Los índices

Un índice es un conjunto de valores claves y apuntadores lógicos que permite ejecutar búsqueda de registros de modo similar a la manera como buscamos un tema en el índice analítico de un libro. Por lo general, todas las consultas se ejecutan más ràpido cuando se utilizan índices. En este capítulo veremos por qué y cómo crear índices.

Esta página se ha dejado en blanco intencionalmente.

Capítulo 11
Los índices

Contenido

- *Indices*
  - ✓ *¿Cómo se almacenan los datos?*
  - ✓ *¿Cómo se accede a los datos?*
- *Criterios para crear índices*
  - ✓ *Razones para crear índices*
  - ✓ *Razones para no crear índices*
  - ✓ *Columnas a indexar*
  - ✓ *Columnas que no deben indexarse*
- *Tipos de índices*
  - ✓ *Indice clustered*
  - ✓ *Indice nonclustered*
- *¿Cómo se accede a los datos mediante los índices?*
  - ✓ *Proceso de búsqueda indexada*
- *Creación de índices – La instrucción CREATE INDEX*
  - ✓ ***Ejercicio 111:*** *Creación de índice clustered*
  - ✓ ***Ejercicio 112:*** *Creación de índice nonclustered – Uso de sp_helpindex*
  - ✓ *Criterios al crear índices UNIQUE*
  - ✓ *¿Cómo determinar si una columna tiene valores duplicados?*
  - ✓ *Creación de índices compuestos*
  - ✓ *Obtención de la deinición de un índice*
- *Eliminación de índices – Uso de DROP INDEX*
  - ✓ ***Ejercicio 113:*** *Uso de DROP INDEX*

- *Las opciones FILLFACTOR y PAD_INDEX*
  - ✓ ***Ejercicio 114:*** *Uso de FILLFACTOR*
  - ✓ *Determinación de la fragmentación de una tabla ó índice*
  - ✓ ***Ejercicio 115:*** *Uso de DBCC SHOWCONTIG*
  - ✓ *Obtención del ID de una tabla – La función OBJECT_ID()*
  - ✓ *Obtención del ID de un índice*
  - ✓ ***Ejercicio 116:*** *IDs de tabla é índice*
  - ✓ *Eliminación de la fragmentación*
  - ✓ ***Ejercicio 117:*** *Reconstrucción de índice*
- *Query Optimizer – El Optimizador de consultas*
  - ✓ *Análisis de los índices*
    - *SET SHOWPLAN_ALL*
    - *SET SHOWPLAN_TEXT*
    - *SET STATISTICS IO*

Los índices

Un índice es un conjunto de valores claves y apuntadores lógicos que permite ejecutar búsqueda de registros de modo similar a la manera como buscamos un tema en el índice analítico de un libro. Por lo general, todas las consultas se ejecutan más ràpido cuando se utilizan índices. En este capítulo veremos por qué y cómo crear índices.

Indices

Cuando la base de datos no tiene índices, SQL Server localiza los datos leyendo cada página de datos en cada tabla que especifica en su instrucción SQL. Este “examen de la tabla” (leer cada página de datos) puede ser un excelente método para la recuperación de datos. Por ejemplo, si una tabla es pequeña, o si esta accesando una porción grande de ella, una exploración podría ser muy bien el mejor plan para acceder a los datos; sin embargo, con mucha frecuencia esto es mucho mas rápido con un índice. Además, al utilizar índices puede acelerar las combinaciones (joins) entre tablas.

Otra razón para crear un índice es reforzar la identificación de registros de manera única. Tener dos filas idénticas en una tabla no es una condición de error. Sin embargo, es probable que ésta no sea la forma en que la mayoría de la gente quiere almacenar datos. Imagine un sistema que lleva un registro de clientes. Si no puede distinguir a los clientes, podría tener dificultades en conservarlos si les factura de manera incorrecta. Tiene varias opciones para identificar a sus clientes de manera única. Podría asignarle números, combinar sus nombres y fechas de nacimiento, emplear sus números de tarjeta de crédito, o utilizar algún otro valor o conjunto de valores. Sin importar la opción que haya seleccionado, la forma de indicarle a SQL Server su elección es por medio de un índice único, que es un índice que no permite valores duplicados.

