CC42A – BASES DE DATOS
Profesores: Claudio Gutiérrez, Gonzalo Navarro
Auxiliar: Mauricio Monsalve

Auxiliar 8

- Optimización de consultas y estimación de costo

<Pauta tipo tutorial>

Optimización lógica de consultas

Ayudamemoria de la heurística:

0.- Se escribe la consulta SQL en álgebra relacional con operaciones básicas (i.e. sin join natural, en vez de eso colocar la selección del producto cruz, etc.). ***Dibujar el árbol de derivación.***

1.- Expandir las selecciones en selecciones más simples, una tras otra, según la siguiente propiedad::

$$\sigma_{a \wedge b \wedge (c \vee d)}(X) = \sigma_a(\sigma_b(\sigma_{c \vee d}(X)))$$

Redibujar el árbol de derivación.

2.- Colocar las selecciones de forma que queden lo más cerca de las hojas sin alterar la integridad de la consulta (típicamente hasta el producto cartesiano más pequeño sobre el cual pueden operar). Típicamente se pueden usar ***joins*** pues están programados en los optimizadotes de consultas.

3.- Colocar tras cada selección (y antes del siguiente producto cartesiano) una proyección que restrinja los atributos usados. Ojo que los atributos restantes deben ser suficientes para realizar las proyecciones y selecciones que se harán después. ***Redibujar el árbol de derivación.***

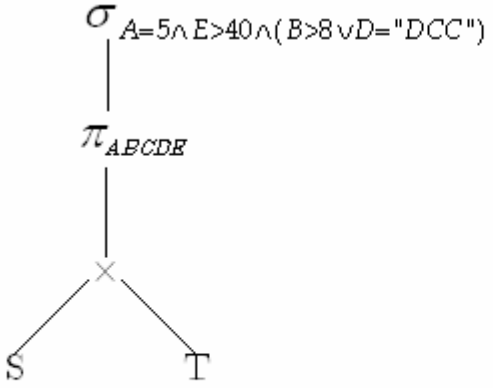
4.- Elegir las ramas a ejecutar primero (a forma de bloques) para reducir el tamaño de la memoria usada por la consulta. Recordar que **las ramas nacen en los productos cartesianos** dada la bifurcación que generan estos últimos. ***Redibujar el árbol de derivación y encerrar los bloques a ejecutar.*** Ojo: Si hay **N** joins, entonces, a lo más, debieran haber **N+1** bloques.

O SEA: Descomponer las selecciones, bajar las selecciones lo más posible, hacer proyección sobre los atributos relevantes lo antes posible (bajar proyecciones) y agrupar por bloques (a.k.a grupos de evaluación).

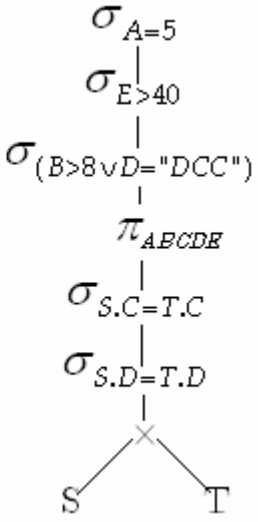
Ejercicios:

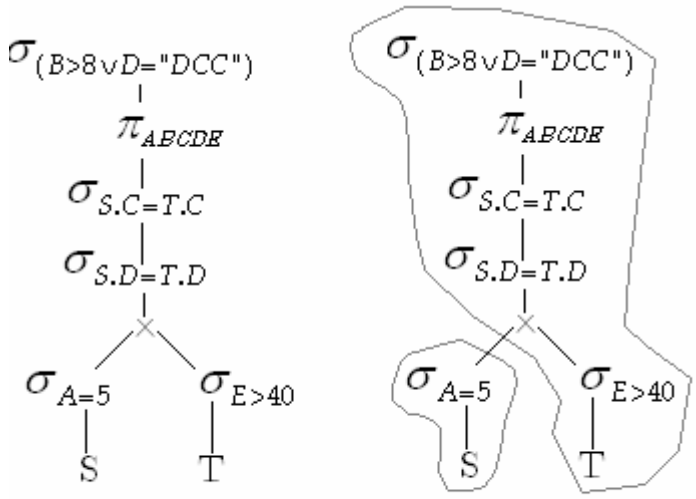
1) **Optimice:** $\sigma_{a \wedge b \wedge (c \vee d)}(S * T)$, con S(A,B,C,D), T(C,D,E), a:A=5, b:E>40, c=B>8, d="DCC".

