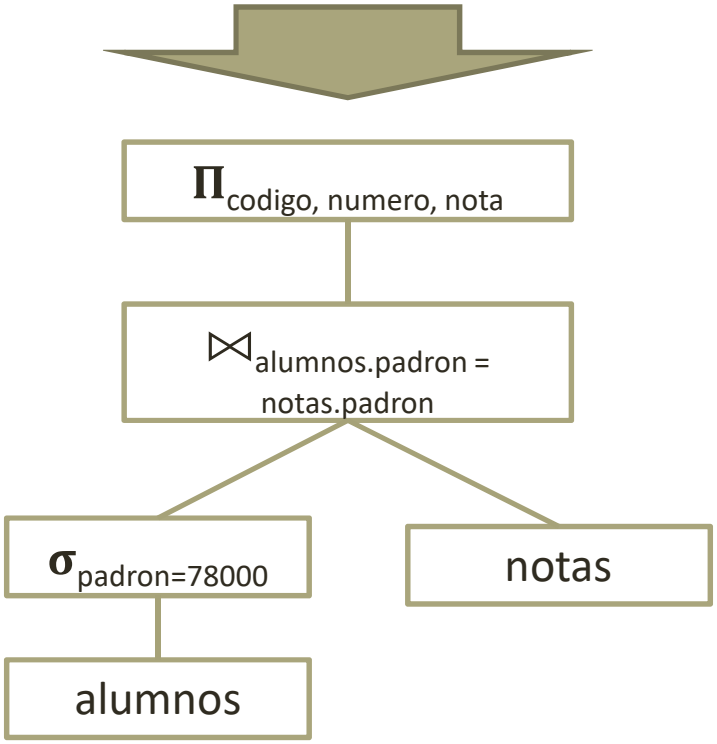
Resolución de consultas

Resolución de consultas

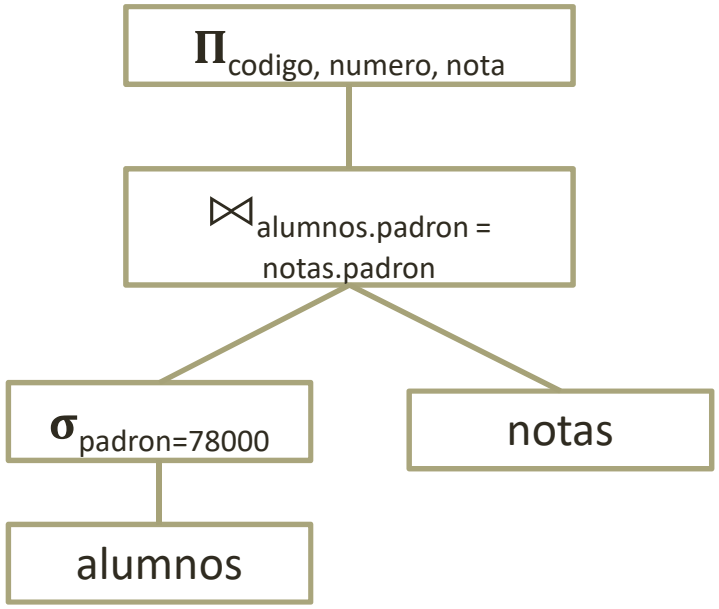
```
SELECT n.codigo, n.numero, n.nota
FROM alumnos a INNER JOIN notas n USING (padron)
WHERE a.padron = 78000
```

