Transact SQL

INTRODUCCIÓN

Características

- Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional cuya principal función es la de almacenar y consultar datos solicitados por otras aplicaciones, sin importar si están en el mismo equipo, si están conectadas a una red local o si están conectadas a través de internet.
- SQL Server es algo mas que un SGBDR, es una plataforma de datos de misión critica, ofreciendo desarrollo dinámico e inteligencia de negocio, y va mas allá de la estructura de datos relacional.

Modelo relacional

- Basado en la teoría conjuntos y el álgebra relacional
- Visión de la organización de los datos muy sencilla
  - Filas independientes
  - Filas únicas
  - Columnas independientes
  - Valores de las columnas unitarios
- Los usuarios no necesitan tener un conocimiento de la posición física de los datos, la navegación por la base de datos se realiza de forma automática
- Lenguaje de acceso a los datos muy sencillo (SQL)
- Sencillez del lenguaje, similar al lenguaje natural

Soporte a otros modelos

- Jerárquico
  - La posibilidad de almacenar datos en formato XML le permite trabajar con datos semiestructurados siguiendo un modelo jerárquico.
  - El uso de esquema XSD impone unas restricciones tan fuertes como las del modelo relacionas.
  - Los datos se pueden consultar mediante XPath y Xquery.
- Objetos
  - La posibilidad de crear nuevos tipos de datos CLR permite el almacenamiento de objetos
- Documental
  - La búsqueda de texto completo permite el tratamiento de los campos de texto muy largo como si fueran documentos

Características habituales

- Almacenamiento, extracción y actualización de datos
- Catálogo accesible por el usuario
- Servicios para mejorar la independencia de datos (metadatos)
- Soporte para la tramitación de datos (comunicaciones)
- Soporte multiusuario y servicios de autorización
- Servicios de control de concurrencia
- Soporte de transacciones
- Servicios de integridad
- Servicios de recuperación (y restauración)
- Servicios de utilidad: Importación/Exportación, Monitorización y Análisis de rendimientos, Automatización

Características

- Fusiona el procesamiento transaccional en línea (OLTP) y el de procesamiento analítico en línea (OLAP)
- Facilidad de uso
  - SQL Server Management Studio
  - SQL Management Objects (SMO)
- Disponibilidad
  - Reflejo de base de datos (1 BD espejo)
  - Clústeres de conmutación por error
  - Instantáneas de base de datos
  - Recuperación rápida
  - Conexión del administrador no compartida
  - Operaciones en línea (Índices, Operaciones y Restauración)
  - Réplica (incluida entre iguales)

Características

- Escalabilidad
  - Partición de tablas e índices
  - Aislamiento de instantáneas
  - Supervisor de réplicas
  - Compatible con Itanium 2 de 64 bits y x64
- Seguridad
  - Autenticación
  - Autorización
  - Cifrado nativo y Cifrado de base de datos transparente
  - SQL Server y Trustworthy Computing (Seguro por diseño, inicio e implementación)

Características

- Rendimiento
  - Compresión de datos y de copia de seguridad
  - Partición de Tablas e Índices
  - Índices filtrados, Columnas dispersas
  - Regulador de recursos
  - Copias de seguridad reflejadas
- Administración
  - Gestión de la administración basada en directivas
  - Servidores de administración central
  - Recopilador de datos y Seguimiento de cambios
  - Auditoría

Características

- Extensibilidad
  - Business Intelligence Development Studio
  - Integración CLR/.NET Framework
  - Tipos y agregados definidos por el usuario
  - Servicios Web y XML
  - Service Broker

Inteligencia empresarial

- La plataforma de datos SQL Server incluye las siguientes herramientas:
  - Motor de base de datos
  - Analysis Services.
  - Reporting Services.
  - Integration Services
  - Servicios de réplica.
  - Master Data Services
  - Service Broker

Motor de base de datos

- El Motor de base de datos es el servicio principal para almacenar, procesar y proteger los datos.
- El Motor de base de datos proporciona acceso controlado y procesamiento rápido de transacciones para cumplir los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa.
- El Motor de base de datos también proporciona una completa compatibilidad para mantener una gran disponibilidad.

Analysis Services

- Datos multidimensionales:
  - Analysis Services admite OLAP al permitir al usuario diseñar, crear y administrar estructuras multidimensionales que contienen datos agregados de otros orígenes tales como bases de datos relacionales.
- Minería de datos:
  - Analysis Services permite al usuario diseñar, crear y visualizar modelos de minería de datos.
  - Estos modelos de minería de datos se pueden construir a partir de otros orígenes de datos empleando una amplia variedad de algoritmos de minería de datos estándar.

Integration Services

- Integration Services es una plataforma para generar soluciones de integración de datos de alto rendimiento, lo que incluye paquetes que proporcionan procesamiento de extracción, transformación y carga (ETL) para almacenamiento de datos.

Reporting Services

- Reporting Services 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.
- Una solución global para crear, administrar y proporcionar tanto informes tradicionales orientados al papel como informes interactivos basados en la Web.

Replicación

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

Master Data Services

- Master Data Services es el origen de datos maestros para la organización.
- Mediante la integración de diferentes sistemas de análisis y operaciones con Master Data Services, puede asegurarse de que todas las aplicaciones de la organización dependen de una fuente de información central y precisa.
- Con Master Data Services, se crea un origen único de datos maestros y se mantiene un registro de los datos, que cambian en el tiempo y que se puede controlar.

Service Broker

- Service Broker ayuda a los desarrolladores de software a crear aplicaciones de base de datos escalables y seguras.
- Esta nueva tecnología de Motor de base de datos proporciona una plataforma de comunicación basada en mensajes que permite a los componentes de aplicación independientes trabajar como un conjunto funcional.
- Service Broker incluye infraestructura para programación asincrónica que se puede utilizar para aplicaciones en una base de datos única o instancia única, y también para aplicaciones distribuidas.

Herramientas de administración

- SQL Server incluye herramientas integradas de administración para administración y optimización avanzadas de bases de datos, así como también integración directa con herramientas tales como Microsoft Operations Manager (MOM) y Microsoft Systems Management Server (SMS).
- Los protocolos de acceso de datos estándar reducen drásticamente el tiempo que demanda integrar los datos en SQL Server con los sistemas existentes.
- Asimismo, el soporte del servicio Web nativo está incorporado en SQL Server para garantizar la interoperabilidad con otras aplicaciones y plataformas.

Herramientas de desarrollo

- SQL Server ofrece herramientas integradas de desarrollo para el motor de base de datos, extracción, transformación y carga de datos, minería de datos, OLAP e informes que están directamente integrados con Microsoft Visual Studio para ofrecer capacidades de desarrollo de aplicación de extremo a extremo.
- Cada subsistema principal en SQL Server se entrega con su propio modelo de objeto y conjunto de interfaces del programa de aplicación (API) para ampliar el sistema de datos en cualquier dirección que sea específica de su negocio.

SQL Server Management Studio

- SQL Server Management Studio es un entorno integrado para obtener acceso a todos los componentes de SQL Server, configurarlos, administrarlos y desarrollarlos.
- SQL Server Management Studio combina un amplio grupo de herramientas gráficas con una serie de editores de script enriquecidos para ofrecer acceso a SQL Server a desarrolladores y administradores de todos los niveles de especialización.
- SQL Server Management Studio combina las características del Administrador corporativo, el Analizador de consultas y Analysis Manager, herramientas incluidas en versiones anteriores de SQL Server, en un único entorno.
- Además, SQL Server Management Studio funciona con todos los componentes de SQL Server, como Reporting Services, Integration Services y SQL Server Compact 3.5 SP2.
- Los administradores de bases de datos obtienen una única herramienta completa que combina herramientas gráficas fáciles de usar con funcionalidad de scripting enriquecida.

TRANSACT-SQL

Lenguaje SQL

- Definición
  - Lenguaje de comandos que se utiliza en los sistemas de gestión de bases de datos relacionales
- Características
  - Unificado para las bases de datos relacionales
  - Unificado para todo tipo de usuarios (dba, usuarios finales, programadores, etc.)
  - Parecido al idioma inglés
  - No procedimental
- Funcionalidad
  - CREAR tablas, vistas y sinónimos
  - ALMACENAR información en las tablas
  - MODIFICAR la información ya almacenada
  - CONSULTAR la base de datos
  - MANTENER la propia base de datos

Estructura del Lenguaje SQL

- El lenguaje de Consultas Estructurado SQL se puede dividir en:
  - Lenguaje DQL (Lenguaje de Consulta de Datos)
    - Sentencia SELECT
  - Lenguaje DML (Lenguaje de Manipulación de Datos)
    - Sentencias INSERT, DELETE, UPDATE y MERGE
  - Lenguaje DDL (Lenguaje de Definición de Datos)
    - Sentencias CREATE, DROP y ALTER
  - Lenguaje DCL (Lenguaje de Control de Datos)
    - Sentencias GRANT, REVOKE y DENY
  - Lenguaje DVL (Lenguaje de Validación de Datos)
    - Sentencias COMMIT, ROLLBACK y SET TRANSACTION

Transact-SQL

- Es el lenguaje que se utiliza para administrar instancias del SQL Server Database Engine, para crear y administrar objetos de base de datos, y para insertar, recuperar, modificar y eliminar datos.
- Transact-SQL es una extensión (supraconjunto) del lenguaje definido en los estándares de SQL publicados por International Standards Organization (ISO) y American National Standards Institute (ANSI).
- Cuenta con:
  - Instrucciones SQL estándar ampliadas
  - Instrucciones procedurales (declaración de variables, expresiones, control, excepciones, …)
  - Instrucciones para administración, monitorización y optimización
  - Tablas, vistas, procedimientos y constantes del sistema

Introducción a la sintaxis

- No es sensible a mayúsculas y minúsculas salvo las cadenas en cuyo caso dependen de la intercalación de la columna.
- Varias líneas por instrucción y una instrucción por línea o usar ; de finalización de instrucción.
- Constantes:
  - Valores numéricos: Conjuntos de dígitos con el punto como separador decimal
  - Valores binarios: Tienen el prefijo 0x y son cadenas de números hexadecimales
  - Valores no numéricos y cadenas: '…', cadenas Unicode: N'…' [COLLATE intercalación]
    - Comillas dentro de una cadena: '…''…', el . como separador de decimales
  - Valores uniqueidentifier: 32 dígitos hexadecimales en formato cadena (8-4-4-4-12) o en formato hexadecimal:
    - '6F9619FF-8B86-D011-B42D-00C04FC964FF' = 0x6F9619FF8B86D011B42D00C04FC964FF
- Comentarios:

  --

  /* */
- Bloques:

  BEGIN

  …

  END

NULL y UNKNOWN

- NULL indica que el valor es desconocido.
- Un valor NULL no es lo mismo que un valor cero o vacío.
- No hay dos valores NULL que sean iguales.
- La comparación entre dos valores NULL, o entre un valor NULL y cualquier otro valor, tiene un resultado desconocido porque el valor de cada NULL es desconocido.
- Normalmente, los valores NULL indican que los datos son desconocidos, no aplicables o que se van a agregar posteriormente.
- Hay que tener en cuenta lo siguiente sobre los valores NULL:
  - Para comprobar si hay valores NULL en una consulta hay que usar IS NULL o IS NOT NULL en la cláusula WHERE.
  - Los valores NULL se pueden insertar en una columna si se indica explícitamente NULL en una instrucción INSERT o UPDATE, o si se deja fuera una columna de una instrucción INSERT.
  - Los valores NULL no se pueden usar como la información necesaria para distinguir una fila de una tabla de otra fila, como, por ejemplo, las claves principales, o bien para la información que se usa para la distribución de filas, como las claves de distribución.
- Cuando hay valores NULL en los datos, los operadores lógicos y de comparación pueden devolver un tercer resultado UNKNOWN (desconocido) en lugar de simplemente TRUE (verdadero) o FALSE (falso).

Identificadores

- Identificadores estándar
  - El primer carácter debe ser un carácter alfabético
  - De uno a 128 caracteres.
  - Otros caracteres pueden incluir letras, números o símbolos: @, $, # o _.
  - Los identificadores que comienzan con un símbolo tienen usos especiales: @, # o ##.
- Identificadores delimitados
  - Se utilizan cuando los nombres contienen espacios incrustados
  - Se utilizan cuando partes de los nombres incluyen palabras reservadas
  - Deben encerrarse entre corchetes ([ ])
  - o dobles comillas (" ") si SET QUOTED_IDENTIFIER ON
- Directrices de denominación para los identificadores
  - Poner nombres cortos
  - Utilizar nombres significativos cuando sea posible
  - Utilizar una convención de denominación clara y sencilla
  - Utilizar un identificador que distinga el tipo de objeto (Vistas o Procedimientos almacenados)
  - Hacer que los nombres de los objetos y de los usuarios sean únicos
- Identificador completo
  - [Servidor.][Base de datos.][Esquema.]Objeto[.Subelemento]

Bases de Datos

- Bases de Datos de sistema
  - Almacenan información necesaria para el funcionamiento del servidor SQL Server
  - SQL Server utiliza estas BBDD para manejar y gestionar el sistema
- Bases de datos de usuarios
  - Bases de Datos donde se implementan los modelos de datos de los usuarios (OLTP)
- Almacén de datos
  - Organizar grandes cantidades de datos estables para facilitar el análisis y la recuperación con OLAP

Objetos de la BBDD

- Diagramas de Bases de Datos
  - Herramienta visual que permite diseñar y ver una base de datos completa o parcialmente
- Tablas
  - Colección de datos organizados en filas y columnas
- Índices
  - Estructura de datos que proporciona una manera rápida de acceder a la información
- Restricciones
  - Sirven para definir reglas relativas a los valores permitidos en las columnas
  - Es el mecanismo utilizado para reforzar la integridad referencial en las Bases de Datos que SQL Server exige automáticamente
  - Las restricciones establecen los valores aceptados por las columnas: nulos, rangos de valores, duplicados, claves principales o ajenas.

Objetos de la BBDD

- Estadísticas
  - Índices de distribución de valores que utiliza el optimizador de consultas para determinar el plan de consultas óptimo.
- Vistas
  - Proporciona una forma de ver los datos provenientes de una o más tablas
  - Sirven para facilitar el acceso a los datos, y garantizar la seguridad de los mismos
- Sinónimos
  - Es un nombre alternativo para un objeto de ámbito de esquema
- Service Broker
  - Tipos y contratos, colas, servicios, rutas, enlaces.
- Almacenamiento
  - Catálogos e índices de texto
    - Permiten las búsquedas de texto complejas en datos de cadenas de caracteres.
  - Esquemas de partición
  - Funciones de partición

Objetos para programación

- Procedimientos Almacenados
  - Conjunto de sentencias de Transact-SQL precompiladas con nombre que se ejecutan como una unidad
  - Mejoran el rendimiento del sistema
  - Se pueden implementar en .NET CLR
- Funciones (SQL definidas por el usuario)
  - Funciones definidas por el usuario
  - Conjunto de sentencias de Transact-SQL o CLR que suelen utilizarse para encapsular una transformación lógica
  - Pueden devolver un valor escalar o una tabla
- Desencadenadores
  - También llamados triggers o disparadores
  - Procedimiento almacenado de ejecución automática ante una operación DML o DDL
  - Se crean sobre una tabla
  - Ayudan a mantener la integridad

Objetos para programación

- Ensamblados
  - Son archivos DLL que se utilizan en una instancia de SQL Server para implementar funciones, procedimientos almacenados, desencadenadores, agregados definidos por el usuario y tipos definidos por el usuario que están escritos en uno de los lenguajes de código administrado que se alojan en Microsoft .NET Framework.
- Tipos de Datos
  - Definen los valores permitidos para una columna o variable
  - SQL Server suministra un conjunto de datos definidos por el sistema, aunque el usuario puede crearse tipos de datos propios
- Reglas
  - Contiene información que define los valores permitidos que serán almacenados en una columna o para un tipo de dato determinado
- Valores predeterminados
  - Valor por defecto para cuando no se suministre valor en una columna determinada

Objetos para la seguridad

- Inicios de sesión
  - Cuentas que controlan el acceso al sistema SQL Server.
- Usuarios
  - Identificador que asocia un inicio de sesión a un usuario dentro de una base de datos para otorgar los permisos.
- Esquemas
  - Es una colección de entidades de base de datos que forma un solo espacio de nombres
- Funciones (Roles)
  - Agrupan los permisos necesarios para los diferentes tipos de usuarios.
- Claves simétricas y asimétricas
  - Encriptación y desencriptación, autenticaciones, ..
- Certificados
  - Autenticación ante terceros (comunicaciones seguras) y cifrado
- Credenciales
  - Una credencial es un registro que contiene la información de autenticación necesaria para conectarse a un recurso fuera de SQL Server. La mayoría de las credenciales constan de un inicio de sesión de Windows y una contraseña.