¿Cómo se almacenan los datos?

Tenga en cuenta lo siguiente respecto a la forma cómo se almacenan los datos:

- Un **heap** (tabla sin índices o archivo plano) es una colección de páginas de los datos que contienen las filas para una tabla:
- Los datos para cada tabla se guarda en una colección de 8 páginas de datos (cada una con capacidad para 8 KB), que es conocida como un **extent**.
- Las filas de los datos no se guardan en un orden particular, y no hay ningún orden particular para la sucesión de las páginas de los datos.
- Una fila se almacena completamente en una página, y no puede dividirse entre dos páginas.

¿Cómo se accede a los datos?

SQL Server accede a los datos de una de estas dos maneras:

- Examinando todos las páginas de datos de la tabla. Este proceso se conoce como **Table Scan**. Cuando SQL Server realiza un Table Scan, él:
  - Lee desde el principio de la tabla.
  - Examina página a página todas las filas de la tabla.
  - Extrae las filas que corresponden al criterio de la consulta.
- Usando los índices. Este proceso se conoce como **Index Seek**. Cuando SQL Server usa un indice, él:
  - Atraviesa la estructura de árbol del índice para encontrar las filas que la consulta requiere.
  - Extrae solo las filas que se corresponden con el criterio de la consulta.

Cuando se envía una consulta, SQL Server determina primero si un índice existe. Entonces, el Query Optimizer, el componente responsable de generar el plan de ejecución óptimo para la consulta, determina si es mas eficaz examinar la tabla o utilizar el índice para acceder a los datos.

Criterios para crear índices

Si está considerando crear un índice, evalúe dos factores para asegurar que la utilización del índice será más eficaz que la lectura directa de la tabla examinada: la naturaleza de los datos y la naturaleza de las consultas basadas en la tabla.

Razones para crear índices

Los índices aceleran la recuperación de los datos. Por ejemplo, sin ningún índice, tendríamos que recorrer todas las páginas de un libro hasta encontrar la información que estamos buscando.

- Refuerzan la unicidad de las filas.
- Incrementan la velocidad de recuperación de datos:
  - Los joins se ejecutan mas rápido si la columna llave foránea está indexada.
  - Las consultas ORDER BY y GROUP BY se ejecutan mas rápido.

Razones para no crear índices

En general, cuando ejecutamos operaciones de lectura, los índices favorecen el proceso; cuando las operaciones son de escritura, los índices hacen que el rendimiento del sistema disminuya.

- Consumen espacio de disco.
- Producen sobrecarga en el sistema.
  - Cuando se modifican datos de columnas indexadas, el índice es actualizado automáticamente para reflejar los cambios.

Columnas a indexar

- Clave primaria.
- Clave foránea.
- Columnas en las que se busca por rango de valores.
- Columnas que se recuperan ordenadas.

Columnas que no deben indexarse

- Columnas en las que no se ejecuta búsqueda.
- Columnas con pocos valores únicos o que retornan un gran porcentaje de filas.
- Columnas de tipo bit, text o image.

Tipos de índices

Hay dos opciones para el almacenamiento físico de sus índices:

- índice agrupado (**clustered**)
- índice no agrupado (**nonclustered**)

Indice clustered

- Ordena físicamente la tabla. Las filas de la tabla se ordenan según el orden de los valores claves del índice clustered.
- Solo se puede definir un índice clustered por cada tabla.

Indice nonclustered

- Es el tipo de índice por defecto.
- Se reconstruyen automáticamente cuando se crea, se elimina o se redefine el índice clustered.

