Capítulo 4

Procesamiento de consultas

Visión general

- Para procesar una consulta, una consulta de SQL se puede **traducir** a una expresión del álgebra de tablas.
- Luego se puede **evaluar** la consulta del álgebra de tablas.
- La evaluación de una consulta del álgebra de tablas va a estar en términos de operadores físicos.

Visión general

- Una expresión de álgebra de tablas puede tener varias expresiones equivalentes.
- **Ejemplo**: $\sigma_{salary<75000}(\prod_{salary}(instructor))$ es equivalente a $\prod_{salary}(\sigma_{salary<75000}(instructor))$
- **¿Qué consecuencias tiene esto con respecto al procesamiento de consultas?**
- Que para procesar una consulta en álgebra de tablas puede convenir evaluar una consulta equivalente
  - con determinados operadores físicos.
  - O sea, que se puede **planear** cómo se va a procesar una consulta.

Visión general

- Dada una consulta *C* del álgebra de tablas un **plan de evaluación** consiste de una expresión *E* equivalente a *C* y operadores físicos para los operadores lógicos de *E*.
- La **máquina de ejecución de consultas** toma el plan de evaluación de consulta, ejecuta ese plan y retorna las respuestas de la consulta.

Visión general

- Los diferentes planes de evaluación para una consulta dada pueden tener diferentes costos.
- El SGBD debería construir un plan de evaluación que minimiza el costo de evaluación de consultas.
  - Una vez que un plan de evaluación es elegido, la consulta es evaluada con este plan.
- **Optimización de consultas**: entre todos aquellos planes de evaluación equivalentes encontrar aquel con menor costo.
  - El costo es estimado usando información estadística de la base de datos.
  - **P.ej:** número de registros en cada tabla, tamaño de los registros, etc.
- Con el fin de optimizar una consulta un optimizador de consultas debe conocer el **costo de cada operación**.

Asuntos de este capítulo

- En este capítulo estudiamos primero cómo ejecutar un plan de evaluación; y luego estudiaremos un poco sobre cómo elegir un plan de evaluación.
- Primero, dada una expresión del álgebra de tablas, hay que **elegir bien los operadores físicos.**
- Luego se estimará el **costo de procesamiento** de la consulta.

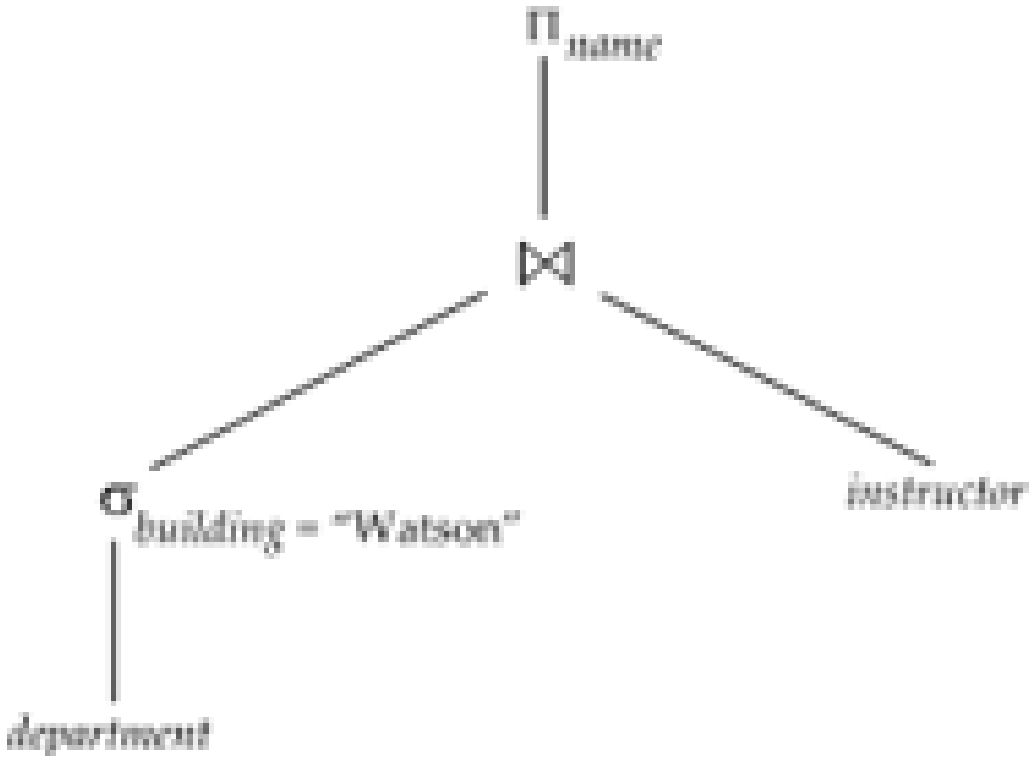
Árbol binario de ejecución

- El **árbol binario de ejecución** de una consulta es el árbol binario de una expresión de consulta.
  - Los nodos hoja son tablas de la base de datos
  - Los nodos internos son operadores del álgebra de tablas
- Para una consulta del álgebra de tablas puedo definir varias expresiones equivalentes y cada una tiene su árbol binario de ejecución.
- La evaluación de un árbol binario de ejecución va a estar en términos de operadores físicos.

Árbol binario de ejecución

- El árbol binario de ejecución asociado a la consulta

$$\Pi_{name}(((\sigma_{building='watson'}(department)) \bowtie instructor)$$



Evaluación del árbol binario de ejecución

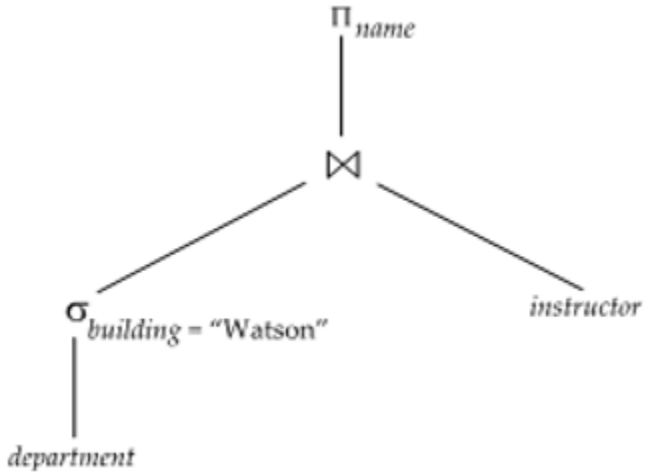
- El resultado de evaluar un operador de un nodo interno del árbol binario de ejecución que no es la raíz del árbol se llama **resultado intermedio**.
- Para la evaluación del árbol binario de ejecución hay dos enfoques:
  - **Materialización**: los resultados intermedios se guardan en disco en tablas temporales a las cuales tiene acceso el sistema gestor de BD (SGBD).
    - No hay índices sobre este tipo de tablas.
  - **Encausamiento**: a medida que se van generando los resultados intermedios se van pasando al siguiente operador.
    - Los resultados intermedios no se guardan en disco.

