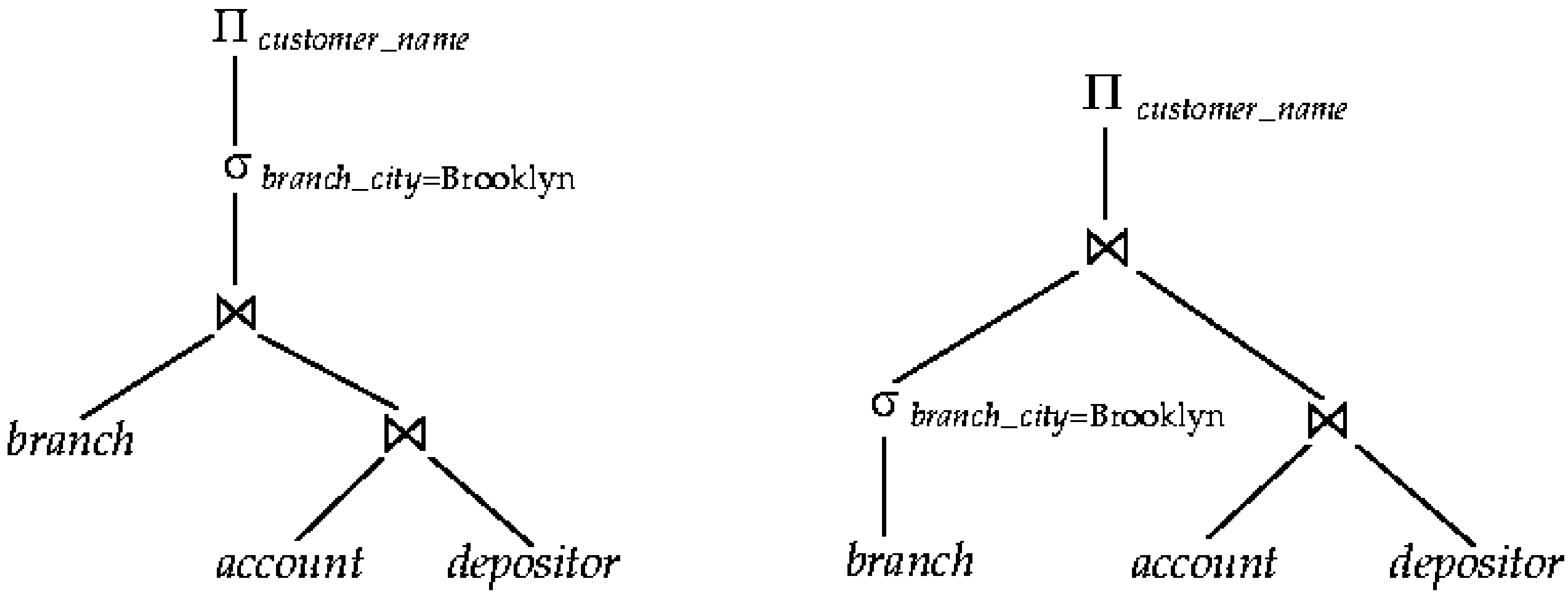
Optimización de Consultas

Base de Datos II

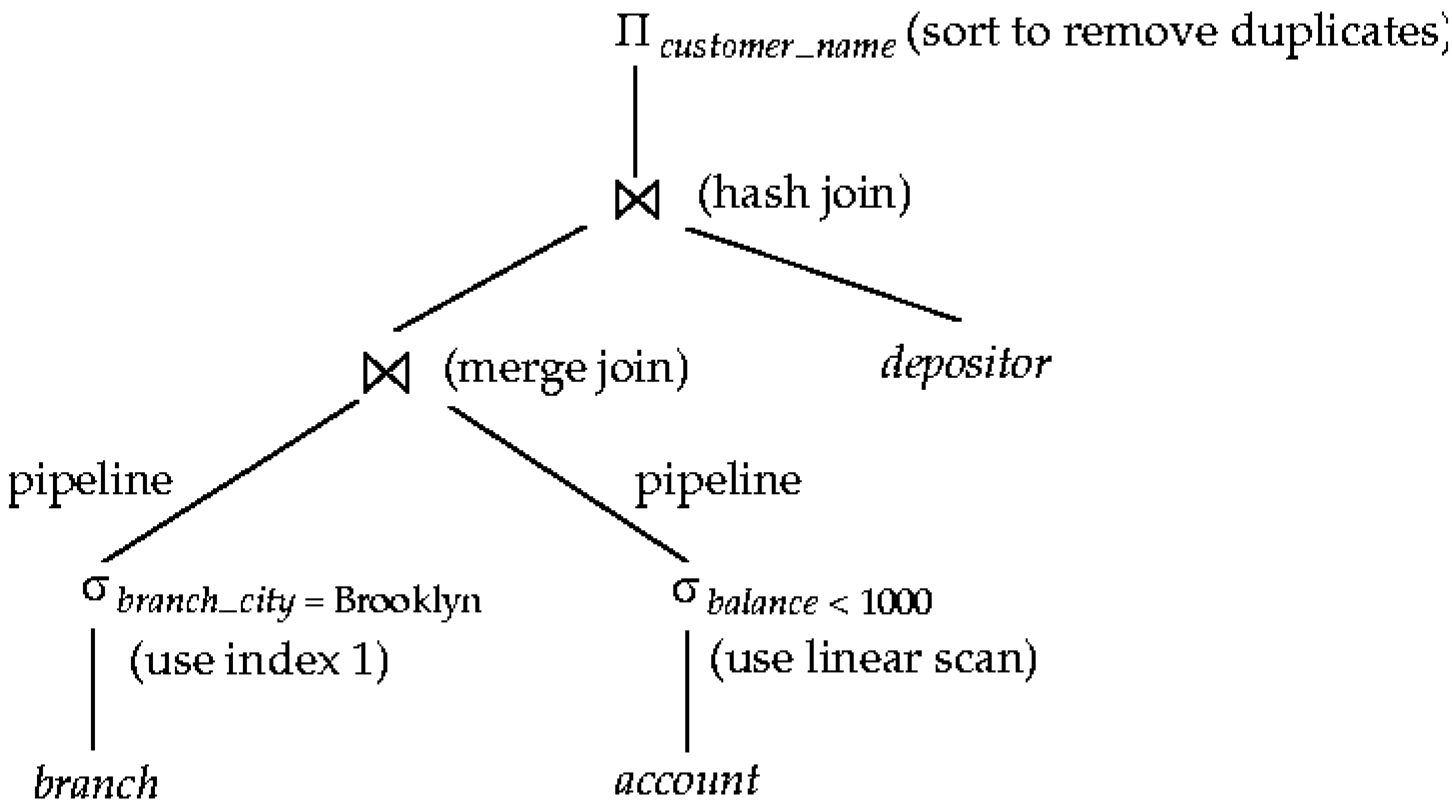
Introducción

- Dada un expresión del álgebra relacional el **Optimizador de Consultas (OC)** debe producir un plan de ejecución que calcule el resultado de la manera menos costosa.
- La generación de planes de ejecución implica dos etapas:
  - Generar expresiones que sean lógicamente equivalentes a la expresión de la consulta.
  - Anotar las operaciones con los algoritmos a utilizar.



Introducción

- Un **Plan de Ejecución** define:
  - El algoritmo que debe ser utilizado para cada operación.
  - El orden de evaluación de los operaciones.



Optimización basada en costo

- Dados dos planes de ejecución la diferencia en tiempo de ejecución entre ambos puede ser enorme.
  - Segundos Vs Días
- Los pasos de la **Optimización Basada en Costo** son:
  - Generar expresiones lógicamente equivalente basada en **reglas de equivalencias**
  - Anotar cada expresión de manera a obtener diferentes planes.
  - Escoger el plan más barato basado en un **costo estimado**.
- La estimación del costo de los planes se basa en:
  - Información estadística acerca de la relaciones.
    - Br, Fr, MBi, Aai
  - Información estadística acerca de los resultados intermedios.
  - Estimación del costo de los algoritmos usando como entrada la información estadística

Trasformación de las expresiones

- Dos expresiones se dicen equivalentes si generan el mismo conjunto resultado en cada ejemplar legal de la base de datos.
- Una **regla de equivalencia** indica las dos formas equivalentes de una expresión.
  - **Se puede substituir una expresión de la primera forma por otra equivalente de la segunda forma y viceversa.**
- El optimizador hace uso de la reglas para transformar las expresiones en otras equivalentemente lógicas.

Reglas de equivalencia

1. Las condiciones conjuntivas pueden descomponerse en una secuencia de selecciones individuales.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Las operaciones de selección son conmutativas.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Sólo la ultima operación de proyección es necesaria, las otras pueden omitirse.

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

4. Las selecciones pueden combinarse con los productos cartesianos y con las reuniones zeta
   a. $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
   b. $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$

Reglas de equivalencia

5. Las reuniones zeta y reuniones naturales conmutativas.

$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$

6. (a) Las operaciones de reunión natural son asociativas.

