UNIVERSIDAD DE GRANADA

DECSAI

Administración de Bases de Datos

Grado en Ingeniería Informática

Tema 2 – Optimización de consultas


I. J. Blanco, A. G. López Herrera

Departamento de Ciencias de la Computación e Inteligencia Artificial

http://decsai.ugr.es

- El problema de la optimización de consultas
- Conversión de árboles de expresión del álgebra relacional
- Transformación de consultas
- Estrategias para la optimización
- Completando planes físicos de consulta

- Organizar más adecuadamente la información a nivel físico no es lo único por hacer cuando se optimiza el funcionamiento de una DB.
- Las operaciones más ejecutadas en una base de datos son las consultas, y estas pueden ser optimizadas (transformadas antes de su ejecución para que se ejecuten de la forma más eficiente posible -en tiempo y espacio-), pero este proceso también gasta tiempo.

- Lenguaje de consulta: lenguaje que nos permite obtener datos de una BD a través de operadores, por ejemplo, SQL.
- Optimizador: modulo del SGBD que efectúa modificaciones en la consulta para obtener los datos consultados de la forma más eficiente tratando de reducir:
  - El tiempo de evaluación o
  - El tiempo de respuesta

- El coste minimizable se basa en:
  - coste de acceso a almacenamiento secundario
  - coste computacional (basado en el número de comparaciones)

- Base de datos:
  - Alumno (DNI, nombre, fec_nac, ciudad, direccion, tlfno, beca)
  - Asignatura (codigo, nombre, creditos, carácter, curso)
  - Matricula (codigo, DNI, calificacion)
- Consulta:

```
SELECT alumno.nombre FROM alumno, matricula
WHERE alumno.DNI = matricula.DNI AND beca = 'N'
AND calificacion LIKE 'SOBRESALIENTE HONOR%';
```

- Hipótesis:
  - Ordenación: O ($n \cdot \log_2(n)$)

Relación	NTuplas	Bfr	Cond1	Cond2	Cond1 Y Cond2
Alumno	500	5	100		15
Matricula	5000	10		120	

Primer plan:

- Producto cartesiano:
  - Registros: 500 * 5000
  - Bloques de E/S: 500/5 * 5000/10 = 50000 bloques
  - Tuplas resultantes: 2500000
  - Tendrá que escribir a disco (Bfr = 3) luego habrá 833334 bloques
- Selección:
  - Leer todos los registros, 833334 bloques
  - Escribir 15 registros de resultado (5 bloques)
- Proyección
  - Leer los 5 bloques de la operación anterior
  - Sólo nombre de 15 registros caben en un bloque.

Primer plan:

50000 + 833334 * 2 + 5 + 5 + 1 = 1716679 bloques

¡Muy costoso!

Segundo plan:

- Ordenar matrícula:
  - 5000 registros en bloques de 10, 500 bloques
  - $500 * \log_2(500) = 4483$ bloques E/S
- Mezclar alumno y matricula por DNI:
  - 500 bloques + 100 bloques = 600 bloques
  - Se escriben 5000 registros de reunión (tamaño mayor y Bfr = 3), o sea, 1667 bloques
  - En total, 1667 + 600 bloques = 2267 bloques de E/S

Segundo plan:

- Selección de tuplas:
  - Leer 5000 registros, o sea 1667 bloques
  - Escribir 15 registros de resultado, o sea, 5 bloques
  - En total, 1667 + 5 = 1672 bloques
- Proyección:
  - Leer los 5 bloques de la operación anterior.
  - Escribir 15 registros de sólo nombre, que caben en un bloque

Segundo plan:

4483 + 600 + 1667 * 2 + 5 + 5 + 1 = 8428 bloques

Mucho mejor

Tercer plan: ¡primero las selecciones!

- Alumnos no becados:
  - Se leen 500 registros, o sea, 100 bloques
  - Se escriben 100 registros, o sea, 20 bloques
  - En total, 100 + 20 = 120 bloques
- Seleccionar matrículas con esa calificación:
  - Se leen 5000 registros, o sea, 500 bloques
  - Se escriben 120 registros, o sea, 12 bloques
  - En total, 500 + 12 = 512 bloques

Tercer plan:

- Ordenar las matrículas seleccionadas:
  - 12 bloques * log2 (12) = 43 bloques
- Reunir alumnos seleccionados (100) y matrículas seleccionadas (120):
  - Se leen 20 + 12 = 32 bloques
  - Se escriben 5 bloques
- Proyección:
  - Leer los 5 bloques escritos en la operación anterior.
  - 15 registros de sólo nombre en 1 único bloque

Tercer plan:

