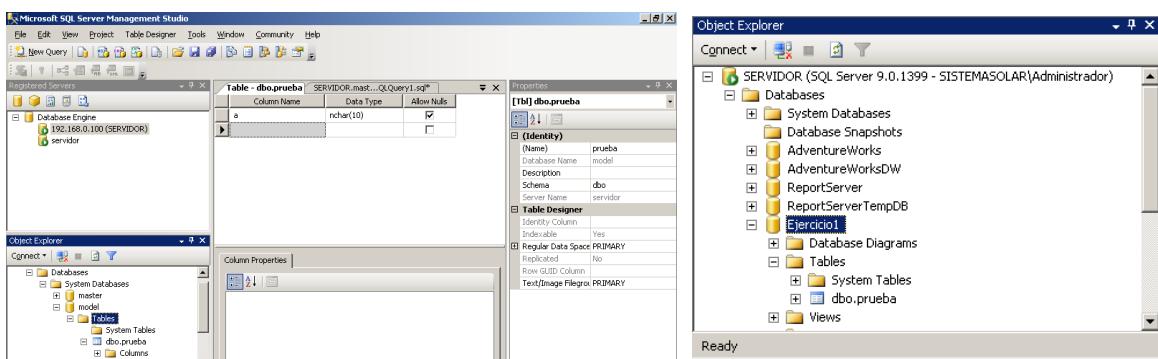


Imagen 21. Creación de una tabla en model y de un BD.

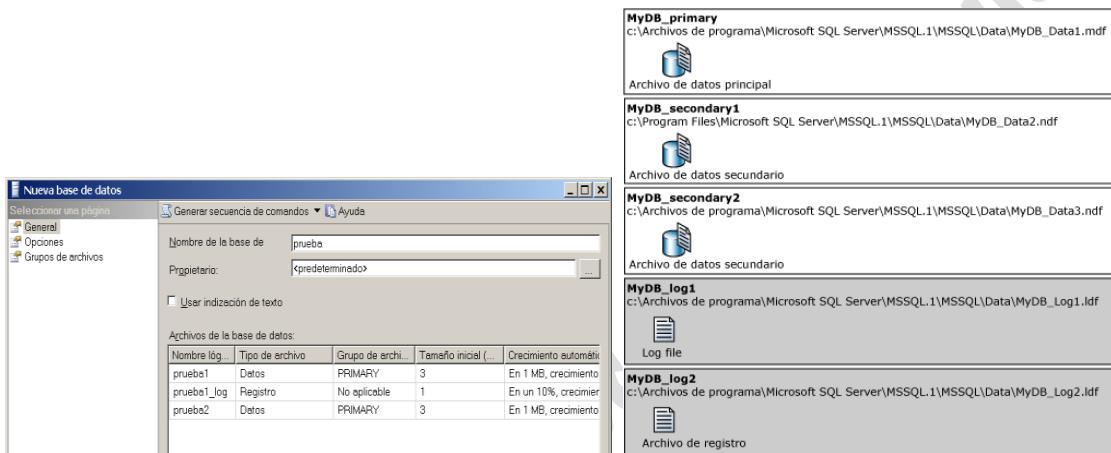


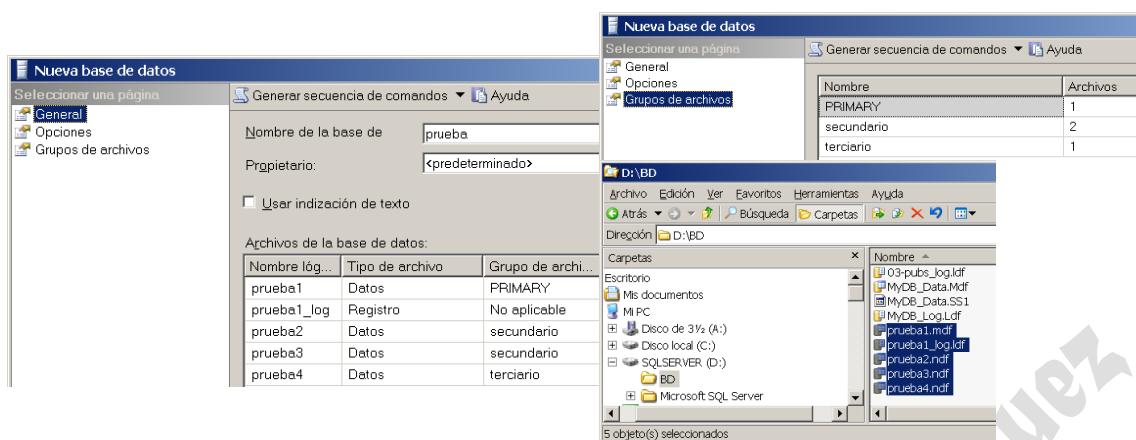
Imagen 22. Ejemplo de cómo poder distribuir en archivos una base de datos

A la hora de crear una BD el nombre de la misma podrá ser diferente al nombre del fichero que almacena los datos. Además constará inicialmente de dos ficheros uno mdf que almacenará los datos y otro ldf en los que existirá un registros de logs, pero podremos agregar de partida el número de archivos que queramos, dandonos la opción así de tener particionado en diferentes discos duros los datos. A partir del primer fichero de datos el resto tendrán la extensión *ndf*. El espacio que ocuparán estos ficheros podrá ir creciendo automáticamente y se indicará en el momento de creación de la BD, aún así existirá un tamaño máximo de 4GB. Por defecto la ruta de almacenamiento será:

C:\Archivos de programa\Microsoft SQL Server\MSSQL.1\MSSQL\Data

Ejemplo 1. Creación de una BD con diferentes archivos y grupos de archivos:

En el siguiente ejemplo se crea una base de datos con una contraseña de SQL Server. La base de datos tiene **un archivo de datos principal (mdf)**, **dos grupo de archivos definido por el usuario y el archivo de registro (ldf)**. El archivo de datos principal está en el grupo de archivos principal. El grupo de archivos definido por el usuario con el nombre secundario tiene dos archivos de datos secundarios (ndf), prueba2 y prueba3. El grupo de archivos definido por el usuario con el nombre terciario tiene el archivo de datos secundario prueba4.



En las opciones de creación de tabla una característica muy importante y que tiene que ver con los ficheros de log anteriormente citados es el Modelo de recuperación. Por defecto viene colocado a **Simple**, esta opción marca que el proceso de recuperación de la BD en caso de error es básico, es decir, se podrá recuperar la última copia de seguridad pero desde ese último backup hasta la producción del error los datos tratados serán perdidos. Otra valor posible para la recuperación será el de **Completa** utilizado en BD de información crítica. En este caso a través del fichero de logs de las transacciones y junto con el backup podríamos recuperar el estado de la BD a justo antes de la producción del error. Por último la opción de **registro masivo** se trata de una opción intermedia a las dos anteriores.

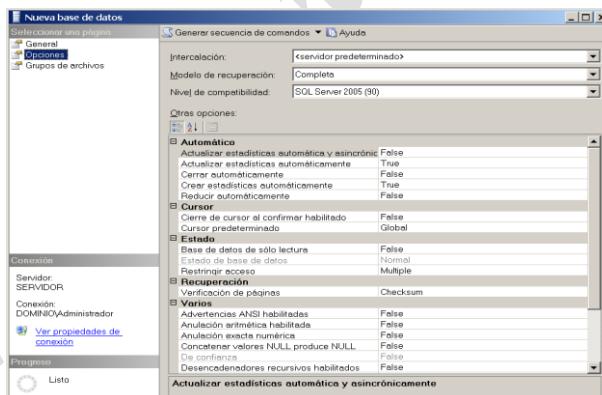


Imagen 23. Opciones de creación de la nueva BD

Alguna de las características más importantes son: posibilidad de que los ficheros disminuyan automáticamente si disminuye la cantidad de datos. (**Reducir automáticamente**). También posibilita que la BD sea de sólo lectura (**Base de datos de sólo lectura**), o que el acceso a la BD sea exclusivo (**Restringir acceso: Single**: Se utiliza en acciones de mantenimiento, sólo un usuario puede tener acceso a la base de datos, **Restricted**: sólo los miembros de las funciones db_owner, dbcreator o sysadmin pueden utilizar la base de datos).

3.3 Crear un diagrama de base de datos



Imagen 24. Creación de un diagrama de base de datos

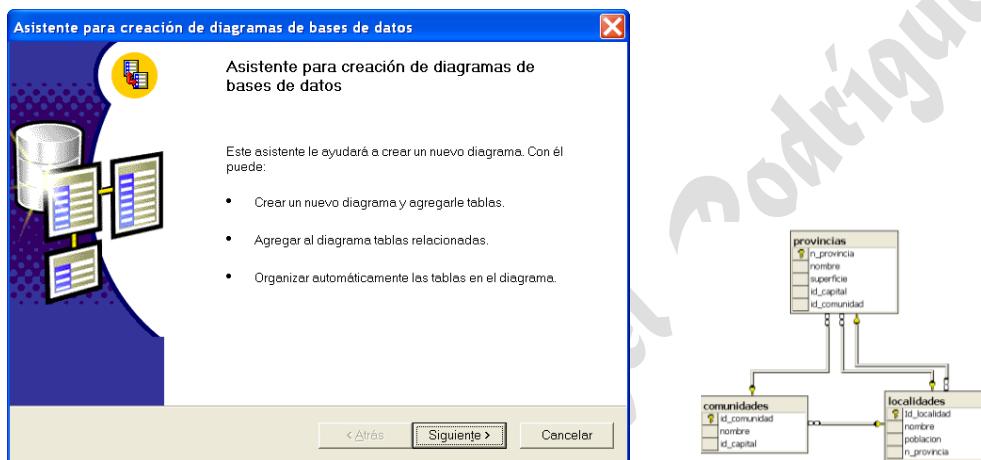
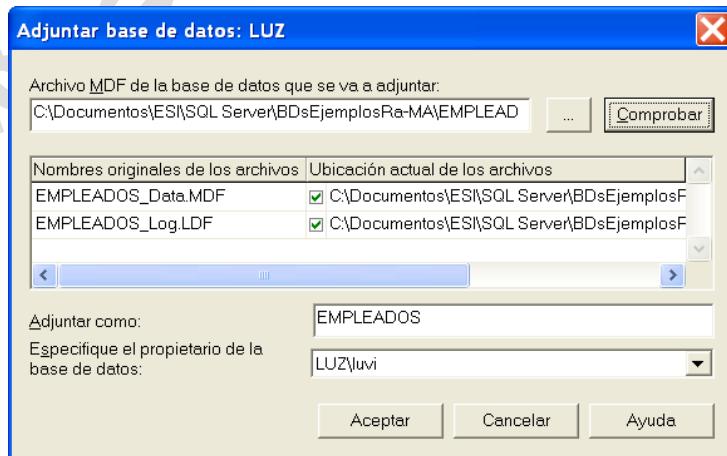


Imagen 25. Asistente para la creación de un diagrama sobre la bd

3.4 Adjuntar BDs

Añadimos ficheros mdf³ como bases de datos al administrador corporativo.



³ Una base de datos de SQL Server está compuesta por dos archivos, uno con extensión **MDB** que es la base de datos primaria y otro con extensión **LDF** que es el registro de transacciones. En algunos casos puede haber una extensión **NDF** que son archivos de datos secundarios, los cuales se utilizan para crear grupos de archivos y permiten colocar una parte de la base de datos en otro disco u otro servidor.

Imagen 26

3.5 Separar

Se separa del servidor la base de datos, desaparece por lo tanto del Administrador Corporativo el enlace a la base de datos.

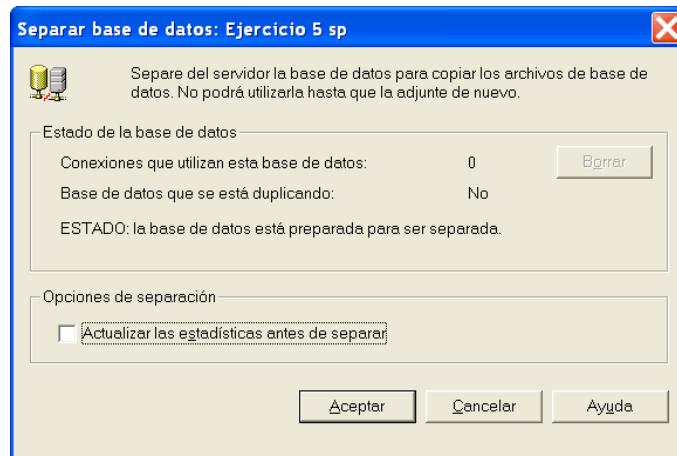
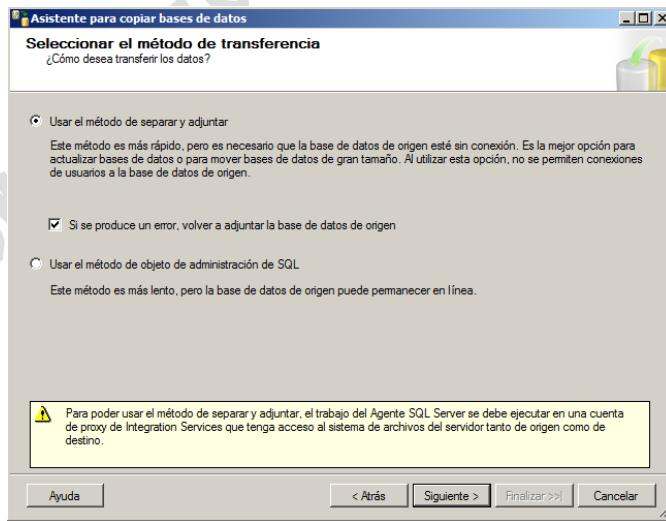
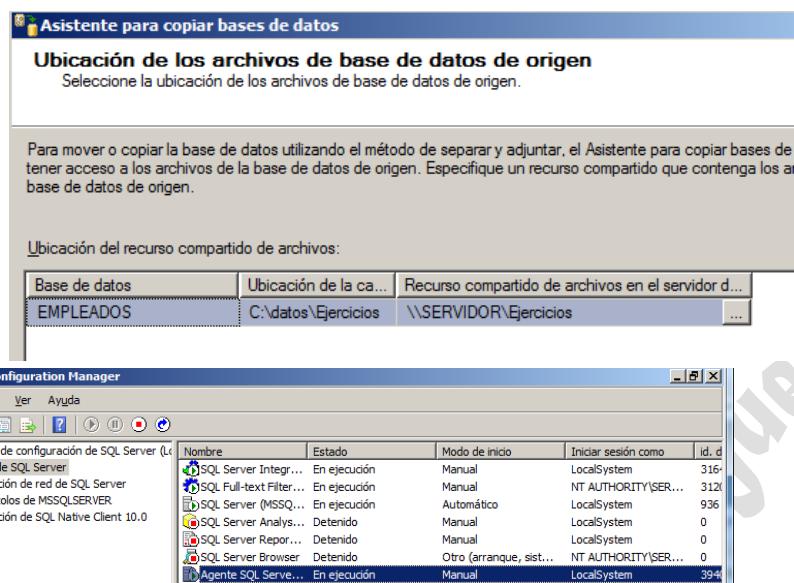


Imagen 27

3.6 Copiar BD

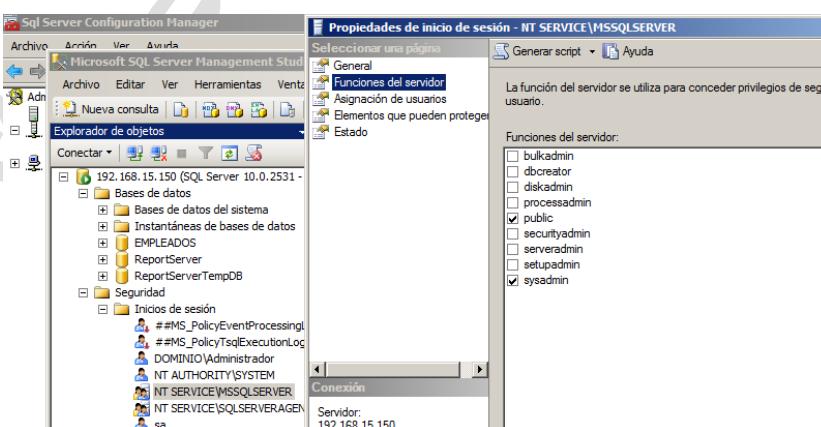


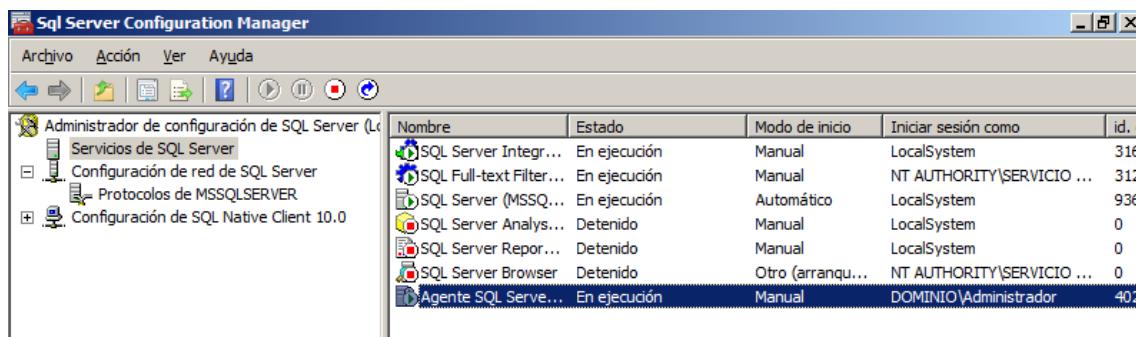
Si utilizas el primer método hay que tener muy en cuenta lo que informa abajo, hay que ejecutar el trabajo en una cuenta Integration Services, además de tener que compartir



La clave está en LocalSystem, al iniciar el servicio con este usuario en el servidor destino no se podrá copiar la BD puesto que no tiene permisos de sysadmin sobre la BD. El servicio se tendrá que iniciar con cualquier cuenta que pertenezca al grupo NT SERVICE\MSSQLSERVER que sí tiene el rol de sysadmin, por ejemplo el Administrador. Nota Microsoft.- Para realizar sus funciones, el **Agente SQL Server** debe configurarse de modo que utilice las credenciales de una cuenta que sea miembro del rol fijo de servidor sysadmin en SQL Server. La cuenta debe tener los siguientes permisos de Windows:

- * Adjust memory quotas for a process (Ajustar las cuotas de memoria de un proceso)
- * Act as part of the operating system (Actuar como parte del sistema operativo)
- * Bypass traverse checking (Omitir la comprobación transversal)
- * Log on as a batch job (Iniciar sesión como proceso por lotes)
- * Log on as a service (Iniciar sesión como servicio)
- * Replace a process level token (Reemplazar un símbolo de nivel de proceso)

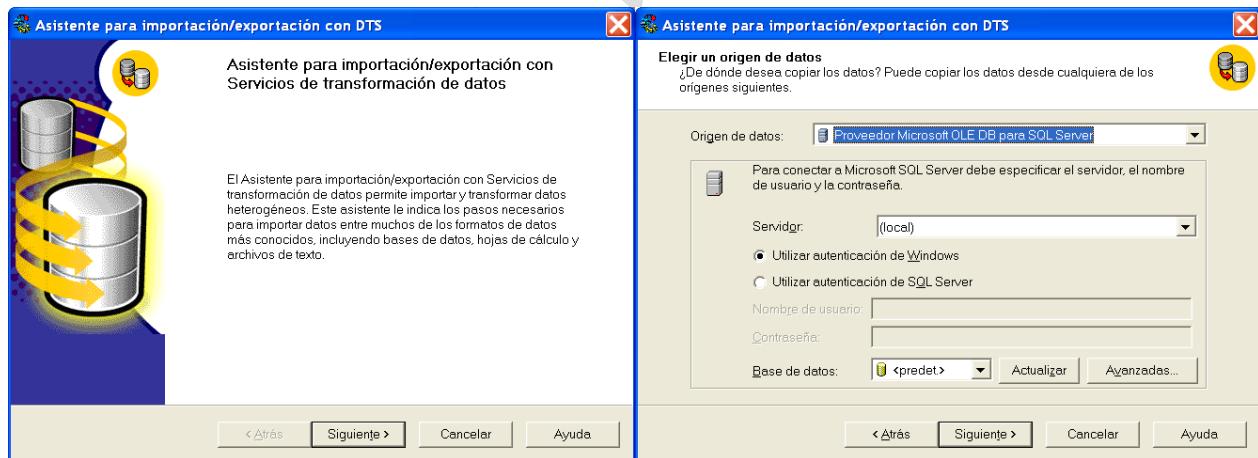




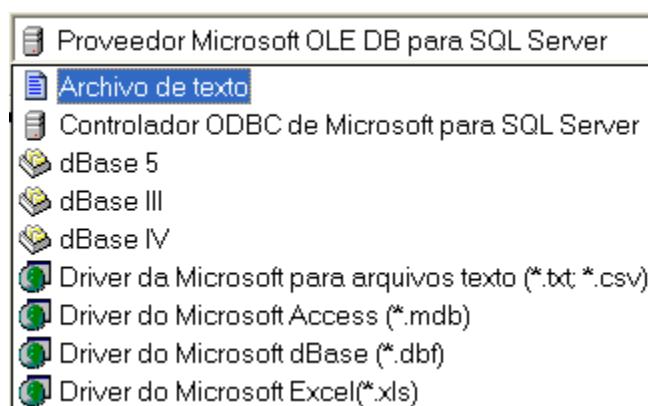
3.7 Importar / Exportar datos

Se utilizará el asistente que hay externo a SQL Server 2008, para pasar de por ejemplo access 2007.

Ejemplo 2. Importación de una base de datos Access.



Elegimos el tipo de fichero de origen de datos:



Por ejemplo Access. Especificamos lugar en el que se encuentra el fichero y finalmente si tuviera alguna contraseña. Los parámetros que se indican cambiarán para cada tipo de fichero que se importe.

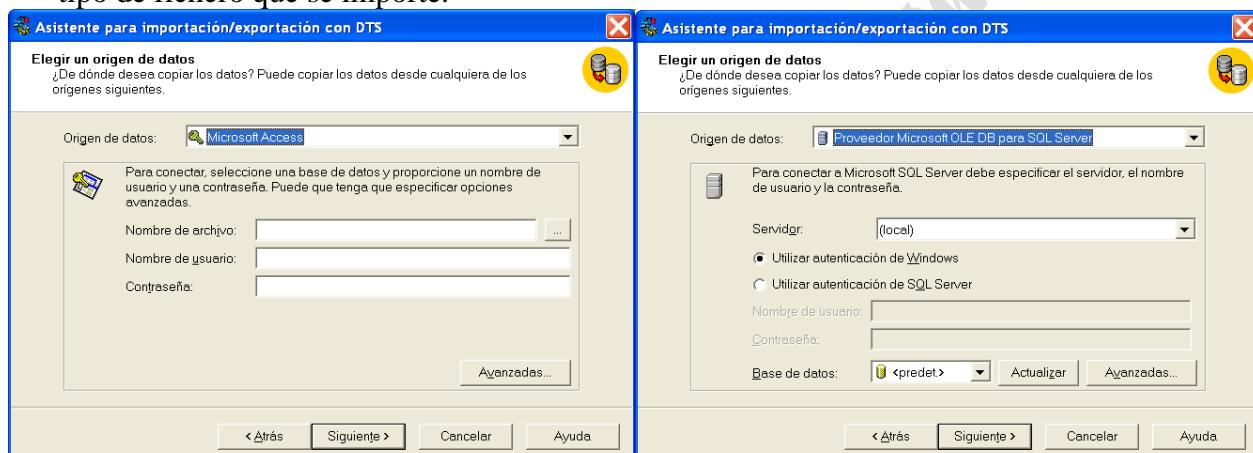
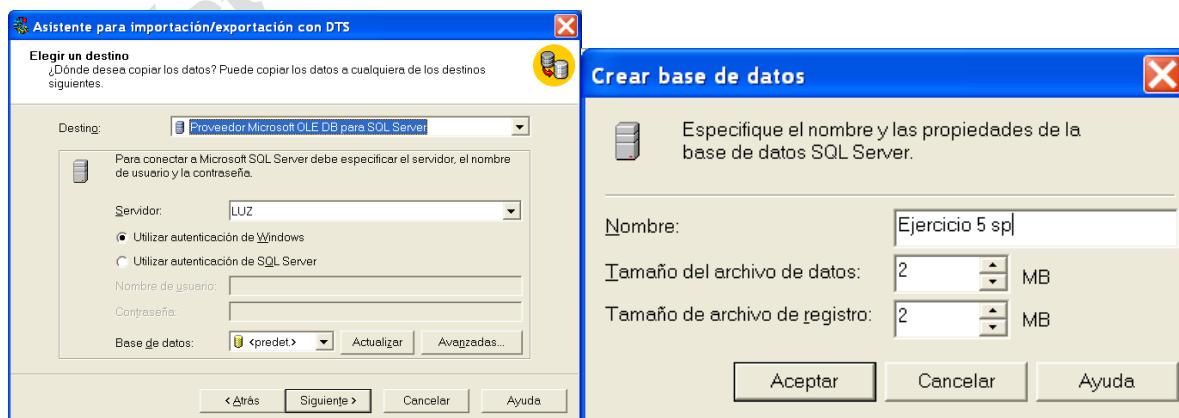
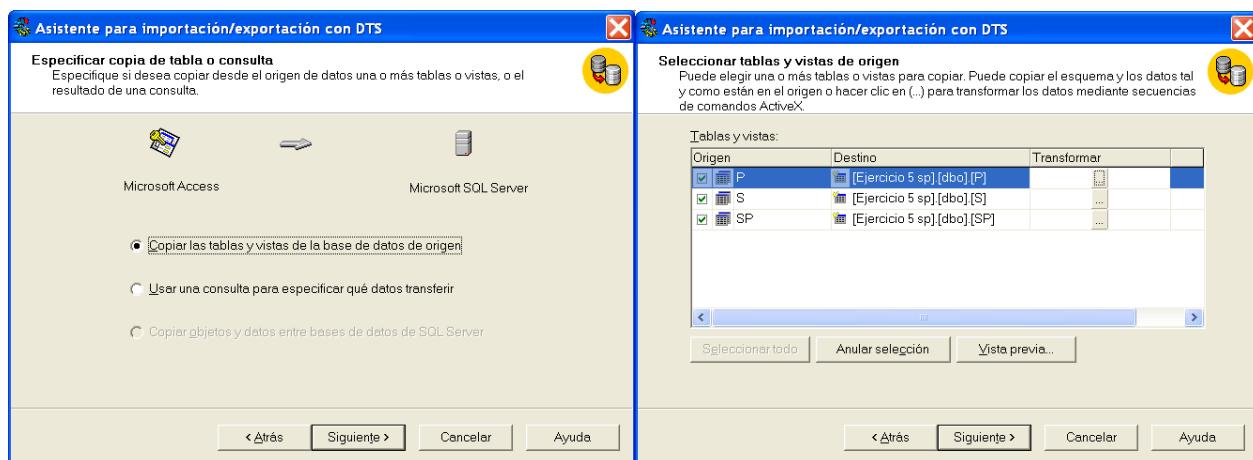


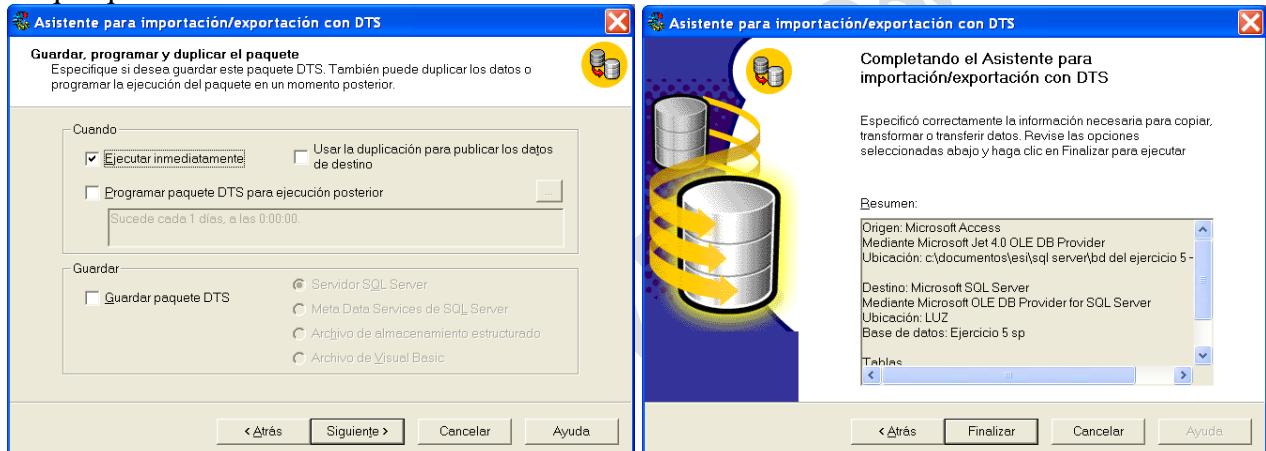
Imagen 28. Parámetros de configuración de un fichero Access o una BD de SQL Server.

Indicaremos ahora el lugar destino donde irán los datos importados, en este caso si elegimos una BD nueva aparecerá opciones de la creación de la nueva BD como nombre y tamaño.





Finalizamos el proceso con la confirmación de la importación de datos de las tablas que queramos.



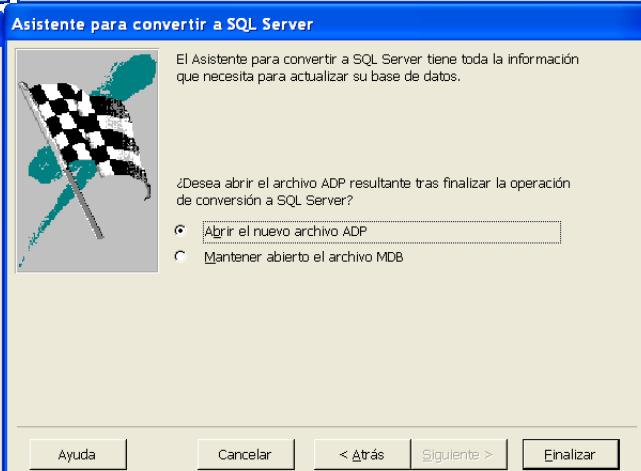
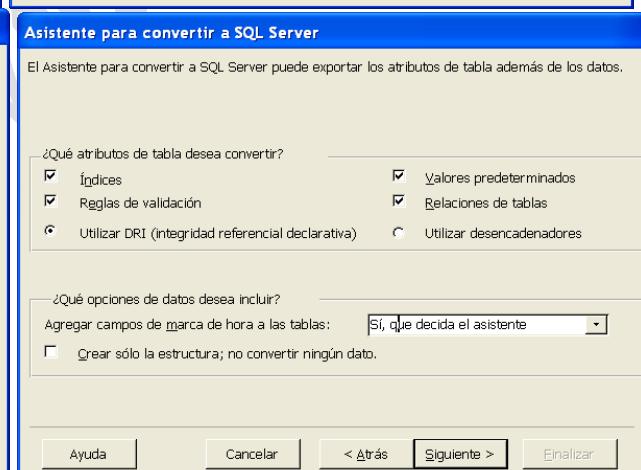
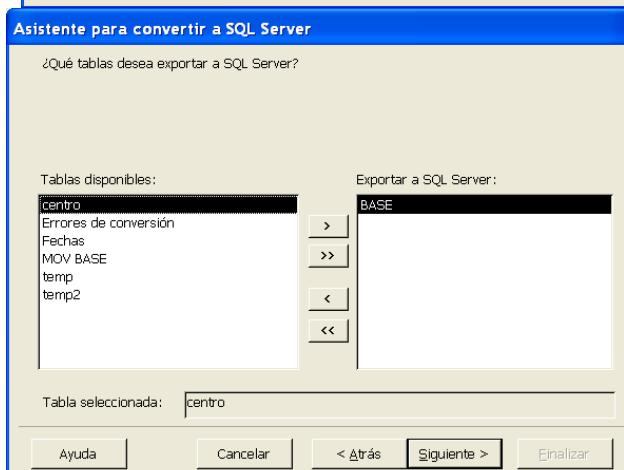
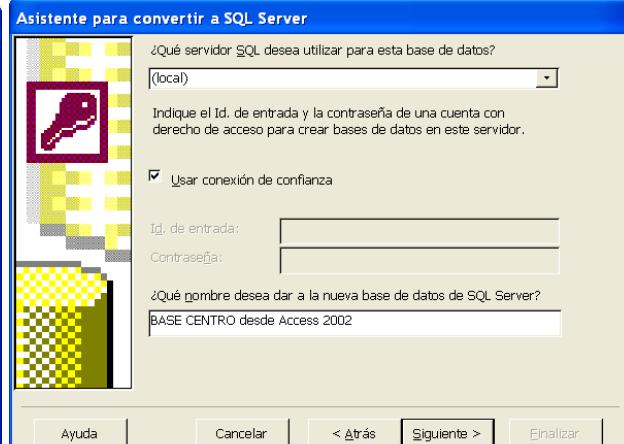
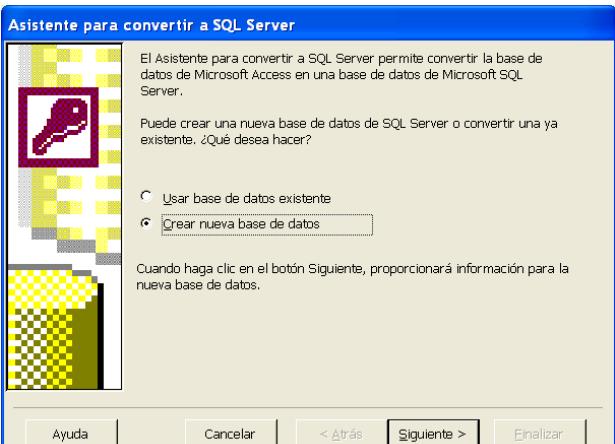
Nota.- Una exportación muy sencilla de datos que podemos generar después de ejecutar una consulta es pulsar botón derecho sobre los resultados y guardar como csv.



Imagen 29. Exportar datos de una query a un fichero csv

Ejemplo 3. Otra posibilidad de llevar datos a SQL Server. Exportación a SQL Server 2008 desde Access.

Una vez convertida la BD a Access 2002, ya tenemos las siguientes opciones en el menú Herramientas -> Utilidades de la base de datos

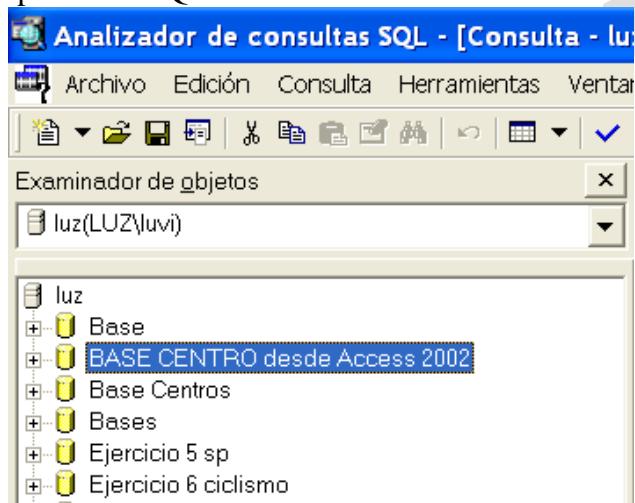




Al finalizar se genera un informe que muestra información sobre la operación realizada:

Asistente para convertir a SQL Server.pdf

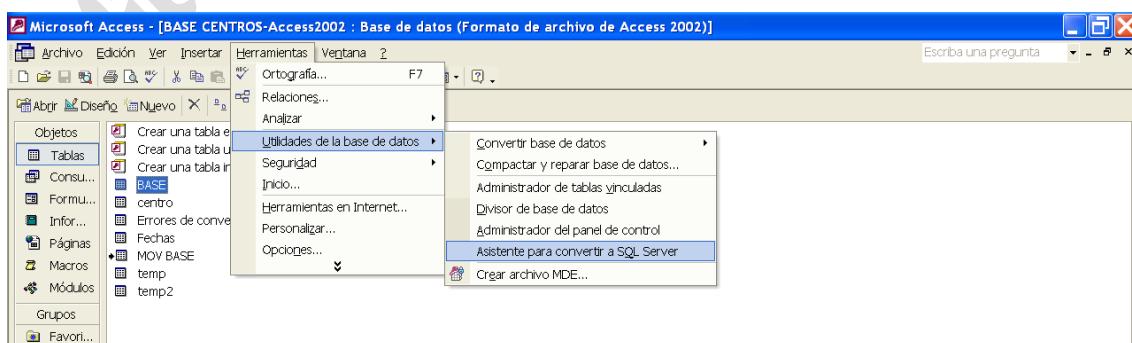
En este momento ya aparece en SQL Server nuestra BD:



Finalmente podemos observar que no hay datos en la tabla BASE de esta base de datos.

3.7.1 Importación de una base de access

Una vez convertida la BD a Access 2002, ya tenemos las siguientes opciones en el menú Herramientas -> Utilidades de la base de datos



Asistente para convertir a SQL Server

El Asistente para convertir a SQL Server permite convertir la base de datos de Microsoft Access en una base de datos de Microsoft SQL Server.

Puede crear una nueva base de datos de SQL Server o convertir una ya existente. ¿Qué desea hacer?

Usar base de datos existente
 Crear nueva base de datos

Cuando haga clic en el botón Siguiente, proporcionará información para la nueva base de datos.

Ayuda | Cancelar | < Atrás | Siguiente > | Finalizar

Asistente para convertir a SQL Server

¿Qué servidor SQL desea utilizar para esta base de datos?
 (local)

Indique el Id. de entrada y la contraseña de una cuenta con derecho de acceso para crear bases de datos en este servidor.

Usar conexión de confianza

Id. de entrada:
 Contraseña:

¿Qué nombre desea dar a la nueva base de datos de SQL Server?
 BASE CENTRO desde Access 2002

Ayuda | Cancelar | < Atrás | Siguiente > | Finalizar

Asistente para convertir a SQL Server

¿Qué tablas desea exportar a SQL Server?

Tablas disponibles:
 centro
 Errores de conversión
 Fechas
 MOV BASE
 temp
 temp2

Exportar a SQL Server:
 BASE

Tabla seleccionada: centro

Ayuda | Cancelar | < Atrás | Siguiente > | Finalizar

Asistente para convertir a SQL Server

El Asistente para convertir a SQL Server puede exportar los atributos de tabla además de los datos.

¿Qué atributos de tabla desea convertir?

Índices
 Reglas de validación
 Utilizar DRI (integridad referencial declarativa)
 Valores predeterminados
 Relaciones de tablas
 Utilizar desencadenadores

¿Qué opciones de datos desea incluir?

Agregar campos de marca de hora a las tablas: Sí, que decida el asistente
 Crear sólo la estructura; no convertir ningún dato.

Ayuda | Cancelar | < Atrás | Siguiente > | Finalizar

Asistente para convertir a SQL Server

El Asistente para convertir a SQL Server puede modificar la aplicación existente o crear una nueva para trabajar con una base de datos de SQL Server.

¿Qué cambios desea realizar en la aplicación?

Crear una nueva aplicación Access cliente/servidor.
 Vincular tablas de SQL Server a una aplicación existente.
 Ninguno.

Guardar contraseña e Id. de usuario.

Ayuda | Cancelar | < Atrás | Siguiente > | Finalizar

Asistente para convertir a SQL Server

El Asistente para convertir a SQL Server tiene toda la información que necesita para actualizar su base de datos.

Desea abrir el archivo ADP resultante tras finalizar la operación de conversión a SQL Server?

Abrir el nuevo archivo ADP
 Mantener abierto el archivo MDB

Ayuda | Cancelar | < Atrás | Siguiente > | Finalizar

Microsoft Access - [BASE CENTROS-Access2002 : Base de datos (Formato de archivo de Access 2002)]

Archivo Edición Ver Insertar Herramientas Ventana ?

Escriba una pregunta

Objetos Tablas Consultas Formularios Informes Páginas Macros Módulos Grupos Favoritos

Crear una tabla en vista Diseño
 Crear una tabla utilizando el asistente
 Crear una tabla introduciendo datos
 TABLA BASE centro Errores de conversión Fechas MOV BASE temp temp2

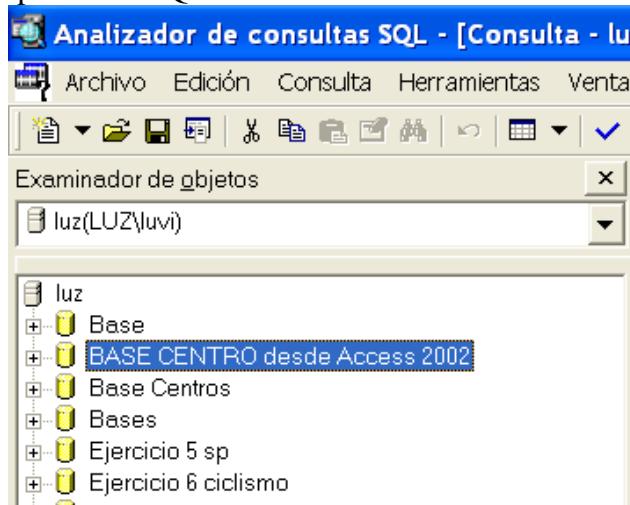
Asistente para convertir a SQL Server

Tabla 'BASE'... Creando la tabla y enviando datos... Tablas: 1/1

Al finalizar se genera un informe que muestra información sobre la operación realizada:

Asistente para convertir a SQL Server.pdf

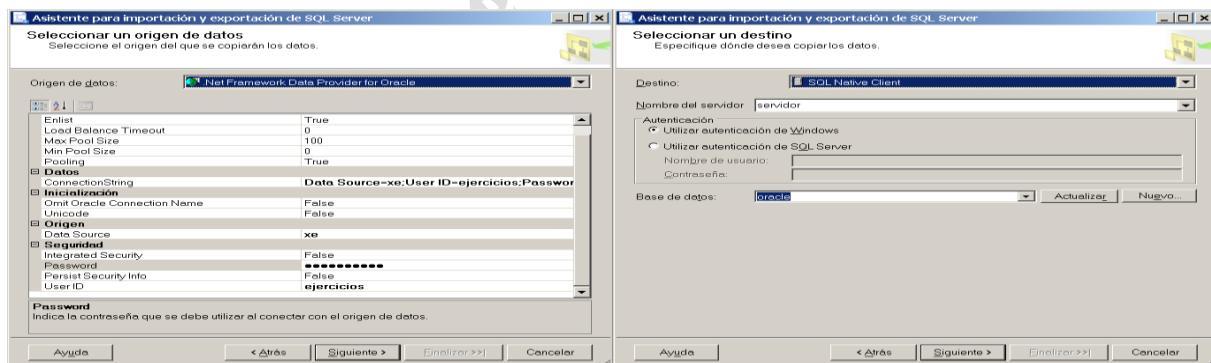
En este momento ya aparece en SQL Server nuestra BD:



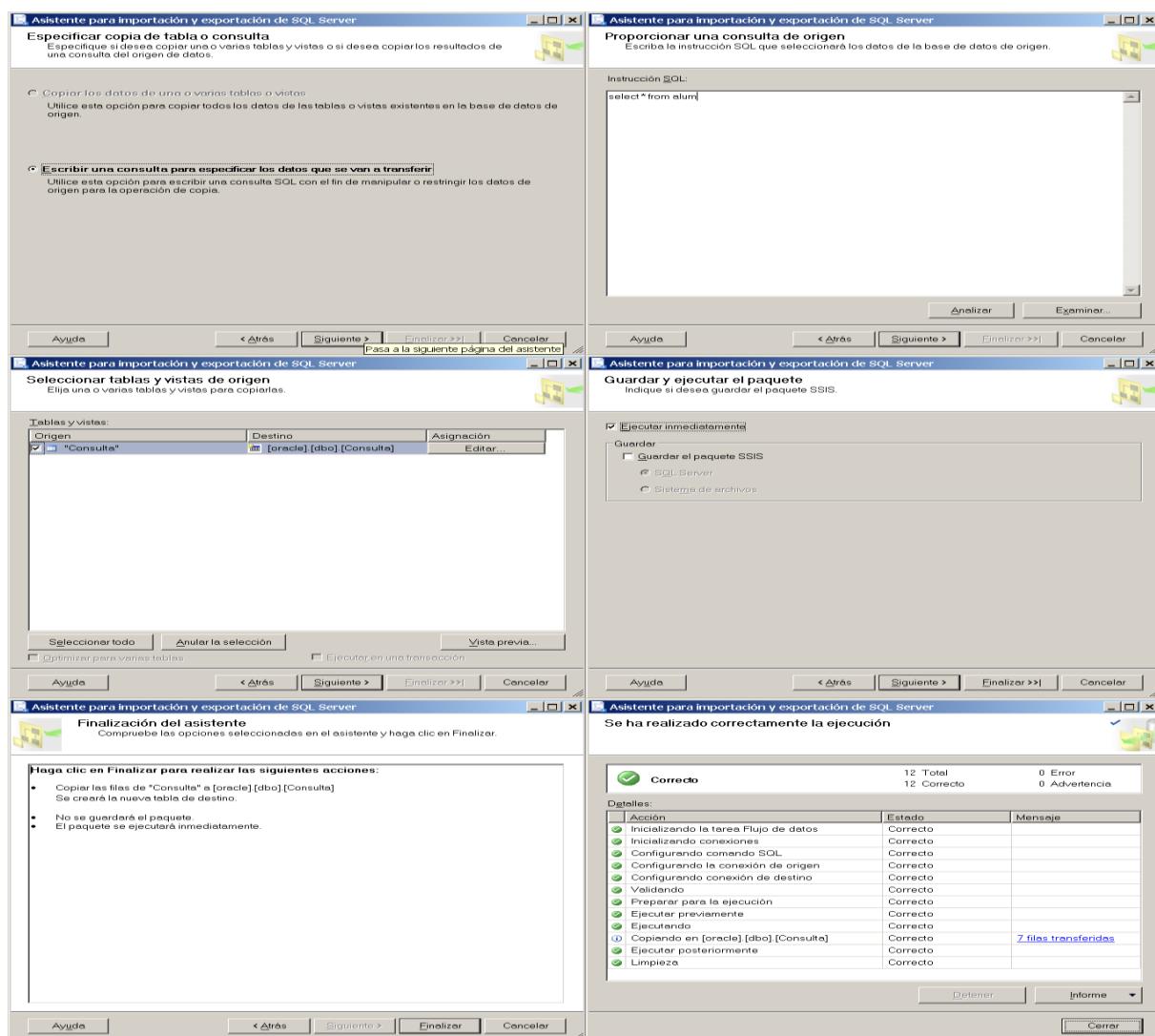
Finalmente podemos observar que no hay datos en la tabla BASE de esta base de datos.

3.7.2 Importar datos desde oracle.

La sincronización entre ambas bases de datos no es tan elevada como pueda serlo entre Access y SQL Server 2008, por lo que lo que nos permite es importar datos recogidos de una consulta contra Oracle.



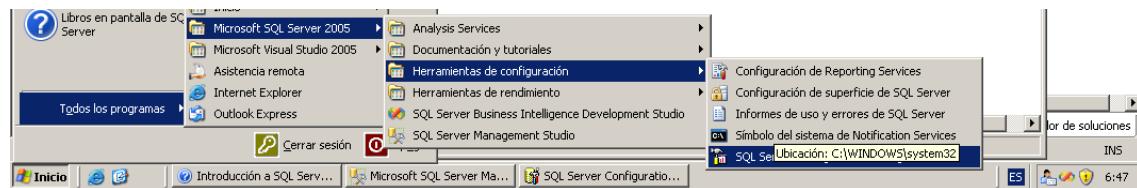
Desde un cliente el DataSource será //servidor/xe, para la versión express de oracle.

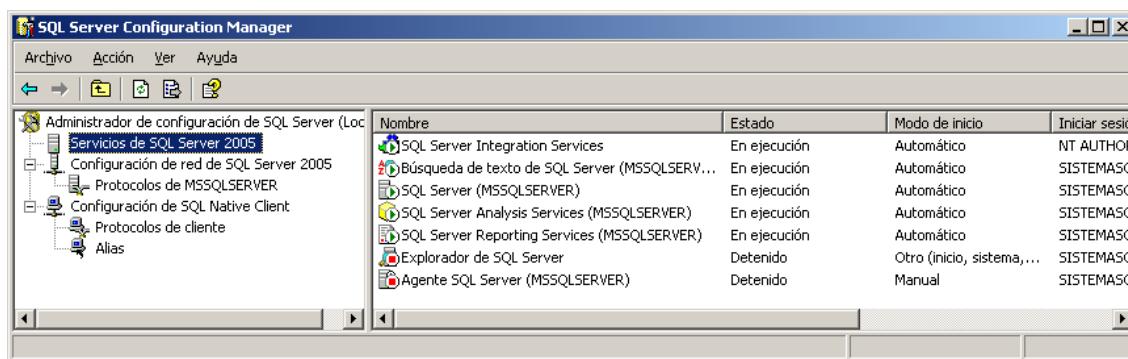


Ejercicio 1. Realizar una consulta sobre cualquier tabla de cualquier BD en la consola administrativa de SQL Server 2008 y a partir de ella crear un archivo csv que posteriormente importaremos a la base de datos prueba.

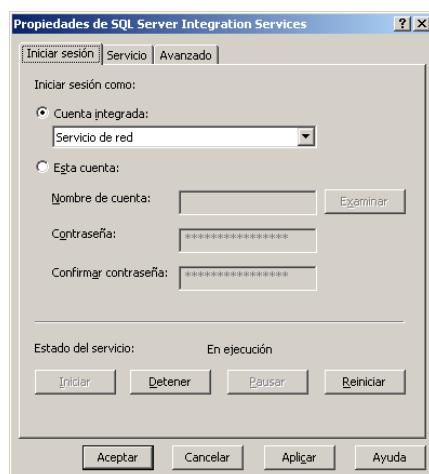
3.8 SQL Configuration Manager

Desde esta consola podremos administrar todos los servicios creados por SQL Server:





Con botón derecho propiedades podremos sacar las propiedades concretas que tengan los diferentes servicios.



Lo mismo ocurrirá con las propiedades de los protocolos de SQL Server, y protocolos de conexión del cliente (SQL Native Client) al servidor:

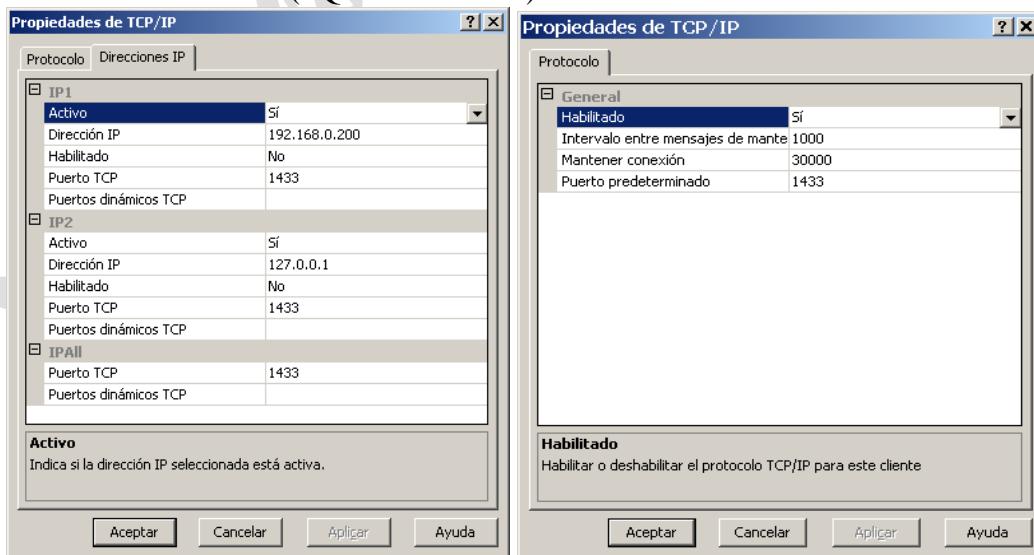


Imagen 30. TCP/IP servidor (izq.), TCP/IP cliente (der.).

Para poder realizar cambios en cualquiera de estos protocolos tendremos que dete-

ner la instancia, realizar los cambios y volverla a iniciar.

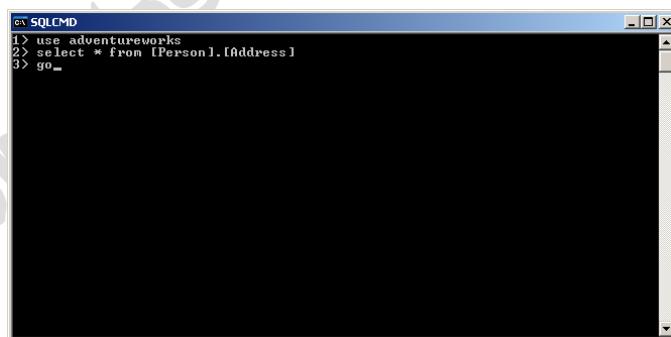
En cuanto a los protocolos de cliente serán distintos para un cliente que para un servidor, puesto que en el cliente podremos controlar tiempo de conexión inactiva por ejemplo.

Ejercicio 2. Sería interesante hacer que SQL Server sólo esté escuchando a través de una ip, en el configuration manager limitaríamos las conexiones tcp a un puerto. Comprobar con netstat -nabo

3.9 SQLCMD

La utilidad sqlcmd es una utilidad del símbolo del sistema de Microsoft® Win32® para la ejecución ad hoc e interactiva de instrucciones y secuencias de comandos Transact-SQL. La utilidad sqlcmd se usa normalmente de dos formas:

- ✓ Los usuarios escriben instrucciones Transact-SQL interactivamente de una forma similar al modo en que trabajan con el símbolo del sistema. Los resultados se muestran en la ventana del símbolo del sistema.
- ✓ Los usuarios envían un trabajo sqlcmd especificando la ejecución de una instrucción Transact-SQL individual o dirigiendo la utilidad hacia un archivo de texto que contiene las instrucciones Transact-SQL que se van a ejecutar. El resultado se dirige normalmente hacia un archivo de texto, aunque también se puede mostrar en la ventana del símbolo del sistema.



```
SQLCMD
1> use adventureworks
2> select * from [Person].[Address]
3> go
```

Los comandos más importantes son:

Go – Ejecuta las sentencias especificadas.

Reset – Borrar de caché las instrucciones escritas anteriormente

Ed – Abre un editor de las sentencias de caché pendientes de ejecución

!cmd – Ejecuta un comando de sistema operativo. Cmd será sustituido por el comando a ejecutar.

Quit – Finaliza la utilidad

:out nomrefichero – Envía los datos solicitados a un fichero.

:help – Ayuda

Ejemplo 4. Con la ayuda de la consola de sql Server podremos generar fácilmente ejecuciones programadas de alguna query.

1.- Generamos un .bat con el siguiente código:

```
set SQLCMDSERVER=localhost
sqlcmd -i query.sql -o SqlCmdOutput.out
echo %ERRORLEVEL%
```

Por ejemplo query.sql podría ser:

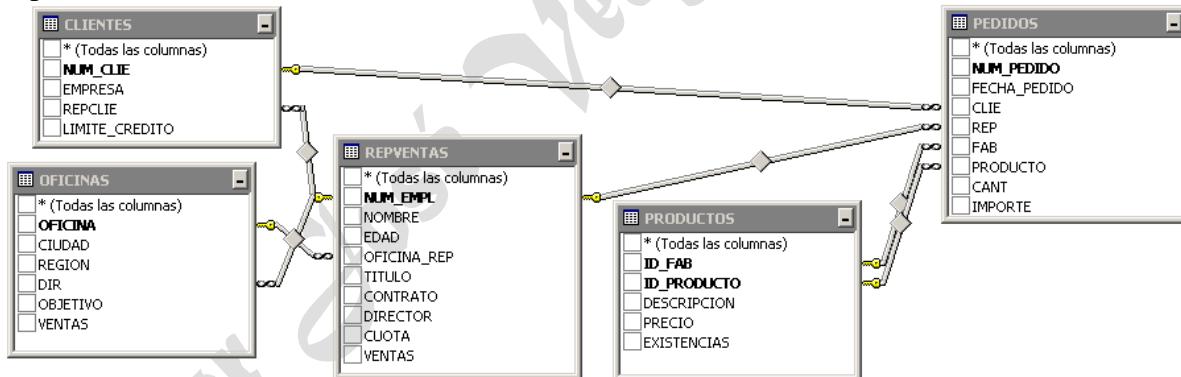
```
SELECT * FROM [AdventureWorks].[Person].[Address]
```

Debemos suponer que query.sql está en la misma carpeta que el .bat. En dicha carpeta también será donde se generarán los resultados (sqlcmdoutput.out). Si ERRORLEVEL es cero, no se habrá producido ningún error en la ejecución del script.

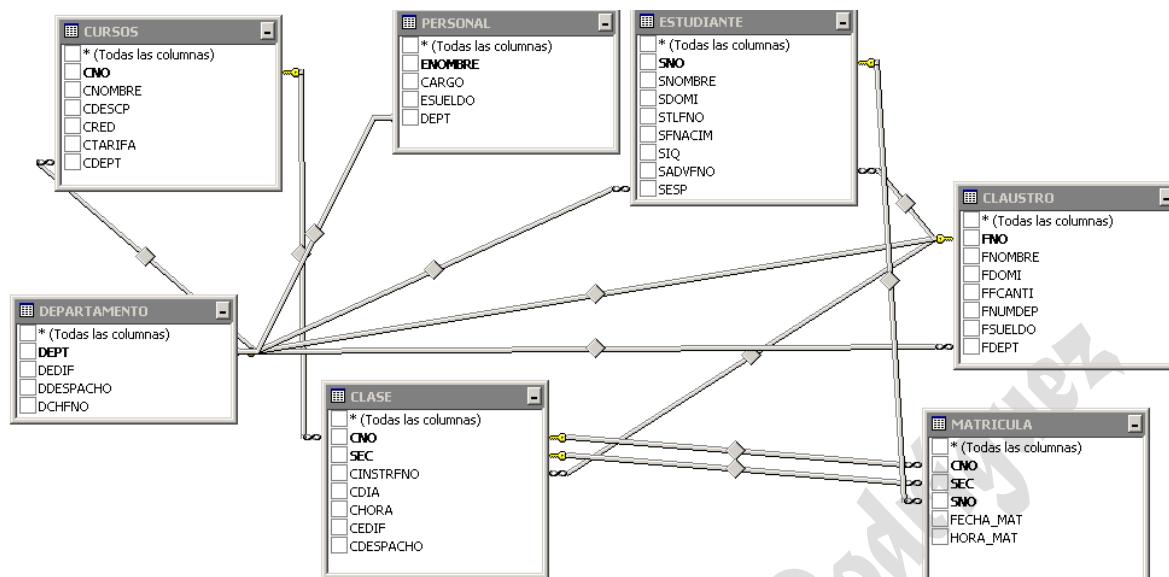
2.- Generamos una tarea programada sobre ese .bat.

Ejercicio 3. Adjuntar la bd empleados y estudiantes. Crear una tarea programada para exportar los datos de la tabla clientes.

Empleados



Estudiantes



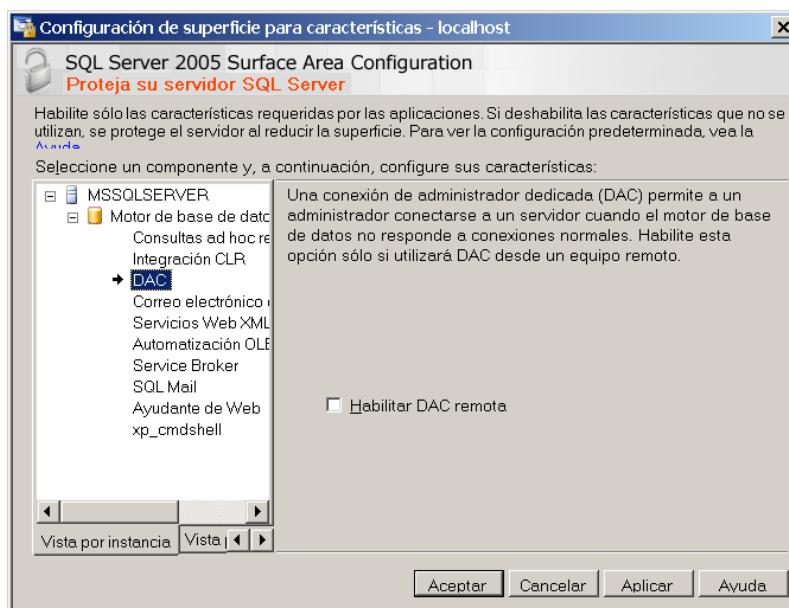
En ésta nueva versión de SQL Server se ha insertado el DAC (Dedicated Administrator Connection- Conexión de Administrador Dedicada). Se trata de la posibilidad de conectarse mediante SQLCMD a un servidor que no responde, únicamente si el servidor estuviera parado no podríamos conectarnos a él mediante esta técnica. De esta manera podríamos ejecutar comandos para diagnosticar el problema por el cual el servidor no responde, terminar conexiones problemáticas, o apagar el servidor de manera correcta y reiniciarlo.

Ejemplo 5. Una conexión de usuario que no responde es usualmente causada por un proceso que está esperando por un lock, que es producido por una transacción larga. Podemos usar sp_lock en combinación con kill para poder ver qué proceso es y después eliminarlo.

```

C:\Documents and Settings\Administrador>sqlcmd -A
1> sp_lock
2> go
spid  dbid  ObjId      IndId  Type Resource
Status
-----
      52     15          0        0 DB
GRANT
      53      5          0        0 DB
GRANT
      54      1  1115151018        0 TAB
GRANT
      55      5          0        0 DB
GRANT
1> kill 52
  
```

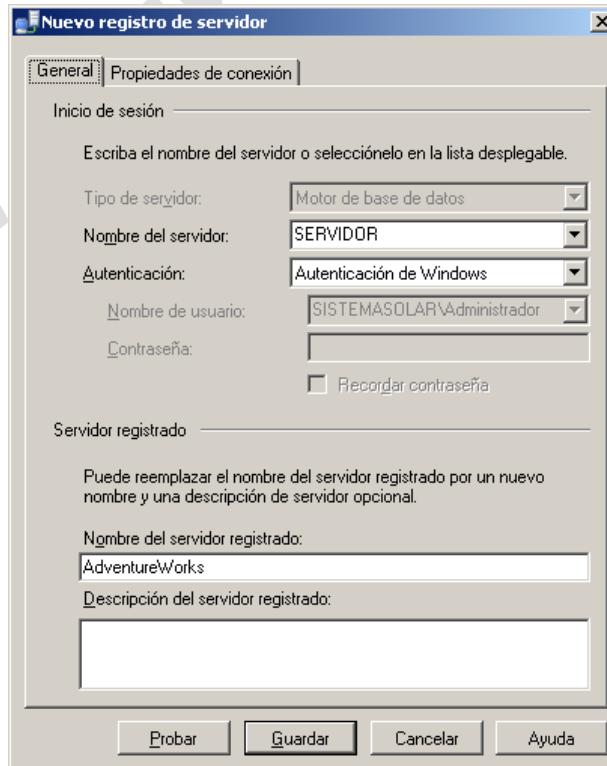
Para establecer conexiones remotas desde SQLCMD debemos habilitarlo en el configurador de superficie:



3.10 SQL Server Management Studio

3.10.1 Registro de servidores

La ventana Servidores registrados aparece encima del Explorador de objetos. Servidores registrados enumera los servidores que el usuario administra habitualmente. Puede agregar y quitar los servidores de esta lista.



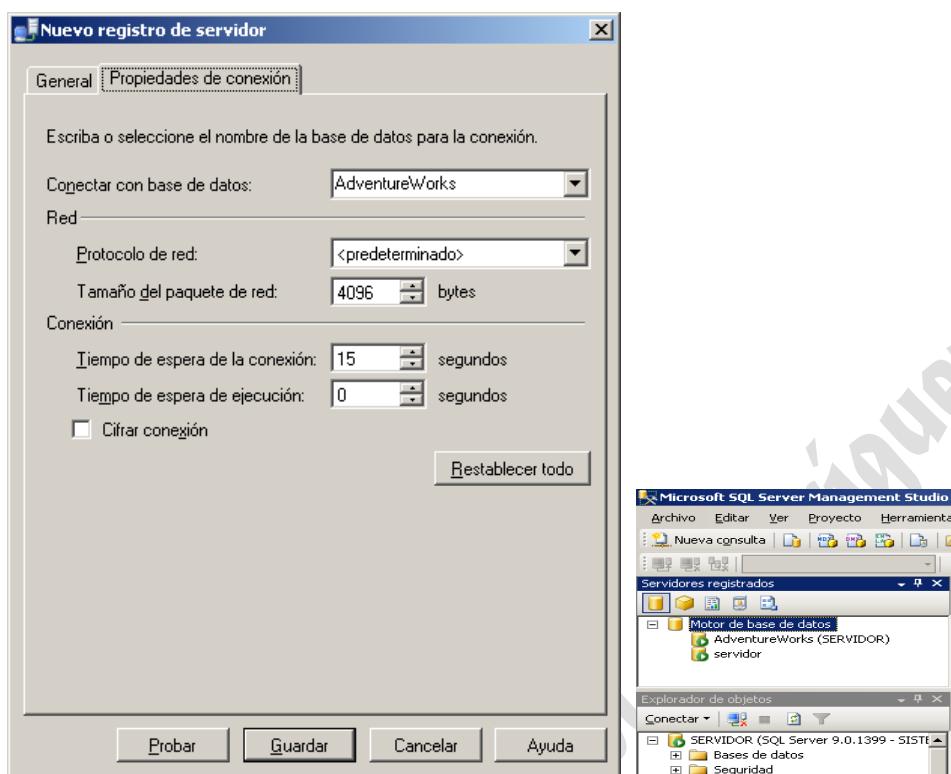


Imagen 31. Registro de la BD AdventureWorks

Ejercicio 4. Registrar todos los servidores de la red, creando un grupo donde serán insertados todos ellos.

3.10.2 Explorador de objetos.

Desde el explorador de objetos podremos acceder a través de un árbol de contenidos a las diferentes partes de nuestro servidor de BD. También podemos añadir nodos para otros servidores como el de Informes, esto se realizará desde el panel de servidores registrados, botón derecho conectar, explorador de objetos.



Imagen 32. Explorador de objetos con el servidor Reporting Services

Una utilidad del explorador de objetos es el filtrado de objetos para localizar por ejemplo tablas de un determinado esquema, o con un nombre concreto.

Nota.- Para poder colocar el nodo de Reporting Services previamente debemos de configurarlo: Microsoft SQL Server 2008\Herramientas de configuración\Configuración de Reporting Services

3.10.3 Explorador de plantillas

Microsoft SQL Server Management Studio ofrece un gran número de plantillas de secuencias de comandos que incluyen instrucciones Transact-SQL para muchas de las tareas habituales. Puede crear sus propias plantillas personalizadas para las secuencias de comandos que escriba con más frecuencia. Ej.: Creación de sinónimos, tablas, usuarios, vistas...etc.

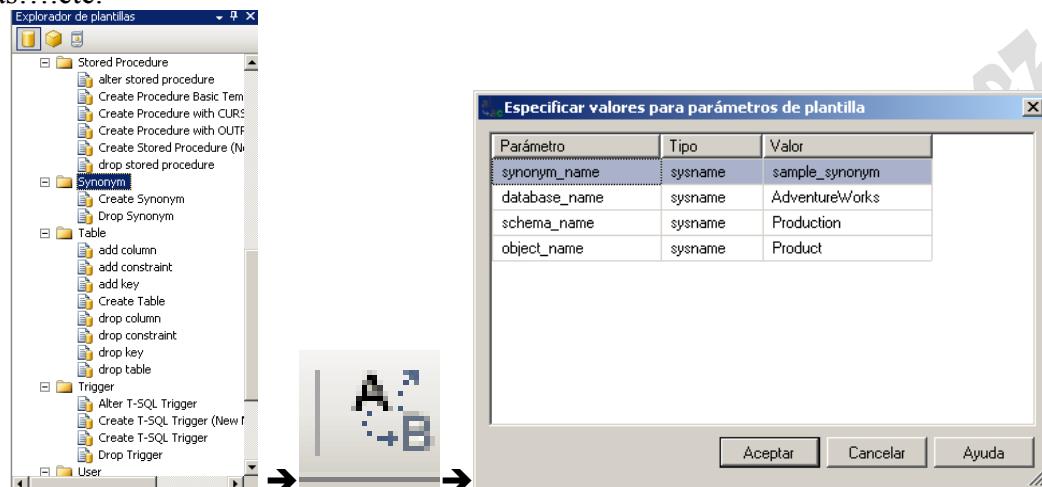


Imagen 33. Explorador de plantillas

Ejemplo 6. Crear a través de las plantillas un sinónimo y una vista de la base de datos de Empleados.

Puedes crear tu propia plantilla siguiendo la sintaxis siguiente:

```
select * from authors
where au_lname like '<nombre, varchar(10),>'
```

3.11 Instantáneas

Son una característica nueva de Microsoft SQL Server 2008. Las instantáneas de base de datos sólo están disponibles en la versión Enterprise Edition. Una instantánea de base de datos es una vista estática de sólo lectura de una base de datos denominada base de datos de origen. Pueden existir varias instantáneas en una base de datos de origen y residir siempre en la misma instancia de servidor que la base de datos. Una instantánea se mantiene hasta que el propietario de la base de datos la quita explícitamente.

Las instantáneas se pueden utilizar para crear informes. Asimismo, si más adelante se daña la base de datos de origen, podrá devolverla al estado en el que se encontraba en el momento de la creación de la instantánea. La pérdida de datos se limitará a las actualizaciones de la base de datos realizadas después de la creación de la instantánea.

A continuación se exponen los motivos para realizar instantáneas de base de datos:

✓ **El mantenimiento de los datos históricos para la creación de informes.**

Puesto que la instantánea de una base de datos proporciona una vista estática de ésta, una instantánea puede ampliar el acceso a datos en un momento determinado. Por ejemplo, se puede crear una instantánea de base de datos al final de un período determinado, como un trimestre financiero, para la creación posterior de informes. A continuación, podrá ejecutar informes de final de período basándose en la instantánea.

✓ **Protección de datos contra errores administrativos.**

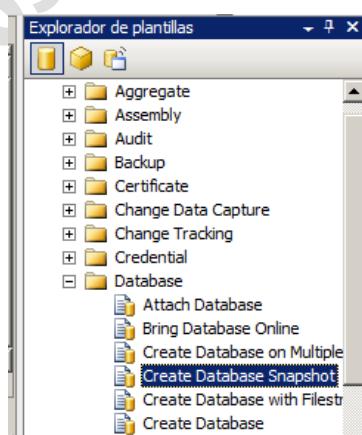
Antes de realizar actualizaciones principales, como las masivas, la creación de una instantánea de base de datos protege los datos. Si comete un error, podrá utilizar la instantánea para la recuperación devolviendo la base de datos al estado en el que estaba cuando se realizó la instantánea. Esta acción suele resultar mucho más rápida para este fin que la restauración a partir de una copia de seguridad, pero después no se puede realizar una puesta al día.

Nota.- La utilización de instantáneas de base de datos para volver a un estado anterior no debe constituir un sustituto de la estrategia de copia de seguridad y restauración. Resulta esencial realizar todas las copias de seguridad programadas.

✓ **Protección de datos contra errores por parte de los usuarios.**

La creación de instantáneas de base de datos regularmente permite mitigar el impacto de un error importante por parte de un usuario, como la eliminación de una tabla.

3.11.1 Asignar nombres a instantáneas de bases de datos



Antes de crear instantáneas, es importante pensar cómo asignarles un nombre. Cada instantánea de base de datos necesita un nombre de base de datos único. Para facilitar la administración, el nombre de una instantánea puede incorporar información que identifique la base de datos, por ejemplo:

- ✓ El nombre de la base de datos de origen.
- ✓ Una indicación de que el nuevo nombre es para una instantánea.

- ✓ La fecha y hora de creación de la instantánea, un número de secuencia o cualquier otra información, por ejemplo, la hora del día, para distinguir instantáneas secuenciales en una base de datos dada.

Ejemplo 7. Creación de instantáneas a través de las plantillas

(Ver - Explorador de Plantillas – Database – create snapshot)

Por ejemplo, crearemos una serie de instantáneas de la base de datos Adventure-Works. Se crean tres instantáneas diarias a intervalos de 6 horas entre las 06:00 y las 18:00. Cada instantánea diaria se conserva 24 horas antes de que se quite y sea reemplazada por una nueva instantánea con el mismo nombre. Recuerde que cada nombre de instantánea indica la hora, pero no el día:

AdventureWorks_snapshot_0600
AdventureWorks_snapshot_1200
AdventureWorks_snapshot_1800

Como alternativa, si la hora de creación de estas instantáneas diarias varía cada día, es posible que sea preferible disponer de una convención de nomenclatura menos precisa, por ejemplo:

AdventureWorks_snapshot_mañana
AdventureWorks_snapshot_mediodía
AdventureWorks_snapshot_tarde

```
CREATE DATABASE AdventureWorks_dbss1800 ON
( NAME = AdventureWorks_Data , FILENAME =
'C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\Data\AdventureWorks_data_1800.ss' )
AS SNAPSHOT OF AdventureWorks;
GO
```

Nota.- En NAME colocaremos el nombre lógico de la BD, en propiedades de la BD podremos verlo.

3.11.2 Conexiones de clientes con una instantánea de base de datos

Para utilizar una instantánea de base de datos, los clientes deben saber dónde encontrarla. Los usuarios pueden leer de una instantánea de base de datos mientras se crea o elimina otra. Sin embargo, si sustituye una nueva instantánea por otra ya existente, debe redirigir a los clientes a la nueva instantánea. Los usuarios pueden conectarse manualmente a una instantánea de base de datos mediante SQL Server Management Studio. Sin embargo, para admitir un entorno de producción, debe crear una solución programática que dirija de un modo transparente a los clientes de escritura de informes a la instantánea de base de datos más reciente de la base de datos.

Ejemplo 8. Crearemos una BD sobre la que insertaremos ciertos datos. Para probar el funcionamiento de la imagen analizaremos qué ocurre sobre la imagen cuando se borran datos de la BD .

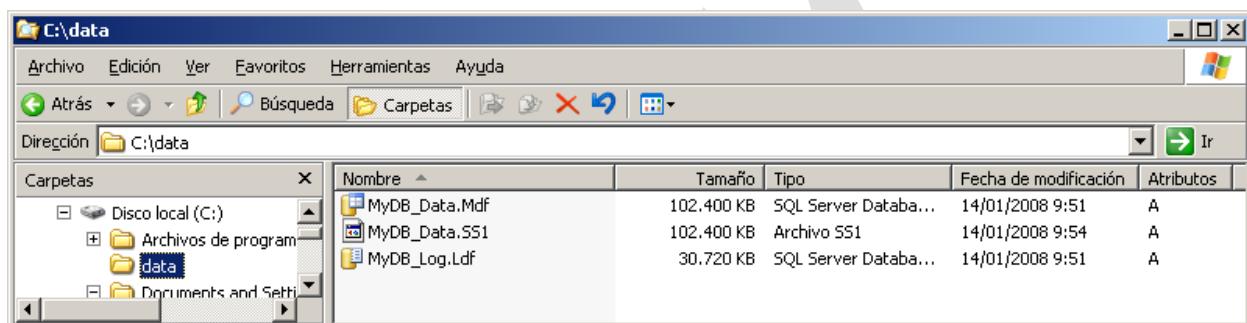
Primero creamos la tabla en C:\data y la BD

—

Insertamos datos:

```
Use MyDB
go
Create table Hubble_Galaxies (Id int, name varchar(100))
go
insert into Hubble_Galaxies values (1, 'MilkyWay')
insert into Hubble_Galaxies values (2, 'Spiral Galaxy NGC 1300')
insert into Hubble_Galaxies values (3, 'Whirpool Galaxy M51')
insert into Hubble_Galaxies values (4, 'Galaxy NGC 1427A')
insert into Hubble_Galaxies values (5, 'NGC 3370')
go
;
```

Creamos la instantánea, con lo que se creará el archivo ss1:



```
use master
go
Create Database MyDB_Snapshot1 on
  (Name = 'MyDB_Data',      --> Nombre del fichero de datos de la BD
   FileName= 'C:\data\MyDB_Data.SS1') --> Creo que da problemas si tiene
                                         espacios la ruta
   AS SNAPSHOT of MyDB;
go
```

Si ahora probamos a borrar:

```
use MyDB
go
delete from Hubble_Galaxies where id in (3,5)
go
```

Podremos ver los datos de la instantánea con

```
use MyDB_Snapshot1
go
Select * from Hubble_Galaxies
go
```

Ejercicio 5. Crear una instantánea de los datos de Empleados.

```
-- Create the snapshot database
CREATE DATABASE Empleados_Imagen ON
( NAME = EMPLEADOS_Data, FILENAME =
'd:\bd\01-EMPLEADOS_Data_Empleados_Imagen.ss' )
AS SNAPSHOT OF EMPLEADOS;
GO
```

Empleados_Imagen: Nombre que aparecerá en el árbol del explorador de objetos “*Instantáneas de base de datos*”.

EMPLEADOS_data: Nombre lógico del archivo, no del archivo físico (puedes verlo en propiedades de la BD)

EMPLEADOS: Nombre de la BD de la cual queremos hacer la imagen

Aunque no es una práctica recomendable restaurar datos perdidos de un snapshot podremos hacerlo de la siguiente manera:

Por ejemplo, un nombre de usuario Fred ha recuperado todas las filas en la tabla Production.WorkOrderRouting en la base de datos AdventureWorks han desaparecido. Puedes restaurar las filas perdidas de la base de datos AdventureWorks_dbsnapshot_1800 usando la statements como muestra el siguiente ejemplo:

```
ALTER TABLE Production.WorkOrderRouting
NOCHECK CONSTRAINT CK_WorkOrderRouting_ActualEndDate
INSERT INTO Production.WorkOrderRouting
SELECT *
FROM
AdventureWorks_dbsnapshot_1800.Production.WorkOrderRouting
ALTER TABLE Production.WorkOrderRouting
CHECK CONSTRAINT CK_WorkOrderRouting_ActualEndDate
```

3.11.2.1 Restricciones a tener en cuenta sobre las instantáneas

Las siguientes restricciones se aplican a las instantáneas de base de datos:

- ✓ No pueden ser creadas para bases de datos model, master, o tempdb.
- ✓ Son de solo lectura; los usuarios no las pueden modificar.
- ✓ No pueden ser restauradas o hacerse backup.
- ✓ No pueden ser adjuntadas o desadjuntadas.
- ✓ No pueden ser creadas en FAT32 o en particiones raw.
- ✓ Todas las instantáneas creadas sobre la base de datos deben ser borradas antes que la misma base de datos sea borrada.
- ✓ Restricciones de Seguridad en objeto instantánea de base de datos no puede ser modificado.

Módulo 4

SQL SERVER 2008. TABLAS, ÍNDICES Y VISTAS

4.1 Crear una nueva tabla /vista

A través del Microsoft SQL Server Management Studio podremos crear de manera más fácil las diferentes tablas y restricciones existentes.

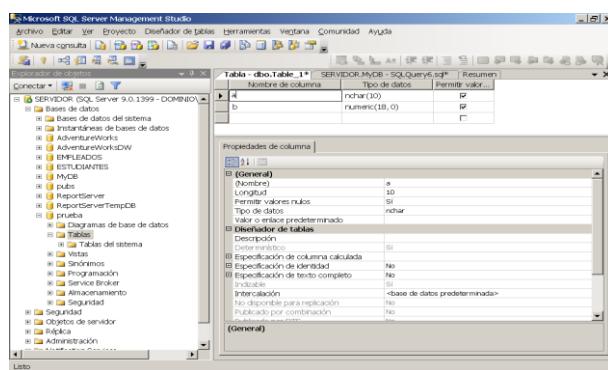


Imagen 34. Creación de una tabla a través del diseñador de tablas

Para crear una tabla deberemos conocer los diferentes tipos de datos que existen (véase apartado 5.2.1.2 Tipos de datos en la pág. 67). La sentencia DML equivalente será la CREATE TABLE (Véase apartado 5.2.1. CREATE TABLE en la pág. 64). Una vez diseñada deberemos pulsar en guardar para que perdure en el sistema.