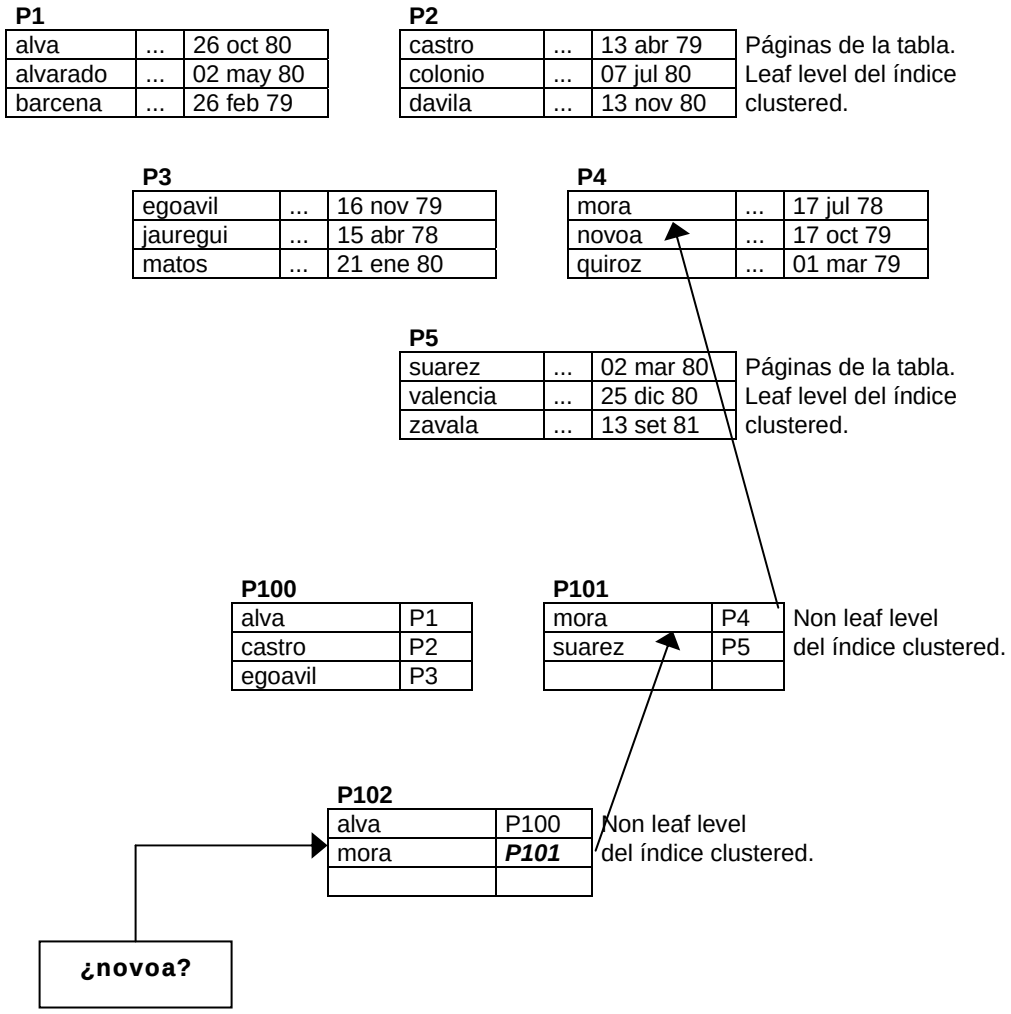
Observación:

- Se pueden definir hasta 249 índices por tabla.
- Siempre crear el índice clustered antes que los índices nonclustered.

¿Cómo se accede a los datos mediante los índices?

Se tiene la tabla **Empleado** con las columnas: **paterno** y **fechaNac** entre otras.

Se ha definido un índice clustered por la columna **paterno**, y un índice nonclustered por la columna **fechaNac**.



Los datos en las tablas se almacenan en páginas, donde cada una de éstas tiene capacidad para 8KB. Por lo tanto, la cantidad de registros que puede contener una página depende de la longitud del registro.

