Investigación

Índices de Tres Estrellas: un caso de estudio en Oracle y SQL Server

Index of Three Stars: a case study in Oracle and SQL Server

Recibido: junio de 2012
Arbitrado: agosto de 2012

Francisco J. Moreno A*. Jhon W. Olarte V. **

Resumen

En este artículo se presentan los conceptos esenciales para el diseño de índices de tres estrellas. El objetivo de este tipo de índices es mejorar el desempeño de las operaciones de manipulación de datos como consultas, actualizaciones y borrados. Con el fin de evaluar su beneficio, se analizó su comportamiento en dos Sistemas de Gestión de Bases de Datos (SGBD): Oracle 11g y SQL Server 2008 R2. Los resultados evidenciaron una mejora significativa en el desempeño de las operaciones en ambos SGBD y corroboraron la importancia de una adecuada indexación. Los resultados también permitieron observar algunas diferencias en el comportamiento de los dos SGBD frente a este tipo de índices.

Palabras clave

índice, predicado, acople, camino de acceso, rebanada de un índice.

Abstract

In this paper, we present the basic concepts for the design of three-star indexes. The goal of this type of index is to improve the performance of data manipulation operations such as queries, updates, and deletes. In order to evaluate their benefit, we analyze their behavior in two Database Management Systems (DBMS): Oracle 11g and SQL Server 2008 R2. Our results showed a significant improvement in the operations' performance in both DBMS and supported the importance of proper indexing. Our results also showed some differences in the behavior of the two DBMS against such indexes.

Keywords

Index, predicate, matching, access path, index slice.

I. Introducción

En muchas organizaciones existen actualmente muchas aplicaciones desarrolladas en Sistemas de Gestión de Bases de Datos (SGBD) que presentan problemas de desempeño, i.e., los tiempos de respuesta no satisfacen las expectativas de

* Doctor en Ingeniería de Sistemas e Informática, Universidad Nacional de Colombia, sede Medellín. Profesor Universidad Nacional de Colombia Sede Medellin. E-mail: fjmoreno@unal.edu.co

** Estudiante Ingeniería de Sistemas. Universidad Nacional de Colombia, sede Medellín. E-mail: jwolartev@unal.edu.co

los usuarios. Una de las posibles causas es que muchas operaciones, como las consultas, no están optimizadas. Por ejemplo, la falta de índices adecuados o la presencia de índices inadecuados pueden llevar a un pobre desempeño de las consultas.

Además, algunos desarrolladores consideran que con la capacidad de procesamiento y de almacenamiento actual, ya no es necesario afinar las operaciones DML (*Data Manipulation Language*) [1], i.e., consultas, actualizaciones y borrados; y se delega este proceso enteramente al optimizador del SGBD, omitiendo en algunas oportunidades la creación de índices valiosos.

En otras ocasiones se crean índices que no responden a las necesidades de las operaciones DML, generando de esta forma índices poco usados o cuyo uso puede ser perjudicial para el rendimiento de estas operaciones.

En forma análoga a los hoteles, los índices se clasifican por el número de estrellas. Las estrellas (primera, segunda y tercera) se asignan con base en criterios que se explican más adelante. Los índices de tres estrellas son los ideales para mejorar el desempeño de una operación DML. Sin embargo, a diferencia de los hoteles, un índice podría tener, e.g., la tercera estrella pero carecer de la primera o de la segunda.

El artículo está organizado así. En la sección 2, se presentan los conceptos esenciales para el diseño de índices de tres estrellas y en la sección 3 se presenta el método para su diseño. En la sección 4, se presentan los experimentos ejecutados en dos SGBD y en la sección 5 se proponen trabajos futuros y se concluye el artículo.

II. Conceptos esenciales

En esta sección se presentan algunos conceptos necesarios para el diseño de índices de tres estrellas.

2.1 Predicado

Un predicado es una expresión condicional que tiene un valor de verdad (TRUE, FALSE o UNKNOWN en SQL). Un predicado especifica un subconjunto de filas en una operación DML [2]. Los predicados son el punto de partida para el diseño de los índices y se clasifican como simples y compuestos. Un predicado simple está conformado así [3]:

Operando OpComp Operando

Donde Operando es una columna o una constante y OpComp es un operador de comparación como =, >, <, etc. Un predicado compuesto consta de dos predicados, ya sean simples o compuestos, conectados por los operadores lógicos AND y OR. Por ejemplo, considérese la siguiente cláusula WHERE:

```
WHERE SEXO = 'M'
        AND
  (PESO > 90 OR ESTATURA > 190);
```

Los predicados de la cláusula anterior son

1. SEXO = 'M'.
2. PESO > 90.
3. ESTATURA > 190.
4. PESO > 90 OR ESTATURA > 190.
5. SEXO = 'M' AND (PESO > 90 OR ESTATURA > 190).

Donde los predicados 1, 2 y 3 son simples y los predicados 4 y 5 son compuestos.