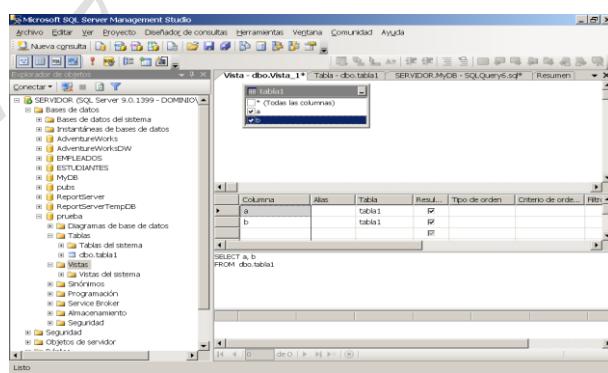


Imagen 35. Creación de una vista a través del diseñador de consultas

Sobre la tabla creada podemos definir las propiedades que tiene, es decir, nombre, propiedades, relaciones con otras tablas, índices y claves, restricciones CHECK...

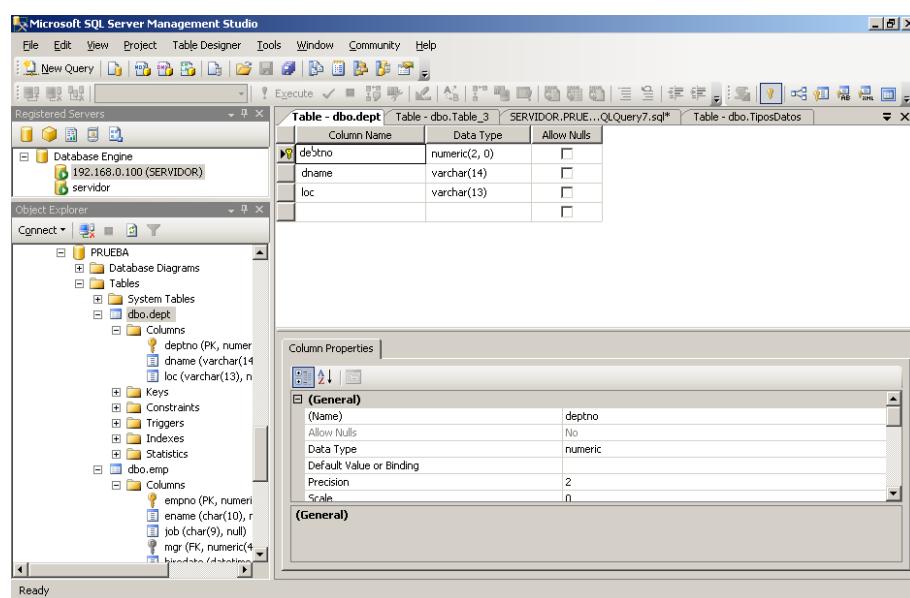


Imagen 36. Propiedades de la tabla

Las propiedades de estos tipos de datos al crear la tabla a través del diseñador son:

Descripción: Comentario acerca del tipo de dato que se almacenará en el campo concreto.

Valor predeterminado: Valor por defecto.

Identidad: Sería como si definimos en Access un autonumérico. No podremos por lo tanto definir esta propiedad sobre campos de tipo texto.

Fórmula: Sería un campo calculado. Para poner número decimales deberemos usar el punto. Funciones numéricas a utilizar:

a+b	Suma de a y b	SIGN(a)	Vale 1 si a es positivo, o si es cero y -1
a-b	Diferencia entre a y b	SIN(a)	Seno de a
a*b	Producto de a y b	SQRT(a)	Raíz cuadrada de a
a/b	Cociente entre a y b	SQUARE(a)	Cuadrado de a
ABS(a)	Valor absoluto de a	TAN(a)	Tangente de a
ASIN(a)	Arco coseno de a (en radianes)		
ATAN(a)	Arco tangente de a (en radianes)		
ATN2(a,b)	Angulo (en radianes) cuya tangente esta entre a y b		
CEILING(a)	Entero más pequeño mayor o igual que a		
COS(a)	Coseno de a (en radianes)		
COT(a)	Cotangente de a (en radianes)		
DEGREES(n)	Pasa a grados n radianes		
EXP(a)	Numero e elevado al exponente a		
FLOOR (a)	Mayor entero menor o igual que a		
LOG(a)	Logaritmo neperiano de a		
LOCI0(a)	Logaritmo decimal de a		
PI	Valor del numero PI		
POWER(a,b)	Halla a elevado a b		
RADIANS(n)	Pasa a radianes n grados		
RAND(semilla)	Halla un numero aleatorio entre cero y 1 con la semilla dada		
ROUND(a,b)	Redondea a con precisión b		

4.1.1 Creación de índices

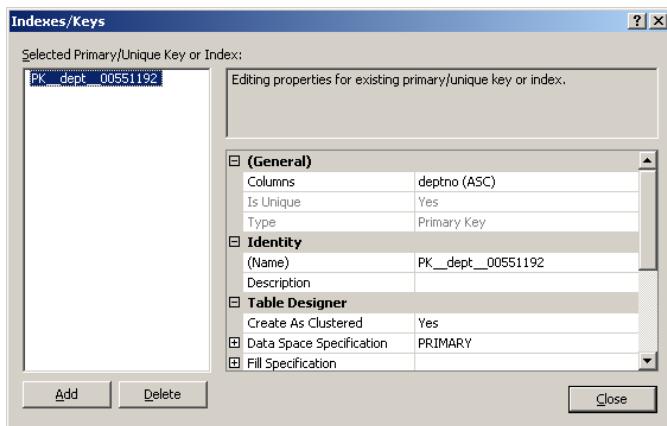


Imagen 37. Creación de índices

Un índice es un mecanismo que proporciona acceso rápido a filas de la tabla o que permite aplicar determinadas restricciones. Se puede por tanto utilizar un índice para acelerar el acceso a datos de una tabla de base de datos. Los índices de tablas tienen un funcionamiento similar al de los índices de los libros, los cuales permiten acceder rápidamente a los datos. Existen 2 tipos de índices: en cluster o sin él.

Clustered (agrupado).

Sabiendo que los datos se van almacenados en grupos, digamos que este índice ordena esos grupos.⁴ Sólo puede existir uno por tabla. Normalmente es más rápido que el resto de índices. El orden físico de las filas coincide con el orden que proporciona las claves de este tipo de índices. Consideraciones:

- ✓ Los índices agrupados no son adecuados para los siguientes atributos:
 - Columnas sometidas a cambios frecuentes. Esto provoca que se mueva toda la fila, ya que Database Engine (Motor de base de datos) debe mantener los valores de los datos de la fila ordenados físicamente.
 - Claves amplias. Las claves amplias se componen de varias columnas o varias columnas de gran tamaño. Los valores clave del índice agrupado se utilizan en todos los índices no agrupados como claves de búsqueda. Los índices no agrupados definidos en la misma tabla serán bastante más grandes, ya que sus entradas contienen la clave de agrupación y las columnas de clave definidas para dicho índice no agrupado.

Nonclustered index (no agrupado).

Un ejemplo fácil de entender de este tipo de índice es asemejarlo con los índices que vienen al final de los libros que también permiten buscar información. Consideraciones:

⁴ No se puede crear un índice si existen varias filas con NULL.

- ✓ Las bases de datos o tablas que exigen pocos requisitos para la actualización, pero suelen contener un gran volumen de datos, se pueden beneficiar de muchos índices no agrupados para mejorar el optimizador de consultas.
- ✓ Los índices deben ser estrechos, es decir, con la menor cantidad de columnas posible. Si se utiliza un gran número de índices en una tabla, el rendimiento de las instrucciones INSERT, UPDATE y DELETE se verá afectado, ya que todos los índices deben ajustarse adecuadamente a medida que cambian los datos de la tabla.

Debido a que las claves primarias identifican únicamente los registros en las tablas, están muy estrechamente relacionados con los índices, puesto que la dificultad de los índices es que sean modificados y las claves primarias no suelen ser cambiadas. Esto es porque cada vez que se modifica un campo con índice éste se debe modificar. Es por esto por lo que los índices se crean automáticamente cuando las restricciones PRIMARY KEY y UNIQUE se definen en las columnas de tabla.

✓ **Único.**

Un índice único garantiza que la clave de índice, valores del campo sobre el que se crea el índice, no contenga valores duplicados y, por tanto, cada fila de la tabla o vista es en cierta forma única. Tanto los índices agrupados como los no agrupados pueden ser únicos.

✓ **Índice con columnas incluidas**

Índice no agrupado que se extiende para incluir columnas sin clave además de las columnas de clave.

✓ **Texto**

Tipo especial de índice funcional basado en símbolos (token) que crea y mantiene el servicio del Motor de texto completo de Microsoft para SQL Server (MSFTESQL). Proporciona la compatibilidad adecuada para búsquedas de texto complejas en datos de cadenas de caracteres. Es decir, los índices habitualmente se crean con los primeros caracteres de una columna, en este caso de texto, esto evita tener una respuesta rápida si realizamos búsquedas con el operador LIKE donde el texto buscado está en el medio o al final del campo, en esos casos concretos debemos usar este tipo de índices.

✓ **XML**

Representación dividida y permanente de los objetos XML binarios grandes (BLOB) de la columna de tipo de datos xml.

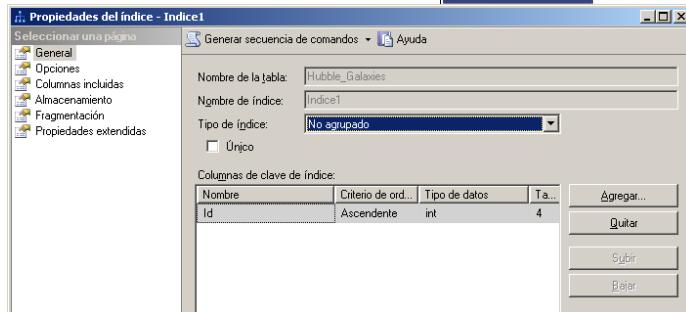


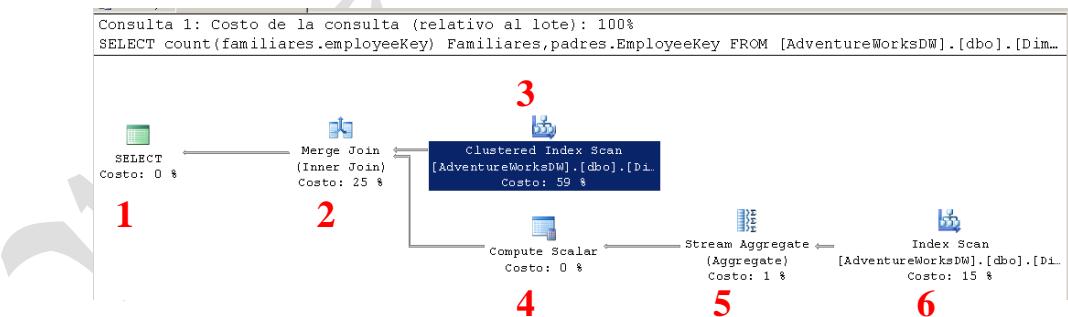
Imagen 38. Propiedades de un índice No agrupado

Para realizar una optimización de la BD mediante índices véase *10.6 Asistente para la optimización de la BD*.

Ejemplo 9. A continuación haremos una comprobación de la ejecución de una query sobre una tabla con índices y otra sin ellos. La Query representa el número de familiares que tiene un empleado en la empresa.

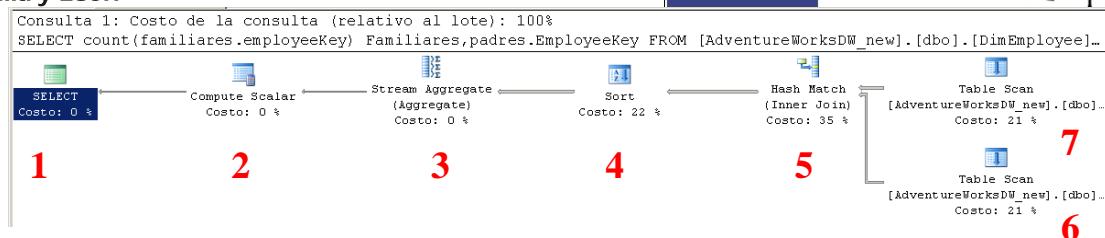
```
SELECT count(familiares.employeeKey) Familiares, padres.EmployeeKey
  FROM [AdventureWorksDW].[dbo].[DimEmployee]
familiares, [AdventureWorksDW].[dbo].[DimEmployee] padres
 WHERE familiares.ParentEmployeeKey=padres.EmployeeKey
 GROUP BY padres.EmployeeKey
```

La tabla tiene tres índices IX_DimEmployee_ParentEmployeeKey(no único, no agrupado), IX_DimEmployee_SalesTerritoryKey (no único, no agrupado), y PK_DimEmployee_EmployeeKey (agrupado). Uno de ellos depende de la clave primaria. El plan de ejecución nos informará de cuales serán los tiempos estimados de ejecución. De aquí se sacará la tabla que viene a continuación:



Nodo	1	2	3	4	5	6	TOTAL
Tiempo CPU		0,0063271	0,0004826	0,0002016	0,0002016	0,0004826	0,0076955
Número de filas	47	47	296	48	48	296	782

Si realizamos la misma query sobre la misma tabla pero en esta ocasión sin índices tendríamos el siguiente plan de ejecución:



Nodo	1	2	3	4	5	6	7	TOTAL
Tiempo CPU	0,0002005	0,0002005	0,0038758	0,0241302	0,0004826	0,0004826		0,0293722
Número de filas	4 7	47	295	295	296	296		1323

Resumiendo el tiempo de CPU es un 74% menor y el número de filas tratadas un 41% menor. Por lo tanto, es evidente la mejoría que nos aporta la el uso de índices, los resultados son infinitamente mejor. Estos costes están calculados para tablas de 296 registros, con grandes tablas el tiempo de CPU ganado es increíble.

4.1.2 Relaciones

Uno a varios. Se produce cuando un registro de una tabla está relacionado con varios registros de la tabla con la que se relaciona. El extremo de la relación que es el “uno” será clave primaria , o bien no puede tener valores duplicados. El extremo “varios”  será un campo que suele ser lo que se llama clave foránea, clave secundaria, clave externa o FOREIGN KEY, es decir, que sus valores serán los valores existentes en el campo con el que se relaciona. Ej.: Una factura sólo pertenece a un cliente y un cliente puede tener varias facturas.

Varios a varios. Ej.: un número de cuenta puede ser de varios clientes (titulares), y cada cliente o titular puede tener varias cuentas.

Uno a uno. Se creará entre dos claves primarias o campos con restricciones de unicidad. Ej.: En una base de datos donde se almacena por un lado características de los aviones y por otro, datos de los pilotos, la relación de *pilotar un avión actualmente* sería 1 a 1.

A la hora de crear relaciones el diagrama de bases de datos funciona perfectamente.

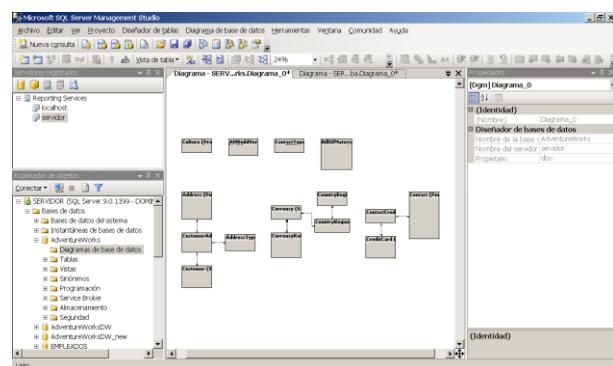


Imagen 39. Creación del diagrama de base de datos mediante SQL Server Management Studio

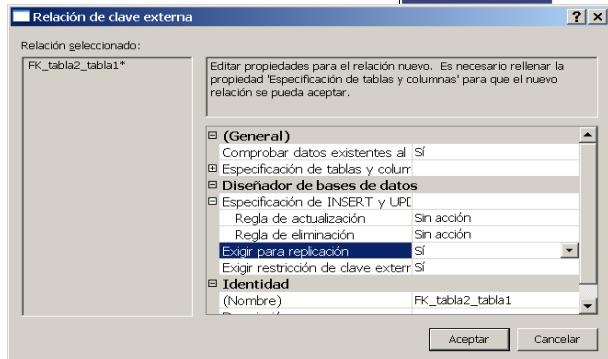


Imagen 40. Creación de relaciones

Regla de actualización y regla de eliminación, definen qué operación realizar se se actualiza o elimina un registro relacionado.

Exigir para replicación: Indica si se exigirá la restricción cuando un agente de réplica realice una inserción o actualización en esta tabla.

Exigir restricción de clave externa: Especifica si se pueden modificar los datos de las columnas de la relación y si estos cambios pueden invalidar la integridad de la relación de clave externa. Elija Sí si no desea permitir esos cambios y No si desea permitirlos.

Véase 5.2.1.1 *Restricciones*:

Podemos ver con botón derecho sobre un objeto del explorador de Windows las dependencias que tiene:

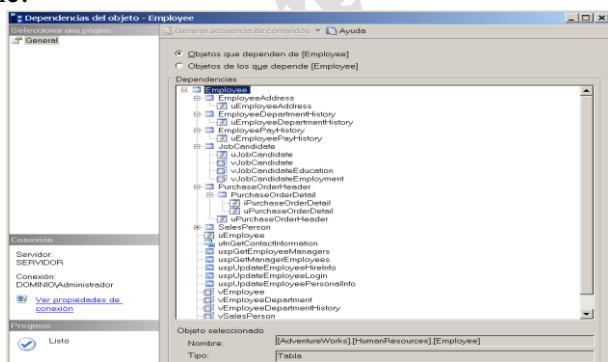


Imagen 41. Dependencias de un objeto

4.1.3 Restricciones – CHECK

[Pérez, 2003b]

Acepta cualquier combinación de operadores lógicos.

OR / AND

IN (valor1, valor2,.....)- da TRUE si el operando es igual a uno de la lista de expresiones.

BETWEEN valor1 and valor2- TRUE si el operando está dentro de un intervalo.

ALL - TRUE si el conjunto completo de comparaciones es TRUE

ANY=SOME - TRUE si algún valor del conjunto de comparaciones es TRUE.

EXISTS - TRUE si una subconsulta contiene a cualquiera de las filas.

LIKE - da TRUE si el operando es igual a uno de la lista de expresiones.

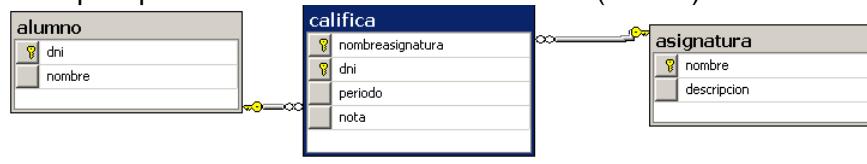
NOT - invierte el valor de cualquier otro operador booleano

Ejemplo de restricciones CHECK:

```
state = 'NY'  
zip LIKE '[0-9][0-9][0-9][0-9][0-9]'  
salary >= 15000 AND salary <= 100000
```

Ejercicio 6

- 1) Suponemos el siguiente supuesto: Un profesor de un ciclo formativo de grado superior desea suspender a todos sus alumnos. Para tener constancia de ello creará una base de datos con dos tablas: alumno (dni, nombre), califica(dni,nombreasignatura, periodo, nota) y asignatura (nombre, descripc i on).
- a) Crear las diferentes estructuras de datos con las correspondientes claves primarias y ajena (todos los campos de tipo texto menos periodo de tipo fecha y la nota num e rico), sabiendo que alumno se relaciona con califica por dni y asignatura por nombre con nombreasignatura.
- b) Además para asegurarse de que suspenden en la creación de la tabla nota se debe verificar que ésta es obligatoriamente inferior a cinco.
- c) En descripción de la asignatura sólo podrán existir tres valores: MARIA, HUESO, NORMAL.
- d) La nota nunca puede tomar un valor nulo. El periodo por defecto tomará la fecha actual de la inserción de la nota
- e) Despu e s de crear todo añadir un campo a califica “regalo” de tipo texto para saber el regalo realizado del alumno al profesor para aprobar la asignatura con valores posibles: jamon,queso,chorizo,lechazo
- 2) Comprobar qué tipo de índice se crea en cada tabla (cluster).



Módulo 5 SQL

5.1 Introducción SQL (*Structured Query Language*)

Es el lenguaje que permite la comunicación con el sistema gestor de bases de datos (Oracle, SQL Server). SQL incluye **DDL** (lenguaje de definición de datos, con sentencias de creación de datos) y **DML** (lenguaje de manipulación de datos).

Por ejemplo en DDL tenemos instrucciones como: CREATE, ALTER.
DML tiene instrucciones como: SELECT, INSERT...

5.2 DDL

5.2.1 CREATE TABLE

Pasos que debemos seguir al crear una tabla:

Hay que tener en cuenta ciertas restricciones en la formación de los nombres de las tablas:

- La longitud máxima ha de ser de 30 caracteres.
- No puede haber nombres de tablas duplicados
- Deben comenzar por un carácter alfabético. A partir de hay admite caracteres alfanuméricos y además en guion bajo.
- Oracle/SQL Server no distingue entre mayúsculas y minúsculas.

Sintaxis de comando⁵:

```
CREATE TABLE
[ database_name.[ owner ] . | owner. ] table_name
( { < column_definition >
  | column_name AS computed_column_expression -- Campo calculado6
  | < table_constraint > ::= [ CONSTRAINT constraint_name ]
    | [ { PRIMARY KEY | UNIQUE } [ ,...n ]
  )
[ ON { filegroup | DEFAULT } ]
[ TEXTIMAGE_ON { filegroup | DEFAULT } ]
```

⁵ Sabiendo las siguientes equivalencias podremos entender mejor la sintaxis:

[] opcional

() agrupar

{ } agrupar a un nivel superior

+ repeticiones de lo que hay entre llaves 1 o más veces

* repeticiones de lo que hay entre llaves 0 o más veces

| optativo

⁶ Se trata de una columna virtual ya que no se almacena físicamente en la tabla. La expresión que calcula el dato no puede ser una subconsulta.

```

<column_definition> ::= { column_name data_type }
[ COLLATE <collation_name> ] -- Intercalación7
[ [ DEFAULT constant_expression ]
| [ IDENTITY [ ( seed , increment ) [ NOT FOR REPLICATION ] ] ] -- Columna autoincre-
mental sólo puede existir una. Si no colocamos la semilla y el incremento por defecto coge-
rá (1,1).
]
[ ROWGUIDCOL] – Sólo se puede asignar a columnas de tipo uniqueidentifier. No genera
valores nuevos automáticamente. Tendríamos que utilizar newid en la instrucción INSERT.
[ <column_constraint> ][ ...n ]

```

```

<column_constraint> ::= [ CONSTRAINT constraint_name ]
{ [ NULL | NOT NULL ]
| [ { PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
[WITH FILLFACTOR = fillfactor]
[ON { filegroup | DEFAULT } ] ]
| [ [ FOREIGN KEY ]
REFERENCES ref_table [ ( ref_column ) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]
]
| CHECK [ NOT FOR REPLICATION ]
( logical_expression )
}

```

```

<table_constraint> ::= [ CONSTRAINT constraint_name ]
{ [ { PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
{ ( column [ ASC | DESC ] [ ,...n ] ) }
[WITH FILLFACTOR = fillfactor]
[ ON { filegroup | DEFAULT } ] ]
| FOREIGN KEY
[ ( column [ ,...n ] ) ]
REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ]
( search_conditions )
}

```

Ejemplo 10 Sobre la base de datos pubs de creación de una tabla

```

/* ***** jobs table *****/
CREATE TABLE jobs
(

```

⁷ SELECT * FROM ::fn_helpcollations() <- Nos muestra todas las posibles intercalaciones existentes.

	name	description
511	Modern_Spanish_BIN	Modern-Spanish, binary sort
512	Modern_Spanish_CI_AI	Modern-Spanish, case-insensitive, a...
513	Modern_Spanish_CI_AI_WS	Modern-Spanish, case-insensitive, a...
514	Modern Spanish CI AI KS	Modern-Spanish, case-insensitive, a...



```

job_id smallint
    IDENTITY(1,1)
    PRIMARY KEY CLUSTERED,
job_desc   varchar(50)  NOT NULL
    DEFAULT 'New Position - title not formalized yet',
min_lvl tinyint NOT NULL
    CHECK (min_lvl >= 10),
max_lvl tinyint NOT NULL
    CHECK (max_lvl <= 250)
)

/*
***** publishers table *****/
CREATE TABLE publishers
(
    pub_id char(4) NOT NULL
        CONSTRAINT UPKCL_pubind PRIMARY KEY CLUSTERED
        CHECK (pub_id IN ('1389', '0736', '0877', '1622', '1756')
            OR pub_id LIKE '99[0-9][0-9]'),
    pub_name   varchar(40)  NULL,
    city       varchar(20)  NULL,
    state      char(2) NULL,
    country    varchar(30)  NULL
        DEFAULT('USA')
)

-- employee table. Esta tabla se debe crear la última debido a que hace referencia a las dos anteriores
CREATE TABLE employee
(
    emp_id char
        CONSTRAINT PK_emp_id PRIMARY KEY NONCLUSTERED
        CONSTRAINT CK_emp_id CHECK (emp_id LIKE
            '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]' or
            emp_id LIKE '[A-Z]-[A-Z][1-9][0-9][0-9][0-9][0-9][0-9][FM]'),
    /* Each employee ID consists of three characters that
    represent the employee's initials, followed by a five
    digit number ranging from 10000 through 99999 and then the
    employee's gender (M or F). A (hyphen) - is acceptable
    for the middle initial. */
    fname   varchar(20)  NOT NULL,
    minit   char(1) NULL,
    lname   varchar(30)  NOT NULL,
    job_id smallint     NOT NULL
        DEFAULT 1
    /* Entry job_id for new hires. */
    REFERENCES jobs(job_id),
    job_lvl tinyint
        DEFAULT 10,
    /* Entry job_lvl for new hires. */
    pub_id char(4) NOT NULL
        DEFAULT ('9952')
    REFERENCES publishers(pub_id),
    /* By default, the Parent Company Publisher is the company
    to whom each employee reports. */
    hire_date   datetime   NOT NULL
        DEFAULT (getdate())
    /* By default, the current system date is entered. */
)

```

5.2.1.1 Restricciones:

Las restricciones de los datos se imponen para asegurarnos que los datos cumplen con una serie de condiciones predefinidas para cada tabla. Estas restricciones ayudan a conseguir la **integridad referencial**. Es decir que todas las referencias dentro de una base de datos, sean validas. Las restricciones se van a definir acompañadas por un nombre, lo que permitirá activarlas o desactivarlas según sea el caso.

Las posibles restricciones son:

Not null: será necesario que el campo que tenga esta restricción tendrá que tener un valor no nulo. Los valores nulos no ocupan espacio, y son distintos del 0 o del espacio en blanco. Habrá que tener cuidado cuando se realicen estas operaciones.

Unique: evita valores repetidos en una columna admitiendo valores nulos.

Default: establece un valor por defecto para esa columna.

Check: comprueba que se cumpla una condición determinada al llenar esa columna.

Primary key: establece el conjunto de columnas que forman la clave primaria de esa tabla. El valor será único y será obligatorio introducir un valor. El campo que sea primary key podrá ser referenciado por otras tablas como clave ajena.

Foreign key: Establece que el contenido de esta columna será uno de los valores contenidos en una columna de otra tabla maestra. No hay límite en número de claves ajenas. Se pueden forzar que cuando una fila de una tabla maestra sea borrada todas las filas de la tabla detalle cuya clave ajena coincida con la clave borrada se borre también. Esto se consigue añadiendo “on delete cascade”

5.2.1.2 Tipos de datos

[Dalton et al, 2001]

El conjunto de tipos de datos existente lo podemos ver en el explorador de objetos, en cada una de las bases de datos existentes:

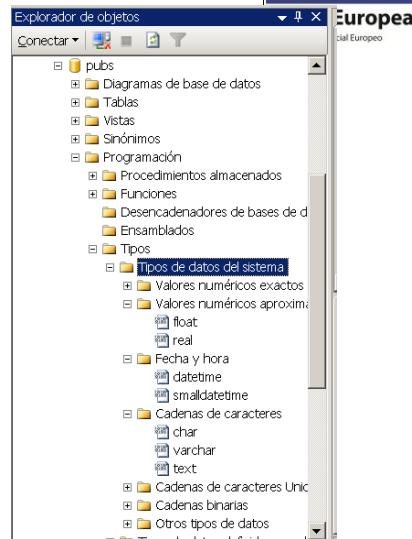
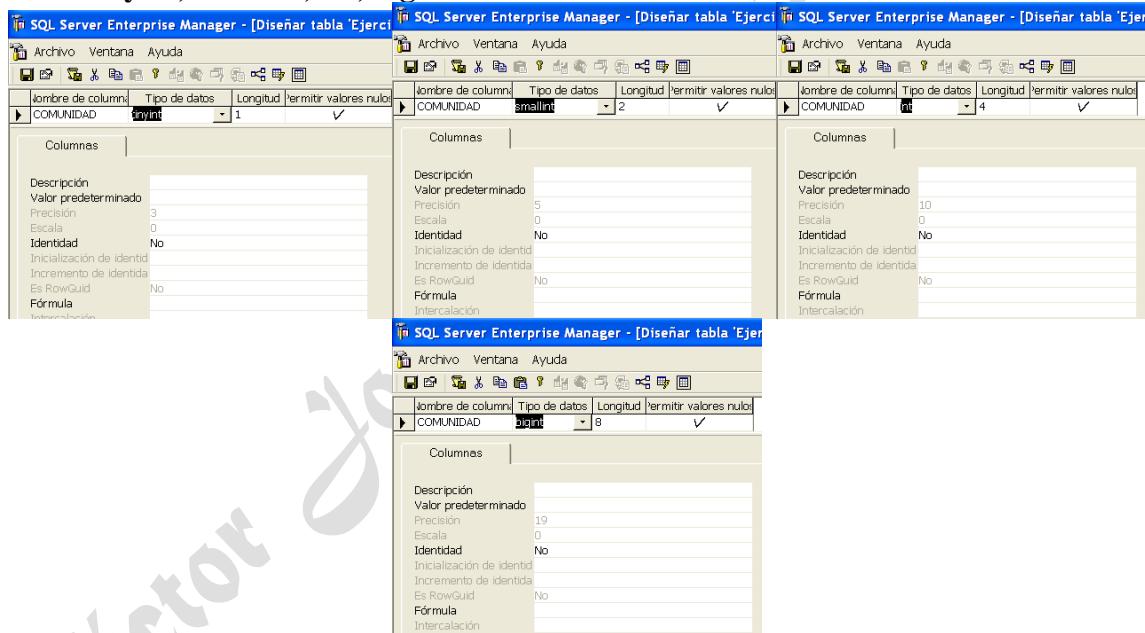


Imagen 42. Tipos de datos localizados en el explorador de objetos

- **tinyint, smallint, int, bigint**



Las capturas de pantalla muestran la definición de columnas para los tipos de datos:

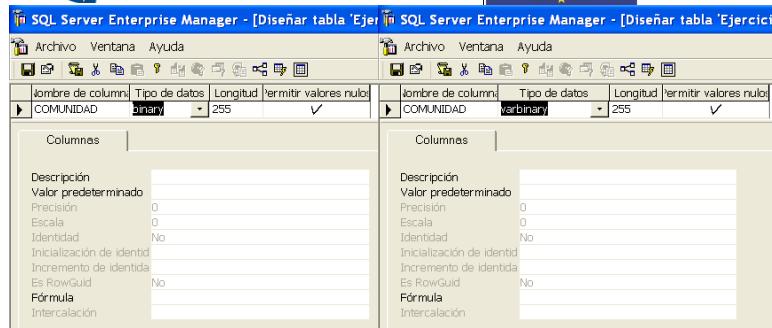
- tinyint:** Longitud: 1, Permitir valores nulos: Sí.
- smallint:** Longitud: 2, Permitir valores nulos: Sí.
- int:** Longitud: 4, Permitir valores nulos: Sí.
- bigint:** Longitud: 8, Permitir valores nulos: Sí.

Las definiciones incluyen campos como Descripción, Valor predeterminado, Precisión, Escala, Identidad, Inicialización de identidad, Incremento de identidad, Es RowGuid, Fórmula e Intercalación.

Tipo de datos	Requisitos de almacenamiento	Rango
TINYINT	1 byte	0 a 255
SMALLINT	2 bytes	-32.768 a 32.768
INT	4 bytes	-2.147.483.648 a 2.147.483.648
BIGINT	8 bytes	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.808

Tabla 1. Datos enteros en SQL Server

- **binary y varbinary**

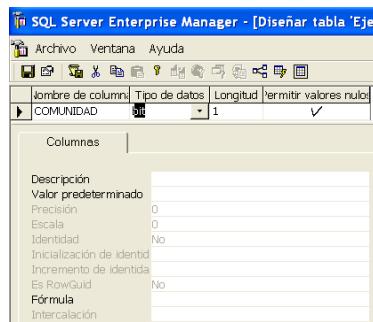



The screenshot shows two side-by-side column definitions in the SQL Server Enterprise Manager 'Diseñar tabla' (Design Table) interface. Both columns are named 'COMUNIDAD'.

- Left Column:** Data type is set to 'binary'. It has a fixed length of 255 bytes and allows null values.
- Right Column:** Data type is set to 'varbinary'. It also has a fixed length of 255 bytes and allows null values.

Tipo de dato binario que tiene una longitud fija (*binary [(n)]*) o variable (*varbinary [(n)]*), donde n puede estar comprendido entre 1 y 8000. El tamaño que ocupe al almacenar el dato será de n+4 bytes en binary y la *longitud del dato+4 bytes* en varbinary. Si no se especifican en la definición se entenderá n=1.

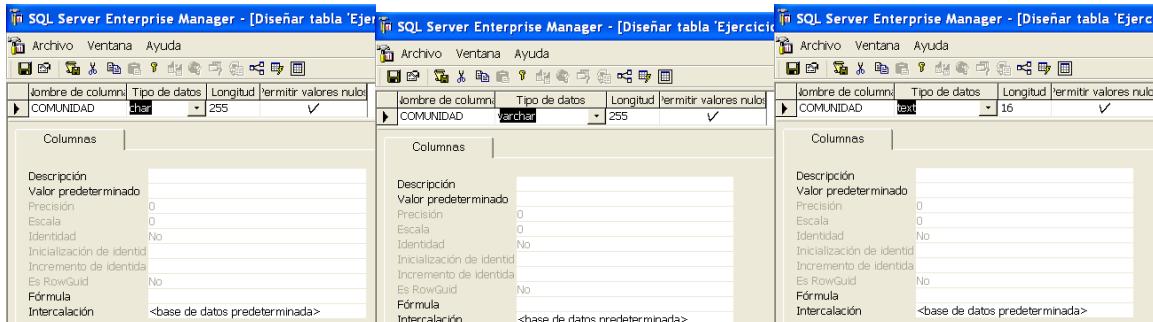
▪ bit



The screenshot shows a single column definition in the SQL Server Enterprise Manager 'Diseñar tabla' interface. The column is named 'COMUNIDAD' and its data type is set to 'bit'.

Almacenará como valores posibles: 1, 0 o NULL. Si en la tabla existen 8 columnas o menos de tipo de dato bit ocuparán todas ellas 1 byte, si hay entre 9 y 16 2 bytes y así sucesivamente.

▪ char, varchar, text



The screenshot shows three side-by-side column definitions in the SQL Server Enterprise Manager 'Diseñar tabla' interface.

- Left Column:** Data type is set to 'char'. It has a fixed length of 255 bytes and allows null values.
- Middle Column:** Data type is set to 'varchar'. It has a fixed length of 255 bytes and allows null values.
- Right Column:** Data type is set to 'text'. It has a fixed length of 16 bytes and allows null values.

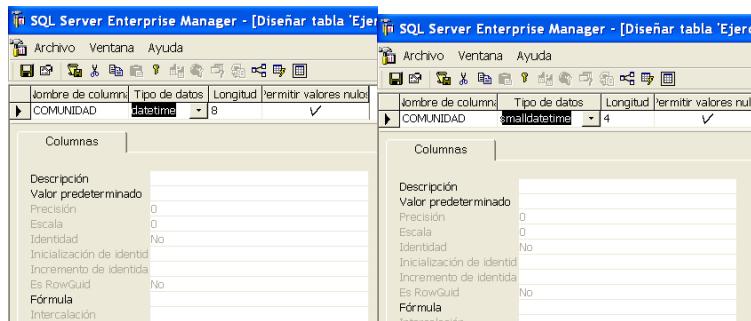
Almacenan números, letras (mayúsculas o minúsculas) y caracteres especiales (@, &, !.....). Longitud máxima 8000 caracteres, unos 8 KBytes. El tipo de datos **char** es un tipo de datos de **longitud fija** cuando se especifica la cláusula NOT NULL. Si en una columna **char** NOT NULL se inserta un valor más corto que la longitud de la columna, el valor se rellena a la derecha con blancos hasta completar el tamaño de la columna. Por ejemplo, si una columna se define como **char(10)** y el dato que se va a almacenar es "música", SQL Server almacena este dato como "música_____".

variable pero inferior a 8Kbytes. Cuando el tamaño supere los 8Kbytes, utilizaremos text, el tamaño máximo de este campo será de hasta 2 GB.

- **nchar, ntext, nvarchar**

Los tipos de datos anteriores registraban caracteres ASCII, con estos otros tipos de datos se utilizarán caracteres UNICODE UCS-2, que fundamentalmente se utiliza para representación de otros lenguajes. Ej.: la ñ es un carácter unicode, palabras acentuadas. El nombre proviene de **national char**, **national text**...

- **datetime, smalldatetime**



Se trata de tipos de datos para representar fecha y hora. En **datetime** se almacenarán datos entre el 1 de enero de 1753 hasta el 31 de diciembre de 9999 con una precisión de 3,33 milisegundos o 0,00333 segundos. Ej.: 01/01/98 23:59:59.997 Tamaño: 4 bytes. En **smalldatetime** se podrá almacenar desde el 1 de enero de 1900 hasta el 6 de junio de 2079, con una precisión de minutos. Tamaño: 2 bytes.

Formatos válidos para especificar fechas:

Apr[il] [15][,] 1996
 Apr[il] 15[,] [19]96
 Apr[il] 1996 [15]
 [15] Apr[il][,] 1996
 15 Apr[il][,][19]96
 15 [19]96 apr[il]
 [15] 1996 apr[il]
 1996 APR[IL] [15]
 1996 [15] APR[IL]

Función útil: `SET DATEFORMAT { format | @format_var }`
 Definimos el formato de la fecha: mdy, dmy, ymd, ydm, myd y dyd

Ej.:

```
-- Set date format to month, day, year.  

SET DATEFORMAT mdy;  

GO  

DECLARE @datevar DATETIME;  

SET @datevar = '12/31/1998';  

SELECT @datevar AS DateVar;  

GO
```

```
-- Set date format to year, day, month.  
SET DATEFORMAT ydm;  
GO  
DECLARE @datevar DATETIME;  
SET @datevar = '1998/31/12';  
SELECT @datevar AS DateVar;  
GO
```

```
-- Set date format to year, month, day.  
SET DATEFORMAT ymd;  
GO  
DECLARE @datevar DATETIME;  
SET @datevar = '1998/12/31';  
SELECT @datevar AS DateVar;  
GO
```

Al instalar SQL Server, se instala también la base de datos master, y en tablas del sistema de esta BD aparece la tabla syslanguages. Aquí podemos ver qué formato de fecha se está utilizando.

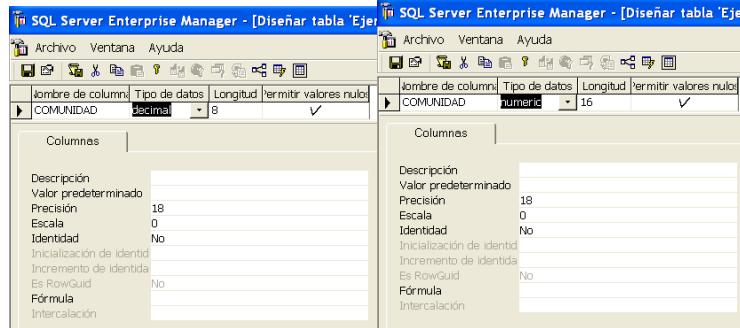
```
SELECT * FROM master.dbo.syslanguages
```

l...	dateformat	dat...	u...	name	alias	months	shortmonths	days	lcid	ms
0	mdy	7	0	us_english	English	January,Februa...	Jan, Feb,...	Monday,Tuesday...	1033	10
1	dmy	1	0	Deutsch	German	Januar,Februar...	Jan, Feb,...	Montag,Dienst...	1031	10
2	dmy	1	0	Français	French	janvier,février...	janv,fév...	lundi,mardi,m...	1036	10
3	ymd	7	0	日本語	Japanese	01,02,03,04,05...	01,02,03...	日曜,日,月,火,...	1041	10
4	dmy	1	0	Dansk	Danish	januar,februar...	jan, feb,...	mandag,tirsd...	1030	10
5	dmy	1	0	Español	Spanish	Enero,Febrero,...	Ene,Feb,...	Lunes,Martes,...	3082	30
6	dmy	1	0	Italiano	Italian	gennaio,febbra...	gen,feb,...	lunedì,marted...	1040	10
7	dmy	1	0	Nederlands	Dutch	januari,februa...	jan, feb,...	maandag,dinsd...	1043	10
8	dmy	1	0	Norsk	Norwegian	januar,februar...	jan, feb,...	mandag,tirsd...	2068	20
9	dmy	7	0	Português	Portuguese	janeiro,fevere...	jan, fev,...	segunda-feira...	2070	20
10	dmy	1	0	Suomi	Finnish	tammikuu-talvi	tammi-hu	maanantai-tii...	1035	10

```
SELECT * FROM sales
```

	stor_id	ord_num	ord_date	qty	payterms	title_id
1	6380	6871	1994-09-14 00:00:00.000	5	Net 60	BU1032
2	6380	722a	1994-09-13 00:00:00.000	3	Net 60	PS2091
3	7066	A2976	1993-05-24 00:00:00.000	50	Net 30	PC8888
4	7066	QA7442.3	1994-09-13 00:00:00.000	75	ON invoice	PS2091
5	7067	B4482	1994-09-14 00:00:00.000	10	Net 60	PS2091

- decimal y numeric

The screenshot shows two side-by-side windows in SQL Server Enterprise Manager. Both windows are titled 'Diseñar tabla 'Ejer''. The left window is for the 'decimal' type, and the right window is for the 'numeric' type. Both windows have columns for 'Nombre de columna', 'Tipo de datos', 'Longitud', and 'Permitir valores nulos'. In both cases, 'COMUNIDAD' is the column name, 'decimal' is the type, '16' is the length, and 'V' (checked) allows null values. Below these columns are detailed settings for precision, scale, identity, and rowguid.

`decimal[(p[, s])]` y `numeric[(p[, s])]` donde p es la precisión y s la escala.

P: Especifica el número máximo total de dígitos decimales que se pueden almacenar, **a la izquierda y a la derecha** del separador decimal. La precisión debe ser un valor comprendido entre 1 y la precisión máxima. La mayor precisión que se puede especificar es 38.

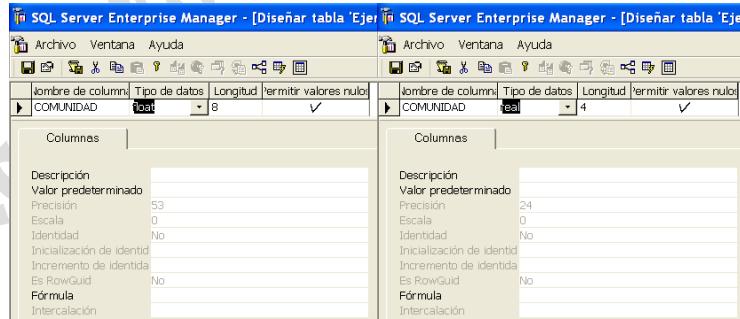
S: Especifica el número máximo de dígitos decimales que se pueden almacenar **a la derecha** del separador decimal. La escala debe ser un valor comprendido entre 0 y p . La escala predeterminada es 0; por tanto, $0 \leq s \leq p$. Los tamaños máximos de almacenamiento varían según la precisión.

Precisión	Bytes de almacenamiento
1 - 9	5
10-19	9
20-28	13
29-38	17

Nota.- `decimal(5,5)` y `decimal(5,0)` se consideran tipos de datos diferentes.

Ambos son datos equivalentes.

- **Float y real**



The screenshot shows two side-by-side windows in SQL Server Enterprise Manager. Both windows are titled 'Diseñar tabla 'Ejer''. The left window is for the 'float' type, and the right window is for the 'real' type. Both windows have columns for 'Nombre de columna', 'Tipo de datos', 'Longitud', and 'Permitir valores nulos'. In both cases, 'COMUNIDAD' is the column name, 'float' is the type, '8' is the length, and 'V' (checked) allows null values. Below these columns are detailed settings for precision, scale, identity, and rowguid.

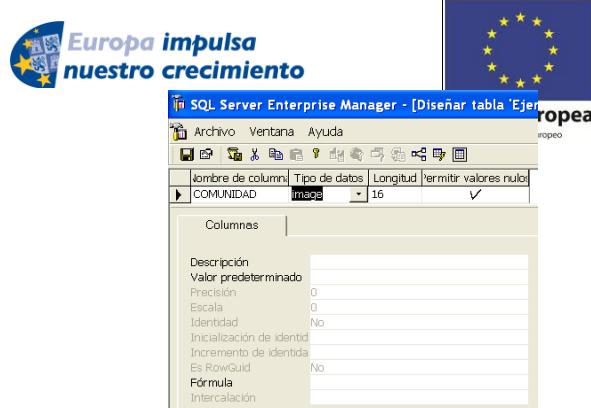
Se encarga de almacenar un número en punto flotante: $-1.79E + 308$ y $1.79E + 308$.

308. n

n is	Precisión	Tamaño de almacenamiento
1-24	7 cifras	4 bytes
25-53	15 cifras	8 bytes

La equivalencia entre los dos tipos de datos es : `real=float(24)`

- **image**



Podremos almacenar imágenes en formatos BMP, TIFF, GIF o JPEG. Realmente son datos binarios de longitud variable desde 0 hasta $2^{31}-1$ (2.147.483.647) bytes, son similares por lo tanto a binary y varbinary. Normalmente en una base de datos no se insertan imágenes en ella, simplemente se utilizan rutas relativas del disco duro para posteriormente cargarlas en las diferentes aplicaciones. Se dice que crean mucha fragmentación en la BD.

- **money, smallmoney**

Nombre de columna	Tipo de datos	Longitud	Permitir valores nulos
COMUNIDAD	money	8	✓
COMUNIDAD	smallmoney	4	✓

money: valores de moneda comprendidos entre -2^{63} (-922.337.203.685.477,5808) y $2^{63} - 1$ (+922.337.203.685.477,5807), con una precisión de una diezmilésima de la unidad monetaria. Tamaño de almacenamiento 8 bytes.

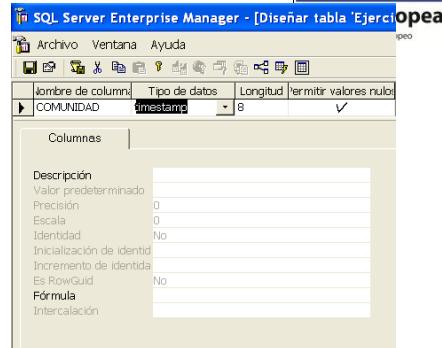
smallmoney: valores de moneda comprendidos entre -214.748,3648 y +214.748,3647, con una precisión de una diezmilésima de la unidad monetaria. Tamaño de almacenamiento 4 bytes.

- **sql_variant**

Nombre de columna	Tipo de datos	Longitud	Permitir valores nulos
COMUNIDAD	sql_variant	0	✓

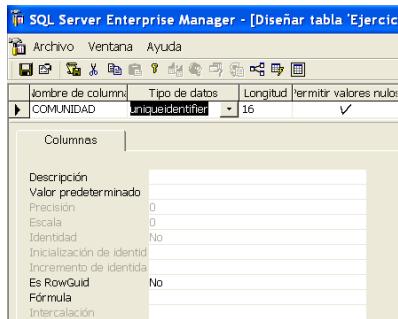
Se trata de un tipo de datos que almacena valores de varios tipos de datos aceptados en SQL Server, excepto text, ntext, image, timestamp y sql_variant. Por lo tanto una columna del tipo sql_variant puede contener filas de tipos de datos diferentes.

- **timestamp**



Es un tipo de datos que expone automáticamente números binarios generados, cuya exclusividad está garantizada en la base de datos. timestamp se suele utilizar como mecanismo para marcar la versión de las filas de la tabla. El tamaño de almacenamiento es de 8 bytes.

■ **uniqueidentifier**



El tipo de datos uniqueidentifier almacena valores binarios de 16 bytes que operan como identificadores exclusivos globales (GUID). Un GUID es un número binario exclusivo; ningún otro equipo del mundo generará un duplicado de ese GUID. El principal uso de un GUID se da cuando se asigna un identificador que debe ser exclusivo en una red que tiene muchos equipos en distintos emplazamientos.

Lo genera a partir del identificador de su tarjeta de red (MAC) más un número exclusivo del reloj de la CPU.

Ejemplo 11. Inserción de un nuevo valor sobre el campo uniqueidentifier:

```
USE AdventureWorks;
GO
IF OBJECT_ID ('dbo.T1', 'U') IS NOT NULL
    DROP TABLE dbo.T1;
GO
CREATE TABLE dbo.T1
(
    column_1 int IDENTITY,
    column_2 uniqueidentifier,
);
GO
INSERT INTO dbo.T1 (column_2)
    VALUES (NEWID());
INSERT INTO T1 DEFAULT VALUES;
GO
SELECT column_1, column_2
FROM dbo.T1;
GO
```

- **XML**

Este tipo de datos se ha introducido en esta nueva versión de SQL Server 2008 dada su alta integración con este tipo de ficheros y las elevadas posibilidades de comunicación actualmente existentes con ficheros XML. Permite almacenar información en este formato de hasta 2 GB.

Véase *Ejemplo 89. Registro de las creaciones de tabla realizadas sobre la BD. Pág. 159. Donde se almacena en una tabla eventos en formato xml, para recuperar los datos concretos de una etiqueta del XML utilizamos **nombrecampo.query('data(//nombreetiqueta)')**.*

Los tipos de datos XML tienen como responsabilidad el garantizar que los documentos XML están bien formados, es decir, que son sintácticamente correctos.

- **Tipos de datos personalizados (definido por el usuario).** Este tipo de datos se agregará con la función `sp_addtype` o bien más fácilmente con el administrador corporativo.

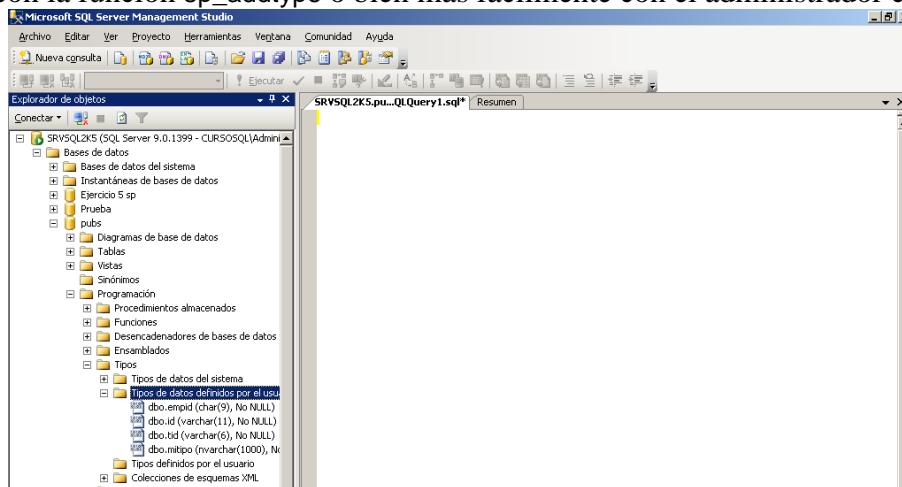


Imagen 43. Los nuevos tipos empid, id y tid únicamente varían char o varchar en su longitud

Estos tipos de datos no aportan realmente demasiada funcionalidad, los tipos de datos generados a partir de clases serían realmente útiles. SQL Server 2008 sí permite la creación de este tipo de datos, y almacenar así objetos en un campo BD, pero se realiza mediante la definición de clases en C# o Visual Basic .net.

Ejemplo 12. Ejemplo de uso de un tipo de datos creado en C#

```

CREATE ASSEMBLY TipoPunto
    FROM 'c:\tipos\Punto.dll'
GO

CREATE TABLE Puntos
{
    idPunto INT NOT NULL IDENTITY(1,1),
    ElPunto TipoPunto
}

SELECT *, ElPunto.m_x, ElPunto.m_y

```

```

        FROM Puntos
GO

UPDATE * Puntos SET ElPunto.SetXY(20,30)
WHERE ElPunto.m_x=0

DECLARE @Pnt TipoPunto
SET @Pnt=(SELECT Pnt FROM Puntos
           WHERE ID=2)

SELECT @Pnt.ToString() AS Pnt
GO

```

Ejemplo 13. Despu s de la creaci n de una tabla con todos los tipos de datos realizaremos diferentes inserciones en ella:

```

CREATE TABLE [dbo].[TiposDatos](
    bigint bigint NULL,
    binary50 binary(50) NULL,
    bit bit NULL,
    char10 char(10) COLLATE Modern_Spanish_CI_AS NULL,
    datetime datetime NULL,
    decimal182 decimal(18, 2) NULL,
    float float NULL,
    image image NULL,
    int int NULL,
    money money NULL,
    nchar10 nchar(10) COLLATE Modern_Spanish_CI_AS NULL,
    ntext ntext COLLATE Modern_Spanish_CI_AS NULL,
    numeric182 numeric(18, 2) NULL,
    nvarchar50 nvarchar(50) COLLATE Modern_Spanish_CI_AS NULL,
    real real NULL,
    smalldatetime smalldatetime NULL,
    smallint smallint NULL,
    smallmoney smallmoney NULL,
    sql_variant sql_variant NULL,
    text text COLLATE Modern_Spanish_CI_AS NULL,
    timestamp timestamp NULL,
    tinyint tinyint NULL,
    uniqueidentifier uniqueidentifier NULL,
    varbinary50 varbinary(50) NULL,
    varbinaryMAX varbinary(max) NULL,
    varchar50 varchar(50) COLLATE Modern_Spanish_CI_AS NULL,
    xml xml NULL
)
INSERT INTO [PRUEBA].[dbo].[TiposDatos]
([bigint],[binary50],[bit],[char10],[datetime],[decimal182],[float],[image]
,[int],[money],[nchar10],[ntext],[numeric182],[nvarchar50],[real]
,[smalldatetime],[smallint],[smallmoney],[sql_variant],[text]
,[tinyint],[uniqueidentifier],[varbinary50],[varbinaryMAX]
,[varchar50],[xml])
VALUES
(9223372036854775807
, 0x4F --suser_sid() -- Probar con 0x4F
,1 --bit
,'  char ' --<char10, char(10),>
, '1998/12/31 22:00'--<datetime, datetime,>
,9999.8 --<decimal182, decimal(18,2),>

```



Junta de
Castilla y León



```

,1.79E + 308 --<float, float,>
,'c:\Pescar.bmp' --<image, image,>
,2147483647 -- <int, int,>
,-922337203685477.5808 --<money, money,>
,'ácharñ' --<nchar10, nchar(10),>
,'abcdefghijklmnoprstuvwxyz' -- <ntext, ntext,>
,9999.8 --<numeric182, numeric,>
,'Leña' --<nvarchar50, nvarchar(50),>
, 1.79E + 308--<real, real(24,0),>
, '01/01/98 23:59:59.997' --<smalldatetime, smalldatetime,>
,32767 --<smallint, smallint,>
,214748.3647 --<smallmoney, smallmoney,>
,'cualquier cosa' --<sql_variant, sql_variant,>
,'Cadena de texto grande.ñññ' --<text, text,>
,255 --<tinyint, tinyint,>
,newid() --<uniqueidentifier, uniqueidentifier,>
,0x4F --<varbinary50, varbinary(50),>
,0x4F --<varbinaryMAX, varbinary(max),>
,'Esto es una cadena' --<varchar50, varchar(50),>
, null --<xml, xml,>
)
)

```

select * from tiposdatos



5.2.1.2.1 *Tipo table*

TABLE ({ column_definition | table_constraint } [,...n])

Un tipo especial de datos que puede utilizarse para almacenar un conjunto de resultados y procesarlo más adelante. Su uso principal es el almacenamiento temporal de un conjunto de filas, que se van a devolver como el conjunto de resultados de una función valorada en tabla.

Ejemplo 14.

USE pubs

```

DECLARE @TablaLibros table
(Titulo varchar(50) PRIMARY KEY,
Precio money,
Fecha datetime)

DECLARE @vartitulo VARCHAR(50)
DECLARE @varprecio MONEY

DECLARE cursortitulos cursor FOR
SELECT title, price
From titles

OPEN cursortitulos
Fetch next from cursortitulos
Into @vartitulo, @varprecio

While @@fetch_status=0
Begin
    INSERT @TablaLibros VALUES (@vartitulo, @varprecio, getdate())
    Fetch next from cursortitulos Into @vartitulo, @varprecio

```

Deallocate cursortitulos

Select * from @tablalibros

Ejercicio 7. Realizar en DML las siguientes tablas:

Tabla dept

deptno - Número de departamento. Numérico de 2 dígitos. No admite nulos. Clave primaria. PRIMARY KEY.

dname - Nombre del departamento. Texto de 14 caracteres.

loc – Localidad a la que pertenece. Texto de 13 caracteres, que puede tomar como valores Valladolid, Soria o Palencia.

Tabla emp

empno – Número de empleado. Número de 4 dígitos. No admite nulos. PRIMARY KEY.

ename – Nombre del empleado. Texto 10 caracteres.

job – Puesto de trabajo. Texto de 9 caracteres.

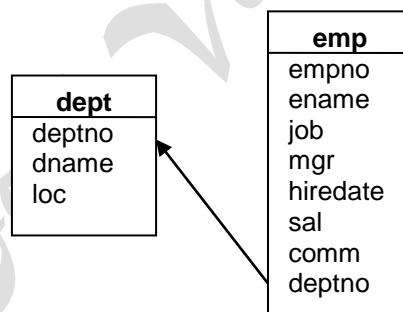
mgr – Número de empleado que es su jefe. Número de 4 dígitos.

hiredate – Contratado desde la fecha indicada. Tipo fecha.

sal – Salario. Tipo moneda.

comm – Complementos asalariados Tipo moneda.

deptno – Departamento al que pertenece. Número de departamento al que pertenece. Número de 2 dígitos. FOREIGN KEY.



Ejercicio 8. Modificar la base de datos del Ejercicio 7 de la siguiente manera.

* Copia la BD en una nueva BD llamada Ejercicio 4 departamentos.

* Añadir a emp un nuevo campo llamado salneto en el que se almacene la suma del salario más los complementos menos un 7% de IRPF sobre sal + comm.

* El salario debe ser mínimo 600 €.

* El puesto de trabajo del empleado (campo job) sólo puede ser SALESMAN, CLERK, ANALYST ó MANAGER.

Ejercicio 9. Consideremos la base de datos EMPLEADOS que contiene información correspondiente a una sencilla aplicación de procesamiento de pedidos para una pequeña empresa de distribución. Consta de cinco tablas:

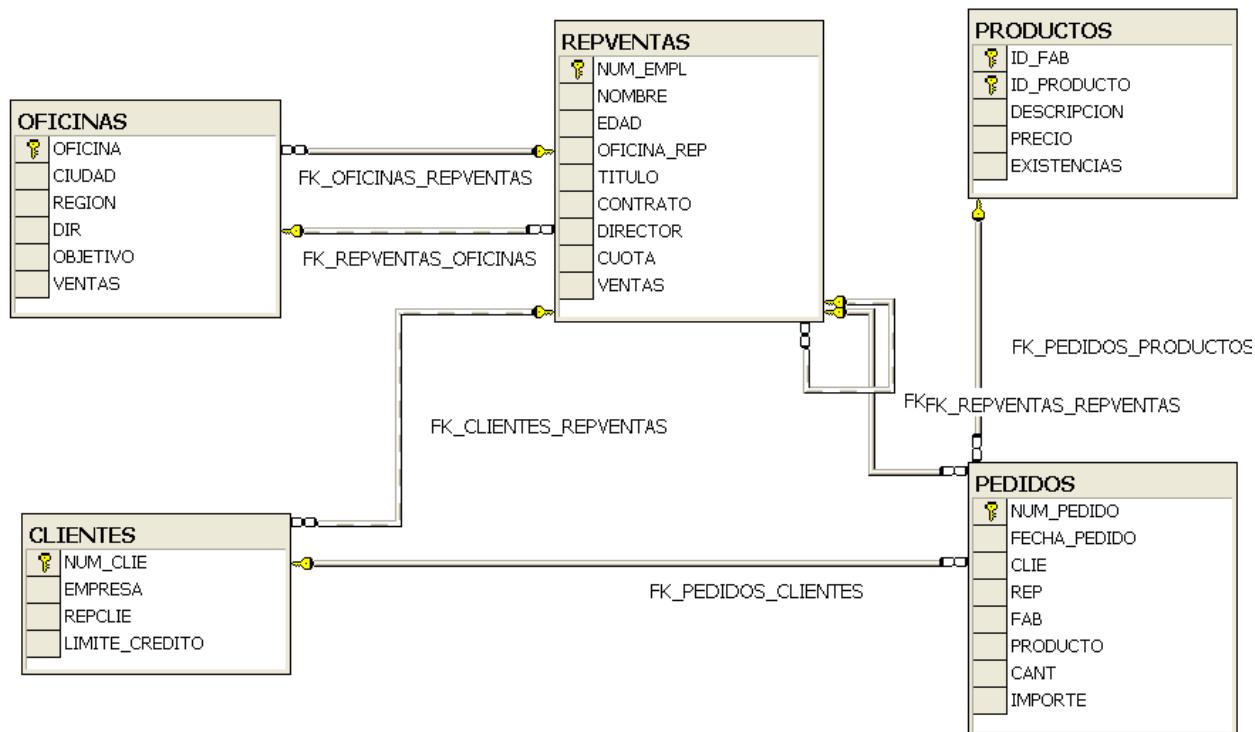
- **CLIENTES**, que contiene una fila por cada uno de los clientes de la empresa. Sus campos son NUM_CLIE (número de cliente), EMPRESA, REP_CLIENTE (número de empleado del representante que atiende al cliente) y LIMITE_CREDITO (límite de crédito).

- **REPVENTAS**, que contiene una fila por cada uno de los diez vendedores de la empresa. Sus campos son NUM_EMPL (número de empleado), NOMBRE, EDAD, OFICINA REP

(número de oficina de representación del vendedor), TITULO, CONTRATO, DIRECTOR, CUOTA (ventas previstas) y VENTAS (ventas realizadas).

- **OFICINAS**, que contiene una fila por cada una de las cinco oficinas en las que trabajan los vendedores. Sus campos son OFICINA (número de oficina), CIUDAD, REGION, DIR (número de empleado del director), OBJETIVO (ventas anuales previstas) y VENTAS (ventas anuales realizadas).
- **PRODUCTOS**, que contiene una fila por cada tipo de producto disponible para la venta. Sus campos son ID_FAB (identificador de fabricante), ID_PRODUCTO (identificador de producto), DESCRIPCION, PRECIO y EXISTENCIA.
- **PEDIDOS**, que contiene una fila por cada pedido ordenado por un cliente. Por simplicidad, se supone que cada pedido se refiere a un sólo producto. Sus campos son NUM_PEDIDO (número de pedido), FECHA_PEDIDO (fecha del pedido), CLIE (número de cliente), REP (número de empleado del representante), FAB (identificador de fabricante), PRODUCTO (identificador de producto), CANT (cantidad) e IMPORTE.

Realizar un diseño apropiado para las cinco tablas y las relaciones padre/hijo entre las columnas que contienen.



5.2.2 CREATE INDEX

```

CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
    ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )
    [ WITH < index_option > [ ,...n ] ]
    [ ON filegroup ]
    < index_option > ::=
        { PAD_INDEX |
            FILLFACTOR = fillfactor |
            IGNORE_DUP_KEY |
            DROP_EXISTING |
            STATISTICS_NORECOMPUTE |
    
```

5.2.3 DROP INDEX

```
DROP INDEX 'table.index | view.index' [ ,...n ]
```

5.2.4 ALTER TABLE

```
ALTER TABLE table
{[ ALTER COLUMN column_name
  { new_data_type [ ( precision [ , scale ] ) ]
    [ COLLATE < collation_name > ]
    [ NULL | NOT NULL ]
    | {ADD | DROP } ROWGUIDCOL }
  ]
  | ADD
    { [ < column_definition > ]
      | column_name AS computed_column_expression
    } [ ,...n ]
  | [ WITH CHECK | WITH NOCHECK ] ADD
    { < table_constraint > } [ ,...n ]
  | DROP
    { [ CONSTRAINT ] constraint_name
      | COLUMN column } [ ,...n ]
  | { CHECK | NOCHECK } CONSTRAINT
    { ALL | constraint_name [ ,...n ] }
  | { ENABLE | DISABLE } TRIGGER
    { ALL | trigger_name [ ,...n ] }
}
```

5.2.5 DROP TABLE

Eliminamos la tabla.

```
DROP TABLE table_name
```

5.2.6 CREATE VIEW

```
CREATE VIEW [ < database_name > . ] [ < owner > . ] view_name [ ( column [ ,...n ] ) ]
[ WITH < view_attribute > [ ,...n ] ]
AS
select_statement
[ WITH CHECK OPTION ]

< view_attribute > ::= 
{ ENCRYPTION | SCHEMABINDING | VIEW_METADATA }
```

Utilización del asistente del administrador corporativo.

Puede ser útil con la función CURRENT_USER en el WHERE de la vista, de esta manera haríamos que un usuario sólo viera los datos suyos, suponiendo que en la tabla

5.2.7 ALTER VIEW

```
ALTER VIEW [ < database_name > . ] [ < owner > . ] view_name [ ( column [ ,...n ] ) ]
[ WITH < view_attribute > [ ,...n ] ]
AS
    select_statement
[ WITH CHECK OPTION ]

< view_attribute > ::= 
{ ENCRYPTION | SCHEMABINDING | VIEW_METADATA }
```

5.2.8 DROP VIEW

```
DROP VIEW { view } [ ,...n ]
```

5.2.9 CREATE TRIGGER

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
    { { FOR | AFTER | INSTEAD OF } { [ INSERT ][ , ][ UPDATE ] }
        [ WITH APPEND ]
        [ NOT FOR REPLICATION ]
        AS
            [ { IF UPDATE ( column )
                [ { AND | OR } UPDATE ( column ) ]
                [ ...n ]
            | IF ( COLUMNS_UPDATED () { bitwise_operator } updated_bitmask )
                { comparison_operator } column_bitmask [ ...n ]
            }
            sql_statement [ ...n ]
        }
    }
}
```

5.2.10 ALTER TRIGGER

```
ALTER TRIGGER trigger_name
ON ( table | view )
[ WITH ENCRYPTION ]
{
    { ( FOR | AFTER | INSTEAD OF ) { [ DELETE ][ , ][ INSERT ][ , ][ UPDATE ] }
        [ NOT FOR REPLICATION ]
        AS
            sql_statement [ ...n ]
    }
    | { ( FOR | AFTER | INSTEAD OF ) { [ INSERT ][ , ][ UPDATE ] }
        [ NOT FOR REPLICATION ]
```

```

AS
{ IF UPDATE ( column )
[ { AND | OR } UPDATE ( column ) ]
[ ...n ]
| IF ( COLUMNS_UPDATED ( ) { bitwise_operator } updated_bitmask )
{ comparison_operator } column_bitmask [ ...n ]
}
sql_statement[ ...n ]
}
}

```

5.2.11 DROP TRIGGER

DROP TRIGGER { trigger } [,...n]

Ejercicio 10. Copia la base de datos Ejercicio 7 departamentos a Ejercicio 10 departamentos. Modificar la tabla EMP definiendo y habilitando en la columna sal un formato numérico de precisión 7 y escala 2 y la restricción NOT NULL. A continuación, modificar la tabla DEPT, con campos deptno (numeric() de anchura 2), dname (CHAR de anchura 9) y loc (CHAR de anchura 10) y habilitar como clave (única de nombre).

2. Crear la tabla dept2 con los mismos campos que la tabla deptl, pero añadiendo Como clave primaria de nombre pk_deptl el campo dept.
3. Crear la tabla EMP3 con los mismos campos que la tabla EMP, de modo que la columna deptno tenga coma clave externa (de nombre fk_deptno) la clave primaria deptno de la columna dept de la tabla EMP.
4. Añadir a la tabla EMP3 la clave única compuesta por las dos columnas sal y comm, nombrando dicha clave mediante unq_sal_comm.
5. Borrar la tabla EMP3 de la base de datos Ejercicio 10 departamentos.

5.3 DML

[Dalton et al, 2001b]

5.3.1 Select

```

SELECT [ ALL | DISTINCT ] select_list
[ INTO new_table ]
[FROM table_name | view_name| alias]
[ INNER | { { LEFT | RIGHT | FULL } [ OUTER ] } ] JOIN table_name | view_name ON
join_condition )
[ WHERE search_condition
[ GROUP BY group _ by_ expression ] [ HAVING search_ condition ]
[ ORDER BY order expression [ ASC | DESC ] ]
[ COMPUTE aggregate_ functions ]
[ FOR { BROWSE | XML [ RAW | AUTO | EXPLICIT ]
[ , XMLDATA ]
[ , ELEMENTS ]
[ , BINARY BASE64 ]
} ]

```

Para acceder al nombre de una tabla podremos usar la notación punto siguiente:
servidor.basededatos.esquema.tabla

Ejemplos:

Vamos a trabajar con ejemplos que vienen por defecto en una base de datos llamada pubs de SQL Server 2000, las tablas principales con las que trabajaremos se llaman Sales (ventas), Titleauthor (títulos de los libros de un autor) y Authors (autores). Los campos de cada tabla aparecen en las tablas inferiores.

En la tabla titleauthor existirá un registro por cada uno de los posibles autores de un libro, indicando el número de autor que lo ha escrito en el campo au_id y el porcentaje de ganancias que se llevará con la venta del libro en el campo royaltyper. En au_ord aparecerá, de ese libro, el número de autores que ha tenido.

titleauthor

Clave	Campo	Tipo de datos	Longitud	Permitir nulos
#	au_id -Identificador autor	id (varchar)	11	0
#	title_id -Identificador título	tid (varchar)	6	0
	au_ord -Nº de escritor	tinyint	1	1
	royaltyper -% de beneficio	int	4	1

En sales se almacenará información sobre las ventas de cada libro en cada comercio. Tendremos un único registro por cada línea de ticket, y en cada línea de ticket aparecerá el libro y las unidades vendidas de él.

sales

Clave	Campo	Tipo de datos	Longitud	Permitir nulos
#	stor_id -Id de la tienda	char	4	0
#	ord_num -Nº factura	varchar	20	0
	ord_date -Fecha factura	datetime	8	0
	qty -Cantidad vendida	smallint	2	0
	payterms -Forma de pago	varchar	12	0
#	title_id -Id. libro vendido	tid (varchar)	6	0

En la tabla authors tendremos un registro por cada autor de un libro con sus datos personales.

authors

Clave	Campo	Tipo de datos	Longitud	Permitir nulos
#	au_id -Id autor	id (varchar)	11	0
	au_lname -Nombre	varchar	40	0
	au_fname -Primer apellido	varchar	20	0
	phone -Teléfono	char	12	0
	address -Dirección	varchar	40	1
	city -Ciudad	varchar	20	1
	state -Estado	char	2	1
	zip -Código postal	char	5	1
	Contract -Contrato	bit	1	0

El esquema completo de la BD es:

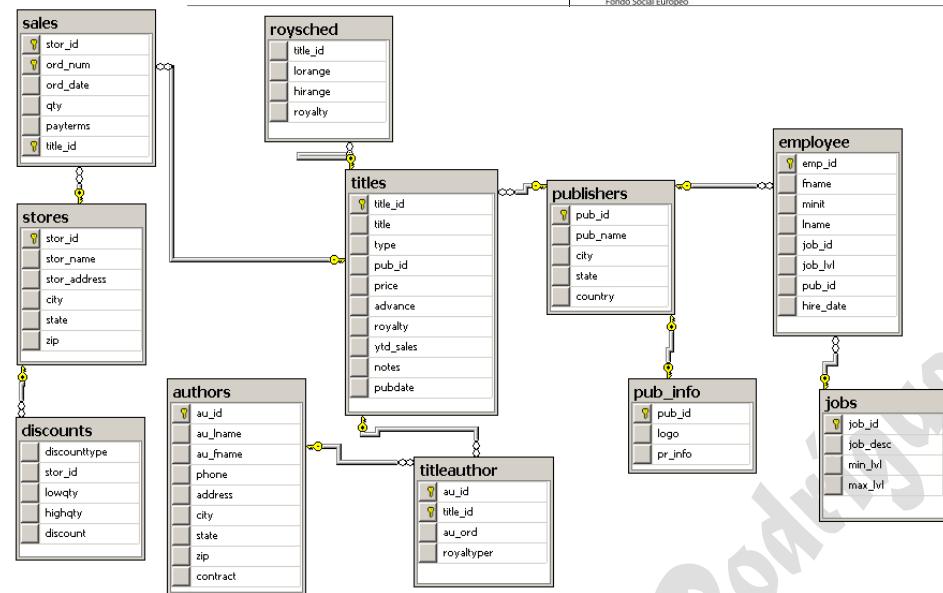


Imagen 44. Esquema de BD pubs

Ejemplo 15. Consultas básicas de SQL

/* Query 1 */

```
SELECT * FROM authors
```

/* Query 2 */

```
SELECT au_id, au_lname, au_fname, address, city, state, zip, phone, contract FROM authors
```

/* Query 3 (Formatted) */

```
SELECT *
FROM authors
```

/* Query 4 (Formatted) */

```
SELECT au_id,
       au_lname,
       au_fname,
       address,
       city,
       state,
       zip,
       phone,
       contract
  FROM authors
```

	au_id	au_lname	au_fname	phone	address	city	state	zip	contract
1	172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	94025	1
2	213-46-8915	Green	Marjorie	415 986-7020	309 43rd St. #411	Oakland	CA	94618	1
3	156-95-7766	Carmos	Cheryl	415 549-7723	580 Darwin Ln.	Berkeley	CA	94705	1
4	267-41-2394	O'Leary	Michael	408 284-2422	22 Cleveland Av. #14	San Jose	CA	95128	1
5	274-60-9391	Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA	94609	1
6	241-22-2782	Smith	Wendell	913 842-0462	10 Mississippi Dr.	Lawrence	KS	66044	0
7	409-56-7000	Bennet	Abraham	415 450-9932	4223 Bateman St.	Berkeley	CA	94705	1
8	427-17-2319	Bull	Ann	415 936-7128	3410 Blonde St.	Palo Alto	CA	94301	1
9	472-27-2349	Gringlasby	Burt	707 930-6445	PO Box 792	Covelo	CA	95428	1
10	486-29-1786	Lockley	Charlene	415 585-4620	18 Broadway Av.	San Francisco	CA	94130	1
11	527-72-3246	Greene	Bonnie	615 297-2723	22 Owings House Rd.	Nashville	TN	37215	0
12	648-07-1873	Blanchard-Walls	Reverend	401 744-2445	44 W. Main St.	Montgomery	AL	36104	1

/* Query 5 (Formatted) */

```
SELECT au_lname,
       au_fname,
       address,
       city,
       state,
       zip
  FROM authors
```

Ejemplo 16. Ejemplo de lista de teléfono



/* Query 1 (name and phone number) */

```
SELECT au_fname,
       au_lname,
       phone
  FROM authors
```

	au_fname	au_lname	phone
1	Johnson	White	408 496-7223
2	Marjorie	Green	415 986-7020
3	Cheryl	Carson	415 548-7723
4	Michael	O'Leary	408 286-2428
5	Dean	Straight	415 834-2919
6	Meander	Smith	913 843-0462
7	Abrraham	Bennet	415 658-9932
8	Ann	Dull	415 836-7128
9	Burt	Gringlesby	707 938-6445
10	Charlene	Locksley	415 585-4620
11	Morningstar	Greene	615 297-2723
12	Benignald	Blanchard-Hallie	503 745-6407

/* Query 2 (first initial, last name, phone number) */

```
SELECT UPPER(LEFT(au_fname, 1)) + '!',
       au_lname,
       phone
  FROM authors
```

	(Sin nombre de columna)	au_lname	phone
1	J.	White	408 496-7223
2	M.	Green	415 986-7020
3	C.	Carson	415 548-7723
4	M.	O'Leary	408 286-2428
5	D.	Straight	415 834-2919
6	M.	Smith	913 843-0462
7	A.	Bennet	415 658-9932
8	A.	Dull	415 836-7128
9	B.	Gringlesby	707 938-6445
10	C.	Locksley	415 585-4620
11	M.	Greene	615 297-2723

/* Query 3 (first initial, last name, formatted phone number) */

```
SELECT UPPER(LEFT(au_fname, 1)) + '!',
       au_lname,
       '(' + LEFT(phone, 3) + ')' + RIGHT(phone, 8)
  FROM authors
```

	(Sin nombre de columna)	au_lname	(Sin nombre de columna)
1	J.	White	(408) 496-7223
2	M.	Green	(415) 986-7020
3	C.	Carson	(415) 548-7723
4	M.	O'Leary	(408) 286-2428
5	D.	Straight	(415) 834-2919
6	M.	Smith	(913) 843-0462
7	A.	Bennet	(415) 658-9932
8	A.	Dull	(415) 836-7128
9	B.	Gringlesby	(707) 938-6445
10	C.	Locksley	(415) 585-4620
11	M.	Greene	(615) 297-2723

/* Query 4 (finished output with formatting and titles) */

```
SELECT UPPER(LEFT(au_fname, 1)) + ' ' + au_lname AS "Name",
       '(' + LEFT(phone, 3) + ')' + RIGHT(phone, 8) AS "Phone"
  FROM authors
```

	(..) au_lname	(Sin nombre ...)
1	J. White	(408) 496-7223
2	M. Green	(415) 986-7020
3	C. Carson	(415) 548-7723
4	M. O'Leary	(408) 286-2428
5	D. Straight	(415) 834-2919
6	M. Smith	(913) 843-0462
7	A. Bennet	(415) 658-9932
8	A. Dull	(415) 836-7128
9	B. Gringlesby	(707) 938-6445
10	C. Locksley	(415) 585-4620
11	M. Greene	(615) 297-2723

Ejemplo 17. Cláusula TOP y WITH TIES

Filtrar los resultados por criterios de agrupación: los 5 más, el más En caso de empate con el operador WITH TIES mostrará todos las coincidencias.

USE pubs

```
SELECT TOP 5 royaltyper,au_id,title_id
   FROM [dbo].titleauthor
  ORDER BY au_id
```

```
SELECT TOP 5 WITH TIES royaltyper,au_id,title_id
   FROM [dbo].titleauthor
  ORDER BY au_id
```

Ejemplo 18. Con la cláusula WHERE



```
/* Query 1 (only 415 area code) */
SELECT UPPER(SUBSTRING(au_fname, 1, 1)) + '.' + au_lname AS
"Name",
  ('+' + LEFT(phone, 3) + ')' + RIGHT(phone,8) AS "Phone"
FROM authors
WHERE LEFT(Phone, 3) = '415'
```

	Name	Phone
1	M. Green	(415) 986-7020
2	C. Carson	(415) 548-7723
3	D. Straight	(415) 834-2919
4	A. Bennet	(415) 656-9932
5	A. Dull	(415) 836-7128
6	C. Locksley	(415) 585-4620
7	A. Yokomoto	(415) 935-4228
8	D. Stringer	(415) 843-2991
9	S. MacFeather	(415) 354-7128
10	L. Karsen	(415) 534-9219
11	S. Hunter	(415) 836-7128

```
/* Query 2 (California authors) */
SELECT UPPER(SUBSTRING(au_fname, 1, 1)) + '.' + au_lname AS
"Name",
  ('+' + LEFT(phone, 3) + ')' + RIGHT(phone,8) AS "Phone"
FROM authors
WHERE State = 'CA'
```

	Name	Phone
1	J. White	(408) 496-7223
2	M. Green	(415) 986-7020
3	C. Carson	(415) 548-7723
4	M. O'Leary	(408) 286-2428
5	D. Straight	(415) 834-2919
6	A. Bennet	(415) 656-9932
7	A. Dull	(415) 836-7128
8	B. Gringlesby	(707) 938-6445
9	C. Locksley	(415) 585-4620
10	A. Yokomoto	(415) 935-4228
11	D. Stringer	(415) 843-2991

Ejemplo 19. Con la cláusula BETWEEN

En la cláusula WHERE podremos utilizar el operador BETWEEN por ejemplo para buscar entre dos fechas:

```
SELECT *
  FROM [pubs].[dbo].[employee]
    WHERE hire_date BETWEEN '1994-01-01' and '2008-01-01'
```

Ejemplo 20. Con la cláusula IN

```
SELECT *
  FROM [pubs].[dbo].[authors]
    WHERE state IN ('IN','MD')
```

Ejemplo 21. Con la cláusula LIKE

Expresión LIKE o NOT LIKE

- % Caracter comodín para sustituir 0 o más caracteres
- _ Caracter comodín para sustituir 1 carácter individual
- [a-c] Caracter comodín para sustituir 1 carácter por un rango de letras, desde la a hasta c en este caso
- [^abc] Caracter comodín para sustituir 1 carácter por una letra que no sea la a la b o la c.

