1

BASES DE DATOS DISTRIBUIDAS

TEMA 3

PROFESOR: M.C. ALEJANDRO GUTIÉRREZ DÍAZ

2

3. PROCESAMIENTO DE CONSULTAS DISTRIBUIDAS

3.1 Metodología del procesamiento de consultas distribuidas
3.2 Estrategias de procesamiento de consultas distribuidas
3.3 Optimización de consultas distribuidas
3.4 Optimización global

3

El procesamiento de consultas es de suma importancia en bases de datos centralizadas. Sin embargo, en BDD éste adquiere una relevancia mayor.

El objetivo es convertir transacciones de usuario en instrucciones para manipulación de datos. No obstante, el orden en que se realizan las transacciones afecta grandemente la velocidad de respuesta del sistema. Así, el procesamiento de consultas presenta un problema de optimización en el cual se determina el orden en el cual se hace la menor cantidad de operaciones. Este problema de optimización es NP-difícil, por lo que en tiempos razonables solo se pueden obtener soluciones aproximadas.

4

En BDD se tiene que considerar el procesamiento local de una consulta junto con el costo de transmisión de información al lugar en donde se solicitó la consulta.

El éxito creciente de la tecnología de bases de datos relacionales en el procesamiento de datos se debe, en parte, a la disponibilidad de lenguajes los cuales pueden mejorar significativamente el desarrollo de aplicaciones y la productividad del usuario final.

5

Ocultando los detalles de bajo nivel acerca de la localización física de datos, los lenguajes de bases de datos relacionales permiten la expresión de consultas complejas en una forma concisa y simple. Particularmente, para construir la respuesta a una consulta, el usuario no tiene que especificar de manera precisa el procedimiento que se debe seguir. Este procedimiento es llevado a cabo por un módulo del DBMS llamado el *procesador de consultas* (query processor).

Dado que la ejecución de consultas es un aspecto crítica en el rendimiento de un DBMS, el procesamiento de consultas ha recibido una gran atención tanto para bases de datos centralizadas como distribuidas. Sin embargo, el procesamiento de consultas es mucho más difícil en ambientes distribuidos que en centralizados, ya que existe un gran

número de parámetros que afectan el rendimiento de las consultas distribuidas. En este capítulo revisaremos el procesamiento de consultas para bases de datos distribuidas.

La función principal de un procesador de consultas relacionales es transformar una consulta en una especificación de alto nivel, típicamente en cálculo relacional, a una consulta equivalente en una especificación de bajo nivel, típicamente alguna variación del álgebra relacional

La consulta de bajo nivel implementa de hecho la estrategia de ejecución para la consulta. La transformación debe ser correcta y eficiente. Es correcta si la consulta de bajo nivel tiene la misma semántica que la consulta original, esto es, si ambas consultas producen el mismo resultado.

6

El mapeo bien definido que se conoce entre el cálculo relacional y el álgebra relacional hace que la correctitud de la transformación sea fácil de verificar. Sin embargo, producir una estrategia de ejecución eficiente es mucho más complicado. Una consulta en el cálculo relacional puede tener muchas transformaciones correctas y equivalentes en el álgebra relacional.

Ya que cada estrategia de ejecución equivalente puede conducir a consumos de recursos de cómputo muy diferentes, la dificultad más importante es seleccionar la estrategia de ejecución que minimiza el consumo de recursos.

Ejemplo 4.1. Considere el siguiente subconjunto del esquema de la base de datos de ingeniería que se presentó en el capítulo 2

E(ENO, ENOMBRE, TITULO)

G(ENO, JNO, RESPONSABLE, JORNADA)

y la siguiente consulta de usuario:
"Encuentre todos los nombres de empleados que manejan un proyecto"
BASES DE DATOSDISTRIBUIDAS MIS 515
4