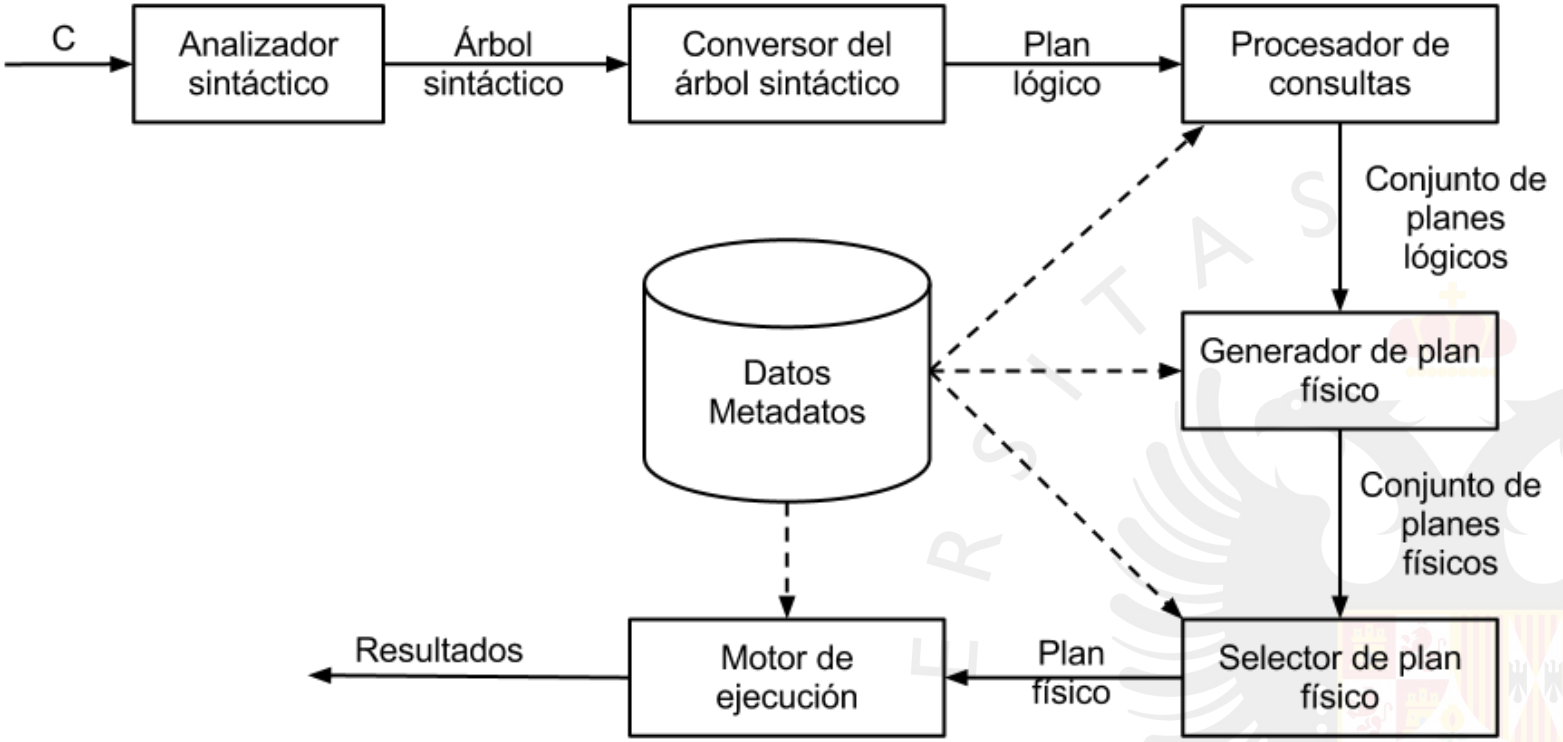
120 + 512 + 43 + 32 + 5 + 5 = 717 bloques

Todavía mejor

Cuarto plan: ¿y si hay un índice sobre calificación?

