J. Ruiz / J. Eng. Technol. Vol.3, N°2. (2014) - ISSN: 2256-3903

Artículo Original

Procesamiento y optimización de consultas

Jonathan Ruiz Rangel[1]

Artículo recibido: 1 de marzo de 2014 / Artículo aceptado: 12 de mayo de 2104

RESUMEN

Actualmente es común que las grandes empresas tengan enormes bases de datos, por tal motivo es normal que sus consultas (query) involucren una gran cantidad de información, con su consiguiente tiempo de respuesta por parte del Sistema Manejador de Bases de Datos (SMBD), en las que muchas veces no satisfacen los tiempos de respuesta que se quieren alcanzar en nuestro sistema de información, ya que el tiempo se ha convertido en una variable crítica que afecta el entorno en que se desenvuelve dicha institución, añadiendo el constante cambio del mismo. Por ende, los DBA (Administradores de Bases de Datos) deben tener un amplio conocimiento sobre cómo el SMBD realiza acciones sobre las consultas, el cálculo del costo y la optimización de su ejecución con el fin de completar el diseño físico de una base de datos relacional. Este artículo presenta el procesamiento y optimación de consultas como algunas pautas para mejorar el acceso a los datos en el área de Bases de Datos (BD).

Palabras clave: consultas, bases de datos, optimizador, parsing, traductor, algoritmos de búsqueda, árbol canónico, plan de ejecución.

1 Ingeniero de Sistemas. Magíster en Ingeniería de Sistemas. Universidad Simón Bolívar-Barranquilla, Colombia. Programa de Maestría en Ingeniería de Sistemas y Computación, Universidad del Norte, Barranquilla. Correspondencia: jruizrangel316@hotmail.com

Processing and optimization of queries

ABSTRACT

Nowadays, it is common that big companies have huge databases. Therefore, it is normal that queries involve great amounts of information and that this means a time lapse to respond for the Data Base Managing System, DBMS, within which, sometimes, the time of response is not satisfying for the goals desired in the information system under these terms. Time has become a crucial variable that affects the environment in which this institution works, and this includes the constant change of that variable. Therefore, database managers must have a wide knowledge about how the DBMS acts on the queries, the calculation of their costs and the optimization of their execution, in order to complete the physical design of a relational database. This article introduces the processing and the optimization of queries as standards for improving the access to data in the database area.

Key words: queries, databases, optimizer, parsing, translator, searching algorithms, canonical tree, execution plan.

Processamento e otimização de consultas

RESUMO

Atualmente é comum que as grandes empresas tenham enormes bases de dados, por tal motivo é normal que suas consultas (query) envolvam uma grande quantidade de informação, com seu imediato tempo de resposta por parte do Sistema Manejador de Bases de Dados (SMBD), nas que muitas vezes não satisfazem os tempos de resposta que se querem alcançar no nosso sistema de informação, já que o tempo se há convertido em uma variável crítica que afeta o entorno em que se desenvolve dita instituição, adicionando a constante mudança do mesmo. Por conseguinte, os DBA (Administradores de Bases de Dados) devem ter um amplo conhecimento sobre como o SMBD realiza ações sobre as consultas, o cálculo do custo e a otimização de sua execução com o fim de completar o desenho físico de uma base de dados relacional. Este artigo apresenta o processamento e melhoramento de consultas como algumas pautas para melhorar o acesso aos dados na área de Bases de Dados (BD).

Palavras chave: consultas, bases de dados, melhoramento, parsing, tradutor, algoritmos de busca, árvore canónico, plano de execução.

INTRODUCTION

El presente apunte está orientado a dar una visión práctica del procesamiento y optimización de consultas sobre bases de datos relacionales. Además de dar una idea básica de las principales acciones que realiza un motor de base de datos relacionales para procesar una consulta, se brinda un panorama simplificado de las diferentes maneras de resolver las consultas, del cálculo de sus costos y de la optimización de su ejecución.

Si bien cada motor de base de datos puede tener sus particularidades, se entiende que lo que aquí se plantea da una base razonable para permitir su comprensión.

El análisis de costos de la ejecución de determinadas sentencias SQL (sobre todo algunas que sean críticas o de uso muy frecuente) es importante en determinadas situaciones para completar el diseño físico de una base de datos relacional. Si bien este análisis puede abarcar todo tipo de sentencias DML, el enfoque de este apunte está orientado solo al procesamiento de las consultas SQL.

Descripción general del procesamiento de consultas

Procesamiento de Consultas

Los lenguajes de consultas relacionales (como el SQL) nos dan una interfaz declarativa de alto nivel para acceder a los datos almacenados en una base de datos.

