Optimización de Consultas Múltiples: Una Introducción

Andrés Felipe Bustamante García

***Resumen*—El problema de la optimización de consultas múltiples es un problema emergente de la necesidad de procesar grandes cantidades de información en las bases de datos, que se fue haciendo cada vez más notorio a medida que los avances tecnológicos relacionados con capacidad de procesamiento y almacenamiento fueron creciendo, y las exigencias para recuperación de datos iban siendo aún mayores. Varios algoritmos y teorías se han propuesto para dar solución a este problema, y cada una de ellas con cierto grado de optimalidad por encima de otras de la misma clase. En este documento se muestran algunas de las más importantes soluciones entregadas por los teóricos del tema, que han dado pie a las implementaciones que utilizan los más populares sistemas gestores de bases de datos en la actualidad.**

***Index Terms*—optimización de consultas múltiples, sistemas gestores de bases de datos, introducción**

I. Introducción

EXISTEN varios escenarios en los sistemas de bases de datos relacionales, así como en los sistemas de bases de datos deductivas, en que se presentan casos donde debe darse respuesta de forma óptima a una consulta compleja, es decir, una consulta se puede interpretar como un conjunto de consultas simples. Uno de estos escenarios es en el área de las bases de datos deductivas, donde una consulta simple de un usuario a la base de datos puede ser procesada como un conjunto de múltiples consultas, ya que un predicado puede tener varias definiciones, como lo menciona Sellis en [8]. El segundo escenario es aquel en donde consultas de varios usuarios se procesan en serie y algunas de esas consultas comparten relaciones de la base de datos. También existe el escenario en donde se procesa una consulta de tipo recursivo dividiéndola en múltiples consultas más pequeñas para simplificar el procesamiento.

En todos y cada uno de estos escenarios se presenta el problema de la optimización de consultas múltiples. En la siguiente sección de este documento se dará una explicación detallada de este problema, y en las secciones siguientes se mostrarán algunas de las formas de atacar el problema que se han dado en los años posteriores a la enunciación del problema, por parte de los investigadores del área. Por último, se presentan algunas conclusiones con base en el análisis de las soluciones entregadas.

Andrés Bustamante es estudiante de la Maestría en Ingeniería de Sistemas de la Universidad Nacional de Colombia, Sede Bogotá. e-mail: afbustamanteg@unal.edu.co

Manuscrito recibido el 25 de Marzo de 2008

II. El problema de la optimización de consultas múltiples

Se tiene una base de datos que consta de un conjunto de relaciones $\{R_1, R_2, \ldots, R_m\}$. Se tiene también un conjunto de consultas $Q = \{Q_1, Q_2, \ldots, Q_n\}$ sobre varias de las relaciones de la base de datos.

Se busca de este conjunto de consultas un plan óptimo de acceso global, que permita hacer las recuperaciones de información necesarias para satisfacer cada una de las consultas en Q con el menor costo computacional posible (costo de CPU, costo de entrada y salida de datos, etc.). Este plan de acceso puede verse como una mezcla de los planes individuales de acceso seleccionados para cada una de las consultas del conjunto. Cuando cada uno de estos planes individuales representa un costo mínimo dentro de los planes estimados para la misma consulta, a cada plan se le denomina *plan óptimo local*, y cuando un plan global representa el mínimo costo entre todos los planes globales determinados como opcionados para un conjunto de consultas, se le denomina *plan óptimo global*.

Tomando la definición formal de Sellis, dado un conjunto de n de planes de acceso $\{P_1, P_2, \ldots, P_n\}$, con $P_i = \{P_{i1}, P_{i2}, \ldots, P_{ik}\}$ como el conjunto de planes posibles para resolver Q_i, donde $0 \leq i \leq n$; se quiere encontrar un plan de acceso global PG a partir de cada P_i, tal que el costo de PG es mínimo.