```
SELECT a1.au_fname NombreA1
  FROM authors a1
    where UPPER(a1.au_fname) LIKE 'A[^L]%'
```

	NombreA1
1	Abraham
2	Ann
3	Anne
4	Akiko

```
SELECT a1.au_fname NombreA1
  FROM authors a1
    where UPPER(a1.au_fname) LIKE 'A%'
```

	NombreA1
1	Abraham
2	Ann
3	Albert
4	Anne
5	Akiko

Ejemplo 22. Con la cláusula ORDER BY. Listado de teléfonos.



```
/* Phone List in Alpha Order (CA) */
SELECT UPPER(SUBSTRING(au_fname, 1, 1)) + ' ' + au_lname AS "Name",
      '(' + LEFT(phone, 3) + ')' + RIGHT(phone,8) AS "Phone"
FROM authors
WHERE State = 'CA'
ORDER BY au_lname
```

Name	Phone
1 A. Bennet	(415) 658-9932
2 C. Carson	(415) 548-7723
3 A. Dull	(415) 836-7128
4 M. Green	(415) 986-7020
5 B. Gringiesby	(707) 938-6445
6 S. Hunter	(415) 836-7128
7 L. Karsen	(415) 534-9219
8 C. Locksley	(415) 585-4620
9 S. MacFeather	(415) 354-7128
10 H. McBadden	(707) 448-4982
11 M. O'Leary	(408) 286-2428
12 N. Serra	(415) 834-7919

Ejemplo 23. Con la cláusula GROUP BY. Listado de teléfonos por estados.

```
/* Phone List grouped by state */
SELECT UPPER(SUBSTRING(au_fname, 1, 1)) + ' ' + au_lname AS "Name",
      '(' + LEFT(phone, 3) + ')' + RIGHT(phone,8) AS "Phone"
FROM authors
GROUP BY state, au_lname, au_fname, phone
```

Name	Phone
1 A. Bennet	(415) 658-9932
2 C. Carson	(415) 548-7723
3 A. Dull	(415) 836-7128
4 M. Green	(415) 986-7020
5 B. Gringiesby	(707) 938-6445
6 S. Hunter	(415) 836-7128
7 L. Karsen	(415) 534-9219
8 C. Locksley	(415) 585-4620
9 S. MacFeather	(415) 354-7128
10 H. McBadden	(707) 448-4982
11 M. O'Leary	(408) 286-2428

Ejemplo 24. Informe de ventas

```
/* Query 1 Listing of sales with no grouping */
SELECT title_id,
      CONVERT(VARCHAR, ord_date, 100) as ord_date,
      qty
FROM sales
ORDER BY title_id
```

	title_id	ord_date	qty
1	BU1032	Sep 14 1994 12:00AM	5
2	BU1032	Sep 14 1994 12:00AM	10
3	BU1111	Mar 11 1993 12:00AM	25
4	BU2075	Feb 21 1993 12:00AM	35
5	BU7832	Oct 28 1993 12:00AM	15
6	MC2222	Dic 12 1993 12:00AM	10
7	MC3021	Sep 14 1994 12:00AM	25
8	MC3021	Sep 14 1994 12:00AM	15
9	PC1035	May 22 1993 12:00AM	30
10	PC8888	May 24 1993 12:00AM	50
11	PS1372	May 29 1993 12:00AM	20

```
/* Query 2 Volume sales > 20 */
SELECT title_id AS "Title ID",
      CONVERT(VARCHAR, MAX(ord_date), 100) AS "Last Sale Date",
      SUM(qty) AS "Total Sales"
FROM sales
GROUP BY title_id
HAVING SUM(qty) > 20
ORDER BY SUM(qty) DESC
```

Title ID	Last Sale Date	Total
PS2091	Sep 14 1994 12:00AM	108
PC8888	May 24 1993 12:00AM	50
MC3021	Sep 14 1994 12:00AM	40
TC3218	Jun 15 1992 12:00AM	40
BU2075	Feb 21 1993 12:00AM	35
PC1035	May 22 1993 12:00AM	30
BU1111	Mar 11 1993 12:00AM	25
PS2106	May 29 1993 12:00AM	25
PS7777	May 29 1993 12:00AM	25

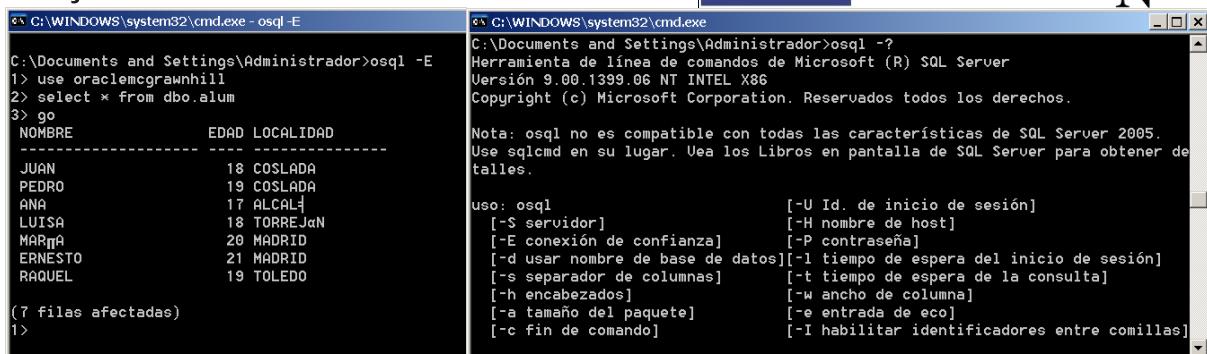
La misma query anterior pero con mayor rendimiento:

```
SELECT title_id AS "Title ID",
      CONVERT(VARCHAR, MAX(ord_date), 100) AS "Last Sale Date",
      SUM(qty) AS "Total Sales"
FROM sales
GROUP BY title_id
HAVING SUM(qty) > 20
ORDER BY 3 DESC
```

Nota.- osql

[Perez, 2006]

La herramienta osql, lo mismo que las Consultas de motor de base de datos de SQL Server 2008, permite la utilización de Transact-SQL de forma interactiva para acceder y cambiar datos mediante consultas.



```
C:\Documents and Settings\Administrador>osql -E
1> use oraclemcgrawhill
2> select * from dbo.alum
3> go
NOMBRE      EDAD LOCALIDAD
-----      --
JUAN        18 COSLADA
PEDRO       19 COSLADA
ANA         17 ALCAL|N
LUISA       18 TORREJdN
MARIA       20 MADRID
ERNESTO     21 MADRID
RAQUEL      19 TOLEDO
(7 filas afectadas)
1>
```

```
C:\Documents and Settings\Administrador>osql -?
Herramienta de línea de comandos de Microsoft (R) SQL Server
Versión 9.00.1399.06 NT INTEL X86
Copyright (c) Microsoft Corporation. Reservados todos los derechos.

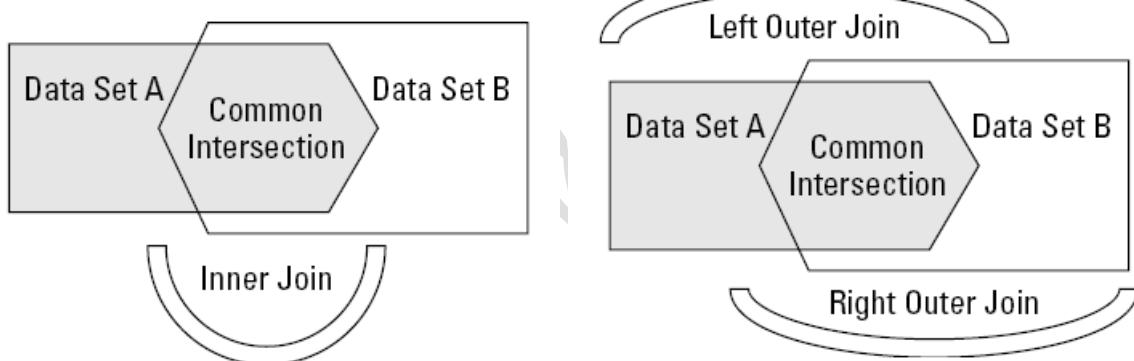
Nota: osql no es compatible con todas las características de SQL Server 2005.
Use sqlcmd en su lugar. Vea los Libros en pantalla de SQL Server para obtener de
talles.

uso: osql           [-U Id. de inicio de sesión]
      [-S servidor]   [-H nombre de host]
      [-E conexión de confianza] [-P contraseña]
      [-d usar nombre de base de datos][-l tiempo de espera del inicio de sesión]
      [-s separador de columnas] [-t tiempo de espera de la consulta]
      [-h encabezados]  [-w ancho de columna]
      [-a tamaño del paquete] [-r entrada de eco]
      [-c fin de comando] [-I habilitar identificadores entre comillas]
```

Imagen 45. Conexión a osql con –E para utilizar las credenciales por defecto del usuario

5.3.1.1 Cláusula FROM

```
[ INNER | { { LEFT | RIGHT | FULL } [ OUTER ] } ] JOIN table_name | view_name ON
join_condition )
```



Cuando cruzamos dos tablas con algún campo relacionado podemos unirlas por dicho campo a través de la cláusula JOIN. Ejemplos:

```
/*q1=q2=q3*/
SELECT t.title_id, a.au_fname from titleauthor t, authors a
WHERE t.au_id=a.au_id
order by t.title_id

/*q2=q1=q3*/
SELECT t.title_id, a.au_fname from authors a
JOIN titleauthor t on T.AU_ID=A.AU_ID
order by title_id

/*q3=q1=q2*/ -- INNER JOIN ES EQUIVALENTE A PONER JOIN
SELECT t.title_id, a.au_fname from authors a
INNER JOIN titleauthor t on T.AU_ID=A.AU_ID
order by title_id
```



Junta de
Castilla y León





	title_id	au_fname
1	BU1032	Marjorie
2	BU1032	Abraham
3	BU1111	Michael
4	BU1111	Stearns
5	BU2075	Marjorie
6	BU7832	Dean
7	MC2222	Innes
8	MC3021	Michel
9	MC3021	Anne
10	PC1035	Cheryl
11	PC8888	Ann
12	PC8888	Sherri

Cuadriculas Mensajes

Completado el proceso por lotes de la consulta.

Luz (8.0) LUZ\Juvi (51) pubs 0:00:00 25 filas Conexiones: 1 MAYÚS

En la siguiente consulta utilizamos el left join. Ésta cláusula permitirá mostrar todos los datos asociados a la “tabla de la izquierda” aunque no tenga coincidencias en la segunda tabla “tabla de la derecha”. El RIGHT JOIN es opuesto.

```
/*q4=q5 puesto que se cambian el orden de las tablas cruzadas*/
SELECT t.title_id, a.au_fname from authors a
    LEFT JOIN titleauthor t on T.AU_ID=A.AU_ID
        order by title_id
/*q5=q4*/
SELECT t.title_id, a.au_fname from titleauthor t
    RIGHT JOIN authors a on T.AU_ID=A.AU_ID
        order by title_id
```

	title_id	au_fname
1	NULL	Morningstar
2	NULL	Heather
3	NULL	Meander
4	NULL	Dirk
5	BU1032	Abraham
6	BU1032	Marjorie
7	BU1111	Stearns
8	BU1111	Michael
9	BU2075	Marjorie
10	BU7832	Dean

Véanse más ejemplos del join en Ejemplo 26, pág. 93.

La sintaxis del inner join se utiliza para hacer más fácil de comprender estructuras SQL cuando éstas se complican con múltiples tablas cruzadas, pero puede ser sustituido perfectamente por condiciones que igualen en el WHERE campos comunes de todas las tablas del FROM.

```
SELECT authors.au_lname, authors.au_fname, titleauthor.au_id, titles.title
    FROM dbo.authors INNER JOIN dbo.titleauthor
        ON titleauthor.au_id = authors.au_id
    INNER JOIN dbo.titles
        ON titles.title_id = titleauthor.title_id
    WHERE authors.au_lname = 'Green'
```

Podremos hacer un SELECT de una tabla que está en otra base de datos e incluso en otro servidor. Ejemplo: Supongamos que estamos conectados a pubs, el siguiente

SELECT será correcto:

```
SELECT * FROM master.dbo.sysaltfiles
```

Si lo que quisiéramos es seleccionar datos de una tabla de una base de datos de otro servidor, utilizaremos el comando, siempre y cuando tengan el mismo modo de autenticación o permisos de acceso de un determinado usuario:

```
sp_addlinkedserver @server = '80.81.114.158' -- El servidor se llama en el consola de Management Studio 80.81.114.158
SELECT * FROM "80.81.114.158".master.dbo.sysfiles
```

El anterior comando agrega registra el servidor en la tabla sysservers de la BD master:

```
SELECT * FROM master.dbo.sysservers
```

De todas las consultas ejecutadas por el cliente contra el servidor se pueden tener una serie de estadísticas como puede verse en la Imagen 46. Dichas estadísticas se pueden inicializar en el menú de Consulta -> Restablecer estadísticas del cliente

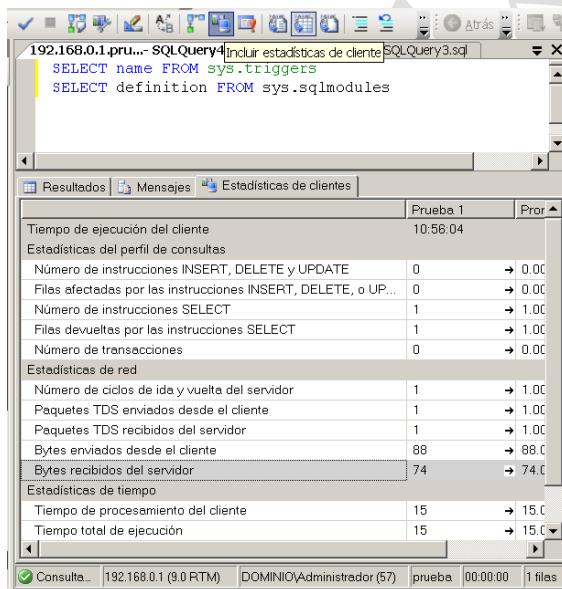


Imagen 46. Estadísticas del cliente

En cuanto a rendimiento podemos afirmar que carece de importancia que método usar para cruzar tablas, si el INNER JOIN o filtrado por campos comunes:

```
select * from titles inner join titleauthor
on titles.title_id=titleauthor.title_id

select * from titles, titleauthor
where titles.title_id=titleauthor.title_id
```

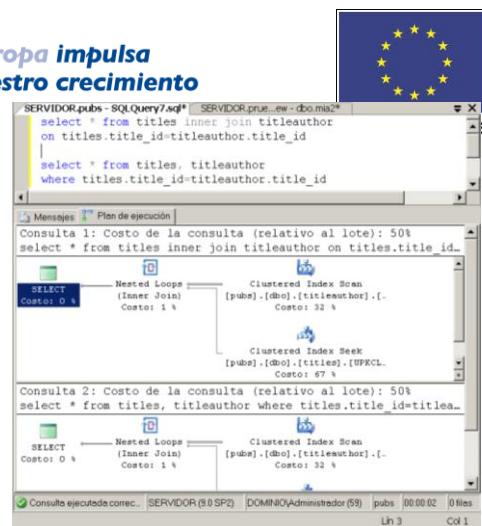


Imagen 47. Plan de ejecución de los dos tipos de consultas

5.3.1.2 GROUP BY - HAVING

Véase la sintaxis del Select de la pág. 82.

De un conjunto resultado de una query se agrupará por uno o varios campos indicados a continuación del GROUP BY.

Ejemplo:

Por ejemplo, **TableX** contiene:

ColumnA	ColumnB	ColumnC
1	abc	5
1	def	4
1	ghi	9
2	jkl	8
2	mno	3

La columnaA es una columna candidata a ser de agrupamiento ya que tiene datos repetidos por los que podríamos agrupar.

Permitido:

```
SELECT ColumnA,
       MAX(ColumnB) AS MaxB,
       SUM(ColumnC) AS SumC
  FROM TableX
 GROUP BY ColumnA
```

Resultado:

ColumnA	MaxB	SumC
1	ghi	18
2	mno	11

No permitido:

```
SELECT ColumnA,
       ColumnB,
       SUM(ColumnC) AS SumC
  FROM TableX
 GROUP BY ColumnA
```

Elección de filas con la cláusula HAVING.

HAVING es a GROUP BY lo que WHERE a SELECT, es decir filtra un conjunto de registros con alguna condición determinada. La diferencia está en que la búsqueda por el WHERE se realiza antes de realizar el agrupamiento y el HAVING se realiza después de agrupar el conjunto de resultados. Las únicas condiciones de búsqueda que se deben especificar en la cláusula HAVING son aquellas que se deben aplicar una



hayan realizado las operaciones de agrupamiento.

```
SELECT ColumnA,
       MAX(ColumnB) AS MaxB
       SUM(ColumnC) AS SumC
```

```
FROM TableX
GROUP BY ColumnA
HAVING SUM(SumC) > 11
```

Resultado:

ColumnA	MaxB	SumC
1	ghi	18



Having también se puede utilizar sin función de agrupación:

```
SELECT type
FROM titles
GROUP BY type
HAVING type LIKE 'p%'
```



5.3.1.3 COMPUTE

Véase la sintaxis del Select de la pág. 82.

Sintaxis:

Genera datos de resumen.

```
[ COMPUTE
  { { AVG | COUNT | MAX | MIN | STDEV | STDEVP
    | VAR | VARP | SUM }
    ( expression ) } [ ,...n ]
  [ BY expression [ ,...n ] ]
]
```

Ejemplo 25.

```
use pubs
SELECT * from titleauthor
compute count(au_id), avg(royaltyper)
```

Resultados					
	au_id	title_id	au_ord	royaltyper	
1	172-32-1176 PS3333	BU1112	1	20	
2	213-46-8915 BU1032	2	40		
3	213-46-8915 BU2075	1	100		
4	238-95-7766 PC1035	1	100		
5	267-41-2394 BU1111	2	40		
	cnt	avg			
1	25	64			

Consulta ejecutada correctamente. SERVIDOR (9.0 RTM) | DOMINIO\Administrador (S2) | pubs | 00:00:00 | 26 filas

Lín 10 Col 1 Car 1 INS

Ejemplo 26. Condiciones JOIN. Combinación de ventas por autor.

/* Query 1 titleauthor table */

```
SELECT *
FROM titleauthor
```

	au_id	title_id	au_ord	royaltyper
1	172-32-1176 PS3333	1	100	
2	213-46-8915 BU1032	2	40	
3	213-46-8915 BU2075	1	100	
4	238-95-7766 PC1035	1	100	
5	267-41-2394 BU1111	2	40	
6	267-41-2394 TC7777	2	30	
7	274-80-9391 BU7832	1	100	
8	409-56-7008 BU1032	1	60	
9	427-17-2319 PC8888	1	50	
10	472-27-2349 TC7777	3	30	
11	486-29-1786 PC9999	1	100	
12	486-29-1786 PS7777	1	100	

```
/* Query 2 Volume sales > 20 by author */
SELECT UPPER(LEFT(authors.au_fname, 1)) + '.' +
authors.au_lname AS "Name",
SUM(sales.qty) AS "Total Sales"
FROM sales
INNER JOIN titleauthor ON sales.title_id = titleauthor.title_id
INNER JOIN authors ON titleauthor.au_id = authors.au_id
GROUP BY authors.au_fname, authors.au_lname
HAVING SUM(qty) > 20
ORDER BY SUM(qty) DESC
```

Name	Total Sales
A. Ringer	148
A. Ringer	133
A. Dull	50
M. Green	50
S. Hunter	50
S. MacFeather	45
M. O'Leary	45
M. DeFrance	40
S. Panteley	40
C. Carson	30
C. Locksley	25

5.3.2 Funciones de agrupación: SUM, MAX, MIN, AVG, COUNT.

La mejor manera de conocer las funciones existentes en el sistema es buscarlas en el explorador de objetos y una vez localizadas utilizar la ayuda de SQL Server 2008 para usarla. A continuación se verán las más importantes.

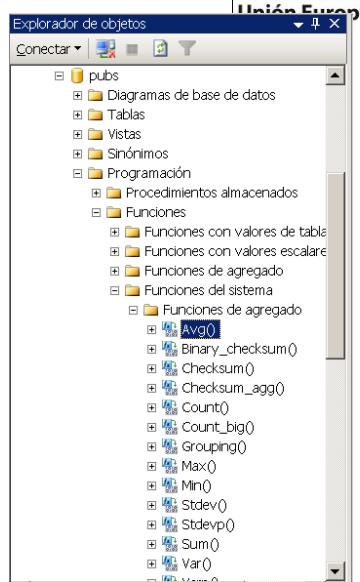


Imagen 48. Funciones localizadas en el explorador de objetos

- **SUM**

La función SUM() devuelve el total de los valores de una columna. Dependiendo de como utilice esta función, funcionara en una tabla entera o en un grupo de registros. Es indispensable que las columnas sean numéricas para que esta función funcione perfectamente. Pueden utilizar la función CONVERT() para cambiar los números almacenados como texto al tipo de datos numéricos requeridos, el cual también se debe especificar en la función. (*Véase la función convert de la página 96*).

Ejemplo 27

```
/* Sumo los porcentajes de beneficio de los autores por cada libro escrito*/
select title_id, sum(royaltyper) MaxBenefi, count(au_id) NumAutores
    from titleauthor
   group by title_id
  order by MaxBenefi ASC
/*order by MaxBenefi DESC*/
```

	title_id	MaxBenefi	NumAutores
1	BU1032	100	2
2	BU1111	100	2
3	BU2075	100	1
4	BU7832	100	1
5	MC2222	100	1
6	MC3021	100	2
7	PC1035	100	1
8	PC8888	100	2
9	PC9999	100	1
10	PS1372	100	2
11	PS2091	100	2
12	PS2106	100	1
13	PS3333	100	1

- **MAX**

La función MAX() devuelve el valor máximo de una columna en una tabla o conjunto de filas devuelto en una consulta. Esta función sirve tanto para texto como para fechas y números.

Ejemplo 28

			
title_id	MaxBenefi	Autor	
1 PS1372	25	724-80-9391	
2 MC3021	25	899-46-2035	
3 TC7777	30	267-41-2394	
4 TC7777	30	472-27-2349	
5 TC7777	40	672-71-3249	
6 BU1111	40	267-41-2394	
7 BU1032	40	213-46-8915	
8 PC8888	50	427-17-2319	
9 PC8888	50	846-92-7186	
10 PS2091	50	899-46-2035	
11 PS2091	50	998-72-3567	

```
select title_id, max(royaltyper) MaxBenefi, au_id Autor
from titleauthor
group by title_id, au_id
order by MaxBenefi ASC
/*order by MaxBenefi DESC*/
```

▪ MIN

MIN() es el opuesto a MAX() en todos los aspectos. Se aplican a la función MIN() las mismas reglas y características que la función MAX(). Recuerde que el valor más bajo devuelto para texto depende del orden de selección y si es sensible al tamaño de carácter.

Ejemplo 29

```
SELECT title_id, MIN(royaltyper) MinBenefi, au_id Autor
FROM titleauthor
GROUP BY title_id, au_id
/*ORDER BY MinBenefi ASC*/
ORDER BY MinBenefi des
```

title_id	MinBenefi	Autor
1 PS3333	100	172-32-1176
2 BU2075	100	213-46-8915
3 PC1035	100	238-95-7766
4 BU7802	100	274-80-9391
5 PC9999	100	486-29-1786
6 PS7777	100	486-29-1786
7 TC4203	100	648-92-1872
8 MC2222	100	712-45-1867
9 TC3218	100	807-91-6654
10 PS2106	100	998-72-3567
11 PS1372	75	756-30-7391
12 MC3021	75	722-51-5454
13 BU1032	60	409-56-7008
14 BU1111	60	724-80-9391
15 PC8888	50	846-92-7186

▪ AVG

Muchos informes dependen de valores medios para seguir las tendencias a lo largo del tiempo. La función AVG() calcula el valor medio de una columna en un tabla o en un conjunto de filas, según como se use. Solo se pueden pasar valores numéricos en la función AVG(). Puede utilizar la función CONVERT() anidada dentro de la función AVG() para que devuelva el valor numérico de un número almacenado como texto para aproximarse a la restricción numérica de esta función.

Ejemplo 30

```
SELECT title_id, AVG(royaltyper) MediaBen
FROM titleauthor
GROUP BY title_id
ORDER BY MediaBen DESC
```

title_id	MediaBen
1 BU2075	100
2 BU7802	100
3 MC2222	100
4 PC1035	100
5 PC9999	100
6 PS2106	100
7 PS3333	100
8 PS7777	100
9 TC3218	100
10 TC4203	100
11 PS2091	50
12 PS1372	50
13 PC8888	50
14 MC3021	50
15 BU1032	50
16 BU1111	50

▪ COUNT

COUNT() es un función muy común que devuelve el numero de filas de una consulta que coincide con una cláusula WHERE en concreto. El número de filas en una tabla puede devolverse a una variable con fines de prueba de una forma muy efectiva. COUNT() devuelve el número de valores no nulos indicados por la cláusula WHERE en una consulta. Se puede pasar el asterisco a la función COUNT(*) para conseguir un contador de filas de una consulta. COUNT() es útil en desencadenadores y procedimientos almacenados para determinar no solo la existencia de

filas, sino también el número exacto de filas con el que está trabajando. El valor que devuelve esta función es numérico.

Ejemplo 31

```
SELECT title_id, COUNT(au_id) NAutores
      FROM titleauthor
     GROUP BY title_id
    ORDER BY NAutores
```

	title_id	NAutores
1	BU2075	1
2	BU7832	1
3	MC2222	1
4	PC1035	1
5	PC9999	1
6	PS2106	1
7	PS3333	1
8	PS7777	1
9	TC3218	1
10	TC4203	1
11	PS2091	2
12	PS1372	2
13	PC8888	2
14	MC3021	2
15	BU1032	2
16	BU1111	2

5.3.3 Funciones: CONVERT, GETDATE, DATEDIFF, DATEPART, SOUNDEX, SUBSTRING, LEFT Y RIGHT, UPPER, CHARINDEX, RTRIM Y LTRIM, LEN, REPLICATE, SPACE, REPLACE STR, CHAR, ASCII

▪ CONVERT

La función CONVERT() se utiliza a menudo en código SQL para asegurar que tiene los tipos de datos correctos para realizar una operación. Esta versátil función convierte un tipo de datos en otro de forma que se puedan realizar ciertas operaciones. El primer parámetro de la función es el tipo de datos al que quiere que se convierta la columna existente o la expresión. (Véase *tipos de datos 5.2.1.2 Tipos de datos*)

Microsoft SQL Server intentará convertir los tipos de datos de forma automática siempre que no pueda procesar una función o consulta con los datos introducidos. El servidor devuelve un mensaje de error que sugiere la utilización de la función CONVERT() para solucionar el problema. Función similar es CAST():

CAST (*expression AS data_type*) – transformamos la expresión al tipo de datos que especifiquemos.

Ejemplo 32. Utilización de la función CONVERT() y otras.

```
/* New Author ID query */
SELECT au_id AS "Old Author ID",
       UPPER(LEFT(au_fname, 1)) +
       UPPER(LEFT(au_lname, 1)) +
       '-' + LEFT(au_id, 3) +
       '-' + UPPER(state) +
       CONVERT(CHAR(1), contract) AS "New Author ID",
       UPPER(LEFT(au_fname, 1)) + '.' +
       LEFT(au_lname, 15) AS "Name"
  FROM authors
 ORDER BY au_lname
```

	Old Author ID	New Author ID	Name
1	409-56-7008	AB-409-CA1	A. Bennet
2	648-92-1872	RB-648-OR1	R. Blotchet-Halls
3	238-95-7766	CC-238-CA1	C. Carson
4	722-51-5454	ND-722-IN1	M. DeFrance
5	712-45-1867	ID-712-M11	I. del Castillo
6	427-17-2319	AD-427-CA1	A. Dull
7	213-46-8915	MG-213-CA1	M. Green
8	527-72-3246	MG-527-TNO	M. Greene
9	472-27-2349	BG-472-CA1	B. Gringlesby
10	846-92-7186	SH-846-CA1	S. Hunter
11	756-30-7391	LK-756-CA1	L. Karsen
12	486-29-1786	CL-486-CA1	C. Locksley

▪ GETDATE

La función GETDATE() devuelve la fecha y hora actual del sistema horario del servidor.

Esta función puede utilizarse de forma automática para insertar valores de fecha en columnas o encontrar la fecha actual en comparaciones en cláusulas WHERE.

Ejemplo 33.

```
SELECT getdate() a, ord_date b,
(convert(decimal,getdate()-ord_date)/365) amenosb
FROM sales
```

	a	b	amenosb
1	2005-03-10 13:14:43.850	1994-09-14 00:00:00.000	10.495890
2	2005-03-10 13:14:43.850	1994-09-13 00:00:00.000	10.498630
3	2005-03-10 13:14:43.850	1993-05-24 00:00:00.000	11.805479
4	2005-03-10 13:14:43.850	1994-09-13 00:00:00.000	10.498630
5	2005-03-10 13:14:43.850	1994-09-14 00:00:00.000	10.495890
6	2005-03-10 13:14:43.850	1992-06-15 00:00:00.000	12.745205
7	2005-03-10 13:14:43.850	1992-06-15 00:00:00.000	12.745205
8	2005-03-10 13:14:43.850	1992-06-15 00:00:00.000	12.745205
9	2005-03-10 13:14:43.850	1994-09-14 00:00:00.000	10.495890
10	2005-03-10 13:14:43.850	1994-09-14 00:00:00.000	10.495890
11	2005-03-10 13:14:43.850	1992-06-19 00:00:00.000	11.791280

▪ DATEDIFF

Puede utilizar la función DATEDIFF() para comparar y obtener la diferencia entre elementos de fecha, como días, semanas, minutos y horas. Si se utiliza en una cláusula WHERE, puede obtener registros que coincidan con un rango de fechas o que coincidan con ciertos intervalos de tiempo. Asegúrese de que especifica los argumentos para esta función en el orden correcto o el valor devuelto podrá ser el opuesto al que esperaba.

Ejemplo 34.

```
SELECT DATEDIFF(day, ord_date, getdate()) dias,
       DATEDIFF(month, ord_date, getdate()) meses ,
       DATEDIFF(year, ord_date, getdate()) años,
       ord_date FechaFactura, getdate() Hoy
FROM sales
```

	dias	meses	años	FechaFactura	Hoy
1	3830 126	11	1994-09-14...	2005-03-10 ...	
2	3831 126	11	1994-09-13...	2005-03-10 ...	
3	4308 142	12	1993-05-24...	2005-03-10 ...	
4	3831 126	11	1994-09-13...	2005-03-10 ...	
5	3830 126	11	1994-09-14...	2005-03-10 ...	
6	4651 153	13	1992-06-15...	2005-03-10 ...	
7	4651 153	13	1992-06-15...	2005-03-10 ...	
8	4651 153	13	1992-06-15...	2005-03-10 ...	
9	3830 126	11	1994-09-14...	2005-03-10 ...	

▪ DATEPART

La función DATEPART() devuelve un valor igual a la parte de la fecha que haya especificado. Por ejemplo, si necesita saber el día de la semana de una fecha particular, puede utilizar esta función para extraer el dato de una columna o una variable.

Ejemplo 35. Utilización de la función CONVERT() y otras.

```
SELECT DATEPART(month, GETDATE())
SELECT DATEPART(year, GETDATE())
SELECT DATEPART(day, GETDATE())
```

año
2005

▪ SOUNDEX

La función SOUNDEX() convierte una cadena en un código de cuatro dígitos que representa la cadena. Si dos cadenas de caracteres son muy similares, devolverán el mismo valor SOUNDEX(). Esta función puede utilizarse para encontrar una lista de coincidencias potenciales para una cadena en un grupo de filas.

Puede pensar en esta función como «sonidos similares». Si las cadenas de caracteres tienen una pronunciación similar, coincidirá el número devuelto por la función. Puede ser útil para encontrar datos cuando el usuario no sabe exactamente como se escribe. SOUNDEX() es

crítico para crear una buena opción de búsqueda de registros en la base de datos.

Ejemplo 36.

```
SELECT a1.au_fname NombreA1 , soundex(a1.au_fname) Sonido, a2.au_fname
NombreA2
FROM authors a1, authors a2
where soundex(a1.au_fname)=soundex(a2.au_fname)
ORDER BY Sonido
```

	NombreA1	Sonido	NombreA2
4	Ann	A500	Anne
5	Anne	A500	Anne
6	Ann	A500	Ann
7	Anne	A500	Ann
8	Burt	B630	Burt
9	Cheryl	C640	Cheryl
10	Charlene	C645	Charlene
11	Dean	D500	Dean
12	Dirk	D620	Dirk
13	Heather	H360	Heather
14	Innes	I520	Innes
15	Johnson	J525	Johnson
16	Livia	L100	Livia
17	Michael	M240	Michael
18	Michel	M240	Michael
19	Michel	M240	Michael
20	Michael	M240	Michael

▪ SUBSTRING

La función SUBSTRING() se utiliza en muchas ocasiones. Cualquier manipulación de caracteres puede llevarse a cabo con esta función junto con unos cuantos operadores de caracteres y algunas otras funciones básicas.

Otra función útil es la función STUFF(); que elimina el número de caracteres especificados e inserta otro conjunto de caracteres en un punto de inicio indicado. Combinada con SUBSTRING(), es muy útil para construir cadenas de caracteres al momento en su código SQL.

Utilización de: STUFF (*character_expression, start, numeroveces , character_expression*)
 Utilización de: SUBSTRING (*expression, start , length*)

Ejemplo 37. STUFF y SUBSTRING

```
SELECT au_id IdAutor,
STUFF(STUFF(au_id,4,1,'/'),7,1,'\'') NuevoldAutor
FROM authors
```

	IdAutor	NuevoldAutor
1	409-56-7008	409\56\7008
2	648-92-1872	648\92\1872
3	238-95-7766	238\95\7766
4	722-51-5454	722\51\5454

```
SELECT au_fname Nombre,
au_lname Apellido,
SUBSTRING(au_fname, 1, 3) +
SUBSTRING(RIGHT(au_lname,5),2,3) Password
-- Password construido con los 3 primeros
-- caracteres del nombre más, de los 5 últimos
-- caracteres del apellido el 2º 3º y 4º.
FROM authors
ORDER BY au_lname
```

Nombre	Apellido	Password
Abraham	Bennet	Abrnne
Reginald	Blotchet-Halls	Regall
Cheryl	Carson	Cherso
Michel	DeFrance	Micanc
Innes	del Castillo	Innill
Ann	Dull	Annnull

▪ LEFT y RIGHT

Las funciones LEFT() y RIGHT() se comportan de la misma forma que la función SUBSTRING(), pero están especialmente diseñadas para trabajar con los extremos izquierdo y derecho de la cadena que estás manipulando. Estas funciones devuelven el número específico de caracteres del extremo apropiado de la cadena. La función SUBSTRING() se puede utilizar fácilmente en vez de la función LEFT, pero es más peligroso codificar la devolución de datos desde el extremo derecho de una cadena sin utilizar la función RIGHT.

`LEFT (character_expression , integer_expression)`
`RIGHT(character_expression , integer_expression)`

▪ **UPPER**

`UPPER()` convierte la cadena que se pasa por la función a letras mayúsculas. Puede utilizar esta función para mantener la integridad de las columnas de texto en sus tablas sin que tenga que intervenir el usuario o el cliente. Algún software cliente da por hecho que esta función está sometida a control y prácticamente solo se utiliza para asegurarse de que se pasan los caracteres en mayúscula cuando es necesario.

Ejemplo 38.

```
SELECT a1.au_fname NombreA1
      FROM authors a1
     where UPPER(a1.au_fname) LIKE 'ANN%'
```

NombreA1	
1	Ann
2	Anne

▪ **CHARINDEX**

La función `CHARINDEX()` puede utilizarse para buscar coincidencias para una cadena de caracteres en una columna. Esta función se utiliza para devolver la posición inicial de una cadena dentro de otra cadena. Si la "cadena de búsqueda" no se localiza en la "cadena de destino", la función devolverá el valor cero. Si utiliza esta función para buscar en una tabla completa, devuelve un conjunto de resultados con un valor distinto de cero para cada columna que contenga una cadena de caracteres.

`CHARINDEX()` **no le permite comodines**. Si tiene que buscar una cadena de caracteres que contenga comodines utilice la función `PATINDEX()`.

`CHARINDEX (expression1 , expression2 [, start_location])`

Donde:

expression1

Es una expresión que contiene la secuencia de caracteres que se desea buscar. *expression1* es una expresión del tipo de cadenas cortas de caracteres.

expression2

Es una expresión, normalmente una columna, en la que se busca la cadena especificada. *expression2* es de la categoría del tipo de datos cadena de caracteres.

start_location

Es la posición del carácter de *expression2* en el que se empieza la búsqueda de *expression1*. Si no se especifica *start_location*, es un número negativo o es cero, la búsqueda empieza al principio de la cadena *expression2*.

Ejemplo 39.

```
SELECT CHARINDEX('o', notes) n, notes
      FROM titles
     WHERE title_id = 'TC3218'
```

n	notes
3	Profusely illustrated in color, thi

```
SELECT CHARINDEX('o', notes, 4) n, notes
      FROM titles
```

```
WHERE title_id = 'TC3218'
```

```
SELECT PATINDEX('%illus%', notes) n, notes
FROM titles
WHERE title_id = 'TC3218'
```

n	notes
27	Profusely illustrated in color, th

▪ RTRIM Y LTRIM

RTRIM() elimina todos los espacios blancos en los extremos de una cadena o columna. En algunas situaciones, ayuda a mantener el formato y el trabajo de información de la forma que esperan sus aplicaciones. RTRIM() es mas útil en informes de texto generados para puro SQL. LTRIM() funciona de la misma forma que RTRIM(), excepto que elimina espacios en blanco al principio de una cadena y no al final.

Ejemplo 40. RTRIM, LTRIM y LEN.

```
SELECT CITY,
RTRIM(SUBSTRING(CITY,1,CHARINDEX(' ',CITY))) +
LTRIM(SUBSTRING(CITY,CHARINDEX(' ',CITY),LEN(CITY))) City-
SinEspacios
FROM authors
```

CITY	CitySinEspacios
Menlo Park	MenloPark
Oakland	Oaklan
Berkeley	Berkele
San Jose	SanJose
Oakland	Oaklan
Lawrence	Lawrenc
Berkeley	Berkele
Palo Alto	PaloAlto
Covel	Covel
San Francisco	SanFrancisco

▪ LEN

La función LEN() se utiliza para devolver el número de caracteres (no bytes) de una cadena dada. Le permite determinar correctamente la longitud de la cadena de caracteres.

▪ REPLICATE

La función REPLICATE() devuelve una cadena de caracteres repetida un número específico de veces. Puede ser útil para formatear el resultado de una consulta. En vez de escribir 50 guiones en un comando PRINT para separar información en el resultado, esta función le permite repetir el guion 50 veces sin escribir tanto.

```
REPLICATE ( character_expression , integer_expression )
```

Ejemplo 41.

```
SELECT UPPER(au_lname) + ',' + SPACE(1) + au_fname +
REPLICATE('.!l.maximo-(LEN(au_lname)+1+len(au_fname))+' +
phone AS DatosPersonales, l.maximo, (LEN(au_lname)+1+len(au_fname)) "Longitud Nombre"
FROM authors,
(SELECT MAX(longitud) maximo -- Obtenemos la longitud máxima de la concatenación de
nombre y apellido
from
(SELECT LEN(au_lname + ',' + SPACE(1) + au_fname) longitud
FROM authors) longitudes )l
order by DatosPersonales DESC
```



	DatosPersonales	maximo	Longitud Nombre
19	DEL CASTILLO, Innes.....615 996-8275	24	18
20	DEFRANCE, Michel.....219 547-9982	24	15
21	CARSON, Cheryl.....415 548-7723	24	13
22	BLOTCHET-HALLS, Reginald.503 745-6402	24	23
23	BENNET, Abraham.....415 658-9932	24	14

▪ SPACE

La función SPACE() devuelve un número específico de veces el número indicado de espacios. Otra función útil para el formato.

▪ REPLACE

La función REPLACE() utiliza tres parámetros de cadenas de caracteres. Todas las veces que aparezca la segunda cadena en la primera se sustituye por el valor de la tercera cadena. Básicamente, se comporta como la función buscar y reemplazar de un procesador de textos.

REPLACE ('string_expression1' , 'string_expression2' , 'string_expression3')

Ejemplo 42.

```
SELECT address,
       REPLACE(address,'#','n')
  FROM authors
```

address	(Sin nombre de columna)
10932 Bigge Rd.	10932 Bigge Rd.
309 63rd St. #411	309 63rd St. n°411
589 Darwin Ln.	589 Darwin Ln.
22 Cleveland Av. #14	22 Cleveland Av. n°14
5420 College Av.	5420 College Av.
10 Mississippi Dr.	10 Mississippi Dr.

▪ STR

La función STR() convierte un valor numérico en un carácter. Al contrario que la función CONVERT(), la función STR() le permite especificar la longitud total y el número de decimales que se deben incluir cuando se convierte a carácter.

Ejemplo 43.

```
SELECT ""+ STR(123.45, 10, 3)+ ""
// "" comilla simple + comilla doble + comilla simple
```

(Sin nombre de columna)
1 " 123.450"

▪ CHAR

La función CHAR() devuelve el carácter que corresponde al código ASCII pasado como un entero en la función. Puede ser útil para agregar a su resultado caracteres que de otra forma resultarían inútiles. CHAR(10) devuelve una alimentación de línea y CHAR(13) un retorno de carro. CHAR(34) son comillas dobles, que pueden ser útiles a la hora de crear una cadena de resultado que hubiese anidado de forma confusa comillas sin utilizar esta función.

▪ ASCII

ASCII() es la función recíproca a CHAR(). Devuelve el código ASCII como entero del carácter más a la izquierda de la cadena.

Ejemplo 44.

```
SELECT ASCII(CHAR(65)) "ASCII", CHAR(65) CHAR, ASCII('A') ASCII
```

	ASCII	CHAR	ASCII
1	65	À	65

5.3.4 Funciones de sistema: ISNULL, COALESCE, USER_ID, USER_NAME, DATALENGTH, COL_LENGTH, CONVERT

De una forma similar a las funciones vistas, las funciones del sistema devuelven valores basados en el servidor o realizan funciones que, realizadas de otra forma, precisarían una gran cantidad de código. Estas funciones se usan normalmente en las secuencias y procedimientos almacenados para aportar un mayor grado de funcionalidad.

- **ISNULL**

Puede sustituir dinámicamente los valores nulos NULL en los conjuntos de resultados con la función ISNULL(). Especifique el valor, como segundo parámetro a la función, que quiere que se devuelva en caso de que la columna correspondiente contenga un valor nulo y esta función lo comprobará al momento. Esta función puede ser muy útil en informes y consultas de ejecución diarias escritas en archivos de texto. NULLIF(p1,p2) devolverá nulo si el valor de p1 = p2.

ISNULL (check_expression , replacement_value)

Ejemplo 45. ISNULL

```
/* Existen nulos en el campo precio, luego el count no los cuenta*/
SELECT type, COUNT(title) NTitulos, COUNT(price) NPrecios, COUNT(ISNULL(PRICE,0)) NPreciosSinNulos
    from titles
    group by type

SELECT title, type, price
    FROM titles
    order by type DESC
```



	type	NTitulos	NPrecios	NPreciosSinNulos
1	business	4	4	4
2	mod_cook	2	2	2
3	popular_comp	3	2	3
4	psychology	5	5	5
5	trad_cook	3	3	3
6	UNDECIDED	1	0	1
	title		type	price
1	The Psychology of Computer Cooking		UNDECIDED	NULL
2	Onions, Leeks, and Garlic: Cooking ...		trad_cook	20.9500
3	Fifty Years in Buckingham Palace Ki...		trad_cook	11.9500
4	Sushi, Anyone?		trad_cook	14.9900
5	Emotional Security: A New Algorithm		psychology	7.9900
6	Life Without Fear		psychology	7.0000
7	Prolonged Data Deprivation: Four Ca...		psychology	19.9900
8	Computer Phobic AND Non-Phobic Indi...		psychology	21.5900
9	Is Anger the Enemy?		psychology	10.9500
10	Secrets of Silicon Valley		popular_comp	20.0000
11	Net Etiquette		popular_comp	NULL

Ejemplo 46. NULLIF

```
/* Obtengo como nulos si el tipo del libro no está decidido aún.*/
SELECT NULLIF(type,'UNDECIDED') "NULLIF", type
      from titles
     group by type
```

	NULLIF	type
1	business	business
2	mod_cook	mod_cook
3	popular_comp	popular_comp
4	psychology	psychology
5	trad_cook	trad_cook
6	NULL	UNDECIDED

▪ COALESCE

La función COALESCE() es similar a la función ISNULL(), pero permite devolver null cuando todos los valores o parámetros a partir del primero son nulos. Se proporciona una lista de valores más o menos larga y SQL Server elige el primer elemento que no sea nulo de la lista. Esta función viene a ser un sustituto de la función ISNULL() en una instrucción CASE para encontrar un sustituto no nulo en una lista de valores.

COALESCE (*expression [,...n]*)

Ejemplo 47.

```
/* Devuelve nulo sólo si las expresiones colocadas como argumentos son todas nulas*/
SELECT COALESCE(notes, convert(varchar,price)) "COALESCE", notes, price
      from titles
```

	COALESCE	notes	price
6	Traditional ...	Traditional...	2.9900
7	NULL	NULL	NULL
8	A survey of ...	A survey of...	22.9500
9	Muckraking r...	Muckraking ...	20.0000
10	A must-read ...	A must-read...	NULL
11	A must-fav +	A must-fav +	21.5000

▪ USER_ID

La función USER_ID() toma una cadena de nombre y devuelve el ID de la base de datos para ese nombre de usuario. La función USER_ID() es útil cuando accede a las tablas del sistema

y necesita comprobar nombres dentro de los miembros de un grupo o valores de permiso.

▪ USER NAME

La función **USER_NAME()** es el opuesto a la función **USER_ID()**. Se pasa el ID como un ID de usuario de base de datos válido y devuelve el nombre de usuario. Como sucede con **USER_ID()**, esta función es buena para comprobaciones de permisos o miembros durante el procesamiento. También es útil en la administración de errores. Cuando tiene lugar un error, puede obtener información más específica de una rutina de tratamiento de error consiguiendo el nombre de usuario y escribiéndolo en un registro o a un procedimiento para la aclaración del error.

Ejemplo 48.

```
SELECT USER_NAME(USER_ID()) "USER_NAME",
       USER_ID() "USER_ID"
```

	USER_NAME	USER_ID
1	dbo	1

La función **SUSER_SNAME()** es parecida. Nos da el nombre de usuario de conexión a windows. Ej: LUZ\luvi

▪ DATALENGTH

La función **DATALENGTH()** devuelve el número de bytes utilizado. Funciona con tipos de datos de texto e imagen así como con datos de caracteres. **DATALENGTH** es especialmente útil con los tipos de datos **varchar**, **varbinary**, **text**, **image**, **nvarchar** y **ntext** porque estos tipos de datos pueden almacenar datos de longitud variable.

Ejemplo 49.

```
SELECT length = DATALENGTH(notes), notes
      FROM titles
```

	length	notes
1	101	An overview of availa
2	76	Helpful hints on how
3	119	The latest medical an
4	91	Annotated analysis of
5	52	Favorite recipes for
6	72	Traditional French go
7	NULL	NULL

▪ COL_LENGTH

COL_LENGTH(tabla, columna) devuelve la longitud definida de una columna en una tabla. Funciona bien con la mayoría de los tipos de datos. Con esta función, puede determinar si se adecuan los datos que quiere insertar en una columna.

Ejemplo 50.

```
SELECT
COL_LENGTH('sales','stor_id') "stor_id",
COL_LENGTH('sales','ord_num') "ord_num",
COL_LENGTH('sales','ord_date') "ord_date",
COL_LENGTH('sales','qty') "qty",
COL_LENGTH('sales','payterms') "payterms",
COL_LENGTH('sales','title_id') "title_id"
```

stor_id	ord_num	ord_date	qty	payterms	ti
4	20	8	2	12	6
	stor_id	char	4		
	ord_num	varchar	20		
	ord_date	datetime	8		
	qty	smallint	2		
	payterms	varchar	12		
	title_id	tid	6		

- **CONVERT**

CONVERT() se utiliza para convertir columnas de un tipo de datos a otro. Un uso típico de esta función es especificar el formato de un campo de datos cuando se convierte desde DATETIME a CHAR o VARCHAR.

Mirar Libros en Pantalla de SQL Server para ver conversiones implícitas que se realizan.

Desde la página 197 hasta la 217 del pdf [Wiley]SQL Server 2005 Bible.pdf existen más ejemplos de estas funciones y otras funciones no vistas en este documento.

Véase *6.4 Funciones y Procedimientos almacenados de la pág. 132*, para saber como crear nuestras propias funciones/procedimientos.

5.3.4.1 Query designer

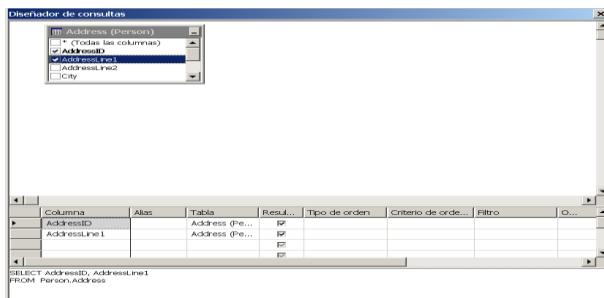


Imagen 49. Diseñador de consultas.

Ejercicio 11. McGrawHill CASE. Crear un diagrama a partir de la base de datos y responder a las siguientes consultas. Funciones.

5. Dada la tabla LIBROS, escribe la sentencia SELECT que visualice dos columnas, una con el AUTOR y otra con el apellido del autor.
6. Escribe la sentencia SELECT que visualice las columnas de AUTOR y otra columna con el nombre del autor (sin el apellido) de la tabla LIBROS.
7. A partir de la tabla LIBROS, realiza una sentencia SELECT que visualice en una columna, primero el nombre del autor y, luego, su apellido.
- 8 A partir de la tabla LIBROS, realiza una sentencia SELECT para que aparezcan los títulos ordenados por su número de caracteres.
9. Dada la tabla NACIMIENTOS, realiza una sentencia SELECT que obtenga la siguiente salida: NOMBRE, FECHANAC, FECHA_FORMATEADA, donde FECHA_FORMATEADA tiene el siguiente formato: "Nació el 12 de mayo de 1982".
13. Visualiza aquellos temas de la tabla LIBRERIA cuyos ejemplares sean 7 con el nombre de tema de 'SEVEN'; el resto de temas que no tengan 7 ejemplares se visualizarán como están.
- 14 . A partir de la tabla EMPLE, obtén el apellido de los empleados que lleven más de 15 años trabajando. A partir de la tabla EMPLE, obtén el apellido de los empleados que lleven más de 15 años trabajando.
15. Selecciona el apellido de los empleados de la tabla EMPLE que lleven más de 16 años trabajando en el departamento 'VENTAS'. Selecciona el apellido de los empleados de la tabla EMPLE que lleven más de 16 años trabajando en el departamento

16. Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario sea el mayor de su departamento. Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario sea el mayor de su departamento
17. Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario supere a la media en su departamento. Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario supere a la media en su departamento.

5.3.5 Consultas avanzadas:

5.3.5.1 Selecciones con conjuntos. UNION; INTERSECT; EXCEPT

- ✓ **Union.** Une los datos resultantes de dos querys que comparten los mismos campos:

```
SELECT *
  FROM [pubs].dbo.authors
 WHERE state='KS'
UNION ALL
SELECT *
  FROM [pubs].dbo.authors
 WHERE state='TN'
```

- ✓ **Intersect.** Realiza la intersección de dos conjuntos de datos, quedándose con aquellos datos que están en los dos.

```
SELECT *
  FROM [pubs].dbo.authors
 WHERE state='CA'
intersect
SELECT *
  FROM [pubs].dbo.authors
 WHERE city IN ('Oakland', 'San Jose', 'Berkeley')
```

- ✓ **Except.** Elimina del primer conjunto de datos los que aparezcan en el segundo.

```
SELECT *
  FROM [pubs].dbo.authors
 WHERE state='CA'
EXCEPT
SELECT *
  FROM [pubs].dbo.authors
 WHERE city IN ('Oakland', 'San Jose', 'Berkeley')
```

5.3.5.2 Selecciones con subconsultas

Ejemplos:

```
SELECT ProductName
  FROM dbo.Product
```

```
WHERE ProductCategoryID
= (Select ProductCategoryID
FROM dbo.ProductCategory
Where ProductCategoryName = 'Kite' )
```

5.3.5.3 Creación de una tabla temporal con una SELECT

SQL Server 2008 permite la utilización de la BD tempdb del sistema para poder almacenar temporalmente tablas en ella. Para utilizar este tipo de tablas se colocará delante del nombre de la tabla el signo #. La recuperación de registros de la SELECT puede redirigirse a este tipo de tablas de la siguiente manera.

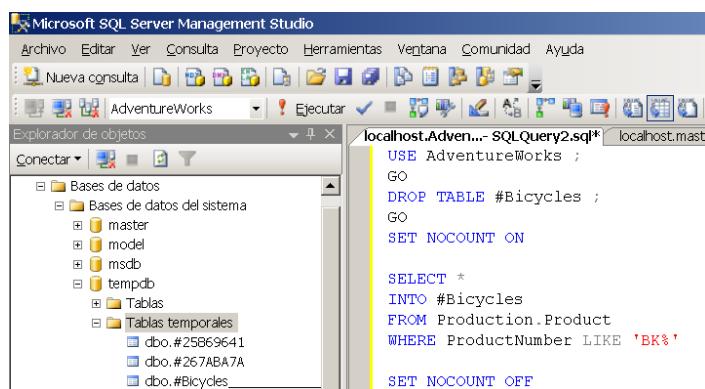


Imagen 50. Creación y selección de una tabla temporal

Ejemplo 51. Creación de tablas temporales (#)

```
USE AdventureWorks ;
GO
DROP TABLE #Bicycles ;
GO
SET NOCOUNT ON

SELECT *
INTO #Bicycles
FROM Production.Product
WHERE ProductNumber LIKE 'BK%'

SET NOCOUNT OFF

SELECT name
FROM tempdb..sysobjects
WHERE name LIKE '#Bicycles%' ;

SELECT * from #Bicycles
GO
```

Ejercicio 12. McGrawHill. CASE. Subconsultas

- Partiendo de la tabla emple, visualiza por cada oficio de los empleados del departamento VENTAS la suma de salarios.
- Selecciona aquellos apellidos de la tabla emple cuyo salario sea igual a la media del salario en su departamento.
- A partir de la tabla emple, visualiza el número de empleados de cada departamento cuyo oficio sea 'EMPLEADO'
- Desde la tabla EMPLE, visualiza el departamento que tenga más empleados cuyo oficio sea 'EMPLEADO'

5. A partir de las tablas EMPLE y DEPART, visualiza el número de departamento y el nombre de departamento que tenga más empleados cuyo oficio sea 'EMPLEADO'
6. Busca los departamentos que tienen más de 2 personas trabajando en la misma posición
7. Visualiza los nombres de los alumnos de la tabla ALUM, que aparezcan en estas dos tablas: ANTIGUOS y NUEVOS.
- Alumnos de la tabla alum que aparecen en antiguos y en nuevos
8. Escribe las distintas formas en que se puede poner la consulta anterior llegando al mismo resultado.

Escribe otras formas de la consulta anterior

9. Visualiza aquellos nombres de la tabla alum que no estén en la tabla antiguos ni en la tabla nuevos.

Alumnos que no están en antiguos ni en nuevos

10. Realiza una consulta en la que aparezca por cada centro y en cada especialidad el número de profesores.

Si el centro no tiene profesores, debe aparecer un 0 en la columna de número de profesores.

Las columnas a visualizar son: nombre del centro, especialidad, y número de profesores

11. Obtén por cada centro el número de empleados. Si el centro carece de empleados, ha de aparecer un 0 como número de empleados.

12. Obtén la especialidad con menos profesores.

13. Obtén el banco con más sucursales. Los datos a obtener son:

Nombre Banco N° Sucursales

XXXXXXXXXX XX

14. El saldo actual de los bancos de 'GUADALAJARA', 1 fila por cada banco:

Nombre Banco Saldo Debe Saldo Haber

XXXXXXXXXX XX XX

15. Datos de la cuenta o cuentas con más movimientos:

Nombre Cta N° movimientos

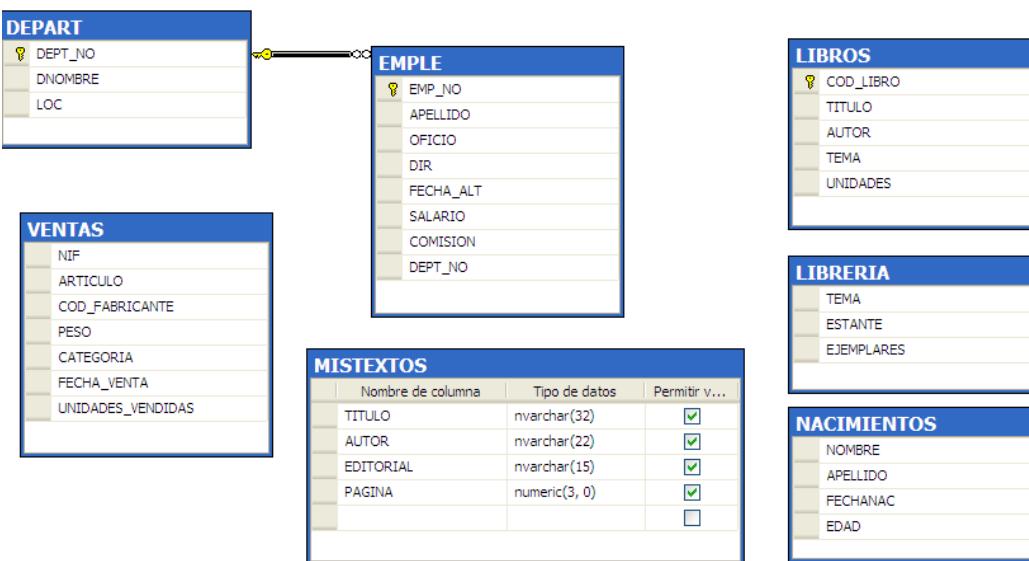
XXXXXX XX

16. El nombre de la sucursal que haya tenido más suma de reintegros:

Nombre Cta N° movimientos

XXXXX XX,XX

Ejercicio 13. Diseñar las siguientes consultas con funciones:



- Visualiza los títulos de la tabla MISTEXTOS sin los caracteres punto y comillas, y en minúscula..
- Dada la tabla LIBROS, escribe la sentencia SELECT que visualice dos columnas, una con el AUTOR y otra con el apellido del autor.
- Escribe la sentencia SELECT que visualice las columnas de AUTOR y otra columna con el nombre del autor (sin el apellido) de la tabla LIBROS.
- A partir de la tabla LIBROS, realiza una sentencia SELECT que visualice en una columna, primero el nombre del autor y, luego, su apellido.
- A partir de la tabla LIBROS, realiza una sentencia SELECT para que aparezcan los títulos ordenador por su número de caracteres.
- Dada la tabla NACIMIENTOS, realiza una sentencia SELECT que obtenga la siguiente salida: NOMBRE, FECHANAC, FECHA_FORMATEADA, donde FECHA_FORMATEADA tiene el siguiente formato:
“Nació el 12 de mayo de 1982”
- Visualiza aquellos temas de la tabla LIBRERÍA cuyos ejemplares sean 7 con el nombre de tema de ‘SEVEN’; el resto de temas que no tenga 7 ejemplares se visualizarán como están.
- A partir de la tabla EMPLE, obtén el apellido de los empleados que lleven más de 15 años trabajando.
- Selecciona el apellido de los empleados de la tabla EMPLE que lleven más de 16 años trabajando en el departamento
- Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario sea el mayor de su departamento
- Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario supere a la media en su departamento.
- Desde la tabla EMPLE, visualiza el departamento que tenga más empleados cuyo oficio sea ‘EMPLEADO’
- A partir de las tablas EMPLE y DEPART, visualiza el número de departamento y el nombre de departamento que tenga más empleados cuyo oficio sea ‘EMPLEADO’
- Busca los departamentos que tienen más de 2 personas trabajando en la misma posición

Ejercicio 14. Considerando la base de datos EMPLEADOS, realizar las siguientes consultas desde el administrador corporativo (OFICINAS, REPVENTAS, PRODUCTOS, CLIENTES y

PEDIDOS).

1. ¿Cuál es el nombre, cuota y ventas del empleado número 107?
 2. Listar los nombres, oficinas y fechas de contrato de todos los vendedores.
 3. Listar el nombre y fecha de contrato de cualquier vendedor cuyas ventas estén entre 100.000 y 300.000.
 4. Mostrar las oficinas en donde las ventas no corresponden al empleado número 108.
 5. Listar los vendedores que trabajan en Neww York, Atlanta o Denver.
 6. Hallar los vendedores a los que se ha asignado ya una oficina.
 7. Media de edad de los diferentes puestos (título) existentes.
 8. Nombre de los representantes de ventas de aquellos clientes que poseen un límite de crédito superior a la media.
 9. Listado de empresas cliente que hayan realizado algún pedido.
- 10. Consulta con parámetros.** Nombre de representantes de ventas mayores de una edad introducida como parámetro.

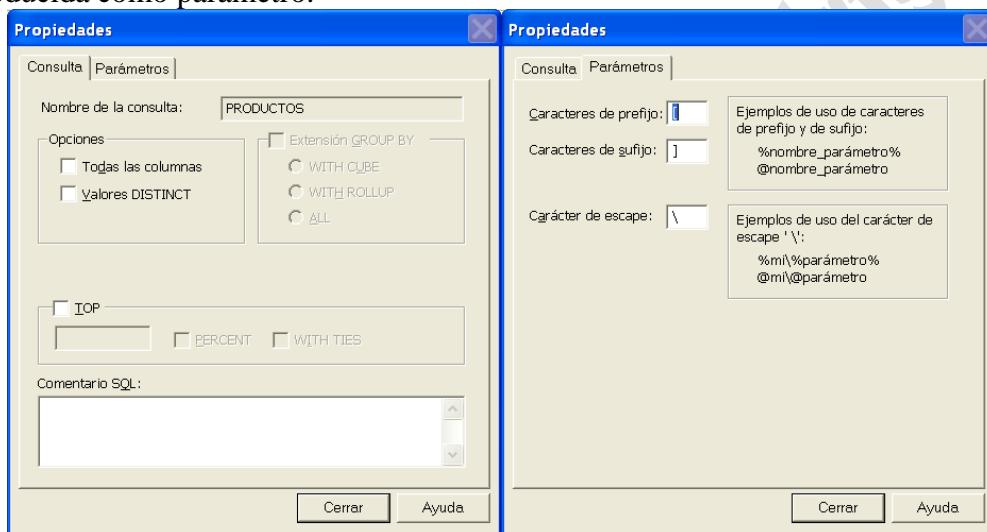


Imagen 51. Utilización del DISTINCT y definición de los símbolos para utilizar parámetros en consultas

11. Insertar en la tabla clientes el siguiente cliente:

NUM_CLIE: 2125 EMPRESA: 'AVIACO' REPCLIE:107 LIMITE_CREDITO: 50000

12. Actualizar la empresa del cliente 2125 'AVIACOM'

13. Eliminar el cliente 2125

14. Crea una nueva tabla temporal con todos los datos de los clientes que superen un crédito de 60000.

15. Crear un nuevo campo nuevo en una query.

Ejercicio 15. Sobre la base de datos de ESTUDIANTES:

1. Obtener el número, nombre y tarifa de los cursos con tarifa mínima no nula.
2. Obtener el número, nombre y tarifa de los cursos con tarifa menor que la media.
3. Obtener el número, nombre y tarifa de los cursos con tarifa mayor o igual que el sueldo de cualquier miembro del personal.
4. Obtener el nombre y cargo de cada miembro del personal que trabaja en el edificio de Humanidades.
5. Obtener el número y nombre de cualquier miembro de la facultad que sea jefe de cualquier departamento que ofrezca un curso de seis créditos.
6. Obtener el número, cargo e identificador de cada departamento de cada miembro del personal



asignado a un departamento no existente (sin identificación)

7. Obtener, para cada departamento que ofrezca cursos, el identificador de departamento y la tarifa media de los cursos ofrecidos por el departamento, siempre y cuando sea mayor que la tarifa media de todas las tarifas de cursos.
8. Obtener el número de curso, timbre y tarifa de cada curso cuya tarifa exceda del salario de cualquier miembro de personal.
9. Obtener el número de curso, nombre y tarifa de cada toro cuya tarifa sea menor que todos los salarios de todos los miembros del personal.
10. Obtener, para cada departamento que ofrezca cursos, el identificador de ese departamento seguido del número, nombre y tarifa del departamento que organiza el curso con mayor tarifa.
11. Obtener el nombre y el cargo de cualquier miembro del personal asignado a un departamento existente.
12. Obtener el nombre y el cargo de cualquier miembro del personal que no esté asignado a un departamento existente.
13. Obtener la longitud real de cada nombre de curso ofrecido por el departamento de Teología.
14. Obtener la primera letra de todos los nombres de cursor de Informática y Ciencias de la Información. Obtener también los caracteres que aparecen en las posiciones tercera, cuarta, quinta y sexta. Por último mostrar los valores del número de curso sin el primer carácter.

5.3.6 INSERT

```
INSERT [ INTO ]
{ table_name WITH ( < table _ hint_ limited > [ ...n ] ) | view_name
) rowset function limited
}
{   [ ( column_list ) ]
VALUES
( { DEFAULT | NULL | expression } [ ,...n ] ) | derived_table | execute statement }
| DEFAULT VALUES <table_hint_limited> ::= { FASTFIRSTROW
{
HOLDLOCK |
PAGLOCK |
READCOMMITTED |
REPEATABLEREAD |
ROWLOCK |
SERIALIZABLE |
TABLOCK |
TABLOCKX |
UPDLOCK
}
```

}

Ejemplo 52. Uso de Insert

INSERT authors

VALUES ('123-45-6789','Víctor','Vergel',666999333,'C/ Jerez','Valladolid','SP',47000,0)

INSERT authors (au_id,au_lname,au_fname,contract)

VALUES ('123-45-5555','Víctor José','Vergel Rodriguez',0)

-- La siguiente instrucción produce error, pero no por estar sintácticamente incorrecto

-- sino porque el campo au_lname no permite nulos (con una cláusula CHECK).

INSERT authors (au_id,au_lname,au_fname,contract)

VALUES ('123-45-5555',NULL,'Vergel Rodriguez',0)

INSERT se puede utilizar también en conjunción con la instrucción SELECT, para insertar en una tabla lo recuperado de una consulta.

Ejemplo 53. Inserción de datos recuperados con un SELECT

```
CREATE TABLE [dbo].[hubble2] (
    [Id] [int] NULL,
    [name] [varchar](100) COLLATE Modern_Spanish_CI_AS NULL
) ON [PRIMARY]

insert into hubble2
select * from hubble_galaxies

delete hubble2
```

5.3.7 Bulk insert

Utilizado para la copia masiva de datos desde un fichero de texto. Sintaxis:

```
BULK INSERT [[ 'database_name' ] [ 'owner' ].] { 'table_name' FROM 'data_file' }
[WITH
(
    [ BATCHSIZE [=batch_size] ]
    [ [,] CHECK_CONSTRAINTS ]
    [ [,] CODEPAGE [= 'ACP' | 'OEM' | 'RAW' | 'code_page' ] ]
    [ [,] DATAFILETYPE =
        { 'char' | 'native' | 'widechar' | 'widenative' } ]
    [ [,] FIELDTERMINATOR [= 'field_terminator' ] ]
    [ [,] FIRSTROW [= first_row ] ]
    [ [,] FIRE_TRIGGERS ]
    [ [,] FORMATFILE = 'format_file_path' ]
    [ [,] KEEPIDENTITY ]
    [ [,] KEEPNULLS ]
    [ [,] KILOBYTES_PER_BATCH [= kilobytes_per_batch] ]
    [ [,] LASTROW [= last_row ] ]
    [ [,] MAXERRORS [= max_errors ] ]
    [ [,] ORDER ( { column [ ASC | DESC ] } [ ,...n ] ) ]
    [ [,] ROWS_PER_BATCH [= rows_per_batch] ]
    [ [,] ROWTERMINATOR [= 'row_terminator' ] ]
    [ [,] TABLOCK ]
)
]
```

Ejemplo 54.

Tiendo la siguiente tabla creada:

```
CREATE TABLE [P] (
    [pn] [nvarchar](2) COLLATE Modern_Spanish_CI_AS NULL ,
    [pnombre] [nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL ,
    [color] [nvarchar](20) COLLATE Modern_Spanish_CI_AS NULL ,
    [peso] [smallint] NULL ,
    [ciudad] [nvarchar](50) COLLATE Modern_Spanish_CI_AS NULL
) ON [PRIMARY]
```

importamos datos desde un fichero:

```
BULK INSERT "Prueba 2".dbo.p
```



```
FROM 'C:\Documentos\ESI\SQL Server\Ejemplo 58-importar desde fichero.txt'
WITH
(
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
```

5.3.8 UPDATE

```
UPDATE
{
    table_name WITH ( < table_hint_limited > [ ...n ] )
    | view_name
    | rowset_function_limited
}
SET
{ column_name = { expression | DEFAULT | NULL }
| @variable = expression
| @variable = column = expression } [ ,...n ]
{ { [ FROM { < table_source > } [ ,...n ] ]
[ WHERE
    < search_condition > ] }
|
[ WHERE CURRENT OF
{ { [ GLOBAL ] cursor_name } | cursor_variable_name }
]
[ OPTION ( < query_hint > [ ,...n ] ) ]
```

Ejemplo 55. Uso de update

```
UPDATE authors
    SET phone = '666999444'
    WHERE au_lname = 'Víctor José'

update employee
SET hire_date = CONVERT(DATETIME, '1992-09-27 00:00:00', 102)
WHERE (emp_id = 'PMA42628M')

update employee
SET hire_date = '1992/28/11'
WHERE (emp_id = 'PMA42628M')

select * from employee
WHERE (emp_id = 'PMA42628M')
```

5.3.9 DELETE

```
DELETE
[FROM ]
{ table_name WITH ( < table_hint_limited > [ ...n ] )
| view_name
| rowset_function_limited
}
```

```
[ FROM { < table_source > } [ ,...n ] ]
[ WHERE
{ < search_condition >
| { [ CURRENT OF
{ { [ GLOBAL ] cursor_name }
| cursor_variable_name
}
]
}
}
]
```

Ejemplo 56. Uso de DELETE

```
DELETE FROM authors
WHERE au_id = '123-45-6788'
```

```
DELETE FROM authors
WHERE au_lname like 'Víctor%'
```

5.3.9.1 TRUNCATE TABLE

Elimina todas las filas de la tabla indicada, es similar a DELETE pero no se registra la eliminación de cada fila individual.

```
TRUNCATE TABLE name
```

Ejercicio 16. McGraw-Hill. Actualizaciones, borrados....

- 1) Modifica el número de plazas con un valor igual a la mitad en aquellos centros con menos de dos profesores
- 2) Elimina los centros que no tengan personal
- 3) Añade un nuevo profesor en el centro o en los centros cuyo número de administrativos sea 1 en la especialidad de ICIOMA, con DNI 8790055 y de nombre 'Clara Salas'
- 4) Borra al personal que esté en centros de menos de 300 plazas y con menos de dos profesores
- 5) Borra a los profesores que estén en la tabla PROFESORES y que no estén en la tabla PERSONAL.
- 6) Da de alta un nuevo artículo de 'Primera' categoría para los fabricantes de 'FRANCIA' y abastecce con cinco unidades de este artículo a todas las tiendas y en la fecha de hoy.
- 7) Inserta un pedido de 20 unidades en la tienda '1111-A' con el artículo que mayor número de ventas haya realizado.
- 8) Da de alta una tienda en la provincia de 'MADRID' y abastécela con 20 unidades de cada uno de los artículos existentes.
- 9) Da de alta dos tiendas en la provincia de 'SEVILLA' y abastécea con 30 unidades de artículos del nombre de fabricante 'GALLO'.
- 10) INSERT INTO ventas (nif,unidades_vendidas)
 SELECT nif,'10' FROM tiendas WHERE poblacion='Toledo'
- 11) UPDATE PEDIDOS
 SET UNIDADES_PEDIDAS=UNIDADES_PEDIDAS+10 WHERE PEDIDOS.ARTICULO
 IN
 (SELECT e.ARTICULO FROM (SELECT SUM(UNIDADES_VENDIDAS)

TOTAL,ARTICULO FROM VENTAS GROUP BY ARTICULO) e

WHERE e.TOTAL>30) AND PEDIDOS.NIF='5555-B'

12) UPDATE Tiendas SET NIF='2222-A'

WHERE NIF=1111-A'

13) UPDATE articulos SET articulos.categoría='Segunda'

WHERE articulos.categoría='Primera'

AND articulos.cod_fabricante IN(SELECT fabricantes.cod_fabricante FROM fabricantes WHERE fabricantes.pais=UPPER('Italia'))

14) UPDATE PEDIDOS SET unidades_pedidas=existencias*0.2

WHERE nif IN (SELECT DISTINCT nif FROM articulos,pedidos

WHERE articulos.articulo=pedidos.articulo AND articulos.existencias< pedidos.unidades_pedidas)

15) DELETE FROM Tiendas

WHERE TIENDAS.NIF IN ((SELECT DISTINCT TIENDAS.NIF FROM TIENDAS)MINUS (SELECT DISTINCT VENTAS.NIF FROM VENTAS))

16) DELETE FROM Tiendas

WHERE TIENDAS.NIF IN ((SELECT DISTINCT TIENDAS.NIF FROM TIENDAS)MINUS (SELECT DISTINCT VENTAS.NIF FROM VENTAS))

AND TIENDAS.NIF IN ((SELECT DISTINCT TIENDAS.NIF FROM TIENDAS)MINUS (SELECT DISTINCT PEDIDOS.NIF FROM PEDIDOS))

17) DELETE FROM pedidos WHERE categoría='Primera' AND cod_fabricante =
(SELECT cod_fabricante FROM fabricantes WHERE pais=upper('Belgica'))

18) DELETE FROM Pedidos WHERE nif IS NULL

19) SELECT UNIDADES_PEDIDAS-1

FROM pedidos

WHERE FECHA_PEDIDO = (SELECT MAX(FECHA_PEDIDO) FROM PEDIDOS
WHERE NIF='5555-B') ORDER BY FECHA_PEDIDO

Ejercicio 17.

Dada las siguientes estructuras de tablas responder a las siguientes consultas:

P				S				
PN	PNOMBRE	COLOR	PESO	Ciudad	sn	snombre	estado	ciudad
P1	tuerca	verde	12	París	S1	Salazar	20	Londres
P2	perno	rojo	17	Londres	S2	Jaimes	10	París
P3	birlo	azul	17	Roma	S3	Bernal	30	París
P4	birlo	rojo	14	Londres	S4	Corona	20	Londres
P5	leva	azul	12	París	S5	Aldana	30	Atenas
P6	engrane	rojo	19	París				

SP		
Sn	Pn	cant
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

- Obtener el nombre de los proveedores que suministran la pieza con el código P2
- Obtener el nombre de los proveedores que suministran por lo menos una pieza roja.

3. Obtener el nombre de las piezas de color rojo suministradas por los proveedores de la ciudad de Londres.
4. Obtener el código de los proveedores que suministran alguna de las piezas que suministra el proveedor con el código S2.
5. Obtener los datos de los envíos de más de 100 unidades, mostrando también el nombre del proveedor y el de la pieza.
6. Obtener el nombre de los proveedores que suministran todas las piezas.
7. Obtener el código de los proveedores que suministran, al menos, todas las piezas suministradas por el proveedor con código S2.
8. Obtener el nombre de los proveedores que no suministran la pieza con el código P2.
9. Obtener los datos de los proveedores que solo suministran piezas de color rojo.
10. Obtener el nombre de los proveedores que suministran, al menos, todas las piezas que se almacenan en la ciudad de Paris.
11. Obtener los datos del envío de más piezas. (Sólo en SQL)
12. Para cada proveedor, mostrar la cantidad total de piezas que envía, la cantidad media y el número de envíos. (Solo SQL)
13. Obtener el código de los proveedores que realizan envíos en cantidades superiores a la cantidad media por envío.
14. Para cada ciudad en la que se almacenan piezas, obtener el número de piezas que almacena de cada color distinto.
15. Todos los proveedores existentes y el número de productos que suministran.
16. De los proveedores que me suministran algún producto quiero saber cuantos productos distintos me suministran en total.

Ejercicio 18. Ejercicio ciclismo.

El siguiente esquema relacional representa una base de datos que almacena información sobre una vuelta ciclista. Ningún atributo acepta nulos, a menos que se especifique lo contrario.

EQUIPO(nombreequipo, director)

Datos de los distintos equipos ciclistas que participan en la vuelta: nombre del equipo y nombre de su director.

CICLISTA(dorsal, nombre, añonacim, nombreequipo)

Datos de los ciclistas que componen los distintos equipos: número del dorsal, nombre del ciclista, año de nacimiento del ciclista y nombre del equipo al que pertenece.

CICLISTA.nombreequipo es clave ajena a EQUIPO; regla de borrado: propagar.

ETAPA(numetapa, kms, salida, llegada, dorsal)

Datos de las etapas que componen la vuelta ciclista: número de la etapa (las etapas se numeran consecutivamente: 1, 2, ...), kilómetros que tiene la etapa, nombre de la población de donde sale la etapa, nombre de la población donde se encuentra la meta de la etapa y número del dorsal del ciclista que ha ganado la etapa.

ETAPA.salida y ETAPA.llegada están definidas sobre el mismo dominio.

ETAPA.dorsal es clave ajena a CICLISTA; acepta nulos (aún no se ha corrido la etapa); regla de borrado: restringir.

PUERTO(nompuerto, altura, categoría, pendiente, numetapa, dorsal)

Datos de los puertos de montaña que visita la vuelta ciclista: nombre del puerto, altura máxima, categoría del puerto: primera, especial, etc., pendiente media del puerto, número de la etapa donde se pasa por él y número del dorsal que ha ganado el puerto al pasar en primera posición.

PUERTO.numetapa es clave ajena a ETAPA; regla de borrado: propagar.

PUERTO.dorsal es clave ajena a CICLISTA; acepta nulos (aún no se ha corrido la etapa que pasa por el puerto); regla de borrado: restringir.

MAILLOT(código, tipo, color, premio)

Datos de los premios que se otorgan mediante los distintos maillots: código del maillot, tipo de clasificación que premia ese maillot: general, montaña, etc., color de la camiseta asociada e importe del premio que corresponde al ciclista que termine la vuelta llevando el maillot.

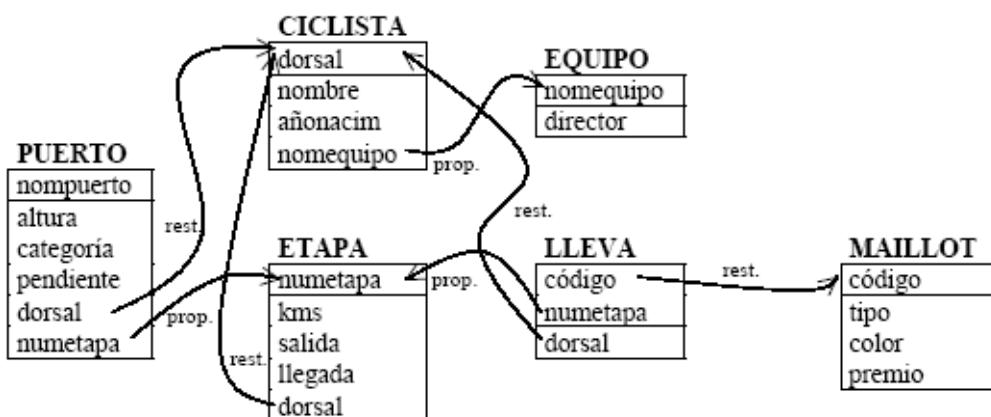
LLEVA(código, numetapa, dorsal)

Información sobre qué ciclistas han llevado cada maillot en cada una de las etapas.

LLEVA.código es clave ajena a MAILLOT; no acepta nulos; regla de borrado: restringir.

LLEVA.numetapa es clave ajena a ETAPA; no acepta nulos; regla de borrado: propagar.

LLEVA.dorsal es clave ajena a CICLISTA; no acepta nulos; regla de borrado: restringir.



1. Obtener los datos de las etapas que pasan por algún puerto de montaña y que tienen salida y llegada en la misma población.
2. Obtener las poblaciones que tienen la meta de alguna etapa, pero desde las que no se realiza ninguna salida.
3. Obtener el nombre y el equipo de los ciclistas que han ganado alguna etapa llevando el maillot amarillo, mostrando también el número de etapa.
4. Obtener los datos de las etapas que no comienzan en la misma ciudad en que acaba la etapa anterior.
5. Obtener el número de las etapas que tienen algún puerto de montaña, indicando cuantos tiene cada una de ellas.

6. Obtener el nombre y la edad de los ciclistas que han llevado dos o más maillots en una misma etapa.
7. Obtener los datos de los ciclistas que han vestido todos los maillots (no necesariamente en la misma etapa).
8. Obtener el código y el color de aquellos maillots que solo han sido llevados por ciclistas de un mismo equipo.
9. Obtener los números de las etapas que no tienen puertos de montaña.
10. Obtener la edad media de los ciclistas que han ganado alguna etapa
11. Obtener el nombre de los puertos de montaña que tienen una altura superior a la altura media de todos los puertos.
12. Obtener las poblaciones de salida y de llegada de las etapas donde se encuentran los puertos con mayor pendiente.
13. Obtener el dorsal y el nombre de los ciclistas que han ganado los puertos de mayor altura.
14. Obtener los datos de las etapas cuyos puertos (todos) superan los 1300 metros de altura.
15. Obtener el nombre de los ciclistas que pertenecen a un equipo de más de cinco ciclistas y que han ganado alguna etapa, indicando también cuantas etapas han ganado.

Módulo 6 TRANSACT SQL. TRANSACCIONES.

Realmente Transact-SQL es el lenguaje sql adaptado concretamente a SQL Server, por lo que incluye también SELECT, INSERT y el resto de sentencias existentes.

6.1 *TCL – Transaction Control Language*

Podemos definir un punto en la BD de coherencia de datos, es decir, con la instrucción **BEGIN TRANSACTION** estamos marcando el inicio de una serie de operaciones que modificarán la BD. Si durante la modificación se produce un error podremos volver a nuestro punto de coherencia con **ROLLBACK TRANSACTION**. Si por el contrario esas operaciones son satisfactorias las confirmaremos con **COMMIT TRANSACTION**, equivalente a esta última operación es **COMMIT WORK**, varía en que no admite especificar el nombre de transacción.

Cada instrucción Transact-SQL se confirma o se revierte cuando finaliza. Si una instrucción termina correctamente, se confirma; si encuentra un error, se revierte. Si realizamos un select en la misma sesión sin haber hecho un COMMIT sí veremos los datos “insertados”, sólo cuando hagamos el COMMIT lo veremos desde cualquier sesión abierta con SQL Server 2008. Lo que significa es que hasta que no realicemos COMMIT no quedarán confirmados los cambios para el resto de sesiones.

Sintaxis:

```
BEGIN TRAN [ SACTION ] [ transaction_name | @tran_name_variable
[ WITH MARK [ 'description' ] ] ] -- Restaurará también el registro de transacciones
```

```
ROLLBACK [ TRAN [ SACTION ]
[ transaction_name | @tran_name_variable
| savepoint_name | @savepoint_variable ] ]
```

```
COMMIT [ TRAN [ SACTION ] [ transaction_name | @tran_name_variable ] ]
```

```
COMMIT [ WORK ]
```

Con **SAVE TRANSACTION** establecemos un punto de restauración dentro de una transacción, es similar a BEGIN TRANSACTION, pero podemos usarlo para establecer puntos intermedios dentro del cuerpo de un BEGIN TRANSACTION. Una vez utilizado COMMIT TRANSACTION no se podrá recuperar este punto de restauración.

