Unidad 2: Procesamiento y Optimización de Consultas

Bases de Datos Avanzadas, Sesión 8 :

Algoritmos de Optimización, encauzamiento y Materialización.

Iván González Diego
Dept. Ciencias de la Computación
Universidad de Alcalá

INDICE

- *Introducción.*
- *Optimización basada en coste*
- *Programación dinámica para elegir planes de evaluación.*
- *Optimización heurística*
- *Materialización y Encauzamiento*

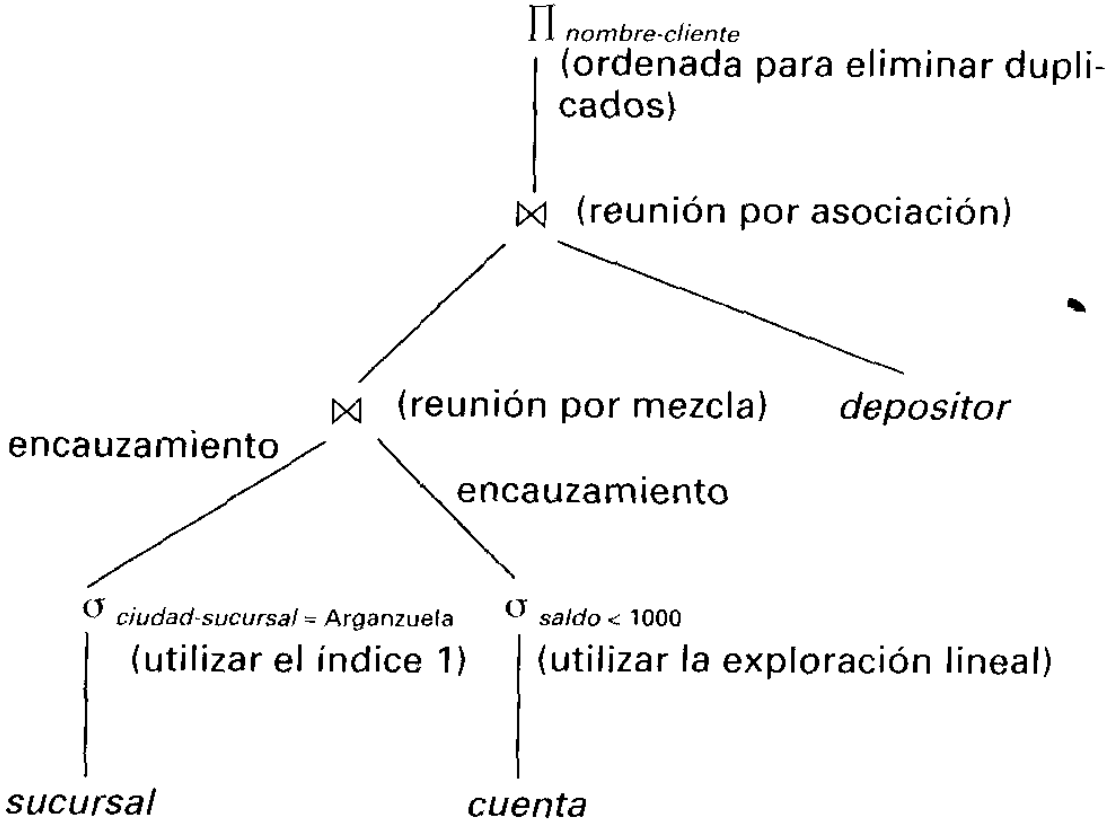
Referencias: Silberschatz 4ª Ed. Pp 343 - 364
Elmasri, 3ª Ed. Pp 553 - 595

Enumeración de expresiones Equivalentes

- *Optimizadores de consultas usan reglas de equivalencia para generar sistemáticamente expresiones equivalentes a una expresión dada.*
- *Genera todas las expresiones equivalentes hasta que no se pueden encontrar más expresiones:*
  - *Para cada expresión encontrada, usar todas las reglas que se pueden encontrar y añadir las nuevas expresiones a las encontradas.*
- *Muy costosa en espacio y tiempo*
- *Requerimiento de espacio se puede reducir compartiendo subexpresiones comunes*
  - *Cuando E1 se genera de E2 $\Rightarrow$ usualmente sólo el nivel alto de las dos son diferentes, los subárboles por debajo son el mismo y se pueden compartir*
- *Requerimiento de tiempo se puede reducir no generando todas las expresiones*