El procesamiento de consultas hace referencia a la serie de actividades implicadas en la extracción de datos de una base de datos. Estas actividades incluyen la traducción de consultas expresadas en lenguajes de bases de datos de alto nivel en expresiones implementadas en el nivel físico del sistema, así como transformaciones de optimización de consultas y su evaluación real [1].

Los pasos básicos son:

1. *Parsing* y traducción
2. Optimización
3. Generación de código
4. Ejecución de la consulta.

Estos pasos en general son realizados por diferentes componentes del motor. Los componentes clave son: el optimizador de consultas y el procesador de consultas.

La idea general es que luego del parsing se construya una expresión algebraica equivalente (o podría ser más de una), se analicen diferentes formas de resolver la consulta (estas formas se llaman planes de ejecución) evaluando sus costos, y se seleccione la más eficiente.

Esto lo hace el componente generalmente llamado Optimizador de Consultas. Luego, esta consulta es pasada al otro componente generalmente llamado Procesador de consultas, que es el que se encarga de ejecutar físicamente la consulta de acuerdo al plan de ejecución, produciendo y devolviendo el resultado, como lo ilustra la Figura 1.

Introducción al optimizador de consultas

Para iniciar a definir los componentes se debe definir qué es optimización y qué es un optimizador. Se tomará la siguiente definición

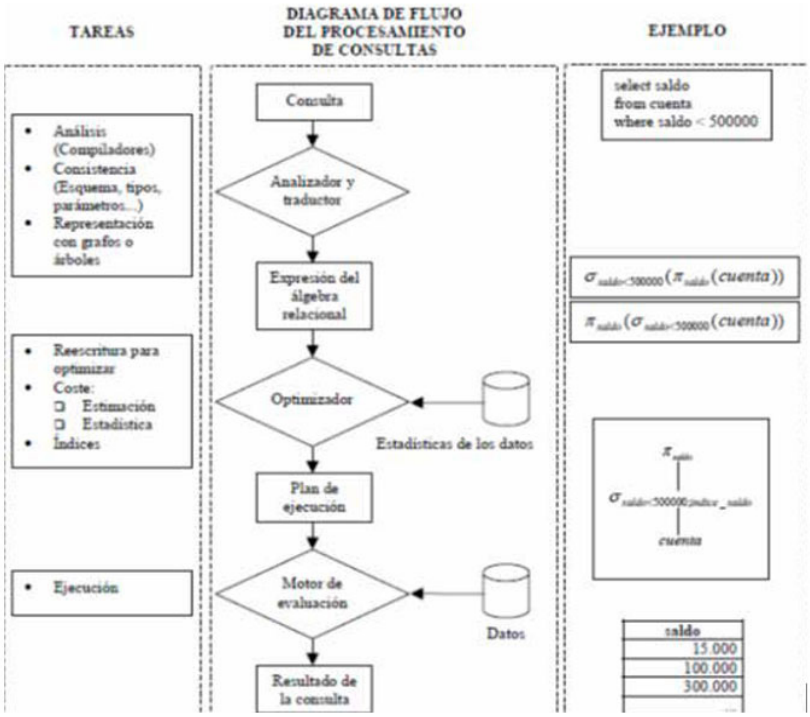

Figura 1. Visión general del procesamiento de consulta

indica cuál es el resultado que desea obtener y no el camino de acceso en la base de datos para llegar a dicho resultado. Esta "navegación automática" permite el desarrollo de sistemas que evalúen y mejoren las sentencias de consulta realizadas por los usuarios.

Debido a que esta posibilidad de mejora de especificación de consultas solo es posible en los SGBDR, algunos autores consideran que un sistema de bases de datos solo se puede considerar relacional si tiene optimizador.

por lo que se ajusta a la función del mismo en los distintos SMBD:

La optimización de consultas es el proceso por el cual se pretende mejorar los tiempos de respuesta en un sistema de gestión de bases de datos relacional [2].

Otra definición podría ser:

La optimización de consultas es el proceso de selección del plan de evaluación de las consultas más eficiente de entre las muchas estrategias generalmente disponibles para el procesamiento de una consulta dada, especialmente si la consulta es compleja [3].

Hay que tener en cuenta que los lenguajes de consulta relacionales son no procedimentales, es decir, que el usuario solo

Hay que tener en cuenta que el proceso de optimización tiene que evaluar no solamente cuál consulta es algebraicamente más correcta, sino también si dicha consulta no sobrecarga los recursos del sistema. Por tanto, la palabra "optimizador" no sería la más correcta (aunque es la más extendida), sino "planificador de consultas".