```
SAVE TRAN [ SACTION ] { savepoint_name | @savepoint_variable }
```

Resumiendo, para comenzar una transacción usaremos sólo una vez **BEGIN TRANSACTION**, pero podremos definir más puntos de restauración con **SAVE TRANSACTION** hasta finalizar la transacción iniciada con el BEGIN.

Las transacciones utilizadas se conocen como explícitas, las implícitas se agrupan por la instrucción GO.

Ejemplo 57. Dependiendo donde vayamos colocando el GO la transacción será confirmada o no hasta ese punto. El GO envía a motor de la base de datos una serie de transacciones que serán confirmadas si no se produce ningún error. Poner por lo tanto un solo GO en la última línea es equivalente a no poner ninguno.

```
USE Prueba;
GO
CREATE TABLE TestBatch (Cola INT PRIMARY KEY, Colb CHAR(3));
GO
INSERT INTO TestBatch VALUES (1, 'aaa');
INSERT INTO TestBatch VALUES (2, 'bbb');
INSERT INTO TestBatch VALUSE (3, 'ccc'); -- Syntax error.
GO
SELECT * FROM TestBatch; -- Returns no rows.
GO
```

```
USE Prueba;
--GO
CREATE TABLE TestBatch (Cola INT PRIMARY KEY, Colb CHAR(3));
--GO
INSERT INTO TestBatch VALUES (1, 'aaa');
INSERT INTO TestBatch VALUES (2, 'bbb');
--GO
INSERT INTO TestBatch VALUSE (3, 'ccc'); -- Syntax error.
--GO
SELECT * FROM TestBatch; -- Returns no rows.
--GO
```

Ejemplo 58. Transacciones explícitas. Podemos ver que la media de Alfredo2 es incorrecta, porque no toma en cuenta para calcular esa media a Alfredo. Para evitar este error podemos hacer la segunda parte del ejemplo:

```
CREATE TABLE [emp2] (
    [empno] [smallint] NOT NULL ,
    [ename] [nvarchar] (10) COLLATE Modern_Spanish_CI_AS NULL ,
    [sal] [smallmoney] NULL ,
    [salmedio] [smallmoney] NULL , -- Almacenamos el salario medio en el momento de entrar en la
empresa
    CONSTRAINT [PK_emp2] PRIMARY KEY CLUSTERED
    (
        [empno]
    ) ON [PRIMARY]
)

declare @SalMedio smallint
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (10, N'Jose', 600,@SalMedio)
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (11, N'Ivan', 1100,@SalMedio)
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (12, N'Maria', 1300,@SalMedio)
SET @SalMedio = (select AVG(sal) FROM emp2)
```

INSERT INTO emp2 VALUES (10, N'Alfredo', 1000, @SalMedio) -- Se produce error porque el empno 10 ya existe.

```
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (14, 'Alfredo2', 2000, @SalMedio)
SELECT * FROM emp2
```

```
drop table emp2
```

El que podamos recuperar el estado inicial de la tabla para corregir el error de la inserción se solucionará creando un punto de restauración:

```
BEGIN TRANSACTION Trans1
declare @SalMedio smallint
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (10, N'Jose', 600, @SalMedio)
```

SAVE TRANSACTION Trans2 -- No podemos utilizar BEGIN TRANSACTION en lugar de SAVE porque ya estamos en una transacción

```
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (11, N'Ivan', 1100, @SalMedio)
```

```
SAVE TRANSACTION Trans3
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (12, N'Maria', 1300, @SalMedio)
```

```
SAVE TRANSACTION Trans4
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (10, N'Alfredo', 1000, @SalMedio)
```

```
SAVE TRANSACTION Trans5
SET @SalMedio = (select AVG(sal) FROM emp2)
INSERT INTO emp2 VALUES (14, N'Alfredo2', 2000, @SalMedio)
```

```
SELECT empno EmpnoAntes, ename EnameAntes, sal SalAntes, salmedio SalMedioAntes FROM emp2
```

-- COMMIT TRANSACTION Trans1 -- Si realizamos el commit ya no podremos deshacer
ROLLBACK TRANSACTION Trans3

```
SELECT * FROM emp2
```

```
drop table emp2
```

Como ya comentamos por defecto cada instrucción se confirma automáticamente. La instancia de Database Engine (Motor de base de datos) inicia automáticamente una transacción la primera vez que ejecuta una de estas instrucciones: ALTER TABLE, INSERT, CREATE, OPEN, DELETE, REVOKE, DROP, SELECT, FETCH, TRUNCATE TABLE, GRANT, UPDATE. Para establecer las transacciones por defecto usaremos SET IMPLICIT_TRANSACTIONS ON.

Ejercicio 19. Crea una tabla temporal a partir de la tabla authors de pubs. Actualiza a 1 todos los registros de la temporal en el campo contract siempre que sean del estado 'CA' y realiza una transacción para deshacer

estos cambios.

6.2 Elementos auxiliares: USE, variables, GO, EXECUTE, PRINT

6.2.1 Comando USE

Sirve para cambiar de base de datos por defecto. Pasará a la BD que se le indique a continuación.

USE "Ejercicio 5 sp"

6.2.2 Variables

6.2.2.1 Variables definidas por el usuario

DECLARE @nombrevariable tipo_dato [, @nombrevariable2 tipo_dato2,...] – Declaro una nueva variable
SELECT @nombrevariable=expresión – Defino el valor de la variable
Ó
SET @nombrevariable=expresión – Defino el valor de la variable

Debemos conocer también un uso concreto de la sentencia SELECT. Podemos utilizarla para mostrar valores, no sólo datos de tablas:

SELECT 'abc';
Resultado: abc

Si en el where colocamos una condición falsa el SELECT no será ejecutado.

DECLARE @test NVARCHAR(15);
SET @test = 'z';
SELECT @test = 'abc' WHERE 1<0;
SELECT @test;
Resultado:
z

Ejemplo 59. Utilización de variables.

```
USE pubs
DECLARE @var1 nvarchar(30)
SELECT @var1 = 'Generic Name'

SELECT @var1
-----
SELECT @var1 = au_lname
FROM authors
WHERE au_fname like 'A%'

SELECT @var1 AS 'Company Name'

select * from authors where au_fname like 'A%'
```

6.2.2.2 Variables de sistema

@@ERROR

Devuelve el número de error de la última ejecución errónea de una sentencia en SQL Server.

Ejemplo 60.

```
USE pubs
GO
UPDATE authors SET au_id = '172 32 1176'
WHERE au_id = "172-32-1176"
```

```
IF @@ERROR = 547
    print " Violacion de restriccion"
-----
IF @@ERROR = 0
    -- No se ha producido error
ELSE
    -- Se ha producido un error
END
```

Para poder obtener toda la información posible sobre un error podríamos hacer lo siguiente:

```
SELECT * FROM master.dbo.sysmessages
```

	error	severity	dlevel	description	msglangid
1	1	10	0	Version date of last upgrade: 10/11/90.	1033
2	1	10	0	Fecha de actualización de la versió...	3082
3	21	10	0	Warning: Fatal error %d occurred at...	1033
4	21	10	0	Advertencia: error fatal %1! el %2!...	3082
5	102	15	0	Incorrect syntax near '%.*ls'.	1033
6	102	15	0	Sintaxis incorrecta cerca de '%1!'.	3082
7	103	15	0	The %S MSG that starts with '%.*ls'...	1033

Imagen 52. Tabla que recoge toda la información sobre mensajes de sistema

En esta tabla se almacenan en todos los idiomas todos los mensajes de error. Cuando msglangid es 3082 será en español. Podemos ver los idiomas definidos en select * FROM SYSLANGUAGES.

Si quisiéramos ver el último error que mensaje tiene:

```
SELECT * FROM master.dbo.sysmessages
where msglangid=3082 and
error=@@error
```

Cuando al ejecutar una sentencia se produce un error, automáticamente en tiempo de ejecución se llama a la función RAISERROR para mostrar el error al usuario:

```
DECLARE @DBID INT
SET @DBID = DB_ID() -- Función que devuelve el identificador de BD.
```

```
DECLARE @DBNAME NVARCHAR(128)
SET @DBNAME = DB_NAME()
```

RAISERROR('El identificador de base de datos es:%d, el nombre de la base de datos es: %s.',16,1,
@DBID, @DBNAME) – Parámetros: mensaje, gravedad 0-25, estado del 1 al 127.

La función RAISERROR tiene la siguiente sintaxis:

```
RAISERROR ( { msg_id | msg_str | @local_variable }
{ ,severity ,state }
[ ,argument [ ,...n ] ]
[ WITH option [ ,...n ] ]
```

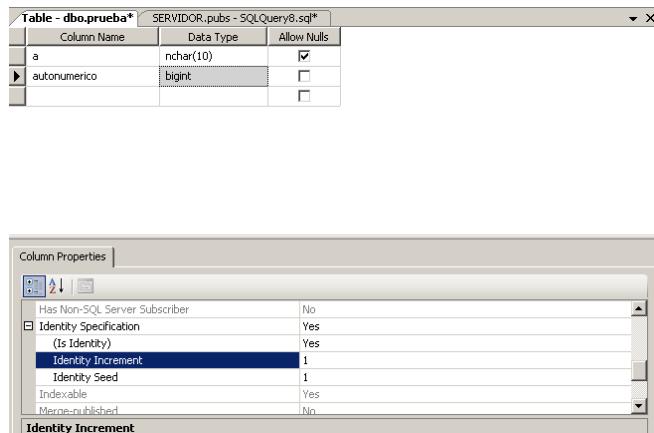
-- msg_id . Identificador de error, los definidos por el usuario aparecen a partir del número 5000.
-- msg_str: Mensaje de error
-- En el mensaje se puede colocar % junto con una letra para mostrar determinados valores.

Type Specification	Represents
d or i	Signed integer
o	Unsigned octal
s	String
u	Unsigned integer
x or X	Unsigned hexadecimal

Todos los usuarios pueden especificar los niveles de **gravedad** del 0 al 18. Sólo los miembros de la función fija de servidor sysadmin o los usuarios con permisos ALTER TRACE pueden especificar los niveles de gravedad del 19 al 25. Para los niveles de gravedad del 19 al 25, se necesita la opción WITH LOG. Los niveles de gravedad del 20 al 25 se consideran muy negativos. Si se encuentra un nivel de gravedad de este tipo, la conexión de cliente termina tras recibir el mensaje, y el error se incluye en el registro de errores y en el registro de la aplicación. En cuanto al **estado** si se genera el mismo error definido por el usuario en varias ubicaciones, el uso de un único número de estado para cada ubicación puede ayudar a averiguar qué sección del código está generando los errores. Para más información véase 6.6. Try/catch de la pág. 148. También existe otro ejemplo en Ejemplo 72. de la página 132.

@@IDENTITY

Devuelve el último valor identidad insertado en un campo autonumérico.



Ejemplo 61. Creo una tabla jobs con un campo autoincrementable de manera que después de realizar dos inserciones puedo ver el valor de @@IDENTITY.

```
CREATE TABLE jobs
(
    job_id smallint
        IDENTITY(1,1) PRIMARY KEY CLUSTERED,
    job_desc   varchar(50) NOT NULL
        DEFAULT 'New Position - title not formalized yet',
    min_lvl tinyint NOT NULL
        CHECK (min_lvl >= 10),
    max_lvl tinyint NOT NULL
        CHECK (max_lvl <= 250)
)
```

```
INSERT INTO jobs (job_desc, min_lvl, max_lvl) VALUES('job_desc', 11, 50)
INSERT INTO jobs (job_desc, min_lvl, max_lvl) VALUES('job_desc', 12, 100)
SELECT * from jobs
```

```
SELECT @@IDENTITY
```

@@ROWCOUNT

Devuelve el número de filas afectadas en la última instrucción sql.

Ejemplo 62. Uso de Rowcount

```
USE pubs
SELECT * from authors
SELECT @@ROWCOUNT
```

@@TRANCOUNT

Número de transacciones activas, es decir, que han sido iniciadas con BEGIN TRANSACTION.

Ejemplo 63. Uso de TRANCOUNT

```
USE pubs
```

```

BEGIN TRANSACTION
UPDATE authors SET au_lname = upper(au_lname)
WHERE au_lname = 'White'
IF @@ROWCOUNT = 2
    COMMIT TRAN

IF @@TRANCOUNT > 0
BEGIN
    PRINT 'Realizaremos una vuelta a un punto consistente'
    ROLLBACK TRAN
END

```

@@SPID

Devuelve el identificador (Id.) de proceso de servidor del proceso de usuario actual.

Ejemplo 64 Uso de SPID

```
SELECT @@SPID AS 'ID', SYSTEM_USER AS 'Login Name', USER AS 'User Name'
```

Una función relacionada con @@SPID podría ser **sp_who**. Esta función devuelve los procesos que está el servidor de BD ejecutando en este momento:

	spid	ecid	status	loginame	hostname	blk	dbname	cmd
6	6	0	background	sa		0	master	TASK MANAGER
7	7	0	sleeping	sa		0	NULL	CHECKPOINT SLEEP
8	8	0	background	sa		0	master	TASK MANAGER
9	9	0	background	sa		0	master	TASK MANAGER
10	10	0	background	sa		0	master	TASK MANAGER
11	11	0	background	sa		0	master	TASK MANAGER
12	51	0	runnable	LUZ\luvi	LUZ	0	pubs	SELECT
13	52	0	sleeping	LUZ\luvi	LUZ	0	master	AWAITING COMMAND
14	53	0	sleeping	LUZ\luvi	LUZ	0	Prueba 2	AWAITING COMMAND
15	54	0	sleeping	LUZ\luvi	VICTOR	0	pubs	AWAITING COMMAND
16	55	0	sleeping	LUZ\luvi	VICTOR	0	master	AWAITING COMMAND

Imagen 53. Conexiones actuales al servidor

De esta manera podríamos finalizar algún proceso a través del comando **KILL** numeroproceso, teniendo en cuenta que no podemos matar el propio proceso donde se lanza el kill y procesos como AWAITING COMMAND, CHECKPOINT SLEEP, LAZY WRITER, LOCK MONITOR, SIGNAL HANDLER. Además para usar este comando deberemos tener permisos especiales: sysadmin y processadmin.

6.2.3 GO

Marcamos el fin de un lote de instrucciones. Se usa principalmente para definir el ámbito del valor de una determinada variable. Además justo en el momento en que colocamos el GO se produce el envío del lote a SQL Server, es decir, no se envía línea a línea si no que se espera hasta encontrar el GO.

Ejemplo 65. Uso de GO

```
USE pubs
GO
DECLARE @MyMsg VARCHAR(50)
SELECT @MyMsg = 'Hola mundo.'
```

GO -- Después del GO no podemos colocar ninguna instrucción en la misma línea, en teoría también
-- se puede poner comentarios, en la práctica no.
-- @MyMsg no está definido a partir de este GO. Por ello se producirá un error en la siguiente línea.

```
PRINT @MyMsg
GO
```

```
SELECT @@VERSION;
-- sp_who debe ser la primera línea de un lote.
sp_who – Función del sistema que devuelve información sobre los procesos y usuarios actuales de SQL
Server
GO
```

6.2.4 EXECUTE

Sirve para ejecutar un procedimiento almacenado. Obsérvese los procedimientos de sistema en el apartado 6.4.2.1.Procedimientos definidos en el sistema:

Ejemplo 66

```
USE master
EXECUTE xp_cmdshell 'dir *.exe'
EXECUTE xp_cmdshell 'mkdir prueba'
– Crea carpeta en c:\windows\system32 . Por defecto en SQL SERVER 2008 viene deshabilitado el uso
de este procedimiento por razones de seguridad. Para activarlo debemos ejecutar lo siguiente:
```

```
-- To allow advanced options to be changed.
EXEC sp_configure 'show advanced options', 1
GO
-- To update the currently configured value for advanced options.
RECONFIGURE
GO
-- To enable the feature.
EXEC sp_configure 'xp_cmdshell', 1
GO
-- To update the currently configured value for this feature.
RECONFIGURE
GO
```

Aunque probablemente sea más fácil en el configurador de superficie:

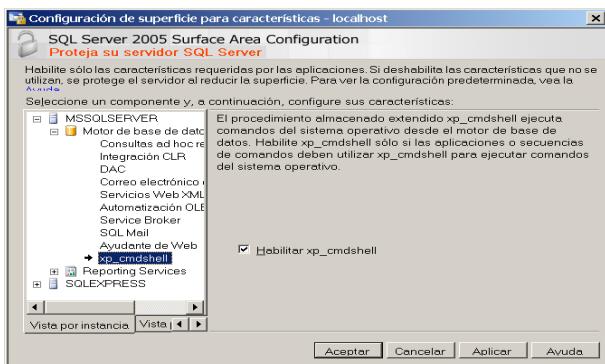


Imagen 54. Habilitar la ejecución de comandos de sistema

6.2.5 PRINT

Muestras un mensaje al usuario.

Sintaxis:

PRINT 'any ASCII text' | @local_variable | @@FUNCTION | string_expr

6.3 Estructuras de control

En la pág. 84 de la biblia de SQL Server 2008 viene la equivalencia entre las estructuras de control de PL/SQL y transact SQL.

6.3.1 IF...ELSE

Define una ejecución condicional y, opcionalmente, una ejecución alternativa si la condición es FALSE.

```
IF Boolean_expression
  { sql_statement | statement_block } – El bloque de sentencias irá entre BEGIN...END
[ ELSE
  { sql_statement | statement_block } ] – El bloque de sentencias irá entre BEGIN...END
```

6.3.2 CASE

Sintaxis:

```
CASE input_expression
  WHEN when_expression THEN result_expression
  [ ...n ]
  [
    ELSE else_result_expression
  ]
END
```

Ejemplo 67. Uso de CASE. Podemos utilizarlo para hacer una especie de resúmenes.

```

USE pubs
GO
SELECT Categoria = -- Igualo Categoria a lo que me devuelva el CASE
CASE type
    WHEN 'popular_comp' THEN 'Informática computacional'
    WHEN 'mod_cook' THEN 'Cocina moderna'
    WHEN 'business' THEN 'Negocios'
    WHEN 'psychology' THEN 'Psicología'
    WHEN 'trad_cook' THEN 'Cocina tradicional'
    ELSE 'Sin categoria'
END ,
CAST(title AS varchar(35)) AS 'Título ordenado',
price AS Precio
FROM titles
WHERE price IS NOT NULL
ORDER BY type, price
COMPUTE AVG(price) BY type
GO

```

	Categoría	Título ordenado	Precio
1	Negocios	You Can Combat Computer Stress!	2.9900
2	Negocios	Cooking with Computers: Surreptitious	11.9500
3	Negocios	The Busy Executive's Database Guide	19.9900
4	Negocios	Straight Talk About Computers	19.9900
	avg		
1		13.7300	

	Categoría	Título ordenado	Precio
1	Cocina moderna	The Gourmet Microwave	2.9900
2	Cocina moderna	Silicon Valley Gastronomic Treats	19.9900
	avg		
1		11.4900	

	Categoría	Título ordenado	Precio
1	Informática computacional	Secrets of Silicon Valley	20.0000
2	Informática computacional	But Is It User Friendly?	22.9500

El anterior ejemplo se puede sustituir por condiciones dobles:

```

USE pubs
GO
SELECT Categoria = -- Igualo Categoria a lo que me devuelva el CASE
CASE
    WHEN (type='popular_comp' or type='mod_cook') THEN 'Informática computacional'

```

Ejemplo 68. Uso de CASE2

```

USE pubs
GO
SELECT  'Categoria por precios' =
CASE
    WHEN price IS NULL THEN 'Sin fijar precio'
    WHEN price < 10 THEN 'Tituto fácil de adquirir'
    WHEN price >= 10 and price < 20 THEN 'Tituto de bolsillo'
    ELSE 'Libro caro!'
END,
CAST(title AS varchar(35)) AS 'Tituto libro'

```

FROM titles
 ORDER BY price
 GO

	Categoría por precios	Título libro
1	Sin fijar precio	The Psychology of Computer Cooking
2	Sin fijar precio	Net Etiquette
3	Titulo fácil de adquirir	The Gourmet Microwave
4	Titulo fácil de adquirir	You Can Combat Computer Stress!
5	Titulo fácil de adquirir	Life Without Fear
6	Titulo fácil de adquirir	Emotional Security: A New Algorithm
7	Titulo de bolsillo	Is Anger the Enemy?
8	Titulo de bolsillo	Cooking with Computers: Surreptitious
9	Titulo de bolsillo	Fifty Years in Buckingham Palace Kitchen
10	Titulo de bolsillo	Sushi, Anyone?

6.3.3 WHILE

Repite instrucciones mientras una condición específica sea TRUE. Si tenemos bucles anidados la cláusula BREAK sale del bucle más interno. Sintaxis:

WHILE Boolean_expression

{sql_statement | statement_block} – Los bloques de instrucciones irán agrupados entre BEGIN..END

[BREAK] – Finaliza la ejecución del bucle

{sql_statement | statement_block} – Los bloques de instrucciones irán agrupados entre BEGIN..END

[CONTINUE] – Reinicia la ejecución del bucle

Ejemplo 69. Utilización de WHILE e IF. Analizo si varios números son par o impar.

```
DECLARE @aux smallint
SELECT @aux = 1
WHILE (@aux<10)
    BEGIN
        -- IF (@aux-(@aux/2)*2)=1
        IF (@aux%2)=1
            BEGIN
                print(convert(varchar,@aux)+' es impar') – Convierto aux a varchar para que no produzca
error la concatenación
            END
        ELSE
            BEGIN
                print(convert(varchar,@aux)+' es par')
            END
        SET @aux=@aux+1
    END
```

Ejemplo 70. Utilización de BREAK y CONTINUE. Cuando el número de valores pares llega a 3 deja de analizar el resto de valores. Además el número 5 no lo analiza, se lo salta con CONTINUE.

```
DECLARE @aux smallint, @npares smallint
SELECT @aux = 1
SET @npares = 0
WHILE (@aux<10)
    BEGIN
        IF @aux=5
```

```

BEGIN
    SET @aux=@aux+1
    CONTINUE
END
IF (@aux-(@aux/2)*2)=1
BEGIN
    print(convert(varchar,@aux) +' es impar')
END
ELSE
BEGIN
    print(convert(varchar,@aux) +' es par')
    SET @npares = @npares + 1
    IF @npares=3
        BREAK
    END
    SET @aux=@aux+1
END

```

Ejercicio 20. Hacer el anterior ejemplo sin utilizar CONTINUE ni BREAK.

Ejercicio 21. Realizar un programa que muestre la tabla de códigos ASCII.

*Ejercicio 22. Hacer un programa que obtenga como resultado por pantalla lo siguiente:
a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, b1, b2, b3.....*

6.3.4 GOTO

GOTO label – Continúa el proceso en la instrucción que sigue a la etiqueta definida por label. No se debe utilizar puesto que no es programación estructurada.

Ejemplo 71. Utilización del GOTO

use pubs

```

goto cuerpo
muchos:
    print 'Existen muchos coches'
    goto fin
pocos:
    print 'Existen pocos coches'
    goto fin

cuerpo:
DECLARE @numeroMax smallint
-- Máximo número de autores por estado. Inicialmente da 15
SELECT @numeroMax= (SELECT max(valor) mAXIMO
                      FROM (SELECT count(*) valor
                            FROM authors
                            GROUP by state) SUBCONSULTA)

-- SET @numeroMax=10
IF @numeroMax>10
    goto muchos

```

```

else
    IF @numeroMax=10
        print 'Ni muchos ni pocos'
    ELSE
        goto pocos

fin:
    print 'Finalización del programa'

```

Ejercicio 23. Realizar el Ejemplo 71 con CASE y sin GOTOS.

Ejercicio 24. Realizar el Ejemplo 72. Sin GOTOS.

Ejemplo 72.

```

USE NorthWind
DECLARE @Error int
--Declaramos una variable que utilizaremos para almacenar un posible código de error

```

```

BEGIN TRAN
--Iniciamos la transacción
UPDATE Products SET UnitPrice=20 WHERE ProductName ='Chai'
--Ejecutamos la primera sentencia
SET @Error=@@ERROR
--Si ocurre un error almacenamos su código en @Error
--y saltamos al trozo de código que deshara la transacción. Si, eso de ahí es un
--GOTO, el demonio de los programadores, pero no pasa nada por usarlo
--cuando es necesario
IF (@Error<>0) GOTO TratarError

--Si la primera sentencia se ejecuta con éxito, pasamos a la segunda
UPDATE Products SET UnitPrice=20 WHERE ProductName='Chang'
SET @Error=@@ERROR
--Y si hay un error hacemos como antes
IF (@Error<>0) GOTO TratarError

--Si llegamos hasta aquí es que los dos UPDATE se han completado con
--éxito y podemos "guardar" la transacción en la base de datos
COMMIT TRAN

```

TratarError:

```

--Si ha ocurrido algún error llegamos hasta aquí
If @@Error<>0 THEN
    BEGIN
        PRINT 'Ha ocurrido un error. Abortamos la transacción'
        --Se lo comunicamos al usuario y deshacemos la transacción
        --todo volverá a estar como si nada hubiera ocurrido
        ROLLBACK TRAN
    END

```

6.4 Funciones y Procedimientos almacenados

Una de las novedades de SQL Server 2008 es que permite escribir funciones, procedimientos y triggers.

mientos y disparadores usando C# o Visual Basic .Net, aunque este no es tema a tratar en este curso, estos procedimientos se llaman extendidos.

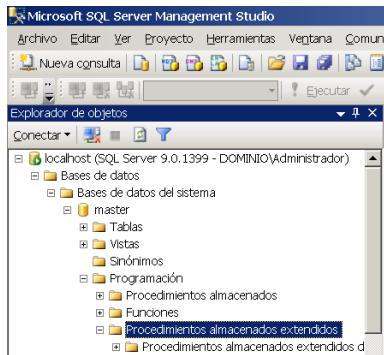


Imagen 55. Procedimientos almacenados extendidos

6.4.1 Función

Será un programa almacenado que devuelve siempre un valor (*funciones con valores escalares*), o incluso pudiera devolver una tabla (*funciones con valores de tabla*). Sintaxis:

```
CREATE FUNCTION [ owner_name. ] function_name
    ([ { @parameter_name [AS] scalar_parameter_data_type [= default] } [ ,...n ] ])
RETURNS scalar_return_data_type
[ WITH <function_option> [ , ] ...n ]
[ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
```

Ejemplo 73.

```
USE pubs
GO
CREATE FUNCTION SalesByStore (@storeid varchar(30))
RETURNS TABLE
AS
RETURN (SELECT title, qty
        FROM sales s, titles t
        WHERE s.stor_id = @storeid and
              t.title_id = s.title_id)
```

Ejecución

```
SELECT * FROM [pubs].[dbo].[SalesByStore] ('6380')
```

Si quisiéramos eliminar la función del sistema simplemente sería: `drop function SalesByStore.`

Ejemplo 74. La siguiente función devuelve el número de semana en el que está un día concreto del año (se le pasa por parámetro).

```

CREATE FUNCTION ISOweek (@DATE datetime)
RETURNS int
AS
BEGIN
    DECLARE @ISOweek int
    SET @ISOweek= DATEPART(wk,@DATE)+1
    -DATEPART(wk,CAST(DATEPART(yy,@DATE) as CHAR(4))+'0104')
--Special cases: Jan 1-3 may belong to the previous year
    IF (@ISOweek=0)
        SET @ISOweek=dbo.ISOweek(CAST(DATEPART(yy,@DATE)-1
        AS CHAR(4))+'12'+ CAST(24+DATEPART(DAY,@DATE) AS CHAR(2)))+1
--Special case: Dec 29-31 may belong to the next year
    IF ((DATEPART(mm,@DATE)=12) AND
        ((DATEPART(dd,@DATE)-DATEPART(dw,@DATE))>= 28))
        SET @ISOweek=1
    RETURN(@ISOweek)
END

```

Ejecutarlo:

```
SELECT [pubs].[dbo].[ISOweek]('06/03/2005')
```

Ejercicio 25. Realizar una función llamada calcular que reciba tres argumentos. En el primero, irá un texto que puede tomar los valores superficie o perímetro, en el segundo y tercero irá un número que dará la anchura y la altura de un rectángulo. Si nos introducen en el argumento 1 la palabra superficie la función devolverá la superficie del rectángulo, si introducen perímetro calculará el perímetro.

Ejercicio 26. McGrawHill-CASE. Escribir una función que reciba dos fechas y devuelva los trienios que han pasado entre dichos valores.

Ejercicio 27. McGrawHill-CASE. Escribe una función que devuelva solamente caracteres alfabéticos, sustituyendo cualquier otro carácter por blancos a partir de una cadena que se pasará en la llamada

Ejercicio 28. Ejercicio con cursores. Sobre la BD de Averías, analizar si todo los códigos de centro y de código responsable de daño de la BD base_averias, están en las tablas centro y responsable_daño. La función deberá devolver ‘están todos’ o ‘no están todos’.

6.4.1.1 Funciones definidas en el sistema:

Véase 5.3.2, 5.3.3 y 5.3.4 donde se habla del funcionamiento de las funciones más importantes del sistema.

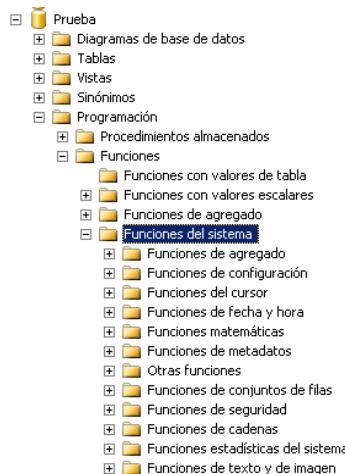


Imagen 56. Funciones existentes en el sistema (vista desde el analizador de consultas)

6.4.2 Procedimiento

La diferencia con la función es que no devuelve ningún valor. Sintaxis:

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
[ { @parameter data_type }
  [ VARYING ] [ = default ] [ OUTPUT ]
  ] [ ,...n ]
[ WITH
  { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]
[ FOR REPLICATION ]
AS sql_statement [ ...n ]
```

Ejemplo 75.

```
USE pubs
IF EXISTS (SELECT name FROM sysobjects
            WHERE name = 'au_info' AND type = 'P')
    DROP PROCEDURE au_info
GO
USE pubs
GO
CREATE PROCEDURE au_info
    @lastname varchar(40),
    @firstname varchar(20)
AS
SELECT au_lname, au_fname, title, pub_name
    FROM authors a INNER JOIN titleauthor ta
        ON a.au_id = ta.au_id INNER JOIN titles t
            ON t.title_id = ta.title_id INNER JOIN publishers p
                ON t.pub_id = p.pub_id
WHERE au_fname = @firstname
```

```
AND au_lname = @lastname
GO
```

Ejecución:

```
DECLARE @RC int
DECLARE @lastname varchar(40)
DECLARE @firstname varchar(20)
SET @lastname = 'White'
SET @firstname = 'Johnson'
-- Establecer valores del parámetro
EXEC @RC = [pubs].[dbo].[au_info] @lastname, @firstname
```

El procedimiento almacenado **au_info** se puede ejecutar de estas formas:

```
EXECUTE au_info 'Dull', 'Ann'
-- Or
EXECUTE au_info @lastname = 'Dull', @firstname = 'Ann'
-- Or
EXECUTE au_info @firstname = 'Ann', @lastname = 'Dull'
-- Or
EXEC au_info 'Dull', 'Ann'
-- Or
EXEC au_info @lastname = 'Dull', @firstname = 'Ann'
-- Or
EXEC au_info @firstname = 'Ann', @lastname = 'Dull'
```

O si este procedimiento es la primera instrucción del proceso por lotes:

```
au_info 'Dull', 'Ann'
-- Or
au_info @lastname = 'Dull', @firstname = 'Ann'
-- Or
au_info @firstname = 'Ann', @lastname = 'Dull'
```

Para poder depurar el procedimiento desde el analizador de consultas:

[Pérez, 2003c].

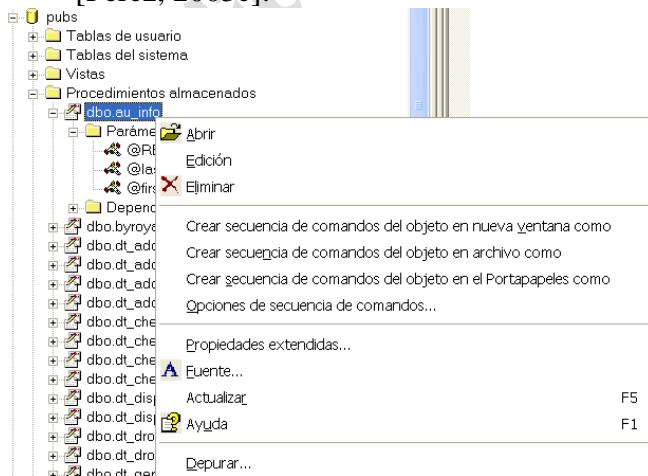
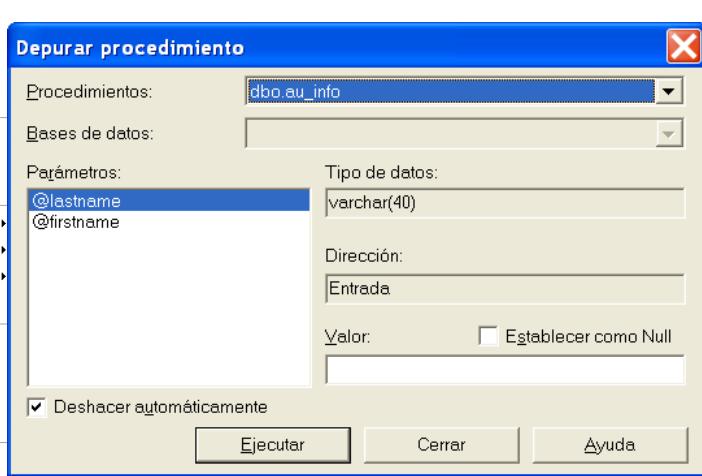
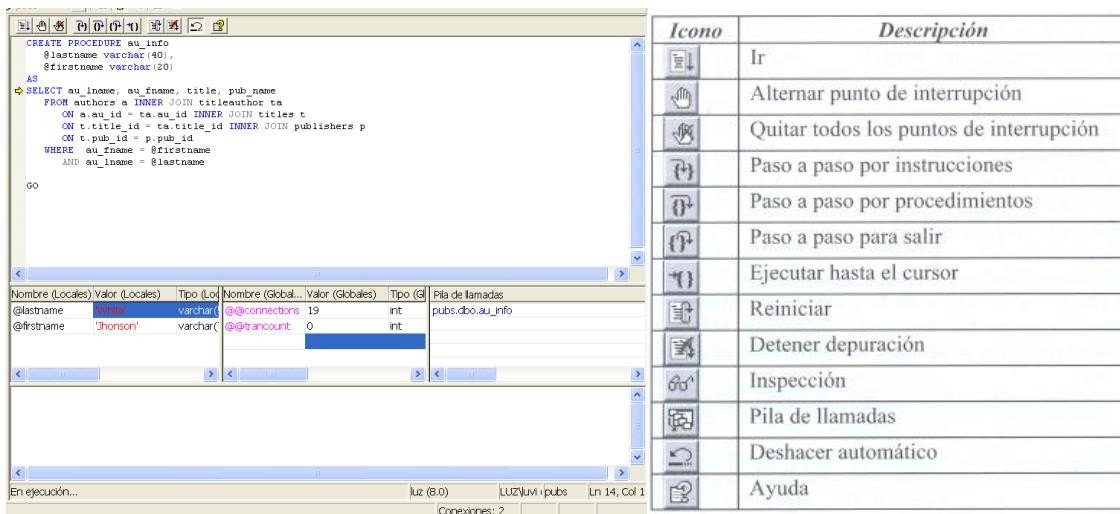



Imagen 57. Depuración del procedimiento**Imagen 58.** Depurador y la barra de herramientas (SQL Server 2000)

Ejemplo 76. Creación de un procedimiento que permite dividir el resultado de una consulta en páginas y de esta manera recuperar solamente una determinada página. BD: AdventureWorks

```
CREATE PROCEDURE Paginacion
@TamañoPag int,
@NúmeroPag int
AS
SELECT *
FROM (
SELECT *, ROW_NUMBER() OVER (ORDER BY EmployeeID, loginid) AS NúmeroFila
FROM HumanResources.Employee
) AS Consulta
WHERE NúmeroFila BETWEEN @TamañoPag * @NúmeroPag + 1
AND @TamañoPag * (@NúmeroPag + 1)
```

Ejercicio 29. McGrawHill - CASE. Escribe un procedimiento con que recibe un número de empleado y un número de departamento y asignará al empleado (tabla emple) el departamento indicado en segundo parámetro.

Ejercicio 30. Repetir el ejercicio anterior teniendo en cuenta que si el departamento es el de contabilidad no se confirmarán los cambios (forzar la realización de las transacciones aunque no sea necesario).

Ejercicio 31. Escribe un procedimiento que reciba una cadena y la visualice al revés (usar substring()).

```
select reverse('casa')
```

6.4.2.1 Procedimientos definidos en el sistema:

En la base de datos master con el prefijo sp_ (stored procedure) se almacenan diferentes procedimientos creados por SQL Server 2008. Muchas de las actividades administrativas en SQL Server 2008 se realizan mediante un tipo especial de procedimiento conocido como procedimiento almacenado del sistema. Por ejemplo, **sys.sp_changedbowner** es un procedimiento almacenado del sistema.

- **xp_fileexist.**

```
EXECUTE xp_fileexist <nombre de archivo> [, <file_exists INT> OUTPUT]
```

Lo podremos utilizar para saber si existe un fichero o no. (*master - Programación- Procedimientos almacenados extendidos – Procedimientos almacenados extendidos del sistema*)

Ejemplo 77.

```
SET NOCOUNT ON
DECLARE @FileName varchar(255)
DECLARE @File_Exists int
SELECT @FileName='C:\boot.ini'
EXEC Master.dbo.xp_fileexist @FileName, @File_Exists OUT
IF @File_Exists = 1
    PRINT 'El fichero existe'
ELSE
    PRINT 'El fichero no existe'
GO
```

La ejecución del procedimiento sin el parámetro segundo es posible y genera una salida como la siguiente:

EXEC Master.dbo.xp_fileexist 'C:\prueba.txt'								
	Resultados	Mensajes						
	<table border="1"> <thead> <tr> <th>El archivo existe</th> <th>El archivo es un directorio</th> <th>El directorio primario existe</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	El archivo existe	El archivo es un directorio	El directorio primario existe	1	0	0	1
El archivo existe	El archivo es un directorio	El directorio primario existe						
1	0	0						

El valor 1 es verdadero y el valor 0 es falso. Como vemos podemos también saber si es un directorio o no.

[Dalton et al, 2001d]. Procedimientos almacenados extendidos.

- **sp_databases.** Lista cualquier base de datos disponible a través de su conexión a SQL Server.
 - **sp_monitor.** Muestra 13 variables de diagnóstico de servidor utilizadas para seguir estadísticas en la sesión actual de SQL Server.
 - **sp_spaceused.** Cuenta y muestra la utilización actual de espacio por parte una tabla o de una base de datos completa en SQL Server, concretamente sobre la que tenemos conexión abierta.
- ```
exec sp_spaceused
exec sp_spaceused authors
```
- **sp\_tables.** Devuelve una lista de las tablas que se pueden consultar en la cláusula FROM de una instrucción SQL
  - **sp\_who.** Informa sobre todos los usuarios actuales de la base de datos y sus procesos para incluir procesos de bloque e información I/O del usuario.
  - **sp\_who2.** De la misma forma que sp\_who pero con un formato algo diferente, sp\_who2 de-

vuelve información adicional sobre los usuarios actuales en su servidor. De vez en cuando, ejecutar `sp_who2` en momentos de mucho procesamiento en el servidor podría causar el bloqueo de otros procesos del servidor. Si no obtiene resultados rápidamente de `sp_who2`, cancele la consulta e inténtelo con `sp_who`.

- **sp\_addlogin.** Agrega un usuario a la lista de usuarios que pueden iniciarse en un servidor y obtener conexión. Este procedimiento no garantiza los permisos de usuario a base de datos u objetos de datos.

- **sp\_adduser.** Agrega un usuario a la base de datos y permite el acceso a los datos. Este procedimiento no permite que el usuario se conecte al servidor. Consulte `sp_addlogin` para garantizar el acceso al servidor.

- **sp\_password.** Permite la administración de contraseñas de acceso a SQL Server.

- **sp\_stored\_procedures.** Muestra una lista de todos los procedimientos almacenados en ese momento en la base de datos con toda la información sobre el propietario y nombres de los procedimientos.

- **sp\_addextendedproc.** Agrega el nombre y la dynamic link library de un procedimiento almacenado de las tablas de SQL Server sysobjects y syscomments de la base de datos master. Una vez registrado en el servidor SQL Server, puede llamar a un procedimiento extendido desde el código SQL y utilizarlo para ampliar la funcionalidad de su servidor.

- **sp\_addextendedproperty.** Agrega una propiedad extendida a un objeto de bases de datos para que los elementos adicionales, como la descripción o información adicional, se pueda asociar directamente con los objetos de SQL Server. Este procedimiento es especialmente útil para aplicaciones de Internet.

- **sp\_addmessage.** Puede utilizarse para agregar a su servidor mensajes definidos por el usuario.

- **sp\_attach\_db.** Adjunta los archivos de bases de datos que albergan la base de datos y los datos para el servidor.

- **sp\_attach\_single\_file\_db.** Adjunta una base de datos a un servidor que solo tiene disponible el archivo de datos (.mdf). Se reconstruye automáticamente el archivo de registro para una base de datos adjuntada con `sp_attach_single_file_db`.

- **sp\_changedbowner.** Cambia el propietario de una base de datos en el caso de que no se necesite más un usuario con esa capacidad.

- **sp\_changeobjectowner.** Reasigna la propiedad de un objeto de bases de datos a un nuevo propietario.

- **sp\_columns.** Muestra las columnas de un único objeto, sea una tabla o una vista, que se puede consultar en el entorno actual. Este procedimiento es útil a la hora de determinar qué columnas están disponibles para la construcción de una consulta personalizada o una herramienta de consulta definible por un usuario.

- **sp\_configure.** Muestra o cambia las opciones actuales de SQL. Algunas opciones son dinámicas y se pueden cambiar sin detener e iniciar el servicio de SQL Server, mientras que otras son estáticas y se pueden modificar solo reiniciando el servicio de SQL Server.

- **sp\_cycle\_errorlog.** Cierra y mueve el registro de error actual un nivel hacia abajo. El nuevo registro no tendrá la información de inicio del servidor. Llamar a este procedimiento o DBCC ERRORLOG coloca una entrada en el registro actual y en el nuevo registro de error, confirmando que se cierra el registro de error.

- **sp\_dbcmptlevel.** Lista o cambia el nivel de compatibilidad de base de datos para una base de datos. Cambiar la compatibilidad de una base de datos altera las características del rendimiento de las instrucciones SQL ejecutadas en una base de datos. Cambie el nivel de compatibilidad de la base de datos tras haber probado todos los procedimientos, desencadenadores y código cliente

que se ejecuta contra la base de datos.

- **sp\_dboption.** Muestra o cambia las opciones de una base de datos en SQL Server.
  - **sp\_depends.** Devuelve una lista con todos los objetos dependientes del objeto que se pasa en el procedimiento.
  - **sp\_detach\_db.** Separa la base de datos del servidor, elimina los datos y archivos de registro desde el servidor y permite que se copien o eliminan desde el sistema operativo. Si existe alguna conexión activa de usuario en la base de datos cuando se utiliza el comando sp\_detach\_db, devuelve un error.
  - **sp\_dropextendedproc.** Elimina la referencia a un procedimiento almacenado desde el servidor.
  - **sp\_getapplock.** Permite que una aplicación cliente instituya un semáforo de bloqueo a nivel de aplicación. Muchas aplicaciones de terceros se han desarrollado para proporcionar una funcionalidad ya antes de SQL Server 2000. La utilización de sp\_getapplock agrega un nivel de bloqueo definido por el usuario para aplicaciones cliente que son administradas por el administrador de bloqueos de SQL Server. Este procedimiento no afecta de ninguna manera los bloqueos de objetos en el servidor.
  - **sp\_help.** Informa sobre cualquier objeto de la base de datos de la tabla sysobjects. Hay una serie de procedimientos de ayuda recogidos en los libros en pantalla. Debería familiarizarse con ellos antes de realizar un examen de certificación.
- ```
exec sp_help sp_databases
exec sp_help authors
```
- **sp_lock.** Proporciona información sobre bloqueos en el servidor. Los números ID de procesos individuales se pueden pasar a este procedimiento para obtener informes mas detallados de bloqueo.
 - **sp_oacreate.** Permite que los procedimientos almacenados creen una instancia de los objetos de automatización OLE mediante instrucciones SQL para extender la funcionalidad de SQL Server. Hay una serie de procedimientos almacenados de tipo OLE disponibles en SQL Server que permiten el acceso requerido y métodos de distribución. En los libros en pantalla de SQL Server podrá obtener más información sobre los procedimientos almacenados OLE.
 - **sp_recompile.** Indica un objeto (tabla, procedimiento almacenado o desencadenador) de forma que la siguiente vez que se requiera la ejecución del procedimiento o del desencadenador, se recompilen la consulta y el plan de consultas. Este procedimiento es útil cuando se agregan nuevos índices o cambia la distribución de datos.
 - **sp_releaseapplock.** Llamar a este procedimiento libera el bloqueo instalado con sp_getapplock, permitiendo que otras conexiones adquieran un bloqueo de nivel de aplicación. Este procedimiento no afecta para nada a los bloqueos de objetos del servidor.
 - **sp_rename.** Cambia el nombre de un objeto definido por un usuario en una base de datos.
 - **sp_resetstatus.** Reconfigura un estado de base de datos sospechoso en SQL Server, permitiendo que el usuario acceda a la base de datos (cualquier daño que hubiera causado y que pudiese considerarse sospechoso no lo repara este procedimiento).
 - **sp_runwebtask.** Desencadena la ejecución de un trabajo Web previamente definido que produce un documento HTML.
 - **sp_start_job.** Inicia un trabajo SQL Server previamente definido.
 - **sp_statistics.** Devuelve una lista de todos los índices y estadísticas para una tabla en concreto de una base de datos.
 - **sp_MStablespace.** Tener que saber la cantidad de almacenamiento que tiene una tabla es algo que sucede todos los días en una empresa. La mayoría de las tablas acaban siendo puntos conflic-

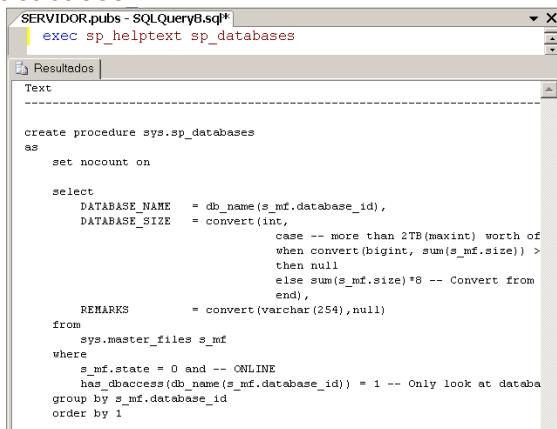
tivos y complejos para el desarrollo, y los requisitos de espacio suelen ser motivo de muchas discusiones. Microsoft ha proporcionado otro procedimiento no documentado para la tarea de determinar el espacio que una tabla está utilizando. El procedimiento sp_MStablespace devuelve el espacio el número de filas, los datos y el espacio de índices utilizado para la tabla y los objetos de índice.

- **sp_executesql**. Nos permitirá ejecutar una consulta dinámica. Es decir, aquella que se define complementariamente en tiempo de ejecución. Recibirá dos parámetros, el primero marca el tipo de dato que recibe como variable y el segundo el valor.

```
EXEC sp_executesql
N'Select * From authors
Where au_lname = @Valor and au_fname= @Valor2',
N'@Valor varchar(40),@Valor2 varchar(20)',
@Valor = 'White', @Valor2='Johnson';
```

- **sp_helptext**. Nos informará del código que contiene un procedimiento que reciba por parámetro.

```
exec sp_helptext sp_databases
```



The screenshot shows a SQL Server Management Studio window titled 'SERVIDOR.pubs - SQLQuery8.sql'. The query entered is 'exec sp_helptext sp_databases'. The results pane displays the T-SQL code for the sys.sp_databases stored procedure. The code creates a procedure that selects database information from the sys.master_files system catalog view, including database name, size, and remarks.

```
create procedure sys.sp_databases
as
    set nocount on

    select
        DATABASE_NAME      = db_name(s_mf.database_id),
        DATABASE_SIZE     = convert(int,
                                    case -- more than 2TB(maxint) worth of
                                    when convert(bigint, sum(s_mf.size)) >
                                    then null
                                    else sum(s_mf.size)*8 -- Convert from
                                    end),
        REMARKS           = convert(varchar(254),null)
    from
        sys.master_files s_mf
    where
        s_mf.state = 0 and -- ONLINE
        has_dbaccess(db_name(s_mf.database_id)) = 1 -- Only look at database
    group by s_mf.database_id
    order by 1
```

6.5 Cursos de Transact-SQL

[Charte, 2004b]

Estos cursos se basan en la sintaxis DECLARE CURSOR y se utilizan principalmente en secuencias de comandos, procedimientos almacenados y desencadenadores de Transact-SQL. Los cursos Transact-SQL se implementan en el servidor y se administran mediante instrucciones Transact-SQL enviadas del cliente al servidor. También se pueden encontrar en lotes, procedimientos almacenados o desencadenadores.

Sintaxis:

```
DECLARE cursor_name [ INSENSITIVE ] [ LOCAL | GLOBAL ] [ FORWARD_ONLY | SCROLL ]
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ] CURSOR
FOR select_statement
[ FOR { READ ONLY | UPDATE [ OF column_name [ ,...n ] ] } ]
```

Con **INSENSITIVE** las modificaciones que podemos realizar sobre el cursor no se realizan sobre las tablas originales sobre las que se definió, se realizan sobre una tabla temporal nueva que se crea: tempdb.

Con **SCROLL** especifica que están disponibles todas las opciones de recuperación (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE). Si no se especifica SCROLL en una instrucción DECLARE CURSOR, la única opción de recuperación que se admite es NEXT. Dependiendo cómo se haya definido el cursor, las operaciones de recuperación de filas se limitarán a un desplazamiento secuencial, desde la primera hasta la última, o será posible un desplazamiento libre a cualquier fila del conjunto de datos. Para lograr esto último hay que poner delante de la palabra SCROLL delante de CURSOR.

Ejemplo 78.

```
DECLARE CursorTitulos CURSOR FOR
    SELECT titulo, precio
        FROM libros
```

Hasta ahora simplemente hemos declarado un cursor. Para que podamos utilizarlo debemos abrirlo (OPEN). Además para recorrer cada registro del cursor deberemos avanzar lo que podríamos llamar como puntero de registro (FETCH). Al finalizar la utilización del mismo lo cerraremos (CLOSE), es importante porque si no el nombre del cursor seguirá existiendo y podrá provocar errores. Con (DEALLOCATE) quitamos la referencia a un cursor. Cuando se ha quitado la última referencia al cursor, Microsoft SQL Server libera las estructuras de datos que componen el cursor.

Sintaxis del FETCH, (utilizado para ir avanzando por los diferentes registros):

```
FETCH fila FROM cursor
    INTO variables
```

Donde fila podrá valer: FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE.

6.5.1 Funciones FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE

Next.- Siguiente a la actual o primera si el cursor está recién abierto. Único que no necesita utilizar SCROLL en la definición del cursor.

Prior.- Anterior al actual.

First.- Primera

Last.- Última

Absolute.- Irá seguido de un número indicando exactamente la fila exacta a recuperar.

Relative.- Irá seguido de un número, positivo o negativo, indicando el número de filas que hay que moverse adelante o atrás.

Variable de sistema utilizada en cursosres: @@FETCH_STATUS, tomará el valor de 0 si la operación sobre el cursor se ejecutó correctamente, -1 si ha finalizado con error o -2 si falta la fila recuperada. También @@CURSOR_ROWS devuelve el número de filas del último cursor



abierto, para usar esta última el cursor debe ser de tipo SCROLL.

Ejemplo 79.

```
DECLARE Employee_Cursor CURSOR FOR
SELECT LoginId, Title FROM AdventureWorks.HumanResources.Employee
OPEN Employee_Cursor
FETCH NEXT FROM Employee_Cursor
print '@@CURSOR_ROWS1:' + convert(varchar(5),@@CURSOR_ROWS)
-- Devuelve -1, es decir, no se puede saber el número exacto de filas
-- del cursor ya que cambia constantemente
WHILE @@FETCH_STATUS = 0
BEGIN
    FETCH NEXT FROM Employee_Cursor
END
CLOSE Employee_Cursor
DEALLOCATE Employee_Cursor
```

Lo mismo pero en este caso insertándolo en una tabla temporal:

```
USE [AdventureWorks]
GO
/****** Objeto: Table [dbo].[datos]      Fecha de la secuencia de comandos:
09/15/2008 09:58:44 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[datos](
    [a] [nvarchar](50) NULL,
    [b] [nvarchar](50) NULL
) ON [PRIMARY]

declare @var1 nvarchar(50),@var2 nvarchar(50)
DECLARE Employee_Cursor CURSOR FOR
SELECT LoginId, Title FROM AdventureWorks.HumanResources.Employee
OPEN Employee_Cursor
FETCH NEXT FROM Employee_Cursor INTO @var1,@var2
insert into AdventureWorks.dbo.datos values (@var1,@var2)
print '@@CURSOR_ROWS1:' + convert(varchar(5),@@CURSOR_ROWS)
-- Devuelve -1, es decir, no se puede saber el número exacto de filas
-- del cursor ya que cambia constantemente
WHILE @@FETCH_STATUS = 0
BEGIN
    FETCH NEXT FROM Employee_Cursor INTO @var1,@var2
    insert into AdventureWorks.dbo.datos values (@var1,@var2)
END
select * from AdventureWorks.dbo.datos
CLOSE Employee_Cursor
DEALLOCATE Employee_Cursor
```

Ejemplo 80. Con este ejemplo recuperaremos el número de registros existente en todas las tablas de la base de datos pubs.



```

USE pubs
DECLARE CursorNombresTablas CURSOR scroll
FOR
    SELECT TABLE_NAME NombTabla
    FROM INFORMATION_SCHEMA.TABLES -- Tabla que almacena los nombres de las tablas de la BD
actual
OPEN CursorNombresTablas
DECLARE @nombretabla varchar(20)
--SET @tablename = 'authors'
FETCH NEXT FROM CursorNombresTablas INTO @nombretabla
WHILE (@@FETCH_STATUS <> -1)
BEGIN
print '@@FETCH_STATUS:' + convert(varchar(5),@@FETCH_STATUS)
print '@@CURSOR_ROWS:' + convert(varchar(5),@@CURSOR_ROWS)
IF (@@FETCH_STATUS <> -2)
BEGIN
    SELECT @nombretabla = RTRIM(@nombretabla) --Borro caracteres a la derecha
    --Muestro el número de registros de cada una de las tablas
    EXEC ('SELECT "' + @nombretabla + '" = count(*) FROM ' + @nombretabla )
--    PRINT ''
END
    FETCH NEXT FROM CursorNombresTablas INTO @nombretabla
END
CLOSE CursorNombresTablas
DEALLOCATE CursorNombresTablas

```

El mismo ejemplo pero esta vez para calcular el número completo de registros de toda la BD:

```

USE [pubs]
GO
***** Objeto: Table [dbo].[registros]      Fecha de la secuencia de
comandos: 09/15/2008 10:14:28 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[registros](
    [numeroregistros] [numeric] (18, 0) NULL
) ON [PRIMARY]

USE pubs
set nocount on
declare @totales numeric,@parcial numeric
set @parcial = 0
set @totales=0
DECLARE CursorNombresTablas CURSOR
FOR
    SELECT TABLE_NAME NombTabla
    FROM INFORMATION_SCHEMA.TABLES -- Tabla que almacena los nombres de las
tablas de la BD actual
OPEN CursorNombresTablas
DECLARE @nombretabla varchar(20)
FETCH NEXT FROM CursorNombresTablas INTO @nombretabla
WHILE (@@FETCH_STATUS <> -1)
BEGIN
    SELECT @nombretabla = RTRIM(@nombretabla) --Borro caracteres a la derecha

```



```

EXEC ('insert into registros SELECT count(*) FROM '+ @nombretabla)
EXEC ('SELECT ''' + @nombretabla + ''' = count(*) FROM '+ @nombretabla )
FETCH NEXT FROM CursorNombresTablas INTO @nombretabla
END

CLOSE CursorNombresTablas
DEALLOCATE CursorNombresTablas

declare CRegistros cursor
for
    SELECT * from registros
open CRegistros
FETCH NEXT FROM CRegistros INTO @parcial
WHILE (@@FETCH_STATUS = 0)
BEGIN
    set @totales=@totales+@parcial
    FETCH NEXT FROM CRegistros INTO @parcial
END
print 'Número total de registros:' +convert(varchar(10),@totales)
close CRegistros
DEALLOCATE CRegistros
-- Fin del proceso de cuenta de registros
select * from registros

declare @valor numeric
select @valor=count(*) from emple
select @valor as DATO
insert into registros SELECT count(*) FROM emple

delete registros

```

Véase otro ejemplo Ejemplo 14. De la pág 77, sobre cursos que en este caso usan el tipo table (tipo especial para campos que almacenarán tablas).

Ejercicio 32. McGraw-Hill. CASE. Escribe un procedimiento que reciba una cadena, que representa un apellido, y visualice el apellido y el número de empleados de todos los empleados (tabla emple) cuyo apellido contega la cadena especificada. Al finalizar, visualiza el número de empleados mostrados. El procedimiento utilizará un cursor para recuperar los empleados.

6.5.2 Tipos de cursos

- ✓ Cursos estáticos
- ✓ Cursos dinámicos
- ✓ Cursos de sólo avance
- ✓ Cursos controlados por conjunto de claves

Los cursos estáticos detectan pocos cambios o ningún cambio, pero consumen relativamente pocos recursos al desplazarse. Los cursos dinámicos detectan todos los cambios pero consumen más recursos al desplazarse. Los cursos controlados por conjunto de claves se encuentran entre los dos anteriores, pero con un consumo menor que los cursos dinámicos.

Aunque los modelos de cursor de la API de base de datos consideran el cursor de sólo avance como un tipo más, SQL Server no establece esta distinción. SQL Server considera que las opciones de desplazamiento de sólo avance y de desplazamiento son las que se pueden aplicar a los cursos estáticos, a los controlados por conjunto de claves y a los dinámicos.

[devjoker, 2008]

✓ [**LOCAL | GLOBAL**]

Local. Especifica que el ámbito del cursor es local para el proceso por lotes, procedimiento almacenado o desencadenador en que se creó el cursor. Global. Especifica que el ámbito del cursor es global para la conexión. Puede hacerse referencia al nombre del cursor en cualquier procedimiento almacenado o proceso por lotes que se ejecute en la conexión. Por defecto los cursos serán globales.

```
DECLARE CEmpleados CURSOR LOCAL|GLOBAL FOR
SELECT emp_no FROM emple
```

```
open CEmpleados
declare @variable numeric(4,0)
fetch CEmpleados into @variable
select @variable
close CEmpleados
deallocate CEmpleados -- Se libera el recurso global
```

✓ [**FORWARD_ONLY | SCROLL**]

FORWARD_ONLY. Especifica que el cursor sólo se puede desplazar de la primera a la última fila. **FETCH NEXT** es la única opción de recuperación admitida. Esta es la opción por defecto. Si se especifica **SCROLL** podremos utilizar **fetch [first | last | etc.]**

✓ [**STATIC | KEYSET | DYNAMIC | FAST_FORWARD**]

STATIC. Define un cursor que hace una copia temporal de los datos que va a utilizar una vez se utilice el comando **OPEN**, no cuando se define el cursor. Todas las solicitudes que se realizan al cursor se responden desde esta tabla temporal de tempdb; por tanto, las modificaciones realizadas en las tablas base no se reflejan en los datos devueltos por las operaciones de recuperación realizadas en el cursor y además este cursor no admite modificaciones.

```
DECLARE CEmpleados CURSOR STATIC FOR
SELECT emp_no, apellido FROM emple
```

```
open CEmpleados
declare @variable1 numeric(4,0),@variable2 nchar(10)
fetch CEmpleados into @variable1,@variable2
select @variable1,@variable2
update emple set apellido='Vergel'
where emp_no=@variable1
fetch first from CEmpleados into @variable1,@variable2
```

```

select @variable1,@variable2
close CEmpleados
deallocate CEmpleados -- Se libera el recurso global

select * from emple

```

Si cambiamos STATIC por SCROLL podremos ver que los datos devueltos son diferentes puestos que la primera vez que recupero datos del cursos tira de la tabla y la segunda, después de hacer el update, también.

KEYSET. Almacena en tempdb sólo las claves primarias.

DYNAMIC. Define un cursor que, al desplazarse por él, refleja en su conjunto de resultados todos los cambios realizados en los datos de las filas. Los valores de los datos, el orden y la pertenencia de las filas pueden cambiar en cada operación de recuperación.

FAST_FORWARD. Especifica un cursor FORWARD_ONLY, READ_ONLY con las optimizaciones de rendimiento habilitadas. No se puede especificar FAST_FORWARD si se especifica también SCROLL o FOR_UPDATE.

[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]

READ_ONLY. Evita que se efectúen actualizaciones a través de este cursor. No es posible hacer referencia al cursor en una cláusula WHERE CURRENT OF de una instrucción UPDATE o DELETE. Esta opción reemplaza la capacidad de actualizar el cursor.

SCROLL_LOCKS. Especifica que se garantiza que las actualizaciones o eliminaciones posicionadas realizadas a través del cursor serán correctas. Microsoft SQL Server bloquea las filas cuando se leen en el cursor para garantizar que estarán disponibles para futuras modificaciones. No es posible especificar SCROLL_LOCKS si se especifica también FAST_FORWARD o STATIC.

OPTIMISTIC. Especifica que las actualizaciones o eliminaciones posicionadas realizadas a través del cursor no se realizarán correctamente si la fila se ha actualizado después de ser leída en el cursor. SQL Server no bloquea las filas al leerlas en el cursor. En su lugar, utiliza comparaciones de valores de columna timestamp o un valor de suma de comprobación si la tabla no tiene columnas timestamp, para determinar si la fila se ha modificado después de leerla en el cursor. Si la fila se ha modificado, el intento de actualización o eliminación posicionada genera un error. No es posible especificar OPTIMISTIC si se especifica también FAST_FORWARD.

Para **actualizar un cursor** debemos especificar FOR UPDATE después de la sentencia SELECT en la declaración del cursor, y WHERE CURRENT OF <nombre_cursor> en la sentencia UPDATE tal y como muestra el siguiente ejemplo.

```

-- Declaracion de variables para el cursor
select * into #temporal from clientes

```

```

DECLARE @Id int,
@Nombre varchar(255),
@Localidad varchar(255),
@NifCif varchar(20)

-- Declaración del cursor
DECLARE cClientes CURSOR FOR
SELECT Nombre, Localidad, Nif
FROM #temporal
FOR UPDATE

-- Apertura del cursor
OPEN cClientes
-- Lectura de la primera fila del cursor

FETCH cClientes INTO @Nombre, @localidad, @NifCif
WHILE (@@FETCH_STATUS = 0)
BEGIN
    UPDATE #temporal
    SET nombre = '*'
    WHERE CURRENT OF cClientes

    -- Lectura de la siguiente fila del cursor
    FETCH cClientes INTO @Nombre, @localidad, @NifCif
END

-- Cierre del cursor
CLOSE cClientes

-- Liberar los recursos
DEALLOCATE cClientes

```

Ejercicio 33. Calcular la media del salario de los empleados (tabla emple) de la base de datos del Oracle-Case, sin utilizar AVG, es decir, con un cursor recorreremos los valores los sumaremos en una variable auxiliar y lo dividiremos por el número de elementos (tampoco podremos usar COUNT)

Ejercicio 34. Cambiar el ejercicio Ejercicio 33 utilizando para recorrer el cursor de tipo SCROLL la función FETCH absolute.

Ejercicio 35. McGraw-Hill. CASE. Codifica un programa que visualice los dos empleados que ganan menos de cada oficio (tabla emple), utiliza cursores de sólo avance.

Ejercicio 36. Modifica el sueldo del los empleados a la media de los sueldos más un 15% de la misma o su sueldo actual, según cual sea el valor más beneficioso para el empleado. Utiliza cursores para modificarlo. No realizar los cambios definitivamente (utilizar un rollback).

6.6 Try/catch

Los errores en el código de Transact-SQL se pueden procesar mediante una construcción TRY...CATCH similar a las características de control de excepciones de otros lenguajes. Cuan-

do se detecta una condición de error en una instrucción de Transact-SQL contenida en un bloque TRY, el control pasa al bloque CATCH en donde puede procesarse. Un bloque TRY se inicia con la instrucción BEGIN TRY y finaliza con la instrucción END TRY. En Transact-SQL, cada bloque TRY se asocia a un sólo bloque CATCH.

Funciones utilizadas en bloques try/catch:

- ✓ **ERROR_LINE:** Devuelva la línea en la que se produjo el error. Habrá que analizar el flujo ejecución del código para ver que es exactamente esa línea, porque no tiene por qué serlo exactamente.
- ✓ **ERROR_MESSAGE:** Mostrará un breve mensaje indicando la naturaleza del error.
- ✓ **ERROR_NUMBER:** Devuelve el número de error. Si el error se produce en un procedimiento o un disparador devolverá el número de línea de la rutina.
- ✓ **ERROR_PROCEDURE:** Devolverá el nombre del procedimiento o disparador en el que se ha producido un error, en cualquier otro caso devolverá nulo.
- ✓ **ERROR_SEVERITY:** Muestra la gravedad del error.
- ✓ **ERROR_STATE:** Da información complementaria al ERROR_NUMBER.

Una manera de personalizar los posibles errores que se produzcan en el código Transact SQL ejecutado es con RAISERROR. Los errores generados por RAISERROR funcionan igual que los generados por el código del Database Engine (Motor de base de datos). Las funciones del sistema ERROR_LINE, ERROR_MESSAGE, ERROR_NUMBER, ERROR_PROCEDURE, ERROR_SEVERITY, ERROR_STATE y @@ERROR informan de los valores especificados por RAISERROR. Cuando se ejecuta RAISERROR con un nivel de gravedad 11 o superior en un bloque TRY, transfiere el control al bloque CATCH asociado. Véase pág. 123 sobre el procedimiento RAISERROR.

Ejemplo 81. Crearemos un procedimiento que recibirá el login y el identificador del usuario con el fin de validarlos en una tabla HumanResources.Employee de la BD AdventureWorks.

```
ALTER PROCEDURE ingresarSistema
@Id numeric(5), @log NVARCHAR(50)
AS
BEGIN TRY
    declare @identificador numeric(5)
    declare datos cursor scroll for
    SELECT EmployeeID
        FROM HumanResources.Employee
        WHERE (EmployeeID = @Id AND LoginID = @log)
    open datos
    if (@@CURSOR_ROWS=1)
        PRINT 'USUARIO CORRECTO'
    else
        RAISERROR('Usuario incorrecto', 16, 1)
END TRY
BEGIN CATCH
    PRINT 'USUARIO INCORRECTO'
END CATCH
```

Ejecución

```
USE [AdventureWorks]
GO
```

Ejecución

```
USE [AdventureWorks]
GO

DECLARE      @return_value int

EXEC  @return_value = [dbo].[ingresarSistema]
        @Id = 1,
        @log = N'adventure-works\guy1'

SELECT      'Return Value' = @return_value
GO
```

Ejemplo 82.

```
USE AdventureWorks;
GO

-- Verify that the stored procedure does not exist.
IF OBJECT_ID ('usp_GetErrorInfo', 'P') IS NOT NULL
    DROP PROCEDURE usp_GetErrorInfo;
GO

-- Create procedure to retrieve error information.
CREATE PROCEDURE usp_GetErrorInfo
AS
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() as ErrorState,
        ERROR_PROCEDURE() as ErrorProcedure,
        ERROR_LINE() as ErrorLine,
        ERROR_MESSAGE() as ErrorMessage;
GO

BEGIN TRY
    set nocount on
    -- Generate divide-by-zero error.
    print 'Llega'
    SELECT 1/0;
    print 'No llega'
END TRY
BEGIN CATCH
    -- Execute the error retrieval routine.
    EXECUTE usp_GetErrorInfo;
END CATCH;
GO
```

Ejemplo 83. Uso de RAISERROR

```

BEGIN TRY
    -- RAISERROR with severity 11-19 will cause execution to
    -- jump to the CATCH block.
    RAISERROR ('Error raised in TRY block.', -- Message text.
               16, -- Severity.
               1 -- State.
               );
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT
        @ErrorMessage = ERROR_MESSAGE(),
        @ErrorSeverity = ERROR_SEVERITY(),
        @ErrorState = ERROR_STATE();

    -- Use RAISERROR inside the CATCH block to return error
    -- information about the original error that caused
    -- execution to jump to the CATCH block.
    RAISERROR (@ErrorMessage, -- Message text.
               @ErrorSeverity, -- Severity.
               @ErrorState -- State.
               );
END CATCH;

```

Ejercicio 37. Mc-GrawHill.CASE. quiero mostrar de la tabla emple los apellidos de todos los trabajadores separados por coma teniendo en cuenta que los numeros de empleados comienzan en 7369. Tratar los errores posibles, por ejemplo que no exista ningún registro y queramos avanzar el cursor.

Ejercicio 38. Mc-GrawnHill. CASE. Escribe un programa para introducir nuevos pedidos según las siguientes especificaciones:

- Recibirá como parámetros PEDIDO_NO, PRODUCTO_NO, CLIENTE_NO, UNIDADES y la FECHA_PEDIDO (opcional, por defecto la del sistema). Verificará todos estos datos así como las unidades disponibles del producto y el límite de crédito del cliente y fallará enviado un mensaje de error en caso de que alguno sea erróneo.

- Insertará el pedido y actualizará la columna DEBE de clientes incrementándola el valor del pedido (UNIDADES * PRECIO_ACTUAL). También actualizará las unidades disponibles del producto e incrementará la comisión para el empleado correspondiente al cliente en un 5% del valor total del pedido. Todas estas operaciones se realizarán como una única transacción.

Usar las tabals productos08 y clientes08

```

select stock_disponible
    into unidad
    from productos08
select limite_credito
    into limiteCred
    from clientes08

```

6.7 Trigger/Disparador/Desencadenador

[Charte, 2004c]

6.7.1 Desencadenadores DML

Se trata de procesos que se ejecutan automáticamente en el momento en que se efectúe una acción sobre una tabla o vista. Sintaxis:

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ] – Si queremos encriptar el procedimiento y así que no se pueda ver su contenido
{
    { { FOR | AFTER | INSTEAD OF } { [ INSERT ][ , ][ UPDATE ][ , ][ DELETE ] }
        [ WITH APPEND ]
        [ NOT FOR REPLICATION ]
        AS
        [ { IF UPDATE ( column )
            [ { AND | OR } UPDATE ( column ) ]
            [ ...n ]
        | IF ( COLUMNS_UPDATED ( ) { bitwise_operator } updated_bitmask )
            { comparison_operator } column_bitmask [ ...n ]
        }
        sql_statement [ ...n ]
    }
}
```

En las tablas o vistas pueden efectuarse básicamente tres operaciones, aparte de la selección de los datos: inserción de filas, actualización y borrado. Un desencadenador puede ejecutarse tras la propia acción que lo ha lanzado (**AFTER**), o bien sustituir por completo a dicha acción. (**INSTEAD OF**). Se podrá definir varios desencadenadores sobre una misma acción, de manera que se irán ejecutando secuencialmente. Ejemplo de uso de instead of: cuando tenemos una vista sobre dos tablas, decíamos que no podíamos insertar datos en las dos a la vez, pues bien, podemos definir un disparador instead of para que cada vez que se inserte en la vista, se sustituya esa inserción por una inserción directa y separada en la dos tablas que forman la vista.

Existen dos tablas fundamentales al trabajar con disparadores: *inserted*, *deleted*. Estas tablas nos van a permitir trabajar con los datos que van a ser insertados o borrados.

Ejemplo 84. Cómo podemos insertar en una vista que hace referencia a dos tablas sin que el usuario aprecie diferencia.

```
CREATE VIEW titulos_autores AS
    SELECT titles.*, titleauthor.au_id, titleauthor.au_ord, titleauthor.royaltyper FROM titles, titleauthor
    where titles.title_id=titleauthor.title_id
```

```
INSERT INTO titulos_autores
VALUES(/*titles*/'BU1112', 'La leyenda del pisuerga', 'Novela', '0877', 25, 2500, 20, 4000, 'Interesante',
getdate(), /*titleauthor*/'172-32-1176',1,20)
```

```
USE pubs
IF EXISTS (SELECT name FROM sysobjects
    WHERE name = 'insertar_vista' AND type = 'TR')
```



```

DROP TRIGGER insertar_vista
GO
CREATE TRIGGER insertar_vista
ON titulos_autores
INSTEAD OF INSERT, UPDATE
AS
BEGIN
    -- Ojo, titles tiene otro disparador
    INSERT INTO titles (title_id, title, type, pub_id, price, advance, royalty, ytd_sales, notes, pubdate)
    SELECT title_id, title, type, pub_id, price, advance, royalty, ytd_sales, notes, pubdate
    FROM inserted

    INSERT INTO titleauthor (au_id, title_id, au_ord, royaltyper)
    SELECT au_id, title_id, au_ord, royaltyper
    FROM inserted
END
GO

SELECT * FROM [pubs].[dbo].[titles]

delete from titles where title_id='BU1112'

SELECT * FROM titleauthor

delete from titleauthor where au_id='172-32-1176'

```

Ejemplo 85. Utilizar valores de una tabla anteriores a la ejecución del trigger.

```

CREATE TABLE [P] (
    [pn] [nvarchar] (2) ,
    [pnombre] [nvarchar] (50) ,
    [color] [nvarchar] (20),
    [peso] [smallint] ,
    [ciudad] [nvarchar] (50)
)

INSERT INTO P (pn, pnombre, color, peso, ciudad)
VALUES (N'1', N'Botella cocacola', N'Negro', 100, N'Valladolid')

INSERT INTO P (pn, pnombre, color, peso, ciudad)
VALUES (N'2', N'Pan', N'Amarillo', 50, N'Valladolid')

INSERT INTO P (pn, pnombre, color, peso, ciudad)
VALUES (N'3', N'Chorizo', N'Rojo', 90, N'Zamora')

CREATE TRIGGER copia_seguridad
on P
AFTER delete
AS
BEGIN
    IF EXISTS (SELECT name FROM sysobjects
    WHERE name = 'P_copia_seguridad' AND type = 'U' /*Indico que es una tabla*/)
        INSERT P_copia_seguridad (pn, pnombre, color, peso, ciudad, fecha)
        (SELECT pn, pnombre, color, peso, ciudad, getdate() from deleted)
    ELSE

```

```

SELECT *, getdate() fecha INTO P_copia_seguridad
from deleted

END

drop trigger copia_seguridad

delete p

delete P_copia_seguridad

SELECT * FROM P_copia_seguridad

```

Nota.- El disparador se ejecuta como una transacción, si se produce un fallo interno no se confirma ningún cambio realizado en él.

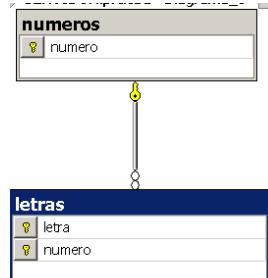
Ejercicio 39. Crear una tabla tmp sobre una base de datos tmp, de manera que tenga dos campos campo1 y campo2. campo1 se irá autoincrementando pero con un disparador, y tomará valores desde la a hasta la z y campo2 será de tipo texto.

Ejercicio 40. Teniendo en cuenta las siguientes tablas:

Tabla numeros y Tabla letras. En numeros existen datos numéricos en un solo campo numero. En letras dos campos: letra y numero. numero relacionado con la tabla numeros.

Por una parte deberemos crear un disparador para que la letra se autoincremente desde a en adelante.

También crearemos otro disparador para que cada vez que se borre de letras, se borre el registro asociado por el campo numero en numeros. Así mismo este disparador llamará a un procedimiento que actualice los registros de letras, para que no tenga saltos en el abecedario en el campo letra.