7

La expresión de la consulta en SQL se puede ver como
SELECT ENOMBRE
FROM E, G
WHERE E.ENO = G.ENO **AND** RESPONSABLE = "ADMINISTRADOR"
Dos consultas equivalentes en el álgebra relacional que son transformaciones correctas de la consulta en SQL son:

$$\prod_{ENAME}(\sigma_{RESP="ADMINSTRADOR" \wedge E.ENO=G.ENO}(E \times G))$$

y

$$\prod_{ENAME}(E \ltimes_{ENO} (\sigma_{RESP="ADMINSTRADOR"}(G)))$$

Como es intuitivamente obvio, la segunda estrategia que evita calcular el producto cartesiano entre E y G, consume mucho menos recursos que la primera y, por lo tanto, es mejor.
En un contexto centralizado, las estrategias de ejecución de consultas pueden ser bien expresadas como una extensión del álgebra relacional. Sin embargo, en sistemas distribuidos, el álgebra relacional no es suficiente para expresar la ejecución de estrategias. Debe ser complementada con operaciones para el intercambio de datos entre nodos diferentes.
Además de elegir el orden de las operaciones del álgebra relacional, el procesador de consultas distribuidas debe seleccionar también los mejores sitios para procesar datos y posiblemente la forma en que ellos tienen que ser transformados.
BASES DE DATOSDISTRIBUIDAS MIS 515
5

8

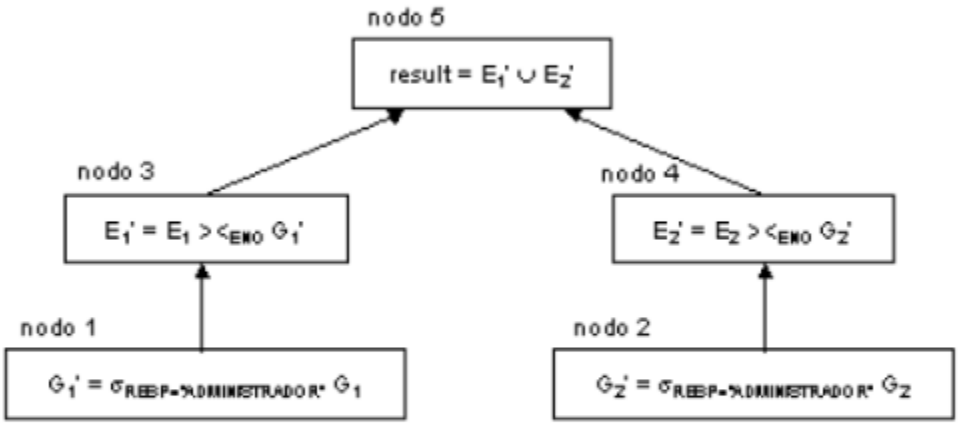
Ejemplo 4.2. Considere la siguiente consulta del Ejemplo ANTERIOR
Supongamos que las relaciones E y G están fragmentadas horizontalmente como sigue:
$E_1 = ó_{ENO \leq "E3"}(E)$
$E_2 = ó_{ENO > "E3"}(E)$

$G_1 = ó_{ENO \leq "E3"}(G)$
$G_2 = ó_{ENO > "E3"}(G)$
Los fragmentos E_1, E_2, G_1 y G_2 están almacenados en los nodos 1, 2, 3 y 4, respectivamente, y el resultado se quiere en el nodo 5. En la Figura 4.2 se presentan dos estrategias distribuidas de ejecución diferentes para la misma consulta (se ha ignorado el operador de proyección por simplicidad del ejemplo).
La estrategia A explota el hecho de que las relaciones E y G están fragmentadas de la misma manera y ejecuta la operación de selección y junta en paralelo.
La estrategia B centraliza todos los datos en el nodo resultante antes de procesar la consulta.
Para evaluar el consumo de recursos, se usará un modelo de costo simple. Suponga que el costo de acceso a un tupla (*tupacc*) es 1 unidad, y la transferencia de un tupla (*tuptrans*) tiene un costo de 10 unidades. Suponga que las relaciones E y G tienen 400 y 1000 tuplas, respectivamente, y que existen 20 administradores en la relación G. También, suponga que los datos están uniformemente distribuidos entre los nodos.
Finalmente, suponga que las relaciones G y E están agrupadas localmente en los atributos RESP y ENO, respectivamente, de manera que, hay un acceso directo a los tuplas de G y E, respectivamente.

