#CrecimientoProfesional
#AprendeConLosPioneros

Online | DMC Perú

DMC Perú | CAPACITACIÓN PROFESIONAL

SESIÓN XII

Docente: Victor Gutierrez
Data Architect

Agenda

MANEJO DE TABLAS E INDICES

- Particiones (Partición By Functions).
- Uso de tablas temporales.
- Técnicas de filtrado.
- Ordenamiento de registros (Random Sorting).
- Tipos y creación de índices.
- Mantenimiento y fragmentación de los índices.

Particiones

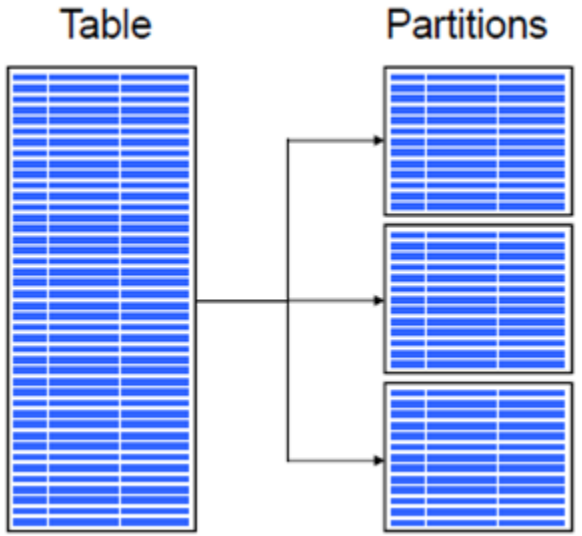
Una partición es **una división de una tabla en diferentes partes**, cuyo almacenamiento está separado en base a unas reglas específicas. Hay varios tipos de particionamiento (según cada versión de motor de BD).

¿Porqué es importante particionar las tablas?

a) Permite enfocar el procesamiento en particiones específicas y no en toda la tabla.
b) Facilita las labores de mantenimiento: estadísticas, backup, restore, depuración

Ventajas

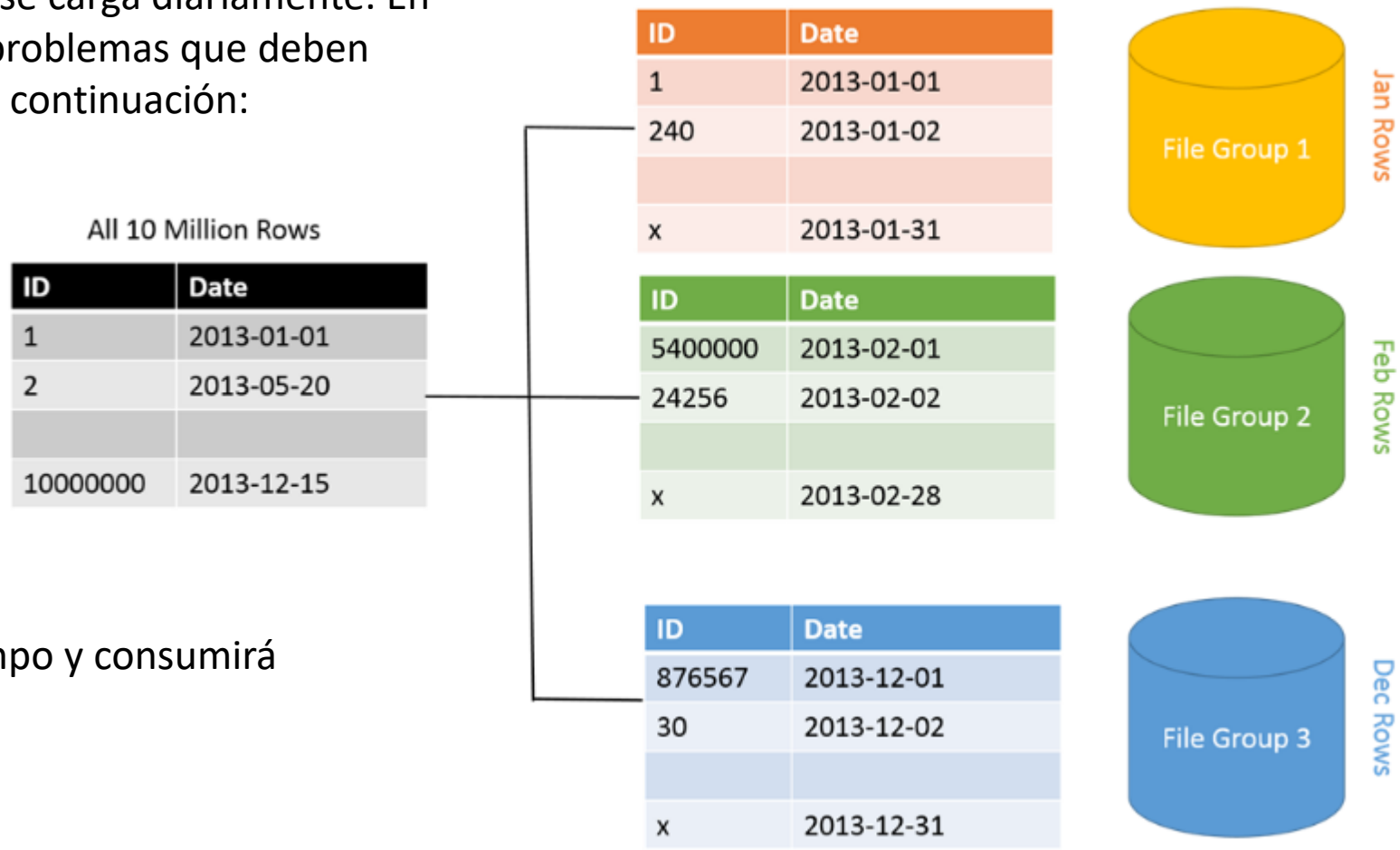
- Funcionalidad que permite dividir objetos en piezas pequeñas
- Transparente a las aplicaciones.
- Incrementa el gestión y disponibilidad de los datos.
- Incremento en el tiempo de respuesta del acceso.



Particiones

¿Particionar?

Supongamos que tenemos una tabla que se carga diariamente. En este caso, la tabla puede causar algunos problemas que deben resolverse mediante los pasos definidos a continuación:

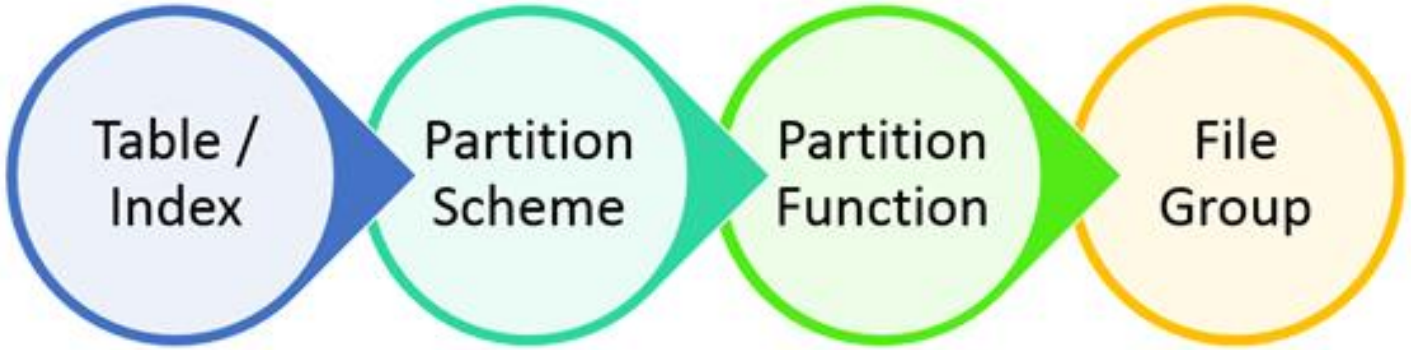


- Mantener la tabla, tomará mucho tiempo y consumirá más recursos (CPU, IO, etc.).
- Problemas de bloqueo.

Particiones

Básicamente, la tabla de particiones se divide en cuatro pasos. Comenzando a almacenar las filas en tablas e índices, si no define un grupo de archivos específico, esas filas se almacenan en el grupo de archivos predeterminado. Sin embargo, es posible crear una tabla o índice que informe el Esquema de partición que se almacenará en un Grupo de archivos definido. Cada esquema de partición podría asignarse en uno o más grupos de archivos. Las filas son dirección al Grupo de archivos derecho basado en el algoritmo creado en la Función de partición.