Evaluación del árbol binario de ejecución

- **Comparando ambos enfoques**: Materialización usa mucho menos memoria y bastante más espacio en disco y encausamiento usa mucho menos espacio en disco, y bastante más memoria.
- Vamos a estudiar solamente materialización.

Materialización

- **Ejemplo**: Supongamos que tenemos el árbol binario de ejecución y usamos materialización:



1. Computamos y almacenamos en disco primero $\sigma_{\text{building = 'Watson'}}$ (department)
2. Computamos y almacenamos en disco la reunión natural del resultado intermedio anterior con *instructor*.
3. Computamos la proyección según *nombre*.

Materialización

- Con **materialización**:
  - Cambiar nodos lógicos por físicos en el árbol binario de ejecución.
  - Si hay más de un operador físico posible, elegir el menos costoso.
  - Evaluamos un operador físico por vez comenzando en el nivel más bajo.
  - Usamos resultados intermedios en tablas temporales para evaluar los operadores físicos del siguiente nivel.

Materialización

- **Repaso**: para estimar la cantidad de registros del resultado intermedio para los operadores de selección y reunión selectiva se usa una función de probabilidad llamada factor de selectividad.
  - A partir de la cantidad de registros se calcula la cantidad de bloques del resultado intermedio.
  - Si el operador de selección o reunión usa predicado P, y el input del operador es i, denotamos al factor de selectividad mediante: fs(P, i).
  - **Para selección:**
    - cantidad de registros resultado intermedio = |r| * fs(P, r)
  - **Para reunión:**
    - Cantidad de registros resultado intermedio = |r| * |s| * fs(P, r, s)

Materialización

- **Repaso (cont):** Si tengo en el resultado intermedio N registros de tamaño R cada uno y B es el tamaño del bloque, entonces la cantidad de bloques del resultado intermedio viene dada por:

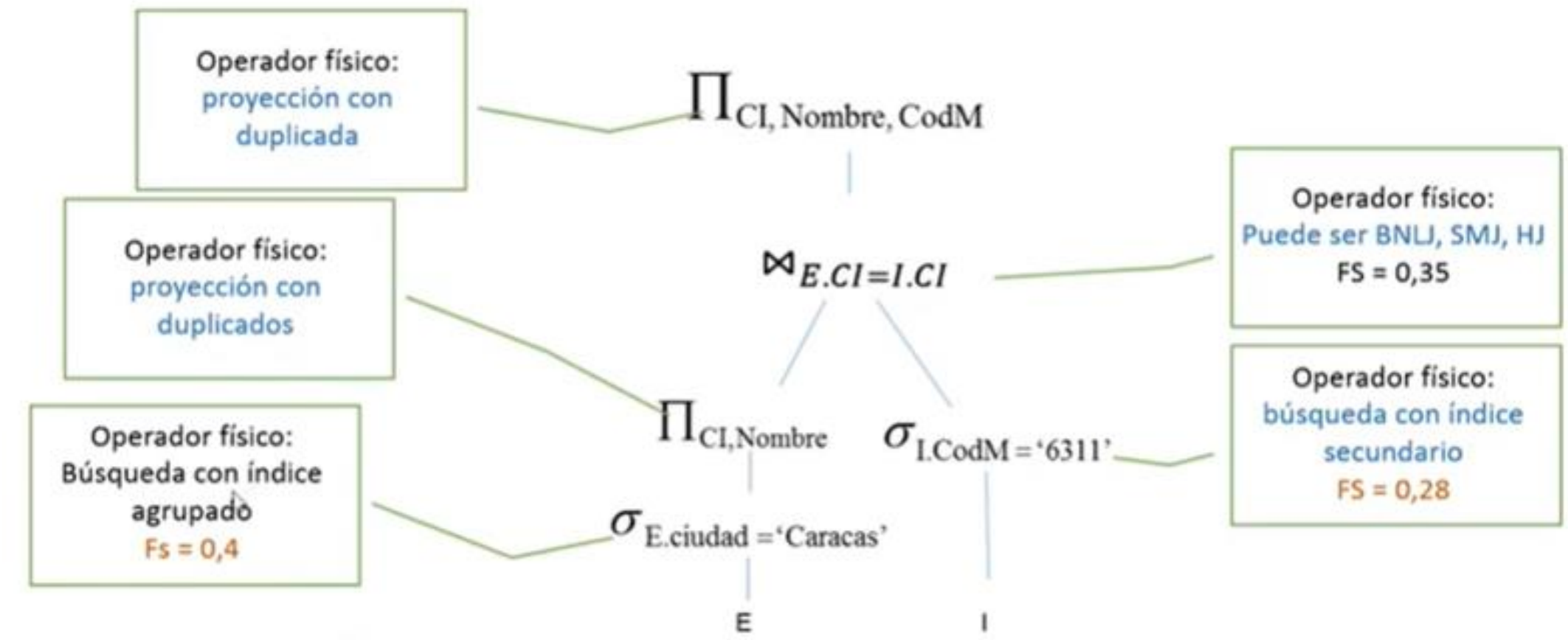
  NumBloques = ⌈(N x R) / B⌉ .

Materialización

- Ahora vemos cómo procesar y estimar el costo de una consulta con materialización.
- **Fase 1: decidir el plan de ejecución**
  1. Armar el árbol binario de ejecución
  2. Calcular el factor de selectividad para selecciones y reuniones (selectivas y naturales).
  3. Decidir operadores físicos.
     - Solo se usan índices si la tabla de la BD lo amerita.

