Primer ciclo 2020

Bases de datos I

09 - Manejo de estructuras de apoyo para consultas

Mgtr. Diana Alejandra Gutiérrez

- Vistas
- Tablas temporales
- Cursores
- CTE (Common table expressions)
- Índices

1

Vistas

Definición, uso y sintaxis

- Es una tabla virtual.
- Su definición surge de una consulta, los resultados que ésta devuelva no se almacenan en la BD. Cualquier consulta en lenguaje de datos estructurado (SQL) puede convertirse en una vista.
- Se consulta y manipula como una tabla física.

Existen dos situaciones donde es útil crear una vista:

- Conveniencia: Cuando se tiene una consulta compleja que se ejecuta de manera frecuente permite que su lectura sea más fácil.
- Seguridad: Cuando un usuario no necesita ver todas las columnas de una tabla y se debe restringir cierta información.

Sintaxis de creación de vista:

```
CREATE VIEW <name> [Columnas]
AS
  Query
GO
```

2

Tablas temporales

Definición, uso y sintaxis

- Tablas que se crean y almacenan de forma temporal en la base de datos de sistema tempdb.
- Sirven como una especie de caché, para poder tener un acceso más rápido de cierta información de la base de datos.
- Aunque pueden ser muy útiles su uso debe ser moderado ya que impactan en los otros recursos del servidor (disco, procesador).

#LOCALES

Solo se pueden utilizar en la conexión del usuario que las crea. Cuando termina la conexión, la tabla desaparece.

##GLOBALES

Son visibles por cualquier usuario conectado al servidor. Desaparece cuando ningún usuario hace referencia a ella.

- Ya sea que se cree una tabla temporal local o global, la buena práctica es eliminarla cuando se termine de usar (DROP TABLE).
- Se manipula igual que una tabla física.

```
Create table #Temp(campo1 int, campo2
varchar(50)

Select * from #Temp
```

3

Cursores

Definición, uso y sintaxis

- Estructura de control que representa un conjunto de datos determinado por una consulta.
- Permite recorrer fila a fila, leer y modificar dicho conjunto de resultados.
- Por ser una estructura temporal, no se almacena formalmente en el servidor; debe crearse, y después de usarlo, eliminar para liberar recursos del servidor.

- Un cursor consta de cinco partes:
  - Declaración
  - Apertura
  - Acceso de datos
  - Cierre
  - Desalojo

```
-- Declaración
DECLARE @Variable  nvarchar(50)  /* Declarar todas las variables requeridas para uso en el cursor */
DECLARE Cursor_Name CURSOR FOR       /* Declare nombre del cursor */
    SQL_Query                          /* Consulta para crear el cursor*/
-- Apertura
OPEN Cursor_Name               /* Abrir el cursor  */
-- Recorrido del cursor y acceso a los datos
FETCH NEXT FROM Cursor_Name INTO @Variable   /*  Recorrer los resultados del cursor, copiando cada fila leída en variables */
    WHILE @@fetch_status = 0 -- Variable de sistema para saber si hay registros aún por leer (0 = lectura correcta)
    BEGIN
        P-SQL_Query             /* Uso de datos del cursor*/
        FETCH NEXT FROM Cursor_Name INTO @Variable   /*  Leer la siguiente fila */
    END
-- Cierre del cursor
CLOSE Cursor_Name              /*  Cerrar el Cursor  */
-- Desalojo del cursor
DEALLOCATE Cursor_Name          /* Liberar recursos y memoria asociados al cursor */
```

4

CTE (Common Table Expressions)

Definición, uso y sintaxis

- Es un conjunto de resultados con nombre temporal al que puede hacer referencia dentro de una instrucción SELECT, INSERT, UPDATE o DELETE. También se la puede usar en una vista.
- La finalidad principal y más común es usarlos para crear subconsultas de una forma más amigable.

Sintaxis:

```
WITH expression_name[(column_name [,...])]
AS
    (CTE_definition)
SQL_statement;
```

- Especificar el nombre de la expresión al que se hace referencia más adelante en una consulta.
- Especificar una lista de columnas separadas por comas. Debe ser el mismo que el número de columnas definidas en el cuerpo del CTE.
- Usar la palabra clave AS después del nombre de la expresión o la lista de columnas.
- Definir una instrucción SELECT cuyo conjunto de resultados complete la expresión de tabla común.
- Consultar la expresión de tabla común en una consulta como SELECT, INSERT, UPDATE, DELETE.