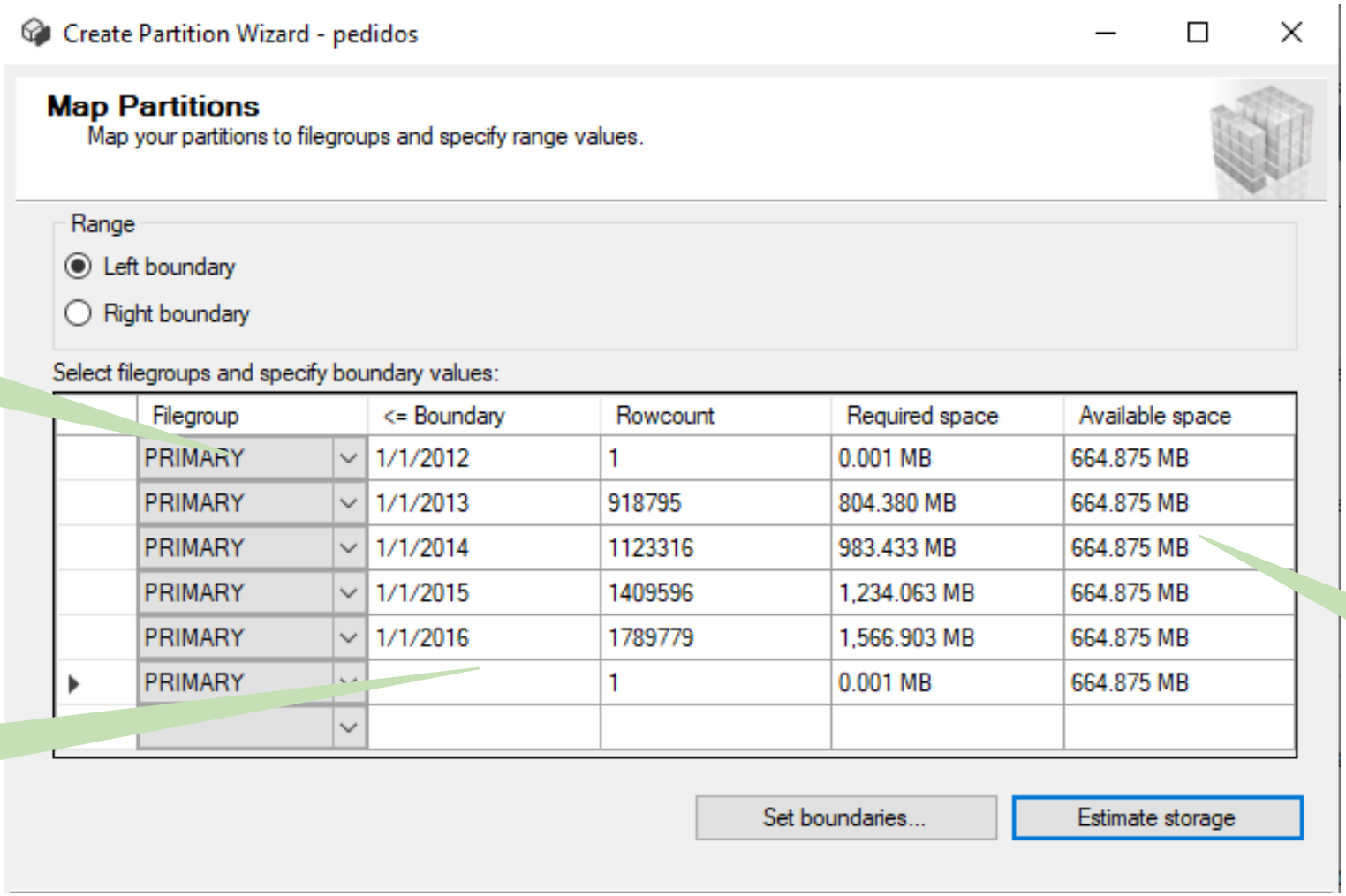
El flujo es algo como esto:



Particiones

Las particiones pueden crearse en diferentes FG

Las particiones permiten mantener los datos por rango



Cada partición tendrá un size y numero de registros.

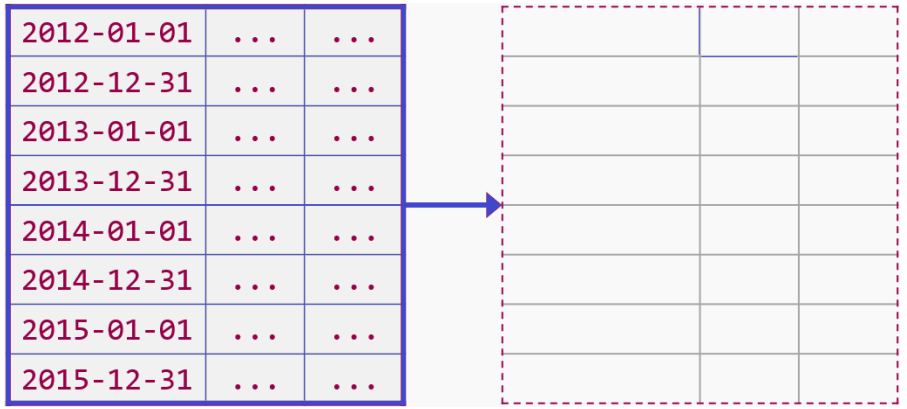
Particiones / Operaciones

Operación	Descripción
MERGE	Unir dos particiones en una sola (obviamente las dos particiones deben estar juntas)
SPLIT	Separar una partición en 2 particiones
SWITCH	Mover una partición de un objeto a otro
TRUNCATE	Permite truncar la partición directamente, evitando el uso de la instrucción *DELETE*

Particiones / Operaciones

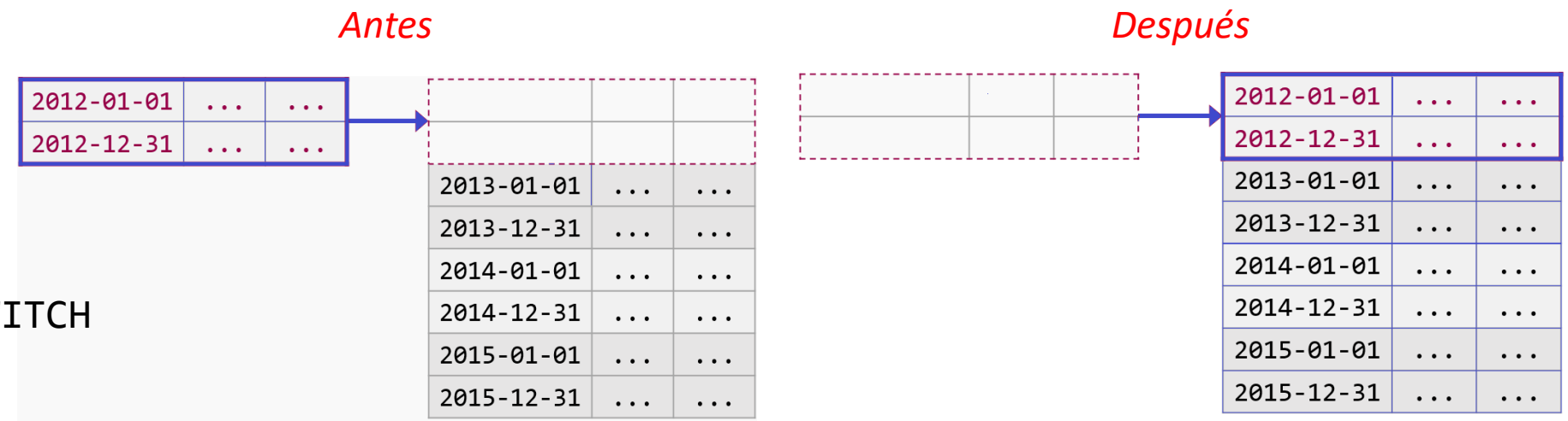
Switch from Non-Partitioned to Non-Partitioned

```
ALTER TABLE Source SWITCH TO
Target PARTITION 1
```



Load data by switching in: Switch from Non-Partitioned to Partition

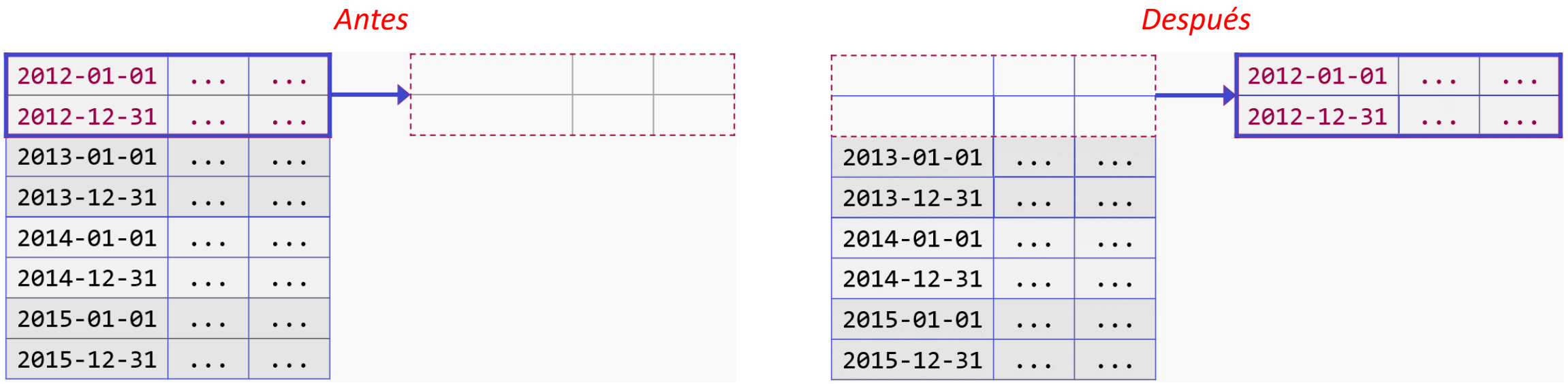
```
ALTER TABLE Source SWITCH
PARTITION 1 TO Target
```



Particiones / Operaciones

Archive data by switching out: Switch from Partition to Non-Partitioned

```
ALTER TABLE Source SWITCH PARTITION 1 TO Target
```