Materialización

- **Ejemplo de fase 1:**

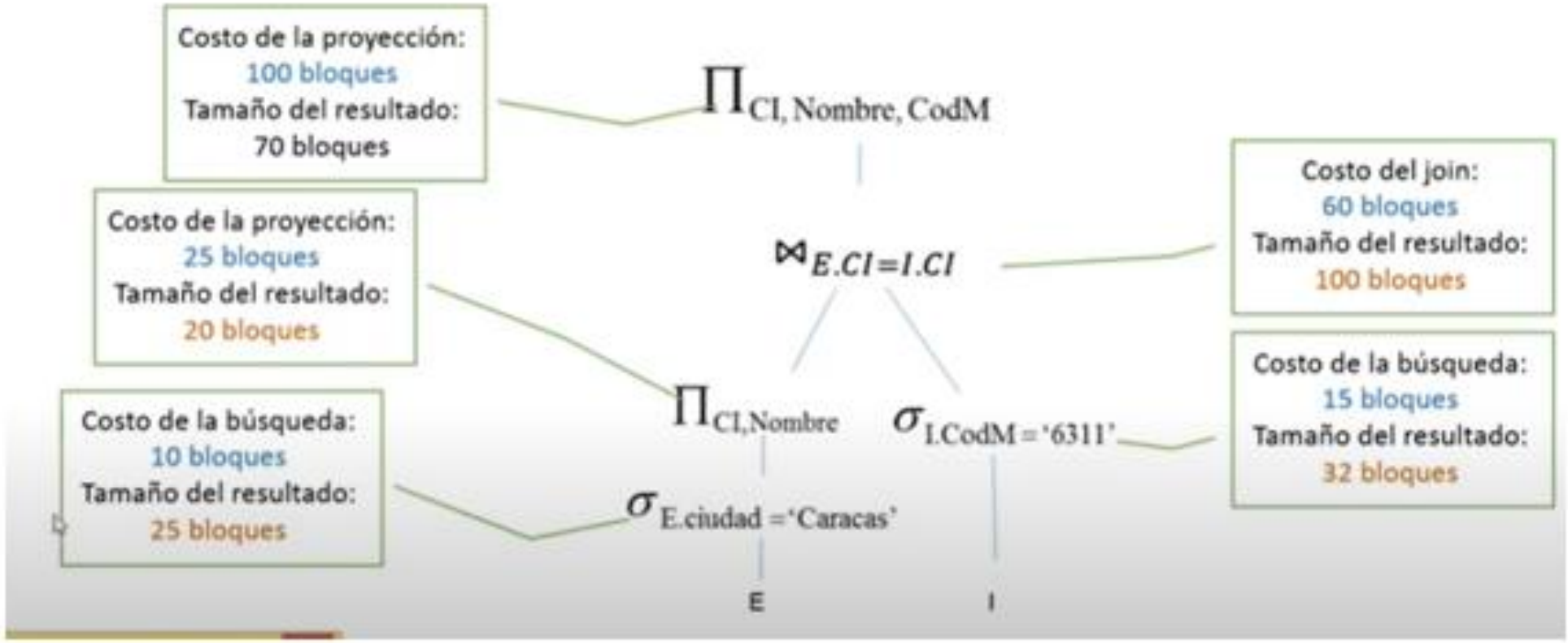


Materialización

- **Fase 2: estimar el costo de ejecutar el plan de evaluación**
  1. Calcular el tamaño en bloques de las tablas de la BD
  2. Calcular el tamaño de los resultados intermedios en bloques
  3. Calcular el costo de los operadores físicos
  4. Sumar los costos totales
- Aplicar el método de dos fases.

Materialización

- **Ejemplo de Fase 2:**



Materialización

- Juntando todo:
  - $Costo\ total = \sum costo(operaciones) + \sum costo(materialización)$
  - Costo total = (10+20+15+60+100) + (25+20+32+100)
  - Costo total = 382 (accesos a disco)
  - Esto se multiplica por la velocidad de transferencia y se tiene el tiempo
- A continuación, vamos a hacer un ejemplo detallado de la aplicación de materialización.

Caso de estudio

- Sea la consulta: $\Pi_{id}\ (\sigma_{edad > 40}\ (persona))$, donde:
  - Persona ocupa 20 bloques de 10 registros cada uno.
  - Las personas tienen edades de 1 a 100
  - Asumir que entran 50 id por bloque.
  - Entran 20 edades por nodo en el índice primario en edad.

Árbol binario de ejecución

Π_{id}

|

σ(edad > 40)

|

persona

Factor de selectividad

- Usamos la propiedad:
- $fs(A \geq c, r) = (\max(A, r) - c) / (\max(A, r) - \min(A, r))$
- Así obtenemos:

$fs(edad > 40, persona)$

$= \max(edad, persona) - 41 / (\max(edad, persona) - \min(edad, persona))$

$= (100 - 41) / (100 - 1) = 59/99 = 0{,}595$

Operadores físicos

- **Proyección**:
  - Requiere recorrer todos los registros y realizar una proyección en cada uno.
  - Se recorren todos los bloques de la tabla.
  - Estimación de costo = b_r transferencias de bloques + 1 acceso a bloque
    - b_r denota el número de bloques conteniendo registros de la tabla *r*
- **Selección:**
  - La tabla está ordenada en *A*.
  - para $\sigma_{A \geq V}(r)$ usar el índice para encontrar el primer registro $\geq v$ y escanar la tabla secuencialmente desde allí.
    - Costo: hi + b transferencias de bloques, hi accesos de bloques.
    - b el número de bloques conteniendo registros con A $\geq v$.

Tamaño en bloques de las tablas

- Según el enunciado persona tiene 20 bloques.
- Cada bloque tiene 10 registros.

Tamaño de los resultados intermedios

- $r = \sigma_{edad > 40}(persona)$
- $|r| = |persona| * fs(edad > 40, persona) = 200 * 0{,}595 = 119$
- $B_r = 119 / 10 = 12$ bloques

Costo de los operadores físicos

- **Proyección**: costo es cantidad de transferencia de bloques de resultado intermedio: 12 transferencias de bloques.
- **Selección**: costo hi + b transferencias de bloques.
  - b es el número de bloques contiendo registros con edad > 40. Vimos que b = 12.
  - hi altura del árbol B+
  - Tenemos 100 edades y asumimos que entran 20 entradas por nodo del árbol B+. Entonces:
  - hi = $\lceil \log_{\lceil n/2 \rceil}(K) \rceil = \lceil \log_{\lceil 20/2 \rceil}(100) \rceil = \lceil \log_{10}(100) \rceil = 2$
  - Costo = 12 + 2 = 14 transferencias de disco.

Sumar los costos totales

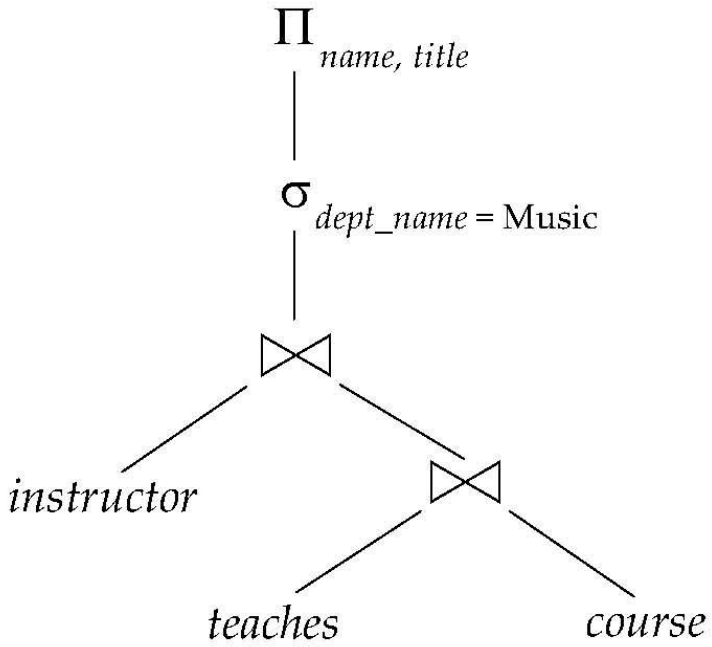
- Costo total = costo de operaciones + costo de materialización
- = (12 + 14) + 12 = 24 + 14 = 38 transferencias de bloque.