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

(b) Las operaciones de reunión zeta son asociativas en la siguiente manera.

$$(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$$

Donde $\theta 2$ solo involucra atributos de E_2 y E_3

Reglas de equivalencia

7. La operación de selección se distribuye sobre la operación de reunión zeta cuando:
   a) La condición de selección solo involucra a atributos de una de la relaciones (en este caso E_1).

   $$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$

   b) Cuando las condiciones de selección involucran de forma separada a atributos de una de las relaciones.

   $$\sigma_{\theta 1 \wedge \theta 2}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 1}(E_1)) \bowtie_{\theta} (\sigma_{\theta 2}(E_2))$$

Reglas de equivalencia

8. La proyección se distribuye sobre una reunión zeta de la siguiente manera:
   a) Si la condición sólo involucra atributos de L_1 U L_2

$$\prod_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = (\prod_{L_1}(E_1)) \bowtie_{\theta} (\prod_{L_2}(E_2))$$

   b) Dada la reunión $E_1 \bowtie_{\theta} E_2$ y,
      - Siendo L1 y L2 atributos de E1 y E2 respectivamente.
      - Siendo L3 atributos de E1 involucrados en la condición de reunión pero que no están en L1 U L2.
      - Siendo L4 atributos de E2 involucrados en la condición de reunión pero que no están en L1 U L2.

$$\prod_{L_1 \cup L_2}(E_1 \bowtie_{\theta} E_2) = \prod_{L_1 \cup L_2}((\prod_{L_1 \cup L_3}(E_1)) \bowtie_{\theta} (\prod_{L_2 \cup L_4}(E_2)))$$

Reglas de equivalencia

9. Las operaciones de unión e intersección son conmutativas.

$$E_1 \cup E_2 = E_2 \cup E_1$$
$$E_1 \cap E_2 = E_2 \cap E_1$$

10. Las operaciones de unión e intersección son asociativas.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$
$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

11. La selección es distributiva sobre las operaciones de unión intersección y diferencia.

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta (E_1) - \sigma_\theta(E_2).$$

Lo cual también es válido para $\cup$ y $\cap$

Además:

$$\sigma_\theta (E_1 - E_2) = \sigma_\theta(E_1) - E_2$$

Válido para $\cap$, pero no para $\cup$

12. La proyección es distributiva sobre la operación de unión.

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

Ejemplo de transformaciones

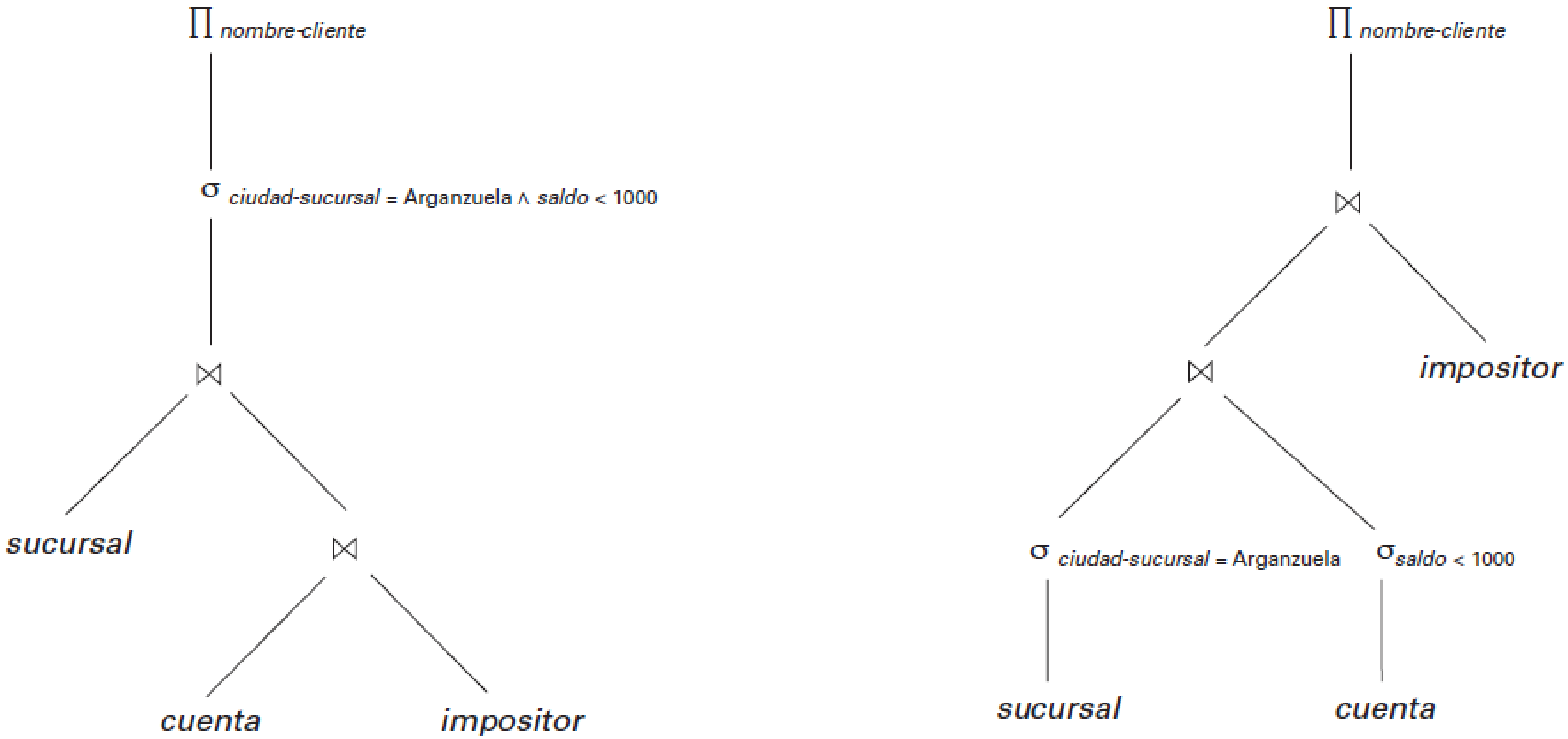
- Considerando los siguientes esquemas:
  - Esquema-sucursal = (nombre-sucursal, ciudad-sucursal, activo)
  - Esquema-cuenta = (número-cuenta, nombre-sucursal, saldo)
  - Esquema-impositor = (nombre-cliente, número-cuenta)
- La consulta en el algebra relacional:

  $\Pi_{\text{nombre-cliente}} (\sigma_{\text{ciudad-sucursal} = «\text{Arganzuela}» \wedge \text{saldo} > 1000} (\text{sucursal} \bowtie (\text{cuenta} \bowtie \text{impositor})))$

  Se transforma en:

  $\Pi_{\text{nombre-cliente}} ((\sigma_{\text{ciudad-sucursal} = «\text{Arganzuela}»} (\text{sucursal}) \bowtie \sigma_{\text{saldo} > 1000} (\text{cuenta})) \bowtie \text{impositor})$

Ejemplo de transformaciones



Orden de las reuniones

- Gracias a la asociación de las operaciones de reunión se puede elegir cual calcular primero en función al tamaño del resultado generado.
- Si se tiene que calcular $r_1 \bowtie r_2 \bowtie r_{3,}$ esto se puede calcular como:
  - $(r_1 \bowtie r_2) \bowtie r_3$, o como
  - $r_1 \bowtie (r_2 \bowtie r_3)$
  - Se elige la opción que requiera de menor espacio temporal.

Orden de las reuniones

- Si se tiene que calcular $(r_1 \bowtie r_2) \bowtie r_3$, pero resulta que r_1 y r_2 no tienen atributos en común, la operación degenera en $(r_1 \times r_2) \bowtie r_3$.
- Si r_1 y r_2 son relaciones grandes se requerirá del uso de mucho espacio temporal
- Si r_3 tiene atributos en común con r_1 y r_2 entonces la operación se puede transformar en $(r_1 \bowtie r_3) \bowtie r_2$.
  - $(r_1 \bowtie r_2) \bowtie r_3$
  - $r_1 \bowtie (r_2 \bowtie r_3)$
  - $r_1 \bowtie (r_3 \bowtie r_2)$
  - $(r_1 \bowtie r_3) \bowtie r_2$
- Por asociatividad se puede elegir la operación que requiera menor tamaño temporal.

Enumeración de expresiones equivalentes

- El optimizador aplica sistemáticamente las reglas de equivalencia para generar expresiones equivalentes a la consulta dada.
- Se pueden generar todas la expresiones como sigue:
  - Repetir hasta que todas las expresiones hayan sido procesadas.
    - Seleccionar una expresión no procesada.
    - Aplicar cada regla de equivalencia aplicable.
    - Agregar las nuevas expresiones al resultado.
- El enfoque anterior es muy costoso en términos de tiempo y espacio.

Elección de los planes de evaluación

- Se debe considerar las diferentes técnicas de evaluación de la operaciones cuando se escogen los planes.
  - Escoger el algoritmo menos costoso de cada operación independientemente no lleva siempre al mejor plan.
    - El join por mezcla podría ser un poco más costos que un join por hash, pero puede proporcionar un resultado ordenado que podría reducir el costo de las operaciones siguientes.
    - El Join por bucle anidado puede dar oportunidad al encausamiento de operaciones.
- En la práctica los optimizadores incorporan elementos de los siguientes enfoques:
  - Buscar todos los planes y escoger el mejor basado en un enfoque de selección basada en costo.
  - Usar heurísticas para escoger los planes.