9

El costo de la estrategia A se puede derivar como sigue:

1. Producir G' seleccionando G requiere 20**tupacc*	=	20
2. Transferir G' a los nodos de E requiere 20**tuptrans*	=	200
3. Producir E' juntando G' y E requiere (10*10)**tupacc**2	=	200
4. Transferir E' al nodo resultante requiere 20**tuptrans*	=	200
El costo total es		620



a) Estrategia A

b) Estrategia B

10

El costo de la estrategia B se puede derivar como sigue:

La estrategia A es mejor por un factor de 37, lo cual es significativo. La diferencia sería aún mayor si se hubiera asumido un costo de

comunicación mayor y/o un grado de fragmentación mayor.

11

Objetivos de la optimización de consultas

Como se estableció antes, el objetivo del procesamiento de consultas en un ambiente distribuido es transformar una consulta sobre una base de datos distribuida en una especificación de alto nivel a una estrategia de ejecución eficiente expresada en un lenguaje de bajo nivel sobre bases de datos locales.

Así, el problema de optimización de consultas es minimizar una función de costo tal que

función de costo total = *costo de* I/O + *costo de* CPU + *costo de comunicación*

Los diferentes factores pueden tener pesos diferentes dependiendo del ambiente distribuido en el que se trabaje. Por ejemplo, en las redes de área amplia (WAN), normalmente el costo de comunicación domina dado que hay una velocidad de comunicación relativamente baja, los canales están saturados y el trabajo adicional requerido por los protocolos de comunicación es considerable.

Así, los algoritmos diseñados para trabajar en una WAN, por lo general, ignoran los costos de CPU y de I/O. En redes de área local (LAN) el costo de comunicación no es tan dominante, así que se consideran los tres factores con pesos variables.

12

La complejidad de las operaciones del álgebra relacional

La complejidad de las operaciones del álgebra relacional afectan directamente su tiempo de ejecución y establecen algunos principios útiles al procesador de consultas. Esos principios pueden ayudar en elegir la estrategia de ejecución final. La forma más simple de definir la complejidad es en términos de la cardinalidad de las relaciones independientemente de los detalles de implementación tales como fragmentación y estructuras de almacenamiento. La Figura 4.3 presenta la complejidad de las operaciones unarias y binarias en el orden creciente de complejidad.

Operación Complejidad

Operación	Complejidad
Selección Proyección (sin eliminación de duplicados)	$O(n)$
Proyección (con eliminación de duplicados) Agrupación	$O(n*\log n)$
Junta Semijunta División Operadores sobre conjuntos	$O(n*\log n)$
Producto Cartesiano	$O(n^2)$

Esta simple mirada a la complejidad de las operaciones sugiere dos principios:
1. Dado que la complejidad es con base en las cardinalidades de las relaciones, las operaciones más selectivas que reducen las cardinalidades deben ser ejecutadas primero.
2. Las operaciones deben ser ordenadas en el orden de complejidad creciente de manera que el producto Cartesiano puede ser evitado o, al menos, ejecutado al final de la estrategia.

13

Caracterización de los procesadores de consultas

Es difícil evaluar y comparar procesadores de consultas para sistemas centralizados y distribuidos dado que ellos difieren en muchos aspectos. En esta sección se enumeran algunas características importantes de los procesadores de consultas que pueden ser usados como base para su comparación.

Tipo de optimización

El problema de optimización de consultas es altamente demandante en tiempo de ejecución y, en el caso general, es un problema de la clase *NP*. Así existen dos estrategias para su solución: búsqueda exhaustiva o

el uso de heurísticas. Los algoritmos de búsqueda exhaustiva tienen una complejidad combinatorial en el número de relaciones de la consulta. Obtienen la transformación óptima, pero sólo se aplican a consultas simples dado su tiempo de ejecución.
Por otro lado, los algoritmos heurísticos obtienen solo aproximaciones a la transformación óptima pero lo hacen en un tiempo de ejecución razonable. Las heurísticas más directas a aplicar son el agrupamiento de expresiones comunes para evitar el cálculo repetido de las mismas, aplicar primero las operaciones de selección y proyección, reemplazar una junta por una serie de semijuntas y reordenar operaciones para reducir el tamaño de las relaciones intermedias.