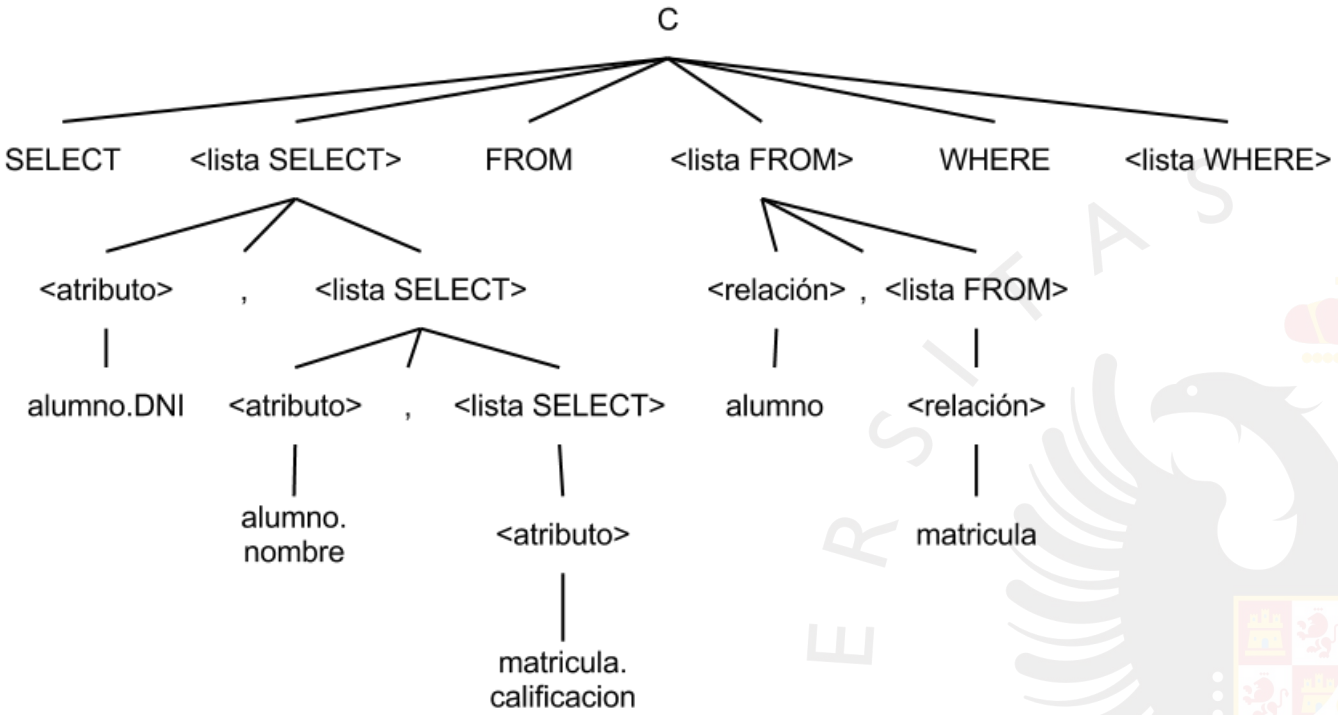
- Usar el índice para seleccionar los DNIs con esa calificación:
  - 120 registros se recuperan en log2 (5000) = 13 bloques
- Ordenar las tuplas resultantes por DNI
- Seleccionar los alumnos becados
- Mezclar ambos

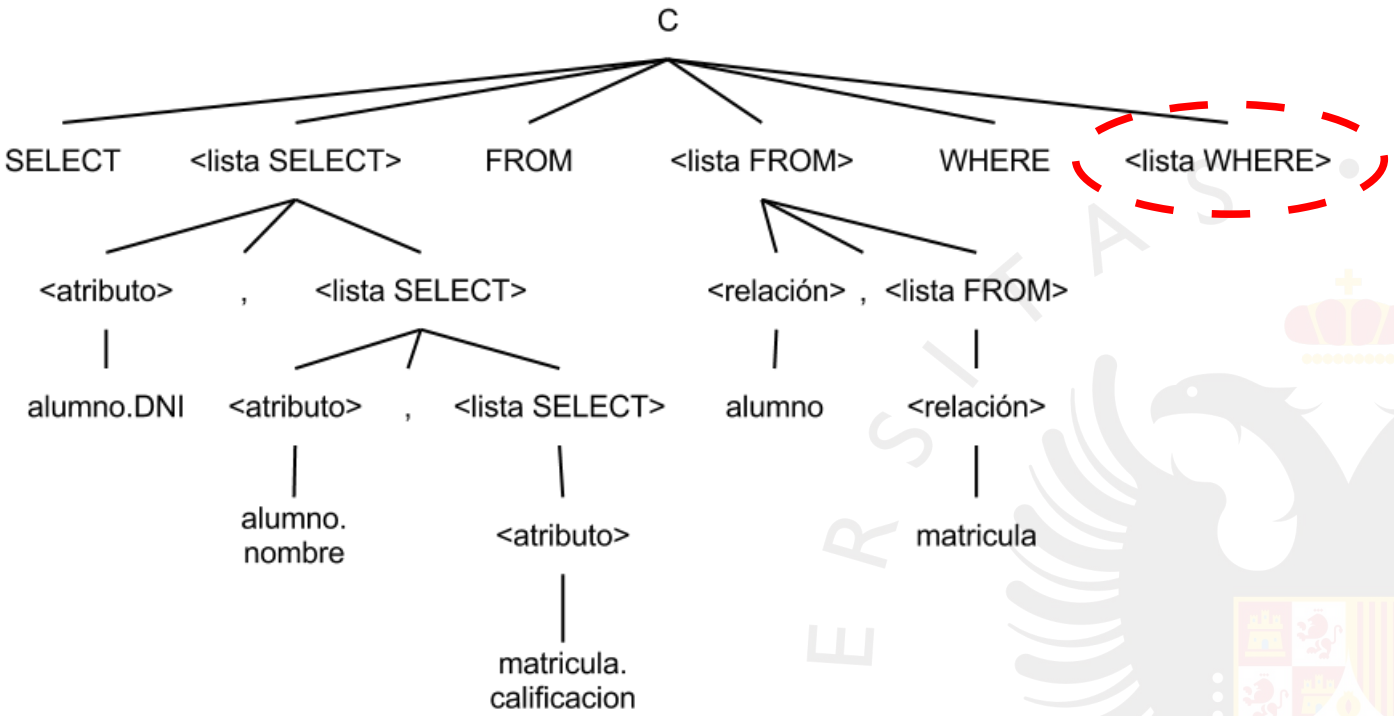
- El uso de lenguajes relacionales (cálculo y álgebra) tan estructurados permiten y aconsejan el uso de optimizadores.
- Un humano puede proponer consultas eficientes pero un optimizador lo es más porque:
  - usa información “privilegiada”
  - si hay reorganización de los datos se puede necesitar re-optimización (en relacional, se reprocesa la consulta; en no relacional, hay que cambiar el optimizador)
  - evalúa muchas más posibilidades (no sólo cuatro, como nosotros)