Otros objetos

- Propiedades extendidas
  - Propiedades definidas por los usuarios para almacenar información adicional o específica sobre los objetos de una base de datos.
- Intercalaciones
  - Controlan el juego de caracteres utilizado en el almacenamiento físico de las cadenas, especificado las combinaciones de bits que representa cada carácter y las reglas por las que los caracteres se ordenan y comparan.

Tipos de datos

- Numéricos exactos
- Numéricos aproximados
- Fecha y hora
- Cadenas de caracteres
- Cadenas de caracteres Unicode
- Cadenas binarias
- Otros tipos de datos

Tipos de datos

Tipo	Bytes	Descripción
bigint	8	Datos enteros (números enteros) comprendidos entre -2^63 (-9223372036854775808) y 2^63 -1 (9223372036854775807).
int	4	Datos enteros (números enteros) comprendidos entre -2^31 (-2.147.483.648) y 2^31 - 1 (2.147.483.647).
smallint	2	Datos enteros comprendidos entre 215 (-32.768) y 215 - 1 (32.767).
tinyint	1	Datos enteros comprendidos 0 y 255.
bit	1*	Datos enteros con valor 1 ó 0.
decimal	5, 9, 13,17	Datos de precisión y escala numérica fijas con 38 dígitos significativos (escala + precisión). [vardecimal con SP2]
numeric		Funcionalmente equivalente a decimal.

Tipos de datos

Tipo	Bytes	Descripción
money	8	Valores de moneda comprendidos entre -922.337.203.685.477,5808 y +922.337.203.685.477,5807, con una precisión de una diezmilésima de la unidad monetaria. (escala 4)
smallmoney	4	Valores de moneda comprendidos entre -214.748,3648 y +214.748,3647, con una precisión de una diezmilésima de la unidad monetaria.
Numéricos aproximados		
float	4, 8	Números con precisión de coma flotante comprendidos entre -1,79E + 308 y 1,79E + 308.
real	4	Números con precisión de coma flotante comprendidos entre -3,40E + 38 y 3,40E + 38. float(24)

Tipos de datos

Tipo	Bytes	Descripción
datetime	8	Datos de fecha y hora comprendidos entre el 1 de enero de 1753 y el 31 de diciembre de 9999, con una precisión de 3,33 milisegundos.
smalldatetime	4	Datos de fecha y hora comprendidos entre el 1 de enero de 1900 y el 6 de junio de 2079, con una precisión de un minuto.
datetime2	6,7,8	Datos de fecha y hora comprendidos entre el Del 1 de enero del año 1 después de Cristo al 31 de diciembre de 9999. De 00:00:00 a 23:59:59.9999999
date	3	De 0001-01-01 a 9999-12-31
time	3 a 5	De 00:00:00.0000000 a 23:59:59.9999999
datetimeoffset	8..10	De 0001-01-01 00:00:00.0000000 a 9999-12-31 23:59:59.9999999 (en UTC)

Tipos de datos

Tipo	Bytes	Descripción
char	1	Datos de caracteres ANSI de longitud fija con una longitud máxima de 8.000 caracteres.
varchar	1…	Datos ANSI de longitud variable con un máximo de 8.000 caracteres.
~~text~~ varchar(max)	8K…	Datos ANSI de longitud variable con una longitud máxima de 231 - 1 (2.147.483.647) caracteres.
nchar	2	Datos Unicode de longitud variable con una longitud máxima de 4.000 caracteres.
nvarchar	2…	Datos Unicode de longitud variable con una longitud máxima de 4.000 caracteres. sysname es el tipo de datos suministrado por el sistema y definido por el usuario que es funcionalmente equivalente a nvarchar(128) y que se utiliza para hacer referencia a nombres de objetos de bases de datos.
~~ntext~~ nvarcha(max)	8K…	Datos Unicode de longitud variable con una longitud máxima de 230 - 1 (1.073.741.823) caracteres.

Tipos de datos

Tipo	Bytes	Descripción
binary	1…	Datos binarios de longitud fija con una longitud máxima de 8.000 bytes.
varbinary	1…	Datos binarios de longitud variable con una longitud máxima de 8.000 bytes.
~~image~~ varbinary(max)	8K…	Datos binarios de longitud variable con una longitud máxima de 231 - 1 (2.147.483.647) bytes.
xml	8K…	Datos XML de longitud variable con una longitud máxima de 2 Gb (2.147.483.647) caracteres.
sql_variant	1…	Un tipo de datos que almacena valores de varios tipos de datos aceptados en SQL Server, excepto text, ntext, timestamp y sql_variant.
~~timestamp~~ rowversion	8	Un número único para toda la base de datos que se actualiza cada vez que se actualiza una fila.
uniqueidentifier	16	Un identificador exclusivo global (GUID).

Tipos de datos

Tipo	Bytes	Descripción
hierarchyid	1..5	Dato utilizado para representar la posición en una jerarquía
geography	2…	Datos elípticos (globo), como las coordenadas de latitud y longitud del sistema GPS. Pueden contener uno o mas puntos.
geometry	2…	Datos en un sistema de coordenadas euclídeo (plano). Pueden contener uno o mas puntos.
Tipos programáticos		
cursor		Una referencia a un cursor.
table		Un tipo de datos especial que se utiliza para almacenar un conjunto de resultados para un proceso posterior.

Tipos de Datos definidos por el usuario

- Los tipos de datos definidos por el usuario están basados en un tipo integrado y permiten establecer su longitud, si acepta nulos, qué regla restringe sus valores y cuál es su valor por defecto.
- Para crear un nuevo tipo es necesario asignar:
  - Nombre del nuevo tipo
  - Tipo de datos del sistema en el que se basa el nuevo tipo de datos
  - Longitud (si el tipo de datos lo permite)
  - Aceptación de valores NULL
- Opcionalmente se le pueden asignar:
  - Regla que definen los valores permitidos
  - Valor predeterminado
- Una vez creados solo se pude modificar la regla y el valor predeterminado.

Directivas

- Indica a las utilidades de SQL Server el final de un lote de instrucciones Transact-SQL.
  - GO
- Cambia el contexto de la base de datos al de la base de datos especificada o a la instantánea en SQL Server.
  - USE { database }
- Muestra un mensaje en el área de mensajes
  - PRINT msg_str | @local_variable | string_expr
- Ejecuta una cadena de comandos o una cadena de caracteres dentro de un lote de Transact-SQL
  - [[EXEC[UTE]] { [@estadoDevuelto =] { nombreProcedimiento [;número] | @ nombreProcedimientoVar } [[@parámetro =] { valor | @variable [OUTPUT] | [DEFAULT]] [,...n] [WITH RECOMPILE]
- Fijar opciones que cambian el tratamiento de información específica por parte de la sesión actual.
  - SET opción valor

Opciones mas comunes

- SET DATEFIRST 1
  - Establece el primer día de la semana en un número del 1 (Lunes) al 7 (Domingo)
- SET DATEFORMAT dmy
  - Determina el orden de los componentes de la fecha (mes/día/año) para escribir datos de tipo datetime o smalldatetime.
- SET LANGUAGE Español;
  - Especifica el entorno de idioma de la sesión. El idioma de la sesión determina los formatos de datetime y los mensajes del sistema
- SET IDENTITY_INSERT { ON | OFF }
  - Permite insertar valores explícitos en la columna identidad de una tabla
- SET NOCOUNT { ON | OFF }
  - Hace que deje de devolverse como parte de los resultados el mensaje que muestra el número de filas afectado por una instrucción Transact-SQL
- SET NOEXEC { ON | OFF }
  - Compila cada consulta, pero no la ejecuta
- SET ROWCOUNT { ON | OFF }
  - Hace que SQL Server detenga el procesamiento de la consulta una vez que se han devuelto las filas especificadas.
- SET IMPLICIT_TRANSACTIONS { ON | OFF }
  - Establece el modo de transacción implícita para la conexión
- SET TRANSACTION ISOLATION LEVEL
  - Controla el comportamiento del bloqueo y de las versiones de fila de las instrucciones Transact-SQL emitidas por una conexión a SQL Server
- SET XACT_ABORT { ON | OFF }
  - Especifica si SQL Server revierte automáticamente la transacción actual cuando una instrucción Transact-SQL genera un error en tiempo de ejecución

Lenguaje de Consulta de Datos

DQL

Lenguaje de Consulta de Datos (DQL)

- Se pueden consultar tablas y vistas
- La sentencia utilizada para la elaboración de consultas es la SELECT
- La sentencia SELECT permite realizar las siguientes operaciones:
  - Recuperar toda la información asociada a una tabla
  - Recuperar todas las columnas de una tabla
  - Recuperar sólo las columnas que se especifiquen
  - Recuperar una selección de filas
  - Controlar el orden de salida de las filas en las consultas
  - Obtener en el momento de la consulta columnas productos de operaciones con otras columnas, constantes, etc.
- Las consultas se realizan por asociación de valores, no por la localización de los mismos

Sintaxis general de la sentencia SELECT

SELECT columna, ...	el qué
FROM tabla, ...	de dónde
WHERE condiciones	bajo qué condiciones
GROUP BY valores	condición de agrupamiento
HAVING condiciones	filtra los grupos
ORDER BY columna	ordena la salida

- Solo son obligatorias las cláusulas SELECT y FROM

Sintaxis general de la sentencia SELECT

- SELECT
  - Se indican las columnas a consultar. Si se quiere consultar todas las columnas de una tabla se pone un '*'
- FROM
  - Se indica la tabla o tablas de donde se va a realizar la consulta
- WHERE
  - Se indican las condiciones para la realización de la consulta
- GROUP BY
  - Agrupaciones por determinadas columnas
- HAVING
  - Se indican las condiciones para la realización de las agrupaciones.
- ORDER BY
  - Se indican las columnas por las cuales se va a realizar la ordenación de la consulta

Orden de interpretación

1. FROM
2. WHERE
3. GROUP BY
4. WITH CUBE o WITH ROLLUP
5. HAVING
6. SELECT
7. DISTINCT
8. ORDER BY
9. TOP

Cláusula SELECT

- SELECT sirve para seleccionar ciertas columnas, o valores derivados de ellas, y puede incluir:
  - Constantes numéricas, alfanuméricas y variables
  - Valores de columnas con el formato [nombre de tabla.] nombre de columna
  - Expresiones con nombres de columnas y constantes
  - Todo tipo de funciones aplicadas a columnas de una tabla y/o variables
  - Permite la especificación del titulo de la columna que aparecerá en la cabecera
- FROM sirve para definir las tablas donde se va a ir a buscar las columnas especificadas en la sentencia SELECT. Se puede asociar un nuevo nombre a las tablas para facilitar la consulta (sinónimo temporal)

Columnas

- *
  - Todas las columnas
- { table_name | view_name | table_alias }.*
  - Todas las columnas de una tabla o vista
- [{ table_name | view_name | table_alias }.]column_name
  - Columna de una tabla o vista
- expresión
  - Resultado de evaluar una expresión o constante.
- $IDENTITY
  - Devuelve la columna de identidad.
- $ROWGUID
  - Devuelve la columna GUID de fila.
- [AS] column_alias
  - Asigna un nombre alternativo a la columna de resultado

DISTINCT / ALL / TOP

- La función que tiene DISTINCT es la de eliminar las filas repetidas que se produzcan en una sentencia SELECT, por defecto es ALL: mostrar también las filas duplicadas.

```
SELECT DISTINCT COD_CAT
FROM EMP;
```

- TOP (expression) [PERCENT] [WITH TIES]
  - Indica que el conjunto de resultados de la consulta solamente devolverá un primer conjunto o porcentaje de filas especificado. expression puede ser un número o un porcentaje de las filas.

```
SELECT TOP(10) COD_CAT
FROM EMP;
```


Cláusula WHERE

- Si la cláusula WHERE está presente en una consulta, sirve para especificar qué filas de las tablas se desean obtener como resultado de la consulta.

- Para determinar las condiciones de selección de filas intervienen tres elementos:
  - Nombre de la columna
  - Operador de comparación
  - Nombre de columna, constante, lista de valores

- SINTAXIS
  - SELECT columnas
  - FROM tabla,…
  - WHERE condición (o condiciones) de selección de filas

Expresiones

- Se trata de una combinación de símbolos y operadores que Motor de base de datos de SQL Server evalúa para obtener un único valor de datos.
- Las expresiones simples pueden ser una sola constante, variable, columna o función escalar.
- Los operadores se pueden usar para combinar dos o más expresiones simples y formar una expresión compleja.
- Dos expresiones se pueden combinar mediante un operador si ambas tienen tipos de datos admitidos por el operador y se cumple al menos una de estas condiciones:
  - Las expresiones tienen el mismo tipo de datos.
  - El tipo de datos de menor prioridad se puede convertir implícitamente al tipo de datos de mayor prioridad.
- Si las expresiones no cumplen estas condiciones, se pueden usar las funciones CAST o CONVERT para convertir explícitamente el tipo de datos de menor prioridad al tipo de datos de mayor prioridad o a un tipo de datos intermedio que se puede convertir implícitamente al tipo de datos de mayor prioridad.

Operadores Expresión

- ARITMÉTICOS
  - Utilizables para formar expresiones con constantes, valores de columnas y funciones de valores de columnas

+	Suma	-	Resta
*	Multiplica	/	Divide
%	Módulo (Resto de la división entera)		

- BINARIOS
  - Aplicables a int, smallint, tinyint, binary o varbinary

&	AND bit a bit	|	OR bit a bit
^	OR exclusivo bit a bit	~	NOT bit a bit

- PARA CADENAS DE CARACTERES
  - Existe el operador de concatenación
    - cadena_1 + cadena_2

Operadores

- GENERALES DE COMPARACIÓN
  - = IGUAL >= !< MAYOR ó IGUAL
  - != <> NO IGUAL < MENOR QUE
  - > MAYOR QUE <= !> MENOR ó IGUAL
- DE CADENAS DE CARACTERES
  - [NOT] LIKE permite los siguientes caracteres especiales en las cadenas de comparación:
    - '%' Cualquier cadena de cero o más caracteres
    - '_' Cualquier carácter
    - [a-z] Coincidencia con alguno de los caracteres del intervalo
    - [^abc] No coincidencia con alguno de los caracteres del intervalo
  - En función a la intercalación, las mayúsculas y minúsculas pueden ser significativas y las tildes se pueden ignorar.
  - Usar ESCAPE para buscar un carácter especial
    - … Where nombre LIKE '%AF$_E%' ESCAPE '$'

Operadores Lógicos

- [NOT] BETWEEN valor_1 AND valor_2
  - Si tiene un valor comprendido entre los valores la fila es seleccionada.
- [NOT] IN (valor_1,valor_2,....,valor_n)
  - Si la fila tiene un valor igual alguno del conjunto de valores la fila es seleccionada
- IS [NOT] NULL
  - Si la fila tiene valor null la fila es seleccionada
- comparación ANY, comparación SOME
  - Sí se cumple la comparación con alguno de los valores del conjunto
- comparación ALL
  - Sí se cumple la comparación con todos los valores del conjunto

Operadores Lógicos

- [NOT] EXISTS
  - si una subconsulta contiene cualquiera de las filas.
- AND
  - si ambas expresiones booleanas se cumplen.
- OR
  - si cualquiera de las dos expresiones booleanas se cumple.
- NOT
  - Invierte el valor de cualquier otro operador lógico.

Operador Condicional

- Evalúa una lista de condiciones y devuelve una de las expresiones de resultado posibles.
- La expresión CASE tiene dos formatos:
  - La expresión CASE genera un valor que se compara con los posibles resultados.

```
CASE input_expression
   WHEN value1 THEN result_expression
   WHEN value2 THEN result_expression [ ...n ]
   [ ELSE else_result_expression ]
END
```

  - Múltiples condiciones que obtiene el resultado de la primera que se cumpla.

```
CASE
   WHEN Boolean_expression1 THEN result_expression
   WHEN Boolean_expression2 THEN result_expression [ ...n ]
   [ ELSE else_result_expression ]
END
```

- Ambos formatos admiten un argumento ELSE opcional que suministra el resultado si no se cumple ninguna de las anteriores.

