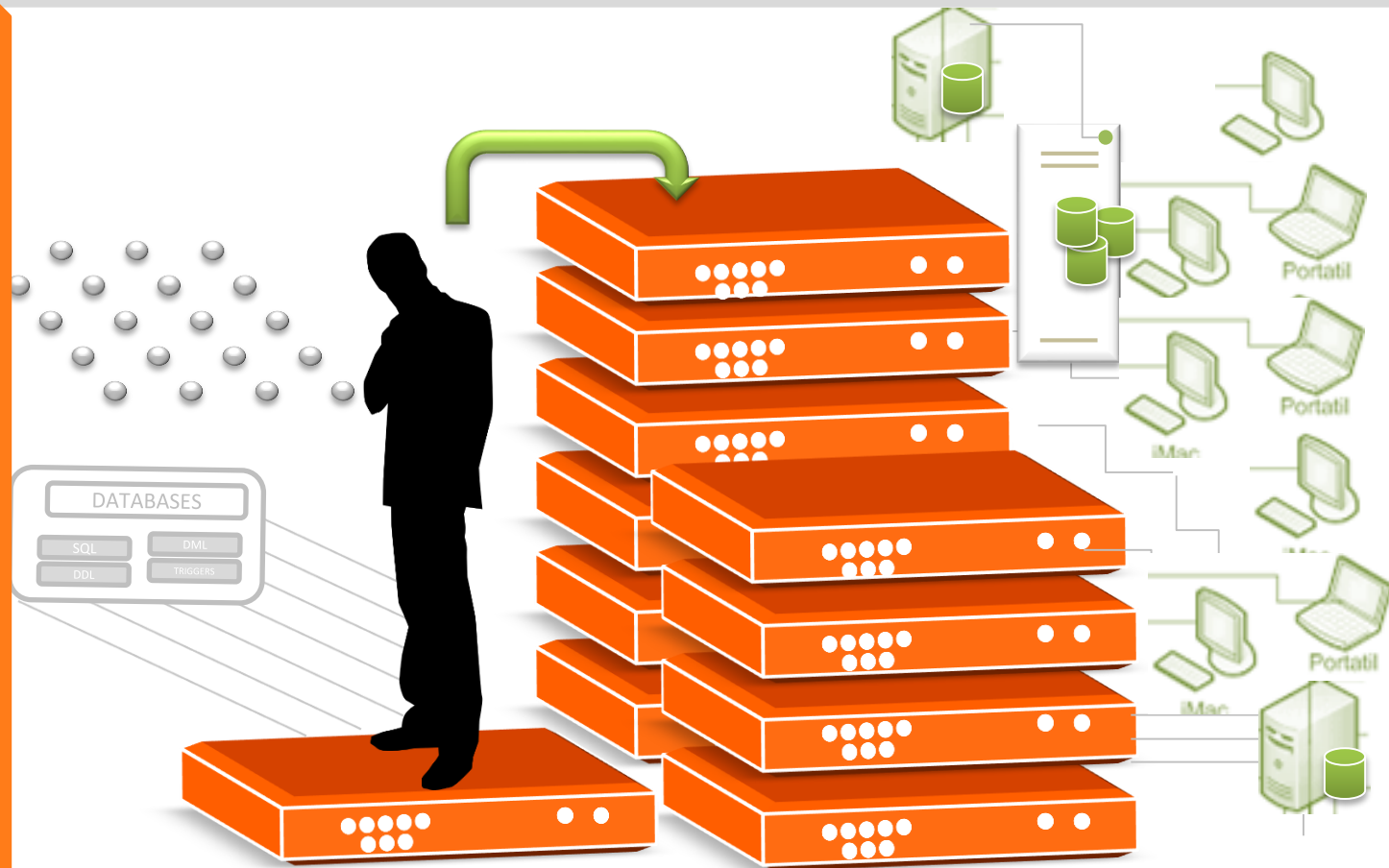


Administración de Bases de Datos



Unidad 3

Configuración y administración del espacio en disco



Por
Verónica Ramírez Jáuregui

COMPETENCIA

- Configura y administra el espacio en disco y memoria del servidor para que el funcionamiento del SGBD sea congruente con la infraestructura existente.



Unidad 3

Configuración y administración del espacio en disco



Por
Verónica Ramírez Jáuregui

TEMAS

- 3.1. Definición de espacio de almacenamiento
- 3.2. Definición y creación del espacio asignado para cada base de datos
- 3.3 Asignación de cuotas de espacio para usuarios
- 3.4. Espacios para objetos de la base de datos
- 3.5 Roles
- 4.3 Índices, reorganización y reconstrucción



Unidad 3

Configuración y administración del espacio en disco



Por
Verónica Ramírez Jáuregui

FORMA DE EVALUACIÓN

Tareas y participación (15%)

Memoria de Prácticas (35%)

Examen Práctico (50%)



¿QUE ES PARTICIONAR TABLAS DE BD's?

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Particionar es el proceso donde tablas muy grandes son divididas en múltiples partes más pequeñas. Al separar una tabla grande en tablas individuales más pequeñas, las consultas que acceden sólo a una fracción de los datos pueden correr más rápido porque hay menos datos que escanear.



El objetivo principal de particionar es ayudar en el mantenimiento de tablas grandes y reducir el tiempo de respuesta general para leer y cargar datos para operaciones SQL particulares.

¿POR QUÉ UTILIZAR LAS PARTICIONES?

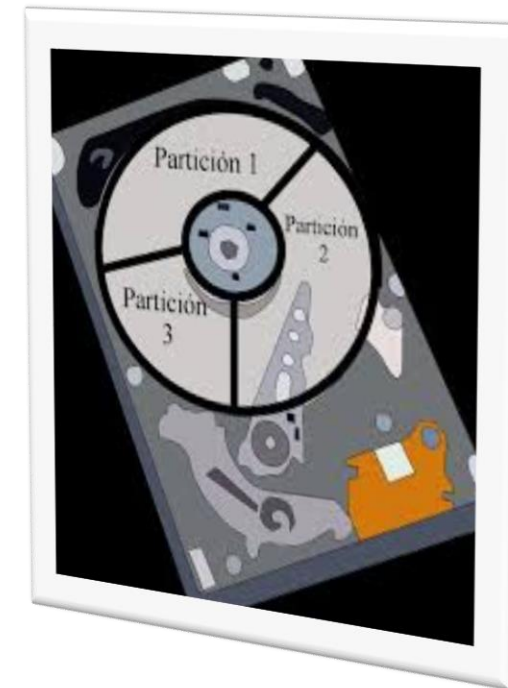
PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Considere la posibilidad de fragmentar sus tablas si tiene como objetivo mejorar al menos uno de los aspectos siguientes:

- Tiempo de respuesta de un solo usuario
- Concurrencia
- Disponibilidad
- Características de copia de seguridad y restauración
- Carga de datos



¿POR QUÉ QUERRÍAMOS PARTICIONAR UNA TABLA FÍSICAMENTE??