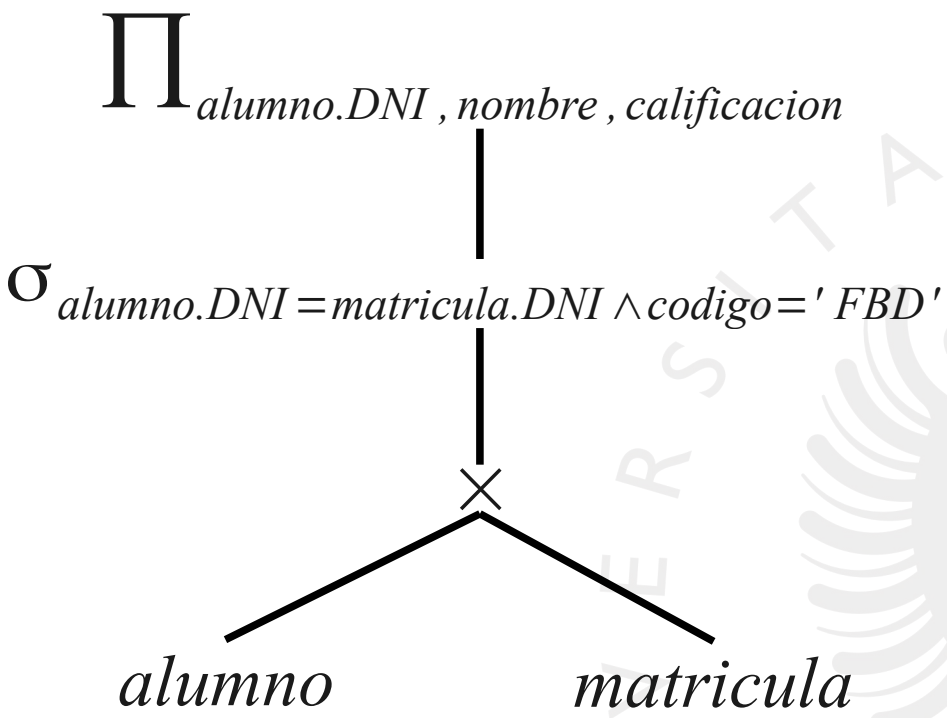


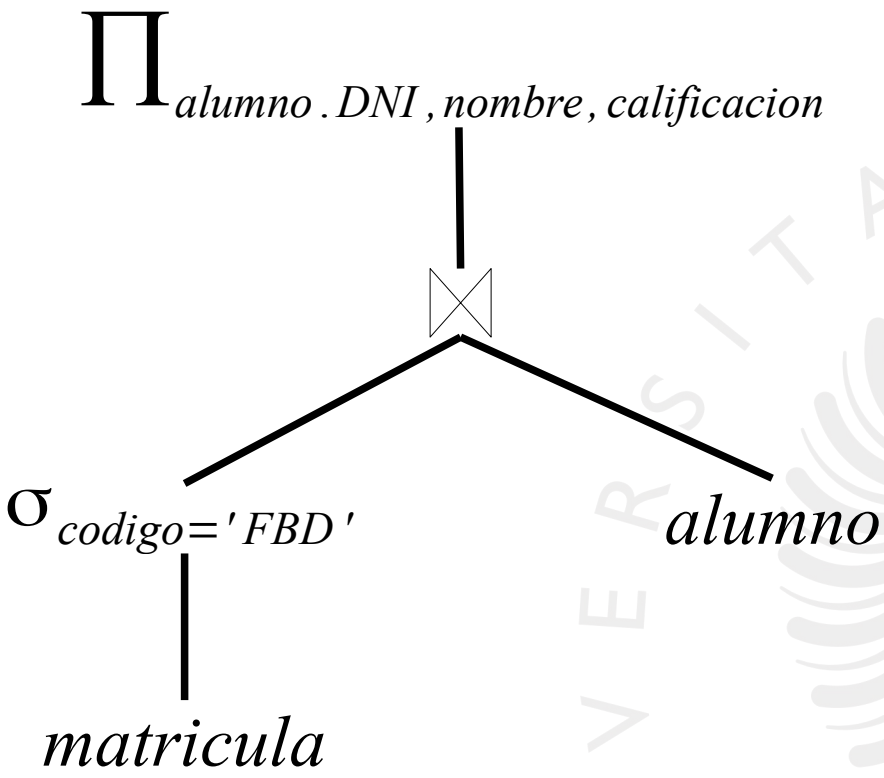
- Herramienta: árbol de expresión algebraica o expresión AR llamado plan lógico
- Hojas: relaciones
- Nodos intermedios: operadores
- Enlace: orden de aplicación

```
SELECT alumno.DNI, alumno.nombre,
matricula.calificacion FROM
alumno, matricula WHERE
alumno.DNI = matricula.DNI AND
codigo = 'FBD';
```

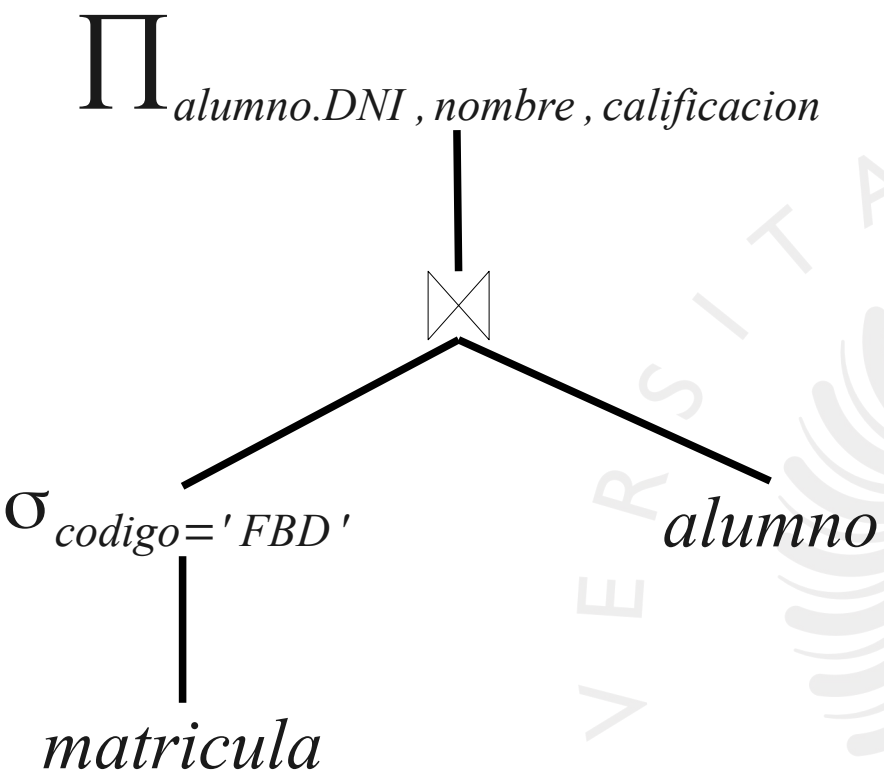








- A partir de una expresión AR puedes encontrar un número limitado de expresiones equivalentes mediante una serie de reglas.
- Transformamos para:
  - simplificar (eliminar redundancias - idempotencia, relaciones vacías, ...-)
  - mejorar la evaluación



- Conmutatividad de producto, reunión, unión, intersección
- Asociatividad de producto, reunión y unión
- Movimiento de selección
- Movimiento de proyección
- Producto cartesiano

- Tenemos varias expresiones equivalentes que difieren en el número de parámetros (tamaño de resultados intermedios, número de tuplas de las relaciones intermedias, …)
- El problema es evaluar el coste de cada una, que también consume un coste.
- En general, con *n* relaciones hay muchas posibilidades:

$$\frac{(2 \cdot n - 2)!}{(n-1)!}$$

- Un optimizador debe generar un conjunto de planes lo suficientemente amplio para contener el óptimo y lo suficientemente pequeño como para que valga la pena evaluar las posibilidades.