Plan de Evaluación

- *Un plan de evaluación define exactamente qué algoritmo se usa para cada operación y cómo se coordinan la ejecución de las operaciones.*



Elección de Planes de Evaluación

- *Se debe de considerar la interacción de las técnicas de evaluación cuando se eligen planes de evaluación $\Rightarrow$ elegir el algoritmo menos costoso para cada operación individual puede que no sea la mejor elección.*

- *Los optimizadores incorporan elementos de las dos aproximaciones siguientes:*

  - *Buscar todos los planes de evaluación y elegir el mejor plan según el coste*
  - *Utilizar la heurística para elegir un plan.*

Optimización basada en el coste

- *Considerar encontrar la mejor reunión para* $r_1 \bowtie r_2 \bowtie \ldots r_n$.
- *Hay* $(2(n-1))!/(n-1)!$ *diferentes planes*
  - *Con* n = *7, el número es 665280*
  - *Con* n = *10, el número es mayor que 17.600 millones!*
- *No es necesario generar todos los planes*
- *Usando programación dinámica* $\Rightarrow$ *el orden menos costoso para* $\{r_1, r_2, \ldots r_n\}$ *se analiza una vez y se almacena para su uso futuro.*

Programación dinámica en Optimización

- *Para encontrar la mejor reunión para un conjunto S de n relaciones*
  - *Considerar todos los posibles planes de la forma:* $S_1 \bowtie (S - S_1)$
  - *Recursivamente analizar los costes para reunir subconjuntos de S para encontrar el coste de cada plan*
  - *Elegir el menos costoso de las* $2^n - 1$ *alternativas.*
  - *Cuando el plan para cualquier subconjunto se analiza* $\Rightarrow$ *se almacena y se reutiliza en vez de volver a calcularlo*
- *Coste temporal* $O(3^n)$
- *Coste espacial* $O(2^n)$

Optimización Heurística

- *Optimización por coste es costosa*
- *Utilizar la heurística para reducir el número de elecciones de una manera basada en costes*
- *Optimización heurística transforma el árbol de consultas usando un conjunto de reglas que generalmente mejoran el rendimiento:*
  - *Realizar la selección cuanto antes ⇒ reduce el número de tuplas*
  - *Realizar la proyección cuanto antes ⇒ reduce el número de atributos*
  - *Realizar las operaciones de selección y reunión más restrictivas antes que otras operaciones similares*
  - *Algunos sistemas usan sólo heurística , otros combinan heurística con optimizaciones parciales basadas en coste.*

Pasos en una Optimización Heurística

1. *Descomponer las selecciones conjuntivas en una secuencia de operaciones de selección sencillas. ⇒ Regla equivalencia 1*
2. *Desplazar las operaciones de selección hacia la parte baja del árbol ⇒ ejecutarlas lo antes posible (Reglas 2, 7a, 7b, 11).*
3. *Ejecutar primero las selecciones y reuniones que produzcan relaciones más pequeñas (Regla 6).*
4. *Reemplazar operaciones producto cartesiano seguidas de una selección por una operación reunión. (Regla 4a).*
5. *Dividir y desplazar hacia abajo del árbol como sea posible las listas de atributos de proyección, creando nuevas proyecciones si se necesitan (Reglas 3, 8a, 8b, 12).*
6. *Identificar aquellos subárboles cuyas operaciones pueden ser encauzadas y ejecutarlos usando encauzamiento*

Estructura de los Optimizadores de consultas

- *Optimizador System R considera sólo órdenes de reunión en profundidad por la izquierda $\Rightarrow$ reduce la complejidad y genera planes que se pueden evaluar por encauzamiento.*

  Utiliza la heurística para poner selecciones y proyecciones abajo en el árbol.
- *Optimización heurística usada en algunas versiones de Oracle:*
  - *Repetidamente tomar la mejor relación para la siguiente reunión.*
  - *Comenzando desde n planes de evaluación. Tomar el mejor de estos*
- *Para exploraciones usando índices secundarios, algunos optimizadores tienen en cuanta la probabilidad de que la página que contiene la tupla se encuentre en la memoria*
- *Complejidad en los optimizadores de consultas SQL*
  - *Ejemplo: subconsultas anidadas*