Con base en esta definición, este problema se convierte algorítmicamente en un problema de tipo no-polinomial duro (NP-hard), ya que entre otras cosas, habría que generar todas las posibles combinaciones de planes entre todas las consultas de forma que al hacer una suma de costos de todos los planes, esta suma sea menor o igual a una constante K correspondiente al costo mínimo parcial de los planes de acceso globales. Adicionalmente, los cálculos realizados en cada uno de estos planes no tienen en cuenta subexpresiones comunes entre las consultas, por lo que se pueden presentar cálculos repetidos. Una demostración más detallada de la complejidad de este problema se muestra en [8].

III. Soluciones de cara a la optimización

El problema de la optimización de consultas múltiples se puede dividir en dos partes, como menciona Park et al. en [1]: la primera parte es la optimización sobre la identificación de subexpresiones comunes entre las consultas, para así comprobar posibles beneficios de compartir los datos; la segunda parte hace referencia a optimizar la búsqueda de un plan global de acceso para un conjunto de consultas, de forma que el costo total de procesamiento de los subplanes sea mínimo.

Varias de las soluciones propuestas hasta ahora han atacado principalmente el problema de tener subexpresiones comunes entre las consultas, que en caso de ser resuelto, puede representar importantes ganancias en cantidad de cálculos, lo cual finalmente se ve representado en la función de tiempo del algoritmo. Sin embargo, el enfoque de las soluciones ha sido el de materializar los cálculos de las subexpresiones comunes, lo cual es conveniente si se tienen costos despreciables a la hora de leer y escribir en memoria, que se va presentando en mayor escala con sistemas de cómputo más avanzados. Aún así, dependiendo de la cantidad de datos a procesar, puede llegar el momento en que se cope el espacio de memoria reservado para los resultados intermedios de estos cálculos.

En las próximas subsecciones se presentan algunas formas de resolver el problema más grande, que es el de la optimización de consultas múltiples, así como los subproblemas emergentes como el problema de utilización de espacios de memoria, recién mencionado.

III-A. Algoritmos heurísticos

Sellis en [7] propone un algoritmo heurístico con el objetivo de encontrar un plan óptimo global, principalmente, antes que identificar subexpresiones comunes entre las consultas presentadas por el usuario. Este algoritmo realiza una búsqueda sobre un espacio de estados definido sobre los planes de acceso.

Un estado, para este caso, se define como un conjunto de planes de acceso, conformado a partir de planes individuales de acceso para cada consulta del conjunto múltiple a procesar.

El espacio de búsqueda de este algoritmo, se construye definiendo un estado por cada combinación posible de planes entre las consultas. El algoritmo recorre este espacio de búsqueda para encontrar la forma menos costosa de ir de un estado inicial, que por defecto es vacío, a un estado final. En este transcurso, desde que se tiene un estado inicial vacío, hasta un estado inicial completo, se van considerando una a una las consultas solicitadas por los usuarios, y de cada una de ellas, se agrega uno de los posibles planes a la descripción del estado.

Sin embargo, para que el algoritmo sea eficiente, debe definirse una buena función de convergencia para un límite inferior, de forma que se minimice el espacio de búsqueda a explorar, ya que en el peor de los casos pueden presentarse tiempos exponenciales. Este límite inferior determina el costo actual de procesar las consultas.

En [8] se hace una mejora sobre este algoritmo, por una parte para reducir aún más el espacio de búsqueda de la solución, agregando un límite superior asociado a una nueva función de convergencia, además del límite inferior ya existente. Este límite superior se deduce de una mezcla entre los planes de acceso de cada P_i. También se propone ordenar las consultas, ya que la función de convergencia que determina el límite inferior depende de este ordenamiento, lo cual finalmente va a repercutir en la complejidad del algoritmo.

En los resultados experimentales realizados sobre el algoritmo en la misma referencia, se evidencia que el algoritmo puede tener una complejidad exponencial de acuerdo al número de planes determinados por consulta, controlada por la función de convergencia del límite inferior. Se observó que las ganancias obtenidas con la aplicación del algoritmo, es decir, los ahorros en cálculos computacionales, se incrementaban cuando se encontraban resultados intermedios compartidos. Sin embargo, estos resultados se obtuvieron sobre la base de que se tienen siempre tareas iguales en cada uno de los planes. Adicionalmente, este modelo no se probó aún en una aplicación de la vida real, por lo que no se puede dar una versión real del desempeño del algoritmo en producción.