A partir de este hecho, podemos decir que el optimizador de consultas es el responsable de generar el plan, que va a ser el input del motor de ejecución. Debe ser un plan eficiente de ejecución de una consulta SQL, perteneciente al espacio de los posibles planes de ejecución de esa consulta.

La cantidad de posibles planes de ejecución para una consulta puede ser grande, ya sea porque algebraicamente se la puede

escribir de diferentes maneras lógicamente equivalentes, o porque hay más de un algoritmo disponible que implemente una expresión algebraica dada.

Como el procesamiento de cada plan de ejecución puede tener un rendimiento diferente, la tarea del optimizador es realmente importante a efectos de encontrar una buena solución.

Típicamente, a partir de una consulta en lenguaje SQL, construye una expresión en álgebra relacional de la siguiente manera:

• Realiza el producto cartesiano de las tablas del From de izquierda a derecha.
• Realiza un Select con las condiciones de la cláusula WHERE.
• Realiza una proyección con las columnas de la cláusula Select.

Una vez obtenida la expresión, pasa a armar un plan de ejecución de la consulta construyendo un árbol, el cual consta de expresiones de álgebra relacional, donde las relaciones son las hojas y los nodos internos, las operaciones algebraicas. Es importante notar que una misma consulta puede tener varios planes de ejecución (es decir, varios árboles pueden producir los mismos resultados).

En general, el objetivo principal es minimizar la cantidad de accesos a disco.

El optimizador construye diferentes planes basándose en:

• Los distintos algoritmos implementados en el procesador de consultas y disponibles al optimizador que implementan las operaciones de álgebra relacional (Proyección, Selección, Unión, Intersección, Resta, Join).

• Información acerca de:

▪ La estructura física de los datos (ordenamiento, clustering, hashing).
▪ La existencia de índices e información sobre ellos (ejemplo: número de niveles que posee).
▪ Estadísticas guardadas en el catálogo (esta información no está "al día" constantemente por razones de eficiencia, sino que se actualiza periódicamente):

- Tamaño de archivos y factor de bloqueo.
- Cantidad de tuplas de la relación.
- Cantidad de bloques que ocupa la relación.
- Cantidad de valores distintos de una columna (esto permite estimar la selectividad de una operación).

▪ Ciertas heurísticas que le permiten encontrar planes de ejecución sin necesidad de generar en forma completa el espacio de búsqueda (como podría ser tratar de armar planes que realicen cuanto antes las operaciones más restrictivas y maximicen el uso de índices y de las estructuras físicas existentes).

Una vez construidos los diferentes planes alternativos, el optimizador selecciona el que sea más eficiente, el cual es su *output*.

OPTIMIZACIÓN DE CONSULTAS

A. Optimización algebraica

Las optimizaciones de este tipo son aquellas que buscan mejorar la performance de la consulta independientemente de la

organización física. Involucran propiedades algebraicas que permiten construir una consulta equivalente a la original.

Algunas propiedades

Cascada de σ
σC1 and C2...and Cn(R)≡σC2(...σCn (R)...)

Conmutatividad de σ
σC1(σC2(R))≡σC2(σC1(R))

Cascada de ϖ
πlist1(πlist2(R))≡πlist1∩list1(R)

Conmutatividad de σ con respecto a ϖ
πA1, A2,...An (σC(R))≡σC(πA1,A2,...An (R))
Si C referencia solamente a atributos dentro de A1...An

Conmutatividad del Producto Cartesiano (o junta)
R x S ≡ S x R

Conmutatividad del σ con respecto al Producto Cartesiano (o junta)
σC(R X S)≡(σCr(R)) X (σCs(S)) donde C ≡ Cr□Cs

Conmutatividad del ϖ con respecto al Producto Cartesiano (o junta)
πL (R X S)≡(πL2 (R)) X (πL2(S)) donde L≡L2 □ L2

Conmutatividad de operaciones de conjuntos
(□ ϵ ∩) R θ S ≡ S θ R Si θ={□ ϵ ∩} Si

Asociatividad del Producto Cartesiano, junta (□ ϵ ∩)
(R θ S) θ T ≡ R θ (S θ T) Si θ = {x,|x|,□ ϵ ∩}

B. Algunas heurísticas aplicables

Los optimizadores de consultas utilizan las reglas de equivalencia del álgebra relacional para mejorar, en la medida de lo posible, el rendimiento esperado de una consulta dada.

A continuación enumeramos algunas heurísticas aplicables tales como:

1) Considerar solo árboles sesgados a izquierda: Al analizar los posibles árboles candidatos, reducir el espacio de búsqueda considerando solo árboles sesgados a izquierda, es decir, árboles donde los sucesores derechos de cualquier nodo sean hojas.

