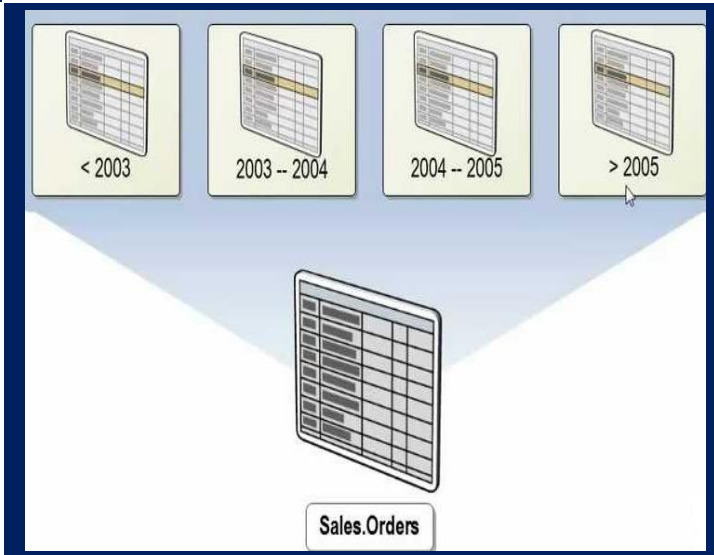
Bases de Datos Avanzadas

Tema 2

Esquemas -Tablas - Vistas



Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Objetivos y resultados de aprendizaje

- Las tablas son los almacenes de datos de los SGBD. Para ser manipulados se deben recuperar de disco y cargados en memoria.

- Tras tratar las decisiones sobre la mejor ubicación de los archivos de datos, se ha de trabajar como particionar los datos de ciertas tabla para obtener un mejor rendimiento.

- El trabajo con Vistas y Esquema ayuda en la seguridad y eficiencia de los accesos.

- Los objetivos específicos consisten en:
  - ✓ Creación: Funciones de Partición, Esquemas de Partición y Tablas Particionadas
  - ✓ Uso de esquema
  - ✓ Generación de Vistas

Evaluación del tema

- Los resultados de aprendizaje correspondiente a este tema se evaluarán con los siguientes tipos de pruebas:

  - ✓ Prueba escrita de carácter teórico
  - ✓ Prueba práctica de laboratorio
  - ✓ Participación en clase

Bibliografía

- Para obtener más información puedes consultar:

  - ✓ Libros en pantalla de SQL Server.

  - ✓ Bibliografía incluida en la guía didáctica

- Lecturas Obligadas

  - ✓ https://learn.microsoft.com/es-es/sql/relational-databases/security/authentication-access/create-a-database-schema?view=sql-server-ver16

  - ✓ https://learn.microsoft.com/es-es/sql/relational-databases/security/authentication-access/principals-database-engine?view=sql-server-ver16

Índice de contenidos

- ✓ Esquema de Bases de Datos
- ✓ Modelos de Tablas
- ✓ Tablas Particionadas
- ✓ Vistas
- ✓ Práctica (1ª Entrega)

Esquema de BBDD (Creación)

- Los esquemas SQL tienen un papel fundamental al permitir que los datos agrupados y no agrupados de la base de datos se organicen en grupos, es decir, un esquema es un contenedor que permite agrupar los objetos y clasificarlos por nombre.
- Un esquema es un conjunto de objetos creado dentro de la misma base de datos para organizar la información mediante tablas, procedimientos, índices, funciones, etc. Esta clasificación se conoce como esquema dentro de la base de datos, y recibe de manera predeterminada el nombre de database owner o dbo.
- El esquema es una entidad en el ámbito de la base de datos, de modo que puedes tener el mismo esquema en diferentes bases de datos de una instancia de SQL Server. De forma predeterminada, SQL Server utiliza el esquema [dbo] para todos los objetos de una base de datos.
- Consulta:

```
SELECT
s.name AS SchemaName,
t.name AS TableName
FROM sys.tables t
  INNER JOIN sys.schemas s
  ON t.schema_id = s.schema_id
WHERE t.name ='TableA'
```

Esquema de BBDD (Creación)

- Que un usuario tenga acceso a una base de datos no quiere decir que tenga acceso a todo su contenido, ni a cada uno de los objetos que la componen. Para que esto ocurra tendremos que irle concediendo o denegando permisos sobre cada uno de los objetos que la componen.