- Opciones:
  - Generar un plan según alguna heurística que no es óptimo pero es bueno.
  - Generar todos los planes lógicos posibles, pero sólo funciona para dos relaciones como mucho y crece exponencialmente con la complejidad de la consulta.
  - Una solución mixta: generar un plan según una heurística y varios planes posibles a partir de él para poder elegir.

- Cada plan lógico genera uno o varios planes físicos, y cada uno incluye:
  - información estadística de datos
  - existencia de índices
  - por cada operador del plan, un algoritmo para aplicarlo
  - los algoritmos de ordenación necesarios

Estimación de costes:

- Ejecutar el plan para ver su coste no es eficiente.
- Al principio del plan, los datos sobre las relaciones están en el catálogo.
- Conforme avanza la evaluación, dependen de relaciones intermedias y es necesario estimar a partir de estadísticas de los datos básicos.

Para una relación:

- N(R): Número de tuplas de R
- L(R): longitud en Bytes de las tuplas de R
- Bfr(R): factor de bloqueo
- B(R): número de bloques de la relación R

Para un atributo:

- V(R, atr): número de valores distintos
- nI(R, atr): si hay indice sobre el atributo, cuántos niveles tiene un índice multinivel o un B-árbol
- Para el sistema: T(B) es el tamaño del bloque en Bytes

- Dada una relación R, la proyección sobre un subconjunto X tendría un tamaño:

$$L(X) = \sum_{atr_i \in K} \text{size}(atr_i)$$

$$Bfr(X) = parte\ entera\left(\frac{T(B) - C}{L(X)}\right)$$

$$B(X) = redondeo\ hacia\ arriba\left(\frac{N(X)}{Bfr(X)}\right)$$

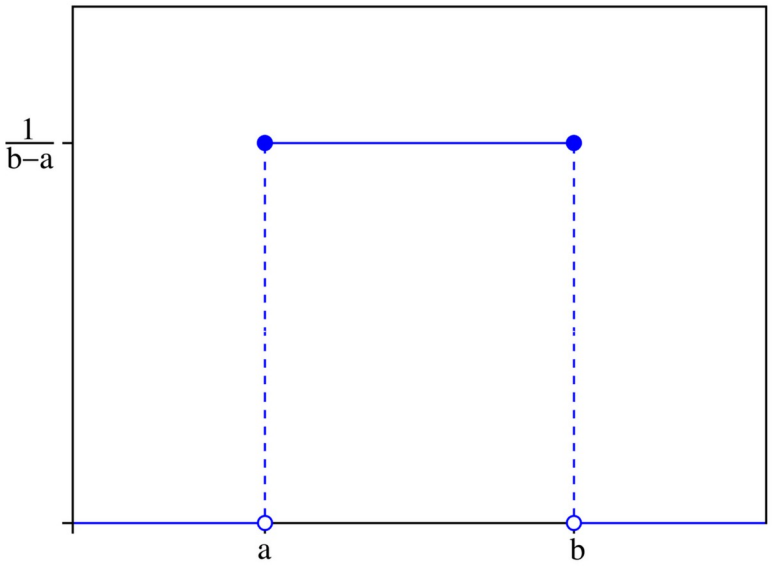
Un ejemplo:

- R (a, b, c) con a y b enteros (2 B) y c cadena de 256 caracteres (256 B), la cabecera ocupa 16 B (C) y el bloque tiene un tamaño T(B) = 2048 B. R tiene 1000 registros.
- Con ello sabemos que L(R)=260 B, que Bfr(R) = 7 y que B(R)=143 bloques.
- ¿Cuál es el factor de bloqueo de $\pi_{a,b}(R)$?

$$Bfr(X) = \frac{2048 - 16}{4} = 508$$

$$B(X) = redondeo\ hacia\ arriba\left(\frac{1000}{508}\right) = 2$$

- Supongamos una distribución uniforme sobre los valores de un atributo.