Optimización Basada en Costo

- Los **optimizadores basados en el coste** generan una gama de planes de evaluación a partir de la consulta dada empleando las reglas de equivalencia y escogen el de coste mínimo.
- Considerar buscar el mejor ordenamiento para $r_1 \bowtie r_2 \bowtie \ldots r_n$.
  - Hay $(2(n - 1))!/(n - 1)!$ órdenes de reunión diferentes.
  - Con n = 7 son 665280 y con n = 10 son ~= 17.600 millones.
- No es necesario generar todos los ordenes, utilizando **Programación Dinámica** se puede calcular el orden de algún subconjunto y luego utilizarlo más tarde.

Programación Dinámica en Optimización

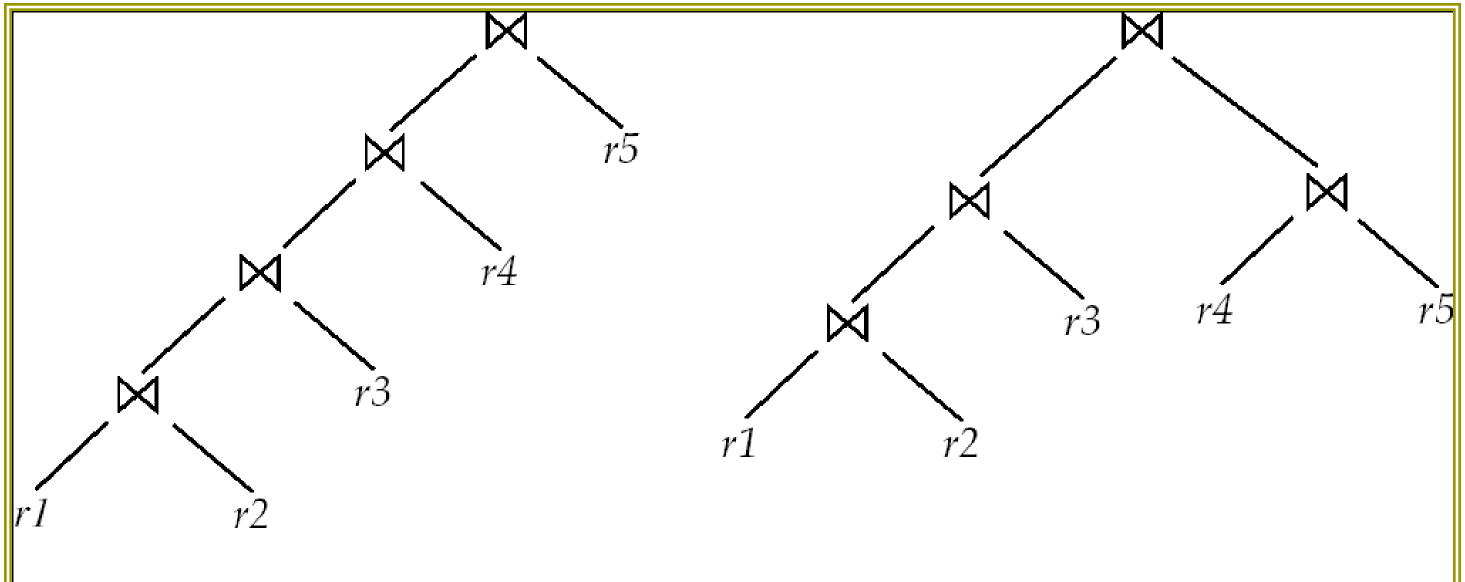
- Para buscar el mejor **árbol de join** de **n** relaciones
  - Considerar todos los planes de la forma: $S_1 \bowtie (S - S_1)$ donde S_1 es un subconjunto no vacío de S.
  - Computar recursivamente el costo de cada subconjunto de reunión de S. Escoger la más barata de las $2^n - 1$ alternativas
  - Caso base de la recursión: acceso a una sola relación.
    - Utilizar el modo de acceso más apropiado.
  - Cuando se computa el plan para cualquier subconjunto, se guarda el resultado para utilizarlo cuando sea requerido nuevamente, en vez de volver a calcularlo.
    - Esto se denomina **Programación Dinámica**

Algoritmo de PD para la Reunión

```
procedure hallarmejorplan(S)
    if (mejorplan[S].coste ≠ ∞)
        return mejorplan[S]
    if (S contiene sólo una relación)
        establecer mejorplan[S].plan y mejorplan[S].coste en términos
        de la mejor forma de acceder a S
    // mejorplan[S] no se ha calculado anteriormente
    else for each subconj. no vacío S1 de S t.q. S1 ≠ S
        P1 = hallarmejorplan(S1)
        P2 = hallarmejorplan(S – S1)
        A = mejor algoritmo para reunir P1 y P2
        coste = P1.coste + P2.coste + coste de A
        if coste < mejorplan[S].coste
            mejorplan[S].coste = coste
            mejorplan[S].plan = «ejecutar P1.plan; ejecutar
                                 P2.plan; reunir resultados de P1
                                 y de P2 utilizando A»
    return mejorplan[S]
```

