3. INTRODUCCIÓN AL PROCESAMIENTO DE SENTENCIAS DISTRIBUIDAS

Índice

3.1. Procesamiento de consultas distribuidas

- En BDD el procesamiento de consultas es mucho más complicado debido al número de parámetros que afectan su rendimiento.
- El uso de SQL oculta los detalles en cuanto a que el usuario no especifica explícitamente el procedimiento para ejecutar la consulta.
- El procesamiento de una consulta se realiza a través del llamado ***procesador de consultas*** liberando entre otras cosas al programador de su optimización.

3.1.1. Objetivo del procesador distribuido de consultas

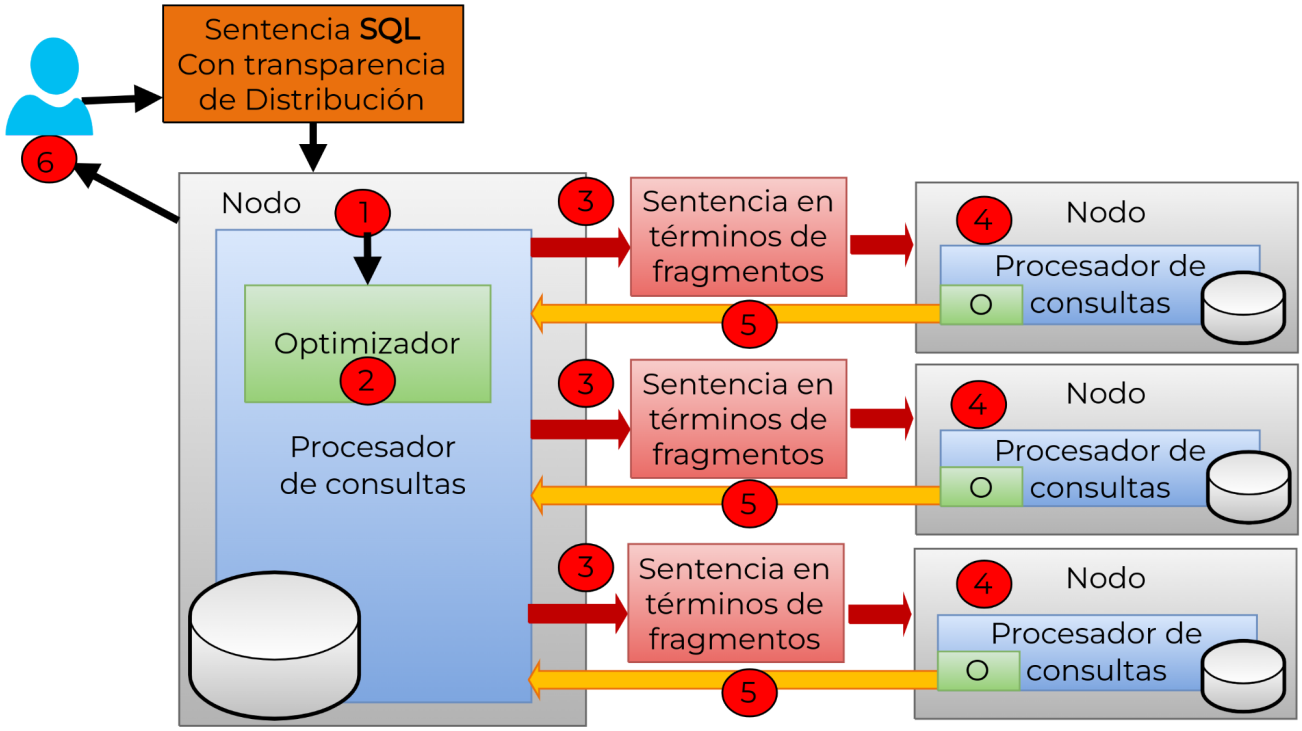
- Mapear o transformar una sentencia global en un conjunto de consultas de la forma más óptima posible en términos de fragmentos para ser procesados en cada sitio.
- Para lograr este mapeo se realizan las siguientes acciones:

A. El nodo origen (***driving node***) transforma la consulta global en términos de álgebra relacional: ***query algebraico***.
B. El nodo origen hace uso del optimizador para generar un plan de ejecución que permita procesar a la consulta distribuida con el menor costo posible.

C. El plan de ejecución incluye la descomposición de la consulta global en una serie de sentencias en términos de fragmentos que serán enviadas a los nodos para su ejecución: ***query algebraico fragmentado***.
D. El Optimizador trata de generar consultas algebraicas fragmentadas de la forma más óptima posible.