2.2 Operaciones de E/S

Una operación de E/S (Entrada/Salida) es una petición del SGBD al procesador, el cual escribe o lee del disco un conjunto de páginas (bloques) las cuales contienen los datos correspondientes (filas de las tablas de la base de datos). Las

operaciones de E/S pueden ser *síncronas* o *asíncronas*. Cuando la operación de E/S es síncrona, el SGBD:

f. Identifica la fila de la tabla solicitada.

g. Accede a la página que contiene la fila solicitada.

h. Procesa la fila.

En una operación síncrona, cada uno de estos pasos se ejecuta cuando se ha terminado el paso anterior; así para procesar una fila (paso 3), los pasos 1 y 2 tienen que haber terminado. De esta forma, el SGBD debe esperar que finalice una operación de E/S, para poder continuar con la lectura o escritura de las siguientes filas. Por otro lado, cuando la operación es asíncrona, el SGBD no tiene que esperar a que la operación E/S se complete, i.e., puede ir ejecutando los pasos 1, 2 y 3 en paralelo para diferentes filas [4].

2.3 Camino de acceso

Antes de ejecutar una operación DML, el optimizador del SGBD debe decidir cómo acceder a los datos, i.e., elegir los caminos de acceso (*access paths*) que estén disponibles para recuperar los datos de la base de datos [5]. Por ejemplo, el optimizador debe decidir si usar

- Índices y la forma de usarlos, e.g., acceso por rango.
- Accesos completos a la tabla (*full table scans*).
- Operaciones de E/S síncronas o asíncronas.

2.4 Rebanada de un índice

Usualmente, las operaciones DML que acceden a un subconjunto pequeño de filas de una tabla usan índices, mientras que un acceso completo a una tabla se usa cuando estas operaciones acceden a un subconjunto grande de filas de una tabla. Una rebanada de un índice (*index slice*) es la porción del índice que se va a recuperar para una determinada operación DML.

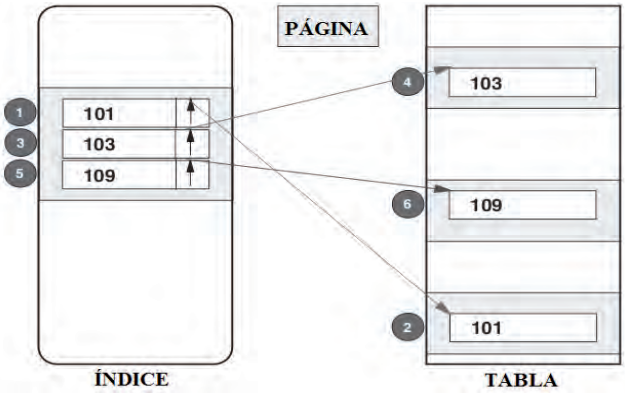

Figura 1. Rebanada de un índice.

En la figura 1 se observa una rebanada de un índice, que tiene los valores 101, 103 y 109. Además, se muestra la forma en que se recuperarán las filas mediante lecturas síncronas. En dicha figura, los números del 1 al 6 indican la secuencia en que se hacen los accesos, i.e., primero se lee la fila requerida (101, paso 1) desde el índice y mediante su puntero se accede a la página correspondiente y se procesa (paso 2). Hasta que no finalice esta operación de E/S no se leerá la siguiente fila (la 103, paso 3).

2.5 Columna Matching y de Screening

Una columna que es referenciada en un predicado simple de una operación DML y que además hace parte de un índice se denomina *M* (*matching*, de acople) o *S* (*screening*, de filtro o control). La diferencia está en que las columnas *M* delimitan con mayor precisión la rebanada del índice a recuperar que las columnas *S*; sin embargo, estas pueden llegar a reducir el número de lecturas síncronas, y representan, por lo tanto, un papel importante en la optimización. Por ejemplo, considérese la figura 2 donde se muestra para una cláusula WHERE como se identifican las columnas *M* y *S*. Examinar las columnas del índice a partir de su primera columna y seguir estos pasos:

a. ¿La columna es referenciada en al menos un predicado simple de la cláusula WHERE?
   Sí: la columna es *M*.
   No: esta y las demás columnas no son *M*.

b. Si el predicado es de rango, las demás columnas del predicado no son columnas *M*.

c. Cualquier columna después de la última columna *M* es una columna *S* si hay al menos un predicado simple que la refiere.

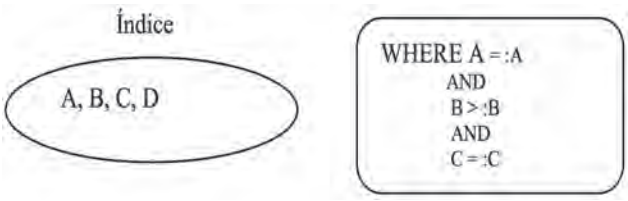

Figura 2. Ejemplo de columnas *M* y *S*.