```

USE [prueba]
GO
***** Objeto: Table [dbo].[numeros]      Fecha de la secuencia de comandos:
09/15/2008 18:09:17 *****
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
CREATE TABLE [dbo].[numeros] (
    [numero] [int] NOT NULL,
    CONSTRAINT [PK_numeros] PRIMARY KEY CLUSTERED
(
    [numero] ASC
)

```



```

) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

USE [prueba]
GO
***** Objeto: Table [dbo].[letras]      Fecha de la secuencia de comandos:
09/15/2008 18:09:31 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[letras](
    [letra] [char](1) NOT NULL,
    [numero] [int] NOT NULL,
CONSTRAINT [PK_letras] PRIMARY KEY CLUSTERED
(
    [letra] ASC,
    [numero] ASC
) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[letras] WITH CHECK ADD CONSTRAINT [FK_letras_numeros]
FOREIGN KEY([numero])
REFERENCES [dbo].[numeros] ([numero])
GO
ALTER TABLE [dbo].[letras] CHECK CONSTRAINT [FK_letras_numeros]

```

Ejemplo 86. Utilizar un desencadenador con un mensaje de correo electrónico de aviso

Este ejemplo envía un mensaje de correo electrónico a una persona especificada (Victor) cuando cambia la tabla **titles**. Para poder realizar el envío de mensajes a través de un servidor de correo debemos configurar la BD.

Microsoft SQL Server Management Studio

Archivo Editar Ver Proyecto Herramientas Ventana Comunidad Ayuda

Nueva consulta Ejecutar Explorador de objetos SRVSQL2K5.pu...QLQuery1.sql* Resumen xp_startmail

```
EXEC master..xp_sendmail 'Victor', 'No olvides lo que comentamos'
```

Conectar Explorador de objetos SRVSQL2K5 (SQL Server 9.0.1399 - CURSOSQL\Administrador)

- Bases de datos
- Seguridad
- Objetos del servidor
- Réplica
- Administración
 - Planes de mantenimiento
 - Registros de SQL Server
 - Monitor de actividad
 - Correo electrónico de base de datos
 - Coordinador de transacciones
 - Búsqueda de texto
 - Heredado
 - Notification Services
 - Agente SQL Server

Configurar Correo electrónico de base de datos

Enviar correo electrónico de prueba... Ver registro de Correo electrónico de base de datos Actualizar

Asistente para configuración de Correo electrónico de base de datos - SRVSQL2K5

Seleccionar tarea de configuración Seleccione tareas de instalación o mantenimiento.

Si está instalando Correo electrónico de base de datos por primera vez, seleccione la opción de instalación.

Instalar Correo electrónico de base de datos realizando las siguientes tareas:

- Crear un nuevo perfil de correo electrónico y especificar sus cuentas SMTP
- Especificar seguridad del perfil
- Configurar parámetros del sistema

Administrar cuentas y perfiles de Correo electrónico de base de datos

Administrar seguridad del perfil

Ver o cambiar parámetros del sistema

Asistente para configuración de Correo electrónico de base de datos - SRVSQL2K5

Asistente para configuración de Correo electrónico de base de datos

Correo electrónico de base de datos es un componente de SQL Server que utiliza el protocolo simple de transferencia de correo (SMTP) para enviar mensajes de correo electrónico. El asistente le ayuda a realizar tareas de instalación y mantenimiento como:

- Administrar cuentas y perfiles de Correo electrónico de base de datos
- Administrar la seguridad de Correo electrónico de base de datos
- Configurar parámetros del sistema de Correo electrónico de base de datos

Haga clic en Siguiente para continuar o en Cancelar para salir del asistente.

Nueva cuenta de Correo electrónico de base de datos

Especificar nombre, descripción y atributos de la cuenta SMTP.

Nombre de cuenta:	victor
Descripción:	
Servidor de correo saliente (SMTP)	
Dirección de correo:	victor@cursosql.net
Nombre para mostrar:	Victor J. Vergel
Correo de respuesta:	
Nombre del servidor:	srvsql2k5
Número de puerto:	25
<input type="checkbox"/> El servidor requiere una conexión segura (SSL)	

Autenticación SMTP

Autenticación de Windows con credenciales del servicio Motor de base de datos

Autenticación básica

Nombre de usuario:
Contraseña:
Confirmar contraseña:

Autenticación anónima

Aceptar Cancelar Ayuda

Asistente para configuración de Correo electrónico de base de datos - SRVSQL2K5

Nuevo perfil Especificar nombre de perfil, descripción, cuentas y prioridad de comutación por error.

Nombre de perfil:	victor
Descripción:	
Un perfil puede estar asociado a varias cuentas SMTP. Si se produce un error de la cuenta al enviar un mensaje de correo electrónico, el perfil usa la siguiente cuenta en la lista de prioridades. Especifique las cuentas asociadas al perfil y muéve las cuentas para establecer la prioridad de comutación por error.	
Cuentas SMTP:	
Nombre de cuenta:	victor
Dirección de correo electrónico:	victor@cursosql.net
<input type="button"/> Agregar... <input type="button"/> Quitar	

Asistente para configuración de Correo electrónico de base de datos - SRVSQL2K5

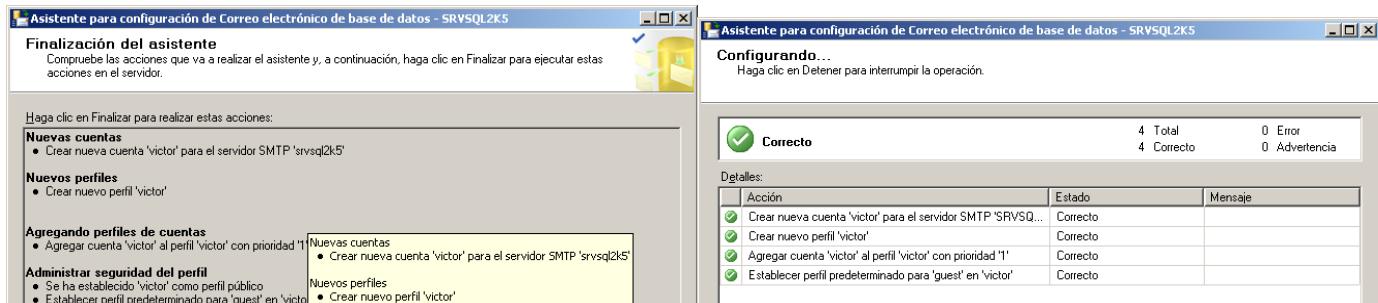
Administrar seguridad del perfil Especifique los usuarios o las funciones de la base de datos que tienen acceso a los perfiles.

Perfiles públicos Perfiles privados
Todos los usuarios de cualquier base de datos host de correo pueden obtener acceso a un perfil público.
Seleccionar perfiles públicos. También puede especificar el perfil público predeterminado.
Publico Nombre de perfil Perfil predeterminado
victor Sí

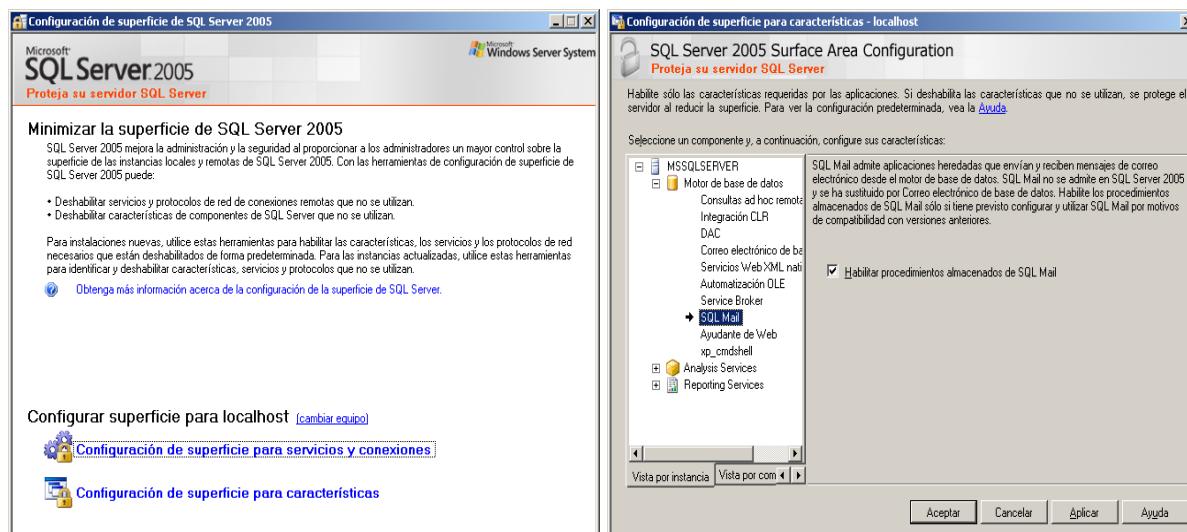
Asistente para configuración de Correo electrónico de base de datos - SRVSQL2K5

Configurar parámetros del sistema Vea o cambie los parámetros del sistema de Correo electrónico de base de datos.

Parámetro	Valor
Número de reintentos de cuenta	1
Intervalo entre reintentos de cuenta (segundos)	60
Tamaño máximo del archivo (bytes)	1000000
Extensiones de archivo adjunto prohibidas	exe,dll,vbs,js
Duración mínima del ejecutable de Correo electrónico de base de d...	600
Nivel de registro	Extendido



Habilitamos el correo electrónico, para ello en configuración de superficie para características:



```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'SQL Mail XPs', 1;
GO
RECONFIGURE;
GO
```

De la siguiente manera podríamos enviar correos electrónicos de todo.

```
-- xp_startmail

-- Deprecated. EXEC master..xp_sendmail 'Victor', 'No olvides lo que
comentamos'

EXEC msdb.dbo.sp_send_dbmail -- Este procedimiento está localizada en la BD
del sistema msdb
@profile_name = 'victor',
```

```
@recipients = 'fernando@cursosql.net',
@body = 'aaaaaaaaaaaaaa',
@subject = 'Automated Success Message' ;
```

Y esté sera el disparador que inicialmente deseábamos crear.

```
USE pubs
IF EXISTS (SELECT name FROM sysobjects
            WHERE name = 'reminder' AND type = 'TR')
    DROP TRIGGER reminder
GO
CREATE TRIGGER reminder
ON titles
FOR INSERT, UPDATE, DELETE
AS
-- Antiguo EXEC master..xp_sendmail 'Victor', 'No olvides lo que comentamos'
EXEC msdb.dbo.sp_send_dbmail
    @profile_name = 'victor',
    @recipients = 'victor@cursosql.net',
    @body = No olvides lo que comentamos,
    @subject = 'Mensaje enviado por un trigger' ;
```

GO

master..xp_readmail – Para leer el correo

6.7.2 Desencadenadores DDL

También podremos crear desencadenadores o triggers a nivel de base de datos no sólo a nivel de consulta. Un sola operación DDL puede disparar múltiples DDL triggers.

Sintaxis:

Trigger on a CREATE, ALTER, DROP, GRANT, DENY, REVOKE, or UPDATE STATISTICS statement (DDL Trigger)

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement [ ; ][ ...n ] | EXTERNAL NAME < method specifier > [ ; ] }
```

donde event_type puede ser CREATE_TABLE, DROP_TABLE, ALTER_TABLE, CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE cuando se aplica sobre la BD (DATABASE) ó CREATE_DATABASE, ALTER_DATABASE, DROP_DATABASE si el trigger es con la cláusula ALL SERVER.

Ejemplo 87. Ejemplo de disparador que evitará que se modifique o borre cualquier tabla.

```
CREATE TRIGGER seguridad
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
```

```
PRINT 'Debe deshabilitar el disparador de seguridad para realizar esta acción.'
ROLLBACK
;
```

Ejemplo 88. Trata de la creación de un trigger de BD que registrará nombre de usuario Windows “texto”, número de tablas creadas “numero” y fecha del sistema en el momento de crearlo “fecha”

```
GO
CREATE TABLE dbo.AuditLog
(Command NVARCHAR(1000),
PostTime NVARCHAR(24),
HostName NVARCHAR(100),
LoginName NVARCHAR(100)
)
GO

-- create DDL trigger
CREATE TRIGGER AuditOperations
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
DECLARE @data XML
DECLARE @cmd NVARCHAR(1000)
DECLARE @posttime NVARCHAR(24)
DECLARE @spid NVARCHAR(6)
DECLARE @hostname NVARCHAR(100)
DECLARE @loginname NVARCHAR(100)
SET @data = eventdata()

SET @cmd = CONVERT(NVARCHAR(100),
@data.query('data(/TSQLCommand/CommandText)'))
SET @posttime = CONVERT(NVARCHAR(24),
@data.query('data(/PostTime)'))
SET @spid = CONVERT(NVARCHAR(6),
@data.query('data(/SPID)'))
SET @hostname = HOST_NAME()
SET @loginname = SYSTEM_USER
INSERT INTO dbo.AuditLog(Command,PostTime,HostName>LoginName)
VALUES(@cmd, @posttime, @hostname, @loginname)
SELECT @data
GO

CREATE TABLE dbo.Test(col INT)
GO
DROP TABLE dbo.Test
GO

select * from auditlog
```

Ejemplo 89. Registro de las creaciones de tabla realizadas sobre la BD.

```
--Necesitamos la siguiente tabla:
CREATE TABLE [dbo].[registro] (
```

```

[registrooperaciones] [xml] NULL
)

CREATE TRIGGER controlEsquema
ON DATABASE FOR CREATE_TABLE
AS
    -- Donde eventdata devuelve un XML que almacenará toda la información
    -- relacionada con el evento generado.
    insert into registro VALUES (EVENTDATA())
go

--Recuperación de datos:
SELECT * FROM registro
select registrooperaciones.query('data(//LoginName)') from registro
select registrooperaciones.query('data(//ObjectType)') from registro

```

El Standard Data Manipulation Language (DML) triggers crea las tablas **inserted** y **deleted**, permitiendo al desarrollador examinar los datos originales mientras son cambiados y los nuevos valores a los cuales están siendo cambiados. El DDL triggers no crea estas tablas. En su lugar, puede usar la función **eventdata** para obtener información acerca del evento disparando el trigger. La función eventdata recupera un documento EVENT_INSTANCE XML del que los contenidos variarán acorde al destino del DDL trigger. Todos los DDL triggers recuperan un documento que incluye los siguientes elementos:

! <PostTime>

- El tiempo en el cual el trigger fue disparado

! <SPID>

- El numero de ID del proceso de la base de datos que causa el disparo del trigger.

! <EventType>

- El tipo de evento que causo el disparo del trigger, como CREATE_TABLE,... El resto del documento XML contiene información que depende del comando que dispara el trigger.

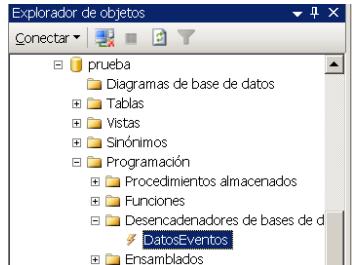
Ejemplo 90. El siguiente código, es un ejemplo que devuelve los elementos PostTime, Database, y TargetObject elements en un CREATE_TABLE DDL trigger y los muestra:

```

CREATE TRIGGER DatosEventos
ON DATABASE
FOR CREATE_TABLE
AS
DECLARE @data XML
DECLARE @posttime NVARCHAR(24)
DECLARE @database NVARCHAR(100)
DECLARE @eventType NVARCHAR(100)
SET @data = eventdata()
SET @posttime = CONVERT(NVARCHAR(24), @data.query('data(//PostTime)'))
SET @database = CONVERT(NVARCHAR(100), @data.query('data(//DatabaseName)'))
SET @eventType = CONVERT(NVARCHAR(100), @data.query('data(//EventType)'))
PRINT @posttime

```

```
PRINT @database
PRINT @eventType
```



```
192.168.0.1.pru... - SQLQuery4.sql* [192.1]
CREATE TRIGGER DatosEventos
ON DATABASE
FOR CREATE_TABLE
AS
DECLARE @data XML
DECLARE @posttime NVARCHAR(255)
DECLARE @database NVARCHAR(128)
DECLARE @targetobject NVARCHAR(128)
SET @data = eventdata()
SET @posttime = CONVERT(NVARCHAR, GETDATE(), 121)
SET @database = CONVERT(NVARCHAR, DB_NAME())
SET @targetobject = CONVERT(NVARCHAR, OBJECT_NAME(@targetobject))
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[emp2](
    [empno] [smallint] NOT NULL,
    [ename] [nvarchar](10) COLLATE Modern_Spanish_CI_AS,
    [sal] [smallmoney] NULL,
    [salmedio] [smallmoney] NULL,
    CONSTRAINT [PK_emp2] PRIMARY KEY CLUSTERED
        ([empno])
)
Mensajes
2008-08-30T10:52:17.780
prueba
CREATE_TABLE
```

Imagen 59. Localización del disparador, y salida por pantalla después de la creación de una tabla

Ejemplo de un fichero XML generado:

```
<EVENT_INSTANCE>
<PostTime>2004-06-18T02:14:20.640</PostTime>
<SPID>58</SPID>
<EventType>UPDATE_STATISTICS</EventType>
<ServerName>SQL2005PC</ServerName>
<LoginName>SQL2005PC\Administrator</LoginName>
<UserName>SQL2005PC\Administrator</UserName>
<DatabaseName>AdventureWorks</DatabaseName>
<SchemaName>Production</SchemaName>
<ObjectType>STATISTICS</ObjectType>
<TargetObjectName>Product</TargetObjectName>
<TargetObjectType>TABLE</TargetObjectType>
<TSQLCommand>
    <SetOptions ANSI_NULLS="ON" ANSI_NULL_DEFAULT="ON"
    ANSI_PADDING="ON" QUOTED_IDENTIFIER="ON"
    ENCRYPTED="FALSE"/>
    <CommandText>
        UPDATE STATISTICS Production.Product
    </CommandText>
</TSQLCommand>
</EVENT_INSTANCE>
```

Algunos datos interesantes que podemos tener sobre los disparadores o procedimientos en general son :

```
SELECT name FROM sys.triggers
SELECT definition FROM sys.sql_modules
SELECT * FROM sys.trigger_events
```

6.7.3 Desencadenadores logon

Los desencadenadores logon activan procedimientos almacenados en respuesta a un evento LOGON. Puede utilizar desencadenadores logon para realizar auditorías y controlar sesiones de servidor, como el seguimiento de la actividad de inicio de sesión, la restricción de

inicios de sesión en SQL Server o la limitación del número de sesiones para un inicio de sesión específico. Por ejemplo, en el siguiente código, el desencadenador logon rechaza los intentos de iniciar sesión en SQL Server iniciados por el inicio de sesión login_test si ya hay tres sesiones de usuario creadas por dicho inicio de sesión.

Nota.- La cláusula WITH EXECUTE AS permite ejecutar cualquier instrucción con permisos de un determinado usuario, ej:

```
USE Prueba;
GO
ALTER PROCEDURE usp_Demo
WITH EXECUTE AS 'dbo'
AS
SELECT user_name()
go

execute usp_Demo
```

```
USE master;
GO
CREATE LOGIN login_test WITH PASSWORD = '3KHJ6dhx(0xVYsdf' MUST_CHANGE,
    CHECK_EXPIRATION = ON;
GO
GRANT VIEW SERVER STATE TO login_test;
GO
CREATE TRIGGER connection_limit_trigger
ON ALL SERVER WITH EXECUTE AS 'login_test'
FOR LOGON
AS
BEGIN
IF ORIGINAL_LOGIN()= 'login_test' AND
    (SELECT COUNT(*) FROM sys.dm_exec_sessions
        WHERE is_user_process = 1 AND
            original_login_name = 'login_test') > 3
    ROLLBACK;
END;
```

Ejercicio 41. Crear un disparador que almacene en una tabla temporal hora/día de conexión usuario, nombre de usuario y número de conexiones que ha realizado en el último mes.

Ejercicio 42. McGraw-Hill. CASE. Escribe un trigger que permita auditar las modificaciones en la tabla EMPLEADOS, insertando los siguientes datos en la tabla auditaremp: fecha y hora, número de empleado, apellido, la operación de actualización MODIFICACIÓN o INSERCIÓN y el valor anterior y el valor nuevo de cada columna modificada (sólo en las columnas modificadas).

Nota.- Cuando se modifica o borra un trigger, debe especificar la cláusula ON DATABASE o ON ALL SERVER como sea apropiado para el trigger. Si omite esta cláusula, el SQL Server 2005 asumirá que se está refiriendo al trigger Standard y reportará un error porque no lo encontrara.

```
DROP TRIGGER DatosEventos
```



ON DATABASE

Módulo 7 SEGURIDAD EN SQL SERVER 2008

SQL Server 2005 incluye varias características de seguridad configurables y de gran precisión. Estas características permiten a los administradores implementar una defensa optimizada para los riesgos de seguridad específicos de su entorno.

7.1 Comprender los modos de seguridad

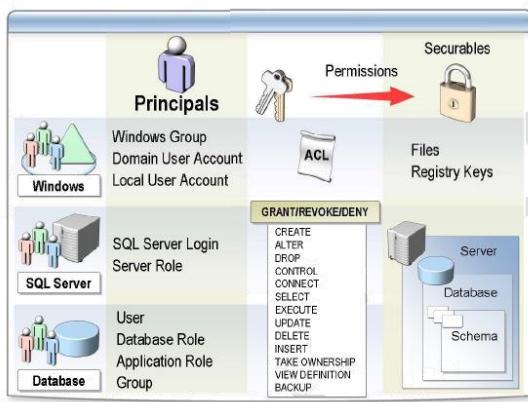


Imagen 60. Esquema de los modelos de seguridad de SQL Server 2005

La seguridad en SQL Server 2005 está lógicamente muy integrada con Windows. Esto hace que tengamos diferentes posibilidades de validación de usuarios a la hora de que diferentes usuarios quieran acceder a la base de datos. La seguridad por tanto estaría en los siguientes niveles:

- ✓ Seguridad a través de Windows: con usuarios locales de un equipo, usuarios de grupos y usuarios del dominio.
- ✓ Seguridad a través de SQL Server. Existen en este caso usuarios a nivel del motor SQL Server, así como Roles.
- ✓ Seguridad a nivel de BD. Cada propia base de datos podrá especificar sus propios usuarios, roles de BD y de aplicaciones que accedan a ella, así como grupos aunque éste último tipo se mantiene sólo por tener compatibilidad con versiones anteriores.

7.1.1.1 Autenticación de Windows

Durante la instalación completa de SQL Server 2005 se agregan los siguientes grupos de seguridad a Windows:

- ✓ SQLServer2005DTSUser\$nombreDeEquipo
- ✓ SQLServer2005MSFTEUser\$nombreDeEquipo\$nombreDeInstancia

- ✓ SQLServer2005MSOLAPUser\$nombreDeEquipo\$nombreDeInstancia
- ✓ SQLServer2005MSSQLServerADHelperUser\$nombreDeEquipo
- ✓ SQLServer2005MSSQLUser\$nombreDeEquipo\$nombreDeInstancia
- ✓ SQLServer2005NotificationServicesUser\$nombreDeEquipo
- ✓ SQLServer2005ReportServerUser\$nombreDeEquipo\$nombreDeInstancia
- ✓ SQLServer2005SQLAgentUser\$nombreDeEquipo\$nombreDeInstancia
- ✓ SQLServer2005SQLBrowserUser\$nombreDeEquipo

Estos grupos simplifican la concesión de los permisos necesarios para ejecutar los servicios de Windows de SQL Server y otros ejecutables. También ayudan a proteger los archivos de SQL Server. El resto de usuarios de Windows podrán conectarse o no a la BD según decida el administrador de la BD.



7.1.2 Autenticación de SQL Server. Inicios de sesión

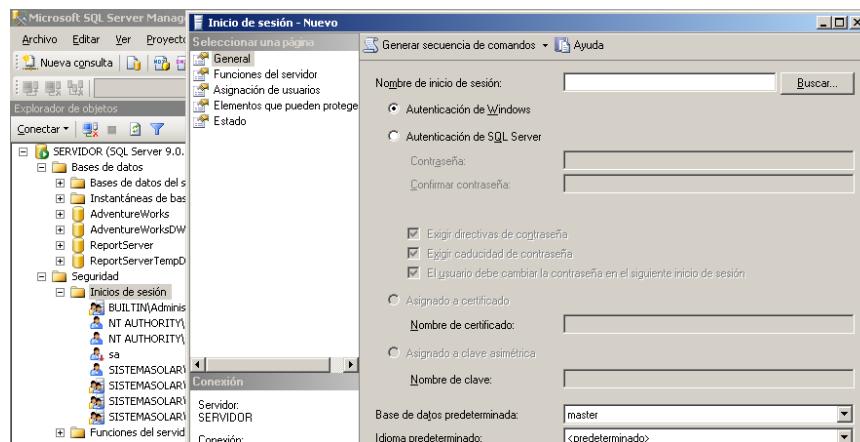


Imagen 61. Creación de un usuario de inicio de sesión en SQL Server

Para poder trabajar con inicios de sesión de SQL Server debemos cambiar la configuración por defecto de nuestro servidor, puesto que está configurado para que inicialmente sólo se validen usuarios de Windows. La validación mixta Windows/SQL Server sería suficiente. Desde sentencias transact-sql también podremos crear usuarios, la sentencia será CREATE LOGIN. Las sentencias ALTER LOGIN y DROP LOGIN permitirán modificarlos o borrarlos.

A estos usuarios podremos asignarles unas determinadas funciones de servidor o roles:

- ✓ **bulkadmin.** Sólo los miembros de la función fija de servidor bulkadmin pueden ejecutar la instrucción BULK INSERT.
- ✓ **dbcreator.** Los miembros de la función fija de servidor dbcreator pueden crear, modificar, quitar y restaurar cualquier base de datos.
- ✓ **diskadmin.** La función fija de servidor diskadmin se utiliza para administrar archivos de disco.
- ✓ **processadmin.** Los miembros de la función fija de servidor processadmin pueden finalizar procesos que se ejecutan en una instancia de SQL Server.
- ✓ **securityadmin.** Los miembros de la función fija de servidor administran los inicios de sesión y sus propiedades. Pueden conceder (GRANT), denegar (DENY) y revocar (REVOKE) permisos en el nivel de servidor. También pueden conceder, denegar y revocar permisos en el nivel de base de datos. Además, pueden restablecer contraseñas para inicios de sesión de SQL Server.
- ✓ **serveradmin.** Los miembros de la función fija de servidor pueden cambiar opciones de configuración en el servidor y cerrar el servidor.
- ✓ **setupadmin.** Los miembros de la función fija de servidor pueden agregar y quitar servidores vinculados, así como ejecutar algunos procedimientos almacenados del sistema.
- ✓ **sysadmin.** Los miembros de la función fija de servidor sysadmin pueden realizar cualquier actividad en el servidor. De manera predeterminada, todos los miembros del grupo de administradores locales, son miembros de la función fija de servidor sysadmin.

Función fija de servidor	Permiso de nivel de servidor
bulkadmin	Se le concede: ADMINISTER BULK OPERATIONS
dbcreator	Se le concede: CREATE DATABASE
diskadmin	Se le concede: ALTER RESOURCES
processadmin	Se le concede: ALTER ANY CONNECTION, ALTER SERVER STATE
securityadmin	Se le concede: ALTER ANY LOGIN
serveradmin	Se le concede: ALTER ANY ENDPOINT, ALTER RESOURCES, ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN, VIEW SERVER STATE
setupadmin	Se le concede: ALTER ANY LINKED SERVER
sysadmin	Se le concede con la opción GRANT: CONTROL SERVER

Tabla 2. Funciones fijas de servidor

Podremos consultar los usuarios de servidor a través de la siguiente vista:

```
SELECT * FROM sys.server_principals
```

	name
1	sa
2	public
3	sysadmin
4	securityadmin
5	serveradmin
6	setupadmin
7	processadmin
8	diskadmin
9	dbcreator
10	bulkadmin
11	##MS_SQLResourceSigningCertificate##
12	##MS_SQLReplicationSigningCertificate##
13	##MS_SQLAuthenticatorCertificate##

Un usuario podrá ver a qué tiene acceso en la base de datos sobre la que esté analizando los resultados de la vista sysobjects:

```
SELECT * FROM sysobjects
```

Nota.-

Una vez conectado como Administrador, una de las primeras tareas que debe realizar es autorizar a otros usuarios a conectarse. Para ello, crearemos un inicio de sesión y le concederemos autorización para obtener acceso a una base de datos como usuario. Utilizar la autenticación de Windows siempre que sea posible.

Ejemplo 91. Creación de un inicio de sesión probando cada la función de servidor dbcreator.

7.2 Seguridad a nivel de base de datos. Usuarios de base de datos

Dentro del servidor de base de datos podemos determinar los usuarios que pueden acceder a los datos de BD concretas y definir a qué nivel queremos que puedan hacer uso de ellas. Al igual que con los inicios de sesión podremos crear usuarios con transact-sql, la instrucción para realizarlo es CREATE USER. ALTER USER y DROP USER para modificar y eliminarlo.

Un **esquema** es una colección de entidades de base de datos que forma un solo espacio de nombres. Un espacio de nombres es un conjunto en el que cada elemento tiene un nombre exclusivo. Por ejemplo, para evitar conflictos de nombre, no puede haber dos tablas en el mismo esquema con el mismo nombre. Dos tablas sólo pueden tener el mismo nombre si se encuentran en esquemas distintos.

Este concepto de esquema es sensiblemente diferente al que existía en SQL Server 2000. En SQL Server 2005, los esquemas existen independientemente del usuario de la base de datos

que los crea. Se puede transferir la propiedad de los esquemas sin cambiar sus nombres. Se pueden crear objetos en esquemas con nombres descriptivos que indican claramente su función. Por ejemplo, en lugar de `accounting.ap.sandra.reconciliation`, puede crear un esquema llamado `accounting.ap.factura.reconciliation`. No es necesario cambiar este nombre cuando se quita un usuario de la base de datos, pues "factura" no es un usuario. Esto simplifica el trabajo de los administradores y programadores de bases de datos.

Ventajas de la separación de esquemas de usuario:

La abstracción de usuarios de bases de datos de esquemas ofrece varias ventajas a los administradores y programadores.

- ✓ Un mismo esquema puede ser propiedad de varios usuarios mediante la pertenencia a funciones o grupos de Windows. Esto amplía esta conocida funcionalidad, pues permite a las funciones y los grupos ser propietarios de objetos.
- ✓ La eliminación de usuarios de base de datos resulta mucho más sencilla.
- ✓ No es necesario cambiar el nombre de los objetos del esquema de un usuario para eliminar el usuario de una base de datos. Por tanto, ya no es necesario revisar y probar las aplicaciones que hacen referencia de forma explícita a objetos incluidos en esquemas después de quitar el usuario que los creó.
- ✓ Varios usuarios pueden compartir un solo esquema predeterminado, con lo que la resolución de nombres resulta uniforme.
- ✓ El uso compartido de esquemas predeterminados permite a los programadores almacenar objetos compartidos en un esquema creado específicamente para una aplicación determinada, en lugar de almacenarlos en el esquema DBO.

En SQL Server 2005, cada usuario tiene un esquema predeterminado, que especifica el primer esquema en el que el servidor realizará la búsqueda cuando resuelva los nombres de objetos. Se puede establecer y cambiar el esquema predeterminado mediante la opción `DEFAULT_SCHEMA` de `CREATE USER` y `ALTER USER`. Si no se define `DEFAULT_SCHEMA`, el esquema predeterminado del usuario de la base de datos será DBO.

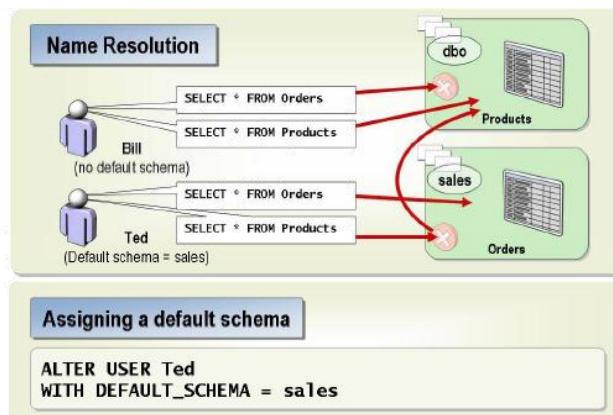


Imagen 62. Ejemplificación uso de dos esquemas dbo y sales.

De la misma manera que para los **inicios de sesión** definimos los “permisos” o **funciones de servidor** para cada **usuario** podremos establecer las **funciones a nivel de base de datos** que poseen:

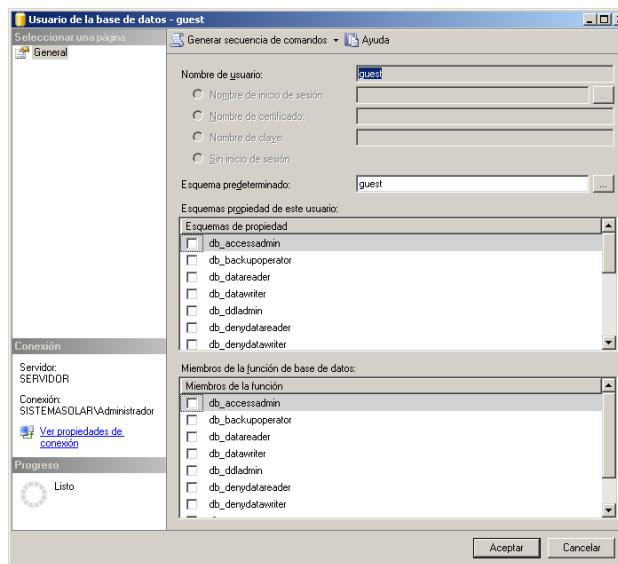


Imagen 63. Funciones de base de datos asignadas al usuario guest/esquema de AdventureWorks

- ✓ **db_accessadmin.** Los miembros de la función fija de base de datos db_accessadmin pueden agregar o quitar accesos a inicios de sesión de Windows, grupos de Windows e inicios de sesión de SQL Server.
- ✓ **db_backupoperator.** Los miembros de la función fija de base de datos db_backupoperator pueden crear copias de seguridad de la base de datos.
- ✓ **db_datareader.** Los miembros de la función fija de base de datos db_datareader pueden ejecutar la instrucción SELECT en cualquier tabla o vista de la base de datos.
- ✓ **db_datawriter.** Los miembros de la función fija de base de datos db_datawriter pueden

agregar, eliminar o cambiar datos en todas las tablas de usuario.

- ✓ **db_ddladmin.** Los miembros de la función fija de base de datos db_ddladmin pueden ejecutar cualquier comando del lenguaje de definición de datos (DDL) en una base de datos.
- ✓ **db_denydatareader.** Los miembros de la función fija de base de datos db_denydatareader no pueden leer datos de las tablas de usuario dentro de una base de datos.
- ✓ **db_denydatawriter.** Los miembros de la función fija de base de datos db_denydatawriter no pueden agregar, modificar ni eliminar datos de tablas de usuario de una base de datos.
- ✓ **db_owner.** Los miembros de la función fija de base de datos db_owner pueden realizar todas las actividades de configuración y mantenimiento de la base de datos.
- ✓ **db_securityadmin.** Los miembros de la función fija de base de datos db_securityadmin pueden modificar la pertenencia a funciones y administrar permisos.

Función fija de base de datos	Permiso de nivel de base de datos	Permiso de nivel de servidor
db_accessadmin	Se le concede: ALTER ANY USER, CREATE SCHEMA	Se le concede: VIEW ANY DATABASE
db_accessadmin	Se le concede con la opción GRANT: CONNECT	
db_backupoperator	Se le concede: BACKUP DATABASE, BACKUP LOG, CHECKPOINT	Se le concede: VIEW ANY DATABASE
db_datareader	Se le concede: SELECT	Se le concede: VIEW ANY DATABASE
db_datawriter	Se le concede: DELETE, INSERT, UPDATE	Se le concede: VIEW ANY DATABASE
db_ddladmin	Se le concede: ALTER ANY ASSEMBLY, ALTER ANY ASYMMETRIC KEY, ALTER ANY CERTIFICATE, ALTER ANY CONTRACT, ALTER ANY DATABASE DDL TRIGGER, ALTER ANY DATABASE EVENT, NOTIFICATION, ALTER ANY DATASPACE, ALTER ANY FULLTEXT CATALOG, ALTER ANY MESSAGE TYPE, ALTER ANY REMOTE SERVICE BINDING, ALTER ANY ROUTE, ALTER ANY SCHEMA, ALTER ANY SERVICE, ALTER ANY SYMMETRIC KEY, CHECKPOINT, CREATE AGGREGATE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE QUEUE, CREATE RULE, CREATE SYNONYM, CREATE TABLE, CREATE TYPE, CREATE VIEW, CREATE XML SCHEMA COLLECTION, REFERENCES	Se le concede: VIEW ANY DATABASE
db_denydatareader	Se le deniega: SELECT	Se le concede: VIEW ANY

		DATABASE
db_denydatawriter	Se le deniega: DELETE, INSERT, UPDATE	
db_owner	Se le concede con la opción GRANT: CONTROL	Se le concede: VIEW ANY DATABASE
db_securityadmin	Se le concede: ALTER ANY APPLICATION ROLE, ALTER ANY ROLE, CREATE SCHEMA, VIEW DEFINITION	Se le concede: VIEW ANY DATABASE

Tabla 3. Datos asociados en la ayuda a Permisos de las funciones fijas de base de datos

La diferencia entre inicios de sesión de SQL Server y de usuarios de base de datos queda claramente patente. Por defecto en la instalación de nuestro servidor se crea un usuario de inicio de sesión llamado **sa** y en cada BD un usuario llamado **guest**, ambos inician deshabilitados por temas de seguridad. Los permisos concedidos al usuario guest se aplican a todos los usuarios que no disponen de una cuenta en la base de datos, eso significa que al iniciar una sesión se considera que la identidad del usuario es guest cuando el nombre de inicio de sesión tiene acceso a una instancia de Microsoft SQL Server, pero no a la base de datos a través de su propia cuenta de usuario y cuando la base de datos contiene una cuenta de usuario llamada guest.

No se puede quitar el usuario guest, pero se puede deshabilitar revocando su permiso CONNECT. El permiso CONNECT se puede revocar ejecutando REVOKE CONNECT FROM GUEST dentro de cualquier base de datos que no sea master ni tempdb.

Normalmente los inicios de sesión se mapean a usuarios de base de datos. Por defecto, todos los logins (inicios de sesión) con el rol fijo de servidor sysadmin es mapeado al usuario dbo en todas las bases (no confundir con el esquema dbo usado por todos los usuarios que no tienen definido ningún esquema por defecto). Algunos usuarios de bases de datos no son mapeados para logins. Por ejemplo, crear un usuario llamado guest en una base de datos permite acceso para cualquiera con un nombre de login valido pero sin una cuenta de usuario. Otros usuarios no mapeados incluyen sys y INFORMATION_SCHEMA.

Las funciones de servidor existentes también pueden ser consultas en la vista database_principals:

```
SELECT * FROM sys.database_principals
```

Ejemplo 92. Crear una cuenta de usuario de Windows y una cuenta de grupo. Se colocará a continuación la cuenta de grupo en la función fija del servidor sysadmin.

- 1.- Conectarse como administrador del equipo.
- 2.- Mi Pc, botón derecho administrar.
- 3.- Administración de equipos, usuarios y grupos locales.
- 4.- Usuarios y grupos locales- usuarios – usuario nuevo.
- 5.- SQLAdmin en el cuadro de texto Nombre de usuario y colocar contraseña.
- 6.- Crear un grupo de nombre SQLManager.

7.- Agregar SQLAdmin a SQLManager

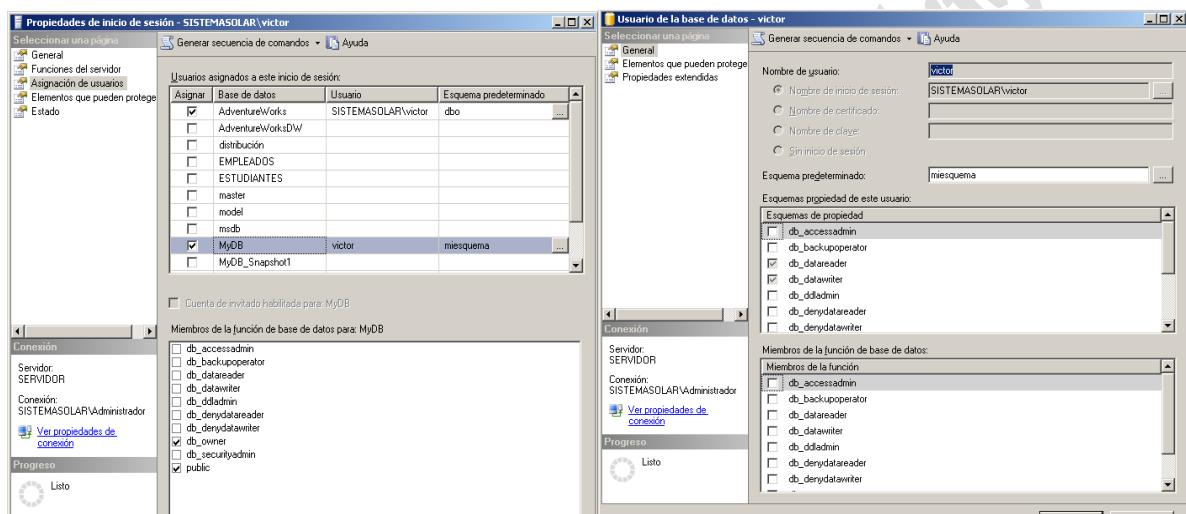
8.- En SQLServer Management Studio ampliamos la carpeta de Seguridad- Inicios de sesión – Nuevo inicio de sesión.

9.- En general colocar el grupo SQLManager.

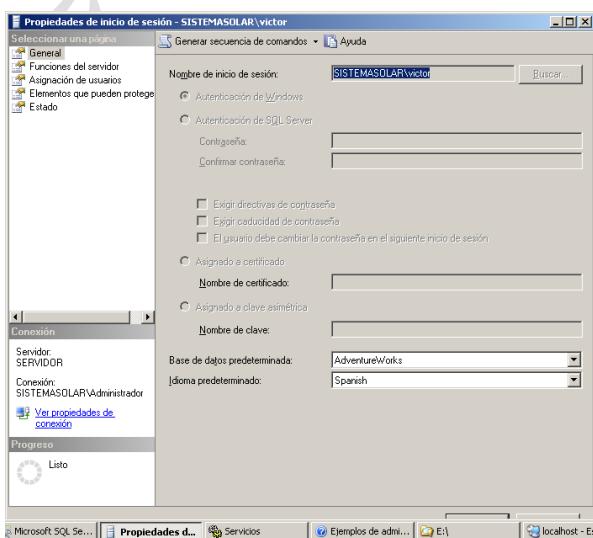
10.- En funciones del servidor, activar sysadmin.

Ejercicio 43. Esquemas

Crear inicio de sesión como victor. Aquí debemos asignarle el db_owner sobre la BD MyDB, para que luego en la BD MyDB le asignemos los permisos de lectura y escritura.



En inicio de sesión victor tiene asociada la bd adventureworks y por eso la puede ver:



Creé un esquema en MyDB, y creo que sólo puede ver ese esquema.

Resumiendo. Por defecto todos los inicios de sesión con la función de servidor de sysadmin, tienen acceso como usuario guest en el esquema dbo en todas las base de datos. El usuario guest habilitado en una base de datos permitirá acceder a la ella a cualquiera que tenga un inicio de sesión en el servidor. Cualquier base de datos tiene un esquema llamado dbo, será el esquema por defecto para todos los usuarios que no tengan definido expresamente un esquema.

SQL Server 2005 a la hora de aplicar una resolución de nombres sigue el siguiente esquema:

1. Si el usuario de la BD con la que estamos ejecutando una sentencia tiene un esquema por defecto, SQL Server intentará encontrar el objeto en ese esquema.
2. Si el objeto no se encuentra en el esquema por defecto, o no lo tiene asignado buscará entonces en el esquema dbo.

(Véase Imagen 62. Ejemplificación uso de dos esquemas dbo y sales.)

Podemos definir entonces permisos también exclusivamente a nivel de BD o de tabla, vista, etc.

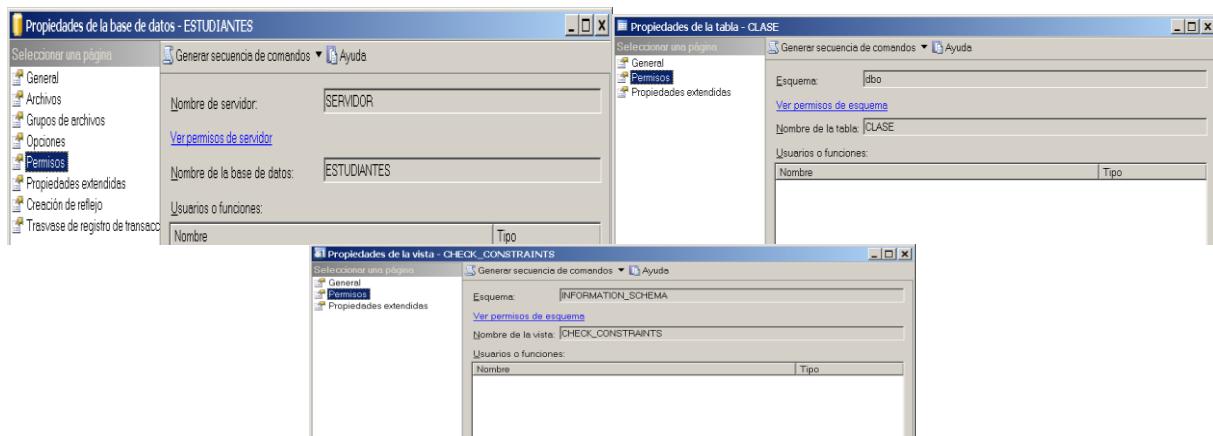


Imagen 64. Propiedades de diferentes objetos de la BD

Ejercicio 44. Crearemos localmente en Windows un grupo de usuarios llamado profesores y otro grupo de usuarios llamado alumnos. En el dominio hay un alumno y un profesor (ej.: pz1garcilaso01 y az1garcilaso01). Estos usuarios pertenecerán a sus grupos del equipo local. Debemos permitir el acceso al servidor sql server a ambos usuarios. Sobre la BD prueba el profesor tendrá el control total y su esquema por defecto se llamará profesor, mientras que el alumno sólo podrá leer datos a excepción de que podrá leer y borrar datos de una tabla sugerencias (comentarios varchar(20)) del esquema del profesor. Crearemos posteriormente un inicio de sesión sólo de sql server que sólo podrá crear tablas en la BD prueba. Comprobar todos los casos posibles.

Ejercicio 45. Verificar que creando un usuario en el dominio que sea introducido simplemente en Active Directory en el grupo SQLSERVER2005MSSQLUSER\$..... podrá entrar en SQL Server como administrador. Analizar por qué.

Probar copia de una base de datos.

Ejercicio 46. Tenemos un usuario llamado milagros con una tabla temperaturas en la que se almacena temperaturas medias de diferentes ciudades.

```
CREATE TABLE TEMPERATURAS (CIUDAD VARCHAR(30), TEMPERATURA NUMERIC(3));
```

```
insert into temperaturas VALUES ('Valladolid',17);
insert into temperaturas values ('Zamora',15);
insert into temperaturas values ('Sevilla',27);
insert into temperaturas values ('Madrid',19);
```

- Crear un usuario francisco para que pueda seleccionar e insertar datos en la tabla temperaturas.
- Milagros concede a todos los privilegios de selección de datos sobre la tabla a todos los usuarios, incluyendo a lo que se creen después de ejecutar esta orden.
- MILAGROS concede privilegios a JUAN (crear dicho usuario con la opción crear, no crear como) sobre temperaturas para que pueda modificar sólo la columna temperatura.
- MILAGROS concede a FRANCISCO el privilegio para insertar en temperaturas, además, para que él pueda pasar este privilegio a otros usuarios.
- Retiramos el permiso de consultar temperaturas a todos los usuarios.
- Habilita a francisco para que pueda entrar a la consola de Enterprise Manager y pueda crear y borrar un usuario desde ahí.

7.3 Credenciales

Una credencial es un registro que contiene la información de autenticación (credenciales) necesaria para conectarse a un recurso situado fuera de SQL Server. Esta información es utilizada internamente por SQL Server. La mayoría de las credenciales incluyen un nombre de usuario y una contraseña de Windows.

La información almacenada en una credencial permite al usuario que se haya conectado a SQL Server 2005 mediante autenticación de SQL Server obtener acceso a recursos situados fuera de la instancia de servidor. Cuando el recurso externo es Windows, el usuario se autentica como el usuario de Windows especificado en la credencial. Se puede asignar una única credencial a varios inicios de sesión de SQL Server. Sin embargo, un inicio de sesión de SQL Server sólo se puede asignar a una credencial.

Las credenciales del sistema se crean de forma automática y se asocian a extremos específicos. Los nombres de las credenciales del sistema comienzan por dos signos de número (##).

7.4 Execute as

Puede usar la cláusula EXECUTE AS en un proceso almacenado o la función para cambiar la identidad usada en esta ejecución de una sentencia. Sintaxis:

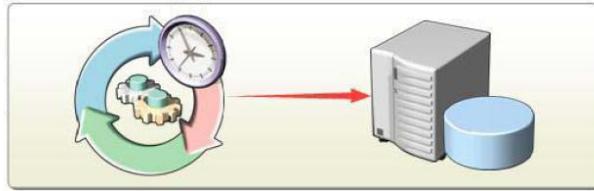
```
EXECUTE AS { CALLER | SELF | OWNER | user_name }
```

Opción	Descripción
CALLER	Ejecute usando la identidad del usuario que llama. Este es el parámetro por defecto.
SELF	Ejecute usando la identidad del usuario que creo el proceso almacenado o la función.
OWNER	Ejecute usando la identidad del propietario de la función.
<i>user_name</i>	Ejecute usando la identidad del usuario específico. Para especificar un nombre de usuario, debe ser un miembro del rol fijo del servidor sysadmin o del rol fijo de la base de datos db_owner , tener permisos CONTROL SERVER en el servidor, tener permisos CONTROL en la base de datos, o tener permiso IMPERSONATE en el login correspondiente al usuario <i>user_name</i> .

Ejemplo 93 .

```
CREATE PROCEDURE GetOrders
WITH EXECUTE AS SELF
AS
SELECT * FROM sales.orders
```

Módulo 8 AGENTE DE SQL SERVER



8.1 Introducción

El Agente SQL Server es un servicio de Microsoft Windows que le permite automatizar algunas tareas administrativas. El Agente SQL Server ejecuta trabajos, supervisa SQL Server y procesa alertas. El servicio Agente SQL Server debe estar en ejecución para poder ejecutar automáticamente los trabajos administrativos locales o multiservidor.

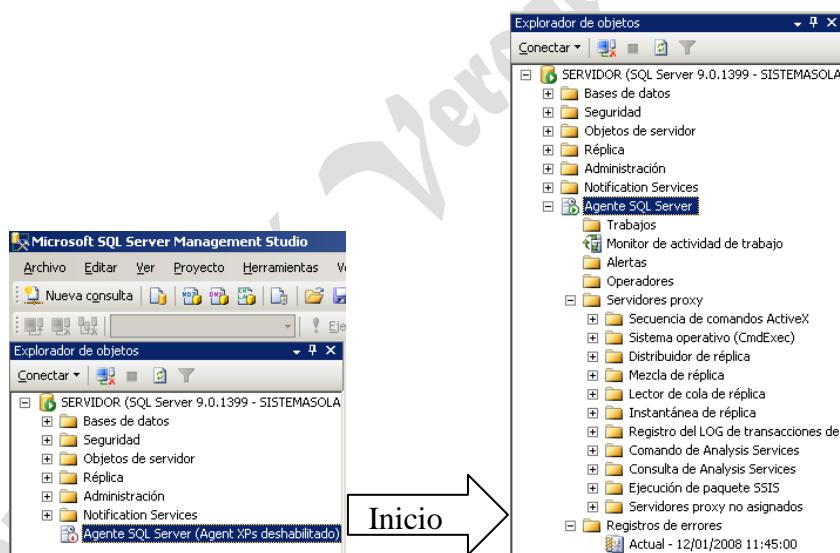


Imagen 65. Agente de SQL Server, por defecto no está iniciado

El Agente SQL Server almacena la mayor parte de la información de configuración en las tablas que residen en la base de datos **msdb**. Para realizar sus funciones, el Agente SQL Server debe configurarse de modo que utilice las credenciales de una cuenta que sea miembro de la función fija de servidor **sysadmin** en SQL Server. La cuenta debe tener los siguientes permisos Windows, ubicados en configuración de seguridad, directivas locales:

- ✓ Ajustar las cuotas de memoria de un proceso
- ✓ Actuar como parte del sistema operativo
- ✓ omitir la comprobación transversal
- ✓ Iniciar sesión como proceso por lotes

- ✓ Iniciar sesión como servicio
- ✓ Reemplazar un símbolo de nivel de proceso

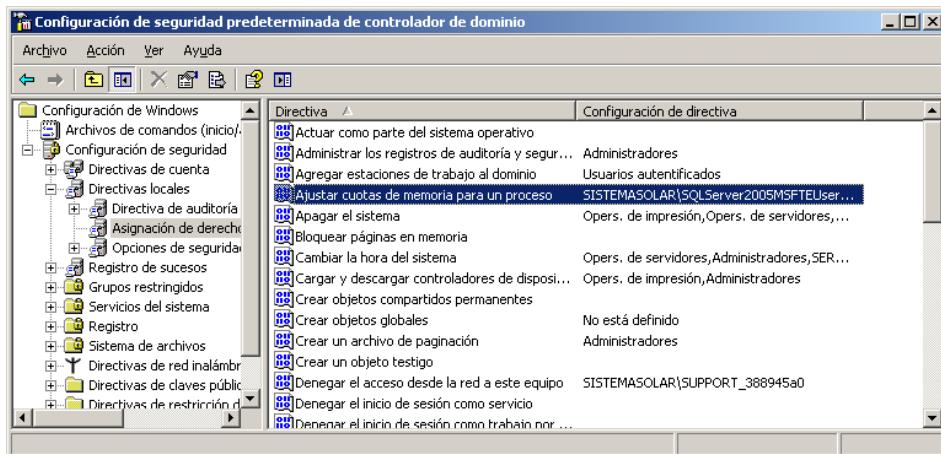


Imagen 66. Directivas de seguridad locales (en este caso de controlador de dominio)

8.2 Propiedades del agente de SQL Server

El agente es configurable para que reinicie el motor de la base de datos o a sí mismo si cualquiera de los dos se detiene inesperadamente. En las propiedades avanzadas se puede especificar el reenvío de eventos a otro servidor dependiendo de la gravedad del mismo.

En Sistema de alerta podremos configurar una cuenta de correo electrónico para enviar mensajes a un administrador.

En el agente viene por defecto deshabilitado la opción de Habilitar perfil de correo, esto junto con el servicio de Messenger de Windows 2003, también deshabilitado por defecto, permitiría notificar los diferentes mensajes del agente a operadores.

Si modificamos el Sistema de correo, podemos elegir a enviar mensajes vía SQLMail (herramienta de SQL Server 2005) o bien con una cuenta configurada por defecto en Outlook Express del servidor para lo cual deberíamos configurar en Herramientas Opciones, no solicitar confirmación de uso de aplicaciones externas. Después de esta configuración en el agente de SQL Server hay que reiniciar el servicio del agente, puesto que tocamos configuración del mismo. (*Por ejemplo véase alertas del agente SQL Server en la pág. 183*).

En Sistema de trabajo se especifica el número de segundos que el Agente SQL Server espera a que los trabajos finalicen antes del cierre. Si el trabajo sigue en ejecución después del intervalo especificado, el Agente SQL Server detendrá de manera obligatoria el trabajo. Por último en Historial podremos decidir cuando y como eliminar el historial del registro del agente SQL Server.

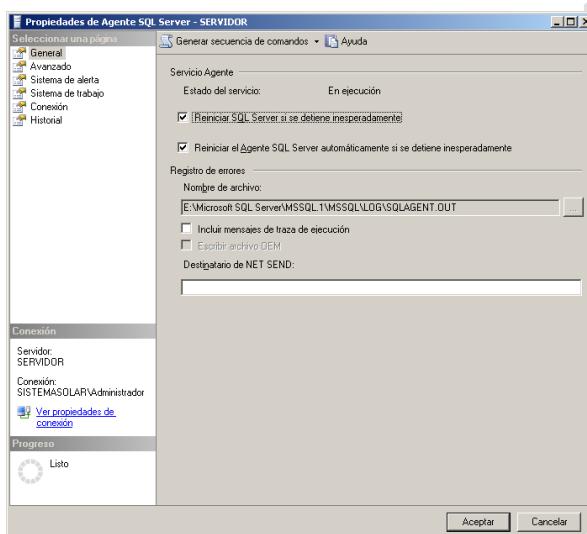
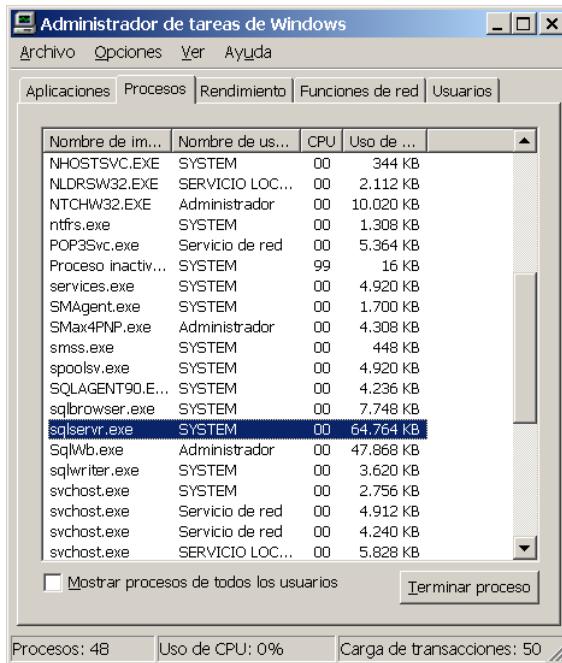


Imagen 67. Propiedades del agente de SQL Server 2005

Para probar que reinicia el servicio automáticamente podemos probar mantando el proceso del servidor de BD:



8.3 Configuración de Operadores

Los operadores son alias para personas o grupos que pueden recibir una notificación electrónica cuando los trabajos finalizan o se activa una alerta.

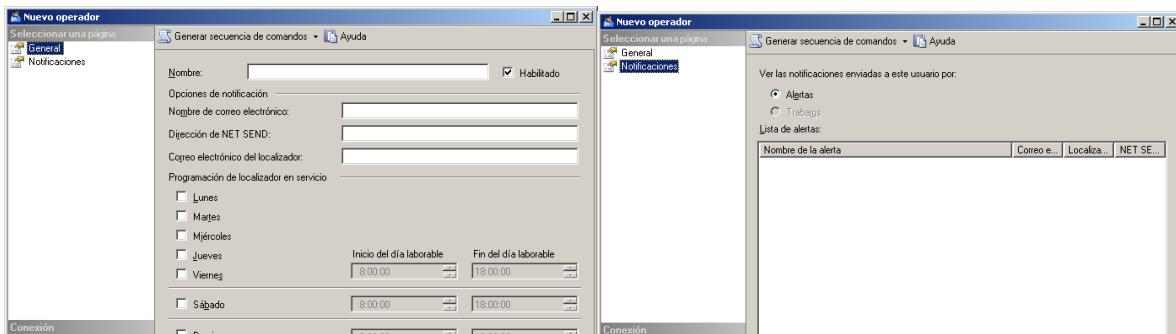


Imagen 68. Creación de un Operador sobre SQL Server.

Será importante seleccionar que el operador está disponible en determinados tramos horarios para recibir notificaciones.

8.4 Creación de Trabajos

Puede utilizar los trabajos del Agente SQL Server para automatizar tareas administrativas rutinarias y ejecutarlas periódicamente, lo que permite que la administración sea más eficaz.

Un trabajo es una serie específica de operaciones que el Agente SQL Server realiza se-

cuencialmente. Un trabajo puede realizar una amplia variedad de actividades, incluidas secuencias de comandos Transact-SQL, aplicaciones de línea de comandos, secuencias de comandos de Microsoft ActiveX, paquetes de Integration Services, comandos y consultas de Analysis Services o tareas de réplica. Los trabajos pueden ejecutar tareas repetitivas o que se pueden programar y pueden notificar automáticamente a los usuarios el estado del trabajo mediante alertas, simplificando mucho la administración de SQL Server.

Puede ejecutar los trabajos manualmente o configurarlos para que se ejecuten de acuerdo con una programación o en respuesta a alertas.

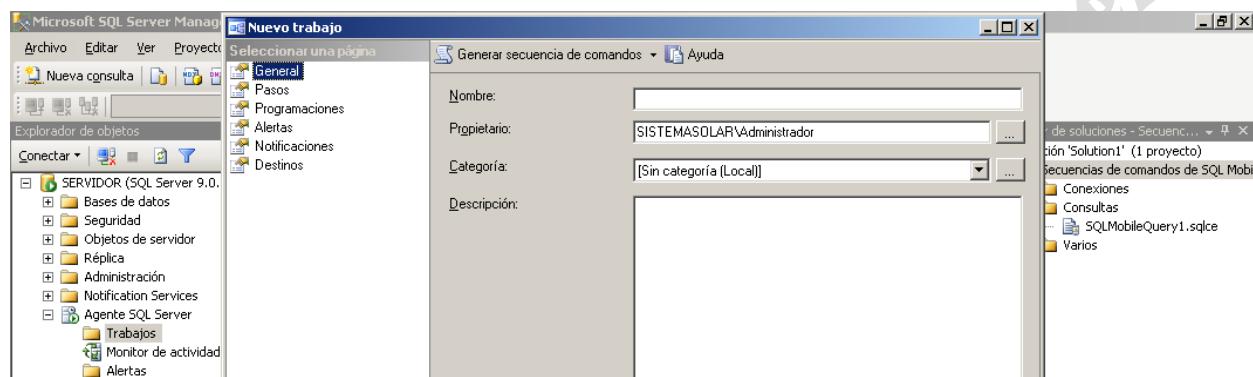


Imagen 69. Creación de un trabajo⁸.

Por razones de seguridad, sólo puede cambiar la definición del trabajo el propietario de éste o un miembro de la función sysadmin. Los miembros de la función sysadmin pueden asignar el valor de propiedad de trabajo a otros usuarios, y pueden ejecutar cualquier trabajo, independientemente del propietario del trabajo.

8.4.1 Categorías

Con botón derecho sobre el explorador de objetos en trabajos, podemos mostrar la ventana de administración de categorías de trabajos. Las categorías de trabajo le ayudan a organizar los trabajos para poder filtrarlos y agruparlos fácilmente. Por ejemplo, puede organizar todos los trabajos de copia de seguridad de las bases de datos en la categoría Mantenimiento de bases de datos. También puede crear sus propias categorías.

⁸ Desactiva la casilla de verificación Habilitado si no desea que el trabajo se ejecute justo después de su creación.

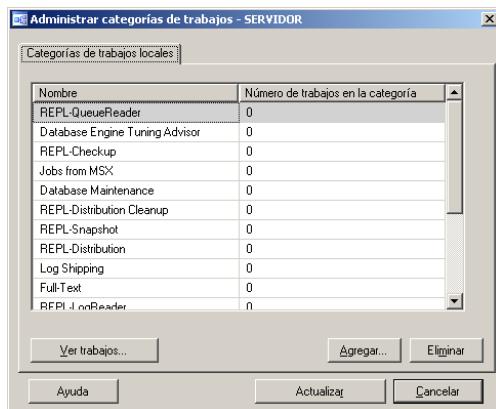


Imagen 70. Administrar categorías de trabajos.

8.4.2 Programación

Cada trabajo evidentemente podrá tener una determinada programación. Para ello debemos configurar cual es nuestra deseado de repetición de una determinada tarea.

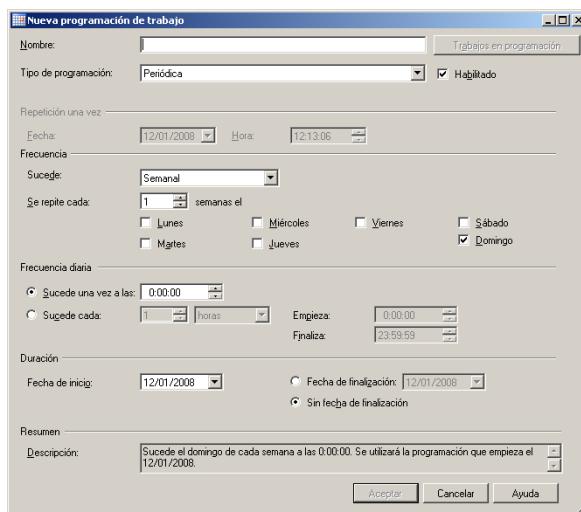


Imagen 71. Programación temporal de un trabajo

8.4.3 Respuestas de trabajo

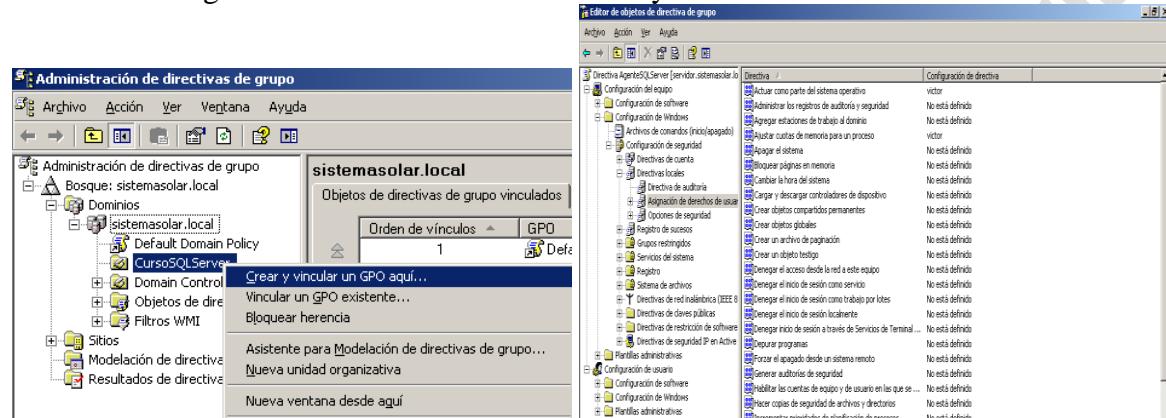
Las respuestas de trabajos especifican acciones que realizará el servicio del Agente SQL Server tras completar un trabajo. Estas respuestas permiten a los administradores de las bases de datos saber cuándo se completan los trabajos y con qué frecuencia se ejecutan. Algunas respuestas de trabajos típicas son:

- ✓ Notificar al operador mediante correo electrónico, localizador electrónico o un mensaje de NET SEND. Utilice una de estas respuestas de trabajo si el operador debe realizar una acción de seguimiento. Por ejemplo, si un trabajo de copia de seguridad se realiza correctamente, se debe notificar de ello al operador para que quite la cinta de copia de seguridad y la guarde en un lugar seguro.

- ✓ Escribir un mensaje de evento en el registro de aplicación de Microsoft Windows. Puede utilizar esta respuesta sólo para trabajos con errores.
- ✓ Eliminar automáticamente el trabajo. Utilice esta respuesta de trabajo si está seguro de que no necesita volver a ejecutar este trabajo.

Ejemplo 94

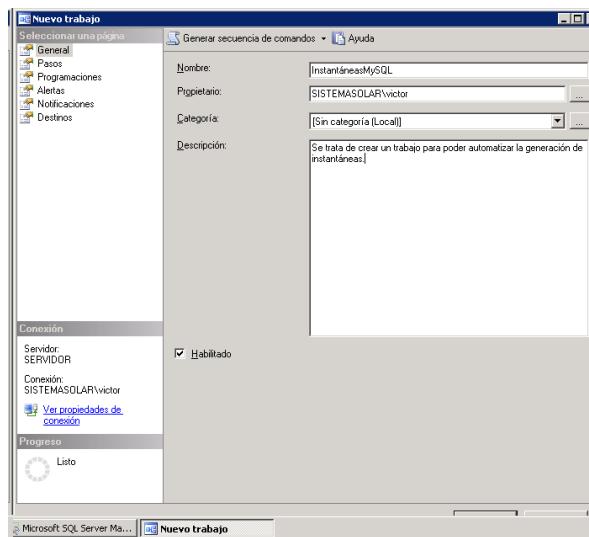
Vamos a crear unas instantáneas de la BD a una determinada hora a lo largo del día, para ello utilizaremos el usuario victor, al cual le crearemos un inicio de sesión SQL Server colocado en la unidad organizativa curso del active directory.



Agregamos el usuario victor al grupo de manejo del agente (active directory) para que le aparezca en el explorador de objetos de la consola de administración y por supuesto para que pueda manejarlo.

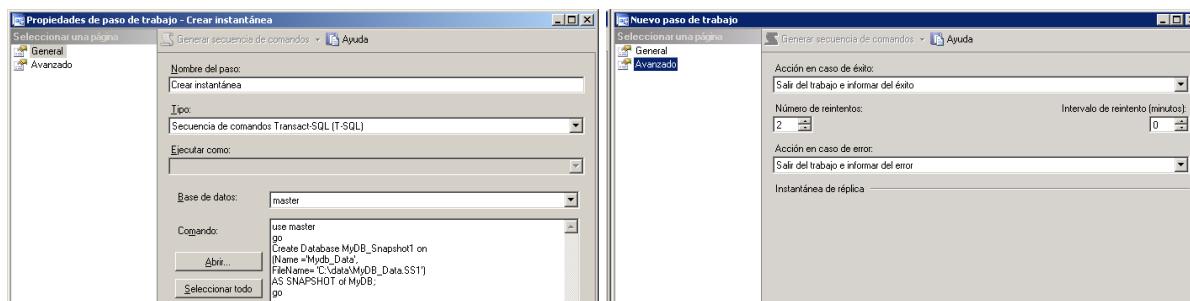


General:

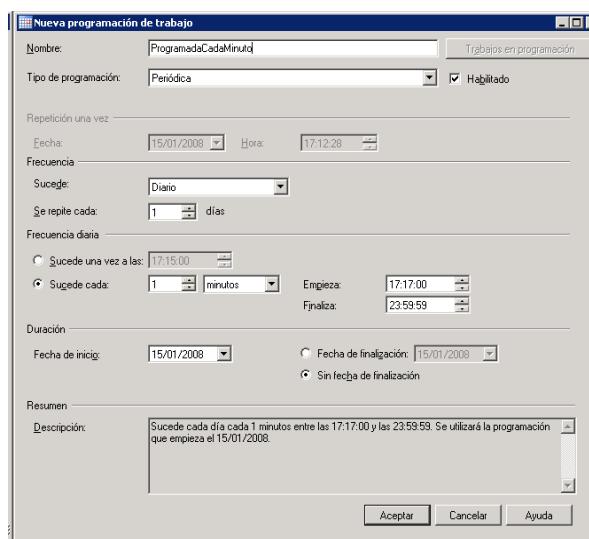


Pasos:

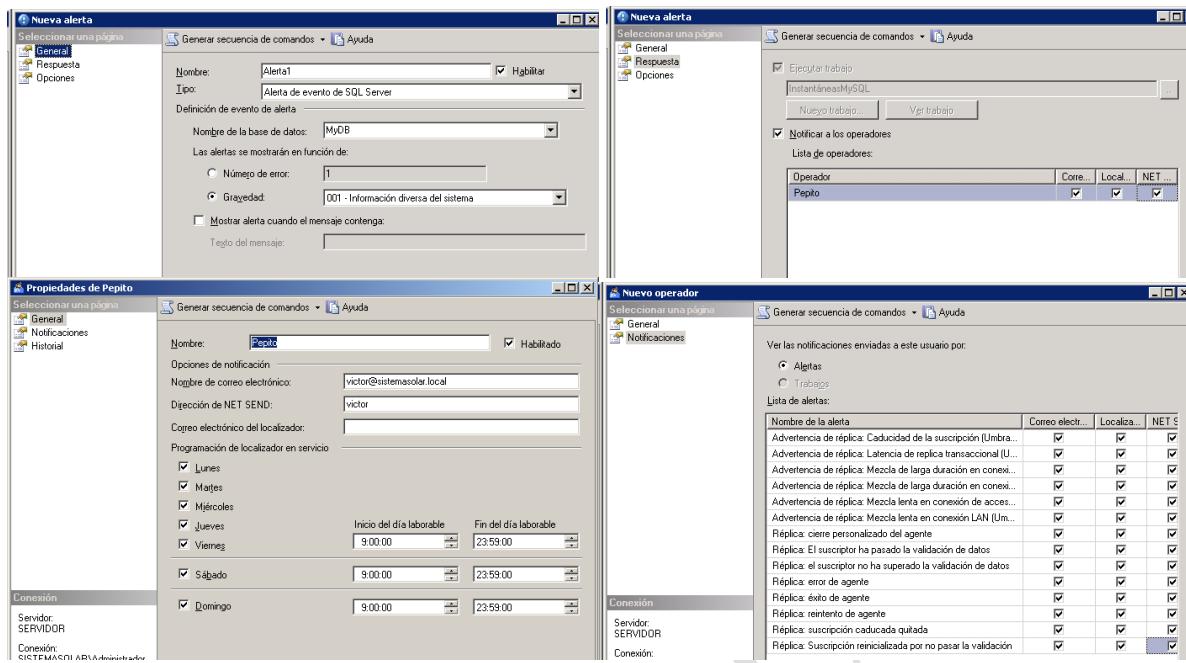
Véase código de creación de instantánea en la pág. 54.



Programaciones:

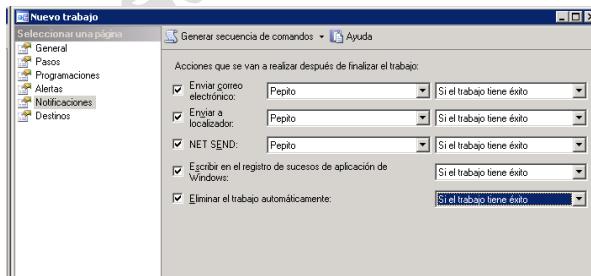


Alertas:



Este tipo de notificación se implementa mediante el correo electrónico. Para la notificación por localizador debe proporcionar una dirección de correo electrónico en la que el operador recibirá los mensajes del localizador. Para establecer la notificación mediante localizador, debe instalar en el servidor de correo un software que procese el correo entrante y lo convierta en mensajes de localizador.

Notificaciones:

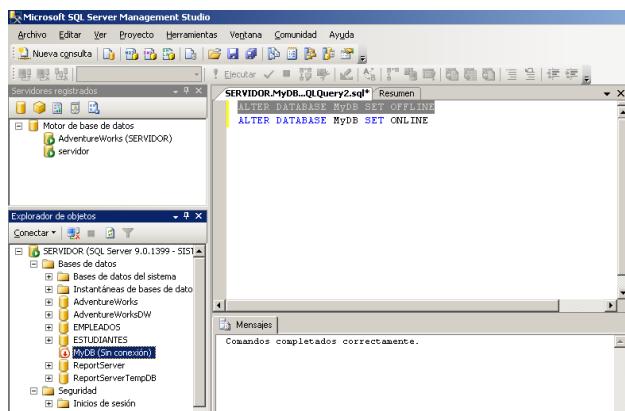


Ojo con éste último check “Eliminar el trabajo automáticamente”. Verificar que el trabajo programado sobre ese agente está habilitada.

Si queremos realizar notificaciones a operadores deberemos configurar la parte del correo electrónico. Para configurar el correo electrónico y poder así enviar a los operadores mensajes véase pág. 155.

Nota importante.- En alguna ocasión al enviar mensaje cuando finaliza el trabajo no llega el correo, da un error porque dice que no esta iniciada sesión de correo. “[264] Se intentó enviar un mensaje de correo electrónico sin establecer una sesión de correo electrónico”. Se soluciona reiniciando el agente.

Ejercicio 47. Queremos que la base de datos MyDB sólo esté levantada en horario laboral. Para ello queremos programar un trabajo que detenga la BD y otro que la levante.



Ejercicio 48 Queremos realizar la copia de una tabla emp de la base de datos Prueba 2 periódicamente añadiendo además un campo nuevo, en el que se indique la fecha en la que se realiza la copia.

Suponemos que tenemos creado el siguiente procedimiento en Prueba 2:

```
CREATE PROCEDURE COPIA_EMP
as
BEGIN
    SELECT *, GETDATE() FECHA INTO EMP3 FROM EMP
End
```

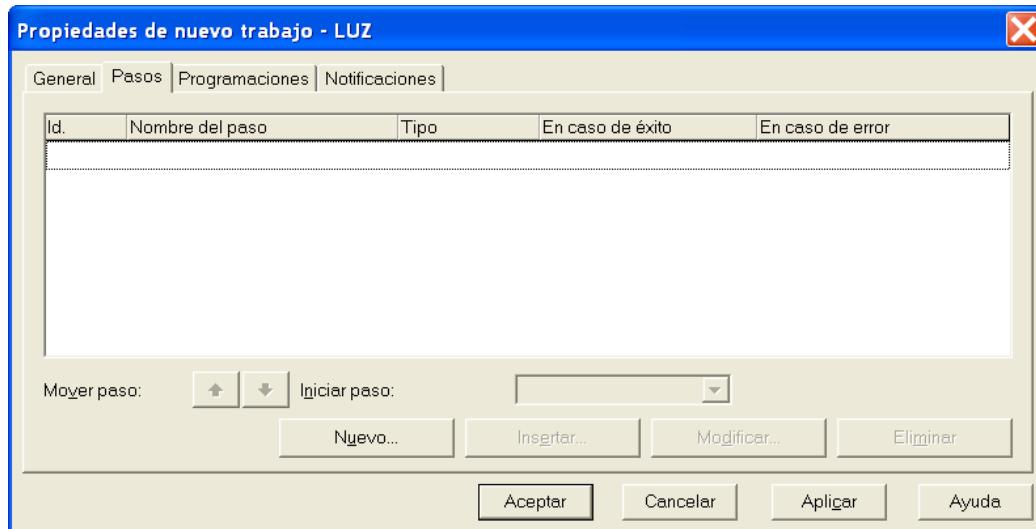


Imagen 72. Creación de pasos del trabajo

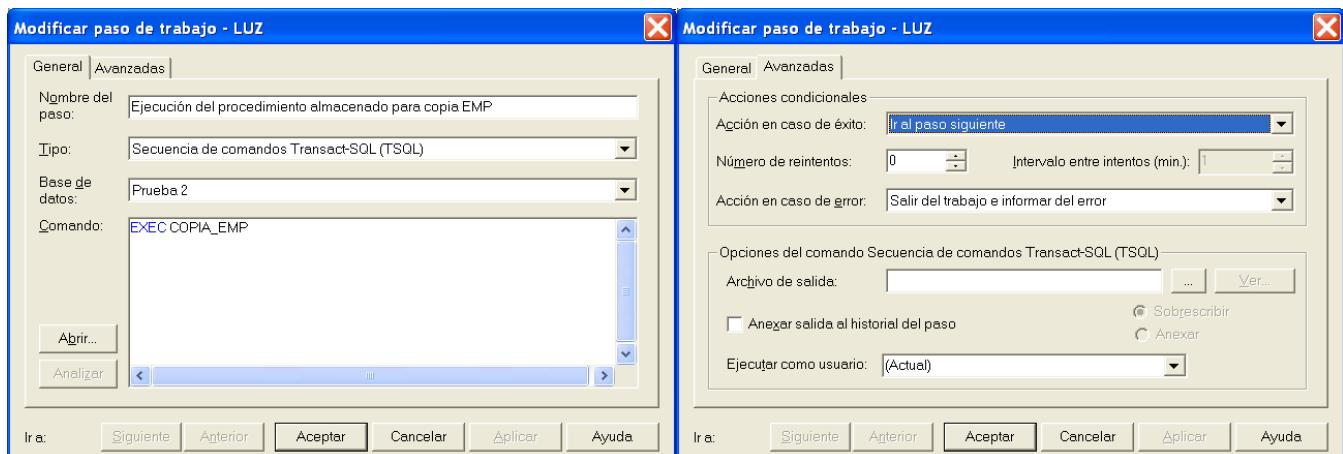


Imagen 73. Configuración del trabajo

El siguiente paso será programar periódicamente su ejecución, será muy importante dejar activado la casilla de Habilitado para que se ejecute el trabajo al llegar la hora de ejecución:

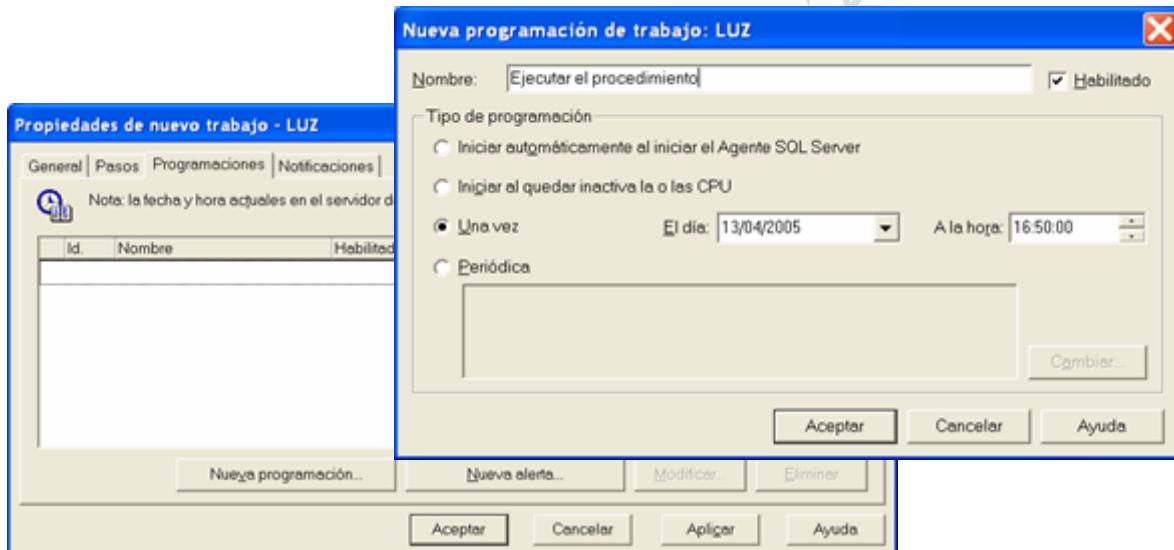


Imagen 74. Programación del trabajo

Por último si queremos notificar la finalización del trabajo si ha tenido éxito, o si queremos registrar en el registro de sucesos de Windows algún comentario, etc.