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Las razones son diversas, pero principalmente podemos entenderla como la necesidad de gestionar una tabla con cientos de millones de filas de forma eficiente. Un típico escenario de uso es el de una tabla con un volumen de inserciones-consultas elevado claramente diferenciado por conjuntos de su información almacenada-consultada:

- Datos catalogados por fecha, de forma que quede claro cómo y cuando historificar
- Datos catalogados por identificador en rangos de forma que cuando se consulta información, siempre acaban tocándose filas asociadas a los id entre rangos 10-100, o entre 101-200,...independientes entre si
- Tabla altamente accedida de forma que tengamos una zona “caliente” altamente concurrente, y una zona “fria” con menor necesidad de “velocidad” (siempre pensemos en grandes volúmenes de datos)



TIPOS DE PARTICIONAMIENTO



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Existen diferentes formas de Particiones:

- a) **Particiones horizontal:** Subdivisión por tuplas de una relación. Con la fragmentación horizontal derivada se utilizan las claves extranjeras.
- b) Particiones **vertical:** subdivisión de la relación por atributos de ésta. En esta fragmentación se conservan todas tuplas de la relación.
- c) Particiones **mixta:** es una mezcla de los dos tipos anteriores. Se divide la relación en tuplas y se cogen sólo los atributos que nos interesen.

BENEFICIOS DE LAS PARTICIONES



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Los índices y tablas particionadas poseen una serie de beneficios cuando se comparan con vistas particionadas manuales y otras formas de particionado manual:

- SQL Server maneja por nosotros donde colocar los datos
- Los objetos particionados se ven como objetos normales a todos los efectos, independientemente del numero de particiones
- Podemos trabajar a nivel de partición
- Se puede elegir una estrategia de particionado independiente para cada partición de un objeto
- Podemos reconstruir una partición de un índice, sin tocar el resto de particiones
- Existen optimizaciones específicas del motor relacional para trabajar de forma eficiente con particiones.
- Operadores específicos optimizados para operaciones con tablas particionadas
- Podemos configurar nivel de escalado de bloqueos a nivel de partición en lugar de a toda la tabla

BENEFICIOS DE LAS PARTICIONES



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Los índices y tablas particionadas poseen una serie de beneficios cuando se comparan con vistas particionadas manuales y otras formas de particionado manual:

- SQL Server maneja por nosotros donde colocar los datos
- Los objetos particionados se ven como objetos normales a todos los efectos, independientemente del numero de particiones
- Podemos trabajar a nivel de partición
- Se puede elegir una estrategia de particionado independiente para cada partición de un objeto
- Podemos reconstruir una partición de un índice, sin tocar el resto de particiones
- Existen optimizaciones específicas del motor relacional para trabajar de forma eficiente con particiones.
- Operadores específicos optimizados para operaciones con tablas particionadas
- Podemos configurar nivel de escalado de bloqueos a nivel de partición en lugar de a toda la tabla

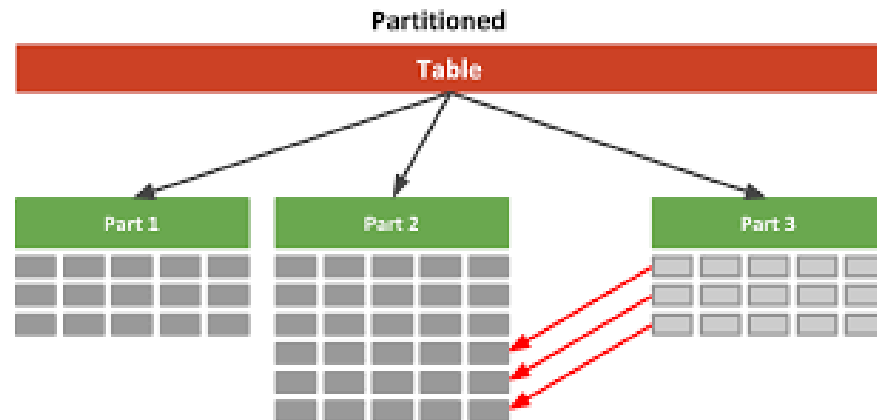
PARTICIONES HORIZONTAL

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

El particionamiento horizontal divide una tabla en múltiples tablas que contienen el mismo número de columnas, pero menos filas. Por ejemplo, si una tabla contiene un gran número de filas que representan reportes mensuales podría ser particionada horizontalmente en tablas por años, con cada tabla representando todos los reportes para un año específico. De esta manera las consultas que requieren datos para un año específico sólo referenciarán la tabla apropiada. Las tablas deberían ser particionadas en una manera que las consultas referencian tan pocas tablas como sea posible.