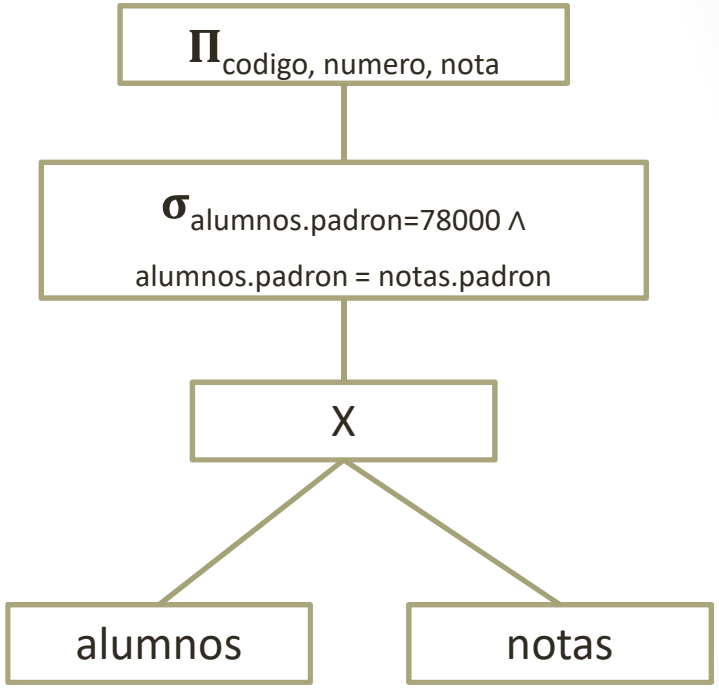


Resolución de consultas

- Dos factores importantes:
  - Cuál árbol de consulta utilizar
    - Puede haber varios equivalentes
    - Pueden influir el costo y los tamaños de relaciones intermedias
  - Cómo resolver cada operador algebraico
    - Dependen de tamaño de tabla, orden físico e índices existentes
- En base a esto, se genera un plan de ejecución
  - Se pueden cachear para consultas similares posteriores
  - Se guarda cuando la consulta es almacenada en la base en vez de ser ejecutada

Árboles de consulta





- ¿Cuál será más óptimo?

Árboles de consulta

- Realizar selecciones lo antes posible
- Reemplazar productos cartesianos y selecciones por juntas
- Proyectar atributos lo antes posible
  - Priorizar selección
- Realizar las juntas más restrictivas primero
- Objetivo general: reducir tamaño de relaciones intermedias

Costos de árbol de consulta

Resumen metadatos

- Sobre las tablas
  - n(R): filas de la tabla R
  - B(R): bloques de la tabla R
  - F(R): filas de R que entran en un bloque (⌈ n(R)/B(R) ⌉)
  - V(A,R): cuantos valores distintos tiene el atributo A en la tabla R
- Sobre cada índice
  - Height(Idx): altura del índice

Resumen costos por operación

σ

- File Scan: B(R)
- Index Scan: Height(Idx) + 1 *(idx primario)*
  Height(Idx) + ⌈ n(R) / V(Ai,R) ⌉ *(idx sec.)*
  Height(Idx) + ⌈ B(R) / V(Ai,R) ⌉ *(idx clust)*

⋈

(NB = bloques en memoria: 1 se usa para R, otro para acumular la salida, el resto para S)

- Loops anidados: B(R) + ⌈ B(R) / (NB-2) ⌉ * B(S)
- Único loop: B(R) + n(R) * Costo_Index_Scan(S)
- Junta Hash: 3 * (B(R) + B(S))

Pipelining

- Salvo consultas simples, tendremos varios operadores en el árbol de consultas
- El resultado de un operador será la entrada de otro operador
- En vez de procesar completamente un operador y luego el siguiente, puede irse procesando parcialmente el resultado a medida que se arma
- Esto se denomina Pipelining
  - Puede requerir mayor uso de memoria
  - Puede disminuir el costo

Pipelining - Ejemplos

Si hacemos una selección podemos hacerlo después de una junta sin costo extra. (junta —> selección)

Cuando hay que evitar los duplicados (usar "distinct") tenemos que guardar en memoria los valores que vamos teniendo para chequear efectivamente que no se dupliquen. Esto incopora un costo extra. Si no me entran en memoria todos los registros tengo que hacer una copia a disco y hacer un sort, etc.