Cláusula ORDER BY

- Sirve para fijar el orden de salida de las filas seleccionadas en una sentencia SELECT
- Permite ordenar las filas utilizando los criterios siguientes:
  - Orden ascendente (ASC valor por defecto)
  - Orden descendente (DESC)
  - Por múltiples columnas (la columna más a la izquierda es por la que primero se clasifica)
  - Con valores nulos
- Debe ser la última cláusula de la sentencia SELECT
- Cuando no se usa, el orden de salida de las filas no está definido

- NOTA: Cuando se quiere ordenar por columnas calculadas se indica en la orden ORDER BY el alias de las columna en la selección.

```
Select nombre, Salario / 12 SalarioAnual
....
Order by SalarioAnual
```


Ordenación Condicional

- La clausula ORDER BY puede contener un expresión según la que se ordenará el conjunto de resultados de la consulta.
- Usando el operador CASE se puede establecer una ordenación condicional que dependa de otras columnas, variables o parámetros.

```
ORDER BY CASE CountryRegionName
   WHEN 'United States' THEN TerritoryName
   ELSE CountryRegionName
END
```

Ordenación con paginación

- OFFSET-FETCH es una extensión de la cláusula ORDER BY que permite filtrar un rango de filas seleccionado de forma similar al TOP
- Proporciona un mecanismo para paginar a través de los resultados, especificando el número de filas a omitir (saltar) y el número de filas a recuperar:

```
OFFSET { offset_rows } { ROW | ROWS }
  [ FETCH { FIRST | NEXT } {fetch_row_count } { ROW | ROWS } ONLY ]
```

- Es obligatorio indicar el valor de offset_rows, pero puede ser cero si no se requiere saltar (primera página)
- La cláusula FETCH es opcional, si se omite se devuelven todas las filas que siguen al valor OFFSET
- Tanto OFFSET como FETCH pueden ser constantes, expresiones, variables y parámetros
- Con el enfoque del lenguaje natural al código, ROW | ROW, por un lado, y FIRST | NEXT, por otro, son intercambiables para obtener expresiones mas legibles.

```
OFFSET 0 ROWS FETCH FIRST 20 ROWS ONLY
OFFSET 20 ROWS FETCH NEXT 20 ROWS ONLY
```


Ejercicios

- Ejercicios propuestos:
  - Seleccionar todos los productos cuyo peso esté comprendido entre 10 y 100.
  - Obtener un listado de todos los colores diferentes que hay en los productos de estilo femenino.
  - Obtener una lista con los productos ordenada por peso y dentro de él por colores.
  - Obtener los productos rojos, azules y negros
  - Se pide una lista que recupere el peso de todos los productos cuyo número de producto empiece por 'BK'.
  - Obtener el Número de producto, Nombre y Precio de venta de los productos de Carretera y Montaña
  - Obtener los 5 productos mas caros de la línea estándar y estilo universal.
  - Obtener los productos de gama alta plateados y los de gama baja negros.

Funciones

- Escalares
  - Opera con elementos de una sola fila como entradas y devuelve un valor único (escalar) como salida
  - Se puede utilizar como una expresión en consultas.
  - Puede ser determinista o no determinista.
  - La intercalación depende del valor de entrada o la intercalación predeterminada de la base de datos
- Agregado Agrupado
  - Toma uno o más valores de una o mas filas pero devuelve un solo valor resumen
  - Requieren grupos (GROUP BY)
- Ventana
  - Opera en una ventana (conjunto) de filas pero devuelve un solo valor resumen
  - Incluye funciones de clasificación, desplazamiento, agregado y distribución.
- Conjunto de filas
  - Devuelve una tabla virtual que se pueden usar posteriormente en sustitución de tablas.

Funciones Aritméticas

- ABS(n)
  - Valor absoluto de n
- CEILING (n)
  - Entero inmediatamente superior a n (n no entero)
- FLOOR (n)
  - Entero inmediatamente inferior a n (n no entero)
- ROUND(valor[,precisión])
  - Redondea valor con la precisión indicada
- POWER(valor,exponente)
  - Eleva valor al exponente indicado
- SQRT (n)
  - Raíz cuadrada de n, si n < 0 NULL
- SIGN (n)
  - Si n < 0, -1; Si n = 0, 0; Si n > 0, 1

Funciones de Cadena de Caracteres

- ASCII(s)
  - Valor ASCII del primer carácter de la cadena "n”
- CHR(n)
  - Devuelve el carácter cuyo valor ASCII es el numero "n”
- LOWER(s)
  - Cadena "s" con todas sus letras convertidas en minúsculas
- UPPER (s)
  - Cadena "s" con todas sus letras en mayúsculas
- LEN(s)
  - El numero de caracteres de "s”
- SPACE (n)
  - Devuelve una cadena de n espacios repetidos.

Funciones de Cadena de Caracteres

- REPLICATE (s, n)
  - Devuelve s repetida n veces.
- RTRIM (s)
  - Suprime los blancos que contenga la cadena "s" a la derecha
- LTRIM (s)
  - Suprime los blancos que contenga la cadena "s" a la izquierda
- TRIM (s)
  - Suprime los blancos de la cadena "s" al principio y al final
- LEFT (s , n)
  - Devuelve los n primeros caracteres de la cadena
- RIGHT (s , n)
  - Devuelve los n últimos caracteres de la cadena

Funciones de Cadena de Caracteres

- SUBSTRING (s,m[,n])
  - Devuelve la subcadena de "s" que empieza con el carácter número "m" y tiene "n" caracteres de longitud.
- CHARINDEX (s , buscar [, start_location])
  - Busca una cadena dentro de otra cadena y devuelve la posición inicial de la primera coincidencia.
- PATINDEX ('%pattern%' , expression)
  - Devuelve la posición inicial de la primera repetición de un patrón en la expresión especificada o cero si el patrón no se encuentra
- TRANSLATE (cad1, cad2, cad3)
  - Cambia los caracteres de la cadena cad1 por los de la cadena cad3 en función de los caracteres de la cadena cad2
- REPLACE (s , buscada, sustituto)
  - Reemplaza todas las instancias de subcadena por otra subcadena.

Funciones de Conversión

- STR (float_expression [, length [, decimal]])
  - Convierte una valor numérico en cadena, se puede indicar número total de digito y cuantos son decimales
- CAST (expression AS data_type [(length)])
  - Convierte una expresión de un tipo de datos a otro
- CONVERT (data_type [(length)] , expression [, style])
  - Convierte una expresión de un tipo de datos a otro aplicándole formato
- PARSE (string_value AS data_type [USING culture])
  - Convierte una cadena a tipos de fecha y hora y de número siguiendo la referencia cultural
- FORMAT (value, format [, culture])
  - Devuelve un valor con el formato y la cultura indicada

Funciones de Fechas

- Funciones de fecha y hora del sistema
  GETDATE()
- Funciones que devuelven partes de fecha y hora
  DATENAME(datepart, date), DATEPART(datepart, date), DAY(date), MONTH(date), YEAR(date)
- Funciones que devuelven fecha y hora a partir de sus partes
  DATETIMEFROMPARTS(year, month, day, hour, minute, seconds, milliseconds), DATEFROMPARTS(year, month, day), TIMEFROMPARTS(hour, minute, seconds, fractions, precision)
- Funciones de calculo con fecha y hora
  DATEADD(datepart, number, date), DATEDIFF(datepart, startdate, enddate), EOMONTH (start_date [, month_to_add])

Funciones Lógicas

- COALESCE(exp1, exp2, exp3,....)
  - Devuelve el valor de la primera expresión que sea distinta de NULL
- CHOOSE (index, val_1, val_2 [, val_n])
  - Devuelve el elemento en el índice especificado
- IIF (boolean_expression, true_value, false_value)
  - Devuelve uno de dos valores, dependiendo de la condición
- NULLIF(exp1, exp2)
  - Devuelve NULL si exp1=exp2 sino devuelve exp1
- ISNULL (check_expression , replacement_value)
  - Sustituye el valor NULL por el valor especificado.
- ISNUMERIC (expression)
  - Determina si una expresión es un tipo numérico válido.

Ejercicios

- Ejercicios propuestos:
  - Calcular el porcentaje de beneficio [(P. Venta- P. Coste)/P. Venta] de los productos.
  - Calcular la edad de los empleados
  - Calcular la nueva lista de precios donde los artículos de estilo masculino suban un 20% y los femeninos un 10%, el nombre del producto se debe mostrar en mayúsculas.
  - Calcular un nuevo código de producto con el formato (utilizar la X como carácter sustitutivo de los valores desconocidos):
    - X: Línea de producto
    - X: Gama
    - X: Estilo
    - XXX: 3 primeras letras del nombre en mayúsculas
    - X: Inicial del color en mayúsculas
    - 999999: 6 dígitos con el identificador de producto

DQL: Lenguaje de Consulta de Datos

AGRUPAMIENTO

Cláusula GROUP BY

- Sirve para crear grupos de filas (resúmenes) y muestra una sólo fila por cada grupo de la SELECT
- Debería aparecer el mismo número de columnas en la clausula GROUP BY que en la sentencia SELECT sin tener en cuenta las columnas de funciones agragadas.
  - SELECT A, B, AVG(SAL)
  - ……..
  - GROUP BY A,B
- RESTRICCIONES
  - No se puede agrupar por columnas calculadas.
  - El NULL lo toma como si fuese un valor mas.
  - No se puede introducir dentro de una SUBSELECT.
  - Las columnas del ORDER BY debería coincidir con las del GROUP BY.

CLÁUSULA HAVING

- La cláusula HAVING, similar al WHERE, se emplea para FILTRAR los conjuntos agrupados por la cláusula GROUP BY que se visualizan
- La evaluación de las cláusulas de la sentencia SELECT en tiempo de ejecución se efectúa en el siguiente orden:
  1. FROM
  2. WHERE
  3. GROUP BY
  4. HAVING
  5. SELECT
  6. DISTINCT
  7. ORDER BY
  8. TOP

Funciones de agrupamiento

- AVG ([ALL | DISTINCT] expression)
  - Valor medio de expression (ignorando los valores nulos)
  - Sólo columnas numéricas.
- SUM ([ALL | DISTINCT] expression)
  - Suma todos los valores o solo de los valores DISTINCT de la expresión (ignorando los valores nulos)
  - Sólo columnas numéricas.
- MAX (expression)
  - Máximo valor de expression
- MIN (expression)
  - Mínimo valor de expression
- COUNT ({ [[ALL | DISTINCT] expression] | * }), COUNT_BIG(expr)
  - Número de veces que "expression" evalúa dato con valor no nulo.
- APPROX_COUNT_DISTINCT(expression)
  - Número aproximado de valores no nulos únicos de un grupo.

Funciones de agrupamiento

- STDEV ([ALL | DISTINCT] expression)
  - Desviación típica estadística de todos los valores de la expresión especificada.
- STDEVP ([ALL | DISTINCT] expression)
  - Desviación estadística estándar para la población de todos los valores de la expresión especificada.
- VAR ([ALL | DISTINCT] expression)
  - Varianza estadística de todos los valores de la expresión especificada.
- VARP ([ALL | DISTINCT] expression)
  - Varianza estadística de la población para todos los valores de la expresión especificada.

Particularidades de las funciones de agrupamiento

- Los valores nulos no participan en el cálculo de las funciones de conjuntos
  - SELECT AVG(SALARIO) y SELECT SUM(SALARIO) / COUNT(*) no siempre tendrán el mismo resultado
- No se pueden combinar resultados agrupados con resultados no agrupados salvo con la clausula OVER.
  - <AGREGADO> OVER ([PARTITION BY value, ... [n]])
- En la cláusula SELECT sólo pueden aparecer funciones de conjuntos que afecten a todas las filas si la consulta no esta agrupada, una única fila para la consulta.
- En la cláusula SELECT sólo pueden aparecer funciones de conjuntos y las columnas incluidas en la cláusula GROUP BY si la consulta esta agrupada, una fila por grupo.

Ventanas

- La cláusula OVER define una ventana o un conjunto de filas definido por el usuario en un conjunto de resultados de la consulta.
- Determina las particiones (agrupamiento) y el orden de un conjunto de filas antes de que se aplique la función de ventana asociada.
- Una función de ventana calcula por fila de resultados un valor único para cada ventana.

```
<FUNC> () OVER (
    [ <PARTITION BY clause> ]
    [ <ORDER BY clause> ]
)
```

- Puede utilizar la cláusula OVER con funciones de agregado, estadísticas, categoría y analíticas.
- El uso de ventanas evita la necesidad de combinaciones con consultas de resumen.

Funciones de categoría

- RANK()
  - Devuelve la clasificación de cada fila en la partición de un conjunto de resultados
- DENSE_RANK ()
  - Devuelve la clasificación de filas dentro de la partición de un conjunto de resultados, sin ningún espacio en los valores repetidos.
- NTILE ()
  - Distribuye las filas de una partición ordenada en un número especificado de grupos
- ROW_NUMBER ()
  - Devuelve el número secuencial de una fila de una partición de un conjunto de resultados, comenzando con 1 para la primera fila de cada partición

Ejercicios

- Ejercicios propuestos:
  - Calcular la diferencia del precio de venta del producto respecto a la media de su subcategoria, el mas caro y el mas barato.
  - Crear un ranking de los productos por su precio de coste.
  - Clasificar los productos por su precio de venta: Caros, Normales, Baratos.

GROUP BY ROLLUP

- Crea un grupo para cada combinación de expresiones de columna. Además, "acumula" los resultados en subtotales y totales generales.
- Para ello, se mueve de derecha a izquierda reduciendo el número de expresiones de columna para las que crea grupos y agregaciones.
- El orden de las columnas influye en la salida de ROLLUP y también puede afectar al número de filas del conjunto de resultados.

```
SELECT Country, Region, Product, SUM(Sales) AS TotalSales
FROM Sales
GROUP BY ROLLUP (Country, Region, Product);
```

- En el ejemplo anterior crea filas adicionales grupos para cada combinación de expresiones de columna en las listas siguientes.
  - col1, col2, col3, rslt (filas normales)
  - col1, col2, NULL, rslt (subtotal por región del país, una fila por país+región)
  - col1, NULL, NULL, rslt (este es el subtotal por país, una fila por país)
  - NULL, NULL, NULL, rslt (este es el total general, una sola fila)

GROUP BY CUBE

- GROUP BY CUBE es similar a GROUP BY ROLLUP pero crea subtotales para todas las combinaciones posibles de columnas.
- El orden de las columnas influye en la salida de ROLLUP y también puede afectar al número de filas del conjunto de resultados.

```
SELECT Country, Region, Product, SUM(Sales) AS TotalSales
FROM Sales
GROUP BY CUBE (Country, Region, Product)
```

- En el ejemplo anterior crea las siguientes filas adicionales:
  - col1, col2, col3, rslt (filas normales)
  - NULL, col2, col3, rslt (subtotal por producto en una región)
  - NULL, NULL, col3, rslt (subtotal por producto)
  - col1, NULL, col3, rslt (subtotal por producto en un país)
  - col1, col2, NULL, rslt (subtotal por región del país)
  - NULL, col2, NULL, rslt (subtotal por región)
  - col1, NULL, NULL, rslt (este es el subtotal por país)
  - NULL, NULL, NULL, rslt (este es el total general, una sola fila)

GROUP BY GROUPING SETS

- La opción GROUPING SETS permite combinar varias cláusulas GROUP BY en una cláusula GROUP BY.
- Los resultados son equivalentes a usar la instrucción UNION ALL en los grupos especificados.

```
SELECT Country, Region, Product, SUM(Sales) AS TotalSales
FROM Sales
GROUP BY GROUPING SETS (Country)
```

- En el ejemplo anterior solo añade los subtotales por país.
- Con el grupo vacío () se especifica que se genere el total general.

```
SELECT Country, Region, Product, SUM(Sales) AS TotalSales
FROM Sales
GROUP BY GROUPING SETS (Country, ())
```

- Ahora se añade los subtotales por país y el total general.