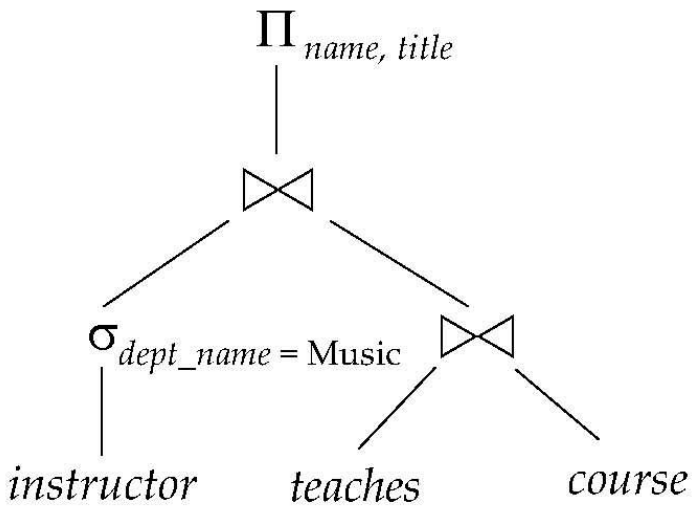
Tamaño del resultado final

- La proyección se aplica a 119 registros
- La proyección me da 119 id
- Se asumió que entran 50 id por bloque
- Bloques de resultado final = Techo(119/50) = 3

Repaso

- Hay formas alternativas de evaluar una expresión de consulta:
  - Usando expresiones equivalentes
  - Usando diferentes algoritmos para cada operación





Repaso

- Un **plan de evaluación** define qué algoritmo es usado para cada operación y cómo se coordina la ejecución de las operaciones.
- La diferencia de costo entre planes de evaluación puede ser enorme.
  - Por ejemplo, de segundos a días en algunos casos.

Introducción a la optimización de consultas

- **Pasos en optimización de consultas basada en costo:**
  1. Generar expresiones lógicamente equivalentes usando **reglas de equivalencia.**
  2. Anotar expresiones resultantes para obtener planes de consulta alternativos.
  3. Elegir el plan más económico basado en el **costo estimado.**

Repaso

- **El costo estimado de un plan se basa en:**
  - Información estadística acerca de tablas.
    - P.ej: número de tuplas, número de valores distintos de un atributo.
  - Estimación estadística para resultados intermedios.
    - Para computar el costo de expresiones complejas.
  - Fórmula de costo para algoritmos, computados usando estadísticas.

Introducción a la optimización de consultas

- Las estadísticas son computadas **periódicamente** porque
  - tienden a no cambiar radicalmente en un corto tiempo;
  - estadísticas algo imprecisas son útiles siempre que sean aplicadas consistentemente a todos los planes.

Repaso

- **El número de transferencias de bloques es influenciado por:**
  1. Los operadores lógicos elegidos para implementar la consulta.
  2. El tamaño de los resultados intermedios.
  3. Los operadores físicos usados para implementar los operadores lógicos.
  4. El método para pasar argumentos de un operador físico al siguiente.
  5. **El orden de operaciones similares, especialmente reuniones y selecciones.**
- El último ítem es menos conocido y va a quedar más claro durante este capítulo.

Transformación de expresiones de consulta

- Dos expresiones del álgebra de tablas son **equivalentes por igualdad** si las dos tablas son equivalentes en esquema e información.
- Dos expresiones del álgebra de tablas son **equivalentes módulo ordenamiento de registros** si las dos expresiones generan el mismo multiconjunto de tuplas (i.e. las mismas tuplas en la misma cantidad para cada tupla).
  - O sea, dos expresiones equivalentes a lo más alteran el orden de las tuplas.
  - $r =_O s \Leftrightarrow O(r) = O(s)$, donde O es la operación de ordenación del álgebra de tablas en base a todas las columnas.
- Una **regla de equivalencia** dice que las expresiones de dos formas son equivalentes usando uno de los tipos de equivalencia de arriba.
  - Podemos reemplazar una expresión de la primera forma por la segunda forma o vice-versa.

Reglas de equivalencia

1. La selección conjuntiva de operaciones puede ser deconstruida en una secuencia de selecciones individuales:

$$\sigma_{q_1 \wedge q_2}(E) = \sigma_{q_1}(\sigma_{q_2}(E))$$

2. Las operaciones de selección son conmutativas:

$$\sigma_{q_1}(\sigma_{q_2}(E)) = \sigma_{q_2}(\sigma_{q_1}(E))$$

3. Solo la última en una secuencia de operaciones proyección es necesitada, las otras pueden ser omitidas.

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L_1}(E)$$

Reglas de equivalencia

- **Nomenclatura**:
- Usaremos Θ_r^N para indicar una lista de N columnas de la tabla r.
- Usaremos $\Theta_r^N = \Theta_s^N$ para indicar la igualación de cada columna de Θ_r^N con las columnas de Θ_s^N . Omitiremos la tabla y/o cantidad de columnas si no hay ambigüedad.
- Indicaremos con p_r a una proposición que sólo utilice columnas de r.

Reglas de equivalencia

4. La reunión natural es asociativa:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

5. La siguiente regla es parecida a la definición de reunión selectiva:

$$\Pi_\theta(\sigma_{\theta_r^N = \theta_s^N}(r \times s)) = r \ {}_{\theta_r^N}\bowtie_{\theta_s^N} \ s \quad \text{con } \theta \text{ las columnas de } \bowtie$$

6. La siguiente regla muestra cómo podemos expresar una selección de una reunión selectiva por medio de una reunión selectiva.

$$\sigma_{\theta_r^N = \theta_s^N}(r \ {}_{\theta_r^M}\bowtie_{\theta_s^M} \ s) = r \ {}_{\theta_r^N, \theta_r^M}\bowtie_{\theta_s^N, \theta_s^M} \ s$$

Reglas de equivalencia

7. La siguiente regla nos permite aplicar selección antes de reunión selectiva:

$$\sigma_{p_r}(r \ {}_{\theta_r}\!\bowtie_{\theta_s} s) = \sigma_{p_r}(r) \ {}_{\theta_r}\!\bowtie_{\theta_s} s$$

8. La siguiente regla se deduce de reglas previas:

$$\sigma_{p_r \wedge p_s}(r \ {}_{\theta_r}\!\bowtie_{\theta_s} s) = \sigma_{p_r}(r) \ {}_{\theta_r}\!\bowtie_{\theta_s} \sigma_{p_s}(s)$$

9. La siguiente regla muestra cómo se comporta la proyección cuando se usa junto con la reunión selectiva.

$$\Pi_{\theta_r^N, \theta_s^O}(r \ {}_{\theta_r^M}\!\bowtie_{\theta_s^M} s) = \Pi_{\theta_r^N}(r) \ {}_{\theta_r^M}\!\bowtie_{\theta_s^M} \Pi_{\theta_s^O}(s) \quad \text{con } \theta_r^M \subseteq \theta_r^N \text{ y } \theta_s^M \subseteq \theta_s^N$$

Reglas de equivalencia

10. La concatenación “conmuta” alterando solo el orden de las tuplas:

    r ++ s $=_O$ s ++ r

11. La concatenación es asociativa.

12. Selección distribuye con concatenación, intersección y resta

$$\sigma_p(r \star s) = \sigma_p(r) \star \sigma_p(s) \quad \text{con} \star \in \{++, \cap, \setminus\}$$

13. La siguiente propiedad es más débil que la anterior.

$$\sigma_p(r \star s) = \sigma_p(r) \star s \quad \text{con} \star \in \{\cap, \setminus\}$$

14. La proyección distribuye con la concatenación:

$$\Pi_\theta(r \mathbin{++} s) = \Pi_\theta(r) \mathbin{++} \Pi_\theta(s)$$

Reglas de equivalencia

15. Eliminación de duplicados distribuye con reunión natural:
    $v(r \bowtie s) = v(r) \bowtie v(s)$
16. Eliminación de duplicados conmuta con selección:
    $v(\sigma_P(r)) = \sigma_P(v(r))$

Empujando selecciones

- **Consulta**: encontrar los nombres de todos los instructores en el departamento de música, junto con los títulos de los cursos que enseñan.