- Una proyección aplicada al resultado de una junta puede procesarse a medida que se genera el resultado de la junta
  - Se evita leer B(R) para la proyección
  - El costo es únicamente el de la junta (siempre que no tenga que evitar repetidos)
- Una junta con loops anidados o loop simple aplicada al resultado de una selección puede ir ejecutándose para cada bloque que devuelve la selección (selección —> junta)
  - Se evita re-leer B(R)
  - El costo es menor
    - Loops anidados ⌈ B(R) / (NB-2) ⌉ * B(S)
    - Loop simple: n(R) * Costo_Index_Scan

Estimación de cardinalidad

Estimación de cardinalidad

- Para poder usar las fórmulas de operadores que trabajan con resultados de otros operadores, precisamos conocer la información que nos dan los metadatos
- Precisamos estimar la cantidad de filas n(R)
- Algunas operaciones modifican F(R)
  - En base a esto, recalcular B(R)

Estimación de cardinalidad

- ¿Cuántas tuplas se devuelven por una operación?
- Selección simple: $n(\sigma_{Ai=v}(R)) = n(R) / V(Ai,R)$
- Proyección: $n(\Pi_{Ai\ ,\ An}(R)) = n(R)$
  - Salvo que tengamos información de valores de los atributos (ej: V(Ai,R) … V(An,R))
- Join: $n(R \bowtie_{R.A=S.A} S) = \frac{n(R) * n(S)}{\text{máx}(V(A,R)\ ,\ V(A,S))}$
  - Asume distribución equitativa y que los valores de la tabla de menor variabilidad están también en la otra tabla

Histogramas

Guarda los valores más frecuentes

- Guardan la frecuencia de ciertos valores
  - Podría ser todos, pero ocupa más
- Las fórmulas se pueden adaptar para estimar con este conocimiento
  - Para la selección, si se conoce la cantidad exacta del valor buscado, se la usa
  - Para el Join, hay que adaptar la fórmula
- PostgreSql tiene la vista pg_stats con información
  - select * from pg_stats WHERE tablename = 'player'

Histogramas - Ejemplo

A	100	200	300	400	Otros
R.A	50	100	50		100
S.A	100		200	150	550

- Se guardan la cantidad de filas para los 3 valores más frecuentes para R y S
  - Pueden no coincidir cuales son los más frecuentes!
- Además se debe almacenar cuantos valores tiene el atributo en cada tabla:
  - V(A,R) = 7
  - V(A,S) = 14
- Del histograma surgen la cantidad de filas
  - n(R) = 300
  - n(S) = 1000

Histogramas - Ejemplo

A	100	200	300	400	Otros
R.A	50	100	50		100
S.A	100		200	150	550

- Para selección en R
  - Puedo tener el valor certero
    - Si la condición es A = 100, se devuelven 50 filas para R
  - Si no, mejor estimación
    - Si la condición es A = 500, hacer 100 / 4
    - V(A,R) = 7, pero ya se la cantidad de 3 valores entonces el "Otros" representa los otros 4 valores

Histogramas - Ejemplo

A	100	200	300	400	Otros
R.A	50	100	50		100
S.A	100		200	150	550

- Para join, un poco más complejo
- Sin tener en cuenta histograma:
  - Cantidad: $\frac{300 * 1000}{máx(7,14)} = 21{,}429$

Histogramas - Ejemplo

A	100	200	300	400	Otros
R.A	50	100	50		100
S.A	100		200	150	550

- Teniendo en cuenta el histograma
  - Para el valor 100 se combinan 50 * 100
  - Para el 300 se combinan 50 * 200
  - El resto de los valores, no sabemos con exactitud
    - Ojo! El valor de R.A = 200 y el de S.A = 400 pueden combinarse
      - Que no se sepa su cantidad exacta, no implica que no estén incluidos en “Otros”
      - Se estiman asumiendo que están en la tabla y tienen distribución equitativa