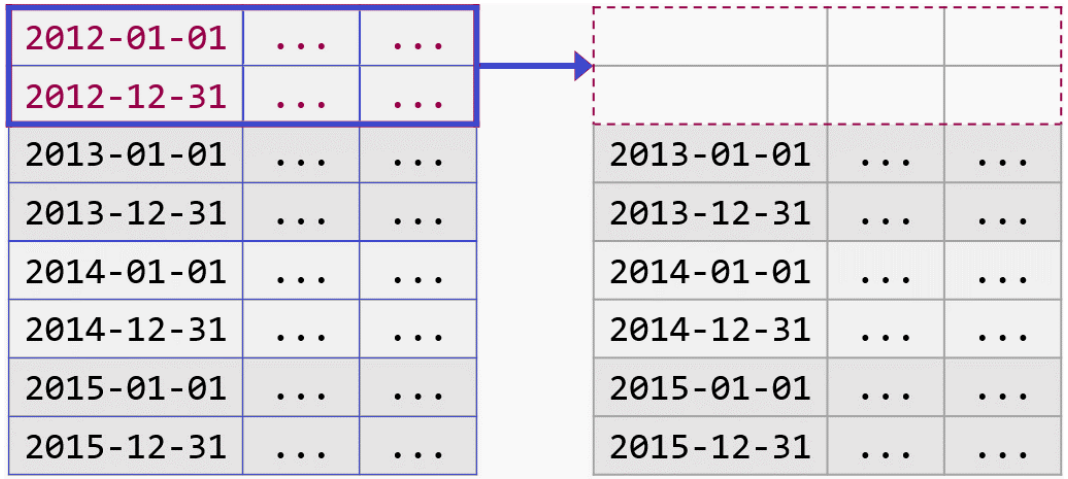


Particiones / Operaciones

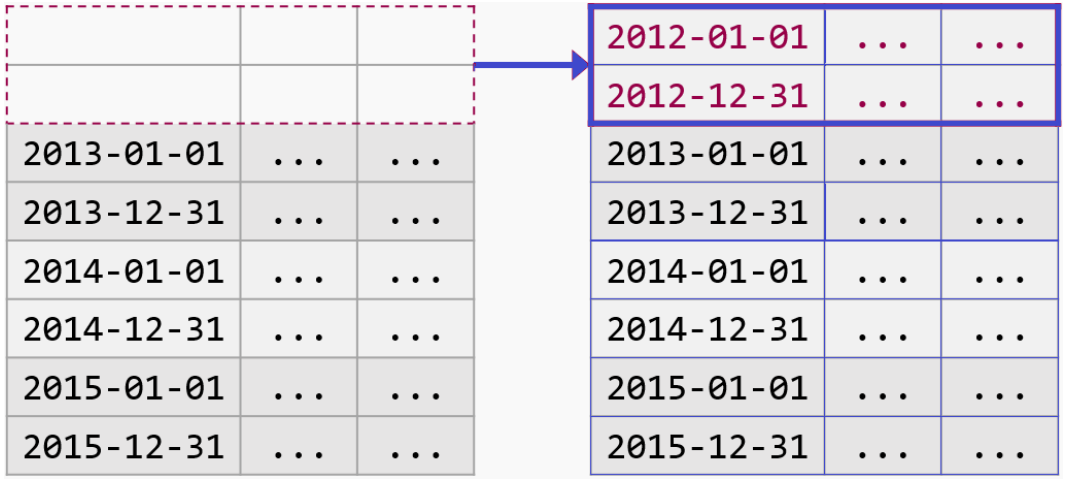
Switch from Partition to Partition

```
ALTER TABLE Source SWITCH PARTITION 1 TO Target PARTITION 1
```

Antes



Después



REFERENCIAS

Alter Partition
https://docs.microsoft.com/es-es/sql/t-sql/statements/alter-partition-function-transact-sql?view=sql-server-ver15

Crear Partición por función
https://docs.microsoft.com/es-es/sql/t-sql/statements/create-partition-function-transact-sql?view=sql-server-ver15

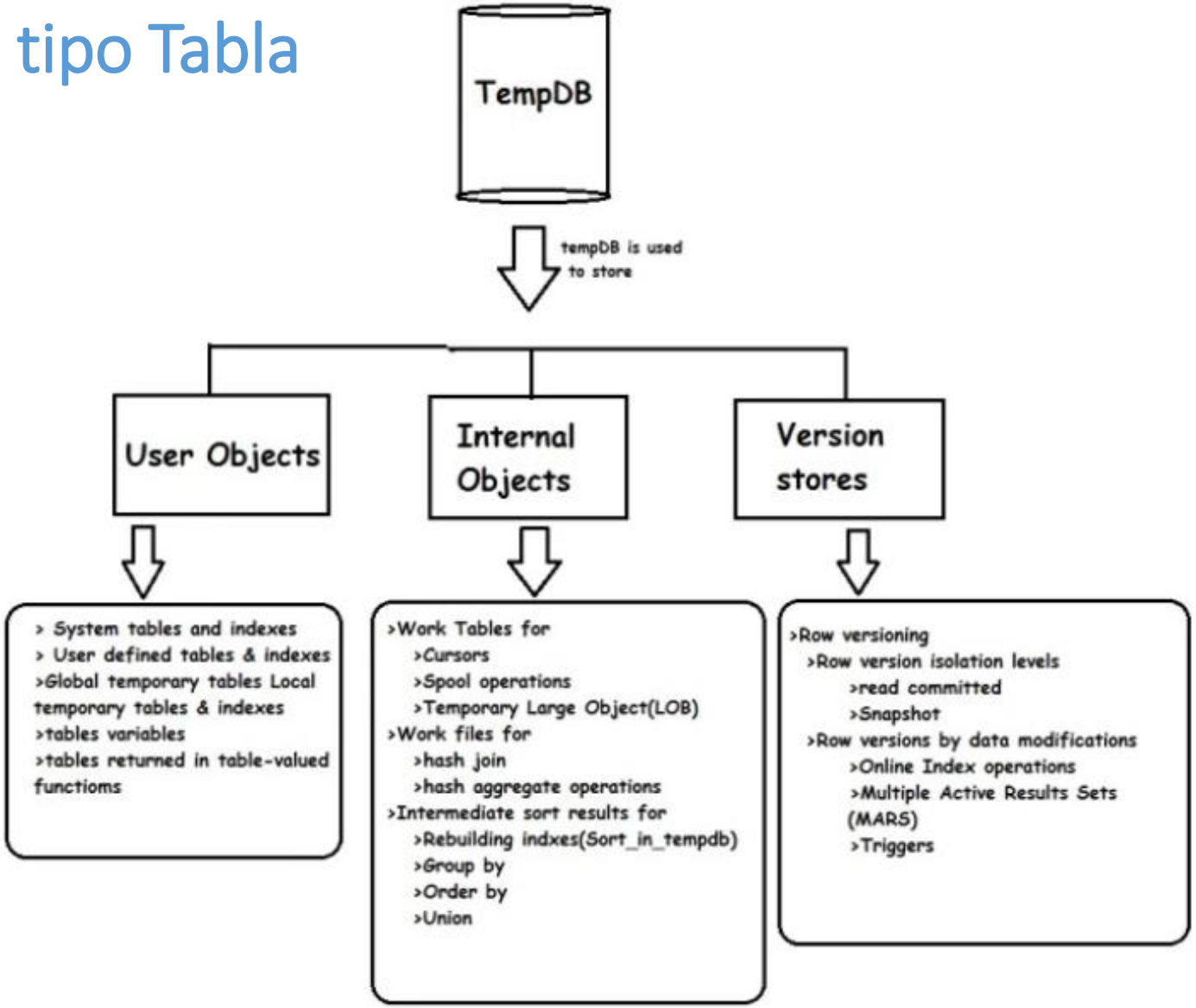
$PARTITION
https://docs.microsoft.com/es-es/sql/t-sql/functions/partition-transact-sql?view=sql-server-ver15

Tablas temporales y Variables tipo Tabla

El uso del **tempDB**

- Las tablas temporales se guardan ahí
- Las Variables de Tipo Tabla se guardan ahí
- Cursores – Loops
- Join
- Agregaciones
- Order By
- Group By

DBCC DROPCLEANBUFFERS nos permite limpiar el cache de datos



Tablas temporales y Variables tipo Tabla

Los objetos de tipo tabla, permiten recolectar información estadística, versus las variables.
Las estadísticas, le permiten al SQL Server recolectar dicha información para la elaboración de los planes de ejecución.

```
-----------------------------------------------------
-- Lecturas Tabla Temporal
-----------------------------------------------------
select
  DB_NAME(mf.database_id) ,sum(fs.num_of_reads) as total_reads
from sys.master_files mf
cross apply sys.dm_io_virtual_file_stats(mf.database_id,NULL) fs
where mf.database_id = 2 and mf.type_desc = 'ROWS' group by mf.database_id
GO
CREATE TABLE #temp1 (col1 int)
insert into #temp1 select database_id from sys.databases
DBCC DROPCLEANBUFFERS
select count(*) from #temp1
DROP TABLE #temp1
GO
select DB_NAME(mf.database_id) ,sum(fs.num_of_reads) as total_reads
from sys.master_files mf
cross apply sys.dm_io_virtual_file_stats(mf.database_id,NULL) fs
where mf.database_id = 2 and mf.type_desc = 'ROWS' group by mf.database_id
GO
```