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”}} (instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}(course))))$$

- **¿Qué regla de equivalencia se puede usar?**

Empujando selecciones

- **Consulta**: encontrar los nombres de todos los instructores en el departamento de música, junto con los títulos de los cursos que enseñan.

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”}} (instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course))))$$

- Transformación usando la regla 7:

$$\Pi_{name,\ title}((\sigma_{dept\_name=\ \text{“Music”}}\ (instructor)) \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course)))$$

- **Conclusión**: realizar la selección tan temprano como sea posible reduce el tamaño de la tabla a ser combinada.

Ejemplo con múltiples transformaciones

- **Consulta**: Encontrar los nombres de todos los instructores en el departamento de música que han enseñado un curso en 2009, junto con los títulos de los cursos que han enseñado.

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”} \wedge year\ =\ 2009} (instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course))))$$

- **¿Qué reglas de equivalencia se pueden usar?¿En qué orden?**

Ejemplo con múltiples transformaciones

- **Consulta**: Encontrar los nombres de todos los instructores en el departamento de música que han enseñado un curso en 2009, junto con los títulos de los cursos que han enseñado.

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”} \wedge year\ =\ 2009} (instructor \bowtie (teaches \bowtie \Pi_{course\_id,\ title}\ (course))))$$

- Por asociatividad de la reunión natural (regla 4):

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”} \wedge gear\ =\ 2009} ((instructor \bowtie teaches) \bowtie \Pi_{course\_id,\ title}\ (course)))$$

- Usamos regla 7 de hacer selecciones temprano:

$$\sigma_{dept\_name\ =\ \text{“Music”}}\ (instructor) \bowtie \sigma_{\ year\ =\ 2009}\ (teaches)$$

Ordenamiento de reuniones naturales

- Considere la expresión:

$$\Pi_{name,\ title}(\sigma_{dept\_name=\ \text{“Music”}}\ (instructor) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}\ (course))))$$

- **¿Como la reunión natural es asociativa, qué reunión conviene hacer primero?**

Ordenamiento de reuniones naturales

- Considere la expresión:

$$\Pi_{name,\ title}(\sigma_{dept\_name=\text{“Music”}}(instructor) \bowtie teaches) \bowtie \Pi_{course\_id,\ title}(course))))$$

- Podemos computar primero $teaches \bowtie \Pi_{course\_id,\ title}(course)$ y luego combinamos el resultado con $\sigma_{dept\_name=\text{“Music”}}(instructor)$ pero el resultado de la primera reunión es probable que sea una tabla grande.
- Solo una pequeña fracción de los instructores es probable que sean del departamento de música. Es mejor computar primero:

$$\sigma_{dept\_name=\text{“Music”}}(instructor) \bowtie teaches$$

Optimización Heurística

- **Problema**: Optimización basada en costo es cara.
  - Aunque el costo de optimización de consultas puede ser reducido por algoritmos inteligentes, el número de diferentes planes de evaluación para una consulta puede ser muy grande y encontrar el plan óptimo para ese conjunto requiere mucho esfuerzo computacional.
- Una **heurística** es un enfoque para resolución de problemas que usa un método práctico o varios atajos con el fin de producir soluciones que pueden no ser óptimas, pero son suficientes dado el tiempo o recursos limitados.
  - También se usa la frase: **estrategia de regla de pulgar**.

Optimización Heurística

- **Solución:** Los sistemas pueden usar heurísticas para reducir el número de elecciones que deben ser hechas cuando se trabaja con costos.
  - La **optimización heurística** transforma el árbol de consulta usando un conjunto de reglas que típicamente pero no siempre mejoran el desempeño de ejecución.
- La mayoría de los optimizadores incluye **heurísticas** para reducir el costo de la optimización de consultas,
  - con el riesgo potencial de no encontrar un plan óptimo.