CTE (Common table expressions)

```
with CursoGRPCarne_CTE(Carne, NoCursos)
AS
(
    select carne, count(1) as NoCursos from calificacion
    group by carne
),
Curso_CTE(Idcurso,NoCursos)
AS
(
    select idcurso, count(1) as NoCursos from calificacion
    group by idcurso
)
Select 'Promedio de cursos recibidos por alumnnos' as [Titulo], AVG(CursoGRPCarne_CTE.NoCursos) as 'Numero'
from CursoGRPCarne_CTE
Union all
Select 'Promedio de cursos recibidos' as [Titulo], AVG(Curso_CTE.NoCursos) as 'Numero'
```

5

Índices

Definición, uso y sintaxis

Metéle un índice a la tabla para que el query corra más rápido”

-- Todos en algún momento en TI

- Estructura de datos sobre tablas que permite localizar rápidamente registros de la misma.
- Crear un índice NO cambia los datos de la tabla, solo hace referencia a la tabla.
  - Puede requerir su propio espacio en disco
- Los usuarios no ven los índices, solo los usan para acelerar la ejecución de sus consultas.

- Índice de un libro
  - Referencia a información que se almacena en el libro
  - Si se necesita un capítulo en especial va al índice, encuentra el número de página y se consulta directamente.
  - Sin índices, el proceso de encontrar su capítulo deseado puede ser muy lento.

ÍNDICE AGRUPADO

- Define el orden en el cual los datos son físicamente almacenados en una tabla.
- Los datos pueden ser ordenados sólo en una forma, por lo tanto, sólo puede haber un índice agrupado por tabla.
- La restricción de llave primaria crea automáticamente un índice agrupado en esa columna en particular.

ÍNDICE AGRUPADO

CarneDocente	PrimerNombre	PrimerApellido	Email
1	Kellen	Horton	khorton@university.com
2	Kamron	Saunders	ksaunders@university.com
3	Dominick	Pope	dpope@university.com
4	Gerardo	Grimes	ggrimes@university.com
5	Elaine	Cabrera	ecabrera@university.com
6	Sophie	Norris	snorris@university.com
7	Julio	Cohen	jcohen@university.com
8	Farrah	Cantrell	fcantrell@university.com
9	Kayden	Price	kprice@university.com
10	Ellie	Bass	ebass@university.com
11	Lilianna	Watts	lwatts@university.com
12	Zariah	Bowers	zbowers@university.com
13	Kylie	Cameron	kcameron@university.com
14	Josie	Hopkins	jhopkins@university.com
15	Jaxon	Moon	jmoon@university.com

ÍNDICE AGRUPADO

```
CREATE CLUSTERED INDEX IndexName
ON TableName(column_name,...)
```

ÍNDICE NO-AGRUPADO

- No ordena los datos físicos dentro de la tabla.
- Es agrupado en un solo lugar y los datos de la tabla son almacenados en otro lugar. (El índice de libro de texto)
- Así, es posible tener más de un índice no agrupado por tabla.
- Los índices no agrupados son más lentos que los índices agrupados.

ÍNDICE NO-AGRUPADO

```
CREATE NONCLUSTERED INDEX IndexName
ON TableName(column_name,...)
```

ÍNDICES AGRUPADOS Y NO-AGRUPADOS

- Los índices agrupados sólo ordenan tablas. Por lo tanto, no consumen almacenamiento extra.
- Los índices no agrupados son almacenados en un lugar separado de la tabla real. Utilizan más espacio de almacenamiento.

ÍNDICE ÚNICO

Aquel en el que no se permite que dos filas tengan el mismo valor en la columna de clave del índice. Es decir que no permite valores duplicados.

```
create [unique] index IndexName on
TableName (column_name)
```

SIMPLE

Definido sobre una sola columna de la tabla

COMPUESTO

Formado por varias columnas de la misma tabla.

- Los registros que se recuperen utilizando el índice aparecerán ordenados por el campo indexado.
  - Para un índice compuesto por las columnas col1 y col2:
    - Las filas aparecerán ordenadas por col1 y las que tengan el mismo valor de col1 se ordenarán por col2. (cláusula ORDER BY).