Granularidad de la optimización

Existen dos alternativas: considerar sólo una consulta a la vez o tratar de optimizar múltiples consultas. La primera alternativa no considera el uso de resultados comunes intermedios. En el segundo caso puede obtener transformaciones eficientes si las consultas son similares. Sin embargo, el espacio de decisión es mucho más amplio lo que afecta grandemente el tiempo de ejecución de la optimización.

Tiempo de optimización

Una consulta puede ser optimizada en tiempos diferentes con relación a tiempo de ejecución de la consulta. La optimización se puede realizar de manera *estática* antes de ejecutar la consulta o de forma *dinámica* durante la ejecución de la consulta. La optimización estática se hace en tiempo de compilación de la consulta. Así, el costo de la optimización puede ser amortizada sobre múltiples ejecuciones de la misma consulta. Durante la optimización de consultas dinámica la elección de la mejor operación siguiente se puede hacer basado en el conocimiento exacto de los resultados de las operaciones anteriores. Por tanto, se requiere tener estadísticas acerca del tamaño de los resultados intermedios para aplicar esta estrategia.
Un tercer enfoque, conocido como *híbrido*, utiliza básicamente un enfoque estático, pero se puede aplicar un enfoque dinámico cuando los tamaños de las relaciones estimados están alejados de los tamaños actuales.

Estadísticas

La efectividad de una optimización recae en las *estadísticas* de la base de datos. La optimización dinámica de consultas requiere de estadísticas para elegir las operaciones que deben realizarse primero. La optimización estática es aún más demandante ya que el tamaño de las relaciones intermedias también debe ser estimado basándose en estadísticas.
En bases de datos distribuidas las estadísticas para optimización de

consultas típicamente se relacionan a los fragmentos; la cardinalidad y el tamaño de los fragmentos son importantes así como el número de valores diferentes de los atributos. Para minimizar la probabilidad de error, estadísticas más detalladas tales como histogramas de valores de atributos se usan pagando un costo mayor por su manejo

Nodos de Decisión

Cuando se utiliza la optimización estática, un solo nodo o varios de ellos pueden participar en la selección de la estrategia a ser aplicada para ejecutar la consulta. La mayoría de los sistemas utilizan un enfoque centralizado para la toma de decisiones en el cual un solo lugar decide la estrategia a ejecutar. Sin embargo, el proceso de decisión puede ser distribuido entre varios nodos los cuales participan en la elaboración de la mejor estrategia.

El enfoque centralizado es simple, pero requiere tener conocimiento de la base de datos distribuida completa. El enfoque distribuido requiere solo de información local. Existen enfoques híbridos en donde un nodo determina una calendarización global de las operaciones de la estrategia y cada nodo optimiza las subconsultas locales.

Topología de la Red

Como se mencionó al principio, el tipo de red puede impactar severamente la función objetivo a optimizar para elegir la estrategia de ejecución. Por ejemplo, en redes de tipo WAN se sabe que en la función de costo el factor debido a las comunicaciones es dominante. Por lo tanto, se trata de crear una calendarización global basada en el costo de comunicación. A partir de ahí, se generan calendarizaciones locales de acuerdo a una optimización de consultas centralizada.

En redes de tipo LAN el costo de comunicación no es tan dominante. Sin embargo, se puede tomar ventaja de la comunicación "broadcast" que existe comúnmente en este tipo de redes para optimizar el procesamiento de las operaciones junta. Por otra parte, se han desarrollado algoritmos especiales para topologías específicas, como por ejemplo, la topología de estrella.

14.

Arquitectura del procesamiento de consultas

El problema de procesamiento de consultas se puede descomponer en varios subproblemas que corresponden a diferentes niveles.

En la Figura
4.4, se presenta un esquema por niveles genérico para el procesamiento de consultas. Para simplificar la discusión, suponga que se tiene un procesador de consultas estático semicentralizado en donde no se tienen fragmentos replicados.

Cuatro capas principales están involucradas en mapear una consulta a una base de datos distribuida en una secuencia optimizada de operaciones locales, cada una de ellas actuando en una base de datos local. Las cuatro capas principales son: *descomposición de consultas*, *localización de datos*, *optimización global de consultas* y *optimización local de consultas*. Las primeras tres se realizan en un nodo central usando información global. La cuarta capa se realiza en cada nodo local.