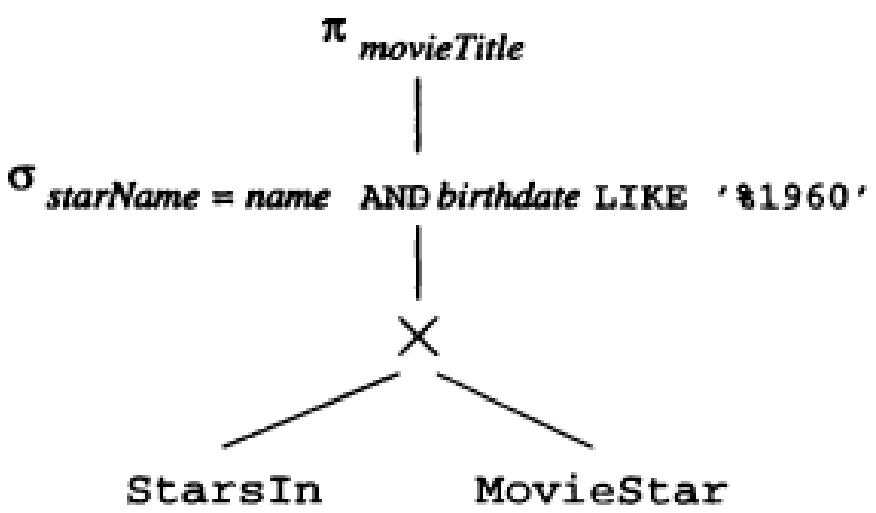
Optimización Heurística

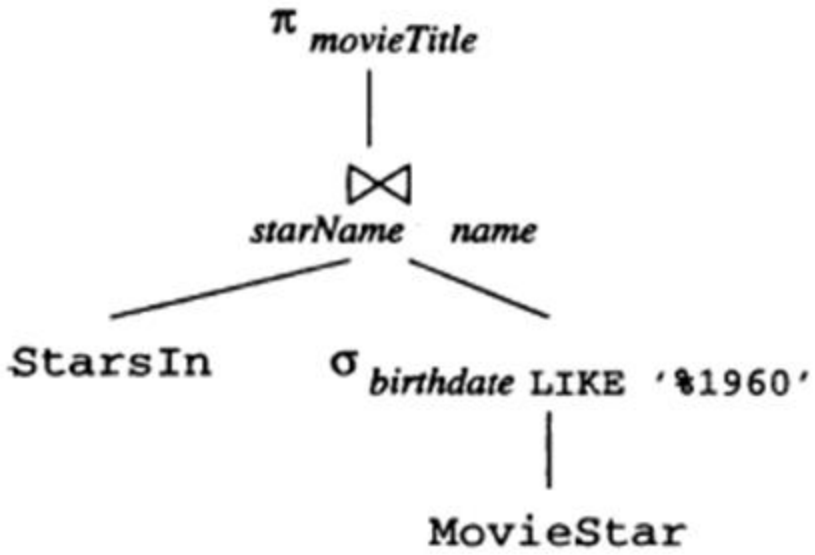
- **Ejemplo de optimización heurística:**
  - Realizar selección tempranamente (reduce el número de tuplas)
    - Empujar selección abajo en el árbol de ejecución tanto como sea posible.
    - Si una condición de selección es un AND de varias condiciones, podemos dividir la condición y empujar cada pieza abajo en el árbol de ejecución separadamente.
  - Realizar proyección temprano (reduce el número de atributos)
    - Proyecciones pueden ser empujadas abajo en el árbol.
  - Hacer la selección más restrictiva (i.e. con tamaño de resultado menor) antes de hacer otras selecciones.
    - P.ej. Si una condición de selección es un AND de varias condiciones, podemos aplicar selecciones sobre esas condiciones separadamente en algún orden buscando hacer la selección lo más restrictiva posible.

Optimización Heurística

- **Ejemplo de optimización heurística (cont):**
  - Hacer operaciones de reunión más restrictivas:
    - Restringir a tipos particulares de ordenes de reuniones (p.ej. Ordenes de reunión profunda a la izquierda)
  - Ciertas selecciones pueden ser combinadas con producto abajo para tornar las operaciones en una reunión (natural o selectiva),
    - la cual es generalmente más eficiente de evaluar esas operaciones separadamente.
    - Para esto suele ser conveniente introducir una proyección también.

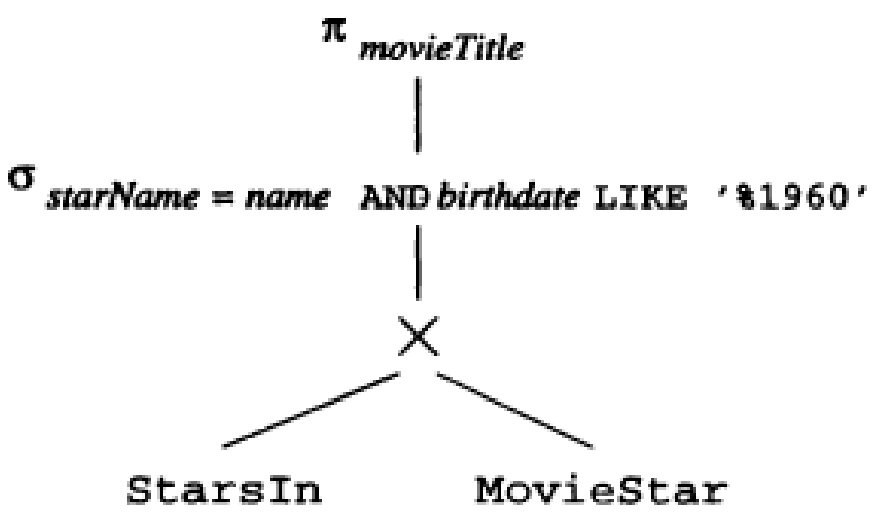
Optimización Heurística

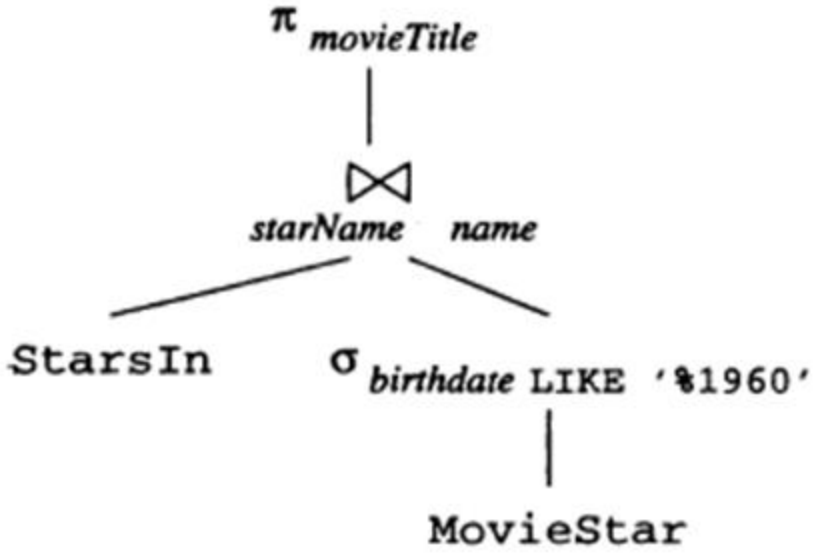




Sugerir como aplicar la optimización heurística anterior para pasar de la expresión de la izquierda a la de la derecha.

Optimización Heurística





Ejemplo de uso de optimización heurística anterior: primero dividimos las dos partes de la selección en $\sigma_{\text{starName = name}}$ y $\sigma_{\text{birthdate LIKE '%1960'}}$. La última puede ser empujada abajo en el árbol debido a que el único atributo involucrado *birthdate* es de tabla *MovieStar*. La primera condición involucra atributos de ambos lados del producto y se puede introducir proyección de todos los atributos menos *name* pues la proyección de la consulta es mucho más restrictiva. Así, podemos aplicar definición de reunión selectiva según *starName* y *name*. Haciendo todo esto obtenemos el árbol de la derecha.

Optimización Heurística

- También una optimización heurística puede referirse a operadores físicos a usar.
- **Ejemplo de optimización heurística**:
  1. Para hacer reunión de varias tablas, comenzar juntando el par de tablas cuyo resultado tiene el tamaño estimado menor; luego repetir el proceso para el resultado de esa reunión y las otras tablas en el conjunto a ser combinado.
  2. Si el plan lógico usa una selección $\sigma_{A=C}$ (r) y r tiene un índice en el atributo A, realizar escaneo de ese índice en r (vimos 2 algoritmos para esto).

Optimización Heurística

- **Ejemplo de optimización heurística (cont)**:
  3. Si un argumento de una reunión tiene un índice en atributos de la reunión para la tabla de la derecha, usar reunión de loop anidado indexada.
  4. Si un argumento de la reunión está ordenado en los atributos de la reunión, es preferible usar sort-join.
  5. Cuando se computa la unión o intersección de 3 o más relaciones, agrupar las más pequeñas primero

Selección de orden de reunión basado en costo

- Si tengo reunión natural/selectiva de varias tablas, la optimización basada en costo es demasiado costosa, la razón es la gran cantidad de casos a examinar.
  - Para n = 3 hay 12 órdenes
  - Para n = 5 hay 1680 órdenes.
  - Para n = 7 hay 665280 órdenes.
  - Con n = 10 hay 17,6 mil millones de órdenes.
  - Con *n* tablas hay
    - (2(n-1))!/(n-1)! diferentes ordenaciones.
- **Observación**: los distintos ordenes pueden cambiar también el orden de las columnas. Pero podemos proyectar según los atributos de la expresión original de la consulta para que eso no suceda.