2) Descomponer las selecciones conjuntivas: en una secuencia de selecciones simples formadas por cada uno de los términos de la selección original.

3) Llevar las selecciones lo más cercano posible a las hojas del árbol: de esta manera se logra la ejecución temprana reduciendo así el número de tuplas que se propagan hacia niveles superiores.

4) Reemplazar los productos cartesianos seguidos de selecciones por *joins*. Evitar en la medida de lo posible los productos cartesianos: Si se tiene un producto cartesiano seguido de una selección con un predicado p sobre atributos que determinan un *join* sobre ese predicado p, se descarta el árbol con las dos operaciones por separado.

De esta manera se evita propagar resultados intermedios muy voluminosos.

5) Descomponer las listas de atributos de las proyecciones y llevarlas lo más cercano posible a las hojas del árbol: Creando nuevas proyecciones cuando sea posible y de esta manera no propagar hacia niveles superiores atributos innecesarios. Así se logra una reducción temprana del tamaño de las tuplas, y se reduce la cantidad de bloques necesaria para almacenamiento intermedio.

6) Realizar primero los joins más selectivos: De esta manera se reduce el tamaño de los resultados intermedios.

7) Utilizar como outer (externas) las relaciones más selectivas: Al planear los árboles, tender a utilizar como relación outer de los joins a aquellas que sean más selectivas, es decir, aquellas en que su selectividad sea menor.

8) En cada nodo, retener los planes menos costosos, pero considerar también los órdenes interesantes: Al analizar el costo de un nodo intermedio, retener para niveles superiores del árbol los subplanes de menor costo. En caso de que haya alguno cuyo resultado intermedio esté en algún orden interesante, retener además el de menor costo para ese orden.

9) Tener en cuenta los índices interesantes al momento de generar los planes de ejecución: En muchas ocasiones puede ser importante no "bajar" al máximo las proyecciones y/o selecciones sobre una relación, y de esta manera poder aprovechar el índice de la relación en alguna operación de un nivel superior, y aplicar recién el filtro y/o la proyección luego de esta operación.

10) Utilizar el pipeline entre las operaciones siempre que sea posible: De esta manera se evitan los costos adicionales a causa de dar persistencia a disco a resultados intermedios en forma innecesaria.

C. Pasos para la optimización

Los pasos que se seguirán para la optimización de una consulta son los siguientes:

1. Construir el árbol canónico.

2. Construir árboles equivalentes alternativos, utilizando alguna heurística para que no se sobredimensione innecesariamente el espacio de búsqueda del mejor plan.

3. Para cada árbol construido, hacer tantos planes de ejecución como surjan de las diferentes combinaciones "interesantes" de reemplazar los operadores lógicos por operadores físicos o algoritmos, indicando en qué casos los resultados intermedios son candidatos y si algún resultado queda en un orden interesante.

4. Para cada plan concreto de ejecución, evaluar sus costos totales.

5. Elegir el plan que haya resultado con menor costo.

Como podemos observar, es importante que los DBA tengan un conocimiento del álgebra relacional en el momento de diseñar la consulta teniendo en cuenta el proceso interno que realiza el optimizador de consultas.

CONCLUSIÓN

La optimización de consultas es el corazón de un DBMS relacional. La optimización heurística es más eficiente de generar, pero no puede obtener el plano óptimo de consultas. La optimización basada en el costo confía en las estadísticas generadas sobre las relaciones. Hasta que la optimización de la consulta sea un hecho, el templar una consulta puede ser un factor de vida.

REFERENCIAS

[1, 2, 3] A. Silberschatz, H. F. Korth, y S. Sudarshan, *Fundamentos de bases de datos,* 4ta ed. McGraw-Hill.

[4] A. Silberschatz, H. Korth, y S. Sudarshan, *Fundamentos de bases de datos*, 3ra ed. McGraw-Hill.

[5] J. Ullman, *Principles of Database and Knowledge Base Systems*. Computer Science Press, 1988.

[6] R. Ramakrishnan, y J. Gehrke, *Data-base Management Systems*, 3ra ed. Mc- Graw-Hill.

[7] B. Manrique, F. J. Moreno, y J. A. Guzmán, "Optimización de algoritmos de join relacional para tecnologías de bases de datos", Universidad Nacional de Colombia Sede Medellín.

[8] Oracle® Database Performance Tuning Guide 10g Release 1 (10.1) Part No. B10752-01.

[9] R. S. G. Lanzelottel y P. Valduriez, *Extending the search strategy in a query optimizer*, 78153 - Le Chesnay cedex – France.

[10] I. F. Ilyas, J. Rao, G. Lohman, D. Gao, y E. Lin, "Estimating compilation time of a query optimizer".