(No column name)	total_reads
tempdb	85

(No column name)
8

(No column name)	total_reads
tempdb	93

Tablas temporales y Variables tipo Tabla

```
-------------------------------------------------------------
-- Lecturas de Variable
-------------------------------------------------------------
select
  DB_NAME(mf.database_id),sum(fs.num_of_reads) as total_reads
from sys.master_files mf
cross apply sys.dm_io_virtual_file_stats(mf.database_id,NULL) fs
where mf.database_id = 2 and mf.type_desc = 'ROWS' group by mf.database_id
GO
DECLARE @temp1 TABLE (col1 int)
insert into @temp1 select database_id from sys.databases
DBCC DROPCLEANBUFFERS
select count(*) from @temp1
GO
select
  DB_NAME(mf.database_id) ,sum(fs.num_of_reads) as total_reads
from sys.master_files mf
cross apply sys.dm_io_virtual_file_stats(mf.database_id,NULL) fs
where mf.database_id = 2 and mf.type_desc = 'ROWS' group by mf.database_id
GO
```

	(No column name)	total_reads
1	tempdb	71

	(No column name)
1	8

	(No column name)	total_reads
1	tempdb	79

Tablas temporales y Variables tipo Tabla

```
-- creamos tabla temporal 1 y la llenamos
select object_id, name into #tmp1 from sys.all_objects

-- creamos tabla temporal 2 y la llenamos
select object_id, column_id, name into #tmp2 from sys.all_columns

SET STATISTICS IO ON

-- obtenemos los datos
select t1.name, count(t2.column_id) from #tmp1 t1
left join #tmp2 t2 on t1.object_id = t2.object_id
group by t1.name
SET STATISTICS IO OFF
```

Observemos la cantidad de lecturas ***realizada*** para ejecutar la consulta es la siguiente:

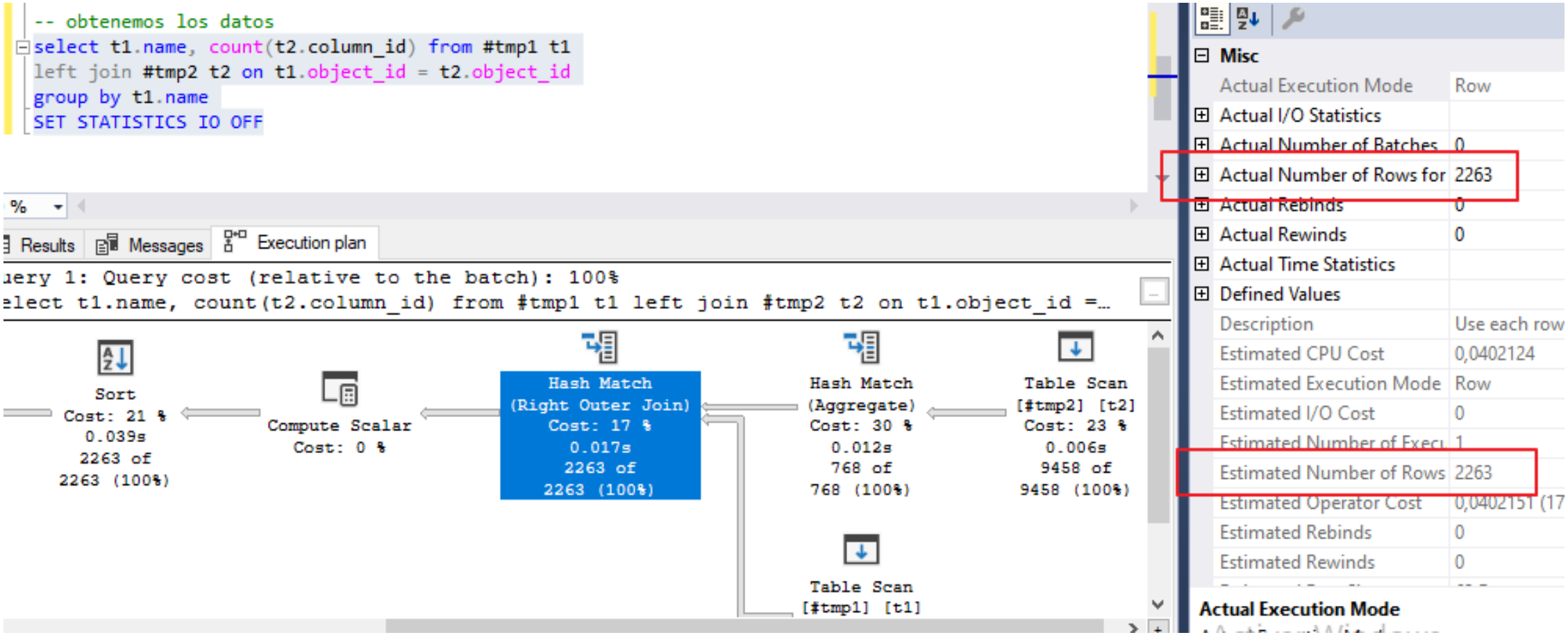
```
(2257 rows affected)
Tabla 'Worktable'. Recuento de exámenes 0, lecturas lógicas 0, lecturas físicas 0, lecturas anticipada
Tabla 'Workfile'. Recuento de exámenes 0, lecturas lógicas 0, lecturas físicas 0, lecturas anticipadas
Tabla '#tmp1__0000000000A7'. Recuento de exámenes 1, lecturas lógicas 18, lecturas físicas 0, lecturas
Tabla '#tmp2_0000000000A8'. Recuento de exámenes 1, lecturas lógicas 57, lecturas físicas 0, lecturas
```

Estas lecturas, se obtiene debido a que como las tablas temporales guardan estadísticas, el motor sabía que para obtener los datos eficientemente debía ejecutar el ***join*** utilizando un operador “Hash Match”.

Tablas temporales y Variables tipo Tabla

Revisemos el plan de ejecución:

```
-- obtenemos los datos
select t1.name, count(t2.column_id) from #tmp1 t1
left join #tmp2 t2 on t1.object_id = t2.object_id
group by t1.name
SET STATISTICS IO OFF
```



Tablas temporales y Variables tipo Tabla

Ahora comparemos contra variables de tipo tabla

```
-- creamos las variables tipo tabla
DECLARE @tmp1 TABLE (object_id int, name sysname)
DECLARE @tmp2 TABLE (object_id int, column_id int, name sysname)

-- insertamos la informacion en las variables tipo tabla
insert into @tmp1 select object_id, name from sys.all_objects
insert into @tmp2 select object_id, column_id, name from
sys.all_columns

-- obtenemos los datos
SET STATISTICS IO ON
select t1.name, count(t2.column_id) from @tmp1 t1
left join @tmp2 t2 on t1.object_id = t2.object_id
group by t1.name
SET STATISTICS IO OFF
```

```
(2257 rows affected)
Tabla 'Worktable'. Recuento de exámenes 0, lecturas lógicas 0, lecturas físicas 0, lecturas an
Tabla '#AE85FD7D'. Recuento de exámenes 1, lecturas lógicas 128991, lecturas físicas 0, lectur
Tabla '#AD91D944'. Recuento de exámenes 1, lecturas lógicas 18, lecturas físicas 0, lecturas a
```

Técnicas de Filtrado

Fases del proceso lógico de un Query

Esta sección cubre el procesamiento de consultas lógicas y las fases involucradas. La declaración principal usado para recuperar datos en T-SQL es la instrucción SELECT.
Las siguientes son las principales cláusulas de consulta especificado en el orden en que se supone que debe escribirlos (conocido como "***keyed-in order***"):

1. **SELECT**
2. **FROM**
3. **WHERE**
4. **GROUP BY**
5. **HAVING**
6. **ORDER BY**

Filtrar datos es uno de los aspectos más fundamentales de las consultas T-SQL. Casi todas las consultas.
que escribes implica alguna forma de filtrado

Técnicas de Filtrado

Consideraciones

```
select * from [dbo].[Orders] where ShipCountry = N'Germany'
```

- Utilice la letra N como prefijo, para denotar un literal de cadena de caracteres Unicode, cuando la columna es NVARCHAR.
  Si la columna de país fuera de un tipo de datos de cadena de caracteres normal, como VARCHAR, el literal no es necesario
- Cuando los valores NULL no son posibles en los datos que está filtrando, T-SQL usa lógica de dos valores; para cualquier fila dada el predicado puede evaluar ya sea verdadero o falso.

Técnicas de Filtrado

Consideraciones

La opción OFFSET-FETCH es una opción de filtrado que, como TOP, puede usar para filtrar datos basados en un número especificado de filas y ordenamiento. Pero a diferencia de TOP, es estándar y también tiene un capacidad de omisión, haciéndola útil para propósitos de paginación ad-hoc.

Las cláusulas OFFSET y FETCH aparecen justo después de la cláusula **ORDER BY** y, de hecho, en T-SQL, requieren una cláusula ORDER BY para estar presente. Primero especifica la cláusula OFFSET indicando cuántas filas desea omitir (0 si no desea omitir ninguna); luego, opcionalmente, puede especificar la cláusula FETCH que indica cuántas filas desea filtrar.