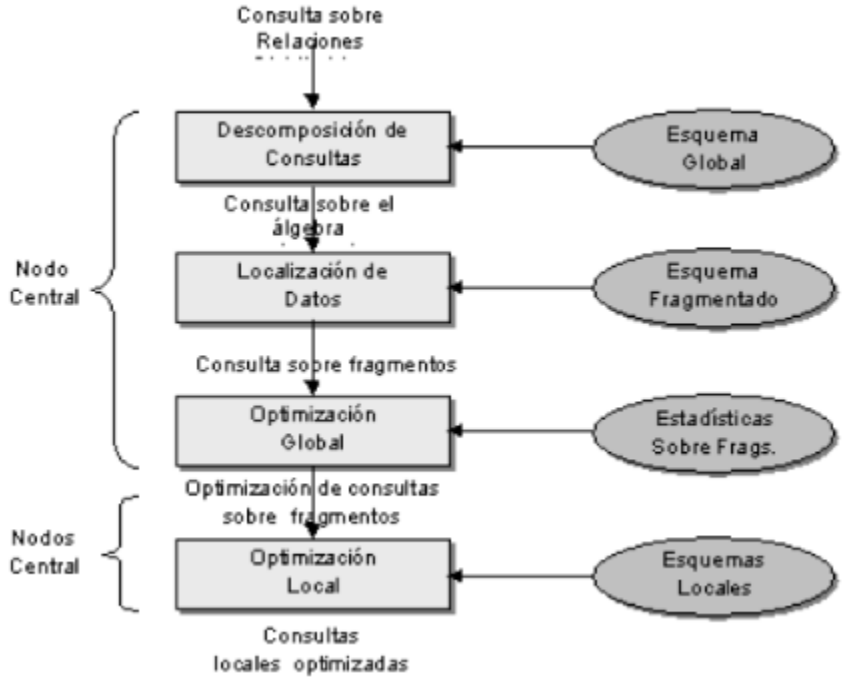


Figura 4.4. Arquitectura en capas del procesamiento de consultas.

Descomposición de consultas

La primera capa descompone una consulta en el cálculo relacional en una consulta en el álgebra relacional que opera sobre relaciones globales. Consiste de cuatro partes:

1. Normalización. Involucra la manipulación de los cuantificadores de la consulta y de los calificadores de la misma mediante la aplicación de la prioridad de los operadores lógicos.

2. Análisis. Se detecta y rechazan consultas semánticamente incorrectas.

3. Simplificación. Elimina predicados redundantes.

4. **Reestructuración**. Mediante reglas de transformación una consulta en el cálculo relacional se transforma a una en el álgebra relacional. Se sabe que puede existir más de una transformación. Por tanto, el enfoque seguido usualmente es empezar con una consulta algebraica y aplicar transformaciones para mejorarla.

Localización de Datos

La entrada a esta capa es una consulta algebraica definida sobre relaciones distribuidas. El objetivo de esta capa es localizar los datos de la consulta usando la información sobre la distribución de datos. Esta capa determina cuales fragmentos están involucrados en la consulta y transforma la consulta distribuida en una consulta sobre fragmentos.

Optimización Global de Consultas

Dada una consulta algebraica sobre fragmentos, el objetivo de esta capa es hallar una estrategia de ejecución para la consulta cercana a la óptima. La estrategia de ejecución para una consulta distribuida puede ser descrita con los operadores del álgebra relacional y con primitivas de comunicación para transferir datos entre nodos. Para encontrar una buena transformación se consideran las características de los fragmentos, tales como, sus cardinalidades. Un aspecto importante de la optimización de consultas es el ordenamiento de juntas, dado que algunas permutaciones de juntas dentro de la consulta pueden conducir a un mejoramiento de varios órdenes de magnitud. La salida de la capa de optimización global es una consulta algebraica optimizada con operación de comunicación incluidas sobre los fragmentos.

Optimización Local de Consultas

El trabajo de la última capa se efectúa en todos los nodos con fragmentos involucrados en la consulta. Cada subconsulta que se ejecuta en un nodo, llamada *consulta local*, es optimizada usando el esquema local del nodo. Hasta este momento, se pueden eligen los algoritmos para realizar las operaciones relacionales. La optimización local utiliza los algoritmos de sistemas centralizados.