III-B. Programación dinámica

Park et al. [1] ataca también el enfoque de encontrar un plan óptimo global, dejando el trabajo de las subexpresiones a los optimizadores de consultas ya existentes, partiendo de la base del uso de subexpresiones comunes determinadas a partir de de un análisis beneficio-costo.

La propuesta de la investigación en mención, se centra en un algoritmo de programación dinámica para la determinación del plan global, teniendo consideraciones sobre tareas con relaciones idénticas, así como tareas que se interrelacionan con implicaciones, es decir, el caso en que una tarea T_4 implique la ejecución de otra tarea T_1. Hay que recordar que este aspecto no se tuvo en cuenta en el algoritmo de Sellis en [8].

Este algoritmo parte del modelamiento de los planes de acceso y sus tareas en un grafo no-dirigido, como el de la figura 1, donde los nodos son los planes y las tareas de los planes.

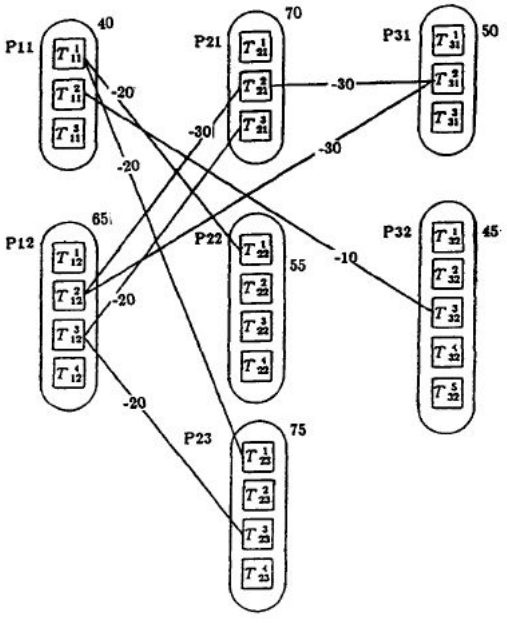

Figura 1. Grafo de distribución de tareas y planes de acceso del algoritmo de programación dinámica de Park. En este ejemplo las tareas tienen relaciones idénticas.

Los resultados experimentales mostrados para este algoritmo muestran una evidente reducción en la cantidad de pasos y de nodos a utilizar para llegar a un plan óptimo global, en comparación con el algoritmo presentado por Sellis la primera vez en 1986 para la Conferencia Internacional de Gestión de Datos ACM-SIGMOD. Sin embargo, tampoco

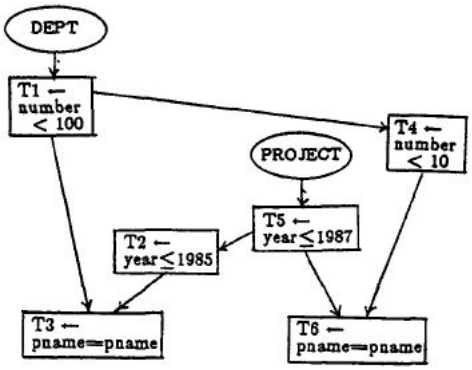

Figura 2. Otro ejemplo de distribución de tareas y planes de acceso del algoritmo de programación dinámica de Park. Este ejemplo clásico de un modelo de una compañía muestra algunas tareas de los planes con implicaciones entre ellas mismas.

se hicieron pruebas sobre entornos de producción reales y con grandes volúmenes de información. La propuesta fue de nuevo meramente teórica, y quedó abierta la posibilidad de investigación en el análisis de desempeño de los algoritmos.

III-C. Pipelines

Las técnicas propuestas hasta ahora para resolver el problema que propone este documento, han tratado el caso en que se comparten operaciones, relaciones o resultados intermedios. Sin embargo, ninguna de ellas aprovecha el concepto de trabajo en paralelo, a modo de líneas de ensamblaje, donde mientras se completan algunas tareas u operaciones, hay otro componente computacional encargado de realizar otras tareas, para luego generar un resultado consolidado con las salidas de las tareas programadas. A esto es lo que se conoce como *pipelining*.