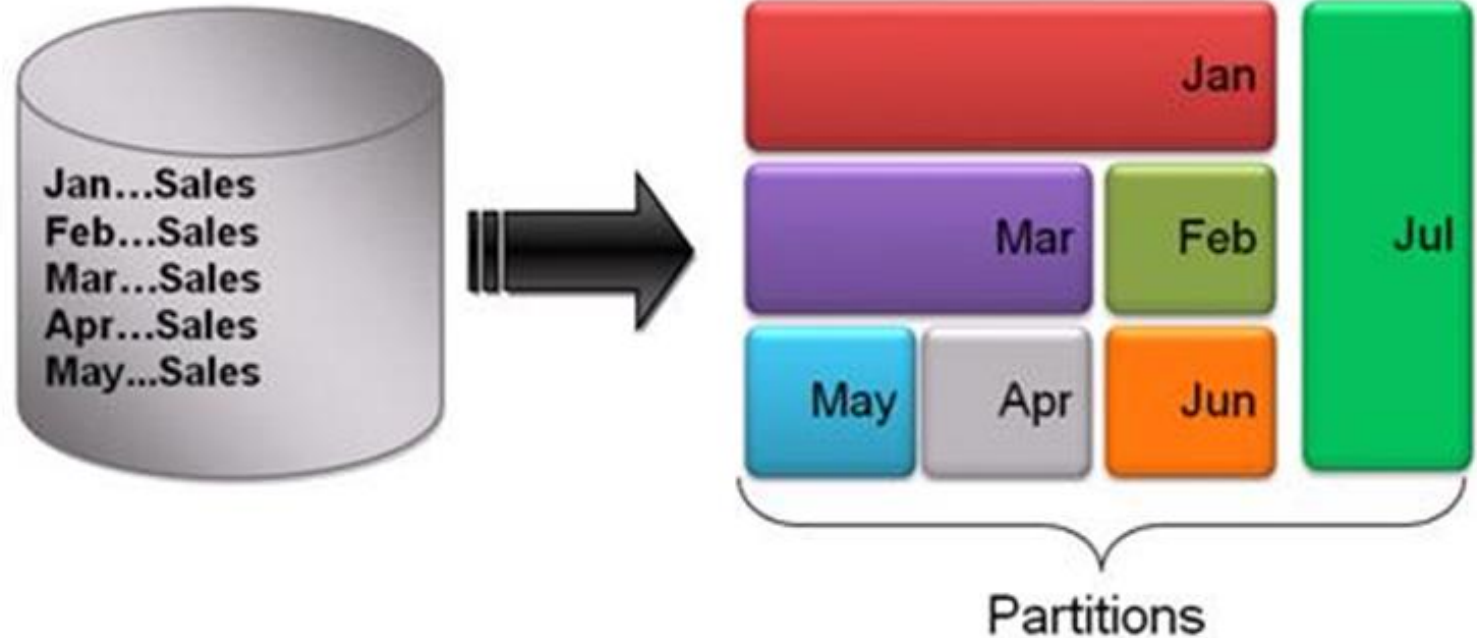
PARTICIONES HORIZONTAL

Las tablas son particionadas horizontalmente basadas en una columna que será usada para particionar y los rangos asociados a cada partición.

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL



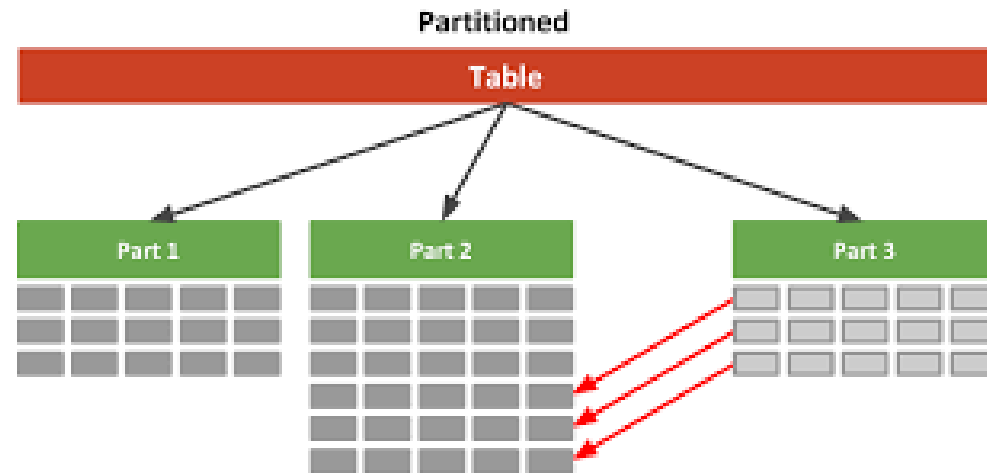
PARTICIONES HORIZONTAL

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

La columna de particionamiento es usualmente una columna de fecha pero todos los tipos de datos que son válidos para usarse como columnas de índice pueden ser usados como columna de partición.



Excepto **text**, **ntext**, **image**, **xml**, **timestamp**, **varchar(max)**, **nvarchar(max)**, **varbinary(max)**, tipos de datos de alias o tipos de datos definidos por el usuario

PARTICIONES HORIZONTAL

Hay dos enfoques diferentes que podríamos usar para lograr la partición de la tabla.



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

- El primero es crear una nueva tabla particionada y simplemente copiar los datos desde su tabla existente en la nueva tabla y renombrarla.
- El segundo enfoque es particionar una tabla existente reconstruyendo o creando un índice agrupado en la tabla.

PARTICIONES HORIZONTAL -SEGURIDAD



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Permiso ALTER ANY DATASPACE. De forma predeterminada, este permiso corresponde a los miembros del rol fijo de servidor **sysadmin** y a los roles fijos de base de datos **db_owner** y **db_ddladmin**.

Permiso CONTROL o ALTER en la base de datos en la que se está creando la función de partición y el esquema de partición.

Permiso CONTROL SERVER o ALTER ANY DATABASE en el servidor de la base de datos en la que se está creando la función de partición y el esquema de partición.

PARTICIONES HORIZONTAL



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

La creación de una tabla o índice con particiones tiene lugar normalmente en cinco partes:

1. Crear un grupo o grupos de archivos (filegroups)
2. Crear archivos correspondientes que contendrán las particiones especificadas por el esquema de partición. (datafiles/ .ndf)
3. Crear una función de partición que asigna las filas de una tabla o un índice a particiones según los valores de una columna especificada.
4. Crear un esquema de partición que asigna las particiones de una tabla o índice con particiones a los nuevos grupos de archivos.
5. Crear o modificar una tabla y un índice, especificar el esquema de partición como ubicación de almacenamiento.



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Para empezar a enseñarnos a particionar vamos a usar el primer escenario (hacer una tabla particionada y copiar los datos, renombrando la tabla)

A) Vamos crear una BD llamada : BDParticiones

```
CREATE DATABASE BDParticiones
ON PRIMARY
(
    NAME = 'BDParticiones.mdf',
    FILENAME = 'C:\ABD2021\BDParticiones.mdf'
)
LOG ON
(
    NAME = 'BDParticiones.ldf',
    FILENAME = 'C:\ABD2021\BDParticiones.ldf'
)
go
```



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

B) Vamos crear una tabla

-- CREAR UNA TABLA Reports

```
use BDParticiones  
go
```

```
CREATE TABLE Reports  
(IdReport int identity(1,1) PRIMARY KEY,  
  ReportDate date not null default getdate(),  
  ReportName varchar (100),  
  ReportNumber varchar (20),  
  ReportDescription varchar (max)  
)  
GO
```