Supóngase un índice con cuatro columnas (A, B, C, D) y la cláusula WHERE de la Figura 2. Para identificar las columnas *M* y *S* se procede así:

1. Se examina la primera columna del índice, i.e., A.
2. Como en la cláusula WHERE hay un predicado simple que se refiere a esta columna, entonces A es una columna *M*.
3. Ahora se examina la segunda columna del índice, i.e., B y se observa que esta también es referenciada en la cláusula WHERE, en un predicado de desigualdad (predicado de rango); por lo tanto, también es una columna *M*.
4. De acuerdo con el paso 2 de la figura 2, se concluye que las demás columnas de la cláusula WHERE no serán columnas *M*.
5. De acuerdo con el paso 3 de la figura 2, como la columna C está en el índice y además se referencia en un predicado simple, entonces es una columna *S*.

En conclusión, hay dos columnas *M* (A y B) y una columna *S* (C).

2.6 Lectura aleatoria y secuencial

El *buffer pool* (área común de memoria intermedia) es un área temporal de almacenamiento de páginas de la base de datos [6]. El proceso del SGBD administrador del *buffer pool* procura que los datos solicitados frecuentemente por los usuarios se encuentren allí, para minimizar así las lecturas de páginas desde el disco. Una lectura aleatoria (*random read*) se refiere a la ubicación y transferencia de una página que se encuentra en disco al *buffer pool*. Por otro lado, una lectura secuencial (*sequential read*) considera que varias páginas de una tabla o de un índice se deben leer secuencialmente, i.e., estas páginas están adyacentes en disco y son leídas una tras otra.

Los tiempos supuestos para las operaciones de E/S son:

1. Lectura aleatoria: 0.4 MB/s o 10 ms (4K o 8K página).
2. Lectura secuencial: 40 MB/s.

2.7 Factor filtro

El factor filtro especifica la *selectividad* del predicado. La selectividad es el porcentaje de filas seleccionadas en una operación DML. Una operación DML que selecciona un subconjunto pequeño de filas tiene selectividad alta, mientras que una operación DML que selecciona un subconjunto grande de filas tiene selectividad baja [7].

Por ejemplo, supóngase que se tienen 500 ciudades y que se tiene una tabla con un millón de clientes, cada uno ubicado en una ciudad. Por lo tanto, el factor filtro para un predicado CIUDAD =: CIUDAD será 0.2% (suponiendo una distribución uniforme de los clientes en las ciudades).

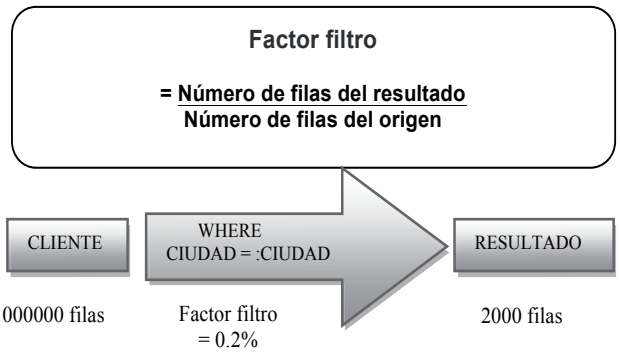

Figura 3. Ejemplo de factor filtro

El factor filtro también se puede obtener para predicados compuestos. Este se puede derivar de los predicados simples siempre y cuando los valores de las columnas de los predicados simples no estén correlacionados. Por ejemplo, considérese el predicado compuesto CIUDAD =: CIUDAD AND APELLIDO =: APELLIDO. El factor filtro de este predicado será el producto del factor filtro de CIUDAD =: CIUDAD y de APELLIDO =: APELLIDO. Por ejemplo, si la columna CIUDAD tiene 500 valores diferentes y APELLIDO tiene 1.000 valores diferentes entonces el factor filtro para el predicado compuesto será 1/500 * 1/1.000 = 1/500.000. Este valor indica que existen quinientas mil posibles combinaciones para los valores de las columnas CIUDAD y APELLIDO.

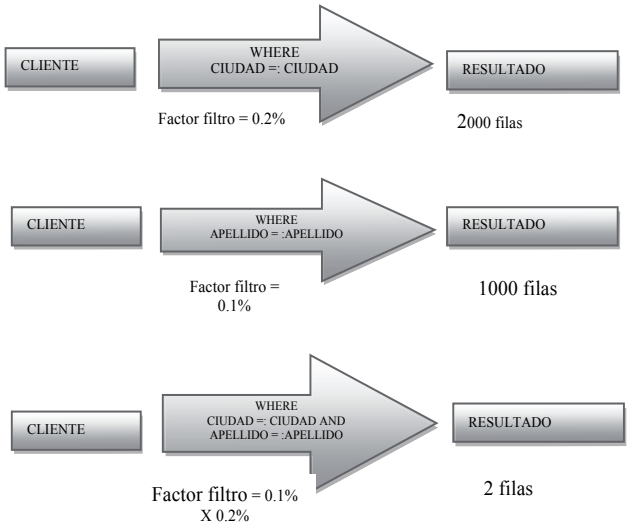

Figura 4. Ejemplo de factor filtro para el predicado compuesto.