Estructura de los Optimizadores de consultas

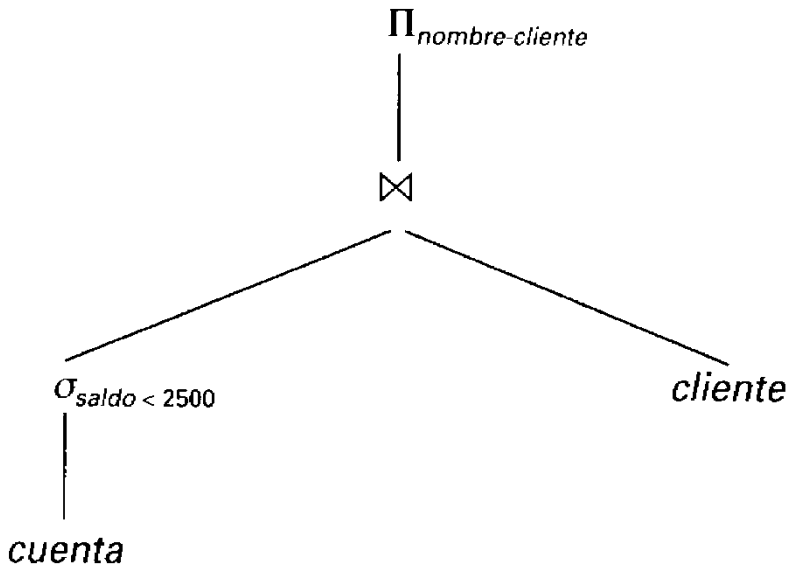
- *Algunos optimizadores integran selección heurística y generación de planes de acceso alternativos*
- *Incluso con el uso de la heurística, optimización basada en coste impone una sobrecarga adicional*
- *Pero este gasto se ve recompensando por el ahorro del tiempo de ejecución de la consulta, que queda dominado por los accesos al disco.*

Evaluación de expresiones

- *Se han visto algoritmos para operaciones individuales*
- *Alternativas para evaluar una expresión entera:*
  - Materialización ⇒ genera resultados de una expresión cuyas entradas son relaciones ó ya están analizadas. Materializa ó almacena en disco → Una operación a la vez.
  - Encauzamientos ⇒ evaluar varias operaciones de manera simultánea, con los resultados de una operación pasados a la siguiente → Evalúa varias en paralelo.

Materialización

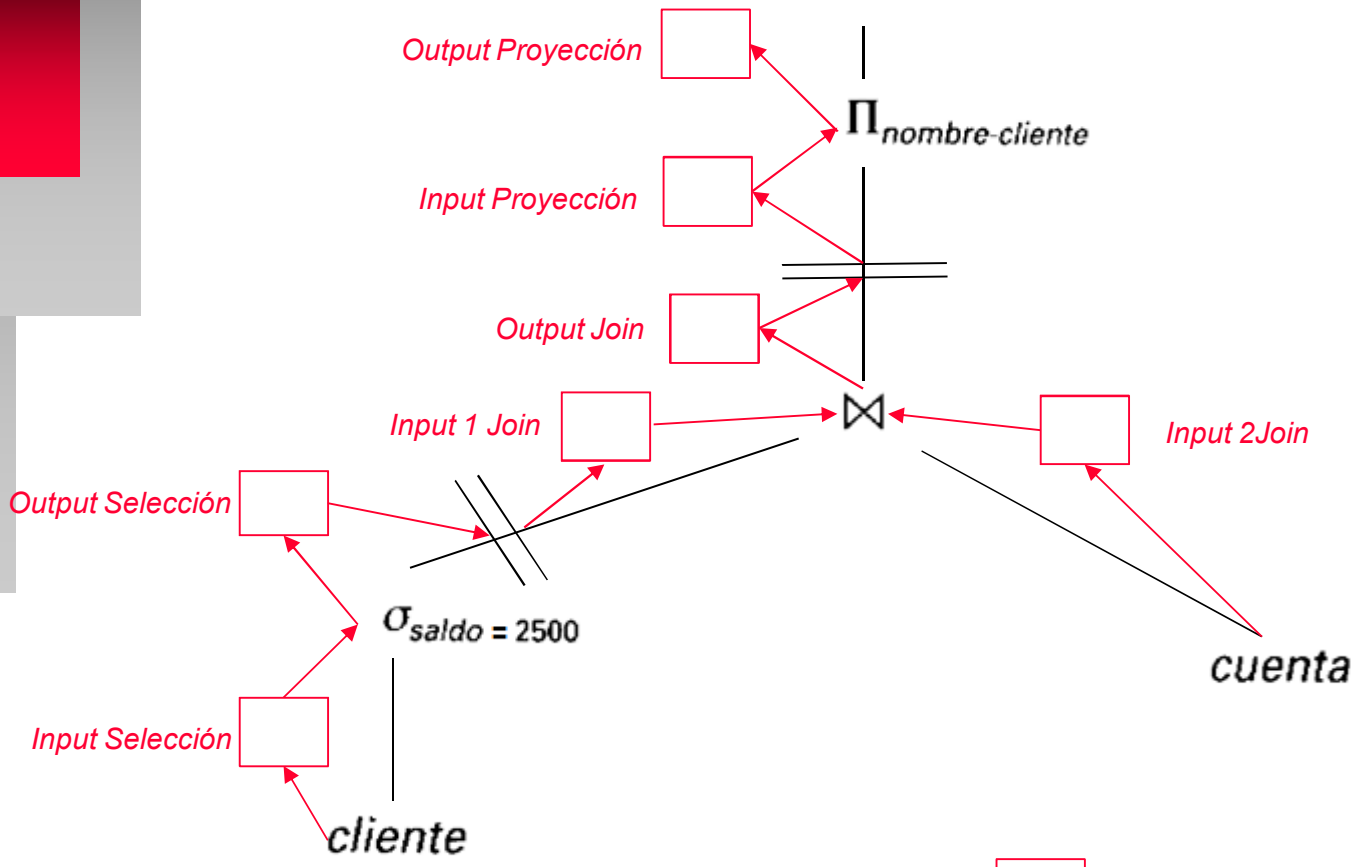
- *Evaluación materializada: evalúa una operación a la vez, comenzando por el nivel más bajo.*
- *Usa resultados intermedios materializados en relaciones temporales para evaluar las operaciones del siguiente nivel*
- *Ejemplo:*