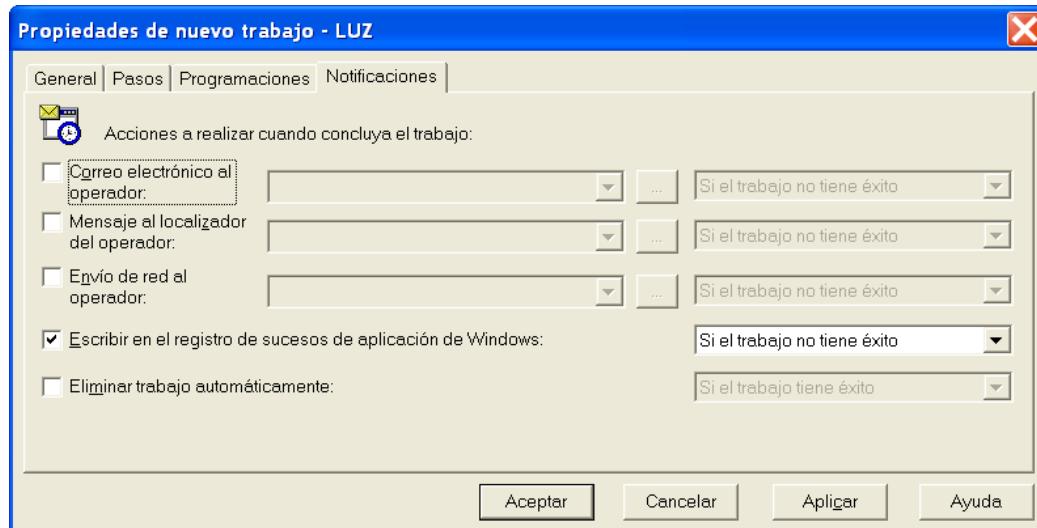


Imagen 75. Notificación de la finalización del trabajo

El trabajo realizado se podrá modificar cuantas veces queramos con botón derecho->Propiedades sobre el trabajo siempre que el usuario sea miembro de la función sysadmin, si esto no fuera así sólo podrá modificar los trabajos que son de su propiedad.

Tenemos que habilitar con botón derecho en la tarea programada para que funcione:

Nombre	Categoría	Habilit...	Ejecutable	Programado	Estado	Estado de la últi...	Fecha de la próxi...
Nuevo trabajo...	SI	SI	SI	No está en eje...	Correcto (14/04...	(No están disponib...	
Copiar tabla emp	SI	SI	SI	No está en eje...	Correcto (13/04...	(No están disponib...	
80.81.114.47 (Windows NT)	SI	SI	SI	No está en eje...	Error (10/04/20...	(No están disponib...	
LUZ (Windows NT)	SI	SI	SI	No está en eje...	Desconocido (17/04/2005 0:0...		
Bases de datos	SI	SI	SI	No está en eje...	Desconocido (17/04/2005 1:0...		
Servicios de transformación de dat...	SI	SI	SI	No está en eje...	Desconocido (17/04/2005 2:0...		
Administración	SI	SI	SI	No está en eje...	Correcto (13/04...	(No están disponib...	
Habilitar trabajo	SI	SI	SI	No está en eje...	Correcto (13/04...	(No están disponib...	

Ejercicio 49. Realizar periódicamente una consulta que me almacene en una tabla temporal el resultado de la misma.

8.5 Uso del Registro de Errores

En el explorador de objetos de nuestra consola de administración podemos ver en Administración el registro de sucesos de SQL Server 2005.

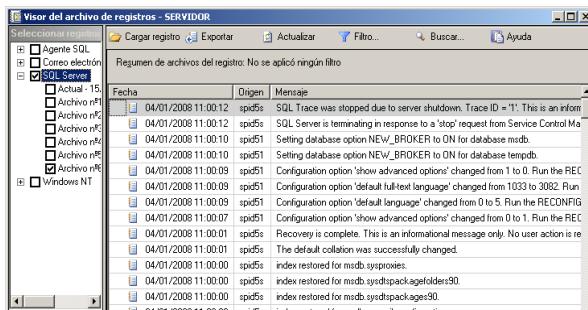


Imagen 76. Visor de registro de SQL Server 2005

8.6 Configuración de Alertas

Una alerta indica al operador designado que ha ocurrido un suceso. Cuando el Agente SQL Server compara los detalles del suceso con las alertas que ha definido el administrador de SQL Server, si encuentra una coincidencia, realizará la respuesta definida. Se pueden definir alertas para notificar a operadores un suceso, ejecutar un trabajo y reenviar el suceso al registro de aplicación de Windows en otro servidor.

SQL Server Management Studio proporciona una forma gráfica y fácil de administrar todo el sistema de alertas, y es la forma recomendada para configurar una infraestructura de alertas. Si una alerta no funciona correctamente deberíamos comprobar:

- ✓ El Agente SQL Server está en ejecución.
- ✓ El evento ha aparecido en el registro de aplicación de Windows.
- ✓ La alerta está habilitada.
- ✓ Los eventos generados mediante xp_logevent se producen en la base de datos master. Por tanto, xp_logevent no desencadena una alerta a menos que el valor de @database_name para la alerta sea 'master' o NULL.

Éstas son las circunstancias en las que los errores y mensajes que generan SQL Server y las aplicaciones de SQL Server se envían al registro de aplicación de Windows y que, por lo tanto, pueden generar alertas:

- ✓ Errores sysmessages de gravedad 19 o superior
- ✓ Cualquier instrucción RAISERROR llamada con la sintaxis WITH LOG.
- ✓ Errores sysmessages modificados o creados mediante sp_altermessage
- ✓ Eventos registrados mediante xp_logevent

Ejemplo 95. Vamos a provocar la generación de una alerta a través del método de RAISERROR.

```
-- RAISERROR with severity 11-19 will cause execution to
-- jump to the CATCH block.
RAISERROR ('Error raised in TRY block.', -- Message text.
           16, -- Severity.
           1 -- State.
```



) WITH LOG;--Muy importante si no no se envía al registro de sucesos

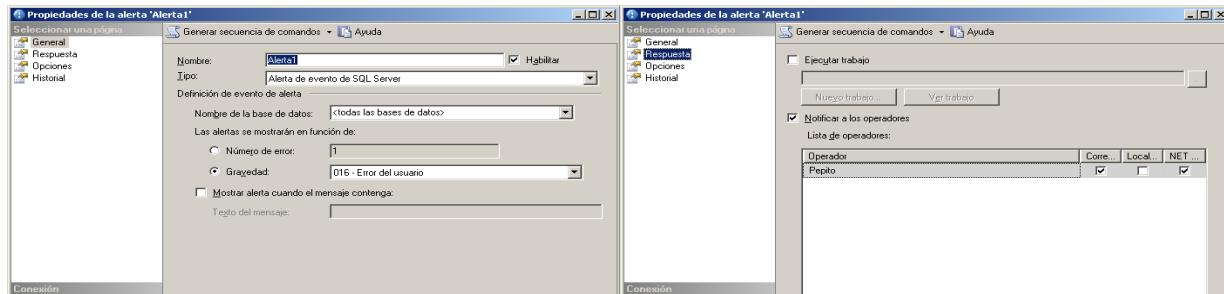
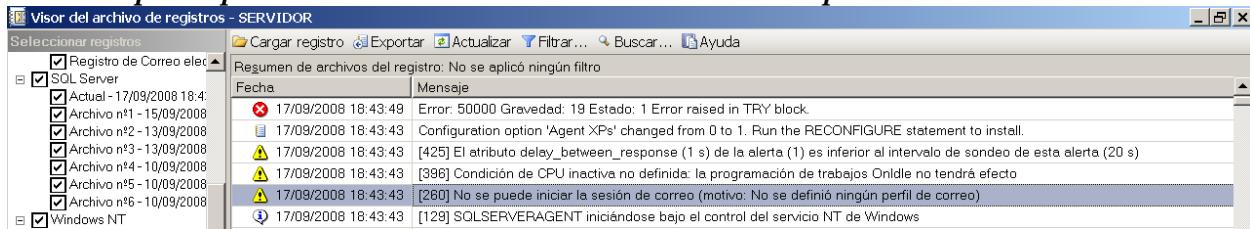
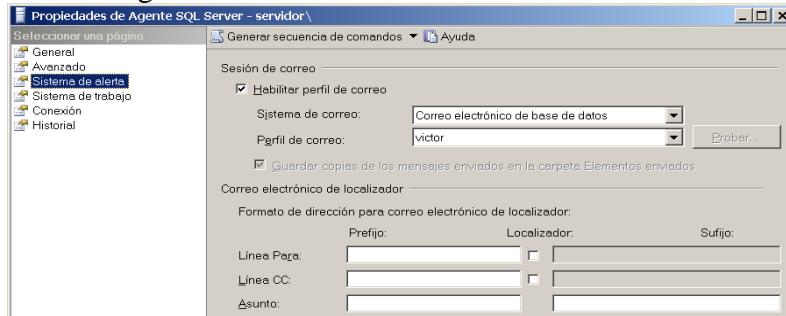


Imagen 77. Definición de una Alerta.

Nota.- Se puede producir un error en el envío de correo en respuesta a la alerta:



En las propiedades del agente debemos habilitar el perfil de correo que usará este agente, posteriormente reiniciar el agente:



Es posible también que debamos habilitar en Outlook que permita enviar a otras aplicaciones en nuestro nombre:

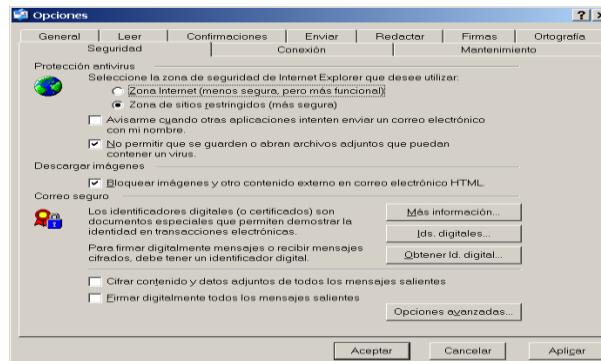


Imagen 78. Deshabilitar avisarme cuando otras aplicaciones intenten enviar un correo electrónico

Eso se soluciona sobre SQL Server SP2 con un hotfix llamado 351279_intl_i386_zip.exe

Ejercicio 50. Queremos definir una alerta para que avise a un operador de que se ha producido un error en la inserción de datos en la tabla emp.

```
INSERT INTO [Prueba 2].[dbo].[emp]([empno], [ename], [job], [mrg], [hiredate], [sal], [comm], [deptno], [salneto])
VALUES(10, 'Prueba de alerta', 'Trabajo', 20, '20/10/05', 30, 40, 100, 110)
```

```
INSERT INTO [Prueba 2].[dbo].[emp] ([empno], [ename], [job], [mrg], [hiredate], [sal], [comm], [deptno], [salneto])
VALUES (10, 'Prueba de alerta', 'Trabajo', 20, '20/10/05', 30, 40, 100, 110)

Servidor: mensaje 8152, nivel 16, estado 4, linea 1
Los datos de cadena o binarios se truncarian.
Se terminó la instrucción.
```

El mensaje⁹ que se envía por red al operador será el siguiente:

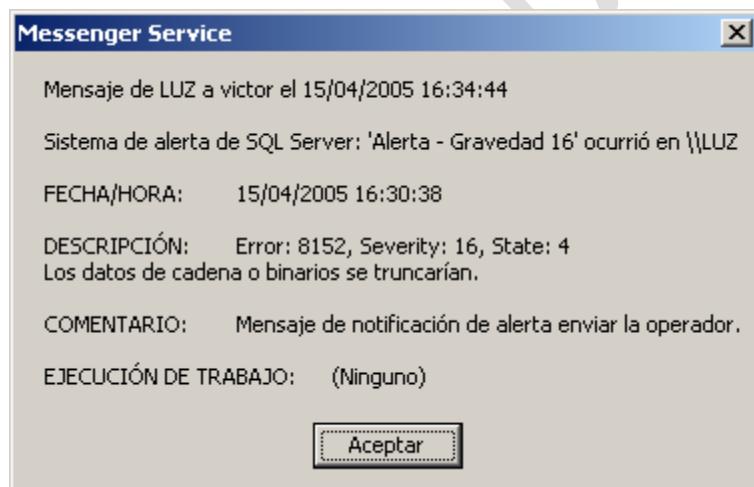


Imagen 79. Notificación de alerta al operador.

En lugar de programar la alerta para que mande un mensaje a un operador podremos hacer que ejecute una trabajo/tarea almacenada.

Ejercicio 51. Crear una alerta para que recoja un error de inserción en una vista de dos tablas.

⁹ Probar a enviar estos mensajes a otra máquina que no sea local, suele no funcionar con local.

Módulo 9 COPIA DE SEGURIDAD, RESTAURACIÓN Y RECUPERACIÓN. BACKUP

“La cantidad de información que una persona está dispuesta a perder en función de lo que eventualmente suceda es la respuesta a contestar para diseñar nuestra política de copias de seguridad.”

Miguel Egea. MVP de portalsql.com

9.1 Introducción

Microsoft SQL Server 2005 proporciona funciones de copia de seguridad y restauración de alto rendimiento. El componente de copia de seguridad y restauración de SQL Server ofrece una protección esencial para los datos críticos almacenados en las bases de datos de SQL Server. La implementación de una estrategia correctamente planeada de copia de seguridad y restauración contribuye a la protección de las bases de datos de la pérdida de datos derivada de daños causados por diferentes errores. La comprobación de la estrategia mediante la restauración de las copias de seguridad y la recuperación de la base de datos permite estar preparado para responder de forma eficaz ante un desastre.

Una copia de los datos que se puede utilizar para restaurar y recuperar los datos se denomina copia de seguridad. Las copias de seguridad le permiten restaurar los datos después de un error. Con las copias de seguridad correctas, puede recuperarse de multitud de errores, por ejemplo:

- ✓ Errores de medios.
- ✓ Errores de usuario, por ejemplo, quitar una tabla por error.
- ✓ Errores de hardware, por ejemplo, una unidad de disco dañada o la pérdida permanente de un servidor.
- ✓ Desastres naturales.

Además, las copias de seguridad de una base de datos son útiles para fines administrativos habituales, como copiar una base de datos de un servidor a otro, configurar la creación de reflejo de la base de datos y el archivo.

Cada modelo de recuperación permite realizar una copia de seguridad completa o parcial de una base de datos de SQL Server, de archivos individuales o de grupos de archivos de la base de datos. No pueden crearse copias de seguridad de las tablas.

9.2 Tipos de copias de seguridad de datos

El ámbito de una copia de seguridad de datos puede ser la base de datos completa, parcial o un conjunto de archivos o grupos de archivos. En todos estos casos, SQL Server admite copias

de seguridad completas y diferenciales:

- ✓ Copia de seguridad completa

Una copia de seguridad completa incluye todos los datos de una base de datos determinada o un conjunto de grupos de archivos, así como una cantidad suficiente del registro como para permitir la recuperación de datos. (Véase *Copia seguridad Completa con Management Studio 9.6.1* pág. 203)

- ✓ Copia de seguridad diferencial

Una copia de seguridad diferencial se basa en la última copia de seguridad completa de los datos. Ésta se denomina base de la copia de seguridad diferencial o base diferencial. Una copia de seguridad diferencial incluye sólo los datos que han cambiado desde la última base diferencial. Por lo general, las copias de seguridad diferenciales que se realizan poco después de su base son más pequeñas y rápidas, ahorrando así tiempo con respecto a la copia de seguridad completa. Por tanto, el uso de copias de seguridad diferenciales acelera el proceso de realización de copias de seguridad frecuentes y reduce el riesgo de pérdida de los datos. Por lo general, una base diferencial se utiliza en varias copias de seguridad diferenciales sucesivas. En el momento de la restauración, se restaura primero la copia de seguridad completa, seguida de la copia de seguridad diferencial más reciente. (Véase *9.6.4 Cómo crear una copia de seguridad diferencial de base de datos* pág. 205)

9.2.1 Copias de seguridad de base de datos, parciales y de archivos.

De los tipos anteriores de copias de seguridad podemos tener tres objetivos de copia de seguridad: de la base de datos, parciales y de archivos.

- ✓ Copias de seguridad de bases de datos. Las copias de seguridad de bases de datos son fáciles de utilizar y se recomienda su uso cuando el tamaño de la base de datos los permite.
- ✓ Copias de seguridad parciales Las copias de seguridad parciales y diferenciales parciales son una novedad en SQL Server 2005. Estas copias de seguridad están diseñadas para ofrecer una mayor flexibilidad al realizar copias de seguridad de bases de datos que incluyen algunos grupos de archivos de sólo lectura con el modelo de recuperación simple. Sin embargo, pueden usarse con todos los modelos de recuperación.
- ✓ Copias de seguridad de archivos. Es posible realizar una copia de seguridad y restaurar individualmente los archivos de una base de datos. El uso de las copias de seguridad de archivos puede aumentar la velocidad de recuperación ya que se pueden restaurar sólo los archivos dañados sin tener que restaurar el resto de la base de datos. Por ejemplo, si una base de datos está compuesta por varios archivos ubicados en diferentes discos y se producen errores en uno de ellos, sólo debe restaurar el archivo situado en el disco en que se produjeron los errores. Sin embargo, la planificación y restauración de copias de seguridad de archivos puede resultar compleja, por lo que sólo deben utilizarse cuando realmente suponen un valor añadido para el plan de restauración.



(Véase 9.6.2 *Cómo realizar copias de seguridad de archivos y grupos de archivos de la base de datos* pág. 204)

9.2.2 Copia de seguridad el registro de transacciones.

Cada copia de seguridad de registros incluye la parte del registro de transacciones que estaba activo al crear la copia de seguridad, además de todas las entradas de registro que no se incluyeron en una copia de seguridad de registros anterior. Antes de crear la primera copia de seguridad de registros, debe crear una copia de seguridad completa, como una copia de seguridad de la base de datos.

(Véase 9.6.3 *Cómo hacer una copia de seguridad del registro de transacciones* pág. 205)

9.3 Restricciones de las operaciones de copia de seguridad en SQL Server

En SQL Server 2005, se puede realizar la copia de seguridad mientras la base de datos está conectada y en uso. Sin embargo, existen las siguientes restricciones.

- ✓ No se pueden realizar copias de seguridad de los datos sin conexión

Cualquier operación de copia de seguridad en la que se haga referencia de forma implícita o explícita a datos sin conexión provocará un error

- ✓ Restricciones de simultaneidad durante una copia de seguridad

SQL Server utiliza el proceso de copia de seguridad con conexión para permitir que se realice la copia de seguridad de una base de datos mientras se está utilizando. Durante la copia de seguridad, se pueden realizar la mayoría de las operaciones (por ejemplo, las instrucciones INSERT, UPDATE o DELETE están permitidas durante la operación de copia de seguridad). Sin embargo, si intenta iniciar una operación de copia de seguridad durante la creación o eliminación de un archivo de la base de datos, la operación de copia de seguridad esperará hasta que la creación o eliminación haya terminado o hasta que se agote el tiempo de espera de la copia de seguridad.

Las operaciones que no se pueden ejecutar durante la copia de seguridad de la base de datos o del registro de transacciones son las siguientes:

- Operaciones de administración de archivos, como la instrucción ALTER DATABASE con la opción ADD FILE o la opción REMOVE FILE.
- Operaciones de reducción de la base de datos o de reducción de un archivo. Esto incluye las operaciones de reducción automática.
- Si intenta crear o eliminar un archivo de la base de datos durante la operación de copia de seguridad, se producirá un error en la operación de creación o eliminación.

9.4 Modelos de recuperación de bases de datos

9.4.1 Modelo de recuperación simple

El modelo de recuperación simple no es adecuado en sistemas de producción en los que es inaceptable la pérdida de cambios recientes. En estos casos, se recomienda el modelo de recuperación completa.

El modelo de recuperación simple proporciona la forma más sencilla de realizar copias de seguridad y restauraciones. La copia de seguridad es fácil de administrar porque nunca se crean copias de seguridad del registro de transacciones. Sin embargo, si no hay copias de seguridad de registros, una base de datos puede restaurarse sólo al final de la copia de seguridad de datos más reciente. Si se produjera un error, se perderían las actualizaciones realizadas después de la copia de seguridad de datos más reciente.

Ejemplo:

La siguiente imagen muestra la estrategia de copia de seguridad y restauración más simple con el modelo de recuperación simple. Existen cinco copias de seguridad completas de la base de datos, pero sólo debe restaurarse la más reciente, realizada a la hora t5. La restauración de esta copia de seguridad devuelve la base de datos a la hora t5. El resto de actualizaciones posteriores, representadas por el cuadro t6, se pierden.

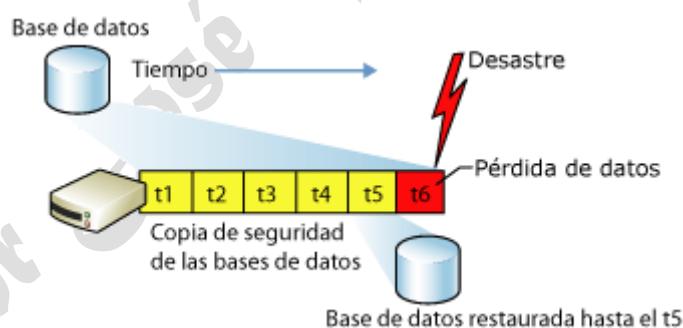


Imagen 80. Ejemplo de copia de seguridad simple

Con el modelo de recuperación simple, el riesgo de pérdida de trabajo aumenta con el tiempo hasta que se realice la siguiente copia de seguridad completa o diferencial. Por tanto, se recomienda programar copias de seguridad con la frecuencia suficiente para evitar la pérdida de un gran volumen de datos sin que las copias de seguridad resulten imposibles de administrar.

La siguiente imagen muestra el riesgo de pérdida del trabajo en un plan de copia de seguridad que sólo utiliza copias de seguridad de bases de datos. Esta estrategia sólo es adecuada para una base de datos pequeña de la que se puede hacer una copia de seguridad con bastante frecuencia.

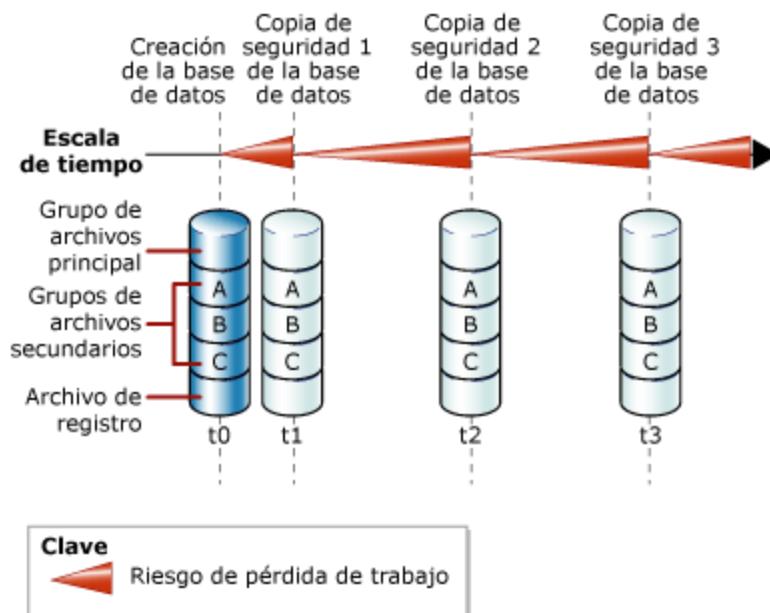


Imagen 81. Estrategia de copia de seguridad en BD pequeña

La siguiente imagen muestra una estrategia de copia de seguridad que reduce el riesgo de pérdida del trabajo complementando copias de seguridad de bases de datos con copias de seguridad diferenciales de bases de datos. Tras la primera copia de seguridad de la base de datos, se realizan tres copias de seguridad diferenciales. La tercera copia de seguridad diferencial es lo suficientemente grande para que la siguiente copia de seguridad sea una copia de seguridad de base de datos. Esto establece una nueva base diferencial.

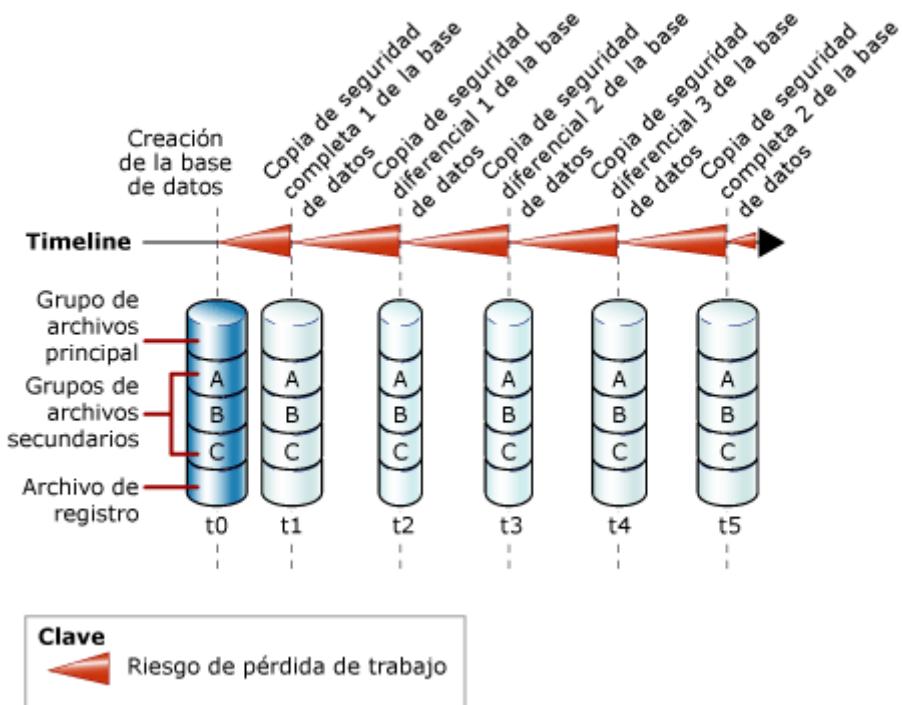


Imagen 82. Copias de seguridad completa junto con diferenciales.

Una secuencia de restauración empieza con la restauración de una copia de seguridad completa, que puede estar seguida, si se desea, de la copia de seguridad diferencial correspondiente. En algunos casos, por ejemplo, al restaurar archivos, es posible que varios pares de copias de seguridad completas y diferenciales requieran su restauración. Después de restaurar las copias de seguridad relevantes, debe recuperar la base de datos.

9.4.2 Modelo de recuperación completo

En el modelo de recuperación completa se usan copias de seguridad de registros para evitar la pérdida de datos en la mayor parte de los casos de error y es necesario realizar copias de seguridad y restaurar el registro de transacciones (copias de seguridad de registros). La ventaja de usar las copias de seguridad de registros reside en que permite restaurar una base de datos a cualquier momento de una copia de seguridad de registros (recuperación a un momento dado). Si consideramos que se puede realizar una copia de seguridad del registro activo después de que ocurre un desastre, se podrá restaurar la base de datos al momento del error sin perder datos. Las desventajas de usar las copias de seguridad de registros son que requieren espacio de almacenamiento y aumentan la duración y la complejidad de las restauraciones.

Ejemplo 96

En la siguiente imagen se muestra la estrategia de copia de seguridad más fácil con el modelo de recuperación completa. En la imagen se han realizado una copia de seguridad de base de datos, Db_1, y dos copias de seguridad de registros rutinarias, Log_1 y Log_2. Algun tiempo después de la copia de seguridad de registros Log_2, se pierden datos de la base de datos. Antes de restaurar estas tres copias de seguridad, el administrador de la base de datos debe realizar una copia de seguridad del registro activo (el final del registro). Entonces, el administrador de la base de datos restaura Db_1, Log_1 y Log_2 sin recuperar la base de datos. A continuación, el administrador de la base de datos restaura y recupera la copia de seguridad de registros después del error (Tail). Así se recupera la base de datos al momento del error, incluidos todos los datos.

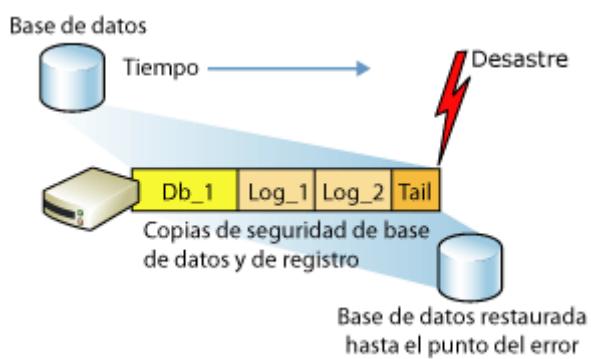


Imagen 83. Estrategia de copia de seguridad del modelo de restauración completa

Una vez que finaliza la primera copia de seguridad completa de la base de datos y se inician las copias de seguridad periódicas de registros, el riesgo potencial de pérdida de trabajo se

limita al tiempo transcurrido entre el momento en que se daña la base de datos y la copia de seguridad periódica de registros más reciente. Por lo tanto, se recomienda que se realicen copias de seguridad de registros con suficiente frecuencia para mantener el riesgo de pérdida de trabajo dentro de los límites establecidos por sus requisitos empresariales.

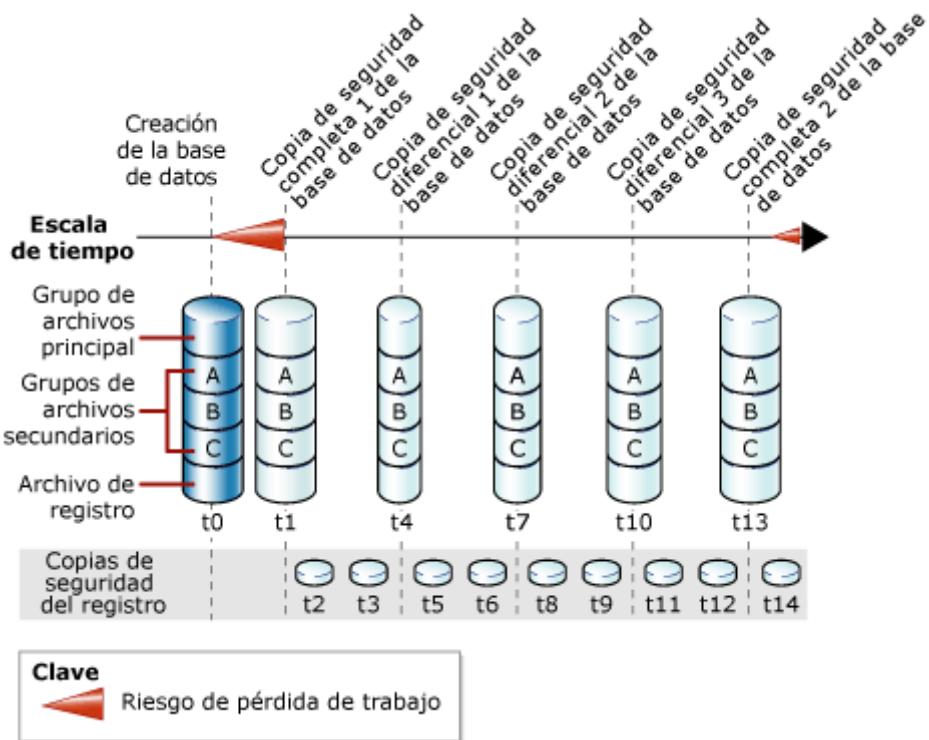


Imagen 84. Riesgo de pérdida de datos en una evolución temporal.

En esta imagen, antes de la primera copia de seguridad de la base de datos, existe un riesgo potencial de pérdida de trabajo en la base de datos (de la hora t0 a la hora t1). Por tanto, las copias de seguridad de registros rutinarias reducen el riesgo de pérdida de trabajo a la posibilidad de perder los cambios realizados después de la última copia de seguridad de registros (realizada a la hora t14 en esta ilustración). Si se produce un error, el administrador de la base de datos debe intentar realizar inmediatamente una copia de seguridad del registro activo (el final del registro). Si esta copia de seguridad de registros después del error se realiza correctamente, la base de datos se puede restaurar hasta el momento del error.

9.4.3 Modelo de recuperación registro masivo

El modelo de recuperación por medio de registros de operaciones masivas es un modelo de recuperación con fines específicos que debe utilizarse sólo de manera intermitente para mejorar el rendimiento de ciertas operaciones masivas a gran escala, como las importaciones masivas de grandes cantidades de datos. La mayor parte de la descripción de la copia de seguridad con el modelo de recuperación completa también se aplica al modelo de recuperación por medio de registros de operaciones masivas.

Es aconsejable minimizar el uso del modelo de recuperación por medio de registros de operaciones masivas. Se recomienda cambiar al modelo de recuperación por medio de registros de operaciones masivas justo antes de un conjunto de operaciones masivas, realizar las operaciones y, a continuación, volver a cambiar al modelo de recuperación completa de inmediato. A diferencia del modelo de recuperación completa, que registra por completo todas las transacciones, el modelo de recuperación por medio de registros de operaciones masivas registra de forma mínima las operaciones masivas, aunque registra totalmente las demás transacciones.

9.4.4 Opciones de recuperación

Restauración de una copia de seguridad completa:

```
USE Master
GO
RESTORE DATABASE MyDB
FROM DISK = 'C:\Backup\MyDB.bak' WITH NORECOVERY
```

Restauración de una copia de seguridad diferencial:

```
USE Master
GO
RESTORE DATABASE MyDB
FROM DISK = 'C:\Backup\MyDB_diff.bak' WITH NORECOVERY
```

Restauración de una copia diferencial del registro de transacciones:

```
USE Master
GO
RESTORE LOG MyDB
FROM DISK = 'C:\Backup\MyDB_log1.trn' WITH NORECOVERY
GO
RESTORE LOG MyDB
FROM DISK = 'C:\Backup\MyDB_log2.trn' WITH NORECOVERY
```

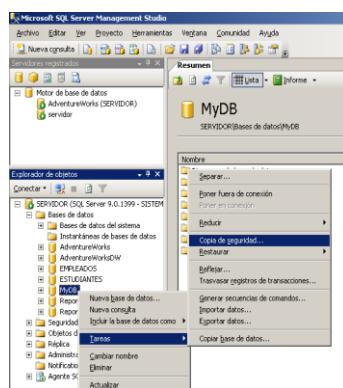
9.5 Resumen de la teoría de copias de seguridad

- ✓ Las copias de seguridad de base de datos completas copia toda la información de la base de datos.
- ✓ Las copias de seguridad diferenciales copian toda la información que ha sido modificada desde la última copia de seguridad completa.
- ✓ Las copias de seguridad del registro de transacciones copian la “parte activa” del registro de transacciones, que es la parte que ha estado activa desde la última copia de seguridad del registro de transacciones.
- ✓ La parte activa del registro de transacciones, la parte que todavía no se ha copiado, se conoce como cola del registro. Debería hacer una copia de seguridad de la cola del registro antes de una operación de restauración siempre que sea posible. Así se puede restaurar la información en el punto en que hubo un fallo en la base de datos.
- ✓ Para comprobar un archivo de copia de seguridad puede utilizar la instrucción RESTORE VERIFYONLY. No obstante, la mejor forma de comprobar una copia de seguridad es realizar una prueba de restauración. Después de restaurar la base de datos, puede ejecutar

la instrucción DBCC para comprobar que la información restaurada no tiene errores.

- ✓ En el modelo de recuperación simple no se pueden realizar copias de seguridad del registro de transacciones. Para restaurar una base de datos configurada en el modelo de recuperación simple en el momento de la última copia de seguridad, restaure primero la última copia de seguridad y a continuación restaure la última copia de seguridad diferencial (si existe alguna) realizada desde la última copia completa. Este modelo de recuperación es adecuado para bases de datos cuya información cambia con muy poca frecuencia.
- ✓ En el modelo de recuperación completa se pueden realizar copias de seguridad del registro de transacciones. Para restaurar una base de datos configurada en el modelo de recuperación completa en el momento de la última copia de seguridad, primero haga una copia de seguridad de la cola del registro, si es posible. A continuación restaure la última copia de seguridad completa y después la última copia de seguridad diferencial (si existe alguna) realizada desde la última copia completa. Por último, aplique todas las copias del registro de transacciones que se han realizado, en orden, desde la última copia completa o diferencial. El modelo de recuperación completa permite recuperar la información hasta el punto de fallo y hasta un punto en el tiempo.
- ✓ El modelo de recuperación de registro masivo se utiliza para mejorar la eficacia durante operaciones de registro masivo. Este modelo se parece al modelo de recuperación completa en que permite que se recupere la información hasta el punto de fallo (pero sólo si el fallo no se produce durante una operación masiva). A diferencia del modelo de recuperación completa, el modelo de recuperación de registro masivo no permite la recuperación hasta un punto en el tiempo.
- ✓ Un plan de recuperación ante desastres es un documento escrito que ayuda a prepararse para manejar los apagones. Un plan de recuperación puede incluir gente con la que contactar en caso de un apagón, un árbol de decisión y una lista de verificación de tareas necesarias en un caso de recuperación y un conjunto de criterios para una recuperación con éxito.

Ejemplo 97. Proceso de copia de seguridad de AdventureWorks



1.-. Creamos una carpeta c:\backup

2.- En la consola de SQL Server ejecutamos la siguiente consulta:

USE AdventureWorks

GO

BACKUP DATABASE AdventureWorks

TO DISK='C:\backup\AWBackup.bak'

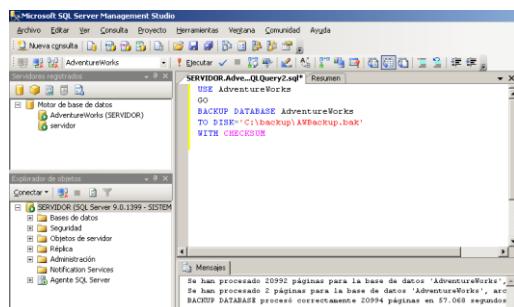
WITH CHECKSUM

-- **WITH INIT** estaríamos haciendo un backup completo.

-- **WITH DIFFERENTIAL** sería una copia diferencial

-- La copia del registro de transacciones sería:

```
BACKUP LOG MyDB
TO DISK = ' C:\Backup\MyDB_log1.bak'
```



3.- Pasemos a comprobar la integridad de la copia de seguridad creada

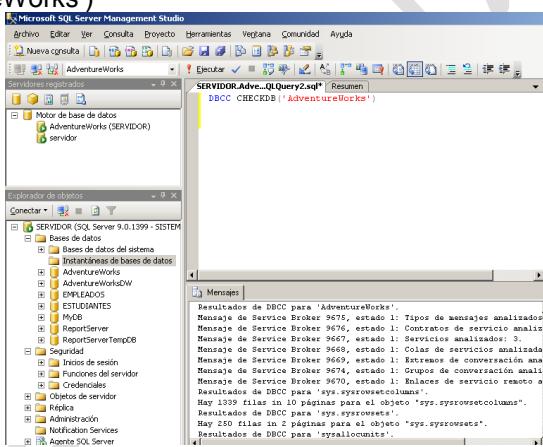
```
RESTORE VERIFYONLY
```

```
FROM DISK='C:\Backup\AWBackup.bak'
```

```
WITH CHECKSUM
```

4.- Después de restaurar una base de datos comprobamos posibles errores con DBCC CHECKDB:

```
DBCC CHECKDB('AdventureWorks')
```



Ejercicio 52. Supuesto de copias de seguridad

En su empresa, la base de datos Pedidos está activa sólo durante horas de trabajo. La base de datos está configurada para el modelo de recuperación completa y se realiza todas las noches una copia de seguridad completa. Las horas laborales son entre las 9:30 y las 16:00 y se realizan copias de seguridad del registro de transacciones todas las horas entre las 10:00 y las 16:00. En una reunión con administradores y analistas, se repasa la directiva de recuperación en vigor y determinan que sólo puede haber una pérdida de datos de 30 minutos, en caso de un desastre. Además, decide que no se deben aplicar más de seis copias del registro de transacciones durante un procedimiento de restauración. ¿Cuál es la mejor forma para cumplir estos requisitos?

- Realizar una copia de seguridad diferencial diaria a las 12:00.
- Realizar una copia de seguridad diferencial diaria a las 13:00.
- Realizar una copia de seguridad del registro de transacciones cada 1/2 hora entre las 10:00 y las 11:30, y entre las 12:30 y las 16:00.
- Realizar una copia de seguridad del registro de transacciones cada 1/2 hora entre las

10:00 y las 11:30, y entre las 13:30 y las 16:00.

Ejercicio 53.

La programación de copia de seguridad de la base de datos Pedidos incluye una copia de seguridad completa una vez a la semana, una copia de seguridad diferencial una vez a la noche, copias de seguridad del registro de transacciones cada hora durante el día. Las copias de seguridad completas que se hacen una vez a la semana tardan 6 horas, las diferenciales 20 minutos y las del registro menos de 5 minutos cada una. Las copias de seguridad se almacenan en una unidad de cinta anexa localmente. Ha determinado que recuperar la base de datos Pedidos le llevará 8 horas.

Si no quiere realizar ninguna copia de seguridad de información durante las horas principales de trabajo entre las 9.00 y las 12.00 y las 13.00 y las 17.00, ¿qué debería hacer para mejorar el tiempo de recuperación de la base de datos Pedidos?

- A. Realizar copias de seguridad completas todas las noches en lugar de diferenciales y hacer las diferenciales a diario a las 12.30.
- B. Realizar copias de seguridad completas a las 12.00.
- C. Cambiar al modelo de recuperación simple y eliminar las copias de seguridad del registro de transacciones programadas.
- D. Realizar una copia de seguridad diferencial durante las horas laborales.

Ejercicio 54

Una base de datos crítica denominada "Información empresarial" está configurada con el modo de recuperación completa. Se lleva a cabo una copia de seguridad completa de la base de datos semanal, una copia de seguridad diferencial a diario y una copia de seguridad del registro de transacciones cada hora durante horas laborales. Si la base de datos Información empresarial fallase y se desconectase, ¿cuál de las siguientes opciones sería el primer paso que realizaría en una secuencia de recuperación ante desastres?

- A. Restaurar la última copia de seguridad completa utilizando la opción NORECOVERY.
- B. Hacer una copia de seguridad de la parte activa del registro con la opción NO_TRUNCATE.
- C. Restaurar la última copia de seguridad diferencial utilizando la opción NORECOVERY.

Ejercicio 55. Recuperación de datos.

1. Tiene que asegurar una recuperación en un punto en el tiempo para una base de datos denominada "Finanzas". ¿En qué modelo de recuperación debe configurar la base de datos Finanzas?

- a) Simple.
- b) De registro masivo.
- c) Completa.
- d) Ninguno.

2. Los requisitos empresariales de una base de datos nueva determinan que debe estar configurada en el modelo de recuperación simple. Ahora tiene que crear un plan de recuperación escrito para esta base de datos. ¿Qué instrucciones debe incluir como parte del documento?

- A. Realizar una copia de seguridad de la parte activa del registro. Aplicar todas las copias de seguridad del registro de transacciones en secuencia desde la última copia de seguridad completa

o diferencial. A continuación restaurar la última copia diferencial, si existe, y por último, restaurar la última copia de seguridad completa.

B. Hacer una copia de seguridad de la parte activa del registro. Restaurar la última copia de seguridad completa. A continuación restaurar la última copia de seguridad diferencial, si existe. Por último, aplicar todas las copias del registro de transacciones en secuencia desde la última copia de seguridad completa o diferencial.

C. Determinar si se han realizado copias de seguridad diferenciales desde la última copia completa. Restaurar la última copia de seguridad completa y a continuación la diferencial, si existe.

D. Determinar si se ha realizado alguna copia de seguridad diferencial desde la última copia completa. Restaurar la copia diferencial, si existe, y a continuación restaurar la última copia de seguridad completa.

3. ¿Cuáles de las siguientes opciones es el primer paso en un procedimiento de recuperación ante desastres para una base de datos configurada en el modelo de recuperación completa en el momento de la última copia de seguridad?

- A. Intentar hacer una copia de seguridad del registro utilizando la opción NO_TRUNCATE.
- B. Aplicar todas las copias del registro de transacciones en secuencia desde la última copia de seguridad diferencial o completa.
- C. Intentar restaurar la última copia de seguridad completa con la opción RECOVERY.
- D. Intentar restaurar la última copia de seguridad completa con la opción NORECOVERY.

4. ¿Cuál de las siguientes opciones representa un posible caso de recuperación de datos para una base de datos configurada en el modo de recuperación de registro masivo en el momento de la última copia de seguridad?

- A. Una tabla nueva se ha eliminado accidentalmente. La información se restaura hasta el punto anterior a la eliminación.
- B. La base de datos y registro se han eliminado por accidente. La información se restaura hasta el punto anterior a la eliminación.
- C. La base de datos falla. La información se restaura hasta el punto de fallo.
- D. La base de datos falla durante una operación de copia masiva. La información se restaura hasta el punto de fallo.

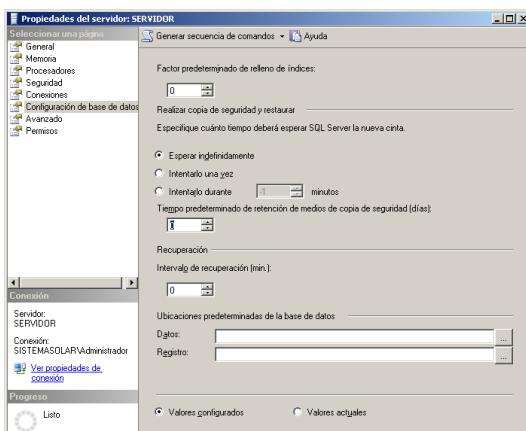
5. Tiene que crear un plan de recuperación para un servidor que está en producción actualmente. ¿Qué debe hacer? (Seleccione todas las opciones válidas.)

- A. Probar la estrategia de restauración restaurando las copias de seguridad en un servidor de prueba.
- B. Mientras realiza el procedimiento de restauración, documente los pasos necesarios para recuperar la base de datos.
- C. Quitarle la conexión al servidor de producción y realizar la restauración.
- D. Compilar una lista de gente a la que contactar en caso de un incidente de pérdida de datos.

9.6 Copias de seguridad en Management Studio

9.6.1 Copia seguridad Completa con Management Studio

1. Clic con el botón secundario en la base de datos, seleccione Tareas y haga clic en Copia de seguridad. Aparece el cuadro de diálogo Copia de seguridad de base de datos, seleccionar completa. Recordar que para hacer diferenciales deberemos tener una completa.
2. En Componente de copia de seguridad, haga clic en Base de datos y colocamos un nombre para la copia de seguridad. Opcionalmente escribimos una descripción del conjunto de copia de seguridad.
3. Indicamos cuando caducará¹⁰ el conjunto de copia de seguridad y se podrá sobrescribir sin omitir explícitamente la comprobación de los datos de caducidad:
 - Para que el conjunto de copia de seguridad caduque al cabo de un número de días específico, haga clic en Después de (opción predeterminada) y escriba el número de días tras la creación del conjunto en que éste caducará. El valor debe estar entre 0 y 99999 días; un valor de 0 días significa que el conjunto de copia de seguridad no caducará nunca.
 - El valor predeterminado se establece en la opción Tiempo predeterminado de retención de medios de copia de seguridad (días) del cuadro de diálogo Propiedades del servidor (página Configuración de base de datos). Para tener acceso a esta opción, en el Explorador de objetos, haga clic con el botón secundario en el nombre del servidor y seleccione Propiedades; a continuación, seleccione la página Configuración de base de datos.



4. Las rutas seleccionadas se muestran en el cuadro de lista Copia de seguridad en.
5. Seleccione una opción de Sobrescribir medios; para ello, haga clic en una de las opciones siguientes:
 - Hacer copia de seguridad en el conjunto de medios existente

¹⁰ Cuando se sobreescreiben conjuntos de copia de seguridad en el medio, la copia de seguridad actual sobrescribe el contenido existente, que deja de estar disponible. Esto sucede si se ha superado el tiempo de caducidad.

- Para esta opción, haga clic en Anexar al conjunto de copia de seguridad existente o Sobrescribir todos los conjuntos de copia de seguridad existentes. Para obtener más información, vea Anexar a conjuntos de copia de seguridad existentes y Sobrescribir conjuntos de copia de seguridad.
- Opcionalmente, seleccione Comprobar nombre de conjunto de medios y fecha de caducidad de conjunto de copia a fin de que la operación de copia de seguridad compruebe la fecha y la hora en que caducan el conjunto de medios y el conjunto de copia de seguridad.
- También puede escribir un nombre en el cuadro de texto Nombre del conjunto de medios. Si no especifica ningún nombre, se creará un conjunto de medios con un nombre en blanco. Si especifica un nombre para el conjunto, los medios (cinta o disco) se comprueban para ver si el nombre real coincide con el nombre especificado aquí.

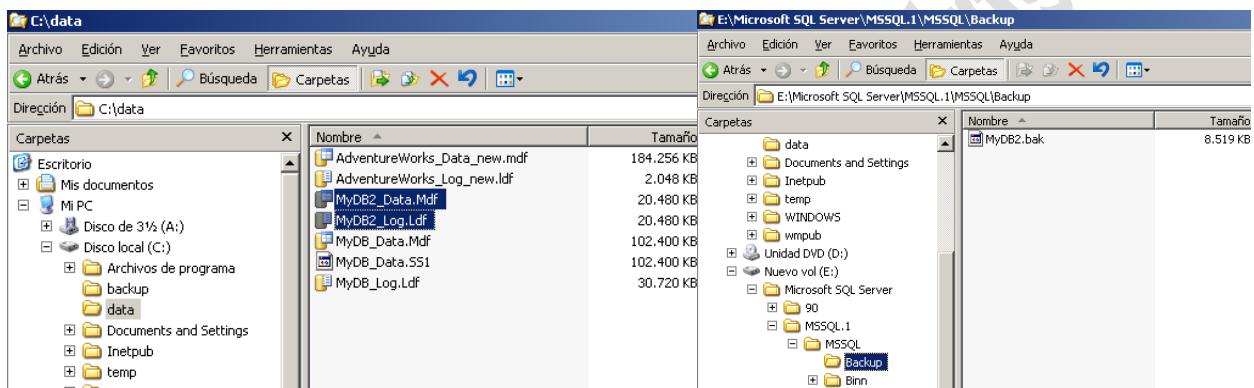
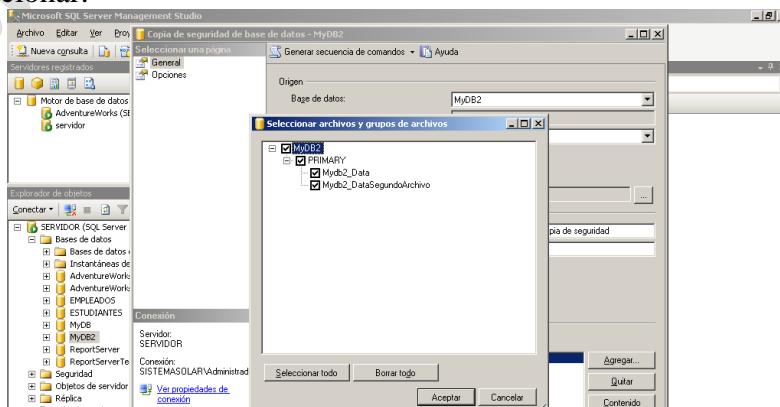


Imagen 85. Rutas de ubicación de los archivos de copia de seguridad

9.6.2 Cómo realizar copias de seguridad de archivos y grupos de archivos de la base de datos

1. Clic con el botón secundario en la base de datos, seleccione Tareas y haga clic en Copia de seguridad. Aparece el cuadro de diálogo Copia de seguridad de base de datos, seleccionar completa o diferencial.
2. En la opción Componente de copia de seguridad, haga clic en Archivo y grupos de archivos. Seleccionar.



9.6.3 Cómo hacer una copia de seguridad del registro de transacciones

1. Clic con el botón secundario en la base de datos, seleccione Tareas y haga clic en Copia de seguridad. Aparece el cuadro de diálogo Copia de seguridad de base de datos, seleccionar completa o diferencial.
2. En el cuadro de lista Tipo de copia de seguridad, seleccione Registro de transacciones.
3. En la sección Registro de transacciones:
 - Para las copias de seguridad rutinarias del registro, conserve la selección predeterminada Truncar el registro de transacciones quitando las entradas inactivas.
 - Para hacer una copia de seguridad del final del registro (es decir, del registro activo), active Realizar copia de seguridad del final del registro y dejar la base de datos en estado de restauración. Una copia de seguridad de registros después del error se lleva a cabo cuando no se consigue realizar la copia de seguridad del final de registro para impedir la pérdida de trabajo. Realice una copia de seguridad del registro activo (copia de seguridad de registros después del error) después de un error, antes de comenzar la restauración de la base de datos o cuando se conmuta por error a una base de datos secundaria. Esta opción equivale a especificar la opción NORECOVERY en la instrucción BACKUP LOG de Transact-SQL. Para obtener más información acerca de las copias de seguridad de registros después del error, vea Copias de seguridad de registros después del error.

9.6.4 Cómo crear una copia de seguridad diferencial de base de datos

1. Clic con el botón secundario en la base de datos, seleccione Tareas y haga clic en Copia de seguridad. Aparece el cuadro de diálogo Copia de seguridad de base de datos, seleccionar completa o diferencial.
2. En el cuadro de lista Tipo de copia de seguridad, seleccione Diferencial.

De esta manera la copia de seguridad se ha ido acumulando en un único fichero:

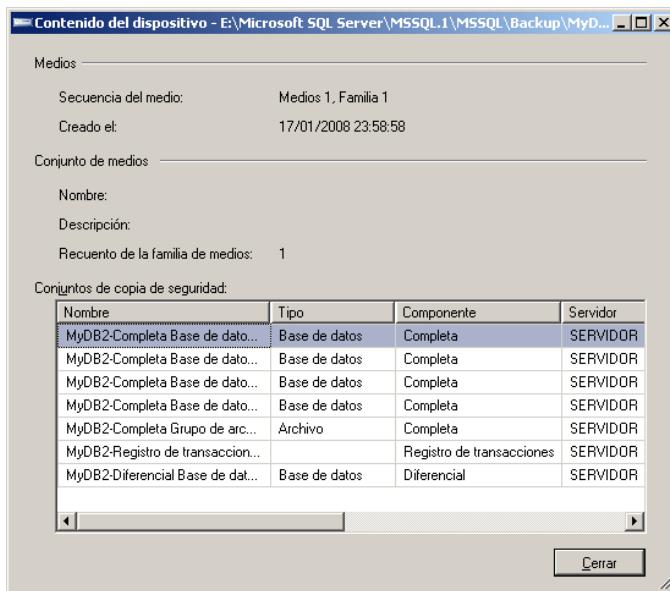
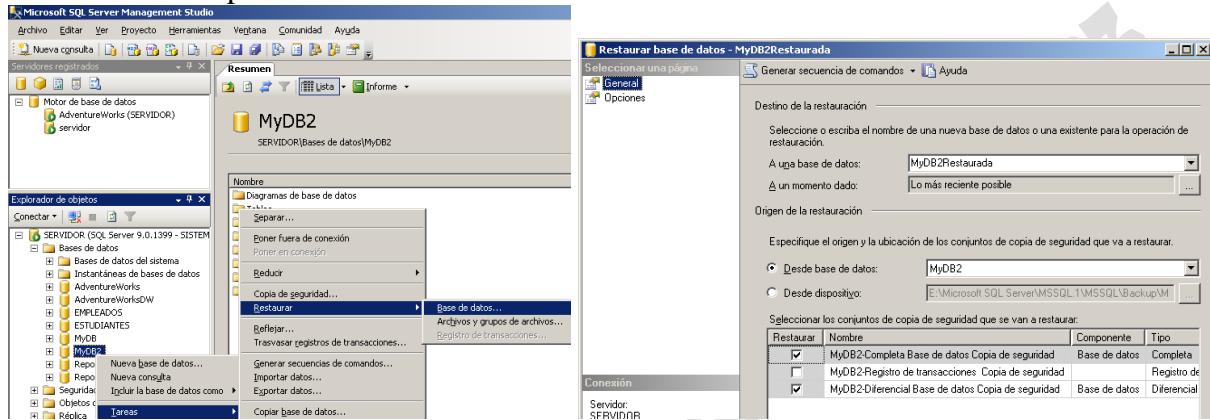


Imagen 86. Contenido del fichero de la copia de seguridad.

9.7 Restauración de una copia de seguridad de base de datos

1. Clic con el botón secundario en la base de datos, seleccione Tareas y haga clic en Copia de seguridad. Aparece el cuadro de diálogo Copia de seguridad de base de datos, seleccionar completa o diferencial.



2. En el cuadro de texto A un momento dado, puede conservar el valor predeterminado (Lo más reciente posible) o seleccionar una fecha y hora determinados haciendo clic en el botón Examinar, que abrirá el cuadro de diálogo Restauración a un momento dado.
3. En el panel Opciones de restauración, puede elegir cualquiera de las siguientes opciones, si son convenientes a su situación:
 - Sobrescribir la base de datos existente
 - Conservar la configuración de réplica
 - Preguntar antes de restaurar cada copia de seguridad
 - Restringir el acceso a la base de datos restaurada

Notas a tener en cuenta al restaurar una copia de seguridad:

- La BD no puede tener instantáneas
- La BD tiene que tener conexiones activas.

Módulo 10 MANTENIMIENTO DE LA BASE DE DATOS

10.1 Reducir una base de datos/archivos

En SQL Server 2005, pueden reducirse todos los archivos de una base de datos para quitar las páginas que no se utilizan. Aunque Database Engine (Motor de base de datos) aprovechará el espacio de manera efectiva, existen ocasiones en las que un archivo no tiene por qué ser tan grande como lo era anteriormente. En estos casos, la reducción del archivo puede ser necesaria. Pueden reducirse los archivos de datos y los archivos de registro de transacciones. Los archivos de la base de datos se pueden reducir manualmente, en grupo o de uno en uno; también se puede configurar la base de datos para que se reduzca automáticamente a intervalos determinados.

Los archivos se reducen siempre desde el final. Por ejemplo, si tiene un archivo de 5 GB y especifica 4 GB como target_size en una instrucción DBCC SHRINKDB, Database Engine (Motor de base de datos) liberará tanto espacio como pueda a partir del último GB del archivo. Si hay páginas utilizadas en la parte del archivo que se va a liberar, Database Engine (Motor de base de datos) reasigna primero las páginas a la parte que se conserva. Sólo se puede reducir una base de datos hasta el punto en el que no le quede espacio disponible. Por ejemplo, si una base de datos de 5 GB tiene 4 GB de datos y especifica 3 GB como target_size de una instrucción DBCC SHRINKDATABASE, sólo se liberará 1 GB.

10.1.1 Reducir automáticamente:

Por defecto una base de datos tiene deshabilitada la opción de reducir automáticamente el tamaño de sus archivos. Podemos cambiar esto en las propiedades de la BD. Esta actividad se lleva a cabo en segundo plano y no afecta a las operaciones que realiza el usuario con la base de datos.

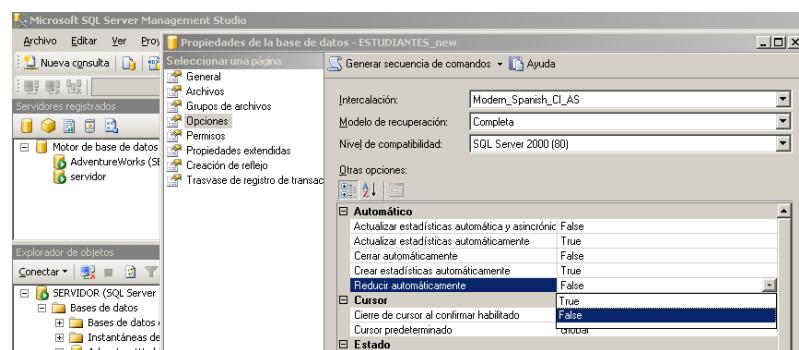


Imagen 87. Propiedades de la BD

10.1.2 Reducir bajo solicitud:

Se puede reducir manualmente una base de datos o los archivos de la base de datos mediante la instrucción DBCC SHRINKDATABASE o la instrucción DBCC SHRINKFILE. Al utilizar la instrucción DBCC SHRINKDATABASE, no es posible reducir toda una base de datos inferior a su tamaño original. Por lo tanto, si se creó una base de datos con un tamaño de 10 MB y ha crecido hasta llegar a 100 MB, sólo podrá reducirla hasta un tamaño de 10 MB, aunque todos los datos de la base de datos se hayan eliminado.

No obstante, se puede reducir uno a uno los archivos de la base de datos por debajo de su tamaño inicial con la instrucción DBCC SHRINKFILE. Por lo tanto, deberá reducir cada archivo individualmente, en vez de intentar reducir toda la base de datos.

Nota.- No se puede reducir la base de datos o el registro de transacciones mientras se realiza una copia de seguridad de la base de datos o del registro de transacciones. Del mismo modo, no se puede crear una copia de seguridad de la base de datos o del registro de transacciones mientras se reduce la base de datos o el registro de transacciones.

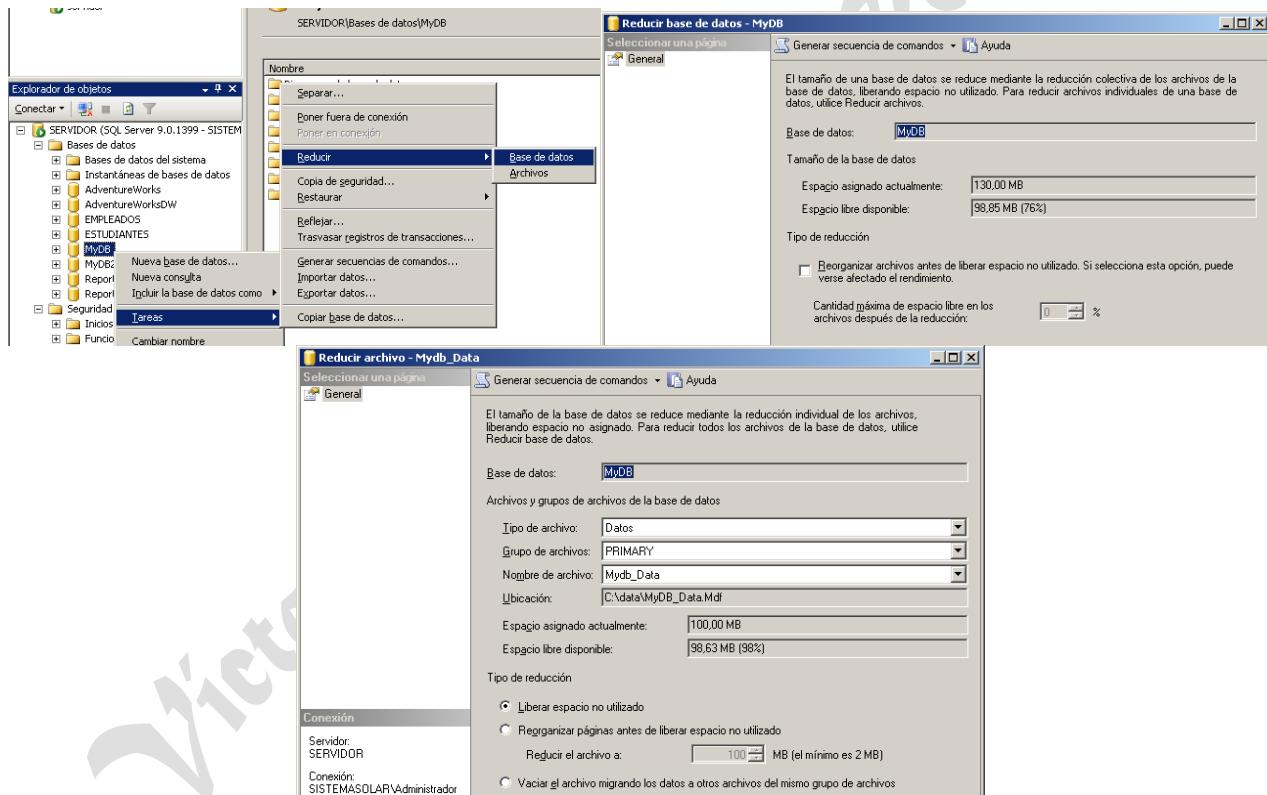


Imagen 88. Proceso de reducción de una BD

10.1.3 Reducir el registro de transacciones

Hay límites fijos para la reducción del archivo de registro de transacciones. El tamaño de los archivos de registro virtuales determina las posibles reducciones de tamaño. En consecuencia, no se puede reducir el archivo de registro a un tamaño inferior que el archivo de registro virtual. Además, el archivo de registro se reduce en incrementos iguales al tamaño del archivo de registro virtual. Por ejemplo, un archivo de registro de transacciones de 1 GB puede estar com-

puesto por cinco archivos de registro virtuales de 200 MB cada uno. Si reduce el archivo de registro de transacciones, se eliminarán los archivos de registro virtuales no utilizados; no obstante, quedarán como mínimo dos archivos de registro virtuales. Dado que cada archivo de registro virtual del ejemplo es de 200 MB, el registro de transacciones podrá reducirse hasta un tamaño mínimo de 200 MB, y sólo en incrementos de 200 MB. Para que un archivo de registro de transacciones pueda reducirse a un tamaño inferior, cree un registro de transacciones pequeño y permita que crezca automáticamente, en vez de crear un archivo de registro de transacciones de gran tamaño.

En SQL Server 2005, una operación DBCC SHRINKDATABASE o DBCC SHRINKFILE intenta inmediatamente reducir un archivo de registro de transacciones al tamaño solicitado (sujeto a redondeo). Debería realizar una copia de seguridad del archivo de registro antes de reducir el archivo, para reducir el tamaño del registro lógico y marcar como inactivos los registros virtuales que no contienen parte alguna del registro lógico.

Desde la consola de SQL Server:

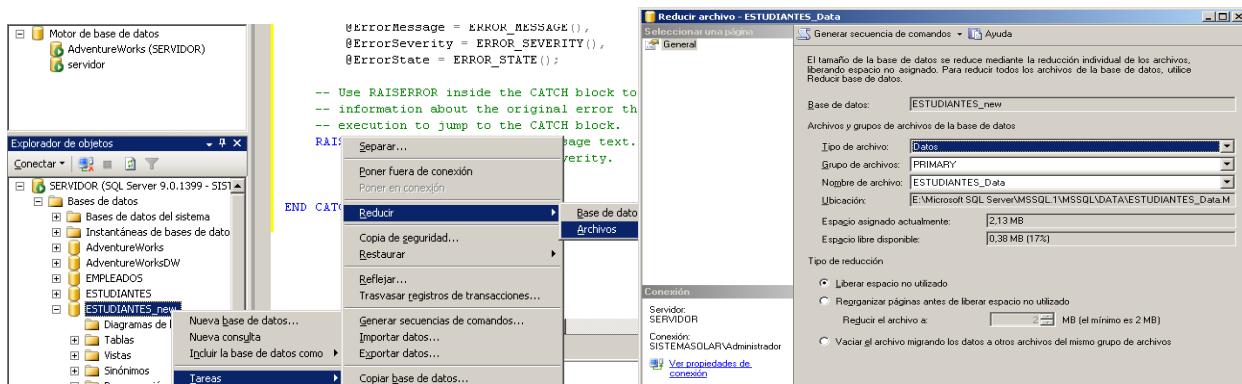


Imagen 89. Enlace a reducción de archivos en la BD

Seleccione el tipo y el nombre del archivo. También puede activar la casilla de verificación Liberar espacio no utilizado. Si activa esta opción, el espacio no utilizado del archivo se libera al sistema operativo y el archivo se reduce a la última extensión asignada. De esta forma, se reduce el tamaño del archivo sin necesidad de mover datos.

También puede seleccionar la casilla de verificación Reorganizar archivos antes de liberar espacio no utilizado. Si activa esta opción, debe especificar el valor Reducir el archivo a. De forma predeterminada, esta opción no está activada. Si activa esta opción, el espacio no utilizado del archivo se libera al sistema operativo y se intentan reubicar las filas en páginas no asignadas.

10.2 Configuración de superficie

La reducción de la superficie es una medida de seguridad que implica detener o deshabilitar componentes no utilizados. La reducción de la superficie ayuda a mejorar la seguridad al proporcionar menos accesos para ataques potenciales al sistema. En nuevas instalaciones de Microsoft SQL Server 2005, se deshabilitan o detienen algunas características, servicios y conexio-

nes para reducir la superficie de SQL Server.

Use la opción Configuración de superficie de SQL Server para habilitar, deshabilitar, iniciar o detener las características, servicios y la conectividad remota de las instalaciones de SQL Server 2005. Puede utilizar la Configuración de superficie de SQL Server en servidores locales y remotos.

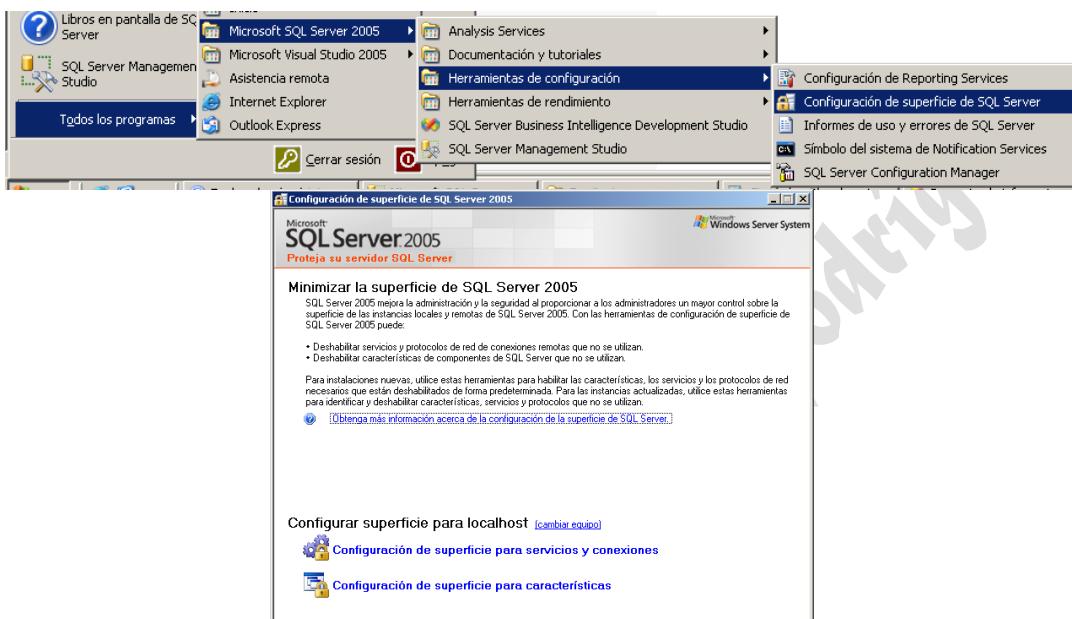
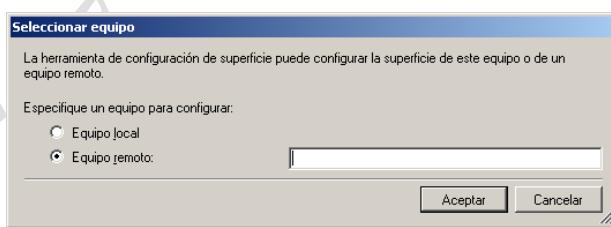


Imagen 90. Configuración de superficie

Como podemos ver en la Imagen 90 podemos pulsar en (cambiar equipo) para poder así modificar la configuración de su superficie de otro equipo de la red.



La Configuración de superficie para servicios y conexiones será para habilitar o deshabilitar servicios de Windows y conectividad remota. Concretamente: Servicio del Database Engine (Motor de base de datos), Conexiones remotas del Database Engine (Motor de base de datos), Servicio de Analysis Services, Conexiones remotas de Analysis Services, Servicio de Reporting Services, Servicio del Agente SQL Server, Servicio de búsqueda de texto, Servicios de instancia de Notification Services, Servicio de Integration Services y Servicio Explorador de SQL Server.

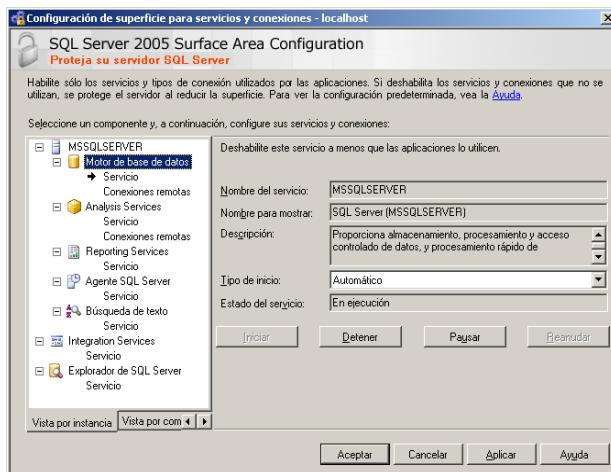


Imagen 91. Configuración de superficie de servicios y conexiones

La Configuración de superficie para características para habilitar y deshabilitar características de Database Engine (Motor de base de datos), Analysis Services y Reporting Services.



Imagen 92. Configuración de superficie para características

<http://msdn2.microsoft.com/es-es/library/ms183753.aspx>

10.3 Plan de mantenimiento

Un plan de mantenimiento es una poderosa herramienta que permite planificar las principales acciones de mantenimiento de la base de datos (organización de los índices, realización de copias de seguridad, comprobar su integridad, etc.)

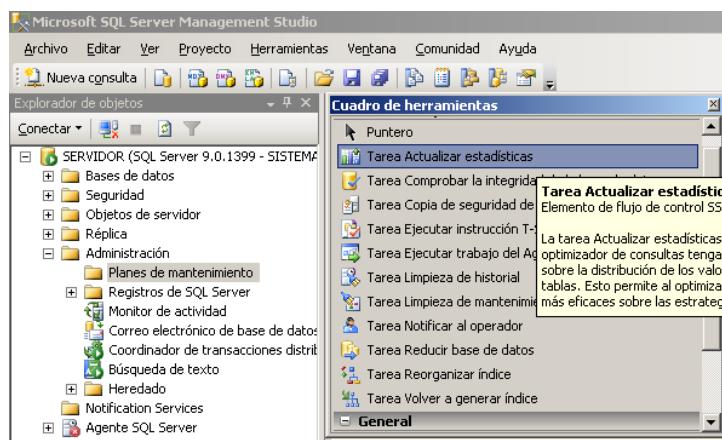


Imagen 93. Plan de mantenimiento de la base de datos

Basta con arrastrar la opción deseada y crear el plan específico:

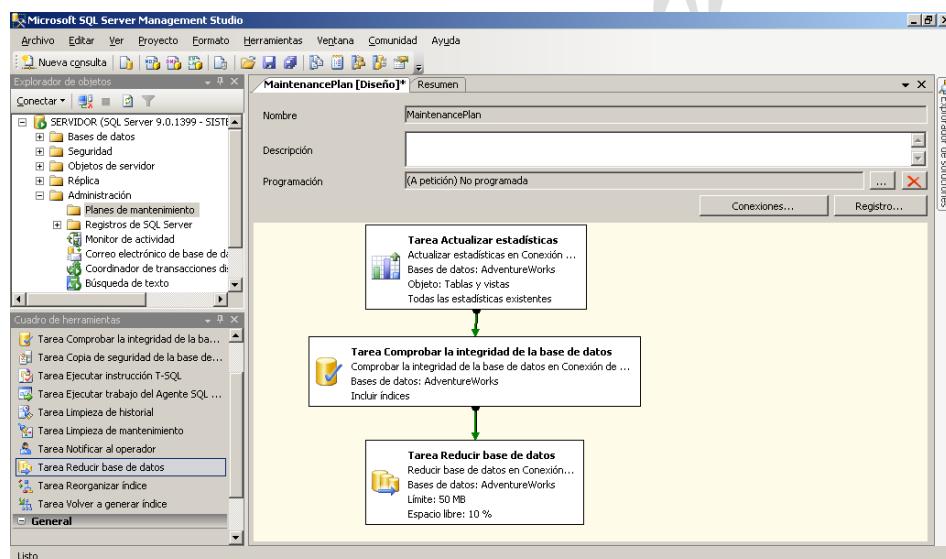


Imagen 94. Ejemplo de plan de mantenimiento de AdventureWorks

Las diferentes tareas a realizar son las siguientes:

- ✓ Utilice el cuadro de diálogo **Tarea Copia de seguridad de base de datos** para agregar una tarea de copia de seguridad al plan de mantenimiento. Es importante realizar una copia de seguridad de la base de datos por si se produce un error de sistema o del hardware (o un error del usuario) que cause algún tipo de daño en la base de datos y que requiera una copia de seguridad para la restauración. Esta tarea le permite realizar copias de seguridad completas, diferenciales, de archivos y grupos de archivos, así como de registros de transacciones.
- ✓ Utilice el cuadro de diálogo **Tarea Comprobar la integridad de la base de datos** para comprobar la asignación e integridad estructural de las tablas de usuario y del sistema, y

los índices de la base de datos por medio de la ejecución de la instrucción DBCC CHECKDB de Transact-SQL. La ejecución de DBCC garantiza que se notifiquen todos los problemas de integridad que puedan existir en la base de datos, lo que permitirá su tratamiento posterior por parte de un administrador del sistema o del propietario de la base de datos.