Considérese la siguiente consulta:

```
SELECT      CNO, NOMBRE
FROM        CLIENTE
WHERE       APELLIDO   = :APELLIDO
            AND
            CIUDAD     = :CIUDAD
ORDER BY    NOMBRE;
```

Figura 5. Consulta 1

Una consulta que involucra una sola tabla, y que usa un índice como el de la tabla 1, usualmente sólo necesita una lectura aleatoria (10 ms) y una exploración (lecturas secuenciales) de una rebanada delgada del índice.

APELLIDO	CIUDAD	NOMBRE	CNO
:::::::::::::	:::::::::::::::	:::::::::::::::::	:::::::::::::
TORRES	BILBAO	MARÍA	0256349
TORRES	MADRID	ANDREA	0002567
TORRES	MADRID	BELINDA	4789513
TORRES	MADRID	CARLOS	2145632
:::::::::::::	:::::::::::::	:::::::::::::	:::::::::::::
TORRES	MADRID	ZULEMA	9876433
TORRES	TOLEDO	RODRIGO	0142789
:::::::::::::	:::::::::::::::	:::::::::::::::::	:::::::::::::::

Tabla 1. Muestra de datos de un índice para la consulta 1. En gris las filas para APELLIDO = 'TORRES' y CIUDAD = 'MADRID'.

Supóngase que para la consulta de la figura 5, hay 1000 filas en el resultado, así el factor filtro para el predicado compuesto APELLIDO = :APELLIDO AND CIUDAD = :CIUDAD es 0.1%. La recuperación ahora de la rebanada del índice es de solo 1000 filas, dado que hay dos columnas *M* (APELLIDO y CIUDAD).

III. Diseño de índices de Tres Estrellas

Para el diseño de los índices de tres estrellas [1], se consideran los siguientes casos.

3.1 Caso 1

Este caso se puede aplicar cuando:

a. La cláusula WHERE no tiene predicados de rango (BETWEEN, >, >=, etc.)

b. No hay reuniones (*joins*).

c. Los predicados no presentan «dificultad» para el optimizador. Por ejemplo, los predicados que involucran funciones sobre las columnas, como CONCAT (COL1, COL2) > :string o COL1 LIKE '%AB', no permiten

definir la rebanada del índice. También son llamados predicados no indexables [8].

Primera Estrella: para que un índice satisfaga la primera estrella, se deben elegir todas las columnas de la consulta que estén referenciadas en predicados de igualdad (WHERE col1 =:col1 AND col2 = :col2 ...), y estas serán las primeras columnas del índice. Por ejemplo, considérese la consulta de la figura 5. Para asignar la primera estrella a un índice con respecto a esta consulta, las columnas CIUDAD y APELLIDO deben ser las primeras columnas del índice (su orden no importa en el índice).

Segunda Estrella: para satisfacer la segunda estrella, el índice debe contener las columnas referenciadas en la cláusula ORDER BY (si la hay). Siguiendo con el ejemplo anterior, se incluye la columna NOMBRE en el índice. De esta forma, con NOMBRE como la tercera columna del índice, los resultados estarán en el orden requerido por la consulta.

Tercera Estrella: para satisfacer la tercera estrella se deben incluir en el índice todas las columnas referenciadas en la consulta. El orden de estas columnas en el índice no afecta el rendimiento de la consulta. Por ejemplo, para asignar la tercera estrella a un índice en el ejemplo anterior, todas las columnas referenciadas en la cláusula SELECT se deben incluir en el índice. Se incluye entonces la columna CNO.

Por lo tanto, para la consulta de la figura 5, el índice de tres estrellas resultante será (APELLIDO, CIUDAD, NOMBRE, CNO), véase Tabla 1, o (CIUDAD, APELLIDO, NOMBRE, CNO)

3.2 Caso 2

Este caso se puede aplicar cuando:

a. No hay reuniones (*joins*).

b. Ninguno de los predicados presenta dificultad para el optimizador.

Considérese la siguiente consulta:

```
SELECT        CNO, NOMBRE
FROM      CLIENTE
WHERE      APELLIDO  BETWEEN   :APELLIDO1  AND :APELLIDO2
AND CIUDAD   =  :CIUDAD
ORDER BY      NOMBRE;
```

Figura 6. Consulta 2 con un predicado de rango (BETWEEN).

El procedimiento es similar al caso 1; sin embargo, el predicado de rango, afectará el diseño de un índice de tres estrellas para esta consulta.

Primera Estrella: para que un índice satisfaga la primera estrella cuando hay predicados de rango, como BETWEEN, >, <, etc.; las primeras columnas del índice deberán ser las columnas que se encuentren referenciadas en predicados de igualdad (si los hay), seguidas de las columnas del predicado de rango. Por ejemplo, considérese la consulta de la figura 6. La primera columna del índice será CIUDAD, seguida de la columna APELLIDO. De esta forma, se tienen dos columnas M, que definirán la porción del índice a explorar.