C) Llenaremos la tabla (copia todo el código y ejecútalo completo)
Desde la declaración de las variables hasta el go después del commit

```
DECLARE @i int
DECLARE @fecha date
SET @i = 1

BEGIN TRAN
WHILE @i<=100000
BEGIN
    IF @i between 1 and 10000
        SET @fecha = '2018/01/15'
    IF @i between 10001 and 25000
        SET @fecha = '2018/03/15'
    IF @i between 25001 and 28000
        SET @fecha = '2018/04/15'
    IF @i between 28001 and 29500
        SET @fecha = '2018/03/15'
    IF @i between 29501 and 31000
        SET @fecha = '2019/02/15'
    IF @i between 31001 and 32000
        SET @fecha = '2019/03/15'
    IF @i between 32001 and 35000
        SET @fecha = '2019/04/15'
    IF @i between 35001 and 42500
        SET @fecha = '2019/05/15'
    IF @i between 42501 and 45000
        SET @fecha = '2019/06/15'
    IF @i between 45001 and 47500
        SET @fecha = '2019/07/15'
    IF @i between 47501 and 52500
        SET @fecha = '2019/08/15'
    IF @i between 52501 and 55000
        SET @fecha = '2019/09/15'
    IF @i between 55001 and 60000
        SET @fecha = '2019/10/15'
    IF @i between 60001 and 62475
        SET @fecha = '2019/11/15'
    IF @i between 62476 and 65345
        SET @fecha = '2019/12/15'
```

NOTA: VA DURAR UN
POQUITO POR LA CANTIDAD
DE REGISTROS

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

PARTICIONES HORIZONTAL

PRACTICA 9



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
IF @i between 65346 and 66000
    SET @fecha = '2020/01/15'
IF @i between 66001 and 67000
    SET @fecha = '2020/02/15'
IF @i between 67001 and 68000
    SET @fecha = '2020/03/15'
IF @i between 68001 and 69000
    SET @fecha = '2020/04/15'
IF @i between 69001 and 70000
    SET @fecha = '2020/05/15'
IF @i between 70001 and 70500
    SET @fecha = '2020/06/15'
IF @i between 70500 and 72000
    SET @fecha = '2020/07/15'
IF @i between 72001 and 73000
    SET @fecha = '2020/08/15'
IF @i between 73001 and 73800
    SET @fecha = '2020/09/15'
IF @i between 73801 and 73950
    SET @fecha = '2020/10/15'
IF @i between 73951 and 75700
    SET @fecha = '2020/11/15'
IF @i between 75701 and 75802
    SET @fecha = '2021/12/15'
IF @i between 75803 and 80000
    SET @fecha = '2021/01/15'
IF @i between 80001 and 90000
    SET @fecha = '2021/02/15'
IF @i between 90001 and 100000
    SET @fecha = '2021/03/15'
INSERT INTO Reports
(
    ReportDate,
    ReportName,
    ReportNumber,
    ReportDescription
)
VALUES
(
    @fecha,
    'ReportName' + CONVERT (varchar (20), @i) ,
    CONVERT (varchar (20), @i),
    REPLICATE ('Report', 1000)
)
SET @i=@i+1
END
COMMIT TRAN
GO
```

D) Verificar que la tabla ya contiene los 100000 registros.

```
select count(*) from Reports
```

E) Empezamos los pasos de la partición

1) Crear un grupo o grupos de archivos (como vamos a particionar por años la tabla, entonces ocupamos 4 filegroups)

```
use BDParticiones
ALTER DATABASE BDParticiones
ADD FILEGROUP Histo2018
GO
ALTER DATABASE BDParticiones
ADD FILEGROUP Histo2019
GO
ALTER DATABASE BDParticiones
ADD FILEGROUP Histo2020
GO
ALTER DATABASE BDParticiones
ADD FILEGROUP Histo2021
GO
```

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

PARTICIONES HORIZONTAL

PRACTICA 9



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

2) Crear archivos correspondientes que contendrán las particiones (datafile - *.ndf) y le asignamos el filegroup correspondiente

*** Crearemos 4 carpetas en nuestra carpeta de trabajo simulando que cada una sera un disco diferente y ahi crearemos cada datafile**

```
ALTER DATABASE BDParticiones
ADD FILE (
    NAME = 'Particion2018.NDF',
    FILENAME = 'C:\ABD2021\DISC01\Particion2018.NDF'
) TO FILEGROUP Histo2018;
```

```
ALTER DATABASE BDParticiones
ADD FILE
(
    NAME = 'Particion2019.NDF',
    FILENAME = 'C:\ABD2021\DISC02\Particion2019.NDF'
) TO FILEGROUP Histo2019;
```

LO MISMO PARA LOS OTROS 2 (2020 Y 2021)

PARTICIONES HORIZONTAL

Para verificar los grupos de archivos creados y disponibles en la base de datos actual ejecute la siguiente consulta:

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
SELECT name AS AvailableFilegroups
FROM sys.filegroups
WHERE type = 'FG'
```

En modo grafico debes dar click derecho sobre la base de datos y elegir propiedades, y elegir grupo de archivos o filegroups debe salir la lista de todos los filegroups creados encabezado por el PRIMARY que se creo al crear la BD. Si quieres checar los archivos NDF es en files y por commando es:

```
SELECT
name as [FileName], physical_name as [FilePath]
FROM sys.database_files
where type_desc = 'ROWS'
GO
```

PARTICIONES HORIZONTAL

PRACTICA 9

3) Crear una función de partición que asigna las filas de una tabla o un índice a particiones según los valores de una columna especificada.

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

NOMBRE DE FUNCION

TIPO DE DATO DEL
CAMPO EN QUE SE
BASARÁ LA PARTICION

```
CREATE PARTITION FUNCTION F_PartitionByAnio(date)
AS RANGE LEFT FOR VALUES ( '2018-12-31',
                              '2019-12-31',
                              '2020-12-31')
```

TIPO DE RANGO

{RIGHT/LEFT}

DEPENDIENDO LO QUE
USEMOS TOMA EL RANGO
HACIA LA IZQUIERDA O HACIA
LA DERECHA

VALORES APARTIR DEL CUAL SE
REALIZARA LA SEGMENTACIÓN
NO. VALORES = NO. PARTICIONES - 1



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

Partition Function

La función de partición o partition function es la encargada de definir los valores límite de las particiones de la columna. Cuando se define la partition function se crean siempre («numero de valores límite» + 1) particiones.

Se pueden definir dos tipos de partition functions en cuanto al rango de valores:

- **LEFT:** El valor LEFT indica que el valor límite se almacena en la partición izquierda, y es el último valor de esa partición.
- **RIGHT:** El valor RIGHT indica que el valor límite se almacena en la partición derecha, y es el primer valor de esa partición.