GROUPING y GROUPING_ID

- Dado que las filas de totales muestran a NULL las columnas totalizadas, si la columna a totalizar puede contener nulos será complicado decidir, en las filas de resultados, si el NULL es de la columna o de la totalización.
- La función GROUPING indica con un 1 si una expresión de columna especificada en una lista GROUP BY es agregada o no con un 0.

```
SELECT Country, Region, Product, GROUPING(Product), SUM(Sales) AS TotalSales
FROM Sales
GROUP BY ROLLUP (Country, Region, Product);
```

- En el ejemplo anterior crea filas adicionales grupos para cada combinación de expresiones de columna en las listas siguientes.
  - col1, col2, NULL, 0, rslt (producto con NULL)
  - col1, col2, NULL, 1, rslt (subtotal por región del país)
- La función GROUPING_ID calcula el nivel de agrupación y devuelve una mascara de bits indicando las columnas que participan en la totalización (con 1 el bit de cada columna NULL por totalización) o un 0 si no es una fila de totalización.
- Para GROUPING_ID(Country, Region, Product)

0 → 000 → col1, col2, col3, rslt (filas normales)
6 → 110 → NULL, NULL, col3, rslt (subtotal por producto)
1 → 001 → col1, col2, NULL, rslt (subtotal por región del país)
3 → 011 → col1, NULL, NULL, rslt (este es el subtotal por país)
4 → 100 → NULL, col2, col3, rslt (subtotal por producto en una regió
2 → 010 → col1, NULL, col3, rslt (subtotal por producto en un país)
2 → 101 → NULL, col2, NULL, rslt (subtotal por región)
7 → 111 → NULL, NULL, NULL, rslt (este es el total general, una sola fila)

Ejercicios

- Ejercicios propuestos:
  - Crear un informe trimestral de las ventas clasificadas por zona y vendedor.
  - Totalizar para cada uno de los grupos
  - Mostrar una columna con los literales apropiado para cada uno de los totales.

DQL: Lenguaje de Consulta de Datos

CONSULTAS MULTITABLA

Múltiples Tablas

- CONJUNTOS (UNION, INTERSECT, EXCEPT)
  - Son utilizado para combinar los resultados de dos o más consultas en un solo conjunto de resultados.
- COMBINACIONES (JOINS)
  - Son utilizadas para recuperar datos de varias tablas mediante el producto cartesiano
- SUBCONSULTAS (SUB-QUERIES)
  - Son utilizadas para generar preguntas subsidiarias necesitadas por la consulta principal

Conjuntos

- Existen tres tipos de relaciones
  - UNION
  - INTERSECT
  - EXCEPT
- UNION [ALL]
  - Combinación de todas las filas del primer conjunto con todas las filas del segundo. Con ALL salen las duplicadas.
- INTERSECT
  - Combinación de las filas de los dos conjuntos, cuando la fila exista en ambos conjuntos
- EXCEPT
  - Combinación de las filas del primer conjunto que no estén en el segundo

Reglas para el uso de conjuntos

- Pueden ser encadenados en cualquier combinación
  Select ..... union select ..... intersect....
- Los conjuntos son evaluados de izquierda a derecha
- No existe jerarquía de precedencia en el uso de estos operadores, pero puede ser forzada mediante paréntesis
- Los operadores de conjuntos pueden emplearse con conjuntos de diferentes tablas siempre que se apliquen las reglas siguientes:
  - Las columnas son relacionadas en orden, de izquierda a derecha
  - Los nombres de las columnas son irrelevantes
  - Los tipos de datos deben coincidir
  - Los nombres de las columnas del primer conjunto se utilizaran en el conjunto resultante

Cláusula FROM

- La cláusula FROM determina las tablas de origen que se utilizarán en la instrucción SELECT
- La cláusula FROM puede contener tablas y operadores
- El conjunto de resultados de la cláusula FROM es una tabla virtual
- Las operaciones lógicas posteriores en la instrucción SELECT consumen esta tabla virtual
- La cláusula FROM puede establecer alias de tablas para su uso en fases posteriores de la consulta
- En la cláusula FROM pueden participar tablas, vistas, tablas derivadas, funciones de tipo tabla virtual, funciones rowset, pivoted/unpivoted y variables de tipo tabla
- Salvo las tablas y vistas, el resto requiere asociarles un alias
  FROM T [AS] Alias

Características de un producto cartesiano

- Se realizan por pares de conjuntos y el resultado con el siguiente conjunto.
- Combinación de todas las filas y columnas de un conjunto con todas las filas y columnas del otro conjunto.
- El resultado tiene tantas filas como el producto del número de filas de todos los participantes y tantas columnas como el sumatorio del número de columnas de todos los participantes.
- Una vez obtenido el conjunto resultante se eliminan las filas que no cumplan la clausula WHERE y se eliminan las columnas que no aparezcan en la clausula SELECT

Id	Name
1	Davis
2	Funk
3	King

Product
Alice Mutton
Crab Meat
Ipoh Coffee

Id	Name	Product
1	Davis	Alice Mutton
1	Davis	Crab Meat
1	Davis	Ipoh Coffee
2	Funk	Alice Mutton
2	Funk	Crab Meat
2	Funk	Ipoh Coffee
3	King	Alice Mutton
3	King	Crab Meat
3	King	Ipoh Coffee

Reglas para las combinaciones (JOINS)

- Se pueden obtener la combinación de tantas tablas como se quiera
- Las combinaciones se evalúan 2 a 2 de izquierda a derecha.
- Puede usarse más de una pareja de columnas para especificar una condición de combinación entre dos tablas
- Se pueden seleccionar columnas de todas las tablas implicadas en la combinación
- Si tenemos columnas con idéntico nombre en dos tablas de la combinación se deberán calificar las columnas con el nombre de la tabla al que pertenecen
- Como regla general, la combinación tendrá como mínimo tantas condiciones en la cláusula WHERE como número de tablas de la cláusula FROM menos uno

Tipos de sintaxis de Combinaciones

- ANSI SQL-92
  - Las tablas se combinan y establecen las combinaciones con el operador JOIN en la clausula FROM
  - Separación lógica entre el filtrado con el fin de combinar y el filtrado de los resultados en WHERE
  - Es la sintaxis recomendada.

```
SELECT ...
FROM   Table1 JOIN Table2
    ON <on_predicate>
```

- ANSI SQL-89
  - Las tablas se combinan en la clausula FROM separándolas por comas y se filtran en la clausula WHERE
  - Sintaxis no recomendada al complicar mucho el WHERE.

```
SELECT ...
FROM   Table1, Table2
WHERE  <where_predicate>
```


Tipos de Combinaciones

- CROSS
  - Combina todas las filas en ambas tablas (crea un producto cartesiano)

```
FROM t1 CROSS JOIN t2
```

- INNER
  - Se inicia con producto cartesiano; aplica el filtro para hacer coincidir las filas entre tablas basadas en el predicado

```
FROM t1 [INNER] JOIN t2 ON t1.column = t2.column [AND …]
```

- OUTER
  - Se inicia con producto cartesiano; se conservan todas las filas de la tabla designada mostrando las coincidencias con las filas de otra tabla recuperada y nulos como marcadores de posición cuando no hay coincidencias

```
FROM t1 { LEFT | RIGHT | FULL } [ OUTER ]  JOIN t2 ON t1.column = t2.column [AND …]
```

- SELF
  - Para las relaciones reflexivas, el producto cartesiano se realiza con la propia tabla, las tablas requieren un alias

```
FROM t [AS] t1 [INNER] JOIN t [AS] t2 ON t1.columnPK = t2.columnFK [AND …]
```

Ejercicios

- Ejercicios propuestos:
  - Completar los ejercicios anteriores sustituyendo los identificadores migrados (FK) por información significativa de la entidad (nombre, nombre y apellidos, …)
  - Crear un listado completo de pedidos sustituyendo los identificadores por sus descripciones.
  - Elaborar un listado de categorías, sub categorías y productos.
  - Comprobar que los totales de los pedidos son correctos usando las líneas como referencias.
  - Listar los productos que ha adquirido cada uno de los clientes.

Tablas Derivadas

- Se denomina tabla derivada a una consulta SELECT que participa en la clausula FROM
- Deben ir entre paréntesis y contar con un alias, tener nombres o alias únicos para todas las columnas y no utilizar la cláusula ORDER BY.

```
FROM (SELECT … ) [AS] Alias
```

- Se pueden anidar tablas derivadas en la definición de otras tablas derivadas.
- Se puede usar la característica de constructor con valores de tabla de Transact-SQL para especificar varias filas.

```
SELECT * FROM (VALUES (1, 'UNO'), (2, 'DOS'), (3, 'TRES')) AS
    MyTable(Id, Name);
```

CTE

- Especifica un conjunto de resultados temporal con nombre, conocido como expresión de tabla común (CTE).
- Se deriva de una consulta simple y se define en el ámbito de ejecución de una sola instrucción SELECT, INSERT, UPDATE, DELETE o MERGE.
- Una expresión de tabla común puede incluir referencias a ella misma (conocida como expresión de tabla común recursiva) y acepta múltiples referencias.

```
WITH Alias_CTE(column_name [ ,...n ] ) AS (
      SELECT ...
)
SELECT ...
FROM ... Alias_CTE
...
```


Funciones

- Funciones tabla virtual (TVF)
  - Funciones que devuelven un conjunto de resultados en formato tabla

    CREATE FUNCTION MyFunc(…)
    RETURNS TABLE …

    FROM MyFunc(…) [AS] f
- Funciones rowset
  - OPENQUERY
    - Ejecuta la consulta de paso a través especificada en el servidor vinculado especificado. (OLEDB)
    - OPENQUERY (linked_server ,'query')
  - OPENDATASOURCE
    - Proporciona información de conexión ad hoc como parte de un nombre de objeto de cuatro partes sin utilizar un nombre de servidor vinculado.
    - OPENDATASOURCE (provider_name, init_string)
  - OPENROWSET
    - Contiene toda la información de conexión necesaria para tener acceso a datos remotos desde un origen de datos OLE DB.
    - OPENROWSET('provider_name' , 'datasource' ; 'user_id' ; 'password', 'query')
  - OPENXML
    - Proporciona una vista de un conjunto de filas en un documento XML.
    - OPENXML(idoc int [in] , rowpattern nvarchar [in] , [flags byte [in]])
  - FROM OPENQUERY(…) [AS] rs

APPLY

- El operador APPLY permite combinar un conjunto con el resultado de la ejecución de una función de tipo tabla que se ejecuta para cada fila del conjunto tomando como argumento una columna de dicha fila.

  source { CROSS | OUTER } APPLY
  table_function(source.colum[, …]) [AS] alias
- Si se especifica CROSS, no se genera ninguna fila cuando la función devuelve un conjunto de resultados vacío.
- Si se especifica OUTER, se genera una fila para cada fila de source, incluso si la función devuelve un conjunto de resultados vacío.

PIVOT

- La cláusula PIVOT permite girar las filas en columnas en la salida de una consulta y aplicar una función de agregación
- Permite la generación de resultados en formato de tablas cruzadas.

```
FROM  (<SELECT query that produces the data>)
   AS <alias for the source query>
PIVOT (
     <aggregation function>(<column being aggregated>)
     FOR [<column that contains the values that will become
          column headers>]
     IN ( [first pivoted column], [second pivoted column],  ... [last
          pivoted column])
) AS <alias for the pivot table>
```

PIVOT

```
SELECT dept, sexo, sum(salar)
FROM templa
GROUP  BY dept, sexo
ORDER BY dept, sexo
```

```
DEP S  SUM(SALAR)
--- - -----------
A00 H     7575000
A00 M     5275000
B01 H     4125000
C01 M     9047000
D11 H    11642000
D11 M     8343000
D11       3225000
D21 H     7012000
D21 M     8080000
E01 H     4017500
E11 H     3155600

DEP S  SUM(SALAR)
--- - -----------
E11 M     7118000
E21 H     9531000
```

```
SELECT *
FROM (
        SELECT dept, sexo, salar FROM templa
) origen PIVOT (
        sum(salar)
        FOR sexo
        IN ('H','M')
) cruzada
ORDER BY dept
```

```
DEP         'H'         'M'
--- ---------- ----------
A00     7575000     5275000
B01     4125000
C01                 9047000
D11    11642000     8343000
D21     7012000     8080000
E01     4017500
E11     3155600     7118000
E21     9531000
```


UNPIVOT

- También tiene una cláusula UNPIVOT que gira columnas en filas en el resultado de una consulta.
- UNPIVOT es el inverso de PIVOT, pero sin la posibilidad de desagregar los datos agregados.
- La cláusula UNPIVOT permite separar una fila con columnas en tantas filas de una sola columna como columnas se indique, fundamentalmente para poder realizar operaciones de agregaciones con varias columnas

```
FROM table_source UNPIVOT(
        value_column FOR pivot_column IN ( column_name [ ,...n ] )
    ) [ [ AS ] table_alias ]
```

- Donde value_column es el nombre de la columna de salida que contendrá los valores de las columnas y pivot_column es el nombre de la columna de salida que contendrá los nombres de las columnas origen de los valores

UNPIVOT

```
SELECT *
FROM ParidadSalarial

SELECT *
FROM ParidadSalarial
    UNPIVOT(
        sumas
        FOR sexo
        IN (HOMBRES, MUJERES)
    )
```

DEP	HOMBRES	MUJERES
A00	7575000	5275000
B01	4125000	
C01		9047000
D11	11642000	8343000
D21	7012000	8080000
E01	4017500	
E11	3155600	7118000
E21	9531000	

DEP	SEXO	SUMAS
A00	HOMBRES	7575000
A00	MUJERES	5275000
B01	HOMBRES	4125000
C01	MUJERES	9047000
D11	HOMBRES	11642000
D11	MUJERES	8343000
D21	HOMBRES	7012000
D21	MUJERES	8080000
E01	HOMBRES	4017500
E11	HOMBRES	3155600
E11	MUJERES	7118000

Ejercicios

- Ejercicios propuestos:
  - Informe trimestral de ventas con una columna para cada trimestre.

Subconsultas

- En muchos casos para filtrar una consulta es necesario realizar una consulta para obtener los datos de filtrado.
- Las subconsultas son consultas anidadas: consultas dentro de consultas.
- Los resultados de la consulta interna se pasan a la consulta externa.
- La consulta interna actúa como una expresión desde la perspectiva de la consulta externa.
- Las subconsultas pueden ser escalares o multivaluadas.
- Las subconsultas pueden ser autónomas o correlacionadas
  - Las subconsultas autónomas no tienen dependencia en la consulta externa
  - Las subconsultas correlacionadas dependen de los valores de la consulta externa

Subconsultas escalares

- La subconsulta escalar devuelve un solo valor a la consulta externa
- Se puede utilizar en cualquier lugar donde se use una expresión de un solo valor: SELECT, WHERE, etc.
- Si la consulta interna devuelve un conjunto vacío, el resultado se convierte a NULL

```
SELECT productid, name, unitprice
FROM Sales.Products
WHERE unitprice > (
    SELECT MAX(unitprice)
    FROM Sales.Products)
```

- La construcción de la consulta externa determina si la consulta interna debe devolver un solo valor

Subconsultas multivaluadas

- La subconsulta multivalor devuelve múltiples valores como una única columna establecida en la consulta externa
- Los predicados deben utilizar operadores que soporten múltiples valores:
  - IN
  - { = | != | <> | > | >= | < | <= } { ANY | SOME | ALL }

```
SELECT productid, name, unitprice
FROM Sales.Products
WHERE productid in (
    SELECT productid
    FROM Sales.OrderDetails
    WHERE qty > 10)
```

- También se pueden expresar como una combinación.

Subconsultas correlacionadas

- Una subconsulta correlacionada se filtra con elementos de las tablas usadas en la consulta externa.
- Dependen de la consulta externa, no se puede ejecutar por separado
- Son más difíciles de probar que las subconsultas autocontenidas.
- Pueden ser, a su vez, escalares o multivaluadas.

```
SELECT productid, name,
    (SELECT Name FROM Category WHERE id = categoryID) category
FROM Sales.Products p
WHERE EXISTS (
    SELECT *
    FROM Sales.OrderDetails od
    WHERE od.Productid = p.productid)
```

- Se comporta como si la consulta interna se ejecutara una vez por cada fila externa.
- También se pueden expresar como una combinación.