Histogramas - Ejemplo

A	100	200	300	400	Otros
R.A	50	100	50		100
S.A	100		200	150	550

- Para R.A = 400
  - 100 registros comparten los 4 “Otros” valores
    - Conocemos 3 de los 7 valores posibles
  - Se asumen 25 con valor 400, 75 con Otros
- Para R.S = 200
  - 550 registros comparten los 11 “Otros” valores
  - Se asumen 50 con valor 200 y 500 para otros

A	100	200	300	400	Otros
R.A	50	100	50	**25**	75
S.A	100	50	200	150	500

Histogramas - Ejemplo

A	100	200	300	400	Otros
R.A	50	100	50	25	75
S.A	100	50	200	150	500

- Se usan los nuevos valores para estimar
  - Para “Otros”, no considerar los 4 valores conocidos en V(A,R) y V(A,S)
- Cantidad: 50 * 100 + 50 * 100 + 50* 200 + 25*150 + $\frac{75 * 500}{máx(3,10)}$
  - 5,000 + 5,000 + 10,000 + 3,750 + 3,750
  - Cantidad del join: 27,500

Estimación de bloques

- Algunas operaciones modifican F(R)
- Para Relaciones intermedias, recalcular la cantidad de bloques si se modifica F(R)
- Proyección:
  - Menos atributos ocupan menos tamaño
  - Aumenta F(R)
- Join:
  - La tupla del join tiene atributos de ambas relaciones, por lo que ocupa más
  - Disminuye F(R)

Estimación de bloques

- Ejemplo de proyección
  - Buscar el padrón de los alumnos del secundario “ILSE”
  - Bloques de 8192 bytes
    - Header de 192 bytes, 8000 bytes para registros
    - Registros de 80 bytes cada uno
      - Padrón y DNI : 4 bytes c/u
      - Nombre y secundaria tamaño promedio: 36 bytes cada uno
    - 100 registros por bloque
  - Si n(alumnos) = 100,000 y V(secundaria,alumnos) = 500, la selección traerá 100,000 / 500 = 200 filas
    - Si usáramos F(alumnos) serian 2 bloques
  - Pero al quedarse sólo con el padrón, entran 2000 padrones por bloque
  - Entonces las 200 filas ocupan 1 bloque

Estimación de bloques

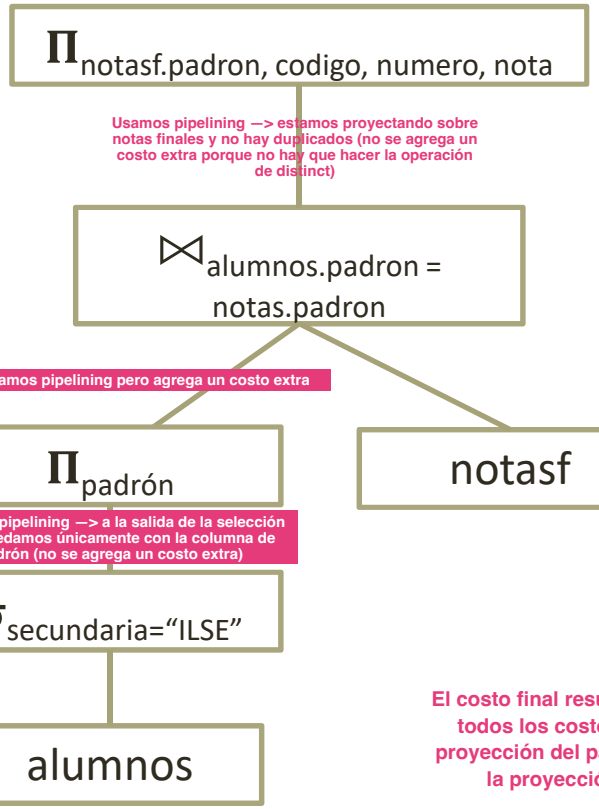
- Ejemplo de join
  - Datos de los alumnos con todas