Árboles de Join

- Los **ordenes de reunión por la izquierda** son aquellos en los que el operando de la derecha de cada reunión es una de la relaciones iniciales.
- Son convenientes para evaluaciones encauzadas**.**
- **El tiempo de consideración de todos los ordenes por la izquierda es O(n!).**



(a) Left-deep join tree (b) Non-left-deep join tree

Costo de la optimización

- La complejidad temporal del algoritmo de P.D. es $O(3^n)$.
  - Con n = 10 son 59.000 pasos en vez de 17.600 M.
- La complejidad espacial es $O(2^n)$
- Para encontrar el mejor ordenamiento en profundidad por la izquierda de **n** relaciones:
  - Considerar **n** alternativas con una de la relaciones como entrada derecha y las otras relaciones como entradas izquierdas.
  - Lo que modifica el algoritmo de optimización de PD
    - Reemplazando el "**for each** subconj. no vacío S1 de S t.q. S1 $\neq$ S" por "**for each** r en S, haciendo S1 = S-r"
- Cuando se consideran solo los ordenamientos en profundidad por la izquierda el costo temporal del algoritmo es $O(2^n)$.
- La optimización basada en costos es costosa, pero vale la pena para BD grandes, en especial si las consultas son pequeñas (n<=10).

Orden Interesante

- Considere la expresión $(r1 \bowtie r2) \bowtie r3$, con un atributo común A.
- Un **orden interesante** es un ordenamiento de las tuplas que puede ser útil en operaciones posteriores.
  - Utilizar Merge-Join para $r1 \bowtie r2$ puede ser mas costoso que un Hash-Join, pero generaría el resultado ordenado por A.
  - Pero esto podría ser útil para hacer un Merge-Join con r3.
  - Un resultado ordenado también es útil para agregaciones, ordenamiento o agrupamiento.
- No es suficiente buscar independientemente el mejor orden de reunión de **n** relaciones.
  - Hay que halla el mejor orden de reunión para cada orden interesante
  - Se extiende el algoritmo de optimización de PD para que aparte del orden de reunión también considera si en el conjunto existe un orden interesante aprovechable.
  - Usualmente solo se da un pequeño número de ordenes interesantes, lo que no aumenta la complejidad temporal y espacial del alg. de PD

Optimización Heurística

- La optimización basa es costo es costosa aun con PD y sus mejoras.
- Los SGBD pueden aplicar Heurísticas para reducir el número de posibilidades a ser consideradas en un análisis de costo.
- Las **Heurísticas** son transformaciones en el árbol de la consulta que por lo general (no siempre) disminuyen el costo de procesamiento.
  - Realizar las selecciones los más antes posible.
    - Descomponer las condiciones conjuntivas.
    - Desplazar las selecciones lo más hacia abajo en el árbol.
  - Realizar las proyecciones los más antes posible.
    - Dividir la lista de atributos de manera a realizar las proyecciones primero (desplazarlas lo más hacia abajo en el árbol).
  - Realizar primero operaciones de join o selección más restrictivas.
  - Sustituir selecciones sobre productos cartesianos por joins basados en la condición de selección.
  - Dar prioridad a la posibilidad de realizar operaciones encauzadas.

Optimización Heurística

- Las heurísticas tratan de reordenar el árbol para reducir el tamaño de los datos intermedios lo antes posible.
- Las heurísticas pueden aplicarse para determinar varios planes basados en caminos de accesos existentes
  - Esta fase de **selección del plan de acceso** del optimizador heurístico selecciona la estrategia más eficiente para cada operación.
- Anteriormente muchos SGBD solo realizaban Optimizaciones basadas en Heurísticas.
  - No tenían un optimizador basado en costo

Estructura de los Optimizadores

- La mayoría de los optimizadores solo considera ordenaciones profundas por la izquierda.
  - Aplicando heurísticas para realizar antes selecciones y proyecciones.
  - Útil para reducir la complejidad de optimización y generar planes que aprovechen el encauzamiento.
- Antiguas versiones de Oracle solo poseían un optimizador heurístico
  - Buscar repetidamente la siguiente “mejor relación” de forma heurística para una reunión.
    - Evaluando n planes para cada una de las n relaciones como punto de partida.
- SQL introduce un alto grado de complejidad en los Optimizadores cuando se presentan consultas anidadas.
  - Subconsultas, op. de conjuntos (unión, intersección, dif.).
  - Se analiza cada parte por separado y luego se combinan en un plan global.