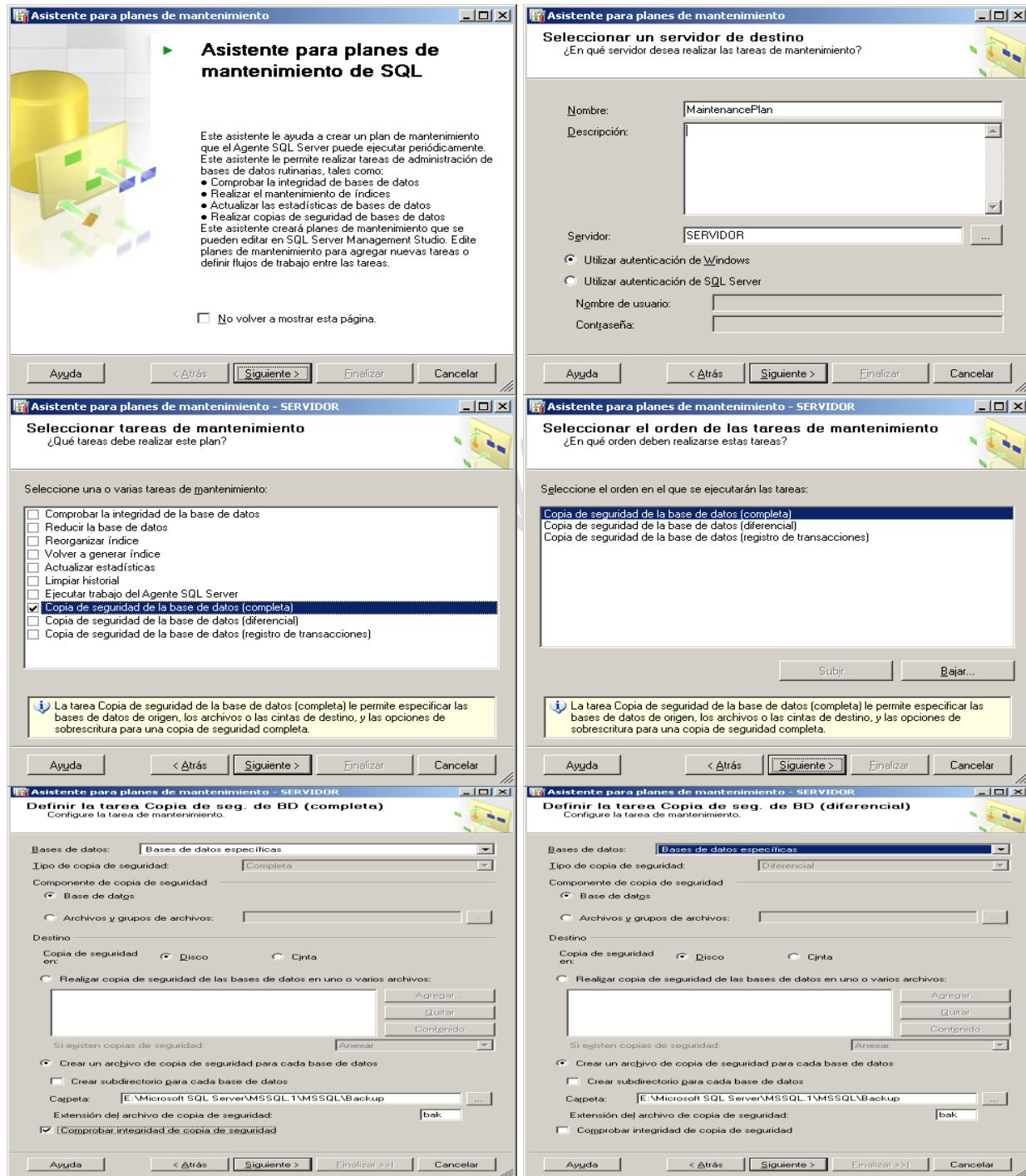
- ✓ Utilice el cuadro de diálogo **Tarea Reorganizar índice** para mover las páginas del índice en un orden de búsqueda más eficaz. Esta tarea utiliza la instrucción ALTER INDEX REORGANIZE con las bases de datos de Microsoft SQL Server 2005 y DBCC INDEXDEFRAG con las bases de datos de Microsoft SQL Server 2000.
- ✓ Utilice el cuadro de diálogo **Tarea Actualizar estadísticas** para actualizar la información de Microsoft SQL Server sobre los datos de tablas e índices. Esta tarea vuelve a muestrear las estadísticas de distribución de cada índice creado en las tablas de usuario de la base de datos. SQL Server utiliza las estadísticas de distribución para optimizar la explotación de las tablas durante el procesamiento de instrucciones Transact-SQL. Para generar automáticamente las estadísticas de distribución, SQL Server muestrea periódicamente los datos de la tabla correspondiente para cada índice. Este tamaño de la muestra se basa en el número de filas de la tabla y en la frecuencia de modificación de los datos. Utilice esta opción para realizar un muestreo adicional con el porcentaje especificado de datos de las tablas. SQL Server utiliza esta información para crear mejores planes de consultas.
- ✓ Utilice el cuadro de diálogo **Tarea Reducir base de datos** para crear una tarea que intente reducir el tamaño de las bases de datos seleccionadas. Utilice las opciones siguientes para determinar el espacio disponible que se mantiene en la base de datos después de reducir su tamaño (cuanto mayor sea el porcentaje, menos se podrá reducir la base de datos). El valor es un porcentaje de los datos actuales de la base de datos. Por ejemplo, una base de datos de 100 MB que contenga 60 MB de datos y 40 MB de espacio disponible, con un porcentaje de espacio disponible del 50 por ciento, dará como resultado 60 MB de datos y 30 MB de espacio disponible (porque el 50 por ciento de 60 MB es 30 MB). Sólo se elimina el espacio de la base de datos que excede el porcentaje indicado. Los valores válidos están comprendidos entre 0 y 100.
- ✓ Utilice el cuadro de diálogo **Tarea Limpieza de historial** para descartar la antigua información histórica de las tablas de la base de datos **msdb**. Esta tarea admite la eliminación y restauración del historial, del historial de trabajos del Agente Microsoft SQL Server y del historial del plan de mantenimiento. No está disponible la eliminación del historial de trabajos del Agente SQL Server en servidores Microsoft SQL Server 2000.
- ✓ Utilice la **Tarea Limpieza de mantenimiento** para eliminar archivos antiguos que estén relacionados con los planes de mantenimiento, incluidos los informes de texto creados por archivos de planes de mantenimiento y copias de seguridad de bases de datos

Como crear un plan de mantenimiento:

<http://msdn2.microsoft.com/es-es/library/ms189953.aspx>

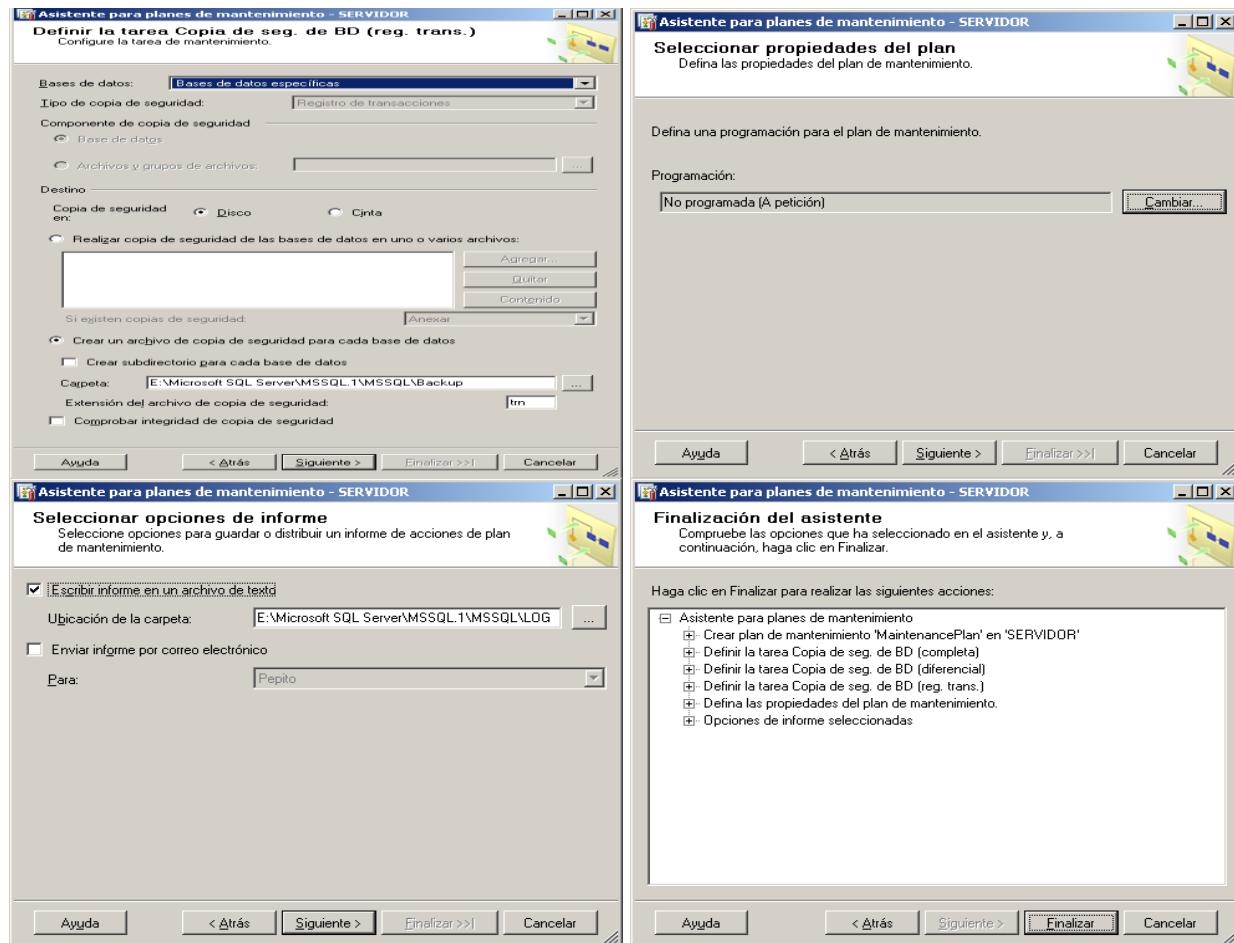
Ejemplo 98. Creación de un plan de mantenimiento con el asistente.

Hay dos maneras de crear un plan de mantenimiento. Puede hacerlo mediante el Asistente para planes de mantenimiento o mediante el uso de una superficie de diseño. El uso del asistente es más conveniente si desea crear planes de mantenimiento básicos, mientras que la superficie de diseño le permite utilizar un flujo de trabajo mejorado. Para crear o administrar planes de mantenimiento, debe ser miembro de la función fija de servidor sysadmin.



The screenshot displays the SQL Server Maintenance Plan Wizard through five sequential windows:

- Asistente para planes de mantenimiento - SERVIDOR**: "Seleccionar tareas de mantenimiento". It lists various maintenance tasks like "Comprobar la integridad de la base de datos" and "Copia de seguridad de la base de datos (completa)". The "Copia de seguridad de la base de datos (completa)" option is selected.
- Asistente para planes de mantenimiento - SERVIDOR**: "Definir la tarea Copia de seg. de BD (completa)". It configures the full backup task, specifying "Bases de datos: Bases de datos específicas", "Tipo de copia de seguridad: Completa", and "Componente de copia de seguridad: Base de datos".
- Asistente para planes de mantenimiento - SERVIDOR**: "Definir la tarea Copia de seg. de BD (diferencial)". It configures the differential backup task, with similar settings to the first step.
- Asistente para planes de mantenimiento**: "Asistente para planes de mantenimiento de SQL". This is the welcome screen for the wizard.
- Asistente para planes de mantenimiento**: "Seleccionar un servidor de destino". It shows the selected server as "SERVIDOR" and the maintenance plan name as "MaintenancePlan".



10.4 Monitor de actividad

El componente Monitor de actividad de Microsoft SQL Server Management Studio sirve para obtener información sobre las conexiones de los usuarios al Database Engine (Motor de base de datos) y los bloqueos que mantienen. El Monitor de actividad tiene tres páginas. La página Información del proceso contiene información acerca de las conexiones. En la página Bloqueos por proceso se ordenan los bloqueos por conexión. En la página Bloqueos por objeto se ordenan los bloqueos por el nombre del objeto.

Para ver el Monitor de actividad, el usuario necesita el permiso SELECT en las tablas **sysprocesses** y **syslocks**, en la base de datos master, en un servidor SQL Server 2005. El permiso para ver estas tablas se concede de forma predeterminada a la función de base de datos **PUBLIC**.

De forma predeterminada, los miembros de las funciones fijas de base de datos sysadmin y processadmin tienen el permiso KILL; este permiso no se puede transferir.

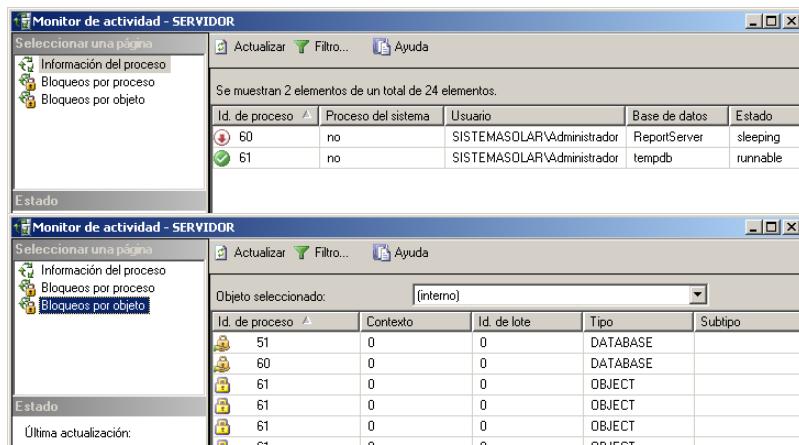


Imagen 95. Monitor de actividad del servidor SQL Server

Existen muchas vistas definidas en el sistema clasificadas en diferentes grupos: Vistas de administración dinámica relacionadas con Common Language Runtime, relacionadas con E/S, relacionadas con la base de datos, con ejecuciones, etc. Algunas de las más importantes son las siguientes:

Vista	Descripción
sys.dm_db_partition_stats	Devuelve información de página y recuento de filas de cada partición en la base de datos actual.
sys.dm_exec_sessions	Devuelve una fila por cada sesión autenticada en SQL Server. sys.dm_exec_sessions es una vista de ámbito de servidor que muestra información acerca de todas las conexiones de usuario activas y las tareas internas. Esta información incluye la versión del cliente, el nombre del programa cliente, la hora de inicio de sesión del cliente, el usuario de inicio de sesión, la configuración de la sesión actual, etcétera.
sys.dm_io_pending_io_requests	Devuelve una fila para cada petición de E/S pendiente de SQL Server.
sys.dm_os_memory_pools	Devuelve una fila para cada almacén de objetos en la instancia de SQL Server. Esta vista se puede utilizar para supervisar el uso de la memoria caché e identificar el comportamiento del almacenamiento en caché incorrecto
sys.dm_os_threads	Devuelve una lista de todos los subprocesos del sistema operativo SQL Server que se están ejecutando en el proceso de SQL Server.
sys.dm_tran_locks	Devuelve información acerca de los recursos del administrador de bloqueos activos actualmente. Cada fila representa una solicitud activa al administrador de bloqueos sobre un bloqueo que se ha concedido o está esperando a ser concedido. Las columnas del conjunto de resultados se dividen en dos grupos principales: recurso y solicitud. El grupo sobre el recurso describe el recurso en que se ha solicitado realizar el bloqueo; el grupo sobre la solicitud describe la solicitud de

bloqueo.

Tabla 4. Principales vistas para la monitorización del sistema

En cuanto a los bloqueos, Microsoft SQL Server 2005 utiliza el bloqueo para asegurar la integridad de las transacciones y la coherencia de las bases de datos. El bloqueo impide que los usuarios lean los datos que otros están modificando y que varios usuarios modifiquen los mismos datos de forma simultánea. Si no se utilizan bloqueos, los datos de la base de datos pueden presentar errores lógicos y las consultas que en ellos se realicen pueden producir resultados inesperados.

Aunque SQL Server impone el bloqueo de forma automática, se pueden diseñar aplicaciones más eficaces si se comprende y se personaliza en éstas el bloqueo.

Ejemplo 99. Creación de un bloqueo de la tabla emp3

```
BEGIN
    WHILE 1<2
        INSERT EMP3 (empno,ename)
        VALUES (10, N'20')
END
```

En otra ventana de conexión a Prueba2:

EXEC master..sp_lock – Obtengo el listado de objetos bloqueados

SELECT object_name(946102411) -- Pido el nombre de objeto bloqueado.

Este no es un bloqueo absoluto, ya que la tabla se bloquea por cada ejecución del insert, por eso puedo realizar un select de la tabla al mismo tiempo que se está ejecutando el bucle.

10.5 Registro de transacciones

Todas las bases de datos de SQL Server 2005 tienen un registro de transacciones que registra todas las transacciones y las modificaciones que cada transacción realiza en la base de datos. El registro de transacciones es un componente esencial de la base de datos y, si se produce un error del sistema, podría ser necesario para volver a poner la base de datos en un estado coherente. El registro de transacciones nunca se debe eliminar o mover, a menos que se conozcan totalmente las implicaciones de esas acciones.

El registro de transacciones permite las siguientes operaciones:

- ✓ Recuperación de transacciones individuales

Si una aplicación emite una instrucción ROLLBACK o si Database Engine (Motor de base de datos) detecta un error, como la pérdida de comunicación con un cliente, los registros se utilizan para revertir todas las modificaciones efectuadas por una transacción incompleta.

- ✓ Recuperación de todas las transacciones incompletas cuando se inicia SQL Server

Si un servidor SQL Server produce errores, las bases de datos pueden quedar en un estado en el que algunas modificaciones no han llegado a escribirse desde la caché del búfer a los archivos de datos; éstos pueden contener modificaciones como resultado de transacciones incompletas. Cuando se inicia una instancia de SQL Server, se ejecuta la recuperación de todas las bases de datos. Todas las modificaciones del registro que no se hayan podido escribir en los archivos de datos se ponen al día. Las transacciones incompletas que se encuentren en el registro de transacciones se revierten para asegurar la integridad de la base de datos.

- ✓ Puesta al día de una base de datos, un archivo, un grupo de archivos o una página restaurados al punto exacto del error

Después de una pérdida de hardware o un error de disco que afecten a los archivos de base de datos, ésta se puede restaurar al punto del error. En primer lugar, restaure la última copia de seguridad completa de base de datos y la última copia de seguridad diferencial de base de datos y, después, restaure la secuencia de copias de seguridad del registro de transacciones al punto del error. Según restaura cada copia de seguridad del registro, Database Engine (Motor de base de datos) vuelve a aplicar todas las modificaciones incluidas en el registro para poner al día todas las transacciones. Cuando se restaura la última copia de seguridad, Database Engine (Motor de base de datos) utiliza la información del registro para revertir todas las transacciones no completadas hasta ese momento.

- ✓ Permitir réplicas transaccionales

El Agente de registro del LOG supervisa el registro de transacciones de cada base de datos configurada para crear réplicas transaccionales y copia las transacciones marcadas para la réplica desde el registro de transacciones a la base de datos de distribución. Para obtener más información, vea Cómo funciona la réplica transaccional.

- ✓ Permitir soluciones de servidor en espera

Las soluciones de servidor en espera, creación de reflejo de la base de datos y trasvase de registros dependen en gran medida del registro de transacciones. En un escenario de trasvase de registros, el servidor principal envía el registro de transacciones activo de la base de datos primaria a uno o varios destinos. Los servidores secundarios restauran el registro en su base de datos secundaria local.

En el registro de transacciones se registran muchos tipos de operaciones. Entre las operaciones se incluyen:

- ✓ El inicio y el final de cada transacción.
- ✓ Todas las modificaciones de los datos (inserción, actualización y eliminación). Esto incluye las modificaciones de las tablas, incluidas las tablas del sistema, hechas por procedimientos almacenados del sistema o por instrucciones del lenguaje de definición de datos (DDL).
- ✓ Las asignaciones o cancelaciones de asignación de páginas y extensiones.

- ✓ La creación o eliminación de una tabla o un índice.

También se registran las operaciones de reversión. Cada transacción reserva espacio en el registro de transacciones para asegurarse de que existe suficiente espacio de registro para admitir una reversión provocada por una instrucción de reversión explícita o cuando se produce un error. La cantidad de espacio reservado depende de las operaciones realizadas en la transacción, pero normalmente equivale a la cantidad de espacio utilizado para cada operación de registro. Este espacio reservado se libera cuando se completa la transacción.

La sección del archivo de registro a partir de la primera entrada de registro que debe estar presente para una reversión correcta en toda la base de datos hasta la última entrada de registro escrita se denomina parte activa del registro o registro activo. Se trata de la sección del registro necesaria para una recuperación completa de la base de datos. No se puede truncar ninguna parte del registro activo.



Imagen 96. Explorador de logs

Para poder acceder al detalle de los logs almacenados podemos utilizar herramientas externas a SQL Server como ApexSQL Log.

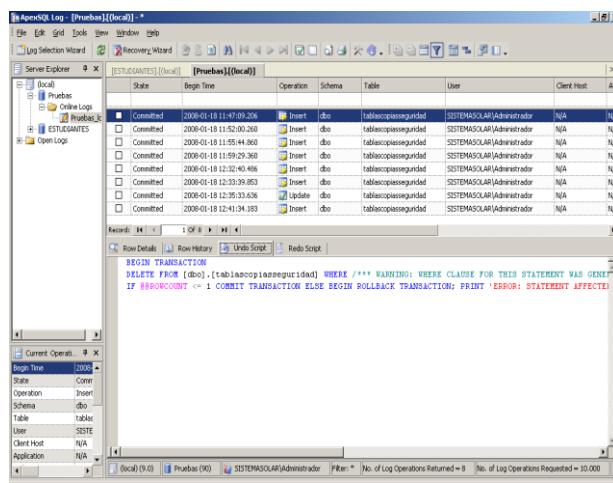


Imagen 97. Aspecto que tiene ApexSQL Log

10.5.1 Cómo detener el crecimiento inesperado del registro de transacciones de una base de datos de SQL Server

El archivo de registro de transacciones está dividido lógicamente en segmentos más pe-

queños que se denominan archivos de registro virtuales. Si el archivo de registro de transacciones que corresponde a una base de datos de SQL Server se llena y ha establecido la opción para que los archivos de registro de transacciones crezcan automáticamente, el archivo de registro de transacciones crecerá en unidades de archivos de registro virtuales. Algunas veces, el archivo de registro de transacciones puede llegar a ser muy grande y puede quedarse sin espacio en disco. Cuando un archivo de registro de transacciones crece hasta que utiliza todo el espacio disponible en el disco y no puede crecer más, ya no puede realizar ninguna operación de modificación de datos en la base de datos. Además, SQL Server puede marcar su base de datos como sospechosa debido a la falta de espacio para la expansión del registro de transacciones.

Para recuperarse de una situación donde los registros de transacciones crecen hasta un límite inaceptable, debe reducir el tamaño de los registros de transacciones. Para ello, debe truncar las transacciones inactivas del registro de transacciones y reducir el tamaño del archivo de registro de transacciones.

Nota.- Los registros de transacciones son muy importantes para mantener la integridad transaccional de la base de datos. Por tanto, no debe eliminar los archivos de registro de transacciones incluso después de realizar una copia de seguridad de la base de datos y de los registros de transacciones.

Cuando los registros de transacciones crecen hasta un límite inaceptable, debe hacer inmediatamente una copia de seguridad del archivo de registro de transacciones. Mientras se crea la copia de seguridad de los archivos de registro de transacciones, SQL Server trunca automáticamente la parte inactiva del registro de transacciones. La parte inactiva del archivo de registro de transacciones contiene las transacciones completadas y, por tanto, SQL Server ya no utiliza el archivo de registro de transacciones durante el proceso de recuperación. SQL Server reutiliza este espacio inactivo truncado del registro de transacciones en lugar de permitir que el registro de transacciones siga creciendo y utilice más espacio.

La operación de copia de seguridad o el método Truncate no reduce el tamaño del archivo de registro. Para reducir el tamaño del archivo de registro de transacciones, debe comprimir el archivo. Para comprimir un archivo de registro de transacciones al tamaño solicitado y quitar las páginas no usadas, debe utilizar la operación DBCC SHRINKFILE. La instrucción DBCC SHRINKFILE de Transact-SQL sólo puede comprimir la parte inactiva del archivo de registro.

Para evitar que los archivos de registro de transacciones crezcan inesperadamente, considere la posibilidad de utilizar uno de los métodos siguientes:

- ✓ Establezca el tamaño de los archivos de registro de transacciones en un valor grande para evitar la expansión automática de dichos archivos.
- ✓ Configure la expansión automática de los archivos de registro de transacciones utilizando unidades de memoria en lugar de un porcentaje después de evaluar detenidamente el tamaño de memoria óptimo.
- ✓ Haga copia de seguridad de los archivos de registro de transacciones periódicamente para eliminar las transacciones inactivas del registro de transacciones.
- ✓ Diseñe las transacciones para que sean pequeñas.
- ✓ Asegúrese de que ninguna transacción no confirmada siga ejecutándose durante un tiempo indefinido.

- ✓ Programe la opción Actualizar estadísticas para que se realice diariamente.

[trans, 2008]

Ejercicio 56. Actualizar todos los datos de una tabla sin clave primaria e intentar volver a restaurarlos a través del ApexSQLLog. Hacerlo posteriormente con una tabla con clave primaria.

10.6 Asistente para la optimización de la BD

El Asistente para la optimización de motor de base de datos es una herramienta nueva de Microsoft SQL Server 2005 que permite optimizar las bases de datos para mejorar el procesamiento de las consultas. El Asistente para la optimización de motor de base de datos analiza la forma en que se procesan las consultas en las bases de datos especificadas por el usuario y, a continuación, recomienda la forma en que se puede mejorar el rendimiento del procesamiento modificando las estructuras de diseño físico tales como índices, vistas indizadas y particiones.

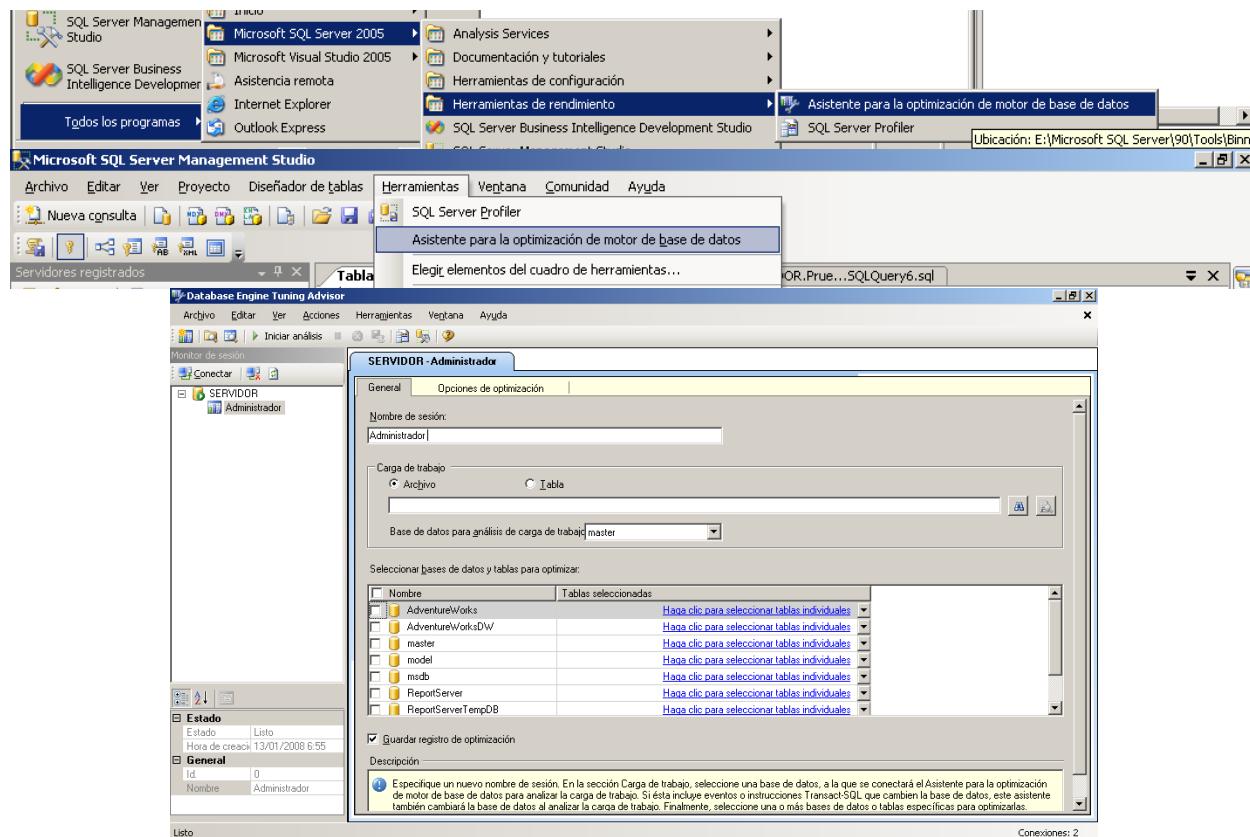


Imagen 98. Asistente para la optimización de la BD

Cuando se abre por primera vez, aparecen dos paneles principales en la GUI del Asistente para la optimización de motor de base de datos.

- ✓ El panel izquierdo contiene el Monitor de sesión, que enumera todas las sesiones de optimización que se han realizado en esta instancia de Microsoft SQL Server. Puede hacer clic con el botón secundario en cualquier sesión para cambiarle el nombre, cerrarla, eliminarla o clonarla. Si hace clic con el botón secundario en la lista, podrá ordenar las sesiones por nombre, estado u hora de creación, o bien crear una sesión nueva. En la sección inferior de este panel, se muestran detalles acerca de la sesión de optimización seleccionada.
- ✓ El panel derecho contiene las fichas General y Opciones de optimización. Aquí es dónde puede definir la sesión de optimización del motor de base de datos. En la ficha General, escriba el nombre de la sesión de optimización, especifique la tabla o el archivo de carga de trabajo que se va a utilizar y seleccione las bases de datos y tablas que desea optimizar en esta sesión. En la ficha Opciones de optimización, puede seleccionar las estructuras de diseño físico de base de datos (índices o vistas indizadas) y la estrategia de partición que desea que el asistente tenga en cuenta durante el análisis. En esta ficha, también puede especificar el tiempo máximo que el Asistente para la optimización de motor de base de datos empleará en optimizar una carga de trabajo. De forma predeterminada, el asistente emplea una hora en optimizar una carga de trabajo.

Ejemplo 100. Optimización de una consulta.

Guardar la siguiente consulta:

```
USE AdventureWorks ;
GO
SELECT *
FROM Production.Product
ORDER BY Name ASC ;
```

En el optimizador nos sugiere la creación de un índice. Esto mejorará el 59% la búsqueda. En el menú de Acciones podremos aplicar las sugerencias dadas por el asistente.

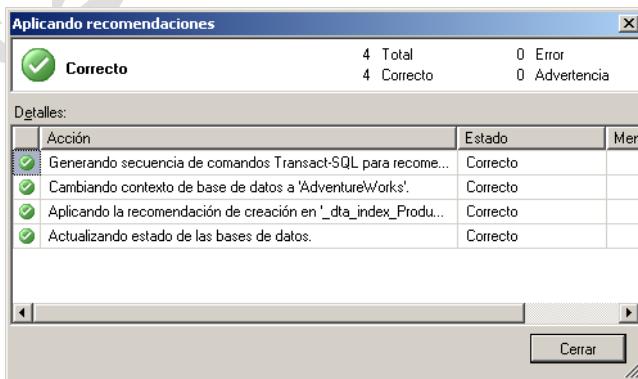


Imagen 99. Ejecución de la recomendación dada por el optimizador de BD.

10.7 Replicación de BD

El administrador del Motor de base de datos es quien suele controlar la administración de la réplica y se encarga de planear y ejecutar las operaciones diarias en áreas como disponibilidad del sistema, supervisión y optimización del rendimiento, implementación, actualizaciones, resolución de problemas y configuración.

La réplica se utiliza con frecuencia en las aplicaciones de almacenamiento de datos e informes para:

- ✓ Consolidar los datos y poder transformarlos y moverlos al entorno de almacenamiento de datos.
- ✓ Distribuir los datos a bases de datos de sólo lectura para los informes.
- ✓ Distribuir datos a una base de datos de proceso analítico en línea (OLAP).

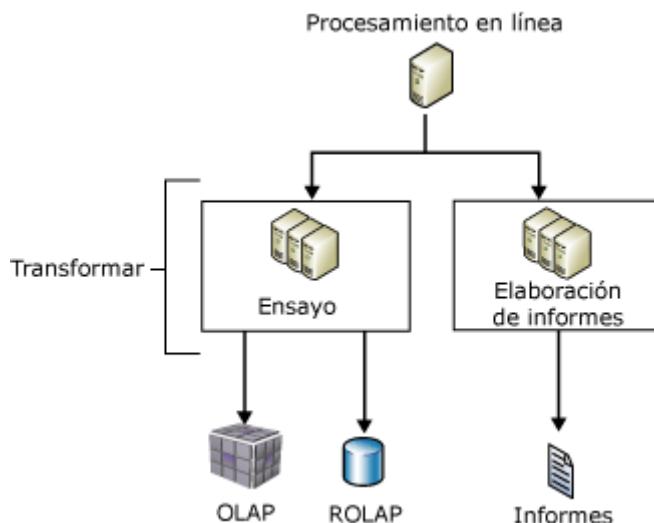


Imagen 100. Esquema ejemplo de una replicación

10.7.1 Componentes de la réplica

Resulta útil pensar en la réplica de Microsoft SQL Server como si fuera una revista:

- ✓ El publicador (editor) de una revista produce una o más publicaciones.
- ✓ Una publicación contiene artículos.
- ✓ El publicador distribuye la revista directamente o a través de un distribuidor.
- ✓ Los suscriptores reciben las publicaciones a las que se han suscrito.

Aunque la metáfora de la revista es útil para comprender la réplica, es importante señalar que la réplica de SQL Server incluye funciones que no están representadas en esta metáfora, en particular, la posibilidad de que un suscriptor realice actualizaciones y de que un publicador envíe cambios incrementales a los artículos de una publicación.

Una topología de réplica define la relación entre los servidores y las copias de los datos, y aclara la lógica que determina cómo fluyen los datos entre los servidores. Hay varios procesos de réplica (denominados agentes) que son responsables de copiar y mover los datos entre el publi-

cador y los suscriptores. En la siguiente imagen se muestra información general acerca de los componentes y procesos que participan en la réplica.

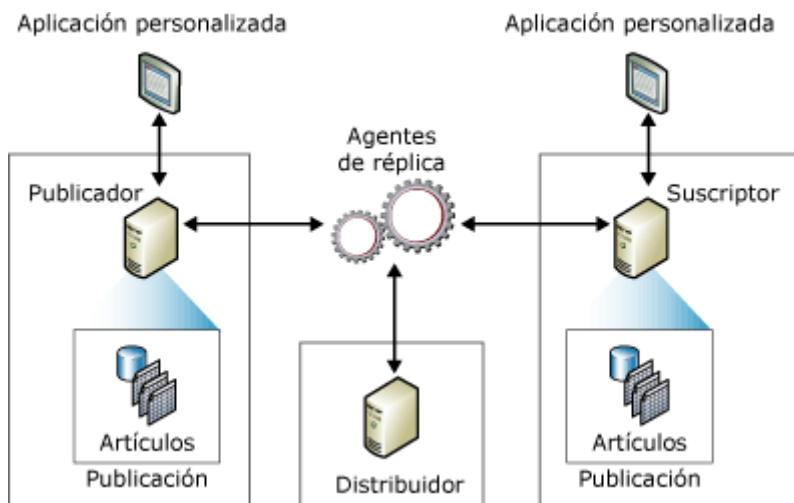


Imagen 101. Componentes de la réplica

✓ **Publicador**

El publicador es una instancia de base de datos que permite que los datos estén disponibles para otras ubicaciones a través de la réplica. El publicador puede tener una o más publicaciones, cada una de las cuales representa un conjunto de datos de objetos relacionados lógicamente para su réplica.

✓ **Distribuidor**

El distribuidor es una instancia de base de datos que funciona como almacén para datos específicos de réplica asociados a uno o varios publicadores. Cada publicador está asociado a una sola base de datos (conocida como la base de datos de distribución) en el distribuidor. La base de datos de distribución almacena los datos de estado de la réplica, los metadatos acerca de la publicación y, en algunos casos, funciona como cola para los datos que se transfieren del publicador a los suscriptores. En muchos casos, una sola instancia de servidor de bases de datos funciona como publicador y como distribuidor. Esto se conoce como un distribuidor local. Cuando el publicador y el distribuidor se configuran en instancias distintas del servidor de bases de datos, el distribuidor se denomina distribuidor remoto.

✓ **Suscriptores**

Un suscriptor es una instancia de base de datos que recibe datos replicados. Un suscriptor puede recibir datos de varios publicadores y publicaciones. En función del tipo de réplica elegida, el suscriptor también puede devolver los datos modificados al publicador o volver a publicar los datos en otros suscriptores.

Importante:

SQL Server Express sólo funciona como suscriptor.

✓ Artículo

Un artículo identifica un objeto de base de datos incluido en una publicación. Una publicación puede contener diferentes tipos de artículos, como tablas, vistas, procedimientos almacenados y otros objetos. Cuando las tablas se publican como artículos, se pueden usar filtros para restringir las columnas y filas de datos que se envían a los suscriptores.

✓ Publicación

Una publicación es un conjunto de uno o más artículos de una base de datos. La agrupación de varios artículos en una publicación permite especificar más fácilmente un conjunto de objetos y datos de bases de datos relacionados lógicamente, que se replican como una unidad.

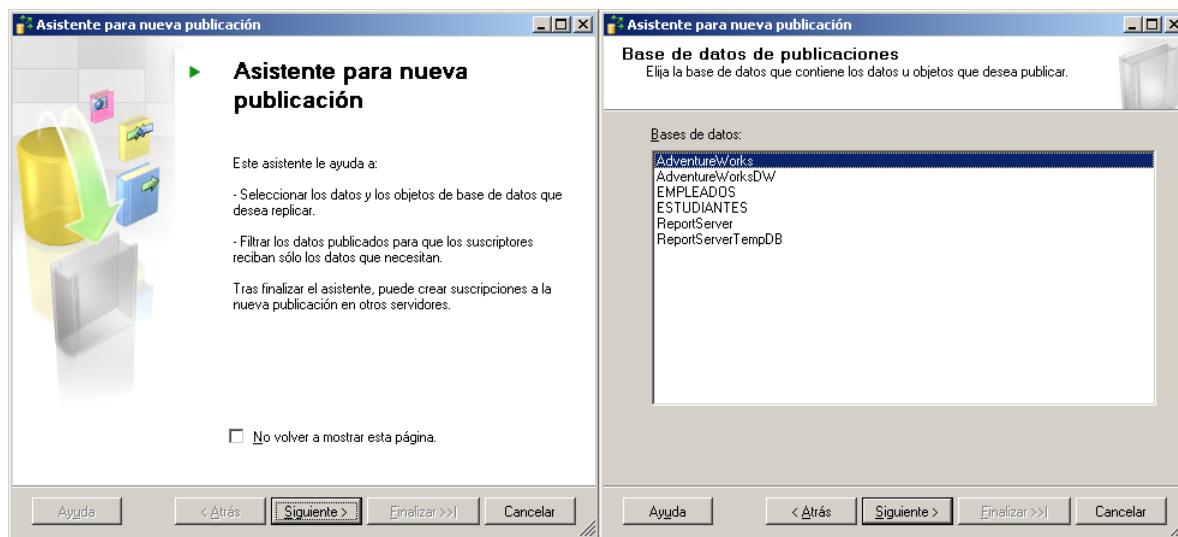
✓ Suscripción

Una suscripción es una solicitud de una copia de una publicación que se entrega a un suscriptor. La suscripción define qué publicación se recibirá, dónde y cuándo. Hay dos tipos de suscripciones: de inserción y de extracción. Para obtener más información acerca de las suscripciones de inserción y de extracción, vea Suscribirse a publicaciones en los Libros en pantalla de SQL Server 2005.

10.7.2 Generación de una réplica

En el árbol del explorador de objetos botón derecho configurar distribución, es probable que debas cerrar las conexiones. Siguiente... siguiente...siguiente....

Creación publicación: Réplica – Nueva – Publicación.





Asistente para nueva publicación

Tipo de publicación
Elija el tipo de publicación que mejor se adapte a los requisitos de su aplicación.

Tipo de publicación:

- Publicación de instantáneas
- Publicación transaccional
- Publicación transaccional con suscripciones actualizables
- Publicación de mezcla

Descripciones de tipos de publicación:

Publicación de instantáneas:
El publicador envía una instantánea de los datos publicados a intervalos programados.

Publicación transaccional:
Transacciones de secuencias del publicador a los suscriptores después de recibir éstos una instantánea inicial de los datos publicados.

Publicación transaccional con suscripciones actualizables:
Transacciones de secuencias del publicador a los suscriptores de SQL Server después de recibir éstos una instantánea inicial de los datos publicados. Las transacciones que se originan en el suscriptor se aplican al publicador.

Objetos para publicar:

- Tablas
- Vistas

Propiedades del artículo

Mostrar sólo los objetos seleccionados en la lista

Ayuda | < Atrás | Siguiente > | Finalizar >> | Cancelar

Asistente para nueva publicación

Problemas de los artículos
Los siguientes problemas pueden requerir cambios en la aplicación para que ésta continúe funcionando correctamente.

Problemas:

- Son necesarias las tablas a las que hacen referencia las vistas.

Descripción:
SQL Server requiere que todas las tablas a las que se hace referencia en vistas publicadas e indexadas estén disponibles en el suscriptor. Si las tablas a las que se hace referencia no están publicadas como artículos en esta publicación, se deben crear manualmente en el suscriptor.

Las siguientes vistas y vistas indizadas están publicadas en esta publicación:

- [dbo].[clientes105]
- [dbo].[clientesespeciales]
- [dbo].[informacion]
- [dbo].[informacionpedidos]
- [dbo].[oficinaeste]
- [dbo].[pedido102]

Filtrar filas de tabla
Agregue filtros para excluir filas no deseadas de las tablas publicadas.

Tablas filtradas:

Haga clic en Siguiente si no necesita filtrar los datos de la publicación.

Haga clic en Agregar para iniciar el filtro de la publicación.

Ayuda | < Atrás | Siguiente > | Finalizar >> | Cancelar

Asistente para nueva publicación

Agente de instantáneas
Especifique cuándo se debe ejecutar el Agente de instantáneas.

Las suscripciones se inicializan con una instantánea de esquema de publicación y datos. El Agente de instantáneas crea la instantánea.

Crear una instantánea inmediatamente y mantenerla disponible para inicializar suscripciones

Programar el Asistente de instantáneas para ejecutarse:

Sucede cada día cada 3 horas entre las 0:00:00 y las 23:59:59. Se utilizará la programación que empieza el 14/01/2008.

Cambiar...

Si planea cambiar las propiedades de la instantánea, no inicie el Agente de instantáneas hasta que haya cambiado las propiedades en el cuadro de diálogo de propiedades de la publicación.

Propiedades de programación del trabajo

Nombre: _____ Trabajos en programación

Tipo de programación: Periódica Habilitado

Repetición una vez

Fecha: 14/01/2008 Hora: 9:41:16

Frecuencia

Sucede: Diario

Se repite cada: 1 días

Frecuencia diaria

Sucede una vez a las: 0:00:00

Sugede cada: 3 horas Empezar: 0:00:00 Finalizar: 23:59:59

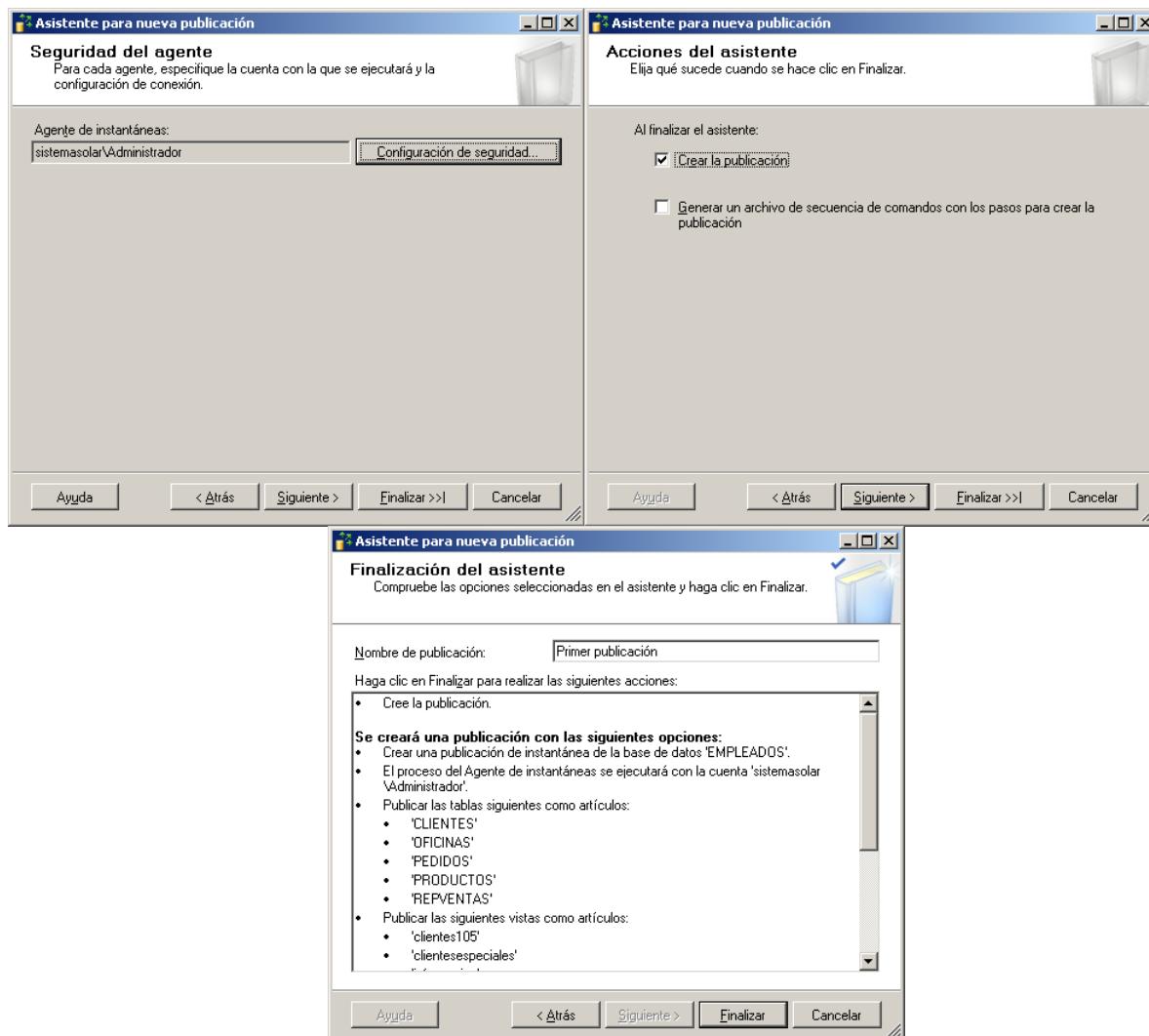
Duración

Fecha de inicio: 14/01/2008 Fecha de finalización: 14/01/2008 Sin fecha de finalización

Resumen

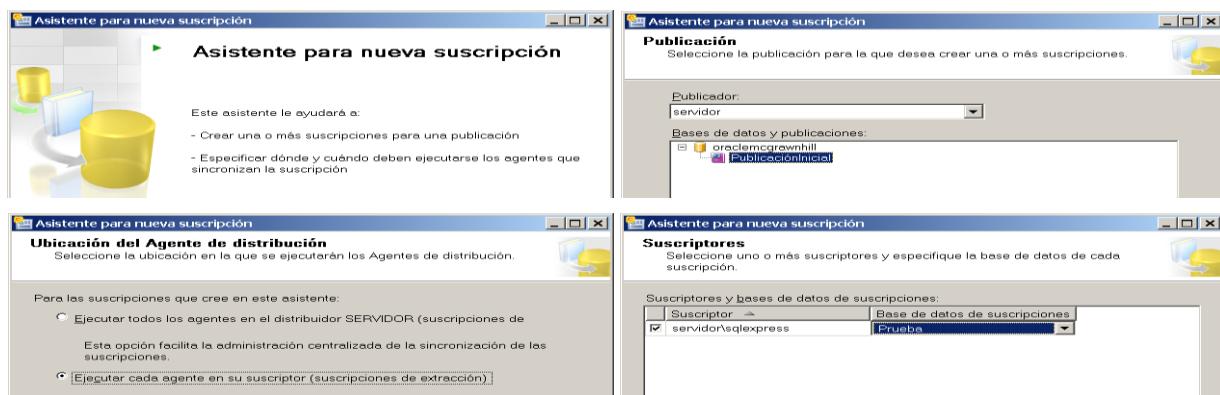
Descripción: Sucede cada día cada 1 horas entre las 0:00:00 y las 23:59:59. Se utilizará la programación que empieza el 14/01/2008.

Aceptar | Cancelar | Ayuda

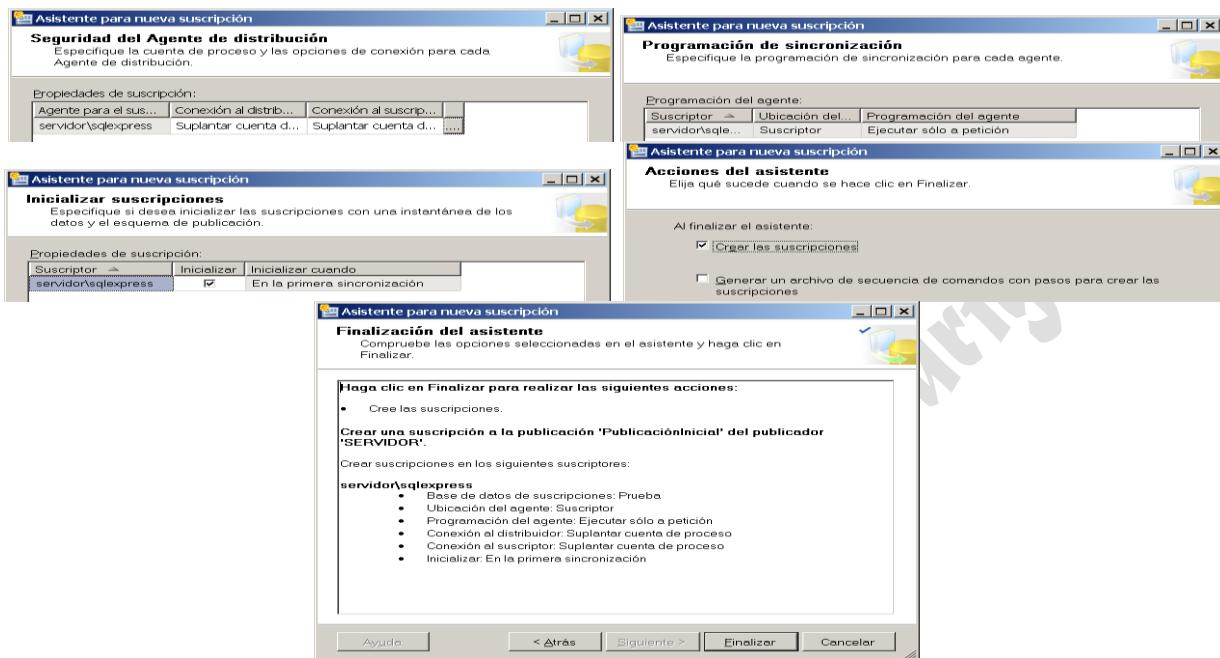


Una vez creada la publicación, botón derecho, en el cuadro de diálogo Ver estado del agente de instantáneas - <publicación>, haga clic en Iniciar.

Ahora deberemos crear las suscripciones a la publicación, botón derecho réplica – nueva suscripción:



Ojo la opción de ejecutar agente, puesto que SQLEXPRESS no lo tiene, por lo que habría que activar ejecutar todos los agentes en el distribuidor SERVIDOR.



Ejemplos:

<http://technet.microsoft.com/es-es/library/ms151866.aspx>
<http://technet.microsoft.com/es-es/library/ms161544.aspx>

<https://www.microsoft.com/nsatc.net/technet/technetmag/issues/2007/03/InsideMSCOM/default.aspx?loc=es/>

Implementar una réplica

<http://technet.microsoft.com/es-es/library/ms151847.aspx>
<http://technet.microsoft.com/es-es/library/aa337389.aspx> - Tutorial



Módulo 11 SOLUCIONES EJERCICIOS

Ejercicio 10 (pág. 82)

1.

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGA1...]

```
ALTER TABLE emp
    ALTER COLUMN sal NUMERIC(7,2) NOT NULL;
```

(0 filas afectadas)

Completado PCCESARP (8.0) IEF-SGA1\B2110 (52) PRUEBA 0:00:00 0 filas Lin 2, Col 36

Conexiones: 1 NUM

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGA1...]

```
CREATE TABLE dept1
    (deptno numeric(2),
    dname char(9),
    loc char(10)
    CONSTRAINT unq_dname UNIQUE (dname))
```

Comandos completados con éxito.

Completado PCCESARP (8.0) IEF-SGA1\B2110 (52) PRUEBA 0:00:00 0 filas Lin 6, Col 7

Conexiones: 1 NUM

2.

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGA1...]

```
CREATE TABLE dept2
    (deptno numeric(2) CONSTRAINT pk_dept1 PRIMARY KEY,
    dname char(9),
    loc char(10)
    );
```

Comandos completados con éxito.

Completado PCCESARP (8.0) IEF-SGA1\B2110 (52) PRUEBA 0:00:00 0 filas Lin 5, Col 1

Conexiones: 1 NUM

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGA1\B2110...]

```
CREATE TABLE dept2
    (deptno numeric(2),
    dname char(9),
    loc char(10)
    CONSTRAINT pk_dept1 PRIMARY KEY
    );
```

Comandos completados con éxito.

Completado PCCESARP (8.0) IEF-SGA1\B2110 (52) PRUEBA 0:00:00 0 filas Lin 5, Col 8

Conexiones: 1 NUM

3.

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGA1\B2110...]

```
CREATE TABLE emp3
    (empno numeric(4),
    ename char(10),
    job char(9),
    mgr numeric(4),
    hiredate datetime,
    sal numeric(7,2),
    comm numeric(7,2),
    deptno numeric(2)
    CONSTRAINT fk_deptno REFERENCES dept(deptno)
    );
```

Comandos completados con éxito.

Completado el: PCCESARP (8.0) IEF-SGA1\B2110 (52) PRUEBA 0:00:00 0 filas Lin 10, Col 8

Conexiones: 1 NUM

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGAI] b2110 - 1 van 100*

```
CREATE TABLE emp4
    empno numeric(4),
    ename char(10),
    job char(9),
    mgr numeric(4),
    hiredate datetime,
    sal numeric(7,2),
    comm numeric(7,2),
    deptno numeric(2),
    CONSTRAINT fk_deptno FOREIGN KEY (deptno) REFERENCES dept(deptno)
) ;

Comandos completados con éxito.
```

Cuadriculas Mensajes

Completo PCCESARP (8.0) IEF-SGAI\b2110 (52) PRUEBA 0:00:00 0 filas Lin 10, Col 20

Conexiones: 1 Tiempo de ejecución: 0:00:0

4.

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGAI] b2110 - 1 van 100*

```
CREATE TABLE emp5
    empno numeric(4),
    ename char(10),
    job char(9),
    mgr numeric(4),
    hiredate datetime,
    sal numeric(7,2),
    comm numeric(7,2),
    deptno numeric(2) REFERENCES dept(deptno)
) ;

Comandos completados con éxito.
```

Cuadriculas Mensajes

Completo PCCESARP (8.0) IEF-SGAI\b2110 (52) PRUEBA 0:00:00 0 filas Lin 1, Col 18

Conexiones: 1 Tiempo de ejecución: 0:00:0

5.

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGAI] b2110 - 1 van 100*

```
ALTER TABLE emp3
    ADD CONSTRAINT una_sal_coma UNIQUE (sal, comm);

Comandos completados con éxito.
```

Cuadriculas Mensajes

Completo PCCESARP (8.0) IEF-SGAI\b2110 (52) PRUEBA 0:00:00 0 filas Lin 3, Col 2

Conexiones: 1 Tiempo de ejecución: 0:00:0

Análizador de consultas SQL - [Consulta - PCCESARP.PRUEBA.IEF-SGAI] b2110 - 1 van 100*

```
DROP TABLE emp5;

Comandos completados con éxito.
```

Cuadriculas Mensajes

Completo PCCESARP (8.0) IEF-SGAI\b2110 (52) PRUEBA 0:00:00 0 filas Lin 2, Col 7

Conexiones: 1 Tiempo de ejecución: 0:00:0

Ejercicio 11. McGrawHill CASE. Crear un diagrama a partir de la base de datos y responder a las siguientes consultas. (pág. 105)

Ej. 5

```
select autor, substring(autor,1,isnull(nullif(charindex(',',',autor)-1,-1),len(autor))) "apellido"
from libros
```

```
select autor, substring(autor,1,charindex(',',',autor+',')-1) "apellido"
from libros
```

--6

```
select distinct autor,
substring(autor,charindex(',',',autor)+2,len(autor)) nombre
from libros
where charindex(',',',autor)>0
```

--7

```
select rtrim(substring(autor,charindex(',',',autor)+2,len(autor))) +' '+
substring(autor,1,charindex(',',',autor)-1) n
from libros
where charindex(',',',autor)>0
```

--8

```
select LEN(titulo), titulo from libros
```

```
order by 1
```

```
--9
select nombre, fechanac, 'Nació el ' + CAST(DAY(fechanac) AS VARCHAR(2)) + ' de ' + DATENAME(MM, fechanac)+' de '+CAST(YEAR(fechanac) AS VARCHAR(4))
from nacimientos
```

```
--13 sin CASE
*****
select 'SEVEN', estante, ejemplares
from libreria
where ejemplares=7
union
select tema, estante, ejemplares
from libreria
where ejemplares<>7
order by 2
*****
select
case ejemplares
when 7 then 'SEVEN'
else tema
end , estante,ejemplares
from libreria
```

```
--14
select apellido, fecha_alt from emple
where datediff(year,fecha_alt,getdate())>15
```

mejor esta:

```
select apellido, fecha_alt from emple
where convert(numeric,(getdate()-fecha_alt))/36>15
```

```
--15
select apellido from emple e, depart d
where
e.dept_no=d.dept_no and
d.dnombre='ventas' and
datediff(year,fecha_alt,getdate())>16
```

```
--16
-- mala pero parece buena
select apellido, salario, dept_no from emple
where salario in (select max(salario) sal from emple group by dept_no)
```

```
-- buena gonzalo
select apellido, salario, dept_no from emple e
where salario in (select max(salario) sal from emple em where em.dept_no =
e.dept_no)
```

```
-- buena pedro
select e.apellido, e.salario, e.dept_no
from emple e, (select max(salario) sal, dept_no dpt from emple group by
dept_no) t
where e.salario = t.sal and e.dept_no = t.dpt
```

```
-- buena en oracle
select e.apellido, e.salario, e.dept_no
```



```

from emple e
where (e.salario,e.dept_no) = (select max(salario) sal, dept_no dpt from
emple group by dept_no)

--17
select e.apellido, e.salario, e.dept_no
from emple e, (select avg(salario) sal, dept_no dpt from emple group by
dept_no) t
where e.salario > t.sal and e.dept_no=t.dpt

```

Ejercicio 12. McGrawHill. CASE. Subconsultas pág. 107

1)

```

SELECT SUM(SALARIO),OFICIO FROM EMPLE,DEPART
WHERE EMPLE.DEPT_NO=DEPART.DEPT_NO AND
DEPART.DNOMBRE = 'VENTAS'
GROUP BY OFICIO

```

2)

```

SELECT APELLIDO FROM EMPLE "E"
WHERE SALARIO = (SELECT AVG(SALARIO) FROM EMPLE
WHERE "E".DEPT_NO=EMPLE.DEPT_NO
GROUP BY DEPT_NO )

```

3)

```

SELECT COUNT(DEPT_NO), DEPT_NO FROM EMPLE
WHERE OFICIO='EMPLEADO'
GROUP BY DEPT_NO

```

4)

```

SELECT DEPT_NO FROM (SELECT COUNT(DEPT_NO) "NUMERO", DEPT_NO FROM EMPLE
WHERE OFICIO='EMPLEADO'
GROUP BY DEPT_NO ) as subconsulta1
WHERE subconsulta1.NUMERO = (SELECT MAX (NUMERO) FROM (SELECT
COUNT(DEPT_NO) "NUMERO", DEPT_NO FROM EMPLE
WHERE OFICIO='EMPLEADO'
GROUP BY DEPT_NO ) as subconsulta2)

```

5)

```

SELECT subconsulta1.DEPT_NO, DNOMBRE FROM (SELECT COUNT(DEPT_NO) as
NUMERO, DEPT_NO FROM EMPLE
WHERE OFICIO='EMPLEADO'
GROUP BY DEPT_NO ) as subconsulta1, DEPART
WHERE NUMERO = (SELECT MAX (NUMERO) FROM (SELECT COUNT(DEPT_NO) as
NUMERO, DEPT_NO FROM EMPLE
WHERE OFICIO='EMPLEADO'
GROUP BY DEPT_NO ) as subconsulta )
AND subconsulta1.DEPT_NO = DEPART.DEPT_NO

```

6)

```

SELECT DEPT_NO, COUNT(OFICIO) FROM EMPLE
GROUP BY DEPT_NO HAVING COUNT(OFICIO)>2

```

7)

```
SELECT alum.nombre
FROM alum,antiguos,nuevos
WHERE alum.nombre = antiguos.nombre AND alum.nombre = nuevos.nombre;
```

8)

```
SELECT nombre
FROM alum
INTERSECT
SELECT nombre
FROM antiguos
INTERSECT
SELECT nombre
FROM nuevos;
```

```
SELECT alum.nombre
FROM alum WHERE alum.nombre IN (SELECT antiguos.nombre from antiguos,nuevos
WHERE antiguos.nombre = nuevos.nombre);
```

9)

```
SELECT nombre
FROM alum
MINUS
SELECT nombre
FROM nuevos
MINUS
SELECT nombre
FROM antiguos
```

```
SELECT nombre
FROM alum
MINUS
(SELECT nombre
FROM nuevos
UNION
SELECT nombre
FROM antiguos)
```

```
SELECT nombre
FROM alum
WHERE nombre NOT IN
(SELECT nombre
FROM nuevos)
```

UNION
 SELECT nombre
 FROM antiguos)

10)

```
SELECT NOMBRE, ESPECIALIDAD,COUNT(DNI) "Num. Profesores"  

FROM CENTROS C, PROFESORES P  

WHERE C.COD_CENTRO=P.COD_CENTRO(+)  

GROUP BY NOMBRE,ESPECIALIDAD;
```

11).

```
SELECT count(DNI) empleados,centros.cod_centro,centros.nombre  

FROM personal,centros  

where centros.cod_centro=personal.cod_centro(+)  

GROUP BY centros.cod_centro,centros.nombre;
```

12)

```
SELECT especialidad, COUNT(especialidad) FROM profesores  

HAVING COUNT(especialidad)=(SELECT MIN(COUNT(especialidad)) FROM  

profesores GROUP BY especialidad)  

GROUP BY especialidad
```

13)

```
SELECT bancos.nombre_banc,count(sucursales.cod_sucur)  

FROM bancos,sucursales  

WHERE bancos.cod_banco = sucursales.cod_banco  

HAVING COUNT(cod_sucur)=(select MAX(count(cod_sucur)) from sucursales group  

by cod_banco)  

GROUP BY bancos.nombre_banc;
```

14)

```
SELECT  

BANCOS.NOMBRE_BANC,sum(CUENTAS.SALDO_DEBE),sum(CUENTAS.SALDO_HAB  

ER)  

FROM BANCOS,CUENTAS  

WHERE NOMBRE_BANC LIKE '%GUADALAJARA%' AND  

BANCOS.COD_BANCO=CUENTAS.COD_BANCO  

group by nombre_banc
```

```
SELECT NOMBRE_BANC Nombre, SUM(SALDO_DEBE) "Debe", SUM(SALDO_HABER)  

"Haber"  

FROM BANCOS, CUENTAS  

WHERE BANCOS.COD_BANCO=CUENTAS.COD_BANCO and  

POBLACION='GUADALAJARA'  

group by nombre_banc
```

15)

```

SELECT      COUNT(*)      a,movimientos.num_cta,cuentas.nombre_cta      FROM
MOVIMIENTOS,cuentas
      where cuentas.num_cta=movimientos.num_cta
      GROUP BY MOVIMIENTOS.NUM_CTA,cuentas.nombre_cta
      having count(*)= (SELECT MAX(COUNT(*)) a FROM MOVIMIENTOS GROUP BY
MOVIMIENTOS.NUM_CTA)
  
```

16)

-- Cuenta el número de reintegros.

```

SELECT sucursales.nombre_suc,count(movimientos.tipo_mov)
      FROM sucursales, movimientos
      WHERE sucursales.cod_sucur=      movimientos.cod_sucur      AND      movimien-
tos.tipo_mov='R'
      HAVING      COUNT      (movimientos.tipo_mov)=(select
MAX(count(movimientos.tipo_mov))from      movimientos      where      tipo_mov='R'      group      by
cod_sucur)
      GROUP BY sucursales.nombre_suc;
  
```

--Suma de los reintegros.

```

SELECT NOMBRE_SUC "Sucursal", SUM(SALDO_DEBE) "Reintegro"
      FROM SUCURSALES,CUENTAS
      WHERE SUCURSALES.COD_BANCO=CUENTAS.COD_BANCO AND
SUCURSALES.COD_SUCUR=CUENTAS.COD_SUCUR
      GROUP BY NOMBRE_SUC
      HAVING      SUM(SALDO_DEBE)=(SELECT      MAX(SUM(SALDO_DEBE))      FROM
CUENTAS
      GROUP BY COD_BANCO, COD_SUCUR);
  
```

Ejercicio 13. Diseñar las siguientes consultas con funciones: (pág. 108)

1. Visualiza los títulos de la tabla mistextos sin los caracteres punto y comillas, y en minúscula..

```

select lower(replace(replace(titulo, '.', ''), '''', '''))
from mistextos
  
```

2. Dada la tabla LIBROS, escribe la sentencia SELECT que visualice dos columnas, una con el AUTOR y otra con el apellido del autor.

```

SELECT substring(autor,1,charindex(',',autor)-1) Apellido,
       ltrim(substring(autor,charindex(',',autor)+1,len(autor))) Nombre,
autor NombreCompleto
      from libros
  
```



3. Escribe la sentencia SELECT que visualice las columnas de AUTOR y otra columna con el nombre del autor (sin el apellido) de la tabla LIBROS.

```
select Concat(Substr(autor,instr(autor,'')+1) , ' ') || Substr(autor,1,instr(autor,',')-1) from libros
```

4. A partir de la tabla LIBROS, realiza una sentencia SELECT que visualice en una columna, primero el nombre del autor y, luego, su apellido.

```
select Concat(Substr(autor,instr(autor,',')+1) , ' ') || Substr(autor,1,instr(autor,',')-1) from libros
```

5. A partir de la tabla LIBROS, realiza una sentencia SELECT para que aparezcan los títulos ordenados por su número de caracteres.

```
select titulo, len(titulo) numero
from libros order by numero
```

6. Dada la tabla NACIMIENTOS, realiza una sentencia SELECT que obtenga la siguiente salida: NOMBRE, FECHANAC, FECHA_FORMATADA, donde FECHA_FORMATADA tiene el siguiente formato:

“Nació el 12 de mayo de 1982”

```
SELECT 'Nació el ' + CONVERT(VARCHAR(10), DATEPART(day, FECHANAC))
      + ' de ' + CONVERT(VARCHAR(10), DATEPART(month, FECHANAC))
      + ' del ' + CONVERT(VARCHAR(10), DATEPART(year, FECHANAC))
FROM NACIMIENTOS
```

7. Visualiza aquellos temas de la tabla LIBRERÍA cuyos ejemplares sean 7 con el nombre de tema de ‘SEVEN’; el resto de temas que no tenga 7 ejemplares se visualizarán como están.

```
SELECT TEMA, EJEMPLARES,
DECODE(EJEMPLARES,7,'SEVEN',TEMA) "TEMA2"
FROM LIBRERIA;
```

```
SELECT TEMA,EJEMPLARES,
DECODE(EJEMPLARES,7,DECODE(TEMA,TEMA,'SEVEN',TEMA),TEMA) "TEMA2"
FROM LIBRERIA;
```

8. A partir de la tabla EMPLE, obtén el apellido de los empleados que lleven más de 15 años trabajando.

```
SELECT apellido FROM emple
WHERE MONTHS_BETWEEN(SYSDATE,FECHA_ALT)/12>15;
```

9. Selecciona el apellido de los empleados de la tabla EMPLE que lleven más de 16 años trabajando en el departamento

```
SELECT APELLIDO, MONTHS_BETWEEN(SYSDATE,FECHA_ALT)/12
FROM EMPLE, DEPART
WHERE (MONTHS_BETWEEN(SYSDATE,FECHA_ALT)/12)>16 AND
DNOMBRE='VENTAS' AND DEPART.DEPT_NO=EMPLE.DEPT_NO;
```

```

SELECT          APELLIDO,
to_number('19'||to_char(fecha_alt,'yy'))
FROM EMPLE, DEPART
WHERE DEPART.DEPT_NO=EMPLE.DEPT_NO AND to_number(to_char(sysdate,'yyyy'))-
to_number('19'||to_char(fecha_alt,'yy'))>=16

SELECT APELLIDO, convert(int, getdate())-fecha_alt)/365
FROM EMPLE, DEPART
WHERE DEPART.DEPT_NO=EMPLE.DEPT_NO

```

10. Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario sea el mayor de su departamento

```

SELECT apellido, salario, dept_no from emple "e"
WHERE SALARIO =(SELECT MAX(SALARIO) FROM
emple WHERE dept_no="e".dept_no)

```

11. Visualiza el apellido, el salario y el número de departamento de aquellos empleados de la tabla EMPLE cuyo salario supere a la media en su departamento.

```

SELECT apellido, salario, dept_no from emple "e"
WHERE SALARIO>(SELECT AVG(SALARIO) FROM
emple WHERE dept_no="e".dept_no)

```

12. Desde la tabla EMPLE, visualiza el departamento que tenga más empleados cuyo oficio sea 'EMPLEADO'

```

SELECT DEPT_NO FROM (SELECT COUNT(DEPT_NO)"NUMERO",DEPT_NO FROM
EMPLE
          WHERE OFICIO='EMPLEADO'
          GROUP BY DEPT_NO )
          WHERE NUMERO = (SELECT MAX (NUMERO) FROM (SELECT
COUNT(DEPT_NO)"NUMERO",DEPT_NO FROM EMPLE
          WHERE OFICIO='EMPLEADO'
          GROUP BY DEPT_NO ))

```

13. A partir de las tablas EMPLE y DEPART, visualiza el número de departamento y el nombre de departamento que tenga más empleados cuyo oficio sea 'EMPLEADO'

```

SELECT      E.DEPT_NO,          DNOMBRE        FROM      (SELECT
COUNT(DEPT_NO)"NUMERO",DEPT_NO FROM EMPLE
          WHERE OFICIO='EMPLEADO'
          GROUP BY DEPT_NO )"E", DEPART
          WHERE NUMERO = (SELECT MAX (NUMERO) FROM (SELECT
COUNT(DEPT_NO)"NUMERO",DEPT_NO FROM EMPLE
          WHERE OFICIO='EMPLEADO'
          GROUP BY DEPT_NO ))
          AND E.DEPT_NO = DEPART.DEPT_NO

```

14. Busca los departamentos que tienen más de 2 personas trabajando en la misma posición

```
SELECT DEPT_NO, COUNT(OFICIO) FROM EMPLE
GROUP BY DEPT_NO HAVING COUNT(OFICIO)>2
```

Ejercicio 14 (pág. 109)

1. *SELECT NOMBRE, CUOTA, VENTAS
FROM REPVENTAS
WHERE (NUM_EMPL = 107)*

2. *SELECT NOMBRE, OFICINA_REP AS Contrato
FROM REPVENTAS*

3. *SELECT NOMBRE, VENTAS, CONTRATO AS Expr1
FROM REPVENTAS
WHERE (VENTAS > 100000 AND VENTAS < 300000)*

4. *SELECT CIUDAD, DIR
FROM OFICINAS
WHERE (DIR <> 108)*

5. *SELECT REPVENTAS.NOMBRE, REPVENTAS.CUOTA, OFICINAS.CIUDAD
FROM REPVENTAS INNER JOIN
OFICINAS ON REPVENTAS.OFICINA_REP = OFICINAS.OFICINA AND
REPVENTAS.NUM_EMPL = OFICINAS.DIR
WHERE (OFICINAS.CIUDAD = 'NEW YORK') OR
(OFICINAS.CIUDAD = 'ATLANTA') OR
(OFICINAS.CIUDAD = 'DENVER')*

6. *SELECT NOMBRE, OFICINA_REP
FROM REPVENTAS
WHERE (NOT (OFICINA_REP IS NULL))*

7.
*SELECT AVG(EDAD) AS MediaEdad, TITULO
FROM REPVENTAS
GROUP BY TITULO*

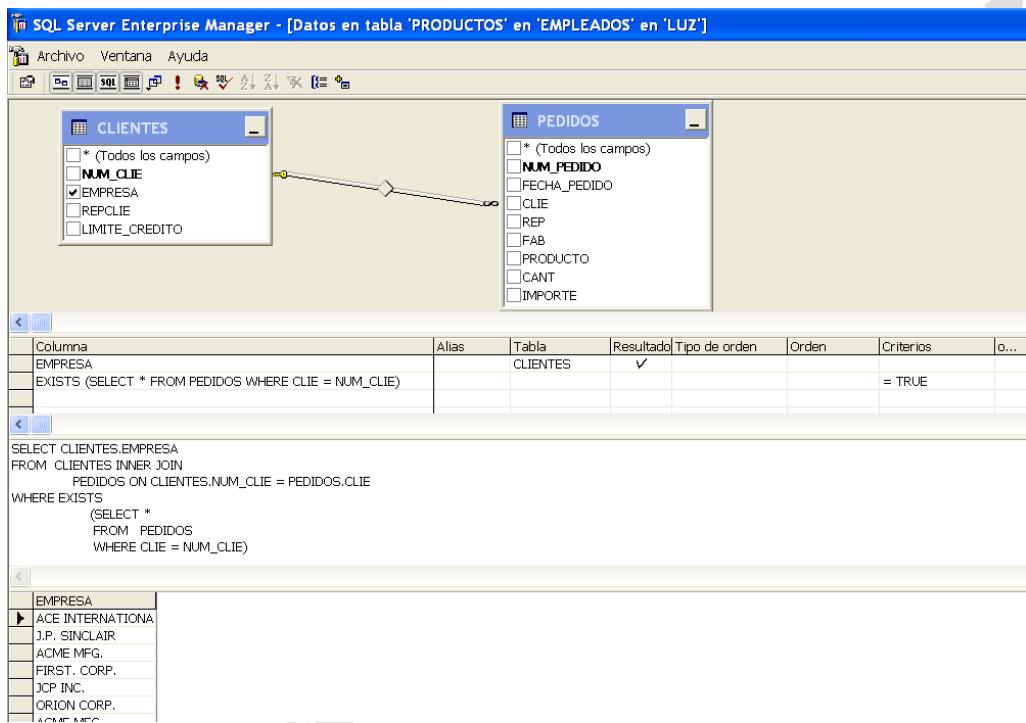
8.

*SELECT REPVENTAS.NOMBRE AS RepresentanteVentas, CLIENTES.LIMITE_CREDITO AS Lim
FROM CLIENTES INNER JOIN
REPVENTAS ON CLIENTES.REPCLIE = REPVENTAS.NUM_EMPL
WHERE CLIENTES.LIMITE_CREDITO > (SELECT avg(limite_credito) FROM
CLIENTES)*

9.

```

SELECT CLIENTES.EMPRESA
FROM CLIENTES INNER JOIN
    PEDIDOS ON CLIENTES.NUM_CLIE = PEDIDOS.CLIE
WHERE EXISTS
    (SELECT *
     FROM PEDIDOS
     WHERE CLIE = NUM_CLIE)
  
```



10.

```

SELECT NOMBRE AS RepresentanteVentas, EDAD
FROM REPVENTAS
WHERE (EDAD > ?)
  
```

Ó si previamente definimos los signos [] con elementos que engloban parámetros:

```

SELECT NOMBRE AS RepresentanteVentas, EDAD
FROM REPVENTAS
WHERE (EDAD > ?)
  
```

15.

```

SELECT LIMITE_CREDITO, REPCLIE, (SELECT MAX(limite_credito) from clientes) AS limitemax
INTO [temp]
FROM CLIENTES
  
```

Ejercicio 15 (pág. 110)

1.

Análizador de consultas SQL - [Consulta - PCCESARP.ESTUDIANTE]

```
SELECT cno, nombre, ctarifa
FROM cursos
WHERE ctarifa = (SELECT MIN(ctarifa)
                  FROM cursos
                  WHERE NOT ctarifa=0)
```

cno	nombre	ctarifa
C22	ESTRUCT. DATOS	50
P22	RACIONALISMO	50

Cuadriculas Mensajes

PCCESARP [IEF-SGA]b2110 (52) ESTUDIANTES 0:00:00 2 filas Lin 3, Col 16

Conexiones: 1 NUM

2.

Análizador de consultas SQL - [Consulta - PCCESARP.ESTUDIANTE]

```
SELECT cno, nombre, ctarifa
FROM cursos
WHERE ctarifa < (SELECT AVG(ctarifa)
                  FROM cursos)
```

cno	nombre	ctarifa
C11	INTROD. CC.	100
C22	ESTRUCT. DATOS	50
C33	MATEMATICAS DISCRETAS	0
C44	CIRCUITOS DIGITALES	0
C55	ARQUIT. COMPUTADOR	100
P11	EMPIRISMO	100
P22	RACIONALISMO	50
P44	SOLIPSISMO	0
T22	FUNDAMENTALISMO	90
T33	HEDONISMO	0

Cuadriculas Mensajes

PCCESARP [IEF-SGA]b2110 (52) ESTUDIANTES 0:00:00 10 filas Lin 4, Col 16

Conexiones: 1 NUM

3.

Análizador de consultas SQL - [Consulta - PCCESARP.ESTUDIANTE]

```
SELECT cno, nombre, ctarifa
FROM cursos
WHERE ctarifa >= (SELECT MIN(esueldo)
                   FROM personal)
```

cno	nombre	ctarifa
C11	INTROD. CC.	100
C55	ARQUIT. COMPUTADOR	100
C66	BASES DE DATOS RELAC.	500
P11	EMPIRISMO	100
P33	EXISTENCIALISMO	200
T11	ESCOLASTICISMO	150
T22	FUNDAMENTALISMO	90
T44	COMUNISMO	200

Cuadriculas Mensajes

PCCESARP [IEF-SGA]b2110 (52) ESTUDIANTES 0:00:00 8 filas Lin 5, Col 19

Conexiones: 1 NUM

4.



Análizador de consultas SQL - [Consulta - PCCESARP.ESTUDIANTE]

```
SELECT nombre, cargo
FROM personal
WHERE dept IN (SELECT dept
                FROM departamento
                WHERE dedir='HU')  


| nombre    | cargo    |
|-----------|----------|
| DICK NIX  | LADRON   |
| HANK KISS | BUTON    |
| JUAN      | EVANGELA |
| LUCAS     | EVANGEL  |
| MARCOS    | EVANGEL  |
| MATEO     | EVANGEL  |



Cuadriculas Mensajes



PCCESARP [IEF-SGA]b2110 (52) ESTUDIANTES 0:00:00 6 filas Lin 5, Col 25



Conexiones: 1 NUM


```

5.

Análizador de consultas SQL - [Consulta - PCCESARP.ESTUDIANTE]

```
SELECT DISTINCT fno, nombre
FROM claustro
WHERE fno IN (SELECT dchfno
              FROM departamento
              WHERE dept IN (SELECT cddept
                             FROM cursos
                             WHERE cred=6))  


| fno | nombre        |
|-----|---------------|
| 10  | JESSIE MARTIN |
| 60  | JULIA MARTIN  |



Cuadriculas Mensajes



PCCESARP [IEF-SGA]b2110 (52) ESTUDIANTES 0:00:00 2 filas Lin 6, Col 21



Conexiones: 1 NUM


```

6.

Análizador de consultas SQL - [Consulta - PCCESARP.ESTUDIANTE]

```
SELECT nombre, cargo, dept
FROM personal
WHERE dept NOT IN (SELECT dept
                     FROM departamento)  


| nombre     | cargo           | dept |
|------------|-----------------|------|
| ARQUIMEDES | AYTE. LAB. ENG  |      |
| EUCLIDES   | AYTE. LAB. MATH |      |



Cuadriculas Mensajes



PCCESARP [IEF-SGA]b2110 (52) ESTUDIANTES 0:00:00 2 filas Lin 5, Col 1



Conexiones: 1 NUM


```

7.