Ejercicios

- Ejercicios propuestos:
  - Listar los productos de gama alta cuyo precio sea menor que alguno de los productos de gama baja
  - Listar todos los clientes que hayan comprado productos blancos
  - Listar todas la mujeres de los departamentos de producción y ventas
  - Calcular el salario anual de los empleados
  - Calcular la brecha salarial entre hombre y mujeres, casados y solteros

BÚSQUEDA DE TEXTO COMPLETO

Búsqueda de texto completo

- Proporciona una forma de realizar consultas en lenguaje natural al buscar, en columnas con datos basados en caracteres, valores que coincidan con el significado en lugar de con las palabras exactas o frases, a una cierta distancia las unas de las otras y coincidencias ponderadas.
- Para poder ejecutar consultas búsqueda de texto completo debe crear primero un catálogo de texto completo y un índice de texto completo en las vistas indexadas o tablas en las que quiere realizar una búsqueda.
- Para hacer las consultas de búsquedas de texto completo:
  - WHERE: CONTAINS y FREETEXT
  - FROM: CONTAINSTABLE y FREETEXTTABLE (KEY y RANK)

FREETEXT

- Es un predicado que se usa en la cláusula WHERE de una instrucción SELECT para realizar una búsqueda de texto completo simple en las columnas indexadas que contienen tipos de datos basados en caracteres. Este predicado busca valores que coincidan con el significado y no solo con la redacción exacta de las palabras en la condición de búsqueda. Cuando se usa FREETEXT, el motor de consultas de texto completo realiza internamente las siguientes acciones en cadena_freetext, asigna a cada uno de los términos una ponderación y busca las coincidencias:
  - Separa la cadena en palabras individuales basándose en límites de palabras (separación de palabras).
  - Genera formas no flexionadas de las palabras (lematización).
  - Identifica una lista de expansiones o reemplazos de los términos basándose en coincidencias en el diccionario de sinónimos.

```
FREETEXT ( { column_name | (column_list) | * } , 'freetext_string' [ ,
    LANGUAGE language_term ] )
```

CONTAINS

- Busca coincidencias precisas o aproximadas (menos precisas) de palabras o frases, palabras que se encuentran a cierta distancia de otra o coincidencias ponderadas.
- CONTAINS es un predicado que se usa en la cláusula WHERE de una instrucción SELECT para realizar una búsqueda de texto completo en las columnas indizadas que contienen tipos de datos basados en caracteres.
- CONTAINS puede buscar:
  - Una palabra o una frase, o una combinación de ellas que estén presentes o se excluyan.
  - El prefijo de una palabra o una frase.
  - Una palabra cerca de otra palabra o no.
  - Una palabra que sea una inflexión de otra (por ejemplo, las palabras controles, controladores, controlando y controlado son inflexiones de control).
  - Una palabra que sea un sinónimo de otra mediante un diccionario de sinónimos.

```
CONTAINS (
    {
      column_name | ( column_list ) | * | PROPERTY ( { column_name }, 'property_name' )
    }
    , '<contains_search_condition>'
    [ , LANGUAGE language_term ]
  )
```


FREETEXTTABLE y CONTAINSTABLE

- Las funciones FREETEXTTABLE y CONTAINSTABLE son las versiones tabla de resultados de las correspondientes funciones escalares FREETEXT y CONTAINS.
- Devuelven una tabla de cero, uno o más filas para las columnas que contienen valores seleccionados por la función escalar.
- Cada fila solo cuenta con dos columnas:
  - KEY: clave primaria (PK) de la fila que contiene la columna o columnas que satisfacen la condición de búsqueda de texto completo para realizar el JOIN.
  - RANK: un valor (de 0 a 1000) de clasificación por relevancia, solo indican un orden relativo de relevancia de las filas en el conjunto de resultados, cuanto mas alto mayor relevancia. Permite las ordenaciones.
- Con top_n_by_rank se limita el número de resultados:

```
FREETEXTTABLE (table , { column_name | (column_list) | * }, 'freetext_string'
    [ , LANGUAGE language_term ]
    [ , top_n_by_rank ] )

CONTAINSTABLE ( table , { column_name | ( column_list ) | * } , ' <contains_search_condition> '
    [ , LANGUAGE language_term]
  [ , top_n_by_rank ]
)
```

Lenguaje de Manipulación de Datos

DML

Tabla5.sql

Lenguaje de Manipulación de Datos (DML)

- Cuenta con las instrucciones para añadir, modificar y borrar las filas de las tablas.
- Solo pueden afectar a una única tabla, salvo las automatizaciones de las integridad referencial.
- Pueden usarse con vistas para manipular los datos de las tablas subyacentes bajo ciertas condiciones:
  - La modificación afecte sólo a una de las tablas subyacentes.
  - En la inserción, deben aparecer todas las columnas requeridas que no dispongan de valor predeterminado.
  - Las columnas a modificar en la vista deben hacer referencia directa a los datos subyacentes de las columnas de la tabla (no puede ser resultado de una función de agregado o cálculo).
  - Las columnas que se modifican no pueden verse afectadas por cláusulas GROUP BY, HAVING o DISTINCT.
  - La cláusula WITH CHECK OPTION exige que todas las instrucciones de modificación de datos ejecutadas en la vista se ajusten a los criterios especificados en la instrucción SELECT que define la vista.

INSERT

- Agrega una o varias filas a una tabla o una vista.
  INSERT table_or_view_name [(column_list)]
  VALUES ({ DEFAULT | NULL | expression } [,...n]) [,...n]
- Las columnas se identifican por su nombre.
- La asociación de columna y su valor es posicional.
- La lista de columnas es opcional si los valores se encuentran en el mismo orden que las columnas de la tabla.
- Los valores deben cumplir con las restricciones de la columna.
- Los valores constantes de tipo carácter o fecha deben ir encerrados entre comillas simples (‘ ‘).
- DEFAULT representa el valor por defecto de la columna y NULL la ausencia de valor.
- Deben aparecer todas las columnas requeridas que no dispongan de valor predeterminado.

INSERT

- Se pueden insertar en la misma instrucción varios conjuntos de valores separando los () por comas.
- INSERT ... SELECT se usa para insertar el conjunto de resultados de una consulta en una tabla existente
  INSERT table_or_view_name [(column_list)]
  SELECT …
- INSERT ... EXEC se utiliza para insertar el resultado de un procedimiento almacenado o una expresión de SQL dinámico en una tabla existente
  INSERT table_or_view_name [(column_list)]
  EXEC …
- BULK INSERT importa un archivo de datos en una tabla o vista con un formato especificado por el usuario.

SELECT INTO

- SELECT ... INTO es similar a INSERT ... SELECT pero SELECT ... INTO crea una nueva tabla cada vez que se ejecuta la instrucción

```
SELECT { column_list | * }
INTO table_name
FROM …
```

- Para crear la tabla copia los nombres de columnas y deduce los tipos de datos y aceptación de nulos usando el conjunto de resultados.
- No copia restricciones ni índices.
- En caso de existir la tabla de destino dará error, es necesario borrar manualmente la tabla antes de volver a ejecutar la consulta.
- Una vez creada la tabla realiza un INSERT ... SELECT del conjunto de resultados en la tabla recién creada.

UPDATE

- La modificación de los datos ya insertados en una tabla se realiza con el comando UPDATE

```
UPDATE [ TOP ( expression ) [ PERCENT ] ]  table_or_view_name
SET     column_name { = | += | -= | *= | /= | %= | &= | ^= | |= } expression [,n]
[FROM table_or_view_name alias JOIN …]
[WHERE condition]
```

- SET
  - Indica la columna a modificar y la expresión que generara el nuevo valor que tomará la columna.
  - Se pueden modificar varias o todas las columnas, pero al meno debe haber una.
  - El valor puede ser DEFAULT o NULL
- FROM
  - Se puede combinar la tabla con otras para obtener los valores que participaran en las expresiones que obtienen los nuevos valores.
- WHERE
  - Indica la fila o filas en las que se va a realizar la modificación
  - Si no se cumple la condición no se modificará ninguna fila
  - Si se omite, la modificación se realiza en todas las filas de la tabla
  - TOP puede limitar el número de filas afectadas

DELETE

- Utilizada para borrar filas de una tabla

  DELETE [TOP (expression) [PERCENT]] table_or_view_name
  [FROM table_or_view_name alias JOIN …]
  [WHERE condition]
- FROM
  - Se puede combinar la tabla con otras para obtener la condición de borrado.
- WHERE
  - Indica la fila o filas en las que se va a realizar el borrado
  - Si no se cumple la condición no se borrara ninguna fila
  - Si se omite, se borraran todas las filas de la tabla
  - TOP puede limitar el número de filas afectadas
- TRUNCATE TABLE es similar a la instrucción DELETE sin una cláusula WHERE; no obstante, TRUNCATE TABLE es más rápida y utiliza menos recursos de registros de transacciones y de sistema al desasociar el almacenamiento físico de toda la tabla.

  TRUNCATE TABLE table_name

MERGE

- Ejecuta operaciones de inserción, actualización o eliminación en una tabla de destino a partir de los resultados de una combinación con una tabla de origen.
- Resume en una única instrucción lo que anteriormente requería múltiples instrucciones.

```
MERGE [ TOP ( expression ) [ PERCENT ] ]
  [ INTO ] <target_table>
  USING <table_source>
  ON <merge_search_condition>
  [ WHEN MATCHED [ AND <clause_apply_condition> ]
    THEN <merge_matched> ] [ ...n ]
  [ WHEN NOT MATCHED [ BY TARGET ] [ AND <clause_apply_condition> ]
    THEN <merge_not_matched> ]
  [ WHEN NOT MATCHED BY SOURCE [ AND <clause_apply_condition> ]
    THEN <merge_matched> ] [ ...n ]
```

- INTO
  - Fija la tabla a actualizar por lo que se omite en las instrucciones INSERT INTO, UPDATE y DELETE.
- USING
  - tablas, vistas, CTE, tablas derivadas, funciones de tipo tabla virtual y variables de tipo tabla

MERGE

- ON
  - Condición que establece las coincidencias o no coincidencias entre el origen y el destino, como las operaciones se realizan a nivel de fila no es necesario el WHERE en UPDATE y DELETE.
- WHEN MATCHED
  - Coincidencia en los dos que requiere una modificación o borrado:
    { UPDATE SET <set_clause> | DELETE }
- WHEN NOT MATCHED [BY TARGET]
  - No existe en el destino por lo que habrá que insertarla
    INSERT [(column_list)] VALUES (values_list)
- WHEN NOT MATCHED BY SOURCE
  - Existe en el destimo pero no en el origen, en algunos casos desencadena el borrado en el destino.
    DELETE
- AND
  - Permite disponer de multiples clausulas MACHED o NOT MATCHED que se aplicaran solo si se cumple la condición asociada con el AND.
- TOP
  - Limita el número de filas afectadas

Cláusula OUTPUT

- Devuelve información de las filas afectadas por una instrucción INSERT, UPDATE, DELETE o MERGE, o expresiones basadas en esas filas.
- Estos resultados se pueden devolver a la aplicación de procesamiento para que los utilice en mensajes de confirmación, archivado y otros requisitos similares de una aplicación.
- Los resultados también se pueden insertar en una tabla o variable de tabla.
- Además, puede capturar los resultados de una cláusula OUTPUT en una instrucción anidada INSERT, UPDATE, DELETE o MERGE, e insertar los resultados en una tabla de destino o vista.

```
OUTPUT <dml_select_list> [
    INTO { @table_variable | output_table } [
        ( { { DELETED | INSERTED | from_table_name } . { * | column_name }
        | $action | scalar_expression } [ [AS] column_alias_identifier ]  [ ,...n ]  )
        ] ]
```

Columnas especiales

- Columnas de numeración automática (Autonúmericas)
  - Identidad, una sola columna por tabla
  - Inicialización de identidad: valor de inicial de una columna de identidad
  - Incremento de identidad: valor en que se incrementa la columna de identidad
  - Vista sys.identity_columns columna last_value
- Columnas de identificadores exclusivos globales (GUID)
  - Es RowGuid: Muestra si SQL Server utiliza la columna como una columna ROWGUID. Este valor se puede establecer como Sí sólo para una columna de identidad
  - Valor predeterminado: newid()
- Columnas calculadas
  - Fórmula: Muestra la fórmula para una columna calculada
  - Ignora tipo y longitud

Columnas automáticas

- Columnas de solo lectura, se omiten en los INSERT
- IDENTITY
  - Columna que genera números secuenciales automáticamente para la inserción en una tabla. Se puede especificar valores de semilla e incremento opcionales. Solo una columna en la tabla puede tener la propiedad IDENTITY definida, destinada a ser la clave primaria.
  - Las Secuencias son objetos que surgen como alternativa a IDENTITY para generar claves primarias que se utilizan en varias tablas: SELECT NEXT VALUE FOR dbo.MySeq
- ROWGUID
  - Columna con un identificador exclusivo global (GUID), destinada a ser la clave primaria distribuida. Valor predeterminado: newid()
- ROWVERSION (timestamp)
  - Un contador único para toda la base de datos que se actualiza cada vez que se actualiza una fila destinado al control de cambios en las filas.
- Columnas calculadas
  - Obtienen el valor de una fórmula de calculo, ignora el tipo y longitud

Ejercicios

- Ejercicios propuestos:
  - Añadir un nuevo pedido con al menos tres productos
  - Subir un 10% el precio de venta de los productos de la categoría Accesorios si son de gama alta.

Lenguaje de Validación de Datos

DVL

Transacciones

- Una transacción es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo.
- Una unidad lógica de trabajo debe exhibir cuatro propiedades, conocidas como propiedades ACID (atomicidad, coherencia, aislamiento y durabilidad), para ser calificada como transacción:
  - Atomicidad: Una transacción debe ser una unidad atómica de trabajo, tanto si se realizan todas sus modificaciones en los datos, como si no se realiza ninguna de ellas.
  - Coherencia**:** Cuando finaliza, una transacción debe dejar todos los datos en un estado coherente, es decir, se deben cumplir todas las reglas de integridad de todos los datos.
  - aIslamiento**:** Las modificaciones realizadas por transacciones simultáneas se deben aislar de las modificaciones llevadas a cabo por otras transacciones simultáneas.
  - Durabilidad**:** Una vez concluida una transacción, sus efectos son permanentes en el sistema. Las modificaciones persisten aún en el caso de producirse un error del sistema.

Especificar y exigir transacciones

- Los programadores de SQL son los responsables de iniciar y finalizar las transacciones en los puntos que exijan la coherencia lógica de los datos.
- El programador debe asegurar la atomicidad de las transacciones.
- SQL Server proporciona:
  - Servicios de bloqueo que preservan el aislamiento de la transacción.
  - Servicios de registro que aseguran la durabilidad de la transacción:
    - Aún en el caso de que falle el hardware del servidor, el sistema operativo o el propio SQL Server, SQL Server utiliza registros de transacciones, al reinicio, para deshacer automáticamente las transacciones incompletas en el momento en que se produjo el error en el sistema.
  - Características de administración de transacciones que exigen la atomicidad y coherencia de la transacción:
    - Una vez iniciada una transacción, debe concluirse correctamente o SQL Server deshará todas las modificaciones de datos realizadas desde que se inició la transacción.

Tipos de transacciones

- Transacciones de confirmación automática
  - Éste es el modo predeterminado de SQL Server. Cada instrucción individual de Transact-SQL se confirma cuando termina. No tiene que especificar instrucciones para controlar las transacciones.
- Transacciones explícitas
  - Se inicia con BEGIN TRANSACTION, se confirma con COMMIT y se descarta con ROLLBACK.
- Transacciones implícitas
  - Establezca el modo de transacción implícita a través de una función de la API o la instrucción SET IMPLICIT_TRANSACTIONS ON de Transact-SQL. La siguiente instrucción inicia automáticamente una nueva transacción. Cuando se concluye la transacción, la instrucción de Transact-SQL siguiente inicia una nueva transacción.
- Transacciones distribuidas
  - Implican a varias instancias y se dividen en Fases de preparación y de confirmación.

Lenguaje de Validación de Datos

- Para fijar el nivel de aislamiento que determina el uso de los bloqueos

```
SET TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED | READ COMMITTED
        | REPEATABLE READ | SNAPSHOT | SERIALIZABLE}[;]
```

- Para indicar que las transacciones se abran automáticamente:

```
SET IMPLICIT_TRANSACTIONS ON
```

- Para abrir explícitamente una transacción

```
BEGIN { TRAN | TRANSACTION }
   [ { transaction_name | @tran_name_variable }
    [ WITH MARK [ 'description' ] ]
   ][ ; ]
```

- Se pueden anidar transacciones para confirmar o descartar parcialmente partes de la transacciones, pendientes de la confirmación o descarte de la transacción principal.
- Se pueden marcar las transacciones en el registro de transacción para facilitar la restauración de la base de datos hasta el momento de iniciar la transacción o de su confirmación.