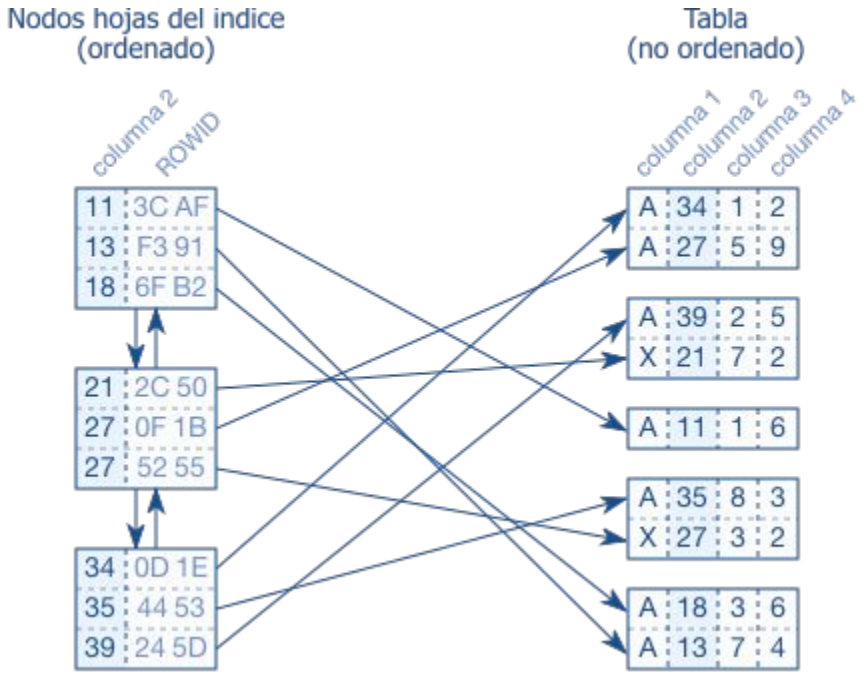
- Mejorar el rendimiento en las consultas ya que los datos de la consulta pueden existir en el propio índice.
  - Reduce la carga de lectura y escritura en el disco.
- Mejora el rendimiento de una vista si tiene agregaciones, combinaciones o una mezcla.

- Pueden ocupar espacio en el almacenamiento secundario.
- Consumen recursos (memoria principal, procesador, disco) ya que se debe actualizar cada vez que se realiza una operación de actualización, inserción o borrado.
  - No definir índices indiscriminadamente.

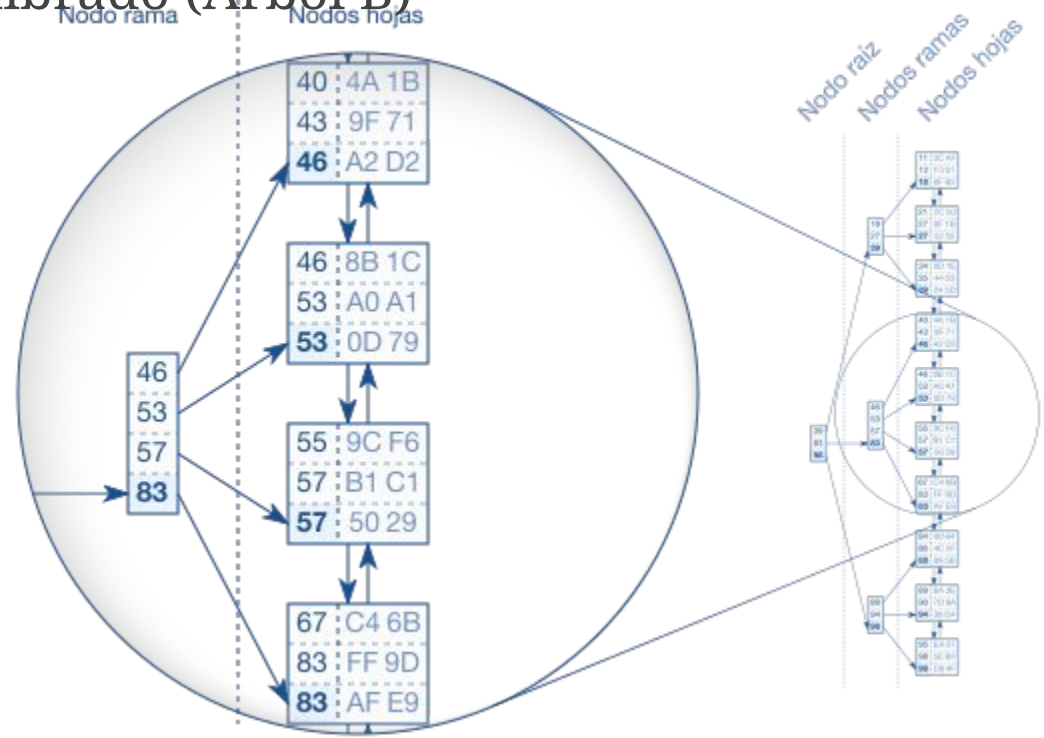
- Evitar crear demasiados índices en tablas con actualizaciones frecuentes.
- Definir índices con el menor número de columnas posible.
- Los índices son útiles en tablas con pocas actualizaciones y grandes volúmenes de datos.