Segunda Estrella: para satisfacer la segunda estrella, la columna NOMBRE (referenciada en la cláusula ORDER BY) debe ir antes de la columna APELLIDO y de esta forma se evita un proceso adicional para ordenar los resultados. Por ejemplo, supóngase que se hubiera incluido en el índice la columna NOMBRE luego de la columna APELLIDO, como se muestra en la tabla 2.

CIUDAD	APELLIDO	NOMBRE	CNO
:::::::::::::::	:::::::::::::::	:::::::::::::::::	::::::::::::
BILBAO	TORRES	MARÍA	0256349
MADRID	TORRES	TATIANA	0002567
MADRID	TORRES	PEDRO	4789513
MADRID	TORRES	PEDRO	2145632
::::::::::::	:::::::::::::::	:::::::::::::::::	::::::::::::
MADRID	TORRES	VÍCTOR	4563111
TOLEDO	TORRES	RODRIGO	0142789
::::::::::::	:::::::::::::::	:::::::::::::::::	::::::::::::

Tabla 2. Muestra de datos del índice (CIUDAD, APELLIDO, NOMBRE) para la consulta 2.

Las filas del índice aquí, no están ordenadas por NOMBRE, e.g., Pedro debe salir en los resultados antes que Tatiana; por lo tanto, se requiere un proceso adicional que ordene los resultados.

Tercera Estrella: para satisfacer la tercera estrella se deben incluir en el índice todas las columnas referenciadas en la consulta. Por lo tanto, para la consulta de la figura 6, el índice de tres estrellas es (CIUDAD, NOMBRE, APELLIDO, CNO).

En la tabla 3 se resumen las características de cada estrella.

ESTRELLA	REQUISITOS	VENTAJAS	DESVENTAJAS
Primera	Se incluyen en el índice primero todas las columnas de los predicados de igualdad y luego las columnas de los predicados de rango.	La rebanada del índice será tan delgada como sea posible.	Al tener la primera estrella no siempre es posible evitar un proceso adicional de ordenamiento de los resultados.
Segunda	Se incluyen en el índice todas las columnas referenciadas en la cláusula ORDER BY (siempre y cuando no hayan sido ya incluidas en el índice). Estas columnas deben conservar el mismo orden que tienen en la cláusula ORDER BY.	Los resultados estarán en el orden requerido por la consulta.	Al tener la segunda estrella, no siempre es posible definir la rebanada más delgada del índice.
Tercera	Se incluyen en el índice todas las demás columnas referenciadas en la consulta. El orden en que son incluidas en el índice no influye en el rendimiento.	El índice tiene todas las columnas necesarias para solucionar la consulta con un acceso al índice y evitar un acceso a la tabla.	

Tabla 3. Consideraciones para la creación de un índice de tres estrellas.

Consideraciones adicionales

Con respecto a una operación DML, siempre se puede diseñar un índice que tenga la tercera estrella, pero podría carecer de la primera o de la segunda estrella. Por ejemplo, el índice puede:

- Evitar un proceso adicional para ordenar los resultados al tener la segunda estrella pero la rebanada del índice puede ser más gruesa en comparación a si tuviera la primera estrella.
- Generar la rebanada más delgada posible para minimizar el número de lecturas del índice al tener la primera estrella, pero se podría necesitar un proceso adicional que ordene los resultados debido a la carencia de la segunda estrella.

Por lo tanto, la presencia de un predicado de rango implica que no siempre se puede tener un índice de tres estrellas; de esta forma se tienen dos índices candidatos para el caso 2.

3.2.1 índice candidato A

1. Elegir las columnas del predicado de igualdad. Estas serán las primeras columnas del índice (su orden no importa en el índice).
2. La siguiente columna del índice es la columna del predicado de rango más selectivo posible.
3. Incluir en el índice las columnas referenciadas en la cláusula ORDER BY, además de las columnas ya incluidas en los pasos 1 y 2.
4. Incluir en el índice todas las columnas restantes referenciadas por la consulta (su orden no importa en el índice).

3.2.2 Índice candidato B

1. Elegir las columnas del predicado de igualdad. Estas serán las primeras columnas del índice (su orden no importa en el índice).
2. Incluir en el índice las columnas referenciadas en la cláusula ORDER BY, además de las columnas ya incluidas en el paso 1.

3. Incluir en el índice todas las columnas restantes referenciadas por la consulta, incluyendo las de predicados de rango (si las hay). Su orden no importa en el índice.

IV. Casos de estudio

A continuación se presentan los casos de estudio, donde se analizó el comportamiento de índices en dos SGBD, Oracle 11g y SQL Server 2008 R2; dichas pruebas se realizaron con una CPU de 2,00 GB de RAM, y procesador AMD Athlon(tm) 1.60 GHz. Para los experimentos se creó la tabla CLIENTE (CNO, NOMBRE, APELLIDO, CIUDAD). La tabla se creó sin clave primaria ya que en estos SGBD se define automáticamente un índice sobre esta clave. La tabla se pobló con un millón de filas, las columnas NOMBRE y APELLIDO cada una con 60 valores diferentes y la columna CIUDAD con 30 valores diferentes.