Una primera aproximación a este concepto fue entregada por Tan et al. en [2], donde se explotan las ventajas de aplicar *pipelining* en dos escenarios: las relaciones compartidas y, los resultados intermedios compartidos. Generando una programación inteligente de las consultas que se requieren procesar, se pueden explotar dichas relaciones y resultados compartidos. Sin embargo, es una propuesta desarrollada sobre consultas con *joins* múltiples.

Esta propuesta también se desarrolla sobre la base de dos fases, donde en una primera instancia se seleccionan por medio de los optimizadores de consultas simples, los mejores planes de acceso para cada consulta individual. Cada uno de estos planes se puede representar como un árbol segmentado, como el que se ilustra en la figura 3. En la segunda fase, se mezclan o integran los planes de consulta de la primera fase, mezcla en la cual se ejecutan dos pasos: el primero, consiste en generar un grafo de comparticiones que refleje las comparticiones de datos entre los planes; el segundo consiste en generar un plan global, a partir del recorrido del grafo de comparticiones.

Más recientemente, Dalvi et al. en [3] hace una crítica de este enfoque, ya que todas las operaciones que se realizan, y principalmente los modelos de los planes de optimización son residentes en memoria, y nunca se materializan, de forma que es una información primero que todo, volátil, además de

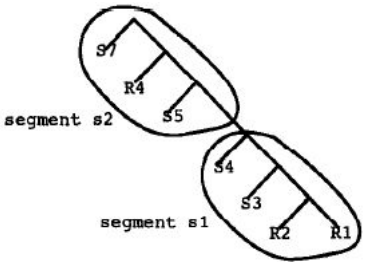

Figura 3. Grafo representativo del plan de acceso para una consulta por la técnica del *pipelining* de Tan y Lun. Es de notarse que el grafo siempre es un árbol segmentado y profundo a la derecha.

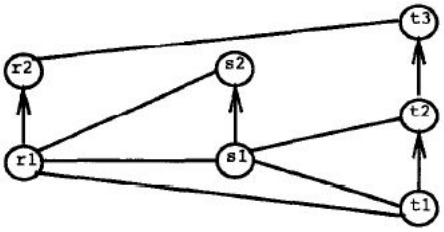

Figura 4. En la técnica del *pipelining* de Tan y Lun se tiene un grafo de comparticiones de los datos similar a este. Allí se representan relaciones, tareas y segmentos de los planes.

la limitante poco despreciable de la capacidad en memoria, especialmente en bases de datos muy grandes. Por otra parte, menciona también otro aspecto importante de esta técnica, y es que los planes de acceso a las consultas se pueden representar únicamente como árboles profundos a la derecha, además del uso restringido de operaciones a solamente *joins*.

En [3] se hace entonces una nueva propuesta algorítmica sobre la misma técnica, con las mismas dos fases, pero con la diferencia que en la segunda fase se hace la optimización de un *pipelining* para un plan dado. Adicionalmente, las entradas del algoritmo son para este caso grafos dirigidos y acíclicos.

Los autores de esta propuesta mencionan que se puede utilizar la técnica sin afectar la complejidad algorítmica de la solución, pero finalmente, pueden generar una alteración en términos de tiempo, aún bajo el mismo orden. Reconocen además, que de nuevo se parten de bases teóricas para justificar parte de la ganancia en rendimiento, considerando despreciables los gastos computacionales que tiene ejecutar las dos fases mencionadas en cuando a memoria y flujo de entrada/salida del disco. También queda abierta la posibilidad de hacer pruebas a este algoritmo sobre bases de datos con grandes volúmenes de información, todo esto encaminado a medir con escalas reales, los beneficios de esta técnica.

III-D. Otras perspectivas

Dalvi et al. en [3] hace un pequeño estado del arte de varias perspectivas tomadas con respecto al problema de la optimización de consultas múltiples. Una primera perspectiva nace de Hall en [4], donde se utiliza una aproximación a la solución en dos fases, donde se parte de obtiene un plan global inicial a partir de los optimizadores de consultas ya existentes, para luego identificar subexpresiones comunes, y por medio de un algoritmo heurístico, se seleccionan aquellas

subexpresiones que vale la pena materializar o persistir, y también compartir.