Estructura de los Optimizadores

- Algunos optimizadores combinan heurísticas de selección y estrategias de generación de planes.
  - Un enfoque puede ser:
    - Reescritura de operaciones basadas en heurísticas
    - Optimización del ordenamiento de joins basada en costo.
- Aún cuando la optimización basada en costo puede ser costosa y suponer una sobrecarga:
  - El mejoramiento de consultas costosas supone una ganancia.
- Así los optimizadores aplican heurísticas para consultas sencillas y realizan una enumeración exhaustiva para consultas complejas.

Información Estadística

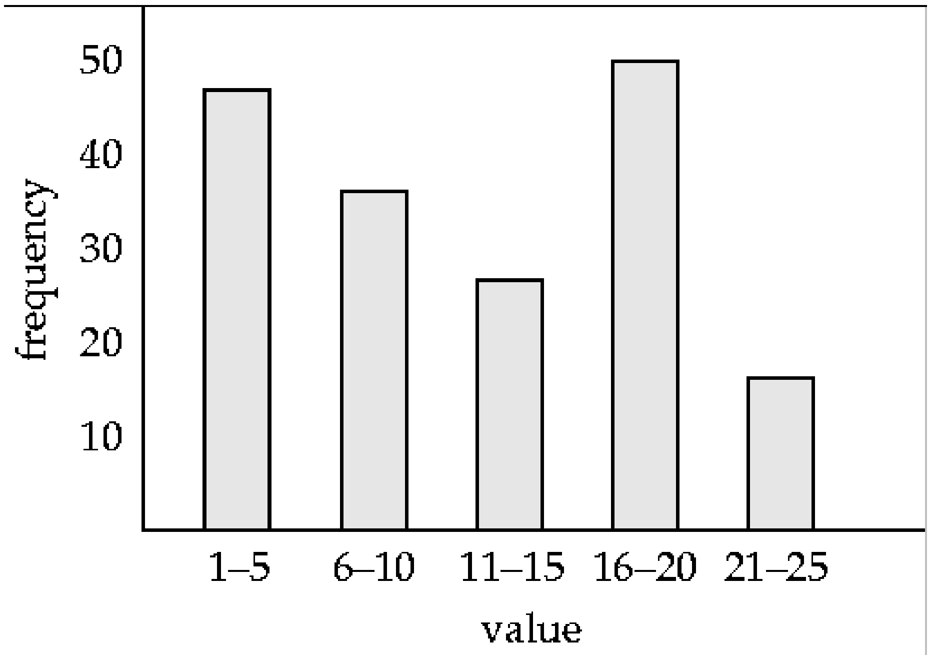
- n_r : número de tuplas en la relación r
- b_r : número de bloques que contienen tuplas de la relación r

$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

- t_r : tamaño de las tuplas de la relación r en bytes
- f_r *:* factor de bloqueo, numero de filas de la relación r que caben en un bloque
- *V(A, r)* : número de valores distintos del atributo o conjunto de atributos *A* en la relación r

Información Estadística

- Histogramas:
  - Guarda información acerca de la distribución de los valores en un atributo de una relación.
  - Esto ayuda al optimizador a estimar con mayor precisión el número de filas.
    - Sin esta información solo se puede suponer que la distribución es uniforme.



Estimación del tamaño de la relaciones

- La estimación del tamaño de una selección depende de la condición
- $\sigma_{A=v}(r)$
  - $n_r / V(A,r)$, lo que supone una distribución uniforme.
    - Aun así normalmente es una aproximación razonable.
  - 1, cuando la condición es sobre una clave candidata.
- $\sigma_{A \leq v}(r)$ (el caso $\sigma_{A \geq v}(r)$ es simetrico)
  - Si están disponible las informaciones min y max en A
    - c = 0 si v < min(A,r)
    - c = $n_r . \dfrac{v - \min(A,r)}{\max(A,r) - \min(A,r)}$
  - Si existe un histograma es posible refinar la aproximación
  - En ausencia de información estadística el tamaño estimado es $n_r / 2$

Selecciones complejas

- La selectividad s_i de una condición θ_i es la probabilidad de que una tupla satisfaga θ_i .
  - $s_i = |\sigma_{\theta 1}(r)|| / n_r$
- **Conjunción:** $\sigma_{\theta 1 \wedge \theta 2 \wedge . . . \wedge \theta n}$ (r). Asumiendo independencia, el numero estimados de tuplas es igual a:

$$n_r * \frac{s_1 * s_2 * \ldots * s_n}{n_r^n}$$

- **Disjuncion:**$\sigma_{\theta 1 \vee \theta 2 \vee . . . \vee \theta n}$ (r). El número estimado de tuplas es:

$$n_r * \left(1 - (1 - \frac{s_1}{n_r}) * (1 - \frac{s_2}{n_r}) * ... * (1 - \frac{s_n}{n_r}) \right)$$