4.1 Caso de estudio Oracle

Se probó la consulta 1, véase figura 5, con los siguientes casos:

- Índice inadecuado.
- Acceso completo a la tabla.
- Índice de tres estrellas.

Para el primer caso se creó un índice, llamado PRUEBA1, con las columnas APELLIDO y NOMBRE. En la tabla 7 se muestra el plan de ejecución para esta consulta. En la consulta se incluyó el *hint* [9] INDEX para forzar el uso del índice en Oracle (un *hint* es una indicación para influir el plan de ejecución). Nótese que este índice es inadecuado porque no tiene ninguna estrella con respecto a esta consulta.

Id	Operation	Name	Rows	Cost(%CPU)	Time
0	SELECT STATEMENT		158	7471 (1)	00:01:30
1	SORT ORDER BY		158	7471 (1)	00:01:30
*2	TABLE ACCESS BY INDEX ROWID	CLIENTE	158	7470 (1)	00:01:30
*3	INDEX RANGE SCAN	PRUEBA1	7018	30 (0)	00:00:01

Tabla 4. Plan de ejecución en Oracle para la consulta 1 con el índice inadecuado.

Luego se ejecutó la misma consulta pero sin forzar el uso del índice, i.e., sin el *hint*, y se obtuvieron los resultados de la tabla 5.

Id	Operation	Name	Rows	Cost(%CPU)	Time
0	SELECT STATEMENT		158	1394 (4)	00:00:17
1	SORT ORDER BY		158	1394 (4)	00:00:17
*2	TABLE ACCESS FULL	CLIENTE	158	1393 (4)	00:00:17

Tabla 5. Plan de ejecución en Oracle para la consulta 1 sin el índice.

Los resultados evidencian que el optimizador de Oracle decide ejecutar un acceso completo a la tabla en vez de usar el índice, dado que el índice inadecuado resulta más costoso que un acceso completo a la tabla.

Aunque la tabla 5 muestra un tiempo de respuesta relativamente bajo (17 s) en comparación con los resultados de la consulta de la tabla 4 (1 min 30 s) es posible mejorar aún más este tiempo con un índice adecuado. Se creó entonces un índice de tres estrellas para la Consulta 1 llamado PRUEBA 2, el cual tiene las columnas APELLIDO, CIUDAD, NOMBRE, CNO en ese orden. Los resultados se muestran en la tabla 6.

Id	Operation	Name	Rows	Cost(%CPU)	Time
0	SELECT STATEMENT		158	4 (25)	00:00:01
1	SORT ORDERBY		158	4 (25)	00:00:01
*2	INDEX RANGE SCAN	PRUEBA2	158	3 (0)	00:00:01

Tabla 6. Plan de ejecución en Oracle para la consulta 1 con el índice de tres estrellas.

Los resultados con el índice de tres estrellas evidencian una mejora significativa en cuanto al rendimiento (1 s). Nótese, sin embargo, que Oracle *no evitó* un proceso adicional para ordenar los resultados como se suponía debería haber sucedido.

Ahora se considera la consulta 2, véase la figura 6, y se analizaron los siguientes casos:

a. Índice correspondiente al candidato A.

b. Índice correspondiente al candidato B (evita un proceso adicional para ordenar los resultados).

- **Índice correspondiente al candidato A.**

Se creó un índice correspondiente al candidato A llamado PRUEBA3, con las columnas CIUDAD, APELLIDO, NOMBRE y CNO, en ese orden. Se eligen estas columnas dado que la primera columna de este índice (CIUDAD) corresponde a una columna referenciada en un predicado de igualdad, seguida del predicado de rango más selectivo (APELLIDO), i.e., el predicado de rango con la menor proporción de filas que satisface el predicado. Los resultados se muestran en la tabla 7.

Id	Operation	Name	Rows	Cost(%CPU)	Time
0	SELECT STATEMENT		20818	326 (2)	00:00:04
1	SORT ORDERBY		20818	326 (2)	00:00:04
*2	INDEX RANGE SCAN	PRUEBA3	20818	124 (1)	00:00:02

Tabla 7. Plan de ejecución en Oracle para la consulta 2 con el índice candidato A.

- **Índice correspondiente al candidato B.**

Se creó un índice correspondiente al candidato B llamado PRUEBA4, con las columnas CIUDAD, NOMBRE, APELLIDO y CNO, en ese orden y de esta manera los resultados estarán ordenados y *se debería* evitar un proceso adicional para ordenar los resultados.

Id	Operation	Name	Rows	Cost(%CPU)	Time
0	SELECT STATEMENT		20818	378 (2)	00:00:05
1	SORT ORDERBY		20818	378 (2)	00:00:05
*2	INDEX RANGE SCAN	PRUEBA4	20818	177 (2)	00:00:03

Tabla 8. Plan de ejecución en Oracle para la consulta 2 con el índice candidato B.

Aquí, se observó de nuevo que aunque el optimizador de Oracle usó el índice, desafortunadamente *no evitó* un proceso adicional para ordenar los resultados como se suponía debería haber sucedido.