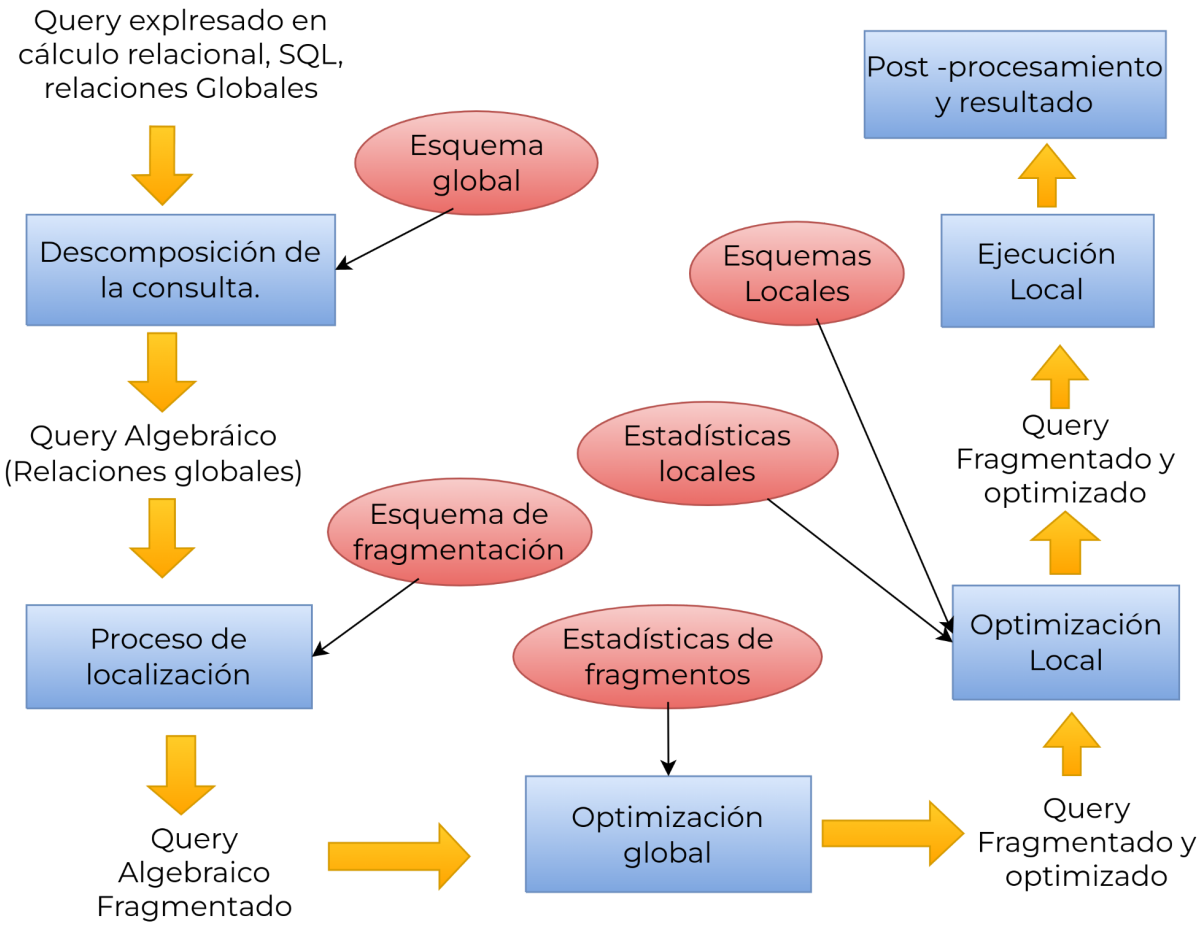
- El éxito de esta tarea depende en buena medida de la forma en la que el programador escriba SQL.
- Para ayudarle al optimizador se pueden emplear técnicas como:
    - Reducción de sentencias de forma estática en términos de álgebra relacional: ***query reducido***.
    - Reescritura de SQL empleando ***inline views,*** etc.

E. En cada nodo se realiza el procesamiento del query algebraico, se realizan optimizaciones adicionales: query algebraico fragmentado y optimizado.
F. Cada nodo envía sus resultados al nodo origen.
G. El nodo origen recibe los resultados y realiza post – procesamiento para obtener el resultado final.



3.1.2. Etapas del procesamiento de una consulta distribuida.

El diagrama anterior puede ser descrito en diferentes vistas. Una de ellas es a partir de la definición de un conjunto de etapas que se aplican al procesamiento de una sentencia SQL distribuida.



- Las 3 primeras etapas se ejecutan en el sitio que genera la estrategia de ejecución.
- Las últimas 2 etapas se ejecutan en cada uno de los nodos de donde se obtendrán los datos.