Selección de orden de reunión basado en costo

- Para la selección de orden de reunión natural basado en costo hay **algoritmos de programación dinámica**.
  - En lugar de generar todas las expresiones de reunión posibles, se consideran conjuntos de relaciones a reunir y optimizar.
  - Estos conjuntos se guardan en tablas.
  - Estas tablas tienen muchos menos elementos que la cantidad de reuniones posibles.
  - Usando **programación dinámica** el orden de reunión de menor costo para cada subconjunto de $\{r_1,\ldots, r_n\}$ es computado solo una vez y almacenado para uso futuro.

Selección de orden de reunión basado en costo

- **Ejemplo**: si hay 4 tablas, tenemos en total:
  - (2*3)!/(3)! = 6!/3! = 720/6 = 120 casos
  - Sin embargo, la cantidad de subconjuntos de 4 elementos es 16.
  - Programación dinámica trabaja con tablas de 16 subconjuntos.
- **Ejemplo**: si hay 5 tablas hay 1680 órdenes. Sin embargo, programación dinámica trabaja con tabla de 32 subconjuntos.

Selección de orden de reunión basado en costo

- **Solución usando programación dinámica**:
  - Construimos una **tabla** con una entrada para cada subconjunto de una o más de las tablas de la reunión.
  - **En la tabla ponemos:**
    - Un índice es un conjunto de tablas e la base de datos.
    - Ponemos en una celda:
    1. El tamaño estimado de la reunión de esas tablas.
       - Ya vimos cómo se hace esto usando factor de selectividad.
    2. El menor costo de computar la reunión de esas tablas.
    3. La expresión que da lugar al menor costo.
  - La construcción de esa tabla se hace por **inducción** en el tamaño del subconjunto.

Selección de orden de reunión basado en costo

- **Solución usando programación dinámica (cont)**:
  - **Caso base:**
    - la entrada para una sola tabla r consiste del tamaño de r, un costo de 0 y la expresión que es r.
    - La entrada para un par de tablas {r_i, r_j}:
      - tiene estimación de tamaño que es el producto de tamaños de r_i y r_j multiplicado por el factor de selectividad;
      - tiene costo 0 porque no hay tablas intermedias involucradas,
      - Además, tomamos la menor de r_i y r_j como el argumento izquierdo de la expresión de la reunión natural.

Selección de orden de reunión basado en costo

- **Solución usando programación dinámica (cont)**:
  - **Inducción**:
    - Consideramos todas las maneras de particionar el conjunto actual de tablas S en dos subconjuntos disjuntos S1 y S2.
    - Para cada una de estas maneras consideramos la suma de:
      1. los mejores costos de S1 y S2.
      2. los tamaños para los resultados para S1 y S2.
    - Sea cual sea la partición que da el mejor costo, usamos esta suma como el costo de S.
    - La expresión de S es la reunión natural de las mejores expresiones para S1 y S2.

Selección de orden de reunión basado en costo

- **Solución alternativa**: modificar el algoritmo de programación dinámica anterior para tomar en cuenta algoritmos de reunión natural.
  - Cuando se computa el costo de $S1 \bowtie S2$ sumamos el costo de S1, el costo de S2 y el menor costo de juntar los dos resultados usando el mejor algoritmo disponible.
- **Complejidad**
  - La complejidad en tiempo del procedimiento puede probarse que es $O(3^n)$.
  - Con n = 10 este número es 59000 en lugar de 176 mil millones de casos a evaluar.
  - La complexidad en espacio es $O(2^n)$.

Selección de orden de reunión basado en costo

- **Ejemplo**: considerar la reunión natural de 4 tablas R, S, T y U. Por simplicidad asumimos que cada una tiene 1000 tuplas. Además asumimos:

$R(a,b)$	$S(b,c)$	$T(c,d)$	$U(d,a)$
$V(R,a) = 100$			$V(U,a) = 50$
$V(R,b) = 200$	$V(S,b) = 100$		
	$V(S,c) = 500$	$V(T,c) = 20$	
		$V(T,d) = 50$	$V(U,d) = 1000$

- Para los conjuntos de una tabla los tamaños, costos y mejores planes son:

	$\{R\}$	$\{S\}$	$\{T\}$	$\{U\}$
Size	1000	1000	1000	1000
Cost	0	0	0	0
Best plan	R	S	T	U

Selección de orden de reunión basado en costo

- **Ejemplo (cont.):** Ahora consideramos pares de tablas.
- El costo para cada uno es 0 porque no hay tablas intermedias en la reunión de las dos tablas (las 2 tablas son chicas y están en memoria principal).
- Hay dos posibles planes. Como todas las tablas tienen igual tamaño tomamos la primera en orden alfabético para ser el argumento izquierdo (independientemente del orden el tamaño es el mismo).
- Los tamaños de las tablas resultantes se calculan como dijimos. El resultado es:

	$\{R,S\}$	$\{R,T\}$	$\{R,U\}$	$\{S,T\}$	$\{S,U\}$	$\{T,U\}$
Size	5000	1,000,000	10,000	2000	1,000,000	1000
Cost	0	0	0	0	0	0
Best plan	$R \bowtie S$	$R \bowtie T$	$R \bowtie U$	$S \bowtie T$	$S \bowtie U$	$T \bowtie U$

Selección de orden de reunión basado en costo

- **Ejemplo (cont.):** ahora consideramos la tabla para reuniones de 3 tablas. La única manera es tomar dos tablas para reunir primero.
- La estimación del tamaño del resultado es como siempre y omitimos los detalles de este cálculo.
- La estimación de costo para cada tripla de tablas es el tamaño de la tabla intermedia – la reunión de las 2 tablas elegidas.
  - Como queremos que este costo sea tan pequeño como posible consideramos cada par de las 3 tablas y tomamos el par con el menor tamaño (usando la tabla previa).
- Si consideramos este par como el argumento del natural join obtenemos la tabla:

	$\{R,S,T\}$	$\{R,S,U\}$	$\{R,T,U\}$	$\{S,T,U\}$
Size	10,000	50,000	10,000	2,000
Cost	2,000	5,000	1,000	1,000
Best plan	$(S \bowtie T) \bowtie R$	$(R \bowtie S) \bowtie U$	$(T \bowtie U) \bowtie R$	$(T \bowtie U) \bowtie S$