- Un registro debe estar contenido completamente en una página; es decir, que no puede estar dividida entre dos páginas.
- Las páginas de una tabla se organizan en **extents**, conjunto de 8 páginas contiguas, por lo que un extent que está siendo utilizado por una tabla ya no puede ser utilizado por otra.
- Así, como en la base de datos tenemos páginas de tablas, también se tiene páginas de índice.
- Todos los índices, con la excepción del índice clustered, están formados completamente por páginas de índice organizadas en dos niveles: el **leaf level** (ó nivel de las hojas), y el **non leaf level** (ó nivel de las ramas).
- El leaf level está formado por páginas de índice, donde cada página de índice contiene los valores claves del índice ordenados. Por ejemplo, en el caso de un índice por apellido paterno, las páginas del índice contienen todos los apellidos paternos ordenados alfabéticamente; en el caso de un índice por fecha, contienen todas las fechas ordenadas cronológicamente.
- Las páginas de índice del non leaf level almacenan los valores claves iniciales de cada página del nivel inmediato superior, con un puntero que señala a la página correspondiente.
- En el caso especial del índice clustered, como éste ordena físicamente la tabla, no existen páginas de índice en el leaf level de un índice clustered. Los punteros del non leaf level del índice clustered apuntan directamente a las páginas de la tabla, ya que éstas tienen los valores claves ya ordenados.

En el diagrama anterior se muestra un índice clustered por **apellido paterno**.

- Las páginas **P1** a **P5** son páginas de la tabla, y forman el leaf level del índice clustered.
- Cada página del non leaf level está formada por los valores claves iniciales de las páginas del nivel inmediato superior. Asi, la página **P100** está formada por los valores claves **alva**, **castro** y **egoavil** que son los valores iniciales en las páginas **P1**, **P2** y **P3** respectivamente.
- Cuando se busca un valor clave utilizando un índice, la búsqueda se inicia en la raíz del índice.

Proceso de búsqueda indexada

Supongamos que estamos buscando el registro de alguien cuyo apellido es **novoa**. La búsqueda se ejecuta de la siguiente manera:

La búsqueda se inicia en la raíz del índice (**P102**), comparando la cadena buscada **novoa** con los valores claves en la página de índice.

1. Ubicación: Raíz del índice, página de índice **P102**, valor clave **alva**
   Comparación: ¿es **novoa >= alva**?
   Resultado: true
   Acción: ir al siguiente valor clave
2. Ubicación: Raíz del índice, página de índice **P102**, valor clave **mora**
   Comparación: ¿es **novoa >= mora**?
   Resultado: true
   Acción: **mora** es el útimo valor clave de la página, entonces ir a la página a la que apunta el valor clave **mora** (**P101**)
3. Ubicación: Non leaf level del índice, página de índice **P101**, valor clave **mora**
   Comparación: ¿es **novoa >= mora**?
   Resultado: true
   Acción: ir al siguiente valor clave
4. Ubicación: Non leaf level del índice, página de índice **P101**, valor clave **suarez**
   Comparación: ¿es **novoa >= suarez**?
   Resultado: false
   Acción: ir a la página a la que apunta el valor clave anterior **mora** (**P4**)
5. Ubicación: Leaf level del índice, página de tabla **P4**
   Acción: recorrer la página de tabla hasta encontrar el valor buscado **novoa**. Cuando se alcance el valor buscado, detener la búsqueda.

Creación de índices – La instrucción CREATE INDEX

Utilice la sentencia CREATE INDEX para crear los índices. Cuando cree índice en una o más columnas en una tabla, considere lo siguiente:

- SQL Server crea los índices automáticamente cuando define una restricción PRIMARY KEY o UNIQUE en una tabla.
- Debe ser el dueño de la tabla para poder ejecutar la sentencia CREATE íNDEX.
- SQL Server guarda la información de la definición del índice en la tabla de sistema sysindexes.
- Antes de crear un índice en una columna, determine si la columna ya tiene índices.
- Procure que sus índices sean pequeños limitando los valores claves a una o dos columnas. Los indices pequeños son más eficaces que los índices con los valores claves grandes.