Lenguaje de Validación de Datos

- Para abrir una transacción distribuida:
  BEGIN DISTRIBUTED { TRAN | TRANSACTION } [name | @variable]
- Para confirmar la transacción como correcta y finalizarla:
  COMMIT { TRAN | TRANSACTION } [name | @variable]]
  COMMIT [WORK]
- Para establecer un punto de retorno dentro de una transacción:
  SAVE { TRAN | TRANSACTION } { savepoint_name|@savepoint_variable }
- Para descartar los cambios realizados en la transacción y finalizarla:
  ROLLBACK {TRAN | TRANSACTION} [trans_name | @tran_variable | savepoint_name]
  ROLLBACK [WORK]
- @@TRANCOUNT:
  - Devuelve el número de transacciones activas de la conexión actual.

Lenguaje de Definición de Datos

DDL

Lenguaje de Definición de Datos DDL

- SQL permite crear, modificar y borrar la estructura física de los objetos de la base de datos
  - CREATE
  - ALTER
  - DROP
  - RENAME
  - ENABLE TRIGGER
  - DISABLE TRIGGER
  - UPDATE STATISTICS
- Las instrucciones DDL no permiten rollback de la acción realizada
- Deben ejecutarse independientemente, con una directiva GO después de cada una.
- SQL Server dispone de un interfaz gráfico que hace innecesario el uso de las instrucciones DDL en la mayoría de los casos.

Bases de datos

- Creación de una BBDD
  - Mediante el SQL Server Management Studio
  - Mediante Transact-SQL
    - Con la instrucción CREATE DATABASE
    - Existe una plantilla para la creación en el SSMS
    - Se puede generar el script de creación en el SSMS, y ejecutarlo posteriormente como sentencia SQL
- Las Bases de Datos heredan las características de la Base de Datos model
  - Algunas propiedades pueden ser determinadas durante la creación de la BBDD
- Se pueden modificar posteriormente
  - Mediante el SSMS
  - Mediante Transact-SQL
    - Exec sp_dboption nombre_bd, opcion, TRUE | FALSE
    - Mediante el comando ALTER DATABASE
- Para ver las propiedades actuales de una Base de Datos
  - Exec sp_dboption nombre_bd

Valores predeterminados

- Los valores predeterminados especifican el valor por defecto de una columna si no se especifica un valor al insertar las filas
- Para crear un nuevo valor predeterminado es necesario asignar:
  - Nombre del valor predeterminado
  - Valor que puede ser cualquier expresión cuyo resultado sea una constante: constantes, funciones integradas o expresiones matemáticas

Hoy: getdate()
Mañana: DATEADD(day, 1, getdate())
Cero: 0

Reglas

- Las reglas contienen la condición que define los valores permitidos para una columna o para un tipo de dato determinado (por compatibilidad con versiones anteriores)
- Para crear una nueva regla es necesario asignar:
  - Nombre de la regla
  - Expresión condicional, válida en una cláusula WHERE, que incluye una variable local que representa el valor a comprobar (@valor)
- Una regla se pude utilizar en tantos objetos como sea necesario.
- Se vinculan mediante Transact-SQL

```
CREATE RULE id_chk AS @id BETWEEN 0 and 10000
sp_bindrule id_chk, 'cust_sample.cust_id‘
```

Números de secuencia (2012)

- Una secuencia es un objeto enlazado a un esquema definido por el usuario que genera una secuencia de valores numéricos según la especificación con la que se creó la secuencia.
- A diferencia de los valores de columnas de identidad que se generan cuando se insertan filas, una aplicación puede obtener el número de secuencia siguiente sin insertar la fila llamando a la función NEXT VALUE FOR .
- Se asigna el número de secuencia cuando se llama a NEXT VALUE FOR aun cuando el número nunca se inserta en una tabla.
- La función NEXT VALUE FOR se puede utilizar como valor predeterminado para una columna en una definición de tabla.
- Se utiliza sp_sequence_get_range para obtener un intervalo de varios números de secuencia enseguida.

Tipos de Tablas

- Definición:
  - Nombre y ubicación
  - Columnas
  - Claves y restricciones
- Tipos de Tablas
  - Tablas del sistema
  - Tablas de datos
  - Tablas temporales
    - Locales (#) y Globales (##)
  - Tablas anchas (1.024 → 30.000 columnas)
  - Tablas internas (variables)

Definición de las columnas

- Nombre de columna
- Tipo de datos
- Longitud de los datos
  - Sólo puede cambiar la longitud de los tipo de datos sin longitud fija
  - Para los tipos decimal y numéricos
    - Precisión: Muestra el número máximo de dígitos
    - Escala: Muestra el número máximo de dígitos decimales
- Aceptación de valores NULL (Restricción)
- Descripción
- Valor predeterminado
- Intercalación (sólo para columnas de texto)

Columnas especiales

- Columnas de numeración automática (Autonúmericas)
  - Identidad, una sola columna por tabla
  - Inicialización de identidad: valor de inicial de una columna de identidad
  - Incremento de identidad: valor en que se incrementa la columna de identidad
  - Vista sys.identity_columns columna last_value
- Columnas de identificadores exclusivos globales (GUID)
  - Es RowGuid: Muestra si SQL Server utiliza la columna como una columna ROWGUID. Este valor se puede establecer como Sí sólo para una columna de identidad
  - Valor predeterminado: newid()
- Columnas calculadas
  - Fórmula: Muestra la fórmula para una columna calculada
  - Ignora tipo y longitud

Restricciones

- Sirven para definir reglas relativas a los valores permitidos en las columnas
- Es el mecanismo utilizado para reforzar la integridad en las Bases de Datos que SQL Server exige automáticamente
- Las restricciones establecen los valores aceptados por las columnas: nulos, rangos de valores, duplicados, claves principales o ajenas.
- Se dividen en restricciones de tabla y de columna.
- Admite cinco clases de restricciones.
  - NOT NULL especifica que la columna no acepta valores NULL (Obligatorio).
  - Las restricciones CHECK exigen la integridad del dominio mediante la limitación de los valores que se pueden asignar a una columna.
  - Las restricciones UNIQUE exigen la unicidad de los valores de un conjunto de columnas (conlleva un índice).
  - Las restricciones PRIMARY KEY identifican la columna o el conjunto de columnas cuyos valores identifican de forma unívoca cada una de las filas de una tabla (NOT NULL + UNIQUE) (conlleva un índice).
  - Las restricciones FOREIGN KEY identifican las relaciones entre las tablas (REFERENCES).

Relaciones

- Se definen a nivel de tabla.
- Puede crear una relación entre las tablas en un diagrama de base de datos para mostrar cómo se vinculan las columnas de una tabla a las columnas de otra tabla.
- Permiten exigir la integridad referencial:
  - No se puede especificar un valor en la columna de clave externa de la tabla relacionada si ese valor no existe en la clave principal de la tabla relacionada (o NULL si la columna lo acepta).
  - No se puede eliminar una fila de una tabla de claves principales si existen filas que coinciden con ella en una tabla relacionada.
  - No se puede cambiar un valor de clave principal en la tabla de clave principal si esa fila tiene filas relacionadas.
- Pueden automatizar el proceso en cascada.

Seguimiento de cambios de datos

- A partir del SQL Server 2008 se proporciona dos características que realizan el seguimiento de los cambios en los datos de una base de datos: captura de datos modificados y seguimiento de cambios.
- Estas características permiten a las aplicaciones determinar los cambios de DML (operaciones de inserción, actualización y eliminación) que se realizaron en las tablas de usuario de una base de datos.
- La captura de datos modificados y el seguimiento de cambios pueden habilitarse en la misma base de datos; no se requiere ninguna consideración especial.

Diagramas de Bases de Datos

- Herramienta visual que permite diseñar y ver una base de datos completa o parcialmente
- Utiliza como repositorio la tabla de sistema: dbo.sysdiagrams.

Vistas

- Una vista es una tabla virtual cuyo contenido está definido por una consulta.
  - Combinar columnas de varias tablas de forma que parezcan una sola tabla.
  - Agregar información agrupada o calculada en lugar de presentar los detalles.
  - Unir información de varios resultados
  - Restringir el acceso del usuario a filas concretas de una tabla.
  - Restringir el acceso del usuario a columnas específicas.
- Tipos:
  - Vistas estándar
  - Vistas indexadas o materializada
  - Vistas con particiones o distribuidas
  - Vistas parametrizadas (funciones de tipo tabla)

Vistas

- Se puede modificar los datos mediante una vista.
  - Son actualizables (UPDATE, DELETE o INSERT) mientras la modificación afecte sólo a una de las tablas base.
  - Las columnas a modificar en la vista deben hacer referencia directa a los datos subyacentes de las columnas de la tabla (no puede ser resultado de una función de agregado o cálculo).
  - Deben aparecer todas las columnas requeridas que no dispongan de valor predeterminado.
  - Las columnas que se modifican no pueden verse afectadas por cláusulas GROUP BY, HAVING o DISTINCT.
  - Permite el uso de desencadenadores INSTEAD OF
  - La cláusula WITH CHECK OPTION exige que todas las instrucciones de modificación de datos ejecutadas en la vista se ajusten a los criterios especificados en la instrucción SELECT que define la vista.

Vistas

- Una vista distribuida combina los datos procedentes de un conjunto de tablas miembro en uno o más servidores, y hace que los datos parezcan proceder todos de una sola tabla que combinado con desencadenadores INSTEAD OF simulan tablas particionadas o distribuidas.
- La cláusula WITH ENCRYPTION convertirá el texto original en un formato ofuscado.
- Las vistas, las tablas o las funciones que participan en una vista creada con la cláusula SCHEMABINDING no se pueden quitar, a menos que se quite o cambie esa vista de forma que deje de tener un enlace de esquema.
- Además, las instrucciones ALTER TABLE sobre tablas que participan en vistas que tienen enlaces de esquemas provocarán un error si estas instrucciones afectan a la definición de la vista.

Índices

- Compuestos por una o varias columnas ordenadas ascendente o descendentemente
- Guardan clave indexación, clave en índice agrupado o marcador fila real y columnas incluidas.
- Se pueden crear sobre Tablas, Vistas y columnas calculadas.
- Los dos tipos:
  - Agrupado
  - No agrupado
- Un índice único garantiza que la columna indexada no contiene valores no nulos duplicados.
- Índices y Restricciones:
  - Las restricciones PRIMARY KEY crean índices agrupados de forma automática de tipo UNIQUE.
  - Las restricciones UNIQUE crean índices UNIQUE de forma automática.
  - Las restricciones FOREIGN KEY no crean índices, pero suelen ser convenientes.
- Los no agrupados permiten columnas INCLUIDAS

Índices

- El factor de relleno determina la densidad de ocupación de páginas por parte del índice.
- El número de índices afecta al rendimiento y ocupación de la base de datos.
- El Asistente para optimización de índices suministra recomendaciones en función a la monitorización del sistema.
- Limitaciones
  - Sólo el propietario de la tabla puede crear índices en la misma tabla.
  - Un índice agrupado por tabla.
  - 249 índices como máximo (incluidos los creados por restricciones PRIMARY KEY o UNIQUE).
  - 16 columnas como máximo.
  - El tamaño máximo de la clave de indexación es 900 bytes.
- Índices especiales:
  - Texto completo, XML y Espaciales

Estadísticas

- SQL Server (2005…) permite crear información estadística acerca de la distribución de valores en una columna.
- El optimizador de consultas utiliza esta información estadística para determinar el plan de consultas óptimo realizando una estimación del costo de usar un índice para evaluar la consulta.
- Se almacenan con las misma estructura que los índices, sustituyendo el localizador por un contador.
- Se pueden crear automáticamente o manualmente. Es necesario actualizarlas periódicamente.

Procedimientos almacenados

- Un procedimiento almacenado es un grupo de instrucciones Transact-SQL compiladas en un único plan de ejecución, se almacena sys.sql_modules y en cache.
- Los procedimientos almacenados:
  - Permiten una programación modular.
  - Permiten una ejecución más rápida.
  - Pueden reducir el tráfico de red.
  - Pueden utilizarse como mecanismo de seguridad.
  - Un máximo de 2.100 parámetros.

Procedimientos almacenados

- Tipos:
  - Procedimientos almacenados del sistema (sp_)
  - Procedimientos almacenados definidos por el usuario
    - Transact-SQL y CLR
  - Procedimientos almacenados extendidos (xp_)
  - Procedimientos almacenados temporales
    - Locales (#) y Globales (##)
- Los procedimientos almacenados extendidos permiten crear rutinas externas propias, en un lenguaje de programación como C.
- Se pueden crear procedimientos almacenados temporales (agregando el prefijo # o ##) que dejan de existir cuando se cierra SQL Server.
- Los procedimientos de inicio deben estar en la base de datos master y no pueden contener parámetros de entrada (INPUT) ni salida (OUTPUT). La ejecución de los procedimientos almacenados se inicia cuando se recupera la base de datos master en el inicio.
  - sp_procoption 'procedure', 'startup', 'on'

Procedimientos almacenados

- Devuelven datos de cuatro formas distintas:
  - Parámetros de salida: datos o cursores.
  - Códigos de retorno, que siempre son un valor entero.
  - Un conjunto de resultados por una o varias instrucciones SELECT.
  - Un cursor global al que se puede hacer referencia desde fuera del procedimiento almacenado.
- Cláusulas:
  - WITH RECOMPILE (en el procedimiento o en el EXEC)
  - WITH ENCRYPTION (convertirá el texto original en un formato ofuscado)
  - { EXEC | EXECUTE } AS { CALLER | SELF | OWNER | 'user_name' }

Funciones definidas por el usuario

- Las funciones son subrutinas formadas por una o varias instrucciones Transact-SQL o CLR que se pueden utilizar para encapsular un código con el fin de utilizarlo de nuevo posteriormente.
- Son subrutinas que se utilizan para encapsular lógica que se ejecuta frecuentemente que no modifique la base de datos.
- Dos tipos de funciones: integradas y definidas por el usuario.
- Las funciones definidas por el usuario pueden tener parámetros de entrada y devuelven un único valor (dato o tabla interna)
- SQL Server admite cuatro tipos de funciones definidas por el usuario:
  - Funciones escalares
  - Funciones de valores de tabla en línea
  - Funciones de valores de tabla de múltiples instrucciones
  - Funciones de agregado (solo CLR)
- Todas las funciones son deterministas o no deterministas:
  - Las funciones deterministas siempre devuelven el mismo resultado cada vez que son llamadas con un conjunto específico de valores de entrada.
  - Las funciones no deterministas podrían devolver resultados diferentes cada vez que son llamadas con un conjunto específico de valores de entrada.

Desencadenadores DML

- Un desencadenador es un tipo especial de procedimiento almacenado que entra en vigor cuando se modifican datos en una tabla especificada utilizando una o más operaciones de modificación de datos: UPDATE (actualización), INSERT (inserción) o DELETE (eliminación).
- Los desencadenadores pueden exigir restricciones más complejas que las definidas con restricciones CHECK y ermiten implementar las reglas de negocio.
- Pueden realizar cambios en cascada a través de tablas relacionadas de la base de datos.
- Los desencadenadores son automáticos: se activan inmediatamente después de que se efectúen modificaciones en los datos de la tabla, como una entrada manual o una acción de la aplicación.
  - Se pueden habilitar y deshabilitar.
  - Se pueden definir varios por tabla, cada uno para una o varias operaciones.
  - Se pude fijar cual es el primero y el ultimo en ejecutarse: sp_settriggerorder 'First|None|Last'

Desencadenadores DML

- Se pueden implementar en Transact-SQL o CLR
- Los desencadenadores también pueden evaluar el estado de una tabla antes y después de realizar una modificación de datos y actuar en función de la diferencia.
- Seudótablas:
  - INSERTED: nuevos valores
  - DELETED: valores antiguos
- Funciones:
  - UPDATE(column) y COLUMNS_UPDATED( )
- Momentos:
  - AFTER
  - INSTEAD OF

Desencadenadores DDL/Logon

- Pueden utilizarse para tareas administrativas como auditar y regular las operaciones de base de datos: CREATE, ALTER y DROP.
  - ámbito del servidor
  - ámbito de la base de datos
- La información acerca de un evento que activa un desencadenador DDL se captura mediante la función EVENTDATA (formato XML)

Sinónimos

- Un sinónimo es un nombre alternativo para un objeto (local o remoto) de ámbito de esquema.
- Proporciona una capa de abstracción que protege una aplicación cliente de cambios hechos en el nombre o la ubicación del objeto base.
- Se pueden proporcionar sinónimos a:
  - Tablas, Vistas, Procedimientos Almacenados y Funciones.