- Colección de objetos de la BBDD cuyo propietario es un único principal y forma un único espacio de nombres (conjunto de objetos que no pueden tener nombres duplicados)

  servidor.basededatos.esquema.objeto

- Una BBDD puede contener múltiples esquemas.
- Cada esquema tiene un propietario (principal): usuario o rol.
- Cada usuario tiene un default schema para resolución de nombres

Esquema de BBDD (Definición)

- CREATE SHEMA Nombre_Esquema

  CREATE SCHEMA EquipoDucati

- Creando una tabla bajo el esquema EquipoDucati:

  CREATE TABLE EquipoDucati.TB_Piloto
  (dorsal int primary key,
  nombre varchar(25),
  nacionalida varchar(25)
)

- Crear usuario propietario del esquema

  CREATE USER JefeEquipo FOR LOGIN JF_Ducati
  WITH DEFAULT_SCHEMA = EquipoDucati;

Esquema de BBDD (Definición)

- Crear usuario y asignarle un esquema

```
CREATE USER JefeEquipo FOR LOGIN JF_Ducati
WITH DEFAULT_SCHEMA = EquipoDucati;
```

- Asignarle un esquema a un usuario existente

```
GRANT SELECT
 ON SCHEMA :: RecursosHumanos
TO JefeManu WITH GRANT OPTION
GO
```

Modelos de Tablas

- Tablas en SQL Server

  - ✓ Aunque depende de las versiones, SQL Server puede tener hasta dos mil millones de tablas por cada base de datos y 1.024columnas por tabla.

  - ✓ El número de filas y el tamaño total de la tabla están limitados solamente por el espacio de almacenamiento disponible. El número máximo de bytes por fila es 8.060.

  - ✓ Esta restricción se amplía en el caso de las tablas con columnas varchar, nvarchar, varbinary o sql_variant, las cuales hacen que el ancho total definido para la tabla sea superior a 8.060 bytes (*) .

(*) Datos de desbordamiento de filas superiores a 8 KB

Modelos de Tablas

- ✓ Cada tabla puede contener un máximo de 249 índices no agrupados y 1 índice agrupado. Éstos incluyen los índices generados para admitir las restricciones PRIMARY KEY y UNIQUE definidas para la tabla.

- ✓ Normalmente, el espacio se asigna a las tablas e índices en incrementos de una extensión a la vez.

- ✓ Cuando se crea la tabla o índice, se le asignan páginas de extensiones mixtas hasta que tiene suficientes páginas para llenar una extensión uniforme. Una vez que haya suficientes páginas para llenar una extensión uniforme, se asigna otra extensión cada vez que se llenan las extensiones asignadas actualmente.

- ✓ Para obtener un informe acerca de la cantidad de espacio asignado y utilizado por una tabla, ejecute sp_spaceused.

Modelos de Tablas

- Sintaxis

CREATE TABLE
[database_name . [schema_name] . | schema_name .] table_name

({ <column_definition> | <computed_column_definition> }

[<table_constraint>] [,...n])

[ON { partition_scheme_name (partition_column_name) | filegroup | "default" }]

[{ TEXTIMAGE_ON { filegroup | "default" }]

Tabla Particionada (Definición)

- Tablas particionadas

  - ✓ Permite separar las particiones en diferentes Tablespaces / Grupo de Ficheros, repartiendo la E/S. Cada partición además puede tener sus propias propiedades de almacenamiento.

  - ✓ Mejorar los tiempos de respuesta de las consultas, así como el mantenimiento de las tablas, y que puede ser especialmente útil en tablas muy grandes.

  - ✓ Las tablas particionadas aparecen en el sistema como una única tabla.

Tabla Particionada (Definición)

- ✓ Se dispone de más opciones a la hora de realizar labores de mantenimiento, pudiendo actuar a nivel de partición.
- ✓ Se realiza utilizando una clave de particionado, que determina en qué partición van a residir los datos que se insertan.
- ✓ La definición de las particiones se indica en la sentencia de creación de las tablas, con la sintaxis oportuna para cada uno de los tipos.

Tabla Particionada (Función de Partición)

- ✓ Crea una función en la base de datos actual que asigna las filas de una tabla o un índice a particiones según los valores de una columna especificada.
- ✓ El uso de CREATE PARTITION FUNCTION constituye el primer paso para la creación de una tabla o un índice con particiones.
- ✓ El ámbito de una función de partición está limitado a la base de datos en la que se crea.
- ✓ En la base de datos, las funciones de partición residen en un espacio de nombres independiente de las otras funciones.
- ✓ Las filas cuya columna de partición tenga valores NULL se colocan en la partición situada más a la izquierda, a menos que se especifique NULL como un valor de límite y se indique RIGHT. En este caso, la partición situada más a la izquierda es una partición vacía y los valores NULL se colocan en la siguiente.