Por ejemplo, la siguiente consulta define el orden en función de la fecha del pedido descendente, seguido del ID del pedido descendente luego omite las primeras 50 filas y recupera las siguientes 25 filas:

```
SELECT [SalesOrderID], [OrderDate], [SalesPersonID], [CustomerID]
FROM [Sales].[SalesOrderHeader]
ORDER BY [OrderDate] DESC, [SalesOrderID] DESC
OFFSET 50 ROWS FETCH NEXT 25 ROWS ONLY;
```

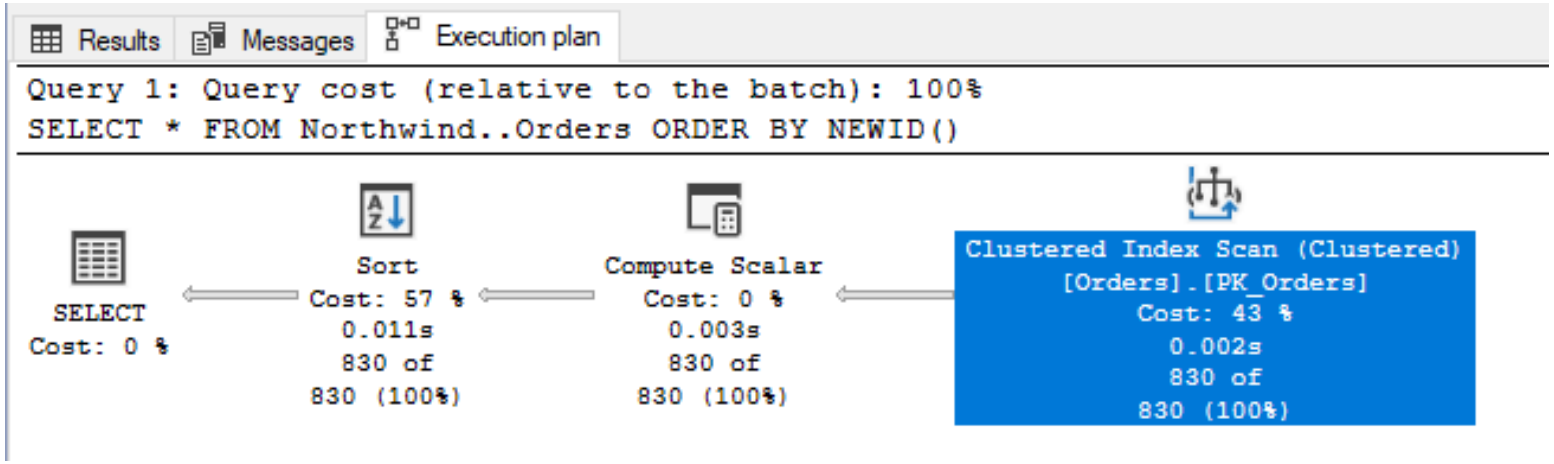
Ordenamiento de Registros

Mostraremos dos formas de escribir una consulta T-SQL SELECT que devuelva filas ordenadas por un número aleatorio.
Comencemos creando una tabla usando la declaración T-SQL CREATE TABLE a continuación.

```
SELECT *
FROM Northwind..Orders
ORDER BY NEWID()
```

NEWID() Crea un valor único del tipo **uniqueidentifier**.

Con un volumen de datos serio, esto puede volverse costoso rápidamente. Mire este plan de ejecución típico y observe cómo el ordenamiento toma el 57% de su tiempo ...



Ordenamiento de Registros

Supongamos entonces:

- Tabla A: tiene 50,000 filas en 2500 páginas de datos. La consulta aleatoria genera 145 lecturas en 42 ms.
- Tabla B: tiene 1.2 millones de filas en 114,000 páginas de datos. Running Order By newid () en esta tabla genera 53,700 lecturas y toma 16 segundos.

Uso ***TABLESAMPLE***

```
SELECT Top (20) *
FROM Northwind..Orders TABLESAMPLE(20 PERCENT)
ORDER BY NEWID()
```

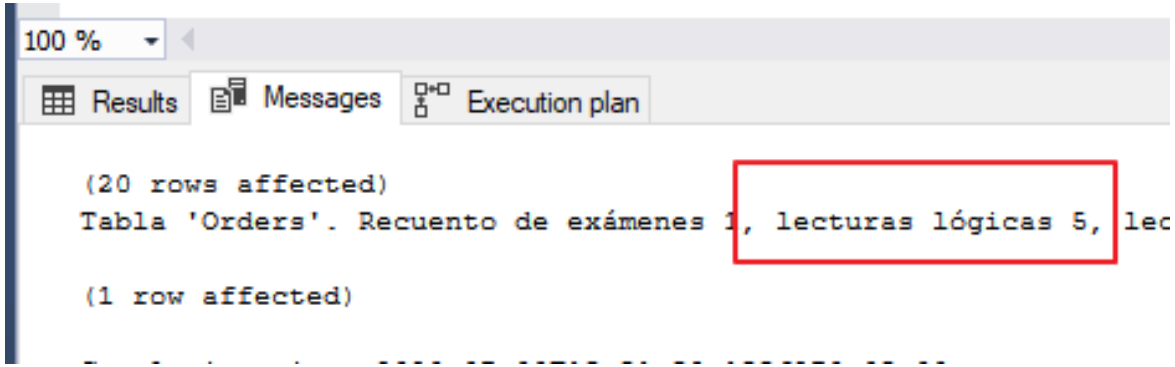
La idea detrás de la muestra de la tabla es darle aproximadamente el tamaño del subconjunto que solicita. SQL numera cada página de datos y selecciona el X por ciento de esas páginas. El número real de filas que obtiene puede variar según lo que exista en las páginas seleccionadas.

```
...FROM tableName TABLESAMPLE (10 PERCENT)
...FROM tableName TABLESAMPLE (1000 ROWS)
```

Ordenamiento de Registros

Personalmente, lo he estado usando para limitar el tamaño de la tabla. Entonces, en esa tabla de millones de filas que está en la parte superior (20) ... TABLESAMPLE (20 POR CIENTO) la consulta cae a 5600 lecturas en 1600 ms.

```
SET STATISTICS IO ON
SELECT Top (20) *
FROM Northwind..Orders
TABLESAMPLE(20 PERCENT)
ORDER BY NEWID()
SET STATISTICS IO OFF
```




```
SET STATISTICS IO ON
SELECT Top (20) *
FROM Northwind..Orders
ORDER BY NEWID()
SET STATISTICS IO OFF
```

```
(20 rows affected)
Tabla 'Orders'. Recuento de exámenes 1, lecturas lógicas 22, lec

(1 row affected)
```

REFERENCIAS

Acceso TempDB
https://docs.microsoft.com/es-es/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15

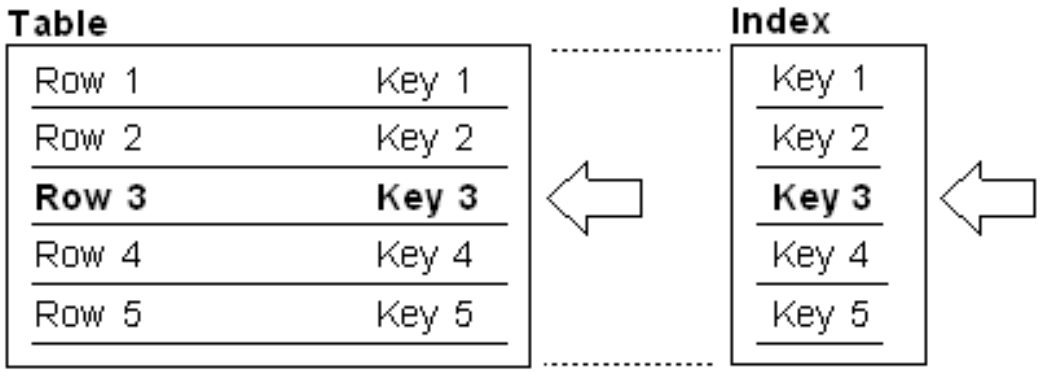
NewID()
https://docs.microsoft.com/en-us/sql/t-sql/functions/newid-transact-sql?view=sql-server-ver15

Tipos de Índices

- Los índices son objetos de base de datos que puede crear para mejorar el rendimiento de algunas consultas.
- El servidor también puede crear índices automáticamente cuando crea una clave primaria o una restricción única

Considere:

- Los índices mal diseñados y la falta de índices son las principales demoras de la
- base de datos.
- Un índice bien diseñado consigue un buen rendimiento de una base de dato.