Se menciona también, que otras aproximaciones, como las que se presentan en [5], [6], [9] revisan cómo hacer optimización seleccionando subexpresiones para una materialización transiente, es decir, en un intervalo temporal corto. Se han hecho desarrollos también en expresiones que se pueden compartir para un determinado plan, así como en las ventajas que puede tener compartir subexpresiones y resultados de exámenes sobre los datos.

La mayoría de estas aproximaciones se centran, ahora sí, en la optimización del uso de las subexpresiones en los planes de acceso a las consultas, por encima de la optimización directamente sobre el plan global de acceso que reúne a los demás a nivel individual.

IV. Conclusiones

El problema de la optimización de múltiples consultas es un problema motivado por la creciente demanda de recuperación de información en grandes sistemas de almacenamiento de datos. A medida que las capacidades de almacenamiento han ido evolucionando, se ha ido evidenciando la necesidad de diseñar soluciones más eficientes para el procesamiento de este tipo de consultas.

Entre estas soluciones se mostraron algunas de las más sobresalientes, no solo a nivel algorítmico, sino a nivel conceptual, que puede abrir campo a nuevas mejoras en el futuro. Es claro que en muchas de estas técnicas falta tener un acercamiento a índices de mejora de un algoritmo con respecto al otro, pero con grandes volúmenes de datos, que es una pregunta que ya se deben haber hecho los diseñadores de los sistemas gestores de bases de datos comerciales más importantes, y a nivel interno debe haber arrojado algún valor al menos estimativo, para incluir una u otra o una mezcla de las soluciones aportadas a este problema.

La mayoría de técnicas se pueden clasificar en dos grandes grupos: aquellas técnicas o algoritmos que se centran en la búsqueda de subexpresiones comunes y la optimización sobre ellas, para luego establecer un plan óptimo global; por otra parte, están las propuestas algorítmicas para comenzar optimizando el plan de acceso global, reutilizando los optimizadores de consultas simples para las subexpresiones. Esto se puede ver como una división de enfoques en *down-top* para el primer caso, y *top-down* para el último caso. Ninguna técnica de las que se explicaron en este documento atacó la optimización en ambos frentes, dada la alta complejidad que tienen estas partes para resolver la optimización de más alto nivel.

Referencias

[1] J. Park y A. Segev. *Using common subexpressions to optimize multiple queries*. Proceedings of the Fourth International Conference on Data Engineering, p. 311 - 319, 1988.

[2] K. Tun y H. Lu. *Workload scheduling for multiple query processing*. Information Processing Letters, Vol. 55. No. 5, p. 251 - 257, 1995.

[3] N.N. Dalvi, S. Sanghai, P. Roy, y S. Sudarshan. *Pipelining in multi-query optimization*. Journal of Computer and System Sciences, Vol. 66, No. 4, p. 728 - 762, 2003.

[4] P.A.V. Hall. *Optimization of single expressions in a relational data base system*, IBM J. Res. Dev. Vol. 20, No. 3, 1976, p. 244 - 257

[5] P. Roy, S. Seshadri, S. Sudarshan, S. Bhobhe. *Efficient and extensible algorithms for multi-query optimization*. SIGMOD International Conference on Management of Data, Dallas, Texas, Estados Unidos, 2000, p. 249 - 260.

[6] S.N. Subramanian, S. Venkataraman. *Cost based optimization of decision support queries using transient views*. SIGMOD International Conference on Management of Data, Seattle, Washington, Estados Unidos, 1998, p. 319 - 330.

[7] T. Sellis. *Multiple-Query Optimization*, ACM Transactions on Database Systems, Vol. 13, p. 23 - 52, Marzo de 1988.

[8] T. Sellis y S. Ghosh, *On the Multiple-Query Optimization Problem*, IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 2, p. 262 - 266, Junio de 1990.

[9] Y. Zhao, P. Deshpande, J.F.Ñaughton, A. Shukla. *Simultaneous optimization and evaluation of multiple dimensional queries*. SIGMOD International Conference on Management of Data, Seattle, Washington, Estados Unidos, 1998, p. 271 - 282.