Tabla Particionada (Función de Partición)

- ✓ Crea una función en la base de datos actual que asigna las filas de una tabla o un índice a particiones según los valores de una columna especificada.

```
CREATE PARTITION FUNCTION partition_function_name ( input_parameter_type )
AS RANGE [ LEFT | RIGHT ]
FOR VALUES ( [ boundary_value [ ,...n ] ] )
```

input_parameter_type

- ❖ Es el tipo de datos de la columna utilizada para la partición.
- ❖ La columna en sí, conocida como columna de partición, se especifica en la instrucción CREATE TABLE

Tabla Particionada (Función de Partición)

CREATE PARTITION FUNCTION partition_function_name (input_parameter_type)
AS RANGE [LEFT | RIGHT]
FOR VALUES ([boundary_value [,...n]])

boundary_value

- Especifica los valores de límite de cada partición de una tabla con particiones que utilice partition_function_name.
- Si el parámetro boundary_value está vacío, la función de partición asigna la tabla completa a una única partición.
- Sólo es posible utilizar una columna de partición, especificada en una instrucción CREATE TABLE.
- boundary_value debe coincidir con el tipo de datos proporcionado en input_parameter_type o ser implícitamente convertible en éste.

Tabla Particionada (Función de Partición)

CREATE PARTITION FUNCTION partition_function_name (input_parameter_type)
AS RANGE [LEFT | RIGHT]
FOR VALUES ([boundary_value [,...n]])

...n

- Especifica el número de valores proporcionado por boundary_value, que no debe superar los 999.
- El número de particiones creadas equivale a n + 1.
- Los valores no tienen que presentarse en orden. Si los valores no están en orden, el SQL Server 2005 Database Engine los ordena, crea la función y muestra una advertencia de que los valores proporcionados no estaban en orden.
- Si n incluye valores duplicados, el motor de base de datos devuelve un error.

Tabla Particionada (Función de Partición)

CREATE PARTITION FUNCTION partition_function_name (input_parameter_type)
AS RANGE [LEFT | RIGHT]
FOR VALUES ([boundary_value [,...n]])

LEFT | RIGHT

- Especifica el lado de cada intervalo de valores límite, derecho o izquierdo, al que pertenece boundary_value [,...n], cuando el Database Engine (Motor de base de datos) ordena los valores del intervalo en orden ascendente de izquierda a derecha.

- Si no se especifica, el valor predeterminado es LEFT.

Tabla Particionada (Función de Partición)

- ✓ Crear una función de partición RANGE LEFT / RIGHT en una columna int
- ✓ La siguiente función de partición realizará cuatro particiones en una tabla. Las tablas muestran cómo se crearían las particiones

CREATE PARTITION FUNCTION myRangePF1 (int)
AS RANGE LEFT FOR VALUES (1, 100, 1000)

Partición	1	2	3	4
Valores	col1 <= 1	col1 > 1 AND col1 <= 100	col1 > 100 AND col1 <= 1000	col1 > 1000

CREATE PARTITION FUNCTION myRangePF2 (int)
AS RANGE RIGHT FOR VALUES (1, 100, 1000);

Partición	1	2	3	4
Valores	col1 < 1	col1 >= 1 AND col1 < 100	col1 >= 100 AND col1 < 1000	col1 >= 1000

Tabla Particionada (Esquema de Partición)

- ✓ Crea un esquema en la base de datos actual que asigna a grupos de archivos las particiones de una tabla con particiones.
- ✓ El número y el dominio de las particiones de una tabla con particiones se determinan en una función de partición.
- ✓ Antes de crear un esquema de partición es necesario crear una función de partición en una instrucción.

Tabla Particionada (Esquema de Partición)

CREATE PARTITION SCHEME partition_scheme_name
AS PARTITION partition_function_name
[ALL] TO ({ file_group_name | [PRIMARY] } [,...n])

partition_scheme_name

- Es el nombre del esquema de partición.
- Los nombres de esquema de partición deben ser únicos en la base de datos.

partition_function_name

- Es el nombre de la función de partición que utiliza el esquema de partición.
- partition_function_name debe existir en la base de datos.

Tabla Particionada (Esquema de Partición))

CREATE PARTITION SCHEME partition_scheme_name
AS PARTITION partition_function_name
[ALL] TO ({ file_group_name | [PRIMARY] } [,...n])

ALL

- Especifica que todas las particiones se asignan al grupo de archivos suministrado en file_group_name, o al grupo de archivos principal si se especifica [PRIMARY].