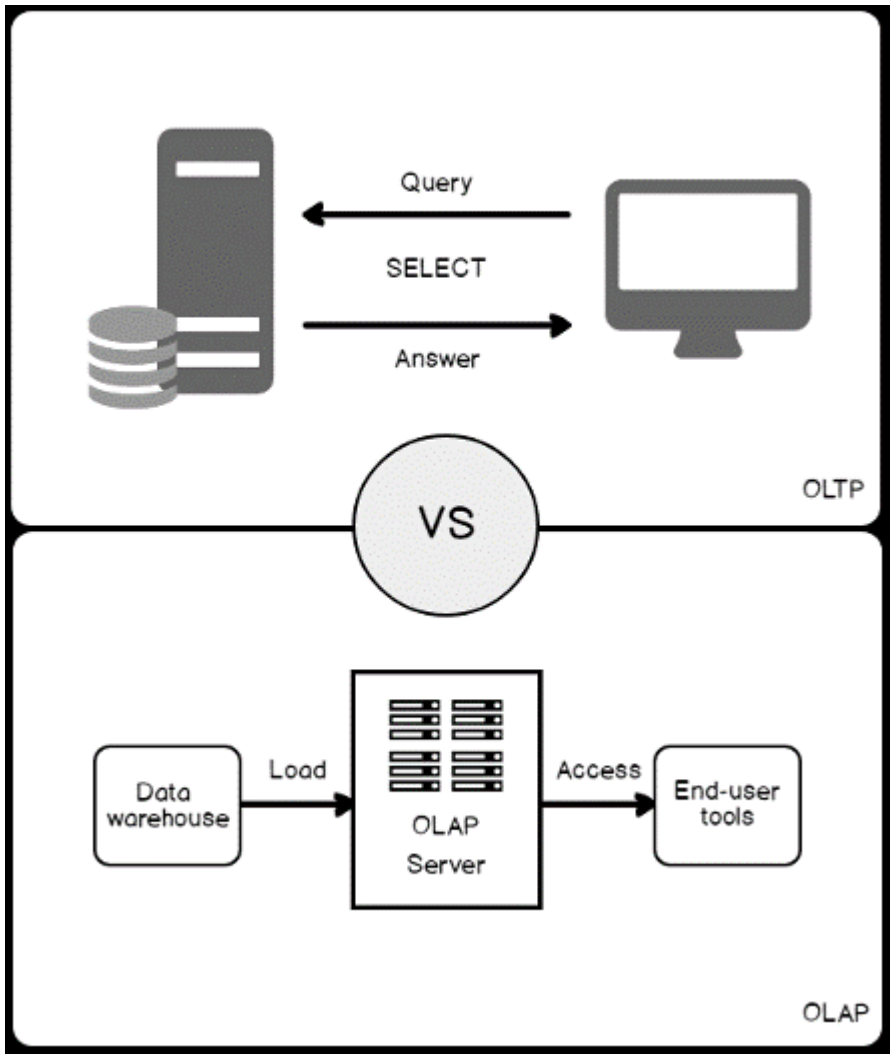


Tipos de Índices

Antes de crear un índice:

Identifique el tipo de carga de trabajo de la base de datos.

- En la base de datos del Procesamiento Transaccional Online (OLTP), las cargas de trabajo son usadas para sistemas transaccionales en los cuales, la mayoría de las consultas presentadas son consultas de modificación de información.
- En cambio, las cargas de trabajo de la base de datos del Procesamiento Analítico Online (OLAP) son usadas para sistemas de almacenamiento de información, en los cuales la mayoría de las consultas presentadas son consultas de recuperación de información que filtra, agrupa, agrega y une grupos grandes de información rápidamente. La diferencia entre las bases de dato OLTP y el OLAP



Tipos de Índices

Tipo de índice	Descripción
Hash	Con un índice hash, se accede a los datos a través de una tabla hash en memoria. Los índices hash utilizan una cantidad fija de memoria, que es una función del número de cubos.
Índice no agrupado optimizado para memoria	Para los índices no clúster optimizados para memoria, el consumo de memoria depende del número de filas y del tamaño de las columnas de clave de índice.
Clúster	Un índice clúster ordena y almacena las filas de datos de la tabla o vista por orden en función de la clave del índice clúster. El índice clúster se implementa como una estructura de árbol b que admite la recuperación rápida de las filas a partir de los valores de las claves del índice clúster.
No agrupado	Los índices no clúster se pueden definir en una tabla o vista con un índice clúster o en un montón. Cada fila del índice no clúster contiene un valor de clave no agrupada y un localizador de fila. Este localizador apunta a la fila de datos del índice clúster o el montón que contiene el valor de clave. Las filas del índice se almacenan en el mismo orden que los valores de la clave del índice, pero no se garantiza que las filas de datos estén en un determinado orden a menos que se cree un índice clúster en la tabla.

Tipos de Índices

Tipo de índice	Descripción
Único	Un índice único se asegura de que la clave de índice no contenga valores duplicados y, por tanto, cada fila de la tabla o vista sea en cierta forma única. La unicidad puede ser una propiedad tanto de índices clúster como de índices no clúster.
columnstore	El índice de almacén de columnas en memoria almacena y administra los datos mediante el almacenamiento de datos basado en columnas y el procesamiento de consultas basado en columnas. Los índices de almacén de columnas funcionan correctamente para las cargas de trabajo de almacenamiento de datos que ejecutan principalmente cargas masivas y consultas de solo lectura. Use el índice de almacén de columnas para aumentar hasta en diez veces el rendimiento de las consultas en relación con el almacenamiento tradicional orientado a filas, y hasta en siete veces la compresión de datos en relación con el tamaño de los datos sin comprimir.
Índice con columnas incluidas	Índice no clúster que se extiende para incluir columnas sin clave además de las columnas de clave.

Tipos de Índices

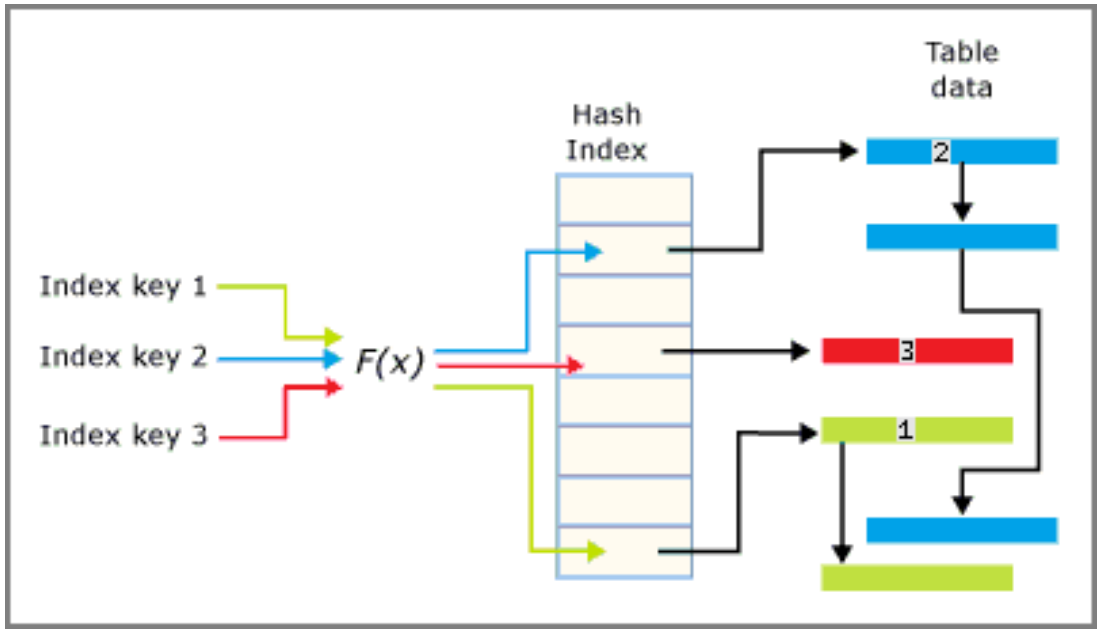
Tipo de índice	Descripción
Índice en columnas calculadas	Índice de una columna que se deriva del valor de una o varias columnas, o algunas entradas deterministas.
Filtered	Índice no clúster optimizado, especialmente indicado para cubrir consultas que seleccionan de un subconjunto bien definido de datos. Utiliza un predicado de filtro para indizar una parte de las filas de la tabla. Un índice filtrado bien diseñado puede mejorar el rendimiento de las consultas y reducir los costos de almacenamiento del índice en relación con los índices de tabla completa, así como los costos de mantenimiento.
Índice con columnas incluidas	Índice no clúster que se extiende para incluir columnas sin clave además de las columnas de clave.

Tipos de Índices

Índices HASH

Consideraciones de rendimiento

- Evite agregar columnas que no sean necesarias.
- Disminuye la cantidad de filas que caben en una página.
- Se necesita más espacio en disco para almacenar el índice.
- Evite usar tipos de datos varchar(max) , nvarchar(max) , varbinary(max) o xml.
- Considere que el índice puede aumentar el tiempo para realizar operaciones de Insert, Update, Delete.



Evalué entonces:

Si la mejora del rendimiento esta compensada en la actualización de los datos de la tabla y en los requisitos de espacio en disco adicionales.