Sintáxis

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX
   nombre_índice
ON nombre_tabla( columna(s) )
```

Ejercicio 111: Creación de índice clustered

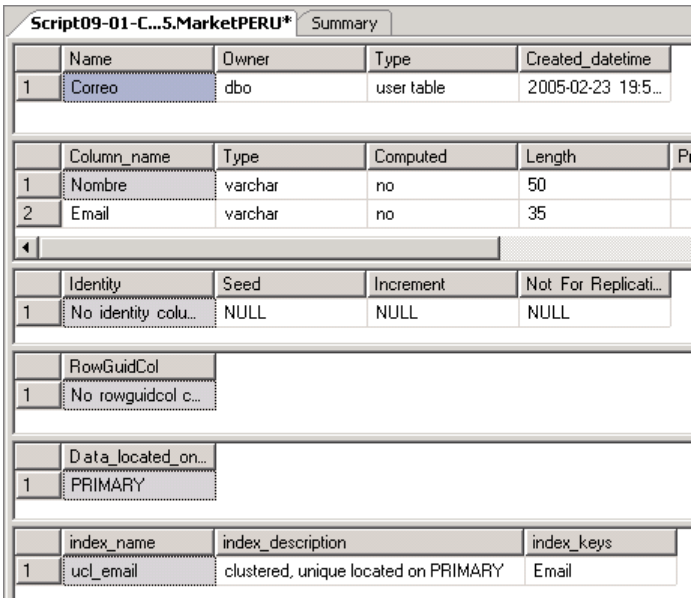
Crear un índice UNIQUE en la columna **email** de la tabla **Correo** para garantizar la unicidad de los valores de la columna.

```
USE MarketPERU
go

CREATE UNIQUE CLUSTERED INDEX ucl_email
ON Correo(email)
go
```

Para verificar puede ejecutar la siguiente instrucción:

```
sp_help Correo
go
```



Script09-01-C...5.MarketPERU* | Summary

	Name	Owner	Type	Created_datetime
1	Correo	dbo	user table	2005-02-23 19:5...

	Column_name	Type	Computed	Length	P
1	Nombre	varchar	no	50	
2	Email	varchar	no	35	

	Identity	Seed	Increment	Not For Replicati...
1	No identity colu...	NULL	NULL	NULL

	RowGuidCol
1	No rowguidcol c...

	Data_located_on...
1	PRIMARY

	index_name	index_description	index_keys
1	ucl_email	clustered, unique located on PRIMARY	Email

Un índice UNIQUE no permite valores duplicados. Si la columna a indexar contiene valores duplicados, la creación del índice falla.

Ejercicio 112: Creación de índice nonclustered - Uso de sp_helpindex

Crear un índice nonclustered formado por los valores de las columnas **fechaSalida** y **transportista** de la tabla **Guia**.

```
CREATE INDEX ncl_fecha_transportista
ON Guia(fechaSalida, transportista)
go
```

Para obtener información sobre los índices de una tabla ejecute la siguiente instrucción.