LEFT FOR VALUES ('G','N')

Si lo hacemos así el primer rango incluirá la letra G
Segundo de la G hasta la N
y tercero después N todos

RIGTH FOR VALUES ('G','N')

hasta antes de la letra G
todos entre G y hasta antes de la N
y el resto incluyendo la N

4) Crear un esquema de partición que asigne las particiones de una tabla o índice con particiones a los nuevos grupos de archivos.



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
CREATE PARTITION SCHEME EsquemaByAnio  
AS PARTITION F_PartitionByAnio  
TO (Histo2018, Histo2019, Histo2020, Histo2021);
```

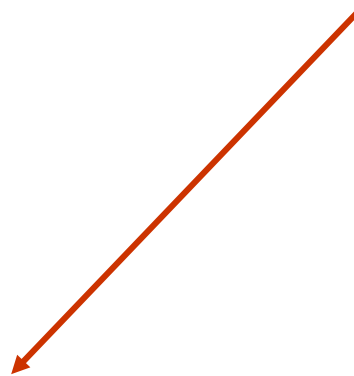
NOMBRE DEL ESQUEMA
DE PARTICION



FUNCION DE PARTICIÓN



FILEGROUPS



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

5) Crear o modificar una tabla y un índice, especificar el esquema de partición como ubicación de almacenamiento.

En este caso como estamos en el escenario 1 y cuando el campo de particion no es la PK.

Vamos a crear una table igual a la original queremos particionar, le vamos a crear un indices clustereado asignado el esquema de partición, despues le vamos a vaciar los datos y por último borramos la otra table y renombramos la table.

LE CREAMOS LA PK PERO LE INDICAMOS QUE EL INDICE SEA NONCLUSTERED, ADEMMAS NO LE PUSIMOS LA PROPIEDAD IDENTITY POR QUE LE VACIAREMOS LOS DATOS DE LA TABLA ORIGINAL

```
CREATE TABLE Reports_Particionada
(IdReport int PRIMARY KEY NONCLUSTERED,
 ReportDate date not null default getdate(),
 ReportName varchar (100),
 ReportNumber varchar (20),
 ReportDescription varchar (max)
)
GO
```

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
-- CREANDO EL INDICE CLUSTEREO POR EL CAMPO  
-- DE LA PARTIÇÃO Y ASIGNAMOS EL ESQUEMA DE -- ---- PARTIÇION,  
CON ESTO ESTAMOS CREANDO EL INDICE Y  
-- PARTIÇIONANDO LA TABLA
```

```
CREATE CLUSTERED INDEX IDX_RepPart  
ON Reports_Particionada (ReportDate)  
ON EsquemaByAnio(ReportDate);
```

ESQUEMA DE PARTIÇION

CAMPO QUE SE UTILIZARA
PARA REALIZAR
LA PARTIÇION

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
-- COPIAR LOS DATOS CON INSERT-SELECT
INSERT INTO Reports_Particionada
SELECT * FROM Reports
-- COMPROBAR QUE SI SE PASARON
SELECT count(*) FROM Reports_Particionada

-- ELIMINAR LA TABLA ORIGINAL Y RENOMBRAR
-- LA PARTICIONADA, EN CASO DE TENER FK, O
-- TABLAS HIJAS RESTRABLECER LAS RELACIONES
-- Y TODOS LOS CONSTRAINT PARA QUEDE COMO
-- LA ORIGINAL

DROP TABLE Reports;

EXEC sp_rename 'Reports_Particionada', 'Reports';
```

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
--VERIFICANDO QUE REALMENTE SE PARTICIONO
```

```
-- CONSULTAR TODA LA TABLA
```

```
SELECT * FROM Reports
```

```
-- MOSTRAR LOS REGISTROS DE CADA PARTICION
```

```
-- SE USA EL NOMBRE DE LA FUNCION DE PARTICION
```

```
-- Y EL NOMBRE DEL CAMPO QUE USAMOS PARA
```

```
-- PARTICIONAR
```

```
SELECT * FROM Reports
```

```
WHERE $partition.F_PartitionByAnio(ReportDate)=1
```

```
GO
```

```
SELECT * FROM Reports
```

```
WHERE $partition.F_PartitionByAnio(ReportDate)=2
```

```
GO
```

```
SELECT * FROM Reports
```

```
WHERE $partition.F_PartitionByAnio(ReportDate)=3
```

```
GO
```

```
SELECT * FROM Reports
```

```
WHERE $partition.F_PartitionByAnio(ReportDate)=4
```

```
GO
```

```
-- otra forma de hacerlo
```

```
SELECT * FROM Reports
```

```
WHERE Year(ReportDate) = '2018'
```

```
-- EJECUTE CADA CONSULTA PARA QUE VEA LA DIFERENCIA
```

```
-- ENTRE CONSULTAR TODA LA TABLA O UNA PARTICION
```

```
-- EN ESPECIFICO O SI USTED HACE UNA CONSULTA
```

```
-- DE UN AÑO EN ESPECIFICO
```

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

-- MUESTRE CUANTOS REGISTRO TIENE CADA PARTICION

```
SELECT p.partition_number AS Num_Particion,  
f.name AS Nombre, p.rows AS Columnas  
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) = 'Reports'  
GO
```

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

1. Verificar si se crearon los filegroups y datafiles

```
SELECT name AS NombresFilegroups
FROM sys.filegroups
WHERE type = 'FG'
```

```
SELECT
name as [FileName], physical_name as [FilePath]
FROM sys.database_files
where type_desc = 'ROWS'
```

2. Muestre cuantos registros tiene cada partición

```
SELECT p.partition_number AS Num_Particion, f.name AS
Nombre, p.rows AS Columnas
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) = 'Reports'
AND p.index_id =1
```

3. Muestre los registros de la tabla

```
select * from NOMBRETABLA
```

4. Muestre los registros de cada partición

```
SELECT * FROM TABLA_PARTICIONADA
WHERE
$partition.FuncionParticion(campodepartición)= #Partición
```




PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

REAFIRMANDO LA PRACTICA 9..... UTILIZANDO LA BD DE ALMACEN.