Tipos de Índices

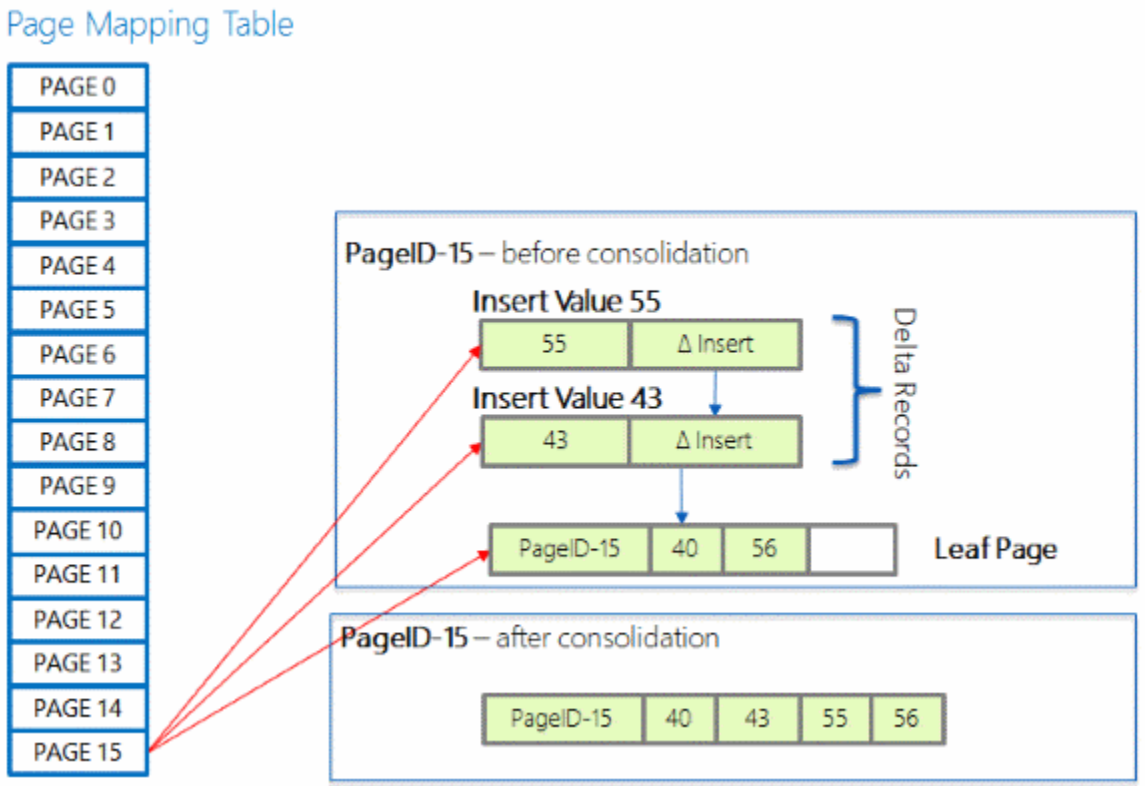
Índice No Agrupado

Consideraciones de rendimiento

Al consultar una tabla optimizada para memoria con predicados de desigualdad, el rendimiento de los índices no agrupados es superior al de los índices de hash no agrupados.

Recomendaciones

- Las consultas tienen una cláusula ORDER BY en la columna indexada.
- Las consultas en las que solo se comprueban las primeras columnas de un índice con varias columnas.
- Las consultas prueban la columna indexada mediante el uso de una cláusula WHERE con:
  1. Una desigualdad: WHERE StatusCode != 'Done'
  2. Un examen de intervalo de valores: WHERE Quantity >= 100



Tipos de Índices

Índice Cluster

Implementación

Restricciones PRIMARY KEY y UNIQUE
Cuando se crea una restricción PRIMARY KEY, se crea automáticamente un índice clúster único en las columnas si aún no existe un índice clúster en la tabla o no se ha especificado un índice no clúster. La columna de clave principal no puede permitir valores NULL.
Cuando cree una restricción UNIQUE, se creará un índice no clúster único para exigir una restricción UNIQUE de forma predeterminada. Puede especificarse un índice clúster único si todavía no existe un índice clúster en la tabla. Un índice creado como parte de la restricción recibe automáticamente el mismo nombre que la restricción. Para obtener más información

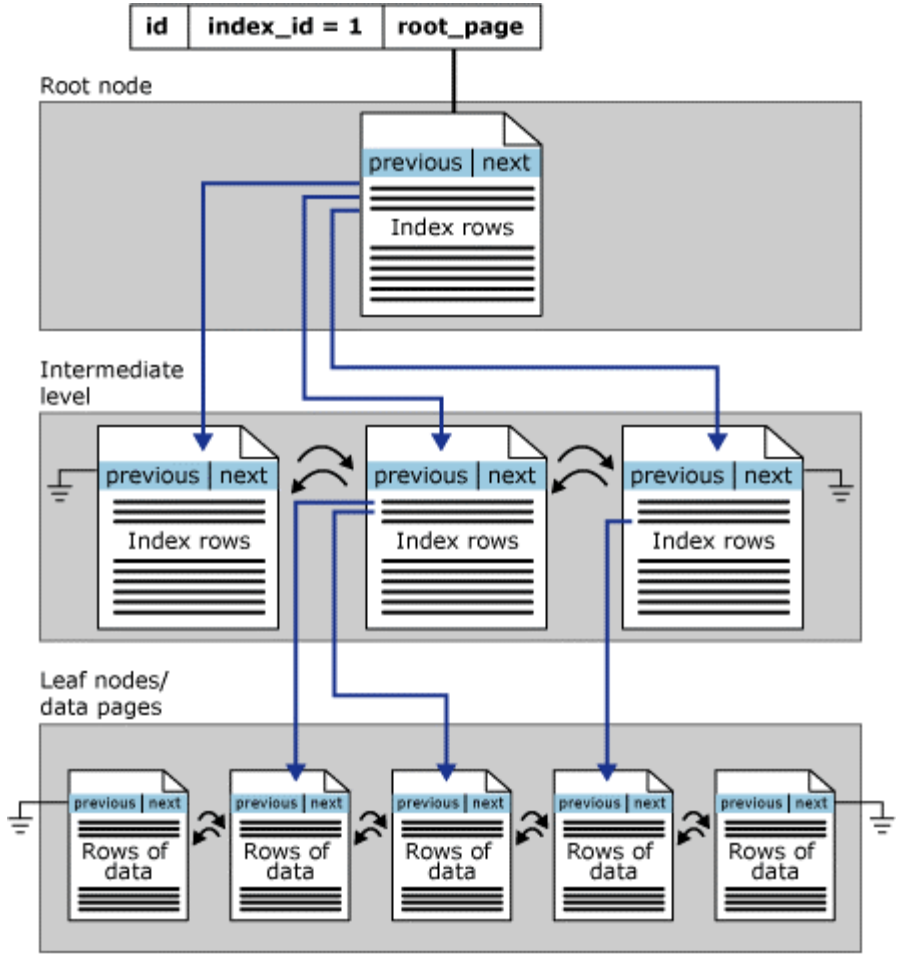
Índice independiente de una restricción
Puede crear un índice clúster en una columna que no sea la de clave principal si se especificó una restricción de clave principal no agrupada.

Tipos de Índices

Índice Cluster

Consideraciones sobre consultas

- Devuelven un intervalo de valores mediante el uso de operadores como BETWEEN, >, >=, < y <=.
- Devuelven grandes conjuntos de resultados.
- Usan cláusulas JOIN; por lo general, son columnas de clave externa.
- Usan cláusulas ORDER BY o GROUP BY.
- Por regla genera, debe definir la clave de índice clúster con el menor número de columnas posible.

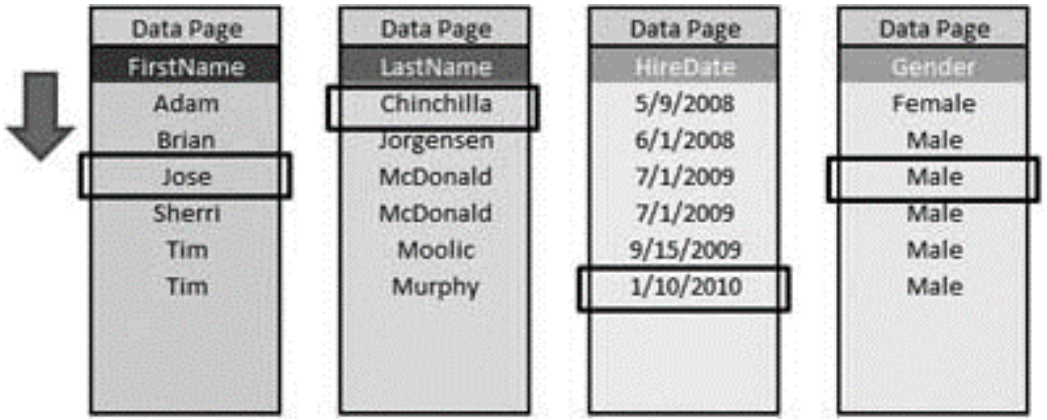


Tipos de Índices

Índices de almacén de columnas

Recomendaciones:

- Use el índice para almacenar tablas de hechos y tablas de dimensiones grandes.
- Este método mejora el rendimiento de las consultas. Para más información, consulte Índices de almacén de columnas para el almacenamiento de datos.
- Use un índice de almacén de columnas no agrupado para realizar análisis en tiempo real en una carga de trabajo OLTP