Hay que notar la presencia del join natural; se debe expandir.



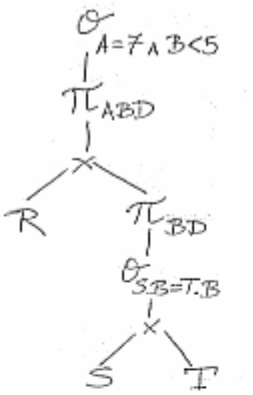
El primer paso es dibujar el arbol de evaluacion de la consulta. Luego hay que expandir los sigma.



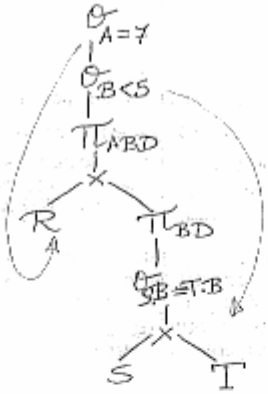


Luego los sigma se bajan lo mas posible. Despues se "bajan" las proyecciones, innecesarias en este caso. Despues se agrupa en bloques de evaluacion.

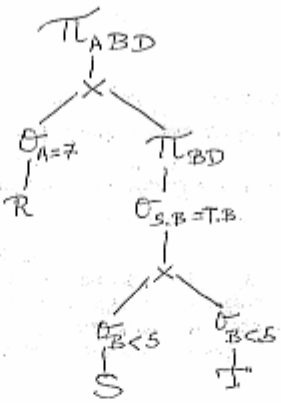
2) **Optimice:** $\sigma_{A=7 \wedge B<5}(\pi_{A,B,D}(R \times (S * T)))$, con R(A,C), S(B,D,E), T(B,F,G).



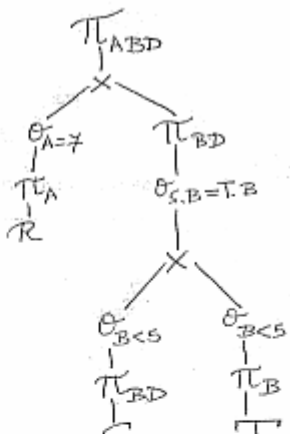
Dibujo el arbol de la consulta original



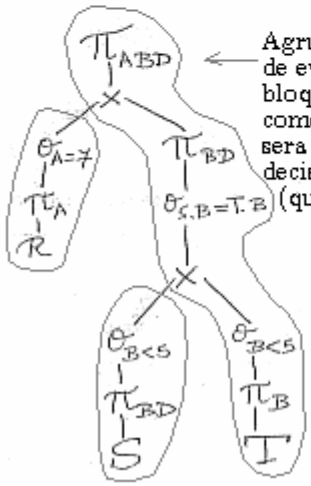
Despedazo los sigma. Veo que los puedo bajar bastante (ver flechas).



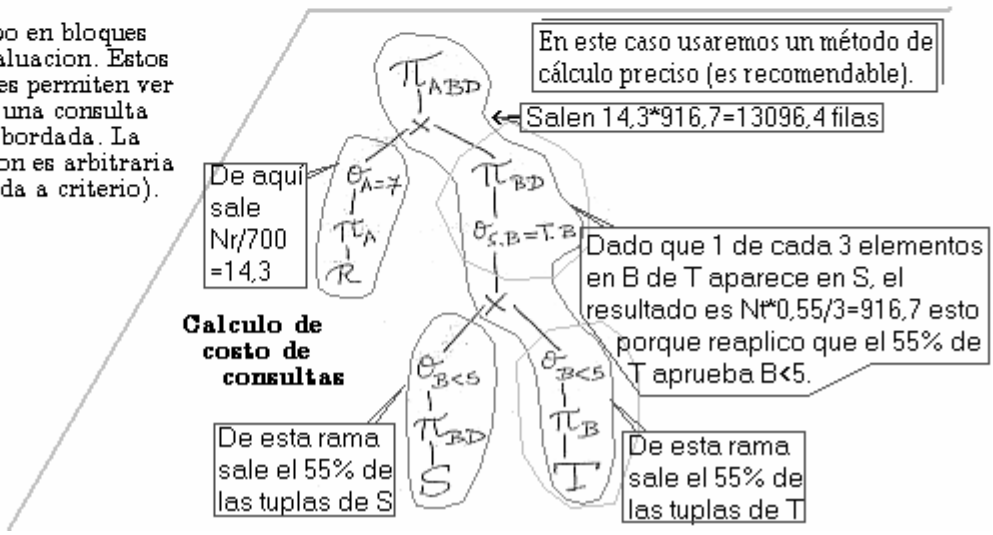
Bajo los sigma lo mas posible.