Lenguaje de Control de Datos

DCL

GRANT

- Concede permisos sobre un elemento protegible a una entidad de seguridad.
- El concepto general es GRANT <un permiso> ON <un objeto> TO <un usuario, inicio de sesión o grupo>.

```
GRANT { ALL [ PRIVILEGES ] }
    | permission [ ( column [ ,...n ] ) ] [ ,...n ]
    [ ON [ class :: ] securable ] TO principal [ ,...n ]
    [ WITH GRANT OPTION ] [ AS principal ]
```

DENY

- Deniega un permiso a una entidad de seguridad.
- Evita que la entidad de seguridad herede permisos por su pertenencia a grupos o roles.
- DENY tiene prioridad sobre todos los permisos, aunque DENY no se aplica a los propietarios de objetos o miembros del rol fijo de servidor sysadmin.

```
DENY   { ALL [ PRIVILEGES ] }
    | <permission>  [ ( column [ ,...n ] ) ] [ ,...n ]
  [ ON [ <class> :: ] securable ]
  TO principal [ ,...n ]
  [ CASCADE] [ AS principal ]
```


REVOKE

- Quita la concesión o denegación previamente establecida.
- Revocar una concesión no equivale a una denegación, dado que una entidad se seguridad puede establecer concesiones a múltiple niveles, se conserva el permiso mientras se mantenga al menos una concesión y no haya ninguna denegación.

```
REVOKE [ GRANT OPTION FOR ]
    {
     [ ALL [ PRIVILEGES ] ]
     |
          permission [ ( column [ ,...n ] ) ] [ ,...n ]
    }
    [ ON [ class :: ] securable ]
    { TO | FROM } principal [ ,...n ]
    [ CASCADE] [ AS principal ]
```

PROGRAMACIÓN CON T-SQL

Programación

- No todas las operaciones con la base de datos se pueden realizar directamente con las instrucciones SQL.
- Para ello Transact-SQL incorpora instrucciones procedurales como la declaración de variables, flujo de control, excepciones, … para poder programar las acciones.
- Los programas se pueden almacenar en:
  - Ficheros: procesos por lotes
  - Base de datos: Procedimientos almacenados, funciones y desencadenadores.

Procesos por lotes

- Los lotes T-SQL son colecciones de una o más declaraciones T-SQL enviadas al SQL Server como una unidad para análisis, optimización y ejecución.
- Los lotes se terminan con GO por defecto por lo que los ficheros pueden contener varios lotes
- Los lotes fijan los límites para el alcance variable.
- Las instrucciones DDL no pueden combinarse con otras en el mismo lote
- Los lotes se analizan para la sintaxis como una unidad y los errores de sintaxis hacen que todo el lote sea rechazado
- Los errores de tiempo de ejecución pueden permitir que el lote continúe después de la falla, de manera predeterminada
- Los lotes pueden contener código de manejo de errores.

Variables

- Las variables se declaran en el cuerpo de un proceso por lotes o un procedimiento con la instrucción DECLARE, y se les asignan valores con una instrucción SET o SELECT.
- Después de la declaración, todas las variables se inicializan como NULL, a menos que se proporcione un valor como parte de la declaración.
- Se declaran antes de ser usados en cualquier punto.
  DECLARE @NOMBRE TIPO (LONGITUD) = VALOR,…
- Ambito.:
  - Variables locales: @proceso
  - Variables globales: @@conexión
- Asignación:
  SET @NOMBRE = Valor
  SELECT @NOMBRE = Valor [| columna FROM …]
- Mostrar valor:
  PRINT @NOMBRE
  SELECT @NOMBRE

IF...ELSE

- Impone condiciones en la ejecución de una instrucción de Transact-SQL.
- La instrucción Transact-SQL que sigue a una palabra clave IF y a su condición se ejecuta si la condición se cumple: la expresión booleana devuelve TRUE.
- La palabra clave opcional ELSE introduce otra instrucción Transact-SQL que se ejecuta cuando la condición IF no se cumple: la expresión booleana devuelve FALSE.
- Ejecutan una sola instrucción o un bloque de instrucciones marcado el inicio con BEGIN y el final con END.

```
IF Boolean_expression
    { sql_statement | statement_block }
[ ELSE
    { sql_statement | statement_block } ]
```


WHILE

- Establece una condición para la ejecución repetida de una instrucción o bloque de instrucciones SQL.
- Las instrucciones se ejecutan repetidamente siempre que la condición especificada sea verdadera.
- Se puede controlar la ejecución de instrucciones en el bucle WHILE con las palabras clave BREAK y CONTINUE.

```
WHILE Boolean_expression
    { sql_statement | statement_block }
```

- Ejecutan una sola instrucción o un bloque de instrucciones marcado el inicio con BEGIN y el final con END.
- Hay que vigilar la condición para no incurrir en un bloque infinito.
- BREAK produce la salida del bucle WHILE más interno y continua por la siguiente instrucción.
- CONTINUE salta a evaluar la condición que determina si sigue en el bucle o se sale.
- BREAK y CONTINUE deben estar dentro de un IF.

Saltos y Esperas

- Saltos:

  NOMBRE_ETIQUETA:

  GOTO NOMBRE_ETIQUETA

  CONTINUE

  BREAK

  RETURN [integer_expression]

- Espera:

  WAITFOR DELAY 'período de tiempo que hay que esperar'

  WAITFOR TIME 'hora a la que termina la espera'

  WAITFOR (RECEIVE receive_statement) [, TIMEOUT timeout]

Tratamiento de excepciones

- Implementa un mecanismo de control de errores, se puede incluir un grupo de instrucciones en un bloque TRY y, si se produce un error, el control se transfiere a otro grupo de instrucciones que está incluido en un bloque CATCH.

```
BEGIN TRY
    { sql_statement | statement_block }
END TRY
BEGIN CATCH
    { sql_statement | statement_block }
END CATCH
```

- Funciones
  - @@ERROR: devuelve el número de error de la última instrucción ejecutada.
  - ERROR_NUMBER() devuelve el número del error.
  - ERROR_SEVERITY() devuelve la gravedad.
  - ERROR_STATE() devuelve el número de estado del error.
  - ERROR_PROCEDURE() devuelve el nombre del procedimiento almacenado o desencadenador donde se produjo el error.
  - ERROR_LINE() devuelve el número de línea de la rutina que provocó el error.
  - ERROR_MESSAGE() devuelve el texto completo del mensaje de error. Este texto incluye los valores suministrados para los parámetros reemplazables, como longitudes, nombres de objetos u horas.

Tratamiento de excepciones

- RAISERROR
  - Genera un mensaje de error e inicia el procesamiento de errores de la sesión.
  - Puede hacer referencia a un mensaje definido por el usuario almacenado en la vista de catálogo sys.messages o puede generar un mensaje dinámicamente.
  - El mensaje se devuelve como un mensaje de error de servidor a la aplicación que realiza la llamada o a un bloque CATCH asociado de una construcción TRY…CATCH.

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

- THROW
  - Genera una excepción y transfiere la ejecución a un bloque CATCH de una construcción TRY…CATCH

```
THROW [ { error_number | @local_variable },
      { message | @local_variable },
      { state | @local_variable } ]
```


Cursores

- Si una consulta produce una única fila se pueden asignar directamente los valores de las columnas a variables.
- Si produce un conjunto completo de resultados hay que procesarlo de fila en fila mediante un cursor.
- Declarar el cursor:
  - SQL 92 Syntax

```
DECLARE cursor_name [ INSENSITIVE ] [ SCROLL ] CURSOR FOR
select_statement [ FOR { READ ONLY | UPDATE [ OF column_name [ ,...n
] ] } ][;]
```

  - Transact-SQL Extended Syntax

```
DECLARE cursor_name CURSOR[ LOCAL | GLOBAL ][ FORWARD_ONLY |
SCROLL ][ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ][ READ_ONLY
| SCROLL_LOCKS | OPTIMISTIC ][ TYPE_WARNING ]FOR
select_statement[ FOR UPDATE [ OF column_name [ ,...n ] ] ][;]
```

Cursores

- Abrir el cursor, se ejecuta la consulta:
  OPEN { { [GLOBAL] cursor_name } | cursor_variable_name }
- La función @@CURSOR_ROWS devuelve el número de filas certificadas que se encuentran en el último cursor abierto en la conexión

- Posicionar el cursor y traspasar los valores de las columnas de la fila actual a las variables:
  FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE { n | @nvar } | RELATIVE { n | @nvar }]
  [FROM] { { [GLOBAL] cursor_name } | @cursor_variable_name }
  [INTO @variable_name [,...n]]
- La función @@FETCH_STATUS devuelve el estado de la última instrucción FETCH de cursor emitida para cualquier cursor abierto en ese momento por la conexión.

Cursores

- Cerrar el cursor y liberar la memoria:
  CLOSE { { [GLOBAL] cursor_name } | cursor_variable_name }
  DEALLOCATE { { [GLOBAL] cursor_name } | @cursor_variable_name }
- Algoritmo tipo:

```
DECLARE ElCursor CURSOR FOR SELECT …
OPEN ElCursor;
FETCH NEXT FROM ElCursor INTO …;
WHILE @@FETCH_STATUS = 0
BEGIN
    … Tratamiento de los datos recuperados …
    FETCH NEXT FROM ElCursor INTO …;
END;
CLOSE ElCursor;
DEALLOCATE ElCursor;
```

OPTIMIZACIÓN DE SENTENCIAS

Introducción

- El SQL, al ser un lenguaje de alto nivel, requiere ser compilado antes de poder ejecutarse.
- El resultado de la compilación se denomina plan de ejecución y se almacenan en una parte de la memoria de SQL Server denominada caché del plan.
- Fases de la ejecución:
  1. Búsqueda en la cache
  2. Validación sintáctica
  3. Conversión y homogeneización (parametrización)
  4. Optimización
  5. Almacenamiento en cache
  6. Establecer contexto
  7. Ejecutar plan

Pasos básicos para procesar una instrucción

1. El analizador examina la instrucción y la divide en unidades lógicas como palabras clave, expresiones, operadores e identificadores.
2. Se genera un árbol de la consulta, a veces denominado árbol de secuencia, que describe los pasos lógicos que se requieren para transformar los datos de origen en el formato que necesita el conjunto de resultados.
3. El optimizador de consultas analiza diferentes formas de acceso a las tablas de origen y a continuación:
   - Selecciona la serie de pasos que devuelve los resultados de la forma más rápida utilizando el menor número posible de recursos.
   - El árbol de la consulta se actualiza para registrar esta serie exacta de pasos.
   - La versión final y optimizada del árbol de la consulta se denomina plan de ejecución.
4. El motor relacional comienza a ejecutar el plan de ejecución. A medida que se procesan los pasos que necesitan datos de las tablas base, el motor relacional solicita al motor de almacenamiento que pase los datos de los conjuntos de filas solicitados desde el motor relacional.
5. El motor relacional procesa los datos que devuelve el motor de almacenamiento en el formato definido para el conjunto de resultados y devuelve el conjunto de resultados al cliente.

Optimizador de consultas

- El optimizador de consultas de SQL Server es un optimizador basado en el costo.
- Cada plan de ejecución posible tiene asociado un costo en términos de la cantidad de recursos del equipo que se utilizan.
- El optimizador de consultas debe analizar los planes posibles y elegir el de menor costo estimado.
- Algunas instrucciones complejas tienen miles de planes de ejecución posibles. En estos casos, el optimizador de consultas no analiza todas las combinaciones posibles. En lugar de esto, utiliza algoritmos complejos para encontrar un plan de ejecución que tenga un costo razonablemente cercano al mínimo posible.
- El optimizador elige, además de por el costo de recursos mínimo, el plan que devuelve resultados al usuario a la mayor brevedad posible con un costo razonable de recursos.

Optimizador de consultas

- El optimizador utilizará un plan de ejecución en paralelo para devolver resultados si esto no afecta negativamente a la carga del servidor.
- El optimizador confía en las estadísticas de distribución y en los índices disponibles cuando calcula los costos de recursos de métodos diferentes para extraer información de una tabla o un índice.
- Las estadísticas de distribución se mantienen para las columnas y los índices, indicando la posibilidad de seleccionar los valores de un índice o de una columna determinados.
- Si las estadísticas no están actualizadas, puede que el optimizador de consultas no realice la mejor elección para el estado actual de la tabla.
- El optimizador de consultas permite que el servidor se ajuste dinámicamente a las condiciones cambiantes de la base de datos

Causas de ejecución lenta

- Comunicaciones de red lentas.
- Memoria inadecuada en el equipo servidor o falta de memoria disponible para SQL Server.
- Bloqueos e interbloqueos.
- Falta de estadísticas útiles
- Falta de índices útiles.
- Falta de vistas indizadas útiles.
- Falta de particiones útiles.
- Inadecuada redacción de la consulta.
- Inadecuada estructura lógica o física de la base de datos.

Planes de ejecución gráficos

- Los planes de ejecución gráficos deben leerse de arriba a abajo y de derecha a izquierda.
- Cada nodo de la estructura en árbol se representa como un icono que especifica el operador lógico y físico utilizado para ejecutar esa parte de la consulta o instrucción.
- Cada nodo está relacionado con un nodo principal.
- Los nodos secundarios que tienen el mismo nodo principal se dibujan en la misma columna. Sin embargo, no todos los nodos de la misma columna tienen necesariamente el mismo nodo principal.
- Cada nodo se conecta a su nodo principal mediante reglas con puntas de flecha.
- Los operadores se muestran como símbolos relacionados con un nodo principal específico.
- El ancho de la flecha es proporcional al número de filas.
- Se utiliza el número real de filas cuando está disponible. Si no, se utiliza el número estimado de filas.
- Cuando la consulta contiene varias instrucciones, se dibujan varios planes de ejecución de la consulta.
- Las partes de las estructuras en árbol se determinan por el tipo de instrucción ejecutada.
- Si se coloca el puntero sobre uno de los iconos o una de las flechas muestra la información ampliada sobre el mismo
- Se pueden incluir y comparar las estadísticas de cliente, para obtener resultados fiables es conveniente forzar un punto de comprobación y borrar el buffer cache para que vuelva a cargar las páginas de disco:

```
CHECKPOINT
DBCC DROPCLEANBUFFERS
```


Operadores

- Los operadores describen cómo SQL Server ejecuta una consulta o una instrucción DML (Lenguaje de manipulación de datos).
- El optimizador de consultas utiliza operadores para generar un plan de consulta con el fin de crear el resultado especificado en la consulta o para realizar la operación especificada en la instrucción DML.
- El plan de consulta es un árbol que consta de operadores físicos.
- Los operadores se clasifican como:
  - Operadores lógicos: describen una operación de procesamiento de consulta relacional a nivel conceptual.
  - Operadores físicos: implementan realmente la operación definida por un operador lógico utilizando un método o algoritmo concreto. Los operadores físicos se inicializan, recopilan datos y se cierran.
- Algunos operadores son a la vez lógicos y físicos.

Recuperación de páginas

- **Scan**: recupera todas las páginas aunque solo se devuelven las filas que cumplan el predicado.


Table Scan


Clustered Index Scan


Index Scan

- **Seek**: usa la capacidad de búsqueda de los índices para recuperar solo las páginas que contengan las filas que cumplan el predicado.


Clustered Index Seek


Index Seek

- **Spool:** guarda un resultado de consulta intermedio en la base de datos tempdb**.**


Spool


Table Spool


Index Spool

Búsquedas

- **Bookmark Lookup**: usa un marcador (identificador de fila o clave de agrupación en clústeres) para buscar la fila correspondiente en la tabla o índice clúster.

  Sustituido por los operadores Clustered Index Seek y RID Lookup.
- **Key Lookup**: es una búsqueda de marcadores en un tabla con un índice clúster.
- **RID Lookup**: es una búsqueda de marcadores en un montón que usa un identificador de fila suministrado (RID).

