**6.- Tecnicas para mejorar el rendimiento de las consultas**

**Lesson1 Tunning queries**

**Query Cost**: Es una metrica interna de SQL Server que considera el uso de CPU y recursos de I/O para realizar una consulta. No considera las UDF ni las rutinas CLR

**Page Reads**: Representa el numero de data pages (8KB) accesados por una consulta. Se activa con SET STATISTICS IO ON, donde al ejecutar una consulta se devuelve lo siguiente:

logical reads: Número de paginas leidas desde memoria

physical reads: Número de paginas leidas desde disco duro

scan count: Numero de veces que se requiere accesar a un indice o al Heap

lob reads: lectura de varchar(max), varbinary(max), text, image, XML.

No considera uso de CPU ni las UDF ni las rutinas CLR

**Query Execution Time**: Se activa con SET STATISTICS TIME ON

**Theoretical query execution order** es:

FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY, TOP, FOR XML

La optimiazcion de consultas consiste en reescribir la consulta, normalizar/desnormalizar tablas, agregar/quitar indices, etc

**Plan estimado de ejecución**

Objetos con un costo alto que se deben revisar en un plan estimado de ejecucion:

Flechas gruesas: Indican un gran numero de filas que viajan de un nodo a otro

Hash Operations: Relacionado a JOINS y GROUP BY que no tienen un indice adecuado

Sorts: Considerar si el ordenamiento es requerido o se si se puede sustituir por un indice

Planes largos: Un plan con la menor cantidad de operaciones generalmente es mejor

Table Scan/Index Scan: Indican que no hay un indice adecuado para el rendiemiento

**SARG** (Search argument): Es una expresion de filtrado que es usado para limitar el numero de filas de una consulta y puede usar un indice que mejorara el rendimiento. Ejemplo:

Considerando que hay una tabla con indice en el campo OrderDate

SELECT \* FROM TABLA WHERE YEAR(OrderDate)= 2004 (No es un SARG)

SELECT \* FROM TABLA WHERE OrderDate BETWEEN ‘20040101’ AND ‘20050101’ (es un SARG)

**JOINS**: Es recomendable usar el menor numero de joins y de preferencia que sean inner. Poner los filtros en el ON o en el WHERE no tiene difdrencia si la consulta esta formada por solo INNER JOIN, en caso contrario si hay diferencia.

**SUBCONSULTAS**

**Subconsultas de clausula WHERE/JOIN**

Si no tienen campos correlacionados con la consulta externa generalmente tienen poca sobrecarga. Ejemplo:

SELECT ID, NOMBRE FROM TABLA1

WHERE PRECIO > (SELECT AVG(PRECIO) FROM TABLA2)

Si tienen campos correlacionados con la consulta externa generalmente tienen poca sobrecarga si se ocupan con un EXISTS. Ejemplo:

SELECT ID, NOMBRE FROM TABLA1 T1

WHERE EXISTS (SELECT \* FROM TABLA2 T2 WHERE T2.ID = T1.ID)

**Subconsultas de clausula SELECT**.

Tienen bajo rendimiento porque se ejecutan por cada fila consultada. Una alternativa podria ser poner la subconsulta en una claula APPLY

**Scalar UDF**

Tienen bajo rendimiento porque no se incluyen en el plan de ejecucion de SQL Server. Pueden ser sustituidos por una subconsulta, lo cual dara mejor rendimiento

**Cursors**

Tienen bajo rendimiento. Pueden ser sustituidos por selects, insert, update, delete o merge. En ultimo caso se puede sustituir por CLR stored procedure o una table-valued UDF

**SQL Server Profiler**

Permite escuchar eventos que ocurren en una instancia de SQL Server. Se debe escuchar los eventos SQL:BatchCompleted y RPC:Completed. Checar las columnas Duration, Reads, Writes y CPU. Los resultados son extensos por lo que se deben redireccionar a un archivo o a una tabla

Nota: Es importante disminuir los datos consultados cuanto antes sea posible en la consulta