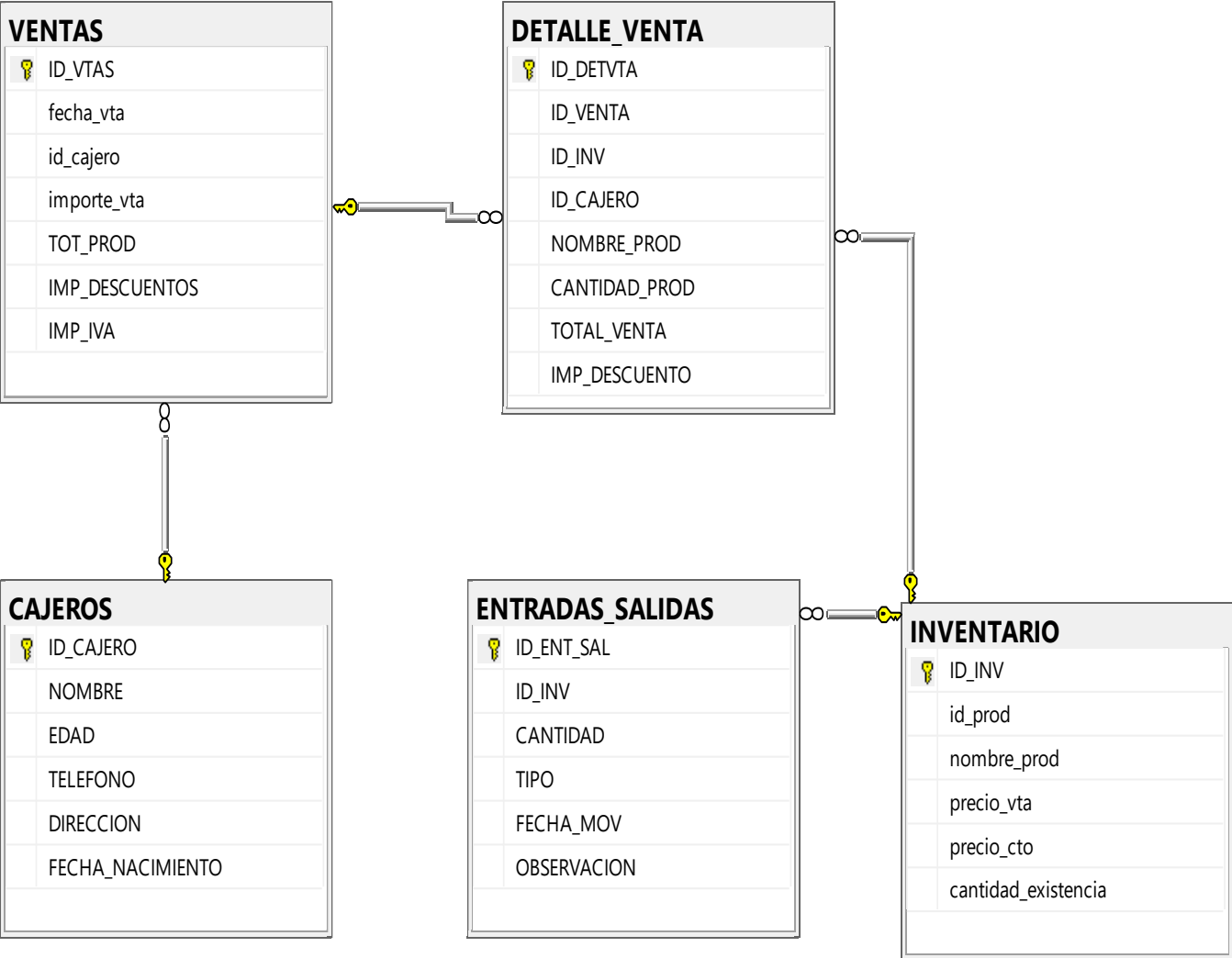
1. Aplique el mismo enfoque de la practica 9 para particionar. (crear tabla nueva y renombrar) para particionar
2. La tabla de ventas de la BD Almacen será la que se particionara. Les subire al drive la BD.
3. Deben insertar primero al menos 100000 (todos los meses del 2020 y lo que va del 2021) en la de ventas, y en las tablas los registros necesarios
4. 4. Particionar por mes del 2020 y una particion final para todo el 2021. Es decir deben quedarle 13 filegroups y 13 datafiles.
5. Reutilice los discos (carpetas ya creadas) solo crear las adicionales que ocupe.
6. Utilice el RIGTH en la función de partición.
7. Reconstruir todo lo que tuvo que quitar para poder realizar la partición, a final la bd de datos debe quedar estructurada a la base de datos original. Anexo en la siguiente diapositiva



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL



PARTICIONES HORIZONTAL

DEBEMOS SABER

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
-- BORRAR FILEGROUP
ALTER DATABASE PartitionDB
REMOVE FILEGROUP December;
GO

USE MASTER
-- QUITAR DATAFILE
ALTER DATABASE PartitionDB
REMOVE FILE [PartitionDBAbr.NDF]
--VACIAR UN DATAFILE
DBCC SHRINKFILE ([PartitionDBAbr.NDF], EMPTYFILE);
GO

-- BORRAR LA FUNCION DE PARTICION
drop partition function
F_PartitionByMonth

-- BORRAR EL ESQUEMA DE PARTICION
DROP PARTITION SCHEME
PartitionBymonth
```

PARTICIONES HORIZONTAL

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

NOTA: Recuerda que cuando el campo de la llave primaria **NO** es el campo de la partición. Tienes que hacer en dos pasos crear la tabla y su PK y después el índice clustereado por el campo de la partición

```
CREATE TABLE Reports
(
  IdReport int identity(1,1) PRIMARY KEY NONCLUSTERED,
  ReportDate date not null default getdate(),
  MonthlyReport varchar(max)
)
GO
```

INDICAR QUE EL INDICE
QUE CREA CON
LA LLAVE PRIMARIA
SEA NO CLUSTEREO

```
CREATE CLUSTERED INDEX IDX_PART_FECHA
ON Reports (ReportDate)
ON PartitionBymonth(ReportDate)
```

CREAR INDICE CLUSTEREO
DEL CAMPO(S) DE LA PARTICION

2DA. FORMA PARTICIONES HORIZONTAL

Cuando la tabla ya tiene datos y constraints.

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
-- CONSULTAR LOS CONSTRAINT DE PRIMARY KEY EN LA TABLA
exec sp_helpconstraint Reports2
-- VERIFICAR SI HAY TABLAS DE LA QUE ES PADRE CONSTRAINT
-- DE FOREIGN KEY DE OTRAS TABLAS REFERENCIADA (REFERENCES)
exec sp_helpconstraint xxxxx

-- QUITAR CONSTRAINT PK Y FK CUANDO ES PADRE.
alter table Reports2
drop constraint PK__Reports2__46F9D6CEF06860FA

-- AGREGAR LOS CONSTRAINT ELIMINADOS PERO EL CONSTRAINT
DE PRIMARY FORZAR QUE NO SEA CLUSTEREDO
alter table Reports2
ADD CONSTRAINT PK_Reports2
PRIMARY KEY NONCLUSTERED (IdReport)

--CREAR INDICE CLUSTERADO ASIGNANDO EL ESQUEMA
--DE PARTICION
CREATE CLUSTERED INDEX IDX_PART_FECHA
ON Reports2 (ReportDate)
ON PartitionBymonth(ReportDate)

--VERIFICAR QUE TODO HAYA QUEDADO BIEN ,EN LOS CONSTRAINT
--GENERAR NUEVO DIAGRAMA
exec sp_helpconstraint Reports2
```