- Si se especifica ALL, sólo se puede especificar un valor de file_group_name

Tabla Particionada (Esquema de Partición)

file_group_name | [PRIMARY] [,...n]

- Especifica los nombres de los grupos de archivos que almacenarán las particiones especificadas en partition_function_name.
- file_group_name debe existir en la base de datos. Si se especifica [PRIMARY], la partición se almacena en el grupo de archivos principal.
- Si se especifica ALL, sólo se puede especificar un valor de file_group_name.
- A partir de la partición 1, las particiones se asignan a los grupos de archivos en el orden en que éstos aparecen en [,...n].
- Se puede especificar el mismo valor de file_group_name más de una vez.
- Si n no es suficiente para contener el número de particiones especificado en partition_function_name, CREATE PARTITION SCHEME genera un error.
- Si partition_function_name genera menos particiones que grupos de archivos, el primer grupo de archivos sin asignar se marca como NEXT USED
- El grupo de archivos NEXT USED recibirá una partición adicional si se crea una en una instrucción ALTER PARTITION FUNCTION.
- Si desea crear más grupos de archivos sin asignar para almacenar las nuevas particiones, utilice ALTER PARTITION SCHEME.

Tabla Particionada (Esquema de Partición)

- ✓ En el ejemplo siguiente se crea una función de partición para dividir una tabla en cuatro particiones. Después, se crea un esquema de partición que especifica los grupos de archivos que van a contener cada una de las cuatro particiones. Se asume que los grupos de archivos ya existen en la base de datos.

```
CREATE PARTITION FUNCTION myRangePF1 (int)
AS RANGE LEFT FOR VALUES (1, 100, 1000)

CREATE PARTITION SCHEME myRangePS1
AS PARTITION myRangePF1
TO (test1fg, test2fg, test3fg, test4fg)
```

Grupo de archivos	test1fg	test2fg	test3fg	test4fg
Partición	1	2	3	4
Valores	**col1** <= 1	**col1** > 1 AND **col1** <= 100	**col1** > 100 AND **col1** <= 1000	**col1** > 1000

Tabla Particionada (Esquema de Partición)

- ✓ Si todas las particiones se asignan al mismo grupo de archivos, utilice la palabra clave ALL. Sin embargo, si se asignan varias particiones (no todas) al mismo grupo de archivos, el nombre del grupo de archivos debe repetirse, tal como se muestra en el ejemplo siguiente.

```
CREATE PARTITION FUNCTION myRangePF2 (int)
AS RANGE LEFT FOR VALUES (1, 100, 1000)

CREATE PARTITION SCHEME myRangePS2
AS PARTITION myRangePF2
TO ( test1fg, test1fg, test1fg, test2fg )
```

Grupo de archivos	test1fg	test1fg	test1fg	test2fg
Partición	1	2	3	4
Valores	**col1** <= 1	**col1** > 1 AND **col1** <= 100	**col1** > 100 AND **col1** <= 1000	**col1** > 1000

Tabla Particionada (Esquema de Partición)

- ✓ En el ejemplo siguiente se crea la misma función de partición que en los ejemplos anteriores y se crea un esquema de partición que asigna todas las particiones al mismo grupo de archivos.

```
CREATE PARTITION FUNCTION myRangePF3 (int)
AS RANGE LEFT FOR VALUES (1, 100, 1000)

CREATE PARTITION SCHEME myRangePS3
AS PARTITION myRangePF3
ALL TO ( test1fg )
```

Tabla Particionada (Esquema de Partición)

- ✓ En el ejemplo siguiente se crea la misma función de partición que en los ejemplos anteriores y se crea un esquema de partición que especifica un número de grupos de archivos superior al número de particiones creadas por la función de partición asociada.

```
CREATE PARTITION FUNCTION myRangePF4 (int)
AS RANGE LEFT FOR VALUES (1, 100, 1000)

CREATE PARTITION SCHEME myRangePS4
AS PARTITION myRangePF4
TO (test1fg, test2fg, test3fg, test4fg, test5fg)
```

- ✓ Al ejecutar la instrucción se devuelve el mensaje indicando que test5fg es marcada como NEXT USED.

- ✓ Si la función de partición myRangePF4 se cambia para agregar una partición, el grupo de archivos test5fg recibe la partición que acaba de crearse.

Tabla Particionada (Esquema de Partición)

- ✓ Antes de crear una tabla con particiones con CREATE TABLE, debe crear una función de partición para especificar cómo se van a crear las particiones en la tabla.