3.1.2.1. Otros aspectos a considerar para optimizar una consulta distribuida.

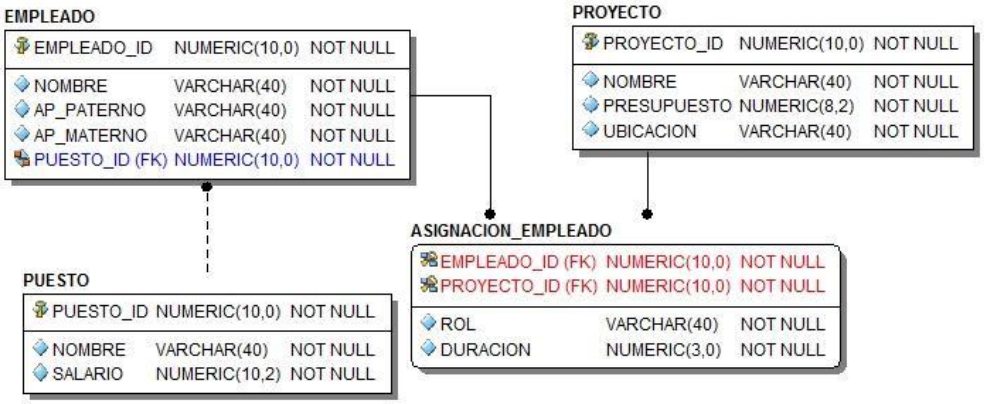
- Existen variantes de procesamiento respecto al diagrama anterior, por ejemplo, el usuario puede seleccionar a un nodo origen (***driving node***) para coordinar la ejecución de la sentencia diferente al nodo donde se lanza la consulta. Esto se logra empleando el concepto de ***Hint***.
- La transformación de una sentencia SQL a un query algebraico (en términos de álgebra relacional) puede generar múltiples soluciones equivalentes que crecen exponencialmente al aumentar el número de relaciones involucradas en la consulta. Algunas soluciones son más eficientes que otras.
- En BDD el universo de posibles soluciones se incrementa al existir diversas formas para realizar el intercambio de datos entre nodos.
- Si la elección de la estrategia más óptima se vuelve muy costosa, se prefiere seleccionar no la mejor, pero si una solución cercana: uso de heurísticas.

- Análisis del plan de ejecución. Es de vital importancia que el programador realice el análisis del plan de ejecución distribuido y aplique algunas mejoras en caso de que el optimizador no sea capaz de generar soluciones aceptables.

Ejercicio en clase 1

Realizar las actividades del siguiente ejercicio. La respuesta no se incluye en estas notas, se sugiere agregar la respuesta en sus apuntes.

Suponer que la tabla `EMP` y `AE` fueron fragmentadas con base a las siguientes reglas:

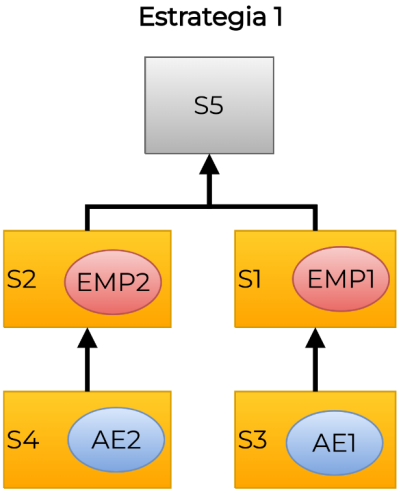


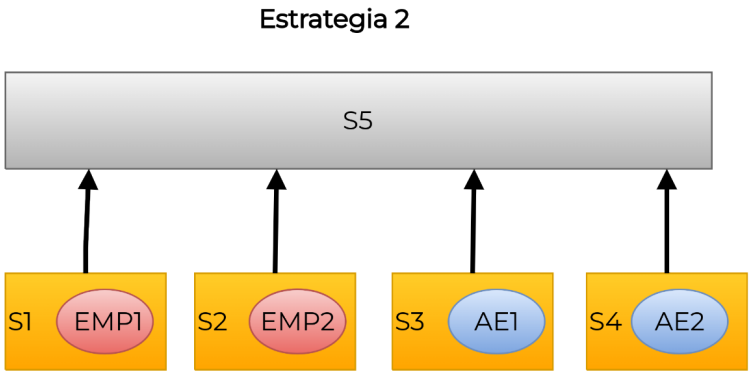
- Empleados cuyo apellido paterno inicia con las letras [A-M]
- Empleados cuyo apellido materno inicia con las letras [N-Z]
- Los datos de AE se distribuyen con base a la fragmentación de EMP
- Se cuentan con 4 sitios. En S1, S2 se almacenan los datos de la tabla EMP, y en S3 y S4 se almacenan los datos de AE
- Esquema de fragmentación:

$$EMP_1 = \sigma_{substr(ap-paterno,1,1)\geq 'A' \text{ and } substr(ap-paterno,1,1)\leq 'M'}(EMP)$$
$$EMP_2 = \sigma_{substr(ap-paterno,1,1)\geq 'N' \text{ and } substr(ap-paterno,1,1)\leq 'Z'}(EMP)$$

$$AE_1 = AE \ltimes_{emp-id} EMP_1$$
$$AE_2 = AE \ltimes_{emp-id} EMP_2$$

A. Mostrar el nombre de los empleados que tienen una duración de 37 meses en sus proyectos, generar una consulta SQL global que obtendrá los datos solicitados.
B. Realizar 2 transformaciones de la consulta anterior en álgebra relacionales equivalentes.
C. Proponer una primera estrategia de procesamiento, describir los pasos que realizará el DDBMS para ejecutar la consulta (ver figura 1).
D. Proponer una segunda estrategia de procesamiento (ver figura - estrategia 2).
E. ¿Qué estrategia será más eficiente?





Tarea

A. Generar las expresiones en álgebra relacional para la estrategia 2.
B. ¿Cuál de las 2 estrategias podría ser más adecuada?
C. Proponer una tercera estrategia de procesamiento.

3.2. Elementos que influyen en la optimización de consultas

En general, el proceso de optimización trata de minimizar los costos de la siguiente función:

$$c = costo\ I/O\ + costo\ procesador\ + costo\ comunicación$$

Las siguientes variables se toman en cuenta para minimizar el costo de la fórmula anterior:

- Cardinalidad de los fragmentos
- Estadísticas.
- Uso de semi Joins.
- Cantidad de datos a transmitir
- Complejidad de las operaciones del álgebra relacional a emplear:

Operación	Complejidad
Selección	O(n)
Proyección	O(n) sin eliminación de duplicados
Proyección	O(n log n) con eliminación de duplicados
Group by	O(n log n)
Join	O(n log n)
Semi-Join	O(n log n)
Set Operators (Union, Union all, intersect, minus)	O(n log n)

Operación	Complejidad
Producto cartesiano	$O(n^2)$

Donde:
- n representa la cardinalidad de la relación.
- $O(n \log log\, n)$ En general aplica para operadores binarios donde se asume que se requiere hacer comparación de los valores entre tuplas en donde se requiere un ordenamiento previo de una de las relaciones para poder comparar.
- Empleando técnicas con tablas Hash la complejidad anterior se puede reducir a $O(n)$

El tipo de optimizador también influye en el proceso de optimización.

3.2.1. Tipos de optimizadores basados en sus características

Optimizador basado en técnicas de búsqueda exhaustiva.
- Genera las N posibles combinaciones que pueden existir para ejecutar una consulta.
- Se puede tener una complejidad $O(n^n)$ donde N es el número de tablas que participan en un Join, esto puede representar un problema de tiempos y procesamiento.
- La selección del resultado se basa en el cálculo de costos.
- La solución obtenida es óptima

Optimizador basado en heurísticas
- La solución que genera no necesariamente es la mejor, pero sí cercana a la óptima.
- Aplica Heurísticas como:
  - Realiza agrupación de operaciones similares
  - Ejecuta operaciones con bajo costo al inicio (selección, proyección, etc.)
  - Reemplaza el uso de Joins por Semi-Joins (aplica en BDD en donde el recurso más costoso es el uso de la Red)
  - Re-ordena operaciones para reducir cardinalidades de resultados parciales.

Optimizador basado en reducción de tiempos
- Estático
  - La consulta es optimizada antes de su ejecución
  - Dificultad para estimar el tamaño de resultados intermedios.
  - Requiere el uso de estadísticas adecuadas para generar estimaciones correctas.
- Dinámico
  - La optimización se realiza al ejecutar la consulta.
  - Ofrece resultados intermedios exactos.
  - No requiere el uso de estadísticas, aunque se suele emplearlas para determinar las operaciones que se ejecutarán al principio y después se emplean los resultados obtenidos para decidir los siguientes pasos.
- Hibrida.
  - Inicia con optimización estática.

    - Si el resultado obtenido en una primera operación sobrepasa un umbral establecido, la consulta es optimizada de forma dinámica.

Optimizadores que explotan fragmentos replicados.

- El uso de replicación permite además de aumentar el grado de confiabilidad, incrementar el desempeño de lecturas (al seleccionar el sitio más cercano).
- Los optimizadores pueden hacer uso de datos replicados para mejorar los costos y tiempos de procesamiento.