- **Negation:** $\sigma_{\neg\theta}$(r). El número estimado de tuplas es n_r – size(σ_θ(r))

Tamaño de las reuniones

- El producto cartesiano r x s contiene $n_r \times n_s$ tuplas de tamaño $t_r + t_s$ bytes.
- Si $R \cap S = \varnothing$, $r \bowtie s$ se reduce a un producto cartesiano r x s.
- Si $R \cap S$ es una clave de r, cada tupla de s se reúne con al menos una tupla de r, siendo el número de tuplas en $r \bowtie s$ es como máximo igual al número de tuplas de s
- Si $R \cap S$ es una clave foránea que referencia a r, entonces el número de tuplas en $r \bowtie s$ es igual al número de tuplas de s.
  - El caso es simétrico para un clave foránea que referencia a s.

Tamaño de las reuniones

- Si $R \cap S = \{A\}$ no es una clave de r o s, y además se asume que cada fila de r está en el resultado, entonces el número de filas en r $\bowtie$ s es estimado como:

  $$\frac{n_r * n_s}{V(A,s)}$$

  Si se cumple lo inverso, entonces el tamaño se estima por:

  $$\frac{n_r * n_s}{V(A,r)}$$

  Por lo general el menor de estos dos valores es la estimación más exacta.
- Las estimaciones anteriores se pueden mejorar si se disponen de histogramas

Tamaño de otras operaciones

- Proyección: tamaño estimado de $\Pi_A(r) = V(A,r)$
- Agregación: tamaño estimado de $_A g_F(r) = V(A,r)$
- Operaciones de conjuntos
  - Uniones / Intersecciones en la misma relación:
    - Se aplica el tamaño estimado de las selecciones
    - $\sigma_{\theta 1}(r) \cup \sigma_{\theta 2}(r) = ||\sigma_{\theta 1}(r)|| + ||\sigma_{\theta 2}(r)||$
  - Para operaciones en diferentes relaciones:
    - $r \cup s = n_r + n_s$.
    - $r \cap s = \min(n_r, n_s)$.
    - $r - s = n_r$.
    - Todos estos valores pueden no ser una buena estimación, sin embargo proporcionan un limite superior para los valores reales.
  - Outer join:
    - r ⟕ $s = ||r \bowtie s|| + n_r$ (right outer join es simetrico)
    - r ⟗ $s = ||r \bowtie s|| + n_r + n_s$

Estimación de valores distintos

Selecciones : $\sigma_\theta\,(r)$

- Si θ indica un valor específico ($\theta \rightarrow$ A= 3), etonces: $V(A, \sigma_\theta\,(r)) = 1$.
- Si θ indica un conjunto de valores:
  - $V(A, \sigma_\theta\,(r))$ = número de valores especificados.
  - $\theta \rightarrow (A = 1$ *or* $A = 3$ *or* $A = 4)$
    - $V(A, \sigma_\theta\,(r)) = 3$
- Si la condición θ es de la forma *A* ***op*** *r*, siendo **op** un operador de comparación
  - $V(A, \sigma_\theta\,(r)) = V(A, r) * s$
  - Donde s es la selectividad de la operación.
- Los casos restantes se aproximan por:
  - $\min(V(A,r), n_{\sigma\theta\,(r)})$

Estimación de valores distintos

Joins: $r \bowtie s$

- Si todos los atributos en A son de r, la estimación es:
  - $V(A, r \bowtie s) = \min(V(A, r), n_{r \bowtie s})$
  - Caso simétrico para s
- Si A contiene atributos $A1$ de r y $A2$ de s, entonces la estimación esta dada por:
  - $V(A, r \bowtie s) = \min(V(A1, r) * V(A2 - A1, s),$
    $V(A1 - A2, r) * V(A2, s), n_{r \quad s})$

Estimación de valores distintos

- Los valores distintos en proyecciones, agrupaciones y agregaciones son iguales al de la relación.
  - V(A, *r*)
- En operaciones de agregación:
  - Para min(*A*) y max(*A*), el número de valores distintos se estima como min(V(*A, r*), *V*(*G, r*)), donde G denota a los atributos de agrupamiento.
  - Para otras agregaciones (suma, cuenta, promedio) se supone que todos los valores son distintos usándose como estimación a *V*(*G, r*)