```
sp_helpindex Guia
go
```

Script09-01-C...5.MarketPERU* | Summary

	index_name	index_description	index_keys
1	ncl_fecha_transportista	nonclustered located on PRIMARY	FechaSalida, Transportista
2	PK__GUIA__07F6335A	clustered, unique, primary key located on PRIMARY	IdGuia

Criterios al crear índices UNIQUE

Genere un índice UNIQUE para un índice clustered o nonclustered cuando el dato es inherentemente único.

Sin embargo, si la singularidad ó unicidad debe reforzarse, cree una restricción PRIMARY KEY o UNIQUE en la columna en lugar de crear un indice UNIQUE. Cuando crea un indice UNIQUE, considere lo siguiente:

- SQL Server crea el índice UNIQUE automáticamente en la columna definida con restricción PRIMARY KEY o UNIQUE.
- Si una tabla ya contiene datos, SQL Server verifica que no existan valores duplicados antes de crear el índice.
- SQL Server verifica los valores duplicados cada vez que utiliza la sentencia INSERT o UPDATE. Si existen valores con claves duplicadas, SQL Server cancela la sentencia y retorna un mensaje del error con el primer duplicado encontrado.
- Asegúrese que cada fila tiene un único valor; dos filas no pueden tener el mismo número de identificación si un índice único se crea en esa columna. Esta regulación asegura que cada ocurrencia de la entidad se identifica singularmente.
- Cree los índices UNIQUE solo en columnas en que la integridad de la entidad puede reforzarse. Por ejemplo, no se nos ocurriría crear un índice UNIQUE en la columna **paterno** de una tabla **Trabajador** porque algunos trabajadores pueden tener el mismo apellido paterno.

¿Cómo determinar si una columna tiene valores duplicados?

Si existen valores duplicados cuando crea un índice UNIQUE, la sentencia CREATE INDEX falla. SQL Server devuelve un mensaje del error con el primer duplicado encontrado, pero otros valores duplicados también pueden existir. Utilice el siguiente formato de consulta para determinar si la columna a indexar de manera única tiene valores duplicados.

```
SELECT columna_a_indexar, COUNT( columna_a_indexar )
FROM nombre_tabla
GROUP BY columna_a_indexar
   HAVING COUNT( columna_a_indexar ) > 1
ORDER BY columna_a_indexar
```

Creación de índices compuestos

Un índice compuesto es aquel cuyos valores claves se forman con datos de dos o mas columnas de la misma tabla. Tenga en cuenta lo siguiente al crear índices compuestos:

- Puede combinar hasta 16 columnas en un índice compuesto. La suma de las longitudes de las columnas que constituyen el índice compuesto no puede exceder de 900 bytes.
- Todas las columnas en un índice compuesto deben ser de la misma tabla.
- Defina la primera columna como única. La primera columna definida en la sentencia CREATE ÍNDEX es referenciada con el orden más alto.
- La cláusula WHERE de una consulta debe referenciar a la primera columna del índice compuesto para que la consulta optimice el uso del índice compuesto.
- Un índice en (*columna1*, *columna2*) no es igual que un índice en (*columna2*, *columna1*). Cada uno tiene un orden de columna distintas. La columna que contiene los datos más selectivos o la que devuelve el porcentaje más bajo de filas determina el orden de las columnas indexadas.
- Utilice los índices compuestos para aumentar el rendimiento de las consultas y reducir el número de índices que crea en una tabla.

Obtención de la definición de un índice

La información sobre índices se guarda en la tabla de sistema **sysindexes**. Para obtener información sobre índices se puede ejecutar

```
sp_helpindex nombre_tabla

sp_help nombre_tabla
```

Eliminación de índices – Uso de DROP INDEX

Use la sentencia DROP INDEX para eliminar un índice en una tabla.

Sintáxis

```
DROP INDEX nombre_tabla.nombre_índice [,...]
```

Cuando elimine un índice, considere los siguiente:

- No puede usar la sentencia DROP INDEX en índices que se crean con las restricciones PRIMARY KEY o UNIQUE. Debe eliminar las restricciones para eliminar estos índices.
- Cuando elimina una tabla, se eliminan todos los índices de esa tabla.
- Cuando elimina un índice clustered, todos los índices nonclustered en la tabla se reconstruyen automáticamente.
- Debe encontrarse en la base de datos en la que un índice reside para poder eliminar el índice.
- La sentencia DROP INDEX no puede ser usada en las tablas del sistema.

Ejercicio 113: Uso de DROP INDEX

La instrucción siguiente elimina el índice **ncl_fecha_transportista** de la tabla **Guia** de la base de datos **MarketPERU**.