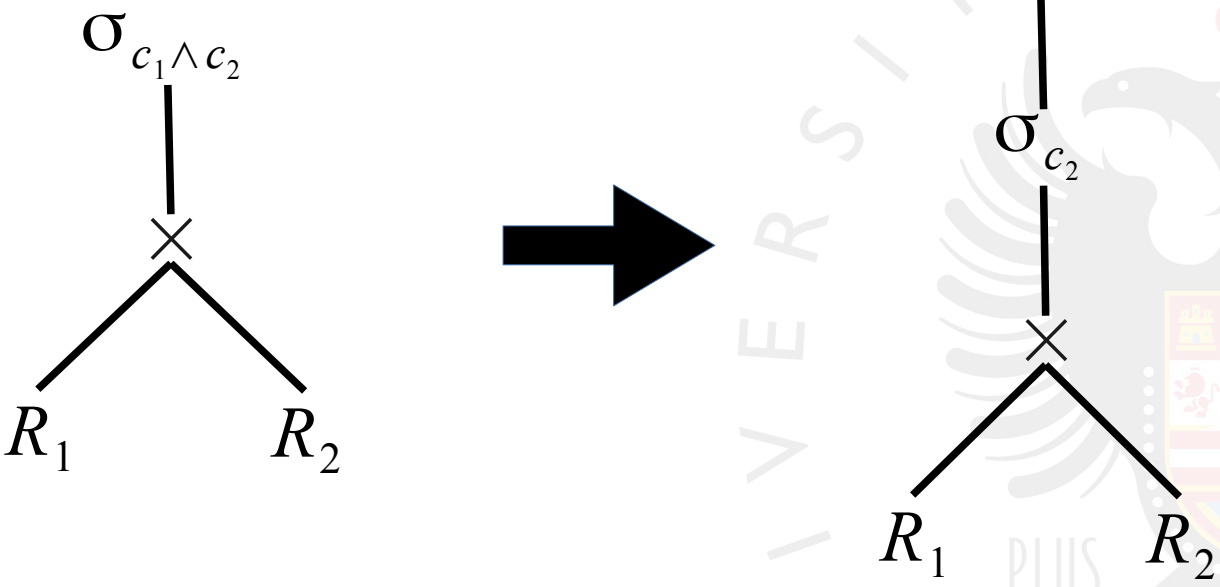
- Dada una relación R, la selección sobre una condición c tendría un tamaño estimado de:

$$N(X) = \alpha \cdot N(R)$$

$$\text{donde } \alpha = \begin{cases} \dfrac{1}{V(R, atr)} & \text{si } \sigma_{atr = valor} \\ \dfrac{1}{3} & \text{si } \sigma_{atr < valor} \\ 1 & \text{si } \sigma_{atr \neq valor} \end{cases}$$

¿Y para condiciones compuestas?

- **AND**: aplicamos las selecciones en cascada y multiplicamos los factores de selección



¿Y para condiciones compuestas?

- **OR**: sustituimos por una unión de selecciones y la estimación es:

$$N(R)\cdot\left(1-\left(1-\frac{n_1}{N(R)}\right)\cdot\left(1-\frac{n_2}{N(R)}\right)\right)$$

- En el primer ejemplo, había dos condiciones que cumplir sobre una reunión natural. Ahora podemos saber cuál de las dos es más selectiva.

$$c_1 \equiv \sigma_{Beca='N'}(Alumnos)$$

$$c_2 \equiv \sigma_{Calificación='SOBRESALIENTE\ HONOR'}(Matrículas)$$

Relación	NTuplas	Bfr	V (Alumnos Beca)	V(Matrículas, calificación)
Alumno	500	5	2	
Matricula	5000	10		6

- Al haber dos valores distintos para Beca podemos suponer que la mitad tienen beca y la mitad no tienen. Luego el número de tuplas del resultado en c_1 sería 250
- El factor de bloqueo de c_1 es el mismo que el de Alumnos (5), luego hacen falta 50 bloques para el resultado de c_1

- Al haber seis valores distintos para *Calificación* podemos suponer que una sexta parte tienen la calificación buscada. Luego el número de tuplas del resultado en c_2 sería 833
- El factor de bloqueo de c_2 es el mismo que el de Matrículas (10), luego hacen falta 84 bloques para el resultado de c_2