Análizador de consultas SQL - [Consulta - PCCESARP.ESTUDIANTE]

```
SELECT cddept, AVG(ctarifa)
FROM cursos
GROUP BY cddept
HAVING AVG(ctarifa)> (SELECT AVG(ctarifa)
                          FROM cursos)  


| cddept | (sin nombre de columna) |
|--------|-------------------------|
| C12    | 125,000000              |



Cuadriculas Mensajes



PCCESARP [IEF-SGA]b2110 (52) ESTUDIANTES 0:00:00 1 filas Lin 5, Col 26



Conexiones: 1 NUM


```

8.

Análizador de consultas SQL - [Consulta - PCCESARP/ESTUDIANTES]

```
SELECT cno, nombre, ctarifa
FROM cursos
WHERE ctarifa > ANY (SELECT esueldo
                      FROM personal)
```

eno	nombre	ctarifa
1	C11 INTROD. CC.	100
2	C55 ARQUIT. COMPUTADOR	100
3	C66 BASES DE DATOS RELAC.	500
4	P11 EMPIRISMO	100
5	P33 EXISTENCIALISMO	200
6	T11 ESCOLASTICISMO	150
7	T22 FUNDAMENTALISMO	90
8	T44 COMUNISMO	200

Cuadriculas Mensajes

PCCESARP [IEF-SGAI]b2110 (52) ESTUDIANTES | 0:00:00 | 8 filas | Lin 4, Col 28

Conexiones: 1 NUM

9.

Análizador de consultas SQL - [Consulta - PCCESARP/ESTUDIANTES]

```
SELECT cno, nombre, ctarifa
FROM cursos
WHERE ctarifa < ALL (SELECT esueldo
                      FROM personal)
```

eno	nombre	ctarifa
1	C22 ESTRUCT. DATOS	50
2	C33 MATEMATICAS DISCRETAS	0
3	C44 CIRCUITOS DIGITALES	0
4	P22 RACIONALISMO	50
5	P44 SOLIPSISMO	0
6	T33 HEDONISMO	0

Cuadriculas Mensajes

PCCESARP [IEF-SGAI]b2110 (52) ESTUDIANTES | 0:00:00 | 6 filas | Lin 3, Col 20

Conexiones: 1 NUM

10.

Análizador de consultas SQL - [Consulta - PCCESARP/ESTUDIANTES]

```
SELECT cdept, cno, nombre, ctarifa
FROM cursos CX
WHERE ctarifa = (SELECT MAX(ctarifa)
                  FROM cursos
                  WHERE cdept=CX.cdept)
```

cdept	eno	nombre	ctarifa
THEO	T44	COMUNISMO	200
PHIL	P33	EXISTENCIALISMO	200
CIS	C66	BASES DE DATOS RELAC.	500

Cuadriculas Mensajes

PCCESARP [IEF-SGAI]b2110 (52) ESTUDIANTES | 0:00:00 | 3 filas | Lin 3, Col 37

Conexiones: 1 NUM

11.

Análizador de consultas SQL - [Consulta - PCCESARP/ESTUDIANTES]

```
SELECT nombre, cargo
FROM personal
WHERE EXISTS (SELECT *
               FROM departamento
               WHERE DEPARTAMENTO.dept = PERSONAL.dept)
```

nombre	cargo
DICK MIX	LADRON
HANK KISS	BUFON
JUAN	EVANGEL
LUCAS	EVANGI
MARCOS	EVANGI
NATEO	EVANGI

Cuadriculas Mensajes

PCCESARP (8,0) [IEF-SGAI]b2110 (52) ESTUDIANTES | 0:00:00 | 6 filas | Lin 5, Col 32

Conexiones: 1 NUM



Análizador de consultas SQL - [Consulta - PCCESARP/ESTUDIANTES]

```
SELECT nombre, cargo
FROM personal
WHERE NOT EXISTS (SELECT *
                   FROM departamento
                   WHERE DEPARTAMENTO.dept = PERSONAL.dept)
```

nombre	cargo
ARQUIMEDES	ATTE. LAB
DAVINCI	ATTE. LAB
EUCLIDES	ATTE. LAB

Cuadriculas Mensajes

Completado PCCESARP (8,0) [IEF-SGAI]b2110 (52) ESTUDIANTES | 0:00:00 | 3 filas | Lin 5, Col 12

Conexiones: 1 NUM

13.

Análizador de consultas SQL - [Consulta - PCCESARP/ESTUDIANTE]

```
SELECT LEN(nombre)
FROM cursos
WHERE cdept='PHIL'
```

(Si no existe la columna)
9
12
15
10

Cuadriculas Mensajes

PCCESARP [IEF-SGAI]b2110 (52) ESTUDIANTE | 0:00:00 | 4 filas | Lin 2, Col 12

Conexiones: 1 NUM

14.

Análizador de consultas SQL - [Consulta - PCCESARP/ESTUDIANTE]

```
SELECT SUBSTRING(nombre, 1, 1) AS [INICIAL],
       SUBSTRING(nombre, 3, 1) AS [C3-6]
  FROM cursos
 WHERE cdept='CIS'
```

INICIAL	C3-6
I	T
X	T
M	T
C	R
A	Q
B	S

Cuadriculas Mensajes

PCCESARP [IEF-SGAI]b2110 (52) ESTUDIANTE | 0:00:00 | 6 filas | Lin 2, Col 39

Conexiones: 1 NUM

15.

16.

Ejercicio 16. McGraw-Hill. Actualizaciones, borrados....(pág. 114)

1)

```
UPDATE centros SET centros.num_plazas = (centros.num_plazas/2)
```

```
WHERE centros.cod_centro in (SELECT cod_centro FROM profesores GROUP BY profesores.cod_centro HAVING COUNT(*)<2))
```

```
UPDATE CENTROS SET NUM_PLAZAS =NUM_PLAZAS/2
WHERE COD_CENTRO IN (
    SELECT COD_CENTRO FROM PROFESORES
    GROUP BY COD_CENTRO
    HAVING COUNT(*)<2
    UNION (SELECT COD_CENTRO FROM CENTROS
    MINUS
    SELECT COD_CENTRO FROM PROFESORES));
```

2) DELETE FROM centros

```
WHERE centros.cod_centro IN
((SELECT centros.cod_centro FROM centros)MINUS (SELECT personal.cod_centro
FROM personal GROUP BY personal.cod_centro))
```

3)

```
INSERT INTO PROFESORES
(PROFESORES.COD_CENTRO,PROFESORES.ESPECIALIDAD,PROFESORES.DNI,PROF
ESORES.APELLIDOS)
SELECT DISTINCT CENTROS.COD_CENTRO,'IDIOMA','8790055','Clara Salas'
FROM(SELECT COUNT(FUNCION) TOTAL,CENTROS.COD_CENTRO
      FROM PERSONAL,CENTROS
     WHERE CENTROS.COD_CENTRO=PERSONAL.COD_CENTRO
       AND FUNCION='ADMINISTRATIVO'
      GROUP BY CENTROS.Cod_Centro) e,CENTROS,PROFESORES
     WHERE e.TOTAL=1 AND e.COD_CENTRO=CENTROS.COD_CENTRO AND
CENTROS.COD_CENTRO=PROFESORES.COD_CENTRO
```

INSERT INTO PROFESORES

```
SELECT DISTINCT COD_CENTRO,8790055, 'Salas, Clara','IDIOMA'
FROM PERSONAL WHERE COD_CENTRO IN
(SELECT COD_CENTRO FROM PERSONAL WHERE FUNCION='ADMINISTRATIVO'
      GROUP BY COD_CENTRO HAVING COUNT(*)=1)
```

4) Borra al personal que esté en centros de menos de 300 plazas y con menos de dos profesores

DELETE FROM personal

```
WHERE personal.cod_centro IN (SELECT cod_centro FROM profesores GROUP BY profesores.cod_centro HAVING COUNT(*)<2)
```

```
AND personal.cod_centro IN (SELECT cod_centro FROM centros WHERE num_plazas<300)
```

5)

DELETE FROM profesores WHERE profesores.dni NOT IN
 (SELECT personal.dni FROM personal)

6)

```
INSERT INTO articulos
  (articulos.articulo,articulos.cod_fabricante,articulos.peso,articulos.categoría)
VALUES('articulo',(SELECT fabricantes.cod_fabricante FROM fabricantes WHERE fabricantes.pais='FRANCIA'),'23','Primera')
```

```
INSERT INTO pedidos
  SELECT NIF , 'articulo',COD_FABRICANTE,4,'Primera',sysdate,5
  FROM tiendas,fabricantes where pais='Francia'
```

7)

```
INSERT INTO pedidos SELECT '1111-A',articulo,'30','23','Primera',sysdate,'20' FROM ventas
  GROUP BY articulo HAVING COUNT(articulo)=(SELECT MAX(COUNT(articulo))
FROM ventas GROUP BY articulo)
```

8)

```
INSERT INTO Tiendas
  VALUES ('1010-C','La Cesta','C/Juan Mazo 30','Alcalá','Madrid',28809)
INSERT INTO Pedidos (nif,Unidades_Pedidas)
  SELECT '1010-C', ARTICULO, COD_FABRICANTE, PESO, CATEGORIA, SYSDATE,20
FROM ARTICULOS;
```

9)

```
INSERT INTO TIENDAS
  VALUES('6666-B','Tienda A','C/Algo','laquesea','SEVILLA','12345')
INSERT INTO TIENDAS
  VALUES('7777-B','Tienda B','C/Algo+','otramas','SEVILLA','54321')
```

```
INSERT INTO PEDIDOS
  SELECT  NIF,  ARTICULO,  A.COD_FABRICANTE,  PESO,  CATEGORIA,
SYSDATE,30
    FROM TIENDAS, ARTICULOS A, FABRICANTES F
   WHERE PROVINCIA='SEVILLA'
   AND F.NOMBRE='GALLO'
   AND A.COD_FABRICANTE=F.COD_FABRICANTE;
```

Sin corregir:

10) INSERT INTO ventas (nif,unidades_vendidas)
 SELECT nif,'10' FROM tiendas WHERE poblacion='Toledo'

11) UPDATE PEDIDOS

```
SET UNIDADES_PEDIDAS=UNIDADES_PEDIDAS+10 WHERE PEDIDOS.ARTICULO
IN
(SELECT e.ARTICULO FROM (SELECT SUM(UNIDADES_VENDIDAS)
TOTAL,ARTICULO FROM VENTAS GROUP BY ARTICULO) e
WHERE e.TOTAL>30) AND PEDIDOS.NIF='5555-B'
```

12) UPDATE Tiendas SET NIF='2222-A'
WHERE NIF=1111-A'

13) UPDATE articulos SET articulos.categoría='Segunda'
WHERE articulos.categoría='Primera'
AND articulos.cod_fabricante IN(SELECT fabricantes.cod_fabricante FROM fabricantes
WHERE fabricantes.pais=UPPER('Italia'))

14) UPDATE PEDIDOS SET unidades_pedidas=existencias*0.2
WHERE nif IN (SELECT DISTINCT nif FROM articulos,pedidos
WHERE articulos.articulo=pedidos.articulo AND articulos.existencias< pedidos.unidades_pedidas)

15) DELETE FROM Tiendas
WHERE TIENDAS.NIF IN ((SELECT DISTINCT TIENDAS.NIF FROM TIENDAS)MINUS (SELECT DISTINCT VENTAS.NIF FROM VENTAS))

16) DELETE FROM Tiendas
WHERE TIENDAS.NIF IN ((SELECT DISTINCT TIENDAS.NIF FROM TIENDAS)MINUS (SELECT DISTINCT VENTAS.NIF FROM VENTAS))
AND TIENDAS.NIF IN ((SELECT DISTINCT TIENDAS.NIF FROM TIENDAS)MINUS (SELECT DISTINCT PEDIDOS.NIF FROM PEDIDOS))

17) DELETE FROM pedidos WHERE categoría='Primera' AND cod_fabricante =
(SELECT cod_fabricante FROM fabricantes WHERE pais=upper('Belgica'))

18) DELETE FROM Pedidos WHERE nif IS NULL

19) SELECT UNIDADES_PEDIDAS-1
FROM pedidos
WHERE FECHA_PEDIDO = (SELECT MAX(FECHA_PEDIDO) FROM PEDIDOS
WHERE NIF='5555-B') ORDER BY FECHA_PEDIDO

Ejercicio 17. Pág. 115

PN	PNOMBRE	COLOR	PESO	CiudadP3	birlo	azul	17	Rom
P1	tuerca	verde	12	París P4	birlo	rojo	14	Lond
P2	perno	rojo	17	Londres P5	leva	azul	12	París

S

sn	snombre	estado	ciudad
S1	Salazar	20	Londres
S2	Jaimes	10	París
S3	Bernal	30	París
S4	Corona	20	Londres
S5	Aldana	30	Atenas

SP

Sn	Pn	cant
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

1. Obtener el nombre de los proveedores que suministran la pieza con el código P2

$$\Pi_{snombre}(\sigma_{p\#=P2}(SP \otimes S))$$

En SQL:

```
SELECT DISTINCT S.snombre
  FROM SP, S
 WHERE SP.sn=S.sn
   AND pn='P2';
```

Otra SQL:

```
SELECT sn,snombre
  FROM S
 WHERE EXISTS (SELECT * FROM SP
      WHERE SP.sn=S.sn AND SP.bn='P2');
```

2. Obtener el nombre de los proveedores que suministran por lo menos una pieza roja.

$$\Pi_{snombre}(\sigma_{color='rojo'}(P) \otimes SP \otimes S)$$

En SQL

```
SELECT DISTINCT S.snombre
  FROM SP, P, S
 WHERE SP.bn=P.bn
   AND SP.bn=S.bn
   AND P.color='rojo';
```

Otra opción sería:

```
SELECT DISTINCT S.snombre ,count(snombre)
  FROM SP, S, P
 WHERE SP.bn=S.bn And P.bn=SP.bn and p.color='rojo'
 GROUP BY s.snombre
 HAVING count(S.snombre)>0
```

Otra opción sería:

```
SELECT sn,snombre
      FROM S
     WHERE EXISTS (SELECT * FROM SP
                   WHERE SP.sn=S.sn
                   AND EXISTS (SELECT * FROM P
                               WHERE SP.bn=P.bn
                               AND P.color= 'rojo'));
```

3. Obtener el nombre de las piezas de color rojo suministradas por los proveedores de la ciudad de Londres.

$$\Pi_{pn}(\sigma_{ciudad='Londres'}(S) \otimes SP \otimes \sigma_{color='rojo'}(P))$$

SQL:

```
SELECT DISTINCT P.bn
      FROM SP, S, P
     WHERE SP.bn=S.bn
       AND SP.bn=P.bn
       AND S.ciudad='Londres'
       AND P.color='rojo'
```

O bien:

```
SELECT pn,pnombre
      FROM P
     WHERE color='rojo'
     AND EXISTS (SELECT * FROM SP
                  WHERE SP.bn=P.bn
                  AND EXISTS (SELECT * FROM S
                              WHERE SP.bn=S.bn
                              AND S.ciudad='Londres'));
```

O bien:

```
SELECT DISTINCT rojos.bn, rojos.bnombre
      FROM (SELECT S.ciudad, s.bn FROM S
                where S.ciudad='Londres') LONDRES,
            (SELECT P.bn, p.bnombre FROM P
                WHERE color='rojo') ROJOS,
            SP
      where LONDRES.bn=sp.bn
      and rojos.bn=sp.bn
```

4. Obtener el código de los proveedores que suministran alguna de las piezas que suministra el proveedor con el código S2.

$$\Pi_{sn}((\Pi_{pn}(\sigma_{sn='S2'}(SP))) \otimes SP)$$

En SQL:

```
select distinct SN
      FROM SP
     WHERE sp.bn IN ( SELECT distinct sp.bn
                      FROM SP
                     where SP.bn='S2');
```

```
SELECT DISTINCT SPY.sn
  FROM SP SPX, SP SPY
 WHERE SPX.sn='S2'
   AND SPX.bn=SPY.bn;
```

O bien:

```
SELECT DISTINCT SPX.bn
  FROM SP SPX
 WHERE EXISTS (SELECT * FROM SP SPY
    WHERE SPY.bn='S2'
      AND SPX.bn=SPY.bn);
```

5. Obtener los datos de los envíos de más de 100 unidades, mostrando también el nombre del proveedor y el de la pieza.

$$\Pi_{sn,pn,cantidad,pnombre,snombre}(\sigma_{cantidad>100}(SP) \otimes P \otimes S)$$

```
SELECT SP.bn,S.bn,S.nombre,SP.bn,P.bn,P.nombre,SP.cantidad
  FROM SP,S,P
 WHERE SP.bn=S.bn
   AND SP.bn=P.bn
   AND SP.cantidad>100
```

6. Obtener el nombre de los proveedores que suministran todas las piezas.

$$\Pi_{snombre}((\Pi_{sn,pn}(SP) \div \Pi_{pn}(P)) \otimes S)$$

En SQL:

```
SELECT DISTINCT S.bn,S.nombre ,count(snomb)
  FROM SP, S, P
 WHERE SP.bn=S.bn And P.bn=SP.bn and p.color='rojo'
 GROUP BY s.bn
 HAVING count(S.bn)>1
```

```
SELECT S.bn,S.nombre
  FROM SP,S
 WHERE SP.bn=S.bn
 GROUP BY SP.bn,S.bn
 HAVING COUNT(*) = (SELECT COUNT(*) FROM P)
```

O bien:

```
-- Seleccionar un proveedor de manera que no exista dicho proveedor sin que exista el producto
-- que provee en SP
SELECT sn,snombre
  FROM S
 WHERE NOT EXISTS (SELECT * FROM P
    WHERE NOT EXISTS (SELECT * FROM SP
      WHERE SP.bn=S.bn
        AND SP.bn=P.bn));
```

7. Obtener el código de los proveedores que suministran, al menos, todas las piezas suministradas por el proveedor con código S2.

$$\Pi_{sn}(SP \div \Pi_{pn}(\sigma_{sn='S2'}(SP)))$$

```
SELECT SP1.sn,S.snombre
  FROM SP SP1,S, SP SP2 /* Proveedores S2*/
 WHERE SP1.sn=S.sn
   AND SP2.sn='S2'
   AND SP1.pn=SP2.pn
 GROUP BY SP1.sn,S.snombre
 HAVING COUNT(*) = (SELECT COUNT(*)
  FROM SP WHERE sn='S2')
```

O bien:

```
SELECT sn,snombre
  FROM S
 WHERE NOT EXISTS (SELECT * FROM SP SPX
   WHERE SPX.sn='S2'
 AND NOT EXISTS (SELECT * FROM SP SPY
   WHERE SPY.sn=S.sn
   AND SPX.pn=SPY.pn))
```

8. Obtener el nombre de los proveedores que no suministran la pieza con el código P2.

$$\Pi_{sn}(S) - \Pi_{sn}(\sigma_{pn='P2'}(SP))$$

En SQL Server no se puede:

```
SELECT sn,snombre
  FROM S
 MINUS
SELECT S.sn, S.snombre
  FROM SP,S
 WHERE SP.sn=S.sn AND SP.pn='P2'
```

O bien:

```
SELECT sn,snombre
  FROM S
 WHERE NOT EXISTS (SELECT * FROM SP WHERE SP.sn=S.sn AND SP.pn='P2')
O bien:
SELECT S.SNOMBRE
  FROM S
 WHERE s.snombre not in (SELECT s.snombre FROM sp,s
   where sp.sn=s.sn
   and sp.pn='P2')
```

9. Obtener los datos de los proveedores que solo suministran piezas de color rojo.

$$\Pi_{sn}(SP \times \sigma_{color='rojo'}(P)) - \Pi_{sn}(SP \times \sigma_{color \neq 'rojo'}(P))$$

En SQL (no vale en SQL Server)

```
SELECT S.sn,S.snombre,S.estado,S.ciudad
  FROM SP,P,S
 WHERE SP.pn=P.pn
```

AND SP.sn=S.sn

AND P.color='rojo'

MINUS

*SELECT S.sn,S.snombre,S.estado,S.ciudad
FROM SP,P,S
WHERE SP.bn=P.bn
AND SP.sn=S.sn
AND P.color<>'rojo'*

O bien:

SELECT sp.sn

*from
(SELECT sp.sn sn2, count(p.color) N
from sp, p
WHERE sp.bn=p.bn
group by sp.sn having count(*)=1) uncolor,
SP, P
WHERE sp.bn=p.bn and
p.color='Rojo' and
sp.sn=uncolor.sn2*

O bien:

*SELECT **

*FROM S
WHERE EXISTS (SELECT s2.sn, s2.snombre, p.bn FROM P, S S2, SP
WHERE p.color='Rojo'
and s2.bn=sp.bn
and sp.bn=p.bn
and s2.bn=s.bn)
and NOT EXISTS (SELECT s2.bn, s2.snombre, p.bn FROM P, S S2, SP
WHERE p.color<>'Rojo'
and s2.bn=sp.bn
and sp.bn=p.bn
and s2.bn=s.bn)*

O bien:

*SELECT S.**

FROM S

*WHERE sn IN (SELECT SP.bn FROM SP, P
WHERE SP.bn=P.bn
AND P.color='rojo')
AND sn NOT IN (SELECT SP.bn FROM SP, P
WHERE SP.bn=P.bn
AND P.color<>'rojo')*

10. Obtener el nombre de los proveedores que suministran, al menos, todas las piezas que se almacenan en la ciudad de Paris.

$$\Pi_{snombre}((\Pi_{sn, bn}(SP) \div \Pi_{bn}(\sigma_{ciudad='Paris'}(P))) \otimes S)$$

En SQL:

SELECT S.bn,S.snombre

```

FROM SP,P,S
WHERE SP.pn=P.pn
AND SP.sn=S.sn
AND P.ciudad='París'
GROUP BY S.sn,S.snombre
HAVING COUNT(*)=(SELECT COUNT(*) FROM P WHERE ciudad='París')

```

O bien:

```

SELECT sn,snombre
FROM S
WHERE EXISTS (SELECT * FROM SP
WHERE SP.sn=S.sn)
AND NOT EXISTS (SELECT * FROM P
WHERE P.ciudad='París'
AND NOT EXISTS (SELECT * FROM SP
WHERE SP.pn=P.pn
AND SP.sn=S.sn))

```

O bien:

```

SELECT sp.sn, s.snombre, count(*)
FROM SP ,P,S
WHERE P.CIUDAD='París'
AND SP.PN=P.PN
AND SP.SN=S.SN
GROUP BY sp.sn, s.snombre having count(*)=(SELECT count(P.pn)
FROM P
WHERE p.ciudad='París'
)

```

11. Obtener los datos del envío de más piezas. (Sólo en SQL)

```

SELECT * FROM SP
WHERE cant = (SELECT MAX(cant) FROM SP)

```

O bien:

```

/* ALL Compara un valor escalar con un conjunto de valores de una sola columna.*/
SELECT * FROM SP
WHERE cant >= ALL(SELECT cant FROM SP)

```

12. Para cada proveedor, mostrar la cantidad total de piezas que envía, la cantidad media y el número de envíos. (Sólo SQL)

```

SELECT sn, SUM(cant) cant_total, AVG(cant) cant_media, COUNT(*) num_envios
FROM SP
GROUP BY sn

```

13. Obtener el código de los proveedores que realizan envíos en cantidades superiores a la cantidad media por envío.

```

SELECT DISTINCT sn FROM SP
WHERE cant > (SELECT AVG(cant) FROM SP)

```

14. Para cada ciudad en la que se almacenan piezas, obtener el número de piezas que almacena de cada color distinto.

SELECT ciudad,color,COUNT() num_piezas FROM P
GROUP BY ciudad,color*

15. Todos los proveedores existentes y el número de productos que suministran.

*/*query 1*/
SELECT s.sn, count(sp.pn) NProductos
FROM s
LEFT JOIN SP ON s.sn=sp.sn
group by s.sn*

*/*query 6=query 1*/
SELECT s.sn, count(sp.pn)
FROM s, sp
WHERE s.sn *=sp.sn
group by s.sn*

16. De los proveedores que me suministran algún producto quiero saber cuantos productos distintos me suministran en total.

*/*query 2=query 3=query 4*/
SELECT s.sn, count(sp.pn)
FROM s,sp
WHERE s.sn=sp.sn
group by s.sn*

*/*query 3=query 2=query 4*/
SELECT s.sn, count(sp.pn)
FROM s JOIN sp ON s.sn=sp.sn
group by s.sn*

*/*query 4=query 3=query 2*/
SELECT s.sn, count(sp.pn)
FROM s INNER JOIN sp ON s.sn=sp.sn
group by s.sn*

Ejercicio 18. Ejercicio ciclismo.Pág. 116

El siguiente esquema relacional representa una base de datos que almacena información sobre una vuelta ciclista. Ningún atributo acepta nulos, a menos que se especifique lo contrario.

EQUIPO(namequipo, director)

Datos de los distintos equipos ciclistas que participan en la vuelta: nombre del equipo y nombre de su director.

CICLISTA(dorsal, nombre, añonacim, namequipo)

Datos de los ciclistas que componen los distintos equipos: número del dorsal, nombre del ciclista, año de nacimiento del ciclista y nombre del equipo al que pertenece.

CICLISTA.namequipo es clave ajena a EQUIPO; regla de borrado: propagar.

ETAPA(numetapa, kms, salida, llegada, dorsal)

Datos de las etapas que componen la vuelta ciclista: número de la etapa (las etapas se numeran consecutivamente: 1, 2, ...), kilómetros que tiene la etapa, nombre de la población de donde sale la etapa, nombre de la población donde se encuentra la meta de la etapa y número del dorsal del ciclista que ha ganado la etapa.

ETAPA.salida y ETAPA.llegada están definidas sobre el mismo dominio.

ETAPA.dorsal es clave ajena a CICLISTA; acepta nulos (aún no se ha corrido la etapa); regla de borrado: restringir.

PUERTO(nompuerto, altura, categoría, pendiente, numetapa, dorsal)

Datos de los puertos de montaña que visita la vuelta ciclista: nombre del puerto, altura máxima, categoría del puerto: primera, especial, etc., pendiente media del puerto, número de la etapa donde se pasa por él y número del dorsal que ha ganado el puerto al pasar en primera posición.

PUERTO.numetapa es clave ajena a ETAPA; regla de borrado: propagar.

PUERTO.dorsal es clave ajena a CICLISTA; acepta nulos (aún no se ha corrido la etapa que pasa por el puerto); regla de borrado: restringir.

MAILLOT(código, tipo, color, premio)

Datos de los premios que se otorgan mediante los distintos maillots: código del maillot, tipo de clasificación que premia ese maillot: general, montaña, etc., color de la camiseta asociada e importe del premio que corresponde al ciclista que termine la vuelta llevando el maillot.

LLEVA(código, numetapa, dorsal)

Información sobre qué ciclistas han llevado cada maillot en cada una de las etapas.

LLEVA.código es clave ajena a MAILLOT; no acepta nulos; regla de borrado: restringir.

LLEVA.numetapa es clave ajena a ETAPA; no acepta nulos; regla de borrado: propagar.

LLEVA.dorsal es clave ajena a CICLISTA; no acepta nulos; regla de borrado: restringir.





1. Obtener los datos de las etapas que pasan por algún puerto de montaña y que tienen salida y llegada en la misma población.

(ETAPA JOIN PUERTO[numetapa]) WHERE salida=llegada

La sentencia SQL que corresponde a esta expresión es la siguiente:

```
SELECT DISTINCT E.numetapa, E.kms, E.salida, E.llegada, E.dorsal
  FROM ETAPA E, PUERTO P
 WHERE E.numetapa = P.numetapa
   AND E.salida = E.llegada;
```

También podemos escribir:

```
SELECT *
  FROM ETAPA E
 WHERE E.llegada=E.salida
   AND EXISTS (SELECT * FROM PUERTO P
              WHERE E.numetapa = P.numetapa);
```

Se puede escribir una sentencia equivalente en la que se utilice el operador IN en lugar de EXISTS y no se necesite la referencia externa:

```
SELECT *
  FROM ETAPA
 WHERE llegada=salida
   AND numetapa IN (SELECT numetapa FROM PUERTO P);
```

2. Obtener las poblaciones que tienen la meta de alguna etapa, pero desde las que no se realiza ninguna salida.

ETAPA[llegada] MINUS ETAPA[salida]

La sentencia SQL que corresponde a esta expresión es la siguiente:

```
SELECT llegada FROM ETAPA
MINUS
SELECT salida FROM ETAPA;
```

En la siguiente sentencia se utiliza el operador NOT IN en lugar de la operación MINUS:

```
SELECT DISTINCT llegada
  FROM ETAPA
 WHERE llegada NOT IN (SELECT Salida FROM ETAPA);
```

También:

```
SELECT llegada
  FROM ETAPA ETAPAX
 WHERE NOT EXISTS (SELECT * FROM ETAPA ETAPAY
                   WHERE ETAPAY.salida=ETAPAX.llegada);
```

Una sentencia equivalente que no utiliza referencia externa es la siguiente:

```
SELECT llegada FROM ETAPA
 WHERE llegada <> ALL (SELECT salida FROM ETAPA);
```

3. Obtener el nombre y el equipo de los ciclistas que han ganado alguna etapa llevando el maillot amarillo, mostrando también el número de etapa.

T1 := (ETAPA JOIN LLEVA JOIN
(MAILLOT WHERE color='amarillo'))[dorsal,numetapa]
T2 := (T1 JOIN CICLISTA)[dorsal,nombre, nomequipo, numetapa]

La sentencia SQL que corresponde a esta expresión es la siguiente:

```
SELECT C.dorsal,C.nombre,C.nomequipo,E.numetapa   FROM CICLISTA C,ETAPA E,LLEVA
LL,MAILLOT M
 WHERE E.dorsal=LL.dorsal AND E.numetapa=LL.numetapa
      AND LL.codigo=M.codigo AND M.color='amarillo'
      AND E.dorsal=C.dorsal;
```

También:

```
SELECT C.dorsal,C.nombre,C.nomequipo,E.numetapa FROM CICLISTA C,ETAPA E
 WHERE E.dorsal=C.dorsal
      AND EXISTS (SELECT * FROM LLEVA LL,MAILLOT M WHERE E.dorsal=LL.dorsal AND
E.numetapa=LL.numetapa AND LL.codigo=M.codigo
      AND M.color='amarillo');
```

4. Obtener los datos de las etapas que no comienzan en la misma ciudad en que acaba la etapa anterior.

T1 := ETAPA
RDO := ((Ti TIMES ETAPA) WHERE
ETAPA.numetapa=T1.numetapa+1
 AND ETAPA.salida<>T1.1legada)
[ETAPA.numetapa,ETAPA.kms,ETAPA.salida,ETAPA.11egada,ETAPA.dorsal]

La sentencia SQL que corresponde a esta expresión es la siguiente:

```
SELECT ETAPAX.numetapa,ETAPAX.kms,ETAPAX.salida,
      ETAPAX.llegada,ETAPAX.dorsal
     FROM ETAPA ETAPAX, ETAPA ETAPAY
    WHERE ETAPAX.numetapa=ETAPAY.numetapa+1
      AND ETAPAX.salida<>ETAPAY.11egada;
```

La sentencia SQL que corresponde a esta expresión es a siguiente:

```

SELECT *
  FROM ETAPA ETAPAX
 WHERE EXISTS (SELECT * FROM ETAPA ETAPAY
              WHERE ETAPAX.numetapa=ETAPAY.numetapa+
                AND ETAPAX.salida<>ETAPAY.11egada);

```

5. Obtener el número de las etapas que tienen algún puerto de montaña, indicando cuantos tiene cada una de ellas.

La sentencia SQL que corresponde a esta expresión es la siguiente:

```

SELECT numetapa, COUNT(*) AS num_puertos
  FROM PUERTO
 GROUP BY numetapa;

```

No se puede resolver en el cálculo relacional.

6. Obtener el nombre y la edad de los ciclistas que han llevado dos o más maillots en una misma etapa.

La sentencia SQL que corresponde a esta expresión es a siguiente:

```

SELECT dorsal,nombre,edad
  FROM CICLISTA C
 WHERE EXISTS (SELECT * FROM LLEVA LLX, LLEVA LLY
               WHERE C.dorsal=LLX.dorsal
                 AND C.dorsal=LLY.dorsal
                 AND LLX.numetapa=LLY.numetapa
                 AND LLX.codigo<>LLY.codigo);

```

7. Obtener los datos de los ciclistas que han vestido todos los maillots (no necesariamente en la misma etapa).

```

SELECT C.dorsal,C.nombre,C.edad,C.nomequipo
  FROM CICLISTA C,T,JgVA LL
 WHERE C.dorsal=LL.dorsal
 GROUP BY C.dorsal,C.nombre,C.edad,C.nomequipo
 HAVING COUNT(DISTINCT LL.codigo)=
        (SELECT COUNT(*) FROM MAILLOT);

```

ó

```

SELECT C.dorsal,C.nombre,C.edad,C.nomequipo
  FROM CICLISTA C
 WHERE NOT EXISTS (SELECT * FROM MAILLOT M
                   WHERE NOT EXISTS
                     (SELECT *
                      FROM LLEVA LL
                      WHERE M.codigo=LL.codigo
                        AND LL.dorsal=C.dorsal));

```

8. Obtener el código y el color de aquellos maillots que solo han sido llevados por ciclistas de un mismo equipo.

```

SELECT LL.codigo,M.color
  FROM LLEVA LL, CICLISTA C, MAILLOT M
 WHERE LL.dorsal=C.dorsal
   AND LL.codigo=M.codigo
 GROUP BY LL.codigo,M.color

```

HAVING COUNT(DISTINCT C.nomequipo)=1;

Ó

```
SELECT DISTINCT LLX.codigo,M.color
  FROM LLEVA LLX, CICLISTA C, MAILLOT M
 WHERE LLX.dorsal=C.dorsal
   AND LLX.codigo=M.codigo
   AND C.nomequipo=ALL (SELECT C.nomequipo
                           FROM LLEVA LLY, CICLISTA C
                          WHERE LLY.dorsal=C.dorsal
                            AND LLY.codigo=LLX.codigo);
```

9. Obtener los números de las etapas que no tienen puertos de montaña.

(No se puede con SQL Server)
 SELECT numetapa FROM ETAPA
 MINUS
 SELECT numetapa FROM PUERTO;

En la siguiente sentencia se utiliza el operador NOT IN en lugar de la operación MINUS:

```
SELECT numetapa
  FROM ETAPA
 WHERE numetapa NOT IN (SELECT numetapa FROM PUERTO);
```

10. Obtener la edad media de los ciclistas que han ganado alguna etapa

```
SELECT AVG(C.edad)
  FROM CICLISTA C, ETAPA E
 WHERE C.dorsal=E.dorsal;
```

11. Obtener el nombre de los puertos de montaña que tienen una altura superior a la altura media de todos los puertos.

```
SELECT nompuerto FROM PUERTO
 WHERE altura > (SELECT AVG(altura)
                  FROM PUERTO);
```

12. Obtener las poblaciones de salida y de llegada de las etapas donde se encuentran los puertos con mayor pendiente.

```
SELECT DISTINCT E.numetapa, E.salida, E.11egada
  FROM ETAPA E,PUERTO P
 WHERE E.numetapa=P.numetapa
   AND P.pendiente = (SELECT MAX(pendiente) FROM PUERTO);
```

Ó

```
SELECT E.numetapa, E.salida, E.11egada
  FROM ETAPA E
 WHERE EXISTS (SELECT * FROM PUERTO PX
                  WHERE E.numetapa=PX.numetapa AND
                    NOT EXISTS (SELECT * FROM PUERTO PY
                                WHERE PY.pendiente>PX.pendiente)); 0:
```

Ó

```
SELECT DISTINCT E.numetapa, E.salida, E.11egada
  FROM ETAPA E, PUERTO P
 WHERE E.numetapa=P.numetapa
   AND P.pendiente ALL(SELECT pendiente FROM PUERTO);
```

13. Obtener el dorsal y el nombre de los ciclistas que han ganado los puertos de mayor altura.

```
SELECT DISTINCT C.dorsal,C.nombre
  FROM PUERTO P,CICLISTA C
 WHERE P.dorsal=C.dorsal
   AND P.altura = (SELECT MAX(altura) FROM PUERTO);
```

Ó

```
SELECT DISTINCT C.dorsal,C.nombre
  FROM PUERTO P,CICLISTA C
 WHERE EXISTS (SELECT * FROM PUERTO PX
              WHERE PX.dorsal=C.dorsal
                AND NOT EXISTS (SELECT * FROM PUERTO P
                               WHERE PY.altura>PX.altura));
```

Ó

```
SELECT DISTINCT C.dorsal,C.nombre
  FROM PUERTO P,CICLISTA
 WHERE P.dorsal=C. dorsal
   AND P.altura >= ALL(SELECT altura FROM PUERTO);
```

14. Obtener los datos de las etapas cuyos puertos (todos) superan los 1300 metros de altura.

```
SELECT E. *
  FROM PUERTO P, ETAPA E
 WHERE P.numetapa=E.numetapa
   AND P.altura>1300
MINUS
SELECT E. *
  FROM PUERTO P, ETAPA E
 WHERE P.numetapa=E.numetapa
   AND P.altura<=1300;
```

En la siguiente sentencia se utiliza el operador NOT IN en lugar de la operación MINUS:

```
SELECT DISTINCT E. *
  FROM PUERTO P, ETAPA E
 WHERE P.numetapa=E.numetapa
   AND P.altura>1300
   AND E.numetapa NOT IN (SELECT numetapa FROM PUERTO
                           WHERE altura<=1300);
```

ó

```
SELECT
  FROM ETAPA E
 WHERE EXISTS (SELECT * FROM PUERTO P
               WHERE P.numetapa=E.numetapa)
   AND NOT EXISTS (SELECT * FROM PUERTO P
```

WHERE P.numetapa=E.numetapa
AND P.altura<=1300);

Ó

```
SELECT *
  FROM ETAPA E
 WHERE EXISTS (SELECT * FROM PUERTO P
                  WHERE P.numetapa=E.numetapa)
                  AND 1300 < ALL (SELECT P.altura
                                    FROM PUERTO P
                                    WHERE P.numetapa=E.numetapa);
```

El predicado EXISTS es necesario porque, cuando una etapa no tiene puertos de montaña, la subconsulta del operador ALL no devuelve ninguna fila y la expresión $1300 < \text{ALL}(\text{null})$ se evalúa a falso; es por ello que se debe mantener la restricción de que la etapa tenga algún puerto de montaña.

El predicado EXISTS es equivalente a la siguiente expresión, en la que no se utiliza referencia externa en la subconsulta:

WHERE numetapa IN (SELECT numetapa FROM PUERTO P)

El predicado de la consulta anterior:

$1300 < \text{ALL} (\text{SELECT P.altura FROM PUERTO P
 WHERE P.numetapa=E.numetapa})$

es equivalente a este otro predicado:

$1300 < (\text{SELECT MIN(P.altura) FROM PUERTO P WHERE P.numetapa=E.numetapa})$

En este caso, cuando una etapa no tiene puertos, la subconsulta devuelve null y la expresión $1300 < \text{null}$ se evalúa a falso, por lo que esta etapa no se obtiene en el resultado de la consulta principal. Esto permite prescindir del predicado EXISTS (o el equivalente NOT IN), obteniéndose la siguiente sentencia SQL:

```
SELECT
  FROM ETAPA E
 WHERE 1300 < (SELECT MIN(P.altura)
                  FROM PUERTO P
                  WHERE P.numetapa=E.numetapa);
```

15. Obtener el nombre de los ciclistas que pertenecen a un equipo de más de cinco ciclistas y que han ganado alguna etapa, indicando también cuantas etapas han ganado.

```
SELECT C.dorsal,C.nombre,COUNT(*) AS num etapas ganadas
  FROM CICLISTA C,ETAPA E
 WHERE C.dorsal=E.dorsal
     AND C.nomequipo IN (SELECT nomequipo
                           FROM CICLISTA GROUP BY Nomequipo
                           HAVING COUNT(*) > 5)
 GROUP BY C.dorsal,C.nombre;
```

Ejercicio 19. Pág. 121

```
select * into #tempPubs from [pubs].[dbo].[authors]
```

```

select * from #tempPubs

BEGIN TRANSACTION Trans

update #tempPubs
set contract=1
where state='CA'

select * from #tempPubs

ROLLBACK TRANSACTION Trans

select * from #tempPubs

```

Ejercicio 20. Hacer el anterior ejemplo sin utilizar CONTINUE ni BREAK. Pág. 131

```

DECLARE @aux smallint, @npares smallint
SELECT @aux = 1
SET @npares = 0
WHILE (@aux<10)
begin
    if @aux<>5
        IF (@aux- (@aux/2)*2)=1
            BEGIN
                print(convert(varchar,@aux) +' es impar')
            END
        ELSE
            BEGIN
                print(convert(varchar,@aux) +' es par')
                SET @npares = @npares + 1
                IF @npares=3
                    set @aux=10
            END
    SET @aux=@aux+1
END

```

Ejercicio 21. Realizar un programa que muestre la tabla de códigos ASCII. Pág. 131

Ejercicio 23 (pág. 132)

use pubs

```

DECLARE @numeroMax smallint, @pp varchar(50)
SELECT @numeroMax= (SELECT max(valor) mAXIMO
                     FROM (SELECT count(*) valor
                           FROM authors
                          GROUP by state) SUBCONSULTA)
select @pp=
case
when @numeroMax>10 THEN 'Existen muchos coches'
when @numeroMax=10 THEN 'Ni muchos ni pocos'

```

```

ELSE 'Existen pocos coches'
end
print @pp
print 'Final Programa'
```

ó

```

use pubs
DECLARE @numeroMax smallint
SELECT @numeroMax= (SELECT max(valor) mAXIMO
                      FROM (SELECT count(*) valor
                            FROM authors
                           GROUP by state) SUBCONSULTA)
select 'Mensaje'=
case
when @numeroMax>10 THEN 'Existen muchos coches'
when @numeroMax=10 THEN 'Ni muchos ni pocos'
ELSE 'Existen pocos coches'
end
```

ó

```

DECLARE @letra smallint, @numero smallint
SELECT @letra = 65
SET @numero = 1

WHILE (@letra<85)
begin
    WHILE (@numero<11)
    begin
        print (char(@letra)+ str(@numero) )
        SET @numero=@numero+1
    END
    SET @letra=@letra+1
    SET @numero = 1
end
```

Ejercicio 25 (PÁG.134)

```

--DROP FUNCTION trienios
GO
CREATE FUNCTION trienios  (@fech1 smalldatetime, @fecha2 smalldatetime )
RETURNS int
AS
BEGIN
    DECLARE @RESUL int
    DECLARE @año1 int
    DECLARE @mes1 int
    DECLARE @AÑOS numeric(6,2)
```

```

SET @AÑOS=convert(decimal,@fecha2-@fecha1)/365.25
SET @RESUL=@AÑOS/3
RETURN (@RESUL)
END
GO
-- llamada a la funcion
SELECT dbo.trienios('01/09/2002','31/10/2008')
O
alter FUNCTION Trienios (@primer datetime , @segundo datetime)
RETURNS integer
AS
begin
RETURN (abs(datediff(qq,@primer,@segundo)))
end

select dbo.trienios('1-1-1976','1-4-1970')

```

Ejercicio 27 (pág. 134).

Sol. Oracle:

```

create or replace function pag2719(cad varchar2)
return varchar2
as
aux varchar2(20);
aux2 varchar2(20);
caracter char;
i number;
begin
aux:=cad;
aux2:='';
i:=1;
while i <=length(aux) loop
caracter:=substr(aux,i,1);
i:=i+1;
if (caracter between 'A' and 'Z' or caracter between 'a' and
'z') then
aux2:=aux2||caracter;
else
aux2:=aux2||' ';
end if;
end loop;
return aux2;
end;

```

Ejercicio 28 (pág. 134)

```

create function consistencia_datos()
returns varchar (30)
begin

```

```

DECLARE @CORRECTO VARCHAR(20)
SET @CORRECTO='CORRECTO'
declare estar cursor
for
select distinct centro,cod_respon_daño1

```

from base_averias

```

open estar
declare @centro varchar(9)
declare @cod_respon_daño1 varchar(2)
FETCH NEXT from estar INTO @centro,@cod_respon_daño1
while
    (@@FETCH_STATUS =0) AND @CORRECTO='CORRECTO'
begin
    if (SELECT @centro) in (SELECT CODIGO FROM DBO.CENTROS)
        and (SELECT @cod_respon_daño1) in (SELECT cod_resp_daño from
dbo.RESPONSABLE_DAÑO)
        set @CORRECTO='CORRECTO'
    ELSE
        set @CORRECTO='INCORRECTO'
    FETCH NEXT from estar INTO @centro,@cod_respon_daño1
end

```

```

CLOSE ESTAR
DEALLOCATE ESTAR
RETURN @CORRECTO

```

End

Óoooooo

```

create function compara()
returns nvarchar (25)
as
begin
declare @centro varchar(9)
declare @cod_respon_daño1 nvarchar(3)
declare @resultado nvarchar(25)
declare cursor_CentBase cursor for
select Centro, cod_respon_daño1 from base_averias
open cursor_CentBase
declare @contador int
set @contador=0
fetch next from cursor_CentBase into @centro, @cod_respon_daño1
while @@fetch_status=0
begin
declare @xx varchar(9)
declare @yy varchar(3)
set @xx=(SELECT @centro where @centro in (select CODIGO from dbo.CENTROS))

```

```

if @xx is null set @contador=@contador+1
else
    set @yy=(SELECT @cod_respon_daño1 where @cod_respon_daño1 in (select
cod_resp_daño from dbo.responsable_daño))
    if @yy is null set @contador=@contador+1
        fetch next from cursor_CentBase into @centro, @cod_respon_daño1
    end
    if @contador>0 set @resultado='no están todos'
    else set @resultado='están todos'
    close cursor_CentBase
return (@resultado)
end

select [Averias].[dbo].[compara]()

--drop function compara
--deallocate cursor_CentBase

```

Ejercicio 29 pág.137

```

USE case
GO
ALTER PROCEDURE empleado
    @num_emple int,
    @num_depto int
AS
BEGIN
    UPDATE #temp_emple
    SET dept_no=@num_dept
    WHERE emp_no=@num_emple
END
GO

EXEC empleado 7369,11

```

Ejercicio 30(pág. 137)

Ejercicio 31 (pág. 137)

Solución Oracle:

```

CREATE OR REPLACE PROCEDURE Ejercicio2a (CADENA VARCHAR2)
IS
    I NUMBER (2);
    CADENA2 VARCHAR2 (20);

BEGIN

```

```
I:=LENGTH (CADENA) ;

WHILE I>=1           LOOP

CADENA2 :=CADENA2 || SUBSTR (CADENA, I, 1) ;
I:=I-1;

END LOOP;

DBMS_OUTPUT.put_line (CADENA2);

END;
```

Ejercicio 32 (pág. 145)

La solución buena irá con lo siguiente:

```
DECLARE @salida as int
DECLARE @cadena as nvarchar(100)
EXEC SP_EXECUTESQL N'SELECT @cantidad = COUNT(*) FROM emple WHERE apellido =
@cadena',
N'@cantidad AS INT OUTPUT, @cadena AS NVARCHAR(100) ',
@cantidad = @salida OUTPUT,
@cadena = 'SANCHEZ'
PRINT @salida
```

La siguiente solución iría con tablas temporales

```
ALTER PROCEDURE ejercicio32
@p1 varchar(30)
AS
BEGIN
    DECLARE datos CURSOR FOR
    SELECT emp_no,apellido FROM EMPLE
    where apellido=@p1;
    declare @vemp_no numeric(4,0),@vapellido nchar(10)

    OPEN datos
    FETCH datos into @vemp_no,@vapellido
    while (@@fetch_status=0)
    BEGIN
        SELECT convert(nchar(10),@vemp_no) + @vapellido
        FETCH datos into @vemp_no,@vapellido
    END
    CLOSE datos
    DEALLOCATE datos

END

exec ejercicio32 'SANCHEZ'

o
USE PUBS
GO
alter PROCEDURE CONTAR_APE @APE_BUSCA VARCHAR(30)
```

AS

```

DECLARE @NUMERO INT
DECLARE @APELIDO VARCHAR(30)
DECLARE CursorApe CURSOR
FOR
    SELECT lname
    FROM DBO.EMPLOYEE
    WHERE LNAME LIKE '%' + RTRIM(@APE_BUSCA) + '%'
BEGIN
    SET @NUMERO=0
    OPEN CursorApe
    FETCH NEXT FROM CursorApe INTO @APELIDO
    WHILE (@@FETCH_STATUS = 0)
    BEGIN
        SET @NUMERO=@NUMERO+1
        print('EL APELLIDO ' + @APELIDO + ' APARECE ' +
CONVERT(VARCHAR(10),@numero))
        FETCH NEXT FROM CursorApe INTO @APELIDO
    END
    CLOSE CursorApe
    DEALLOCATE CursorApe
END
GO

SELECT * FROM EMPLOYEE
EXEC CONTAR_APE 'Afonso'
EXEC CONTAR_APE 'P'
```

O

```

-- Merche
use case
drop procedure Ejercicio32
CREATE PROCEDURE Ejercicio32
    @apellido nchar(10)
AS

DECLARE CursorEjer32 INSENSITIVE CURSOR
FOR
    SELECT apellido from emple
    where apellido=@apellido
open CursorEjer32
print 'número de empleados cuyo apellido es:' +
@apellido+ ' '+convert(varchar(5),@@CURSOR_ROWS)
CLOSE CursorEjer32
deallocate CursorEjer32

exec dbo.Ejercicio32 'SANCHEZ'
```

Ejercicio 33 (pág. 148)

```
DECLARE Cursoremp CURSOR
```

```

FOR
SELECT salario FROM emple

OPEN Cursoremp
DECLARE @num smallint, @salario smallint, @sal smallint, @media smallint
set @num=0
set @sal=0
FETCH NEXT FROM Cursoremp INTO @salario

WHILE (@@FETCH_STATUS =0)
    BEGIN
        set @num=@num+1
        set @sal=@sal+@salario
        FETCH NEXT FROM Cursoremp INTO @salario
    END
CLOSE Cursoremp
set @media=@sal/@num
--print @sal
--print @num
print @media
DEALLOCATE Cursoremp

```

Ejercicio 34 (pág. 148)

```

DECLARE Cursoremp scroll CURSOR
FOR
SELECT sal FROM emp
OPEN Cursoremp
DECLARE @num smallint, @salario smallint, @sal smallint, @media smallint
set @num=1
set @sal=0
FETCH absolute @num FROM Cursoremp INTO @salario
WHILE (@@FETCH_STATUS =0)
    BEGIN
        set @num=@num+1
        set @sal=@sal+@salario
        FETCH absolute @num FROM Cursoremp INTO @salario
    END
CLOSE Cursoremp
set @media=@sal/(@num-1)
--print @sal
--print @num
print @media
DEALLOCATE Cursoremp

```

Ejercicio 35. (pág. 148)

```

alter procedure Ejercicio35
as
declare @al varchar(10), @s1 numeric(7,0), @of1 nchar(10), @i
integer, @vaux nchar(10)

```

```

DECLARE CursorEmpleados CURSOR FAST_FORWARD FOR
    SELECT Apellido, Salario, oficio
        FROM Emple order by oficio, salario
open cursorEmpleados
fetch next from cursorEmpleados into @al, @s1, @of1
set @vaux=@of1
while (@@fetch_status=0)
begin
    set @vaux=@of1
    set @i=0
    while (@vaux=@of1 and @@fetch_status=0)
        begin
            if (@i<2)
            begin
                print @al+' '+@of1+' '+ltrim(str(@s1))
            end
            set @i=@i+1
            fetch next from cursorEmpleados into @al, @s1, @of1
        end
    end
close cursorEmpleados
deallocate cursorEmpleados

--exec Ejercicio35

```

```

/* Con una query */
select t.oficio, salario from emple,
((select distinct min(salario)minidos,oficio from emple
where salario not in(select min(salario)mini from emple group by oficio)group
by oficio)
union (select distinct min(salario)mini,oficio from emple group by oficio))t
where emple.oficio=t.oficio and emple.salario=t.minidos;

```

Otra opción con cursos anidados:

```

alter procedure Ejercicio35
as
declare @al varchar(10), @s1 numeric(7,0), @of1 nchar(10),@i integer, @vaux
nchar(10)
DECLARE CursorOficios CURSOR FOR
    SELECT DISTINCT oficio FROM Emple
open CursorOficios
fetch next from CursorOficios into @of1
--set @vaux=@of1
while (@@fetch_status=0)
begin
    --set @vaux=@of1
    set @i=0
    DECLARE CursorSalarios CURSOR FOR
        SELECT apellido,salario,oficio FROM Emple WHERE oficio=@of1
        order by salario

```

```

open CursorSalarios
fetch next from CursorSalarios into @a1,@s1,@of1
while (@i<2 and @@fetch_status=0)
begin
    print @a1+' '+@of1+' '+ltrim(str(@s1))
    set @i=@i+1
    fetch next from CursorSalarios into @a1, @s1, @of1
end
close CursorSalarios
deallocate CursorSalarios
fetch next from CursorOficios into @of1
end
close CursorOficios
deallocate CursorOficios

--exec Ejercicio35

```

Ejercicio 36 Pág. 148

```

alter procedure Ejercicio36
as
declare @s1 numeric(7,0),@s2 numeric(7,0)
DECLARE CursorEmpleados CURSOR FOR
    SELECT Salario
        FROM Emple
        for update
open cursorEmpleados
fetch next from cursorEmpleados into @s1
set @s2=(select avg(salario)*1.015 from emple)
while (@@fetch_status=0)
begin
    update emple
    set salario=case
        when salario<@s2 then @s2
        else
            @s1
    end
    where current of cursorEmpleados
    fetch next from cursorEmpleados into @s1
end
close cursorEmpleados
deallocate cursorEmpleados

--exec Ejercicio36

```

Ejercicio 37 Pág. 151

Solución Oracle:

```

eclare
    i number ;
    maximo number;

```

```

apellidos varchar2(1000);
aux2 varchar2(1000);
excepciones number;
begin
    i:=7368;
    excepciones:=0;
    apellidos:='';
    aux2:='';
    select max(emp_no) into maximo from emple;
    while i<=maximo
    loop
        begin --Bloque hijo que trata la excepcion
            select apellido into aux2 from emple where emp_no=i;
            apellidos:=apellidos||','||aux2;
            i:=i+1;
        exception
            when no_data_found then
                i:=i+1;           excepciones:=excepciones+1;
            end;
        end loop;
        apellidos:=Substr(apellidos,2,length(apellidos));
        dbms_output.put_line(apellidos);
        dbms_output.put_line('Se han producido ' || excepciones || '
excepciones');
    end;

```

Ejercicio 38 (pág. 151)

```

ALTER PROCEDURE AltaPedidos @Pedido_No numeric(4,0),
    @Producto_No numeric(4,0),
    @Cliente_No numeric(4,0),
    @Unidades numeric(4,0),
    @Fecha_Pedido smalldatetime
AS
--set @Pedido_N0=15
DECLARE @registros int
----- Verificación de existencia
    SELECT @registros= count(Producto_No) from Productos08 where
    Producto_No=@Producto_No

    if (@registros=0)
    begin
        RAISERROR('No existe ese producto',16,1)
    end
    SELECT @registros= count(Cliente_No) from Clientes08 where
    Cliente_No=@Cliente_No

    if (@registros=0)
    begin
        RAISERROR('No existe ese Cliente',16,1)
    end
    if (@Fecha_Pedido=NULL)
    begin
        set @Fecha_Pedido=convert(smalldatetime,getdate())
    end

```

```
end
```

----- Verificación condiciones de pedido

```
DECLARE @numUnidades numeric(8,0), @precio numeric(8,0),
        @debe numeric(9,0), @limite_credito numeric(9,0)

SELECT @numUnidades=stock_disponible, @precio=precio_Actual from
Productos08 where Producto_No=@Producto_No

if (@numUnidades<@Unidades)
begin
    RAISERROR('No hay suficientes unidades',16,1)
end

SELECT @debe=debe, @limite_Credito=Límite_Credito from clientes08
where cliente_no=@cliente_no

if ((@debe+(@unidades*@precio)>@limite_credito))
begin
    RAISERROR('No tiene credito el cliente',16,1)
end
```

----- Actualizaciones

```
begin transaction transl
update Clientes08 set debe = (@debe+(@unidades*@precio)) where
cliente_no=@cliente_no
update Productos08 set stock_disponible = @numUnidades-@unidades where
Producto_no=@producto_no
insert into Pedidos08
(Pedido_no, Producto_no, cliente_no, unidades, fecha_pedido) values (@pedido_no,
@producto_no, @cliente_no, @unidades, @fecha_pedido)
commit transaction transl
```

```
begin try
```

```
---- pedido producto cliente unidades fecha
exec AltaPedidos 16, 10, 101,1, '15/12/2005'
end try
```

```
begin catch
    Select error_message()
end catch
```

Ejercicio 39 (pág. 154)

```
USE [prueba]
GO
SET ANSI_NULLS ON
GO
```

```

SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[tmp] (
    [campo1] [nchar](1) NULL,
    [campo2] [nchar](20) NULL
) ON [PRIMARY]

USE TMP
INSERT INTO TMP (CAMPO2) VALUES ('XXX')

SELECT * FROM TMP
DELETE TMP

DROP TRIGGER inserta_CAMPO1

ALTER TRIGGER inserta_CAMPO1
ON TMP
INSTEAD OF INSERT, UPDATE
AS
BEGIN
DECLARE @LETRA CHAR(1)
DECLARE @NUM INT
SET @LETRA=(SELECT MAX(CAMPO1) FROM TMP)
    IF @LETRA IS NULL -- caso en el que no hay registros aún
    BEGIN
        INSERT INTO TMP (CAMPO1,CAMPO2)
        SELECT 'a',CAMPO2
        FROM inserted
    END
    ELSE
    BEGIN
        SET @num = ASCII(@letra)
        SET @num=@num+1
        INSERT INTO TMP (CAMPO1,CAMPO2)
        SELECT CHAR(@num),CAMPO2
        FROM inserted
    END
END
GO

```

Ejercicio 40 (pág. 154)

```

CREATE PROCEDURE actualiza_letras
    @letraborrada varchar(3)
AS
BEGIN
    UPDATE letras SET letra=char(ascii(letra)-1)
    WHERE letra>=@letraborrada
END

```

drop procedure actualiza_letras

```

SELECT * FROM letras
order by letra

```

```
SELECT * FROM numeros
order by numero
```

```
delete letras where letra='g'
```

```
CREATE TRIGGER inserta_campo_letra
ON letras
INSTEAD OF INSERT
AS
BEGIN
DECLARE @LETRA INT
DECLARE @NUM VARCHAR(3)
SET @NUM=(SELECT MAX(letra) FROM letras)
IF @NUM IS NULL
BEGIN
    INSERT INTO letras (letra,numero)
    SELECT 'a',numero
    FROM inserted
END
ELSE
BEGIN
    SET @LETRA = ASCII(@NUM)
    SET @LETRA=@LETRA+1
    INSERT INTO letras (letra,numero)
    SELECT CHAR(@LETRA),numero
    FROM inserted
END
END
```

```
/* Borra de numeros el registro asociado a letras cuando se ha borrado de esta*/
ON letras
AFTER DELETE
AS
BEGIN
DECLARE @numeroborrado int
SET @numeroborrado=(SELECT numero from deleted)

DELETE numeros WHERE numero=@numeroborrado

DECLARE @letraborrada varchar(3)
SET @letraborrada=(SELECT letra from deleted)
```

EXEC actualiza_letras @letraborrada

END

drop trigger borra_numeros

Ejercicio 41 Pág. 162

```
use prueba
drop table historico
create table historico
(fecha datetime,
usuario NVARCHAR(100)
)

alter TRIGGER GuardaEnHistorico
ON ALL SERVER with execute as 'cursosql\Merche'
FOR LOGON
AS
BEGIN

    insert into prueba.dbo.historico values (getdate(),original_login())
    print ''

END;

use prueba
select * from historico
```

Ejercicio 42. Pág. 162

Ejercicio 52. Supuesto de copias de seguridad (pág. 200)

- A. Incorrecta: Una copia de seguridad diferencial a las 12.00 requeriría que se restaurasen más de seis copias de seguridad del registro de transacciones para recuperar la base de datos por la tarde.
- B. Correcta: Una copia de seguridad diferencial a las 13.00 aseguraría que no se necesitase restaurar una secuencia de más de seis registros de transacciones para recuperar la base de datos por la mañana o por la tarde.
- C. Incorrecta: Esta solución propone que se necesitaría restaurar una secuencia de ocho registros de transacciones para recuperar la base de datos por la tarde. Este número excede el límite establecido de seis.
- D. Correcta: Esta solución propone que se necesitaría restaurar una secuencia de no más de seis registros de transacciones para recuperar la base de datos por la mañana o por la tarde.

Ejercicio 53. (pág. 201)

- A. Correcta: Hay tiempo suficiente para realizar una copia de seguridad completa por la noche y una copia de seguridad diferencial entre las 12.00 y las 13.00. Una copia de seguridad diaria agilizaría el proceso de recuperación porque habría que restaurar menos cambios desde la última copia de seguridad completa.
- B. Incorrecta: No debe realizar ninguna copia de seguridad durante horas laborales, por lo que no puede realizar una copia de seguridad completa durante el día.
- C. Incorrecta: Un cambio al modelo de recuperación simple recorta el tiempo de recuperación, pero no tanto como aumentar la frecuencia de las copias de seguridad completas. Además, no se recomienda usar el modelo de recuperación simple para una base de datos de producción como Pedidos (cuya data seguramente se actualizará con mucha frecuencia).
- D. Incorrecta: No debe realizar ninguna copia de seguridad durante horas laborales, por lo que no puede realizar una copia de seguridad completa durante el día.
3. Respuesta correcta: B
- A. Incorrecta: Una copia de seguridad diferencial cada 60 minutos no reduce el tiempo necesario para copiar la información.
- B. Correcta: De las cuatro opciones, esta estrategia es la que más reduce el tiempo necesario para realizar una copia de seguridad de la información y cumplir el requisito de no perder más de 60 minutos de actividad de la base de datos.
- C. Incorrecta: Esta estrategia cumple el requisito de no perder más de 60 minutos de actividad de la base de datos, pero no es la que más reduce el tiempo necesario para copiar la información. Los registros de transacciones han sido programados cada 60 segundos (no sólo durante el día) y los tres tipos de copias de seguridad se realizan durante a diario.
- D. Incorrecta: Esta estrategia no es una opción válida porque no hay una copia de seguridad completa especificada.

Ejercicio 54 (pág. 201)

- A. Incorrecta: Sólo debería realizar este paso después de haber realizado una copia de seguridad de la cola (parte activa) del registro.
- B. Correcta: Si la base de datos no ha sido eliminada y el registro aún está disponible, debería hacer una copia de seguridad de la parte activa (cola) del registro antes de comenzar la secuencia de restauración. Esto asegura que toda la información será recuperada hasta el momento del fallo.
- C. Incorrecta: Este paso debería realizarse después de que se haya hecho una copia de seguridad de la cola del registro y después de que se haya restaurado la copia de seguridad completa.
- D. Incorrecta: Esta opción no forma parte de una secuencia de restauración.

Ejercicio 55. Recuperación de datos. (pág. 201)

- 1.
- A. Incorrecta: El modelo de recuperación simple no posibilita la recuperación de un punto en el tiempo.
- B. Incorrecta: El modelo de recuperación de registro masivo no posibilita la recuperación de un punto en el tiempo.
- C. Correcta: Sólo el modelo de recuperación completa posibilita una recuperación completa de un punto en el tiempo.
- D. Incorrecta: Todas las bases de datos están configuradas en uno de los tres modelos: simple, de registro masivo o completa.

2.

- A. Incorrecta: En el modelo de recuperación simple no se realizan las copias de seguridad del registro de transacciones, por lo que no puede aplicarlas. Además, esta secuencia de restauración está presentada en orden inverso.
- B. Incorrecta: En el modelo de recuperación simple no se realizan las copias de seguridad del registro de transacciones, por lo que no puede aplicarlas.
- C. Correcta: En el modelo de recuperación simple sólo puede realizar copias de seguridad completas y (opcionalmente) diferenciales. Para restaurar de forma adecuada estas copias de seguridad, restaure primero la última copia de seguridad completa y después la última copia de seguridad diferencial (si la hay) realizada desde la última copia de seguridad completa.
- D. Incorrecta: La secuencia de restauración de esta respuesta está en orden inverso.

3.

- A. Correcta: En los modelos de recuperación completa y de registro masivo, al realizar una recuperación ante desastres deben intentar primero hacer una copia de seguridad de la parte activa del registro especificando la opción NO_TRUNCATE. Si puede realizar esta copia de seguridad, puede restaurar la información hasta el punto de fallo.
- B. Incorrecta: Aplicar los registros de transacciones es el último paso en la secuencia de restauración para una base de datos configurada en el modelo de recuperación completa o de registro masivo.
- C. Incorrecta: Restaurar la última copia de seguridad completa es el primer paso en la secuencia de restauración para una base de datos configurada en el modelo de recuperación simple. Si utiliza la opción RECOVERY, se vuelve a conectar la base de datos sin restaurar ninguna copia de seguridad diferencial ni del registro de transacciones.
- D. Incorrecta: Restaurar la última copia de seguridad completa es el primer paso en la secuencia de restauración para una base de datos configurada en el modelo de recuperación simple. Si utiliza la opción NORECOVERY, no se vuelve a conectar la base de datos, por lo que puede restaurar las copias de seguridad diferenciales o del registro de transacciones.

4.

- A. Incorrecta: El modelo de recuperación de registro masivo no posibilita la recuperación de un punto en el tiempo.
- B. Incorrecta: Si se borra el registro activo, no se puede hacer una copia de seguridad de él. Por lo tanto, la base de datos sólo se puede recuperar hasta el punto de la última copia de seguridad válida.
- C. Correcta: Si falla una base de datos, puede poder realizar una copia de seguridad de la cola del registro, lo que le permite recuperar la base de datos hasta el punto de fallo.
- D. Incorrecta: Si una base de datos configurada en el modelo de recuperación de registro masivo falla durante una operación de copia de registro masivo, entonces no puede recuperar la base de datos hasta el punto de fallo.

5.

- A. Correcta: Como el servidor está en producción, debería probar la estrategia de restauración en un servidor que no esté en producción.
- B. Correcta: Como parte del plan de recuperación escrito, debería incluir un procedimiento paso a paso describiendo cómo restaurar el servidor en cuestión.
- C. Incorrecta: Si el servidor está en producción, es preferible no desconectarlo y como está

creando el plan de recuperación, debe comprobar su éxito en otro servidor por si no funciona.
D. Correcta: Una directiva de recuperación escrita debe incluir una lista de gente con la que poder contactar.

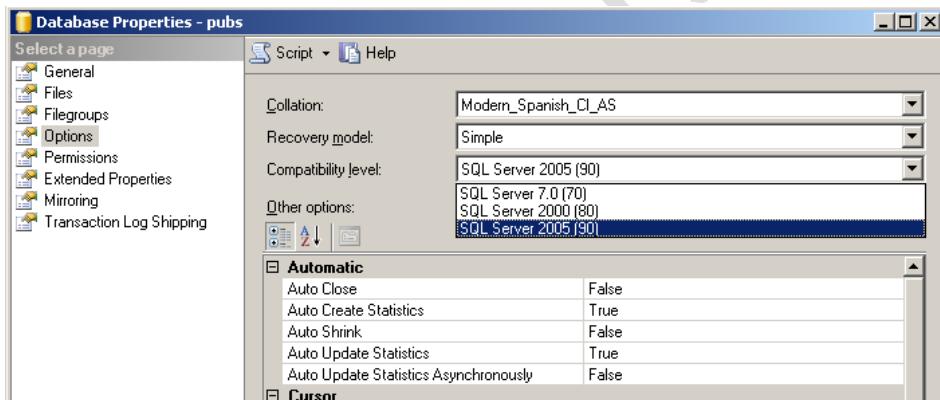
Módulo 12 ERRORES

a.- Cuando vamos a crear un diagrama sobre la base de datos pubs, ejemplo de reutilizado de SQL Server 2005, aparece el siguiente error:

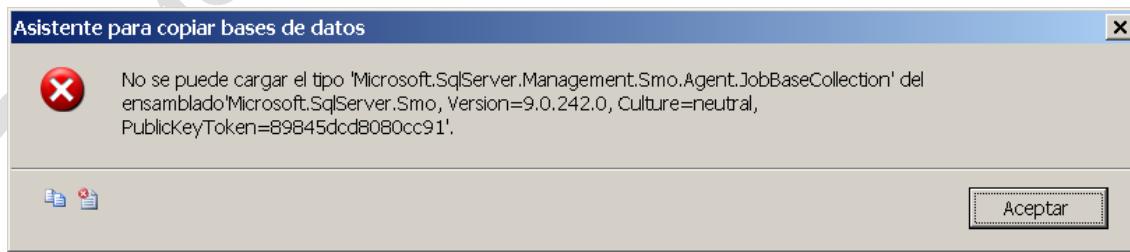


Solución:

En las propiedades de la base de datos cambiar el nivel de compatibilidad a SQL Server 2005 (90)

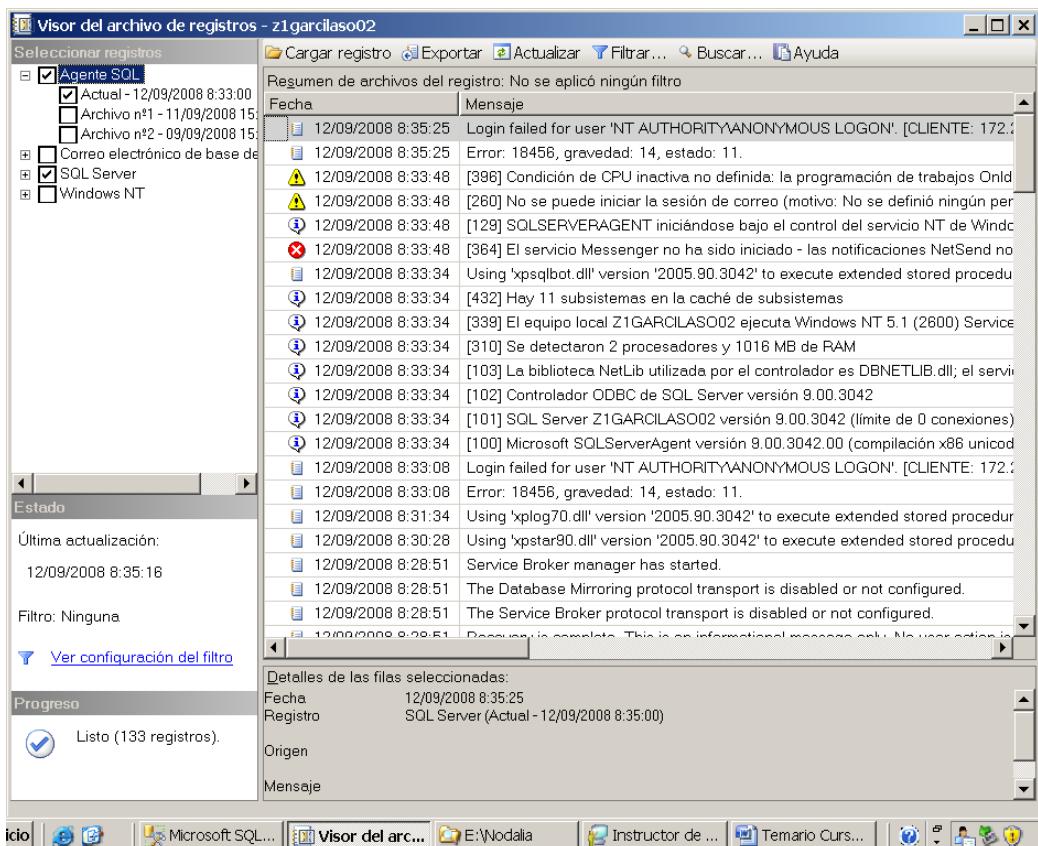


b) A la hora de realizar una copia/mover de una base de datos a otra, se produjo el siguiente error:



El sistema tenía SQL Server 2005 Developer o Enterprise en una instancia y en el mismo servidor SQL Server 2005 Express SP2 en otra instancia. Por lo visto, existían conflictos entre el SP2 de la express con el SQL Server 2005 sin SP2. Después de instalar el SP2 sobre la instancia del Developer no hubo problemas a la hora de copiar bases de datos.

Otro error al respecto:



Cuando metes el NT AUTHORITY\ANONYMOUS LOGON da el siguiente error:

The screenshot shows the Windows Event Viewer interface. The left pane displays a tree view of event sources, with 'Agentes SQL' selected. The right pane lists events in a table format. The first event is highlighted:

Fecha	Mensaje
12/09/2008 13:57:39	Login failed for user 'CURSOSQL\SRVCURSO\$'. [CLIENTE: 172.26.224.21]
12/09/2008 13:57:39	Error: 18456, gravedad: 14, estado: 11.
12/09/2008 11:27:41	Configuration option 'show advanced options' changed from 1 to 0. Run the RECONFIGURE command.
12/09/2008 11:27:41	Configuration option 'xp_cmdshell' changed from 1 to 1. Run the RECONFIGURE command.
12/09/2008 11:27:41	Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE command.
12/09/2008 11:23:34	Configuration option 'show advanced options' changed from 1 to 0. Run the RECONFIGURE command.
12/09/2008 11:23:34	Configuration option 'xp_cmdshell' changed from 1 to 1. Run the RECONFIGURE command.
12/09/2008 11:23:34	Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE command.
12/09/2008 11:23:21	Configuration option 'show advanced options' changed from 1 to 0. Run the RECONFIGURE command.
12/09/2008 11:23:21	Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE command.
12/09/2008 11:23:21	Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE command.
12/09/2008 8:35:25	Login failed for user 'NT AUTHORITY\ANONYMOUS LOGON'. [CLIENTE: 172.26.224.21]
12/09/2008 8:35:25	Error: 18456, gravedad: 14, estado: 11.
12/09/2008 8:33:48	[398] Condición de CPU inactiva no definida: la programación de trabajos Online.
12/09/2008 8:33:48	[260] No se puede iniciar la sesión de correo (motivo: No se definió ningún proveedor de correo electrónico).
12/09/2008 8:33:48	[128] SQLSERVERAGENT iniciándose bajo el control del servicio NT de Win32.
12/09/2008 8:33:48	[384] El servicio Messenger no ha sido iniciado - las notificaciones NetSend no se han recibido.
12/09/2008 8:33:34	Using 'xpsqlbot.dll' version '2005.90.3042' to execute extended stored procedure 'xp_startnetlib'.
12/09/2008 8:33:34	[432] Hay 11 subsistemas en la caché de subsistemas.
12/09/2008 8:33:34	[339] El equipo local Z1GARCLASO02 ejecuta Windows NT 5.1 (2600) Service Pack 2.
12/09/2008 8:33:34	[310] Se detectaron 2 procesadores y 1016 MB de RAM.
12/09/2008 8:33:34	[1021] La biblioteca NetLib utilizada por el controlador es DRIVENET. ID: null, el nombre es: null.

Detalles de las filas seleccionadas:

Fecha	12/09/2008 13:57:39
Registro	SQL Server (Actual - 12/09/2008 13:57:00)
Origen	
Mensaje	

Que es el equipo cliente.

En el servidor destino y origen con estos permisos sí funciona.

Destino:

Origen:

The screenshot shows the Object Explorer in SQL Server Management Studio. The left pane shows the database structure of the 'luzxpsqlexpress' database, including 'Bases de datos', 'Seguridad', and 'Funciones de sesión'. The right pane shows the detailed list of users under the 'Inicios de sesión' node. Some users listed include 'distributor_admin', 'DOMINIO\Administrador', 'DOMINIO\SQLServer2005MSFTEUser\$', 'DOMINIO\SQLServer2005MSSQLUser\$', 'DOMINIO\SQLServer2005SQLAgentUser\$', 'DOMINIO\Victor', 'DOMINIO\Victortarde', 'login_test', 'NT AUTHORITY\Servicio de red', 'NT AUTHORITY\SYSTEM', 'pepin', 'sa', and 'victor2'.

Módulo 13 BIBLIOGRAFÍA

[Borja, 2005] Borja Sotomayor <http://borja.casa-sotomayor.net/> Marzo 2005

[Charte, 2004] Francisco Charte Ojeda. “*Microsoft SQL Server 2000. Tema 1. Instalación.*” Ediciones Anaya Multimedia, Tercera impresión Mayo 2004.

[Charte, 2004b] Francisco Charte Ojeda. “*Microsoft SQL Server 2000. Tema 8. Introducción a Transact SQL.*” Ediciones Anaya Multimedia, Tercera impresión Mayo 2004.

[Charte, 2004c] Francisco Charte Ojeda. “*Microsoft SQL Server 2000. Tema 9. Aplicaciones de Transact-SQL.*” Ediciones Anaya Multimedia, Tercera impresión Mayo 2004.

[Charte, 2004d] Francisco Charte Ojeda. “*Microsoft SQL Server 2000. Tema 10. Mantenimiento de la BD.*” Ediciones Anaya Multimedia, Tercera impresión Mayo 2004.

[Cirilo, 2005] Ramón Cirilo “*Apuntes de Bases de datos I*” Universidad Valenciana, Marzo 2005.

[Dalton et al, 2001] Patrick Dalton, Paul Whitehead “*La biblia de SQL Server 2000*” Anaya 2001. *Tema 10. Nuevas características*

[Dalton et al, 2001b] Patrick Dalton, Paul Whitehead “*La biblia de SQL Server 2000*” Anaya 2001. *Tema 9. SQL*

[Dalton et al, 2001c] Patrick Dalton, Paul Whitehead “*La biblia de SQL Server 2000*” Anaya 2001. Pág. 228, 229.

[Dalton et al, 2001d] Patrick Dalton, Paul Whitehead “*La biblia de SQL Server 2000. Tema 14. Procedimientos almacenados y procedimientos almacenados externos.*” Anaya 2001.

[devjoker, 2008] <http://www.devjoker.com/contenidos/Tutorial-de-Transact-SQL/240/Cursores-en-Transact-SQL.aspx>

[Hernández, 2003] Carmen Hernández, profesora Universidad de Valladolid. “*Apuntes de asignatura Bases de datos*”. Año 2003

[Medina, 2005] Rafael Medina y Luis Najera. “SQL Server Microsoft” 00032721.ppt

[Maria Jesús, 2007] María Jesús Godos, profesora de la asignatura de CASE (2º ASI) del Centro Gregorio Fernández (Valladolid)

[Marqués, 2005] <http://nuvol.uji.es/%7Emmarques/f47/teoria/arcrsol.pdf> Merche Marqués. Mar-

zo 2005

[Pérez, 2003] César Pérez “*Domine Microsoft SQL Server 2000. Administración y análisis de Bases de Datos*” Ra-Ma, Año 2003

[Pérez, 2003b] César Pérez “*Domine Microsoft SQL Server 2000. Administración y análisis de Bases de Datos, Capítulo 6: elementos de transact-sql: formatos, tipos...*” pág. 262, Ra-Ma, Año 2003

[Pérez, 2003c] César Pérez “*Domine Microsoft SQL Server 2000. Administración y análisis de Bases de Datos, Capítulo 4: Análisis de la información y herramientas*” pág. 140, Ra-Ma, Año 2003

[Pérez, 2006] César Pérez “*Microsoft SQL Server 2005. Administración y Análisis de Bases de datos*, . Capítulo 5. Ver y modificar datos con el ordenador, pág. 204. .Ra-Ma, Año 2006.

[SQLServer2005,2008] J.C. Mackin & Mike Hotek. “*Diseño de una infraestructura de Servidor de Base de Datos-MCITP Examen 70-443*” Anaya

[trans, 2008]

Introducción a los registros de transacciones

<http://technet.microsoft.com/es-es/library/ms190925.aspx>

Registro de transacciones de escritura anticipada

<http://technet.microsoft.com/es-es/library/ms186259.aspx>

<http://www.todoexpertos.com/categorias/tecnologia-e-internet/bases-de-datos/sql-server/resuestas/1647511/truncar-base-de-datos>

13.1 Webs

Foros:

<http://www.configuracionesintegrales.com/miguel/default.asp?Tipo=S>

<http://www.clikear.com/sqlserver/>

<http://www.programacion.com/bbdd/foros/38/>

<http://www.sqlmax.com/func1.asp>

<http://usuarios.lycos.es/cursosgbd/UD5.htm>

<http://www.clikear.com/sqlserver/>