Combinaciones

- **Nested Loops**: realiza las operaciones lógicas de combinación interna, combinación externa izquierda, semicombinación izquierda y anti semicombinación. Las combinaciones de bucles anidados buscan en la tabla interna cada fila de la tabla externa, normalmente mediante un índice. El procesador de consultas decide, en función de los costos anticipados, si debe ordenar o no la entrada externa para mejorar la ubicación de las búsquedas en el índice de la entrada interna. Se devuelven las filas que cumplen el predicado.
- **Merge Join**: requiere dos entradas ordenadas por sus respectivas columnas, que se pueden realizar mediante la inserción de operaciones de ordenación explícitas en el plan de consulta. Es especialmente eficaz si no se necesita un orden explícito, por ejemplo, si hay un índice idóneo de árbol b en la base de datos o si el orden se puede aprovechar en varias operaciones, como en una combinación de mezcla y una agrupación con acumulación.

Combinaciones

- **Hash Match**: genera una tabla hash y calcula un valor hash para cada fila de su entrada de compilación. A continuación, por cada fila de sondeo (como corresponda), calcula un valor hash (con la misma función hash) y busca las coincidencias en la tabla hash. El comportamiento depende de la operación lógica que se esté realizando:
  - Para cualquier combinación, utilice la primera entrada (superior) para generar la tabla hash y la segunda entrada (inferior) para sondear la tabla hash. Obtendrá como resultado las coincidencias (o las no coincidencias) que indique el tipo de combinación. Si varias combinaciones utilizan la misma columna de combinación, estas operaciones se agrupan en un equipo hash.
  - Para los operadores Distinct o Aggregate, utilice la entrada para generar la tabla hash (para ello, quite los duplicados y calcule las expresiones de agregado). Cuando se haya generado la tabla hash, recorra la tabla y presente todas las entradas.
  - En el caso del operador Union, utilice la primera entrada para generar la tabla hash (para ello, quite los duplicados). Use la segunda entrada (que no debe tener duplicados) para sondear la tabla hash, devolver todas las filas que no tengan coincidencias y, a continuación, recorrer la tabla hash para devolver todas las entradas.

Tipos de combinación

- **Cross Join**: combina cada fila de la primera entrada (superior) con cada fila de la segunda entrada (inferior).
- **Inner Join**: devuelve todas las filas que cumplen la combinación de la primera entrada (superior) con la segunda entrada (inferior).
- **Full Outer Join**: devuelve cada fila que cumple el predicado de combinación de la primera entrada (superior) combinada con cada fila de la segunda entrada (inferior). También devuelve filas de La primera entrada que no tenga coincidencias en la segunda entrada y de la segunda entrada que no tenga coincidencias en la primera entrada. La entrada que no contiene valores coincidentes se devuelve como un valor nulo.
- **Left Outer Join**: devuelve cada fila que cumple la combinación de la primera entrada (superior) con la segunda entrada (inferior). También devuelve las filas de la primera entrada que no tienen filas coincidentes en la segunda entrada. Las filas que no coinciden en la segunda entrada se devuelven como valores NULL. Si no hay ningún predicado de combinación, cada una de las filas es coincidente.
- **Left Semi Join**: devuelve todas las filas de la primera entrada (superior) cuando hay una fila coincidente en la segunda entrada (inferior). Si no hay ningún predicado de combinación, cada fila es una fila coincidente.
- **Left Anti Semi Join**: devuelve todas las filas de la primera entrada (superior) cuando no hay ninguna fila coincidente en la segunda entrada (inferior). Si no hay ningún predicado de combinación, cada fila es una fila coincidente.
- **Right Outer Join, Right Semi Join, Right Anti Semi Join**: Similares a los Left empezando con la segunda entrada.
- **Union**: recorre varias entradas, obtiene cada fila recorrida y quita los duplicados.

Filtrados

Filter: recorre la entrada y solo devuelve las filas que cumplen la expresión del filtro (predicado).

Bitmap: usado para implementar filtros de mapas de bits en planes de consulta paralelos. Los filtros de mapas de bits agiliza la ejecución de la consulta al eliminar las filas con valores de clave que no pueden generar ningún registro de combinación antes de pasar las filas a través de otro operador, por ejemplo, el operador Parallelism. Un filtro de mapas de bits usa una representación compacta de un conjunto de valores de una tabla en una parte del árbol de operadores para filtrar filas de una segunda tabla en otra parte del árbol. Al quitar las filas innecesarias en las primeras fases de la consulta, los operadores subsiguientes tienen que trabajar en menos filas y el rendimiento total de la consulta mejora. El optimizador determina cuándo un mapa de bits es suficientemente selectivo para resultar útil y en qué operadores se aplica el filtro. Bitmap es un operador físico.

Otros operadores

- **Sort**: ordena todas las filas entrantes. Las columnas llevan como prefijo el valor ASC, si el orden de las columnas es ascendente, o el valor DESC, si es descendente.
- **Stream Aggregate**: agrupa las filas por una o varias columnas y, a continuación, calcula una o varias expresiones agregadas devueltas por la consulta. El resultado de este operador puedes ser utilizado por operadores posteriores de la consulta, devuelto al cliente, o ambas cosas. Requiere una entrada ordenada por las columnas dentro de sus grupos.
- **Compute Scalar**: evalúa una expresión para generar un valor escalar calculado que se puede devolver al usuario, hacer referencia a él en cualquier otra parte de la consulta o ambas cosas a la vez.
- **UDX**: implementan una de las múltiples operaciones XQuery y XPath. Los operadores extendidos (UDX): FOR XML, XML SERIALIZER, XML FRAGMENT SERIALIZER, XQUERY STRING, XQUERY LIST DECOMPOSER, XQUERY DATA, XQUERY CONTAINS, UPDATE XML NODE.
- **Top**: recorre la entrada y solo devuelve el primer número o porcentaje especificado de filas, basándose en un criterio de ordenación si es posible.

Otros operadores

- **Assert**: comprueba una condición y si la expresión da como resultado un valor distinto a NULL se generará el error apropiado (validaciones).
- **Concatenation**: explora varias entradas y devuelve cada fila explorada, se utiliza para implementar la construcción UNION ALL, tiene dos o más entradas y una salida. Concatenation copia filas del primer flujo de entrada en el flujo de salida y, a continuación, repite esta operación con cada flujo de entrada adicional.
- **Segment**: divide el conjunto de entrada en segmentos basados en el valor de una o varias columnas.
- **Split**: se utiliza para optimizar el procesamiento de actualizaciones. Divide cada operación de actualización en una operación de eliminación y una operación de inserción.
- **Table-valued Function**: evalúa una función con valores de tabla y almacena las filas resultantes en la base de datos tempdb. Cuando los iteradores principales solicitan las filas, la función con valores de tabla devuelve las filas desde tempdb.

Detalles del nodo

- **Operación física**: Operador físico utilizado. Los operadores físicos presentados en color rojo indican que el optimizador de consultas ha emitido una advertencia, por ejemplo, la falta de estadísticas de columna o de predicados de combinación. Esto puede causar que el optimizador de consultas elija un plan de consulta menos eficaz que el esperado.
- **Operación lógica**: Operador lógico que coincide con el operador físico, como el operador Inner Join.
- **Tamaño de fila estimado**: Tamaño estimado de la fila creada por el operador (bytes).
- **Costo de E/S estimado**: Costo estimado de toda la actividad de E/S para la operación. Este valor *debe ser lo más pequeño posible*.
- **Costo de CPU estimado**: Costo estimado de toda la actividad de la CPU para la operación.
- **Costo de operador estimado**: Costo del optimizador de consultas para ejecutar esta operación. El costo de esta operación como porcentaje del costo total de la consulta se muestra entre paréntesis. Debido a que el motor de consultas selecciona la operación más eficaz para realizar la consulta o ejecutar la instrucción, este valor *debe ser el menor posible*.
- **Costo de subárbol estimado**: Costo total del optimizador de consultas para ejecutar esta operación y todas las operaciones del mismo subárbol anteriores a ésta.
- **Número de filas** o **Número de filas estimado**: Número de filas que produce el operador.

- En función del nodo: Objeto de salida, Lista de salida (columnas), Número de ejecuciones, Número de filas, Tamaño, Ordenación
- En algunos casos se mostraran advertencias o sugerencias sobre el nodo.

Sugerencias de consulta

- Las sugerencias de consulta (hint) invalidan el comportamiento predeterminado del optimizador de consultas mientras dura la instrucción de consulta.
- Son sugerencias por que el optimizador no está obligado a seguirlas.
- Se pueden usar para especificar un método de bloqueo en las tablas afectadas, uno o varios índices, una operación de procesamiento de la consulta como un recorrido de tabla o una búsqueda de índice, u otras opciones.
- Las sugerencias de consulta se especifican en la cláusula OPTION y se aplican a toda la consulta.
- Como el optimizador de consultas de SQL Server suele seleccionar el mejor plan de ejecución para las consultas, se recomienda que solo se hagan como último recurso.

Sugerencias de consulta

- { HASH | ORDER } GROUP
- { CONCAT | HASH | MERGE } UNION
- { LOOP | MERGE | HASH } JOIN
- FAST number_rows
- FORCE ORDER
- MAXDOP number_of_processors
- OPTIMIZE FOR (@variable_name { UNKNOWN | = literal_constant } [, ...n])
- OPTIMIZE FOR UNKNOWN
- PARAMETERIZATION { SIMPLE | FORCED }
- RECOMPILE
- ROBUST PLAN
- KEEP PLAN
- KEEPFIXED PLAN
- EXPAND VIEWS
- MAXRECURSION number
- USE PLAN N'xml_plan'
- TABLE HINT (exposed_object_name [, <table_hint> [[,]...n]])

Sugerencias de tabla

- Las sugerencias de tabla invalidan el comportamiento predeterminado del optimizador de consultas mientras dura la instrucción de lenguaje de manipulación de datos (DML), especificando un método de bloqueo, uno o varios índices, una operación de procesamiento de la consulta como, por ejemplo, un examen de tabla o una búsqueda de índice, u otras opciones.
- Las sugerencias de tabla se especifican en la cláusula FROM … WITH de la instrucción DML y solo afectan a la tabla o a la vista a la que se hace referencia en esa cláusula.

SELECT

- INDEX (index_value [,...n]) | INDEX = (index_value)
- FASTFIRSTROW
- FORCESEEK [(index_value (index_column_name [,...]))]
- FORCESCAN
- HOLDLOCK
- NOLOCK
- NOWAIT
- PAGLOCK
- READCOMMITTED
- READCOMMITTEDLOCK
- READPAST
- READUNCOMMITTED
- REPEATABLEREAD
- ROWLOCK
- SERIALIZABLE
- TABLOCK
- TABLOCKX
- UPDLOCK
- XLOCK

INSERT, UPDATE, DELETE

- KEEPIDENTITY
- KEEPDEFAULTS
- FASTFIRSTROW
- HOLDLOCK
- IGNORE_CONSTRAINTS
- IGNORE_TRIGGERS
- NOLOCK
- NOWAIT
- PAGLOCK
- READCOMMITTED
- READCOMMITTEDLOCK
- READPAST
- REPEATABLEREAD
- ROWLOCK
- SERIALIZABLE
- TABLOCK
- TABLOCKX
- UPDLOCK
- XLOCK

Guías de plan

- Las guías de plan se pueden utilizar para optimizar el rendimiento de las consultas cuando no pueda o no desee cambiar directamente el texto de dichas consultas.
- Las guías de plan pueden ser de gran utilidad cuando el rendimiento de un pequeño subconjunto de consultas de una aplicación de base de datos implementada por otro proveedor no es el esperado.
- Las guías de plan influyen en la optimización de las consultas adjuntando sugerencias de consulta o un plan de consulta fijo para ellas.
- En la guía de plan, se especifica la instrucción de Transact-SQL que se desea optimizar y además una cláusula OPTION que incluye las sugerencias de consulta que se desean utilizar o un plan de consulta específico con el que desea optimizar la consulta.
- Cuando la consulta se ejecuta, el SQL Server hace coincidir la instrucción de Transact-SQL con la guía de plan y además asocia en tiempo de ejecución la cláusula OPTION a la consulta o utiliza el plan de consultas especificado.

Tipos de guías de plan

- OBJECT: compara las consultas que se ejecutan en el contexto de procedimientos almacenados de Transact-SQL, funciones escalares definidas por el usuario, funciones definidas por el usuario con valores de tabla de múltiples instrucciones y desencadenadores DML.
- SQL: compara las consultas que se ejecutan en el contexto de instrucciones independientes de Transact-SQL y lotes que no forman parte de un objeto de base de datos.
  - Las guías de plan basadas en SQL también se pueden usar para comparar consultas que se parametrizan en un formulario especificado.
  - Las guías de plan de SQL se aplican a las instrucciones y lotes independientes de Transact-SQL.
  - Con frecuencia, las aplicaciones envían esas instrucciones utilizando el procedimiento almacenado del sistema sp_executesql.
- TEMPLATE: compara consultas independientes que se parametrizan en un formulario especificado.
  - Estas guías de plan se usan para reemplazar la opción PARAMETERIZATION actual de una base de datos para una clase de consultas por medio de SET.

Ejemplos de guías de plan

```
sp_create_plan_guide @name = N'Guide1',
      @type = N'OBJECT',
      @module_or_batch = N'Sales.GetSalesOrderByCountry',
      @stmt = N'SELECT * FROM Sales.SalesOrderHeader AS h,
            Sales.Customer AS c,
            Sales.SalesTerritory AS t
            WHERE h.CustomerID = c.CustomerID
              AND c.TerritoryID = t.TerritoryID
              AND CountryRegionCode = @Country_region',
      @params = NULL,
      @hints = N'OPTION (OPTIMIZE FOR (@Country_region = N''US''))';

sp_create_plan_guide @name = N'Guide2',
      @type = N'SQL',
      @module_or_batch = NULL,
      @stmt = N'SELECT TOP 1 * FROM Sales.SalesOrderHeader ORDER BY OrderDate DESC',
      @params = NULL,
      @hints = N'OPTION (MAXDOP 1)';
```


Limitaciones para las guías

- El número total de guías de plan que se pueden crear solo está limitado por los recursos de los que disponga el sistema.
- Las guías de plan deberían limitarse a aquellas consultas de gran importancia cuyo rendimiento se desea mejorar o estabilizar.
- No se deben usar las guías de plan para influenciar la mayor parte de la carga de la consulta de una aplicación implementada.
- Se debe volver a evaluar y probar las definiciones de guías de plan al actualizar la aplicación a una nueva versión de SQL Server.
  - Los requisitos de optimización del rendimiento y el comportamiento de la coincidencia de las guías de plan pueden cambiar.
  - Aunque una guía de plan no válida no hará que una consulta provoque un error, el plan se compilara sin utilizar la guía de plan y posiblemente no sea la mejor opción
- No están disponibles en las ediciones Express.

Planes de ejecución forzados

- xml_plan es un literal de cadena derivado del plan de consulta con formato XML que se genera para una consulta.
- Un xml_plan se puede redactar manualmente u obtenerse mediante “Guardar plan de ejecución”, consultando la columna query_plan de la función sys.dm_exec_query_plan o eventos de traza: Showplan XML, Showplan XML Statistics Profile y Showplan XML For Query Compile.
- No todos los elementos de los planes de consulta con formato XML se fuerzan con la sugerencia USE PLAN. Los elementos que calculan expresiones escalares se omiten, así como algunas expresiones relacionales.
- El plan de consulta con formato XML especificado en xml_plan debe validarse con el esquema XSD Showplanxml.xsd.
- La sugerencia de consulta USE PLAN se puede especificar como una sugerencia de consulta en una instrucción SQL independiente, o especificarse en el parámetro @hints de una guía de plan.
- Se debe indicar siempre xml_plan como literal Unicode especificando el prefijo N, como en N'xml_plan', con lo que se garantiza que no se pierda ninguno de los caracteres del plan específico del estándar Unicode cuando SQL Server Database Engine (Motor de base de datos de SQL Server) interprete la cadena.

Limitaciones para Planes forzados

- Los cambios que se produzcan en la base de datos, como la eliminación de índices, pueden invalidar un plan de consulta forzado.
- Un plan de consulta puede quedarse obsoleto aunque no se haga directamente referencia en el plan a un objeto eliminado.
- Instalar un Service Pack o una nueva versión de SQL Server puede impedir que fuerce un plan generado en una versión anterior, por lo que deberán probarse todos los planes forzados siempre que se actualice el servidor.
- El uso de planes forzados en una consulta reemplaza todas las sugerencias de combinación y de índice utilizadas en la misma consulta.
- Los únicos planes de consulta que pueden forzarse son los que pueden encontrarse mediante la estrategia de búsqueda típica del optimizador de consultas.