**Lesson2 Creating indexes**

Los **indices** son arboles balanceados donde el nivel hoja es el mas bajo y es el que contiene los datos

Los **indices clustered** contienen las filas y columnas de la tabla

Los **indices nonclustered** contiene solo las columnas del indice mas un puntero a la fila actual

La nocion de **covered index** es que SQL Server no necesita buscar entre nonclustered index y la tabla para regresar los datos de la consulta, ya que todos los campos de la consulta se encuentran en el indice. Esto genera mejor rendimiento

Como un clustered index es la tabla actual, entonces siempre cubre la consulta y se recomienda considerar los campos que mas sean filtrados y devueltos o campos Foreign Key. Tambien es importante que contenga el menor numero de campos

Una tabla cuando tiene un clustered index se llama sorted table en caso contrario heap o unsorted table.

Crear indices nonclustered que sean covered index se facilita con la caracteristica **INCLUDE** que permite agregar columnas sin agregarlas al index keys. Las columnas include solo se deben agregar para usarse en el SELECT (avoiding lookup) y no para filtrado u ordenamiento

Los indices con campos include ocupan menos espacio

CREATE NONCLUSTERED INDEX <nombre\_indice> ON <tabla> (columnas keys) INCLUDE (columnas include)

Agregar indices aunque favorece el rendiento de la lectura de datos tambien genera carga en la escritura de datos y en el espacio en disco

Los indices tambien funcionan en campos calculados y encampos con diferente COLLATION

Las vistas materializadas pueden tener indices pero es necesario poner el hint WITH(NO EXPAND) (a menos que se tenga la version Enterprise). Las vistas indexadas brindan mejor rendimiento que los indices

Para mejorar la importacion de datos se pueden usar particiones en tablas e indices

Para revisar/depurar los indices que no se ocupan se puede utilizar la vista sys.dm\_db\_index\_usage\_stats

**Metodos automaticos de tunning:**

Se pueden revisar los indices faltantes en las vistas de sistema sys.dm\_db\_missing\_index\_details, sys.dm\_db\_missing\_index\_groups, sys.dm\_db\_missing\_index\_group\_stats

Tambien se puede abrir el Database Engine Tuning Advisor el cual es una utilidad grafica que permite obtener informacion util para la mejora del rendimiento

**Adicional. Operaciones basicas del plan de ejecucion**

**Scan Table**: Lee completamente la tabla sin utilizar indices.

**Clustered Index Scan**: Lee completamente el indice clustered de la tabla.

**Index Scan**: Lee completamente el indice non-clustered de la tabla.

**Clustered Index Seek**: Realiza una busqueda en el indice clustered de la tabla.

**Index Seek**: Realiza una busqueda en el indice non-clustered de la tabla.

**Bookmark Lookup**: Ocurre en consultas que usan campos que no estan en el indices non-clustered y se tiene que recurrir a la tabla real para obtenerlos. Se puede corregir incluyendo solo los campos necesarios o usar indices con INCLUDE

**RID Lookup**: Ocurre cuando no se encuentra un indice clustered y el motor elige utilizar el indice unico ROW ID

**Sort**: Ocurre cuando se ordena por un campo que no esta preordenado por un indice

**Nested Loop**: Ocurre en un join donde la primera tabla esta ordenada y con elementos unicos(por un indice) y la segunda tabla no esta ordenada. Entonces se realiza una busqueda de cada elemento de la primera tabla en **todos** los elementos de la segunda que no esta ordenada

**Merge Join**: Ocurre en un join donde la primera tabla esta ordenada y con elementos unicos(por un indice) y la segunda tabla esta ordenada. Entonces se realiza una busqueda de cada elemento de la primera tabla en solo **algunos** los elementos de la segunda porque esta ordenada

**Hash Match**: Ocurre en un join donde ninguna de las dos tablas esta ordenada. Entonces el motor tiene que crear **una tabla temporal** ordenada para sutituir una de las tablas y luego emular un join como el Nested Loop

**Stream Aggregate**: Ocurre cuando hacemos uso de Group by, Having y funciones de agregado