Podemos decir que c_1 es más selectiva que c_2

Número de tuplas: $N(X) = N(R) \cdot N(S)$

Longitud de tupla: $L(X) = L(R) + L(S)$

- La estimación es posible únicamente si se supone que para uno de los atributos de uno de los operandos no se pierde ningún valor al reunir (si había seis valores antes de reunir, hay seis valores después de reunir):

$$V(R,a)=V(R \ \ JOIN \ \ S,a)$$

- Si la intersección tiene un atributo en común *b*, que toma los mismos valores en ambas tablas:

$$V(R,b)=V(S,b)$$

$$N(X)=\frac{N(R)\cdot N(S)}{V(R,b)}=\frac{N(R)\cdot N(S)}{V(S,b)}$$

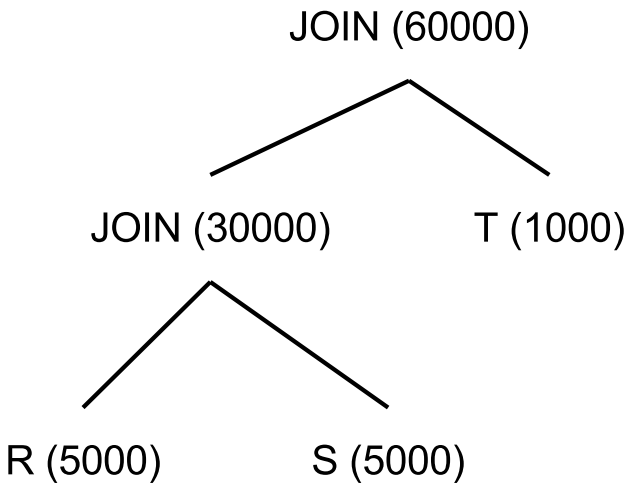
- Y, en general:

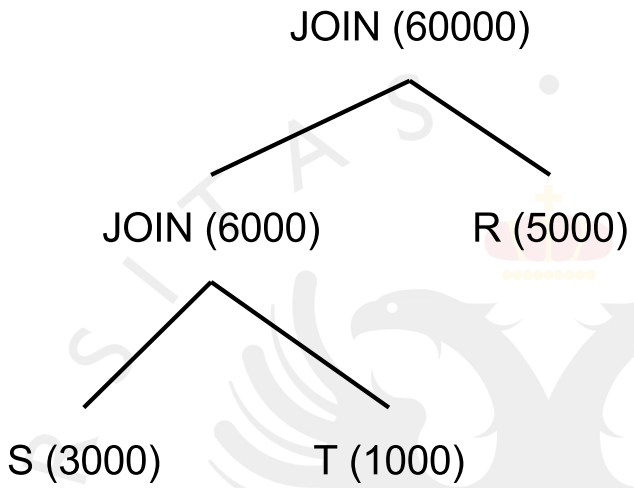
$$N(X) = \frac{N(R) \cdot N(S)}{max\{V(R,b), V(S,b)\}}$$

$$L(X) = L(R) + L(S) - \text{size}(b)$$

$$R \ \ JOIN \ \ S \ \ JOIN \ \ T$$

R(a,b)	S(b,c)	T(c,d)
N(R)=5000	N(S)=3000	N(T)=1000
V(R,a)=200		
V(R,b)=500	V(S,b)=400	
	V(S,c)=500	V(T,c)=100
		V(T,d)=20





- Los SGBD aplican heurísticas de transformación de consultas que, en general, mejoran la eficiencia:
  - selección lo antes posible,
  - combinar producto cartesiano y selección en operaciones de reunión,
  - aplicar la asociatividad para reordenar los nodos hoja de un árbol para realizar las operaciones más restrictivas lo antes posible (lo más a la izquierda y abajo)
  - proyecciones donde sea posible

- Mostrar los DNIs de los alumnos becados y sus calificaciones, para las asignaturas de quinto curso.

```
SELECT AL.DNI, nombre_al, calificacion
FROM alumnos AL, matricula MAT, asignaturas
AS
WHERE AL.DNI = MAT.DNI AND
   MAT.cod_asig = AS.cod_asig AND
   curso = 5 AND
   beca = 'S';
```

- Varios planes lógicos para cada consulta.
- Varios planes físicos para cada plan lógico (conjuntos de operaciones básicas diferentes)
- Un SGBD puede tener varias implementaciones para cada operación del álgebra relacional (varias selecciones, reuniones, ...)

```
Inicio
  Para cada tupla r de la tabla R
    Para cada tupla s de la tabla S
      Si r y s cumplen la condición de
      reunión entonces
        (componer la tupla t del resultado)
    Fin
  Fin
Fin
```

```
Inicio
   Para cada bloque bs de S
      Para cada bloque br de R
         Para cada tupla r de la tabla R
            Para cada tupla s de la tabla S
               Si r y s cumplen la condición de reunión
               entonces
                  (componer la tupla t del resultado)
            Fin
         Fin
      Fin
   Fin
Fin
```