PARTICIONES HORIZONTAL

• PRACTICA 11



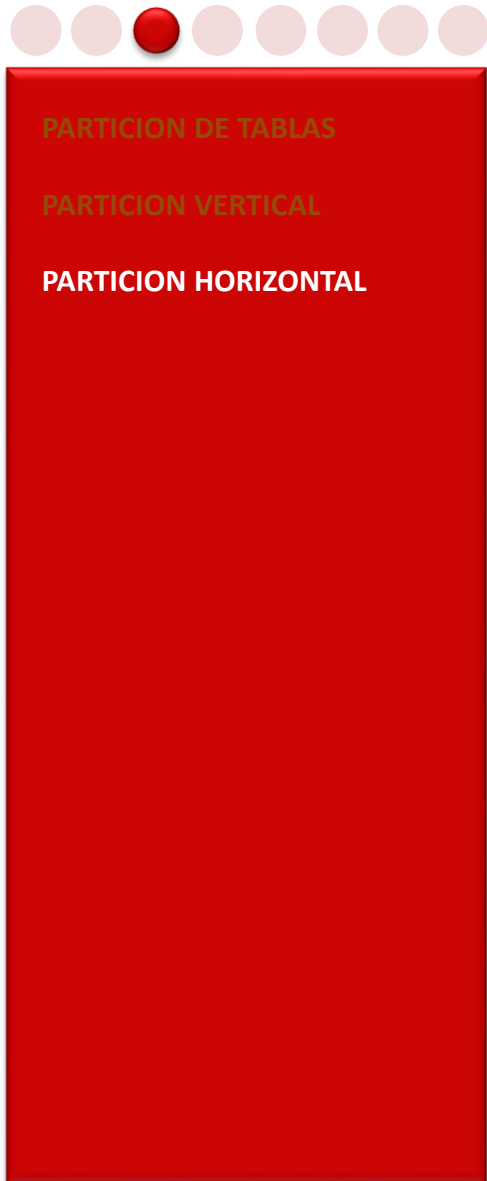
PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

1. Usa El Query De La Bd De Datos Northwind, Denuevo Y Cree La Bd Pero Con Otro Nombre Aunque Tenga El Mismo Contenido.
2. Actualice Las Fechas De Las Ordenes, Que Sean De Los Años Mas Recientes.
3. Particione La Tabla Ordes Por Rangos De Ordenes Las Id

RANGO	PARTICION
10000 A 10300	PRIMERA
10301 A 10500	SEGUNDA
10501 A 10700	TERCERA
10701 A 10900	CUARTA
10901 A 11100	QUINTA
11101 EN ADELANTE	SEXTA



1. Usa La Bd De OFICIALIA DE PARTES, es una BD en donde se registran los oficios que llegan a Gobierno del Estado para las diferentes oficinas y departamentos del mismos.
2. Inserta los siguientes datos a la table de tipos de documentos
(OFICIOS, MEMORANDUM, TARJETA INFORMATIVA, RECONOCIMIENTO)
3. Inserta al menos 10 destitatarios – Emisores
4. Inserta usando GO# documentos recibidos en la table de RECEPCION debes insertar de los diferentes tipos de documentos asi que usa el GO con diferente cantidad.
50,000
5. Debe particionar la table de RECEPCION, Usted decida como se debe particionar con que campo y con que enfoque

PARTICIONES HORIZONTAL

- PRACTICA 13



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

LA BD AdventureWorks es una bd de ejemplo que nos proporciona Microsoft para sqlserver hay diferentes versiones, esta práctica deberá realizarlo sin crear una tabla adicional.

REALIZAR PARTICION HORIZONTAL EN TRES ARCHIVOS DE LA TABLA Person.Address DE AdventureWorks POR LA COLUMNA DE 'City' TOMANDO EL SIGUIENTE ORDEN

Particion1 de A -> I

Particion2 de J -> P

Particion3 de Q -> Z

Te recuerdo que en todas las practicas de particiones deberás realizar ciertas consultas de comprobación y que la BD siempre deberá quedar con la misma estructura que tenía.

Así que otra comprobación sería generar el diagrama al final

PARTICIONES HORIZONTAL

- PRACTICA 14



PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

LA BD AdventureWorks es una bd de ejemplo que nos proporciona Microsoft para sqlserver hay diferentes versiones, esta práctica deberá realizarlo sin crear una tabla adicional.

REALIZAR PARTICION HORIZONTAL EN TRES ARCHIVOS DE LA TABLA Person.Address DE AdventureWorks POR LA COLUMNA DE 'City' TOMANDO EL SIGUIENTE ORDEN

Particion1 de A -> I

Particion2 de J -> P

Particion3 de Q -> Z

Te recuerdo que en todas las practicas de particiones deberás realizar ciertas consultas de comprobación y que la BD siempre deberá quedar con la misma estructura que tenía.