```
USE MarketPERU
go

DROP INDEX Guia.ncl_fecha_transportista
go
```

Las opciones FILLFACTOR y PAD_INDEX

La opción FILLFACTOR determina cuán llenas estarán las páginas del índice.

Si las páginas de un índice están llenas, una operación INSERT o UPDATE puede provocar un reacomodo de los datos en muchas páginas del índice y de la tabla. Por ejemplo, en un índice clustered por apellido **paterno**, la inserción de un nueva fila con el valor **avila** para el apellido paterno puede ocasionar que se tengan que reacomodar todas las filas de la tabla si es que las páginas de datos están llenas. Imagínese el efecto de la inserción de 1000 filas en una tabla con esas características.

La especificación de un valor de FILLFACTOR adecuado minimiza el número de páginas afectadas por la operaciones que modifican datos. FILLFACTOR permite especificar el espacio libre en cada página para las operaciones de inserción y modificación de datos. Para el ejemplo anterior, si la página en la que se debe insertar el registro de **avila** tiene espacio suficiente, la inserción afectará solo esa página. Por supuesto que con el tiempo, el espacio disponible en las páginas se va consumiendo, pero siempre tenemos la opción de recosntruir el índice con el valor de FILLFACTOR adecuado.

Al usar la opción FILLFACTOR, considere lo siguiente:

- El rango de FILLFACTOR va de 1 a 100 por ciento.
- El valor predefinido es 0. Este valor indica que las páginas del índice a nivel de las hojas deben llenarse al 100 por ciento.
- El valor predefinido para FILLFACTOR puede cambiarse a nivel del servidor usando el procedimiento almacenado del sistema **sp_configure**.
- La tabla de sistema **sysindexes** guarda el último valor aplicado para FILLFACTOR, junto con otra información del índice.
- El valor de FILLFACTOR se especifica en porcentaje. El valor cuán llenas deben estar las páginas de índice al nivel de las hojas al momento de crear el índice. Por ejemplo, un FILLFACTOR de 65 por ciento indica que debe dejarse 35 por ciento de espacio reservado para nuevas filas. El tamaño de la fila determina cuántas filas pueden encajar en o pueden llenar la página para el porcentaje FILLFACTOR especificado.
- Use la opción FILLFACTOR en tablas en que se insertan muchas filas o cuando en el índice clustered se modifican frecuentemente los valores claves.

- Use un valor bajo de FILLFACTOR para transacciones en línea que se procesan en las aplicaciones OLTP.
- Use un valor de FILLFACTOR alto valor para el almacenamiento de datos de sistemas de apoyo a las toma de decisiones.

Ejercicio 114: Uso de FILLFACTOR

```
CREATE INDEX ncl_fecha_transportista
ON Guia(fechaSalida, transportista)
WITH FILLFACTOR = 70
go
```

Especifica que las páginas del índice **ncl_fecha_transportista** de la tabla **Guia** tendrán 30% de espacio libre para futuras inserciones. El espacio libre se define para el nivel de las hojas. En el nivel de las ramas, el espacio libre será el equivalente a dos entradas del índice.

Si desea que el porcentaje de espacio libre sea el mismo a nivel de hojas y ramas, ejecute:

```
CREATE INDEX ncl_fecha_transportista
ON Guia(fechaSalida, transportista)
WITH FILLFACTOR, PAD_INDEX = 70
go
```

Determinación de la fragmentación de una tabla ó índice

Las tablas sumamente volátiles, es decir, aquellas que sufren gran cantidad de modificaciones y con bastante frecuencia suelen fragmentarse muy rápido, ya que SQL Server busca **extents** libres para almacenar la nueva data.

Para averiguar cuan fragmentada está una tabla ó índice, y si las páginas de datos e índices están llenas utilice la instrucción DBCC SHOWCONTIG.

Sintáxis

```
DBCC SHOWCONTIG( nombre_tabla | id_tabla
       [ , nombre_índice | id_índice ] )
```

Ejercicio 115: Uso de DBCC SHOWCONTIG

```
DBCC SHOWCONTIG(Guia_detalle)
go
```

DBCC SHOWCONTIG entrega un reporte con los siguientes datos:

```
DBCC SHOWCONTIG scanning 'GUIA_DETALLE' table...
Table: 'GUIA_DETALLE' (2073058421); index ID: 1, database ID: 8
TABLE level scan performed.
- Pages Scanned................................: 4
- Extents Scanned..............................: 0
- Extent Switches..............................: 0
- Avg. Pages per Extent........................: 0.0
- Scan Density [Best Count:Actual Count].......: 0.00% [0:0]
- Logical Scan Fragmentation ..................: 25.00%
- Extent Scan Fragmentation ...................: 0.00%
- Avg. Bytes Free per Page.....................: 0.0
- Avg. Page Density (full).....................: 98.13%
DBCC execution completed. If DBCC printed error messages, contact
your system administrator.
```

Extent Switches indica el número de "saltos" de un extent a otro cuando DBCC recorre las páginas de la tabla ó índice. Un valor de 0 (cero) indica que DBCC encontró que los extents de la tabla ó índice son contiguos.

En **Scan Density**, **Best Count** indica el número ideal de "saltos" de extents cuando DBCC recorre las páginas de la tabla ó índice; **Actual Count** indica el número de "saltos" ejecutados durante el recorrido. En una tabla ó índice sin fragmentación, el valor de **Scan Density** es 100% ó 0%. Cuando el valor es menor que 100% ya presenta fragmentación.

Obtención del ID de una tabla – La función OBJECT_ID()

Sintáxis

```
SELECT OBJECT_ID( 'nombre_tabla' )
```

Obtención del ID de un índice

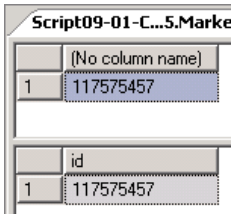
Sintáxis

```
SELECT id FROM sysindexes
WHERE name = 'nombre_índice'
```

Ejercicio 116: IDs de tabla é índice

```
SELECT OBJECT_ID('Guia')
go

SELECT id FROM sysindexes
WHERE name = 'ncl_fecha_transportista'
go
```



Eliminación de la fragmentación

La fragmentación se elimina reconstruyendo el índice clustered con un valor de FILLFACTOR adecuado.

Ejercicio 117: Reconstrucción de índice

```
CREATE UNIQUE CLUSTERED INDEX ucl_email
ON Correo(email)
WITH DROP_EXISTING, FILLFACTOR = 50
go
```

Elimina y reconstruye el índice **ucl_email**. Esta forma de la sentencia CREATE INDEX permite redefinir un índice. Puede redefinir el tipo de índice, las columnas a indexar, y las opciones de indexación.

Query Optimizer – El Optimizador de consultas

Cuando se ejecuta una consulta, SQL Server determina si la columna a analizar está indexada. Si existe un índice, el Query Optimizer revisa las estadísticas del índice para determinar si la consulta se resolverá de acuerdo a uno de los planes siguientes:

- **Table scan**, la consulta lee las páginas de datos de la tabla hasta encontrar los datos solicitados.
- **Index search**, la consulta utiliza el índice para buscar los datos solicitados

La información estadística del índice está formada por la distribución de valores claves de la columna indexada, y permite determinar el grado de selectividad del índice. Es creada automáticamente cuando el índice se crea.

Análisis de los índices

Las siguientes sentencias permiten efectuar un seguimiento de los índices para evaluar su rendimiento.

SET SHOWPLAN_ALL

```
SET SHOWPLAN_ALL ON | OFF
```

Cuando se especifica, SQL Server no ejecuta las instrucciones Transact-SQL. En vez de ello, proporciona información detallada acerca de cómo las instrucciones son ejecutadas, y estima los recursos a utilizar para ejecutarlas.

SET SHOWPLAN_TEXT

```
SET SHOWPLAN_TEXT ON | OFF
```

Cuando se especifica, SQL Server no ejecuta las instrucciones Transact-SQL. En vez de ello, proporciona información detallada acerca de cómo las instrucciones son ejecutadas.

SET STATISTICS IO

```
SET STATISTICS IO ON | OFF
```

Cuando se especifica, SQL Server muestra información acerca de la actividad de disco generada por las instrucciones Transact-SQL.