Los datos de la tabla se almacenan en páginas separadas para cada columnas

Tipos de Índices

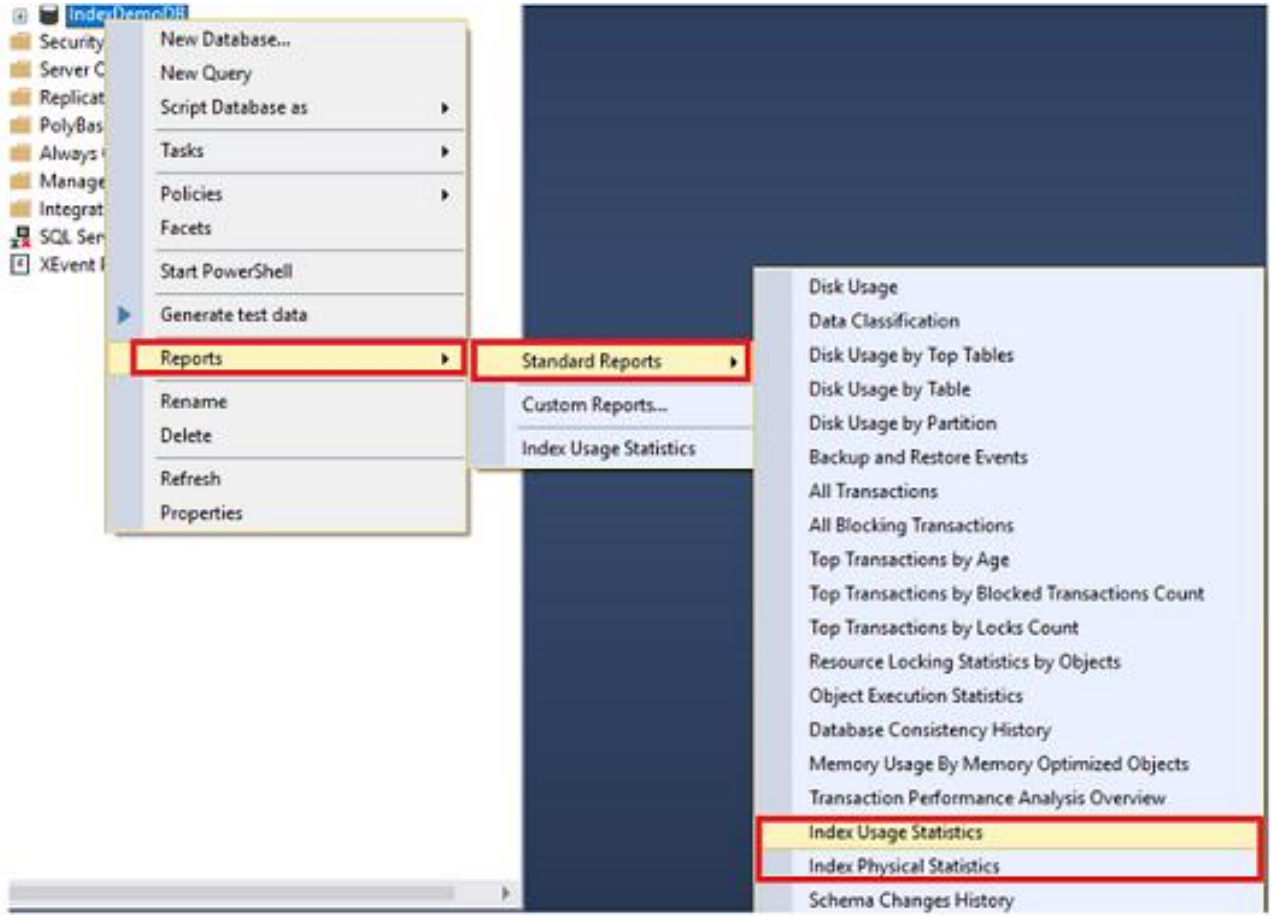
Revisando el uso de los índices

```
SELECT OBJECT_NAME(IX.OBJECT_ID) Table_Name
      ,IX.name AS Index_Name
      ,IX.type_desc Index_Type
      ,SUM(PS.[used_page_count]) * 8 IndexSizeKB
      ,IXUS.user_seeks AS NumOfSeeks
      ,IXUS.user_scans AS NumOfScans
      ,IXUS.user_lookups AS NumOfLookups
      ,IXUS.user_updates AS NumOfUpdates
      ,IXUS.last_user_seek AS LastSeek
      ,IXUS.last_user_scan AS LastScan
      ,IXUS.last_user_lookup AS LastLookup
      ,IXUS.last_user_update AS LastUpdate
FROM sys.indexes IX
INNER JOIN sys.dm_db_index_usage_stats IXUS ON IXUS.index_id = IX.index_id AND IXUS.OBJECT_ID =
IX.OBJECT_ID
INNER JOIN sys.dm_db_partition_stats PS on PS.object_id=IX.object_id
WHERE OBJECTPROPERTY(IX.OBJECT_ID,'IsUserTable') = 1
GROUP BY OBJECT_NAME(IX.OBJECT_ID) ,IX.name ,IX.type_desc ,IXUS.user_seeks ,IXUS.user_scans
,IXUS.user_lookups,IXUS.user_updates ,IXUS.last_user_seek ,IXUS.last_user_scan ,IXUS.last_user_lookup
,IXUS.last_user_update
```

El número de ***Búsquedas*** es el número de veces que el índice es usado, el número de ***Escaneos*** muestra el número de veces que las páginas hoja en el índice son escaneadas, el número de ***Consultas*** indica el número de veces que un índice Agrupado es usado por el índice No Agrupado para buscar la fila entera y el número de ***Actualizaciones*** muestra el número de veces que la información del índice es modificada.

Tipos de Índices

El SQL Server nos provee de dos reportes incorporados que nos ayudan a monitorear la base de datos de fragmentación de índices y el uso de estadísticas, el Index Usage Statistics y el Index Physical Statistics. Estos reportes standard usan los DMOs previamente descritos, y la información de los reportes será actualizada cuando el servicio de SQL Server es reiniciado. Ambos reportes pueden ser vistos haciendo clic derecho en la base de datos, de la cual necesitas monitorear sus índices, escoge Reports -> Standard Reports y selecciona el reporte Index Usage Statistics o Index Physical Statistics, como se muestra abajo:



Tipos de Índices

Revisando el uso de los índices

- Todos los valores cero significa que la tabla no es usada, o el servicio SQL Server reinició recientemente.
- Un índice con cero o pequeño número de búsquedas, escaneos o consultas y un gran número de actualización es un índice inútil y debe ser removido, después de verificar con el propietario del sistema, ya que el principal objetivo de añadir el índice es hacer más rápidas las operaciones de lectura.
- Un índice que es escaneado excesivamente con cero o pequeño número de búsquedas significa que el índice es mal usado y debería ser remplazado por uno más óptimo.
- Un índice con un gran número de consultas significa que necesitamos optimizar el índice al añadir las columnas frecuentemente consultadas a las columnas de índice no-clave existentes usando la cláusula INCLUDE.
- Una tabla con un gran número de Escaneos indica que las consultas SELECT * son excesivamente usadas, recuperando más columnas de lo que es requerido, o las estadísticas de índice deben ser actualizadas.
- Un índice agrupado con un gran número de Escaneos significa que un nuevo índice No agrupado debería ser creado para cubrir las consultas no cubiertas.
- Las fechas con valores NULL significan que esta acción no ha ocurrido todavía.
- Grandes escaneos están BIEN en pequeñas tablas
- Si tu índice no está aquí, entonces ninguna acción es realizada en ese índice todavía.

Mantenimiento y Fragmentación

Identifiquemos las tablas de mayor tamaño y sus lecturas

```
SELECT OBJECT_NAME(IDX.object_id)  Table_Name
      , IDX.name  Index_name
    , PAR.rows  NumOfRows
    , IDX.type_desc  TypeOfIndex
FROM sys.partitions PAR
INNER JOIN sys.indexes IDX ON PAR.object_id = IDX.object_id  AND
PAR.index_id = IDX.index_id AND IDX.type = 0
INNER JOIN sys.tables TBL
ON TBL.object_id = IDX.object_id and TBL.type ='U'
```

Table_Name	Index_name	NumOfRows	TypeOfIndex
DatabaseLog	NULL	1597	HEAP
ProductProductPhoto	NULL	504	HEAP
testing_table	NULL	1	HEAP

Mantenimiento y Fragmentación

Cuando ocurre la fragmentación?

- Las operaciones de ***Insert, Update o Delete*** efectúan cambios sobre las columnas de las tablas.
- Estos cambios deben ser replicados a los índices relacionados.
- Con el tiempo y como resultando de muchas operaciones de inserción, actualización y eliminación, los índices se volverán fragmentados
- Se requiere entonces un número mas grande de páginas, degradando el rendimiento de las consultas
- Como resultado el índice puede ser ignorado en la mayoría de los casos por el Optimizador de Consultas SQL Server.
- Cuando ocurre una división de la página, debido a que los valores insertados o actualizados nuevamente no entren en el espacio disponible de la página

Mantenimiento y Fragmentación

Listado de Índices y su fragmentación

```
SELECT  OBJECT_NAME(IDX.OBJECT_ID) AS Table_Name,
IDX.name AS Index_Name,
IDXPS.index_type_desc AS Index_Type,
IDXPS.avg_fragmentation_in_percent  Fragmentation_Percentage
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, NULL) IDXPS
INNER JOIN sys.indexes IDX  ON IDX.object_id = IDXPS.object_id
AND IDX.index_id = IDXPS.index_id
ORDER BY Fragmentation_Percentage DESC
```

Table_Name	Index_Name	Index_Type	Fragmentation_Perc
vStateProvinceCountryRegion	IX_vStateProvinceCountryRegion	CLUSTERED INDEX	50
Product	AK_Product_rowguid	NONCLUSTERED INDEX	50
Product	AK_Product_ProductNumber	NONCLUSTERED INDEX	50
SpecialOfferProduct	AK_SpecialOfferProduct_rowguid	NONCLUSTERED INDEX	50

REFERENCIAS

Tipos de Índices
https://docs.microsoft.com/es-es/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver15

Requisitos de almacenamiento
https://docs.microsoft.com/es-es/sql/relational-databases/indexes/disk-space-requirements-for-index-ddl-operations?view=sql-server-ver15

Operaciones en línea
https://docs.microsoft.com/es-es/sql/relational-databases/indexes/perform-index-operations-online?view=sql-server-ver15

Specify Fill Factor for an Index
https://docs.microsoft.com/en-us/sql/relational-databases/indexes/specify-fill-factor-for-an-index?view=sql-server-ver15

DIIC Perú | CAPACITACIÓN **PROFESIONAL**

Online | DIIC Perú