Materialización

- *Evaluación materializada siempre se puede aplicar*
- *Coste de escribir resultados al disco y leerlos puede ser muy alto*
  - *Las fórmulas de coste para las operaciones ignoran el coste de escribir resultados al disco, por lo que*
    - *Coste total = Suma del coste de operaciones individuales + coste de escribir resultados intermedios al disco*

Materialización: Ejemplo



→ Bloque de memoria

→ Materializar Disco

Coste total con bucle anidado por bloques?

Encauzamiento

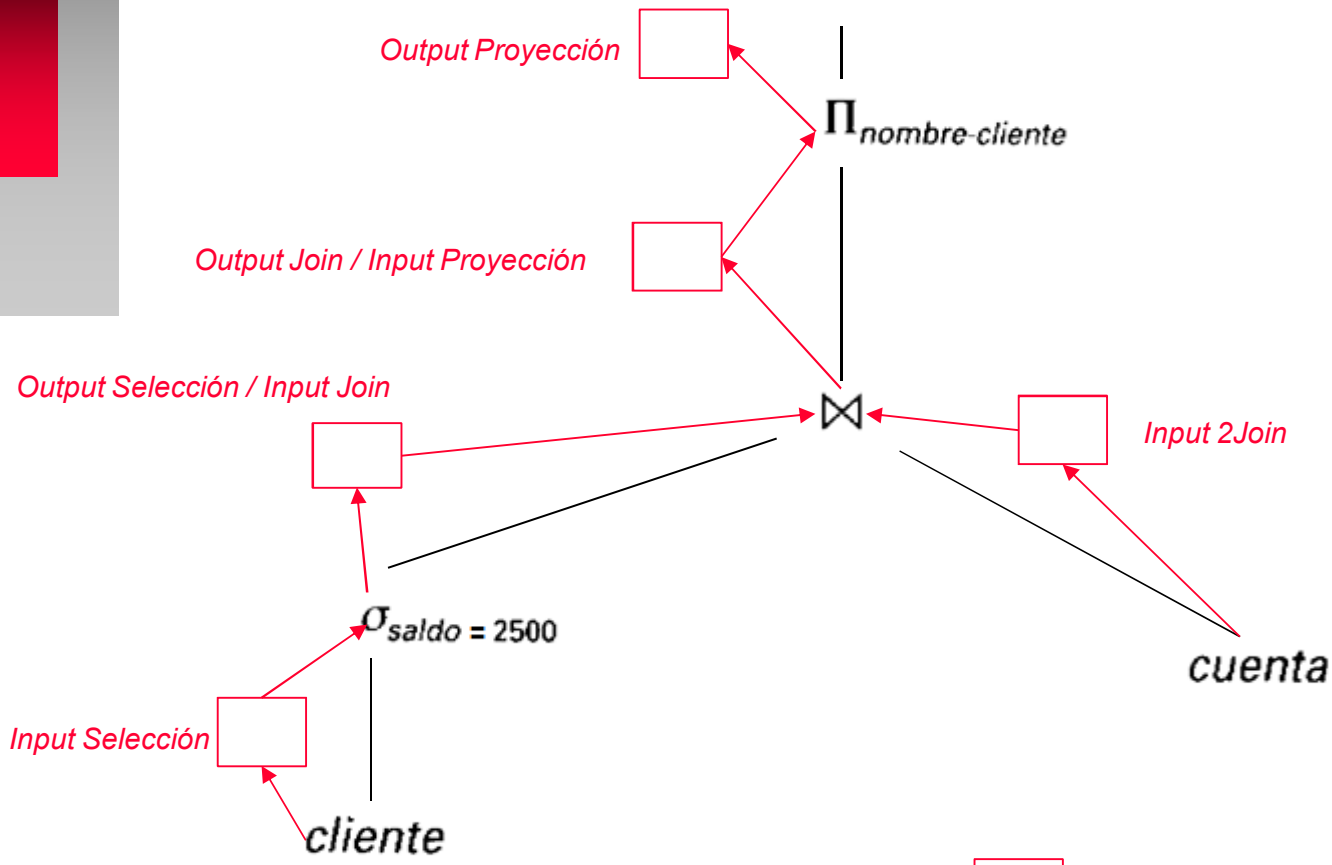
- *Evaluación encauzada: evalúa varias operaciones a la vez, pasando los resultados de una operación a la siguiente.*
- *Ejemplo: en la expresión anterior, no almacenar*

$$\sigma_{saldo<2500}(cuenta)$$

Pasa directamente las tuplas

- *Menos costosa que la materialización.*
- *No siempre es posible: ordenación, reunión por asociación.*
- *Necesidad de algoritmos de evaluación que generen tuplas de salida incluso cuando se reciben tuplas de entrada.*
- *Se puede ejecutar de dos maneras:*
  - Bajo demanda
  - Por los productores

Encauzamiento: Ejemplo



→ Bloque de memoria

→ Materializar Disco

Coste total con bucle anidado por bloques?

Encauzamiento

- *Bajo demanda*
  - *Sistema pide la siguiente tupla desde el nivel alto de operación.*
  - *Cada operación pide la siguiente tupla a sus operaciones hijas para producir su siguiente tupla*
  - *Entre llamadas ⇒ la operación debe de mantener el estado*