Así que otra comprobación sería generar el diagrama al final

PARTICION VERTICAL

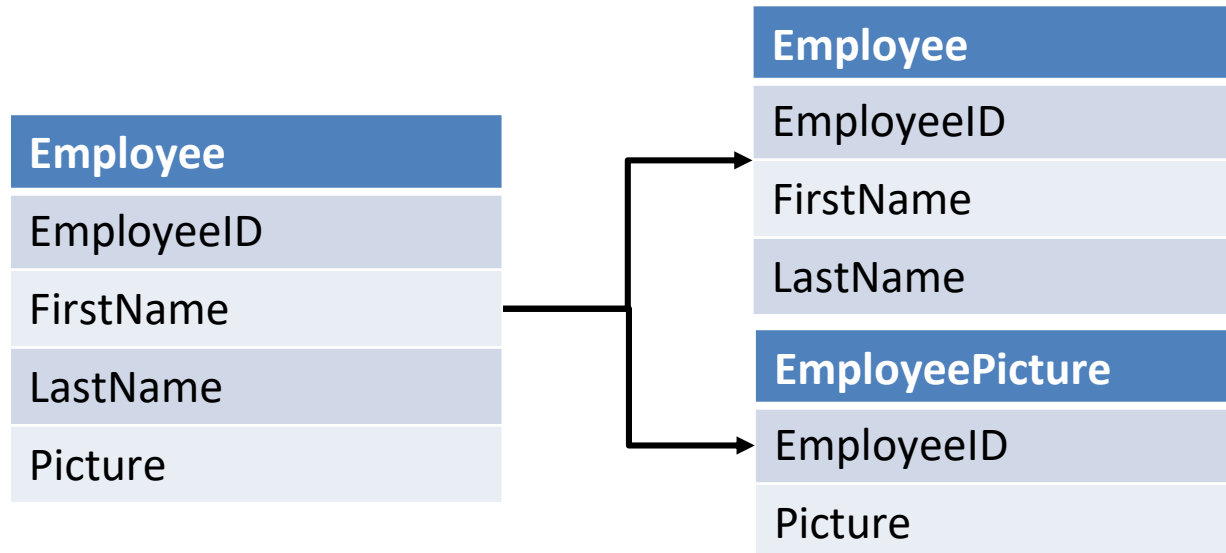
PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

El particionamiento vertical de tablas es principalmente usado para incrementar el desempeño del manejador de BD especialmente en casos cuando una consulta retorna todas las columnas de una tabla que contiene un número de columnas de texto muy amplio o BLOB.

En este caso, para reducir los tiempos de acceso, las columnas BLOB pueden ser divididas a su propia tabla. Otro ejemplo es restringir el acceso a datos sensibles, por ejemplo contraseñas, información salarial, etc. La partición vertical divide una tabla en dos o más tablas que contienen diferentes columnas:





PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

PASO 1. Crearemos la tabla con el índice clustereado de la llave primaria.

```
CREATE TABLE EmployeeReports
(
  ReportID int IDENTITY (1,1) NOT NULL,
  ReportName varchar (100),
  ReportNumber varchar (20),
  ReportDescription varchar (max)
  CONSTRAINT PK_EReport PRIMARY KEY
  CLUSTERED (ReportID)
)
```

PARTICION VERTICAL

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

PASO 2. Le pondremos 100000 registros

```
DECLARE @i int
SET @i = 1

BEGIN TRAN
WHILE @i<100000
BEGIN
INSERT INTO EmployeeReports
(
ReportName,
ReportNumber,
ReportDescription
)
VALUES
(
'ReportName' + CONVERT (varchar (20), @i) ,
CONVERT (varchar (20), @i),
REPLICATE ('Report', 1000)
)
SET @i=@i+1
END
COMMIT TRAN
GO
```

PARTICION VERTICAL

Nota: solo para que al final comprobar la mejora de rendimiento con particiones consultaremos la tabla

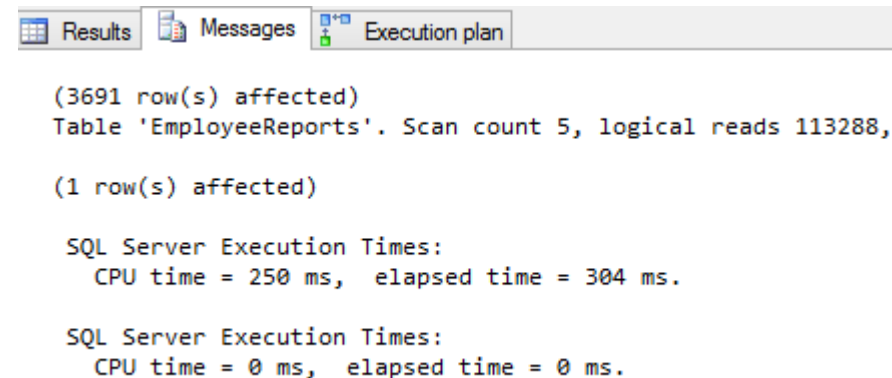
PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
SET STATISTICS IO ON
SET STATISTICS TIME ON
SELECT er.ReportID, er.ReportName,
er.ReportNumber
FROM dbo.EmployeeReports er
WHERE er.ReportNumber LIKE '%33%'
SET STATISTICS IO OFF
SET STATISTICS TIME OFF
```

Ver los messages



The screenshot shows the 'Messages' tab in SQL Server Enterprise Manager. It displays the results of the SQL query executed, including the number of rows affected, scan count, logical reads, and execution times for both the SQL Server and the user.

```
(3691 row(s) affected)
Table 'EmployeeReports'. Scan count 5, logical reads 113288,

(1 row(s) affected)

SQL Server Execution Times:
    CPU time = 250 ms,  elapsed time = 304 ms.

SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 0 ms.
```

PARTICION VERTICAL

PASO 3. Particionaremos verticalmente la tabla EmployeesReport

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

```
CREATE TABLE ReportsDesc
( ReportID int REFERENCES EmployeeReports(ReportID),
  ReportDescription varchar(max)
  CONSTRAINT PK_ReportDesc PRIMARY KEY CLUSTERED(ReportID)
);
CREATE TABLE ReportsData
(
  ReportID int NOT NULL,
  ReportName varchar (100),
  ReportNumber varchar (20),
  CONSTRAINT DReport_PK PRIMARY KEY CLUSTERED (ReportID)
)

INSERT INTO dbo.ReportsData
(
  ReportID,
  ReportName,
  ReportNumber
)
SELECT er.ReportID,
er.ReportName,
er.ReportNumber
FROM dbo.EmployeeReports er
```

PARTICION VERTICAL

PARTICION DE TABLAS

PARTICION VERTICAL

PARTICION HORIZONTAL

NOTA: Corremos la misma consulta de búsqueda

```
SET STATISTICS IO ON
SET STATISTICS TIME ON
SELECT er.ReportID, er.ReportName,
er.ReportNumber
FROM ReportsData er
WHERE er.ReportNumber LIKE '%33%'
SET STATISTICS IO OFF
SET STATISTICS TIME OFF
```

```
(3691 row(s) affected)
Table 'ReportsData'. Scan count 1, logical reads 421,
```

```
(1 row(s) affected)
```

```
SQL Server Execution Times:
    CPU time = 31 ms,  elapsed time = 104 ms.
```

```
SQL Server Execution Times:
    CPU time = 0 ms,  elapsed time = 0 ms.
```