Aplico proyeccion segun sea posible.



Agrupo en bloques de evaluacion. Estos bloques permiten ver como una consulta sera abordada. La decision es arbitraria (queda a criterio).



También se usa otro método con estimación de costo (tamaño) de diferentes estrategias de consultas. Este método puede ser más preciso, pero es más complejo y costoso en tiempo y la heurística entrega buenos resultados.

Estimación de costo de consultas

Para estimar el costo de las consultas (costo en tamaño de tablas) hay que usar la intuición.

- Si conocemos el tamaño de un registro en bloques (b_R) y el número de registros de una tabla R (n_R), entonces los bloques usados son $\boldsymbol{b_R \cdot n_R}$. El tamaño en bytes de un registro es t_R, y se usa igual que b_R.
- Si conocemos el número de valores diferentes que asume un atributo *X* en una tabla *Y* (esto es **V(Y,X)**) de n_Y registros (filas), entonces se puede estimar que el número de veces que aparece repetido ese atributo es:

$$\frac{n_Y}{V(Y,X)}$$

(Notación, V(tabla,atributo)). Se ha hecho el supuesto de distribución uniforme de datos.

- También se pueden hacer estimaciones del % de registros que satisfacen una consulta. En esos casos se reduce el número de registros según ese porcentaje.

Ejercicios:

1) Calcular los costos de ejercicio 2 de la parte anterior teniendo en cuenta: N_R=10.000, N_S=7.000, N_T=5.000, *uno de cada tres valores de B en T aparece en S*, el 45% de los valores de B de S y T son mayores o iguales a 5, V(R,A) = 700. Suponer que cada atributo usa el mismo espacio de disco (uniformidad). **Con esto estime la carga de la consulta sin y con optimización y estime la mejora.**

*El cálculo de costos ya se hizo. La carga de la consulta queda propuesta, pero es simplemente sumar el resultado de cada sub-bloque de consulta. Ojo: El resultado debiera entregar una **GRAN** diferencia (casi no hay comparación).*

Problemas más difíciles

1) Suponga que le cambian las reglas de optimización de consultas y puede hacer un triple join (tres tablas y condición), ¿cómo optimizaría $\sigma_{A=7 \wedge B<5}(\pi_{A,B,D}(R \times (S * T)))$?

La solución es sencilla y va por la parte de que se hace un producto cruz especial que saca 3 ramas en vez de dos, lo que permite obtener más proceso con menos altura de árbol. Las reglas siguen siendo las mismas porque un triple join es una condición sobre la cruza de tres tablas (aquí RxSxT). El triple join absorberá la triple cruza y la consulta en sólo una operación. ☺

2) Escriba la consulta SQL que calcula V(*Tabla*,*Atrib*). Si *Atrib* es clave, ¿cuánto vale V?

Esta es la consulta SELECT COUNT(Distinct Atrib) FROM Tabla;

3) γ y δ son agrupar y eliminar duplicados. Indique algunas propiedades para álgebra relacional.

Por ejemplo, la agrupación es conmutativa, eliminar duplicados luego de agrupar no tiene efecto, eliminar duplicados dos veces o agrupar dos veces sobre un mismo atributo es lo mismo que hacer la operación una vez (eso es bastante trivial), etc. las propiedades son bien pocas y son poco potentes.