  - *Se puede implementar como un iterador:*
    - *Open()*
      - *Exploración del fichero: inicializar, almacenar puntero*
      - *Reunión por mezcla: ordenar relaciones y almacenar punteros al comienzo de las relaciones ordenadas.*
    - *Next()*
      - *Salida de la siguiente tupla y actualizar puntero*
      - *Continuar con la mezcla desde el estado anterior hasta encontrar la siguiente tupla de salida. Actualizar puntero*
    - *Close()*

Encauzamiento

- *Por producción:*
  - *Los operadores producen tuplas impacientemente y las pasan a sus padres.*
    - *Se necesita memoria intermedia, para colocar las tuplas y los padres cogerlas.*
    - *Si la memoria se llena, el hijo espera hasta que haya espacio libre.*
  - *Sistema cambia entre operaciones cuando tiene espacio en la memoria intermedia para generar tuplas.*

Algoritmos de Evaluación del encauzamiento

- *Algunos algoritmos no son capaces de producir resultados incluso aunque tengan tuplas de entrada*
  - *Ejplo: reunión por mezcla o por asociación*
  - *Los resultados intermedios se escriben en disco para leerlos después.*
- *Hay variantes de algoritmos para generar por lo menos resultados, cuando las tuplas se leen*
  - *Reunión por asociación híbrida genera tuplas incluso cuando la relación prueba particionada en memoria (0) está leída*
  - *Técnica de reunión encauzada $\Rightarrow$ Reunión por asociación híbrida modificada, que usa memoria intermedia para la partición 0 de ambas relaciones.*