CREATE PARTITION FUNCTION

- ✓ A continuación, debe crear un esquema de partición para especificar los grupos de archivos que van a contener las particiones indicadas mediante la función de partición.

CREATE PARTITION SCHEME

- ✓ La colocación de restricciones PRIMARY KEY o UNIQUE para separar grupos de archivos no se puede especificar para las tablas con particiones

Tabla Particionada (Creación)

- ✓ En el siguiente ejemplo se crea una función de partición para crear cuatro particiones en una tabla.

```
CREATE PARTITION FUNCTION myRangePF1 (int)
AS RANGE LEFT FOR VALUES (1, 100, 1000)

CREATE PARTITION SCHEME myRangePS1
AS PARTITION myRangePF1
TO (test1fg, test2fg, test3fg, test4fg)

CREATE TABLE PartitionTable (col1 int, col2 char(10))
ON myRangePS1 (col1) ;
```

- ✓ En función de los valores de la columna col1 de PartitionTable, las particiones se asignan de los siguientes modos.

Grupo de archivos	test1fg	test2fg	test3fg	test4fg
Partición	1	2	3	4
Valores	col 1 <= 1	col1 > 1 AND col1 <= 100	col1 > 100 AND col1 <= 1,000	col1 > 1000

Tabla Particionada (Comprobación)

```
CREATE PARTITION FUNCTION myRangePF1 (int)
AS RANGE LEFT FOR VALUES (1, 100, 1000)

CREATE PARTITION SCHEME myRangePS1
AS PARTITION myRangePF1
TO (test1fg, test2fg, test3fg, test4fg)

CREATE TABLE PartitionTable (col1 int, col2 char(10))
ON myRangePS1 (col1) ;
```

- ✓ Para comprobar que se está realizando correctamente la inserción en cada fichero de partición se realiza la siguiente query.

```
SELECT  partition_number, row_count FROM sys.dm_db_partition_stats
WHERE object_id = OBJECT_ID('PartitionTable')
```

- ✓ /* A continuación, después de haber insertado los datos en las tablas indicadas, mostramos las filas de datos en cada filegroup.*/

```
SELECT
p.partition_number AS PartitionNumber,
f.name AS PartitionFilegroup,
p.rows AS NumberOfRows
FROM sys.partitions p
JOIN sys.destination_data_spaces dds ON p.partition_number = dds.destination_id
JOIN sys.filegroups f ON dds.data_space_id = f.data_space_id
WHERE OBJECT_NAME(OBJECT_ID) = 'PartitionTable'
```

Vistas

- Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con un nombre.

- Sin embargo, a menos que esté indizada, una vista no existe como conjunto de valores de datos almacenados en una base de datos. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

- Una vista actúa como filtro de las tablas subyacentes a las que se hace referencia en ella. La consulta que define la vista puede provenir de una o de varias tablas, o bien de otras vistas de la base de datos actual u otras bases de datos.

- Asimismo, es posible utilizar las consultas distribuidas para definir vistas que utilicen datos de orígenes heterogéneos

Vistas

- Las vistas suelen usarse para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario.

- Las vistas pueden emplearse como mecanismos de seguridad, que permiten a los usuarios obtener acceso a los datos por medio de la vista, pero no les conceden el permiso de obtener acceso directo a las tablas base subyacentes de la vista.

https://learn.microsoft.com/es-es/sql/relational-databases/views/views?view=sql-server-ver16

https://learn.microsoft.com/es-es/sql/relational-databases/views/create-views?view=sql-server-ver16

Práctica (1ª Entrega)

- Primera entrega: *(Si se evalúa, no se califica)*

  - ✓ El alumno deberá presentar un entorno de práctica funcional (Sistema Operativo Virtualizado con un SGBD SQL Server).

  - ✓ Deberá contener una Base de Datos en la que se establezcan los parámetros adecuados de eficiencia estudiados en el tema 1. De igual modo, deberá incluir una Tabla Particionada y algunas vistas estudiadas en el presente tema 2. (cada tabal deberá contener datos)

  - ✓ Cada grupo entregará los siguientes archivos:
    1) Documento *Entrega1_Clase_Grupo.docx* que incluya una descripción de las especificaciones de la BBDD, su DER y una explicación del diseño de la tabla particionada.
    2) Archivo *Entrega1_Clase_Grupo.sql* con el script completo de la práctica.
    3) Ejemplo: Entrega1_A_7.docx y Entrega1_A_7.sql