Selección de orden de reunión basado en costo

- **Ejemplo (cont.):**
  - Considerar {R, S, T}: consideramos los 3 pares posibles de tablas.
  - El costo de $R \bowtie S$ es 5000, comenzar con $R \bowtie T$ da un costo de 1000000, comenzar con $S \bowtie T$ da un costo de 2000.
  - Como el último es el menor costo, elegimos el plan $(S \bowtie T) \bowtie R$ (en este caso la estimación del tamaño no depende del lugar donde ponemos R).
  - Tamaño: $|(S \bowtie T) \bowtie R| = |(S \bowtie T)| * |R| * fs(S \bowtie T.b = R.b, S \bowtie T, R)$
    $= 2000 * 1000 / 200 = 10000$

Selección de orden de reunión basado en costo

- **Ejemplo (cont.):**
  - Consideramos la situación de join de 4 tablas. Hay dos formas para hacerlo:
    - Tomar 3 tablas para combinar de la mejor manera posible y hacer la reunión natural con la cuarta.
    - Dividir las 4 tablas en pares de 2 y reunir los pares y luego combinar los resultados.
  - Hay 7 maneras de agrupar las reuniones naturales basados en los agrupamientos preferidos de las dos tablas previas:

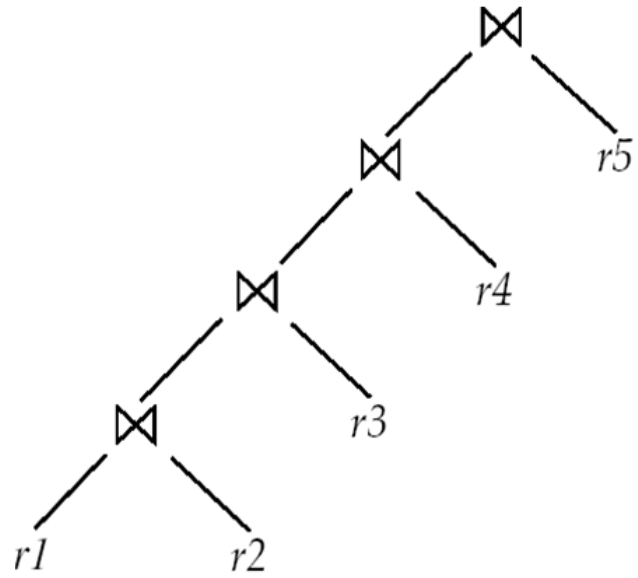
Grouping	Cost
$((S \bowtie T) \bowtie R) \bowtie U$	12,000
$((R \bowtie S) \bowtie U) \bowtie T$	55,000
$((T \bowtie U) \bowtie R) \bowtie S$	11,000
$((T \bowtie U) \bowtie S) \bowtie R$	3,000
$(T \bowtie U) \bowtie (R \bowtie S)$	6,000
$(R \bowtie T) \bowtie (S \bowtie U)$	2,000,000
$(S \bowtie T) \bowtie (R \bowtie U)$	12,000

Selección de orden de reunión basado en costo

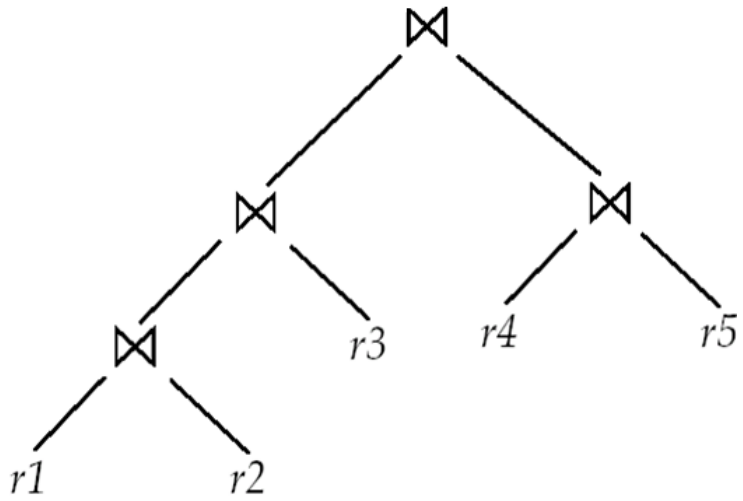
- **Ejemplo (cont.):**
  - El costo de (S ⋈ T)⋈ (R ⋈ U) es la suma de los tamaños y los costos de (S ⋈ T) y (R ⋈ U). Los costos de los pares son 0 y los tamaños son 2000 y 10000 respectivamente.
  - En la última línea de la tabla consideramos los pares (S ⋈ T) y (R ⋈ U).
  - Debido a que elegimos la tabla más pequeña como argumento izquierdo, entonces la expresión es: (S ⋈ T)⋈ (R ⋈ U).
    - El hacer eso hace que los costos de los algoritmos vistos para reunión que tienen fórmula sea más conveniente.
  - En la tabla el menor de todos los costos es asociado con ((T ⋈ U) ⋈ S) ⋈R.
  - Esta expresión es la que se elige para computar la reunión natural; el costo es 3000.

Árboles de reunión profunda a la izquierda

- En **árboles de reunión profunda a la izquierda** el lado derecho de cada reunión natural es una tabla, no el resultado de una reunión intermedia.



(a) Left-deep join tree



(b) Non-left-deep join tree

Árboles de reunión profunda a la izquierda

- **Para encontrar el mejor árbol de reunión profunda a la izquierda para un conjunto de n tablas**:
  - **Paso inductivo:**
    - Si S tiene k tablas, para cada r en S, primero computamos la reunión de S – {r} y luego hacemos la reunión natural con r.
    - La expresión de reunión para S tiene la mejor expresión reunión para S – {r} como argumento izquierdo de la reunión final y r como el argumento derecho.
    - El costo de la reunión de S es el costo de S – {r} más el tamaño del resultado para S - {r}. Tomamos el r que da el menor costo.
    - El tamaño de S se calcula por la fórmula que usa factor de selectividad.
- **Complejidad**:
  - Si solo árboles de reunión profunda a la izquierda son considerados, el tiempo de complejidad para encontrar el mejor orden de reunión es: $O(n\ 2^n)$.
  - La complejidad en espacio permanece en $O(2^n)$.

Enfoques híbridos

- **Enfoques híbridos**: Algunos sistemas gestores de BD combinan heurísticas con optimización parcial basada en costo.
  - Enfoques de optimización de consultas que aplican elecciones heurísticas de plan para algunas partes de la consulta con elección basada en costo basada en la generación de planes alternativos para otras partes de la consulta han sido adoptados en varios SGBD.
- **Ejemplo**: Muchos optimizadores siguen un enfoque basado en usar transformaciones heurísticas para manejar construcciones diferentes de reuniones y aplican el algoritmo de orden de reunión optimizado por costo para subexpresiones que involucran solo reuniones y selecciones.
  - Los detalles de tales heurísticas son específicos a optimizadores individuales.

Enfoques híbridos

- **Ejemplo**: Varios optimizadores consideran solo reunión profunda a la izquierda más heurísticas que empujan selecciones y proyecciones hacia abajo en el árbol de consulta.
  - Esto reduce la complejidad de la optimización y genera planes que permiten evaluación usando tubería.