- Para poder atender las solicitudes insert, delete y update de inmediato conservando el orden del índice, éste combina internamente dos estructuras: lista doblemente enlazada y un árbol de búsqueda.

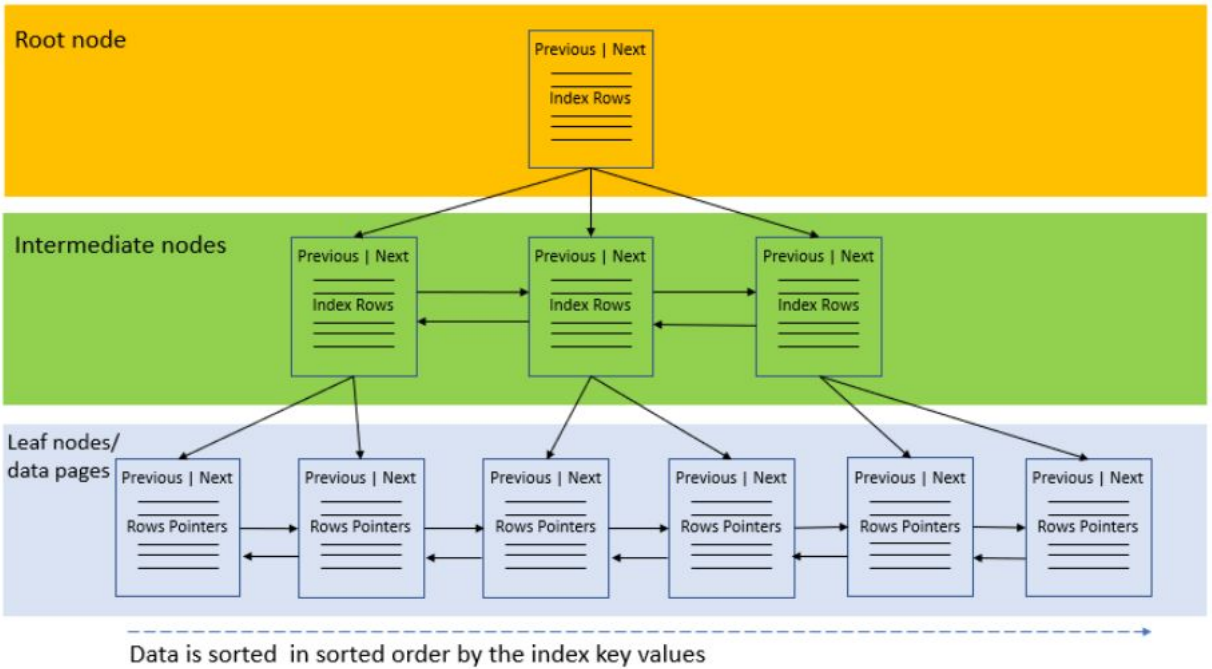
Orden lógico: Lista doblemente enlazada



- Orden entre nodos mezclados: árbol de búsqueda equilibrado (Árbol B)



- Orden entre nodos mezclados: árbol de búsqueda equilibrado (Árbol B)



Fin de curso de Base de Datos I

¡Éxitos a todos en su carrera profesional!