4.2 Caso de estudio SQL Server

En esta sección se mostrarán los resultados obtenidos en el SGBD SQL Server.

Se probó la consulta 1, véase figura 5, con los siguientes casos:

- Índice inadecuado
- Acceso completo a la tabla
- Índice de tres estrellas

Para el primer caso se creó un índice, llamado PRUEBA 1, con las columnas APELLIDO y NOMBRE. En la tabla 9, se muestra el plan de ejecución para esta consulta. Además, al igual que en Oracle, se requiere el *hint* INDEX para forzar el uso del índice en SQL SERVER.

Id	Stmt Text (Cost)	Phisical Op	Logical Op	Estimate CPU	Estimate Rows
1	SELECT STATEMENT (0%)	NULL	NULL	NULL	560,7078
2	|-SORT ORDER BY (0%)	Sort	Sort	0,0080870	560,7078
3	|-FILTER (0%)	Filter	Filter	0,008088	560,7078
4	|-NESTED LOOPS (0%)	Nested Loops	Inner Join	0,0697433	16685
5	|-SORT ORDER BY (6%)	Sort	Sort	1,07429	16685
6	||-Compute Scalar (0%)	Compute Scalar	Compute Scalar	0,0016685	16685
7	||-Index Seeek (0%)	Index Seek	Index Seek	0,001685	16685
8	|-RID Lookup (94%)	RID Lookup	RID Lookup	0,0001581	1

Tiempos de ejecución SQL Server
Tiempo de CPU = 156 ms
Tiempo transcurrido = 526 ms

Tabla 9. Plan de ejecución en SQL Server [10] para la consulta 1 con el índice inadecuado.

Ahora se ejecuta la misma consulta 1, pero para este caso no se usa el *hint* INDEX, obteniendo los resultados de la tabla 10.

Id	Stmt Text (Cost)	Phisical Op	Logical Op	Estimate CPU	Estimate Rows
1	SELECT STATEMENT (0%)	NULL	NULL	NULL	560,7078
2	SORT ORDER BY (14%)	Sort	Sort	0,0080870	560,7078
3	TABLE SCAN (86%)	Table Scan	Table Scan	1,100157	560,7078

Tiempos de ejecución SQL Server
Tiempo de CPU = 125 ms
Tiempo transcurrido = 376 ms

Tabla 10. Plan de ejecución en SQL Server para la consulta 1 sin el índice.

En forma similar a Oracle, los resultados muestran que el optimizador de SQL Server hace un acceso completo a la tabla en vez de usar el índice, ya que si usa el índice inadecuado resulta más costoso que hacer un acceso completo a la tabla (526 ms vs. 376 ms). De igual manera, es posible mejorar aún más este tiempo con un índice adecuado. Para ello, se creó un índice de tres estrellas para la consulta 1, llamado PRUEBA 2, el cual tiene las columnas APELLIDO, CIUDAD, NOMBRE y CNO, en ese orden. Los resultados se muestran en la tabla 11.

Id	Stmt Text (Cost)	Phisical Op	Logical Op	Estimate CPU	Estimate Rows
1	SELECT STATEMENT (0%)	NULL	NULL	NULL	555,5555
2	INDEX SEEK (100%)	Index Seek	Index Seek	0,00076811	555,5555

Tiempos de ejecución SQL Server
Tiempo de CPU = 0 ms
Tiempo transcurrido = 2 ms

Tabla 11. Plan de ejecución en SQL Server para la consulta 1 con el índice de tres estrellas.

Los resultados obtenidos con el índice de tres estrellas evidencian una mejora significativa en los tiempos de ejecución (de 1000 ms a 2 ms). Además se observa que SQL Server *evita* un proceso adicional para ordenar los resultados, como se esperaba conforme a la teoría, a diferencia de Oracle.

Ahora se considera la consulta 2, véase la figura 6, y se analizaron los siguientes casos:

1. Índice correspondiente al candidato A.
2. Índice correspondiente al candidato B (evita un proceso adicional para ordenar los resultados).

• Índice correspondiente al candidato A

Al igual que en el caso de estudio anterior, se creó un índice llamado PRUEBA 3, con las columnas CIUDAD, APELLIDO, NOMBRE y CNO, en ese orden. Los resultados se muestran en la tabla 12.

Id	Stmt Text (Cost)	Phisical Op	Logical Op	Estimate CPU	Estimate Rows
1	SELECT STATEMENT (0%)	NULL	NULL	NULL	23372,31
2	SORT ORDER BY (93%)	Sort	Sort	1,556987	23372,31
3	INDEX SEEK (7%)	Index Seek	Index Seek	0,025866	23372,31

Tiempos de ejecución SQL Server
Tiempo de CPU = 93 ms
Tiempo transcurrido = 882 ms

Tabla 12. Plan de ejecución en SQL Server para la consulta 2 con el índice candidato A.

Los resultados indican que en SQL Server se usa el índice. Además se observa que para el optimizador resulta más costoso ordenar, que buscar en el índice (índice 7% y ordenar 93%).

• Índice Correspondiente al Candidato B

Se creó un índice correspondiente al candidato B llamado PRUEBA 4, con las columnas CIUDAD, NOMBRE, APELLIDO y CNO, en ese orden y de esta manera los resultados estarán ordenados y se debería evitar un proceso adicional para ordenarlos.

Id	Stmt Text (Cost)	Phisical Op	Logical Op	Estimate CPU	Estimate Rows
1	SELECT STATEMENT (0%)	NULL	NULL	NULL	23372,31
2	INDEX SEEK (100%)	Index Seek	Index Seek	0,0369069	23372,31

Tiempos de ejecución SQL Server
Tiempo de CPU = 0 ms
Tiempo transcurrido = 797 ms

Tabla 13. Plan de ejecución en SQL Server para la consulta 2 en SQL Server con el índice candidato B.

De la tabla 13, se observa que SQL Server no requiere ordenar los resultados, como se esperaba que sucediera, a diferencia de Oracle. Sin embargo, se observa también que la diferencia en relación con el tiempo de ejecución del índice candidato A y del índice candidato B (véanse tablas 12 y 13) no es significativa (882 ms vs. 797 ms).

En la tabla 14 se resumen los resultados obtenidos.

MÉTODO USADO	ORACLE	SQL SERVER	CONSULTA	TIEMPO DE EJECUCIÓN	
				Oracle	SQL Server
Índice inadecuado (se fuerza su uso)	Se requiere un hint para forzar el uso del índice inadecuado.		1	90 s	526 ms (0,526 s)
Índice inadecuado (no se fuerza su uso)	El optimizador decide hacer un acceso completo a la tabla en vez de usar un índice inadecuado.		1	17 s	376 ms (0,376 s)
Índice de tres estrellas	Se evidencia una mejora significativa en cuanto al rendimiento (con respecto al índice inadecuado y al acceso completo a la tabla).		1	1 s	2 ms (0,002 s)
	El optimizador no evitó un proceso de ordenamiento adicional como se esperaba.	El optimizador evitó un proceso de ordenamiento adicional como se esperaba.			
Índice candidato A	El optimizador usó el índice sin necesidad de forzar su uso.		2	4 s	882 ms (0,882 s)
Índice candidato B	El optimizador usó el índice sin necesidad de forzar su uso Se evitó un acceso completo a la tabla (se usó como camino de acceso sólo el índice).		2	5 s	797 ms (0,797 s)
	El optimizador no evitó un proceso de ordenamiento adicional como se esperaba.	El optimizador evitó un proceso de ordenamiento adicional como se esperaba.			

Tabla 14. Resumen de resultados caso de estudio Oracle y SQL Server.

V. Conclusiones y trabajo futuro

En este artículo se presentó y ejemplificó un método para mejorar el rendimiento de las operaciones DML, mediante la creación de los índices de tres estrellas. Los experimentos realizados en dos SGBD corroboran la importancia de una adecuada indexación.

Se observó como el SGBD Oracle no evita un proceso adicional para ordenar los resultados cuando se usa un índice de tipo candidato B, proceso que se debería haber evitado. Sin embargo, en el SGBD SQL Server sí se evitó dicho proceso.

También se mostró cómo Oracle decide no usar un índice si este es inadecuado, ya que esto resultaría más costoso que hacer un acceso completo a la tabla. Este mismo comportamiento se evidenció en SQL Server, al igual que cuando se usó un índice de tipo candidato A.

Como trabajo futuro se planea analizar el diseño de índices para consultas que involucran reuniones, además de sus respectivos casos de estudio en diferentes SGBD.

Referencias

[1] T. Lahdenmäki, M. Leach. *Relational Database Index Design and the Optimizers: DB2, Oracle, SQL Server*, ét al. Nueva Jersey. Estados Unidos: John Wiley & Sons, Inc. 2005. p. 3 - 56

[2] http://www.orafaq.com/wiki/Predicate – 23/01/2012

[3] S. Sumathi, S. Esakkirajan. Fundamentals of Relational Database Management Systems. Nueva York, Estados Unidos: Springer-verlag. 2007. p. 569.

[4] http://www.oracle-base.com/articles/misc/DirectAndAsynchronousIO.php – 24/07/11

[5] J Hellerstein, M. Stonebraker. Readings In Database Systems. Nueva York,. United States: MIT Press. 2005. P 23.

[6] http://webdocs.caspur.it/ibm/udb-6.1/db2d0/bufpool.htm – 02/09/11

[7] http://download.oracle.com/docs/cd/E14072_01/server. 112/e10713/indexiot.htm – 10/12/11

[8] E. Rivero, C. Guardia, J. Reig. Bases de datos relacionales: Diseño físico (orientado al DB2 para z/OS de IBM). Madrid. España. R.B. Servicios Editoriales S.L.2004. p 557.

[9] J. Gennick. Oracle SQL*Plus: The Definitive Guide. Washington. Estados Unidos: O'reilly media.2005.p 391.

[10] http://msdn.microsoft.com/en-us/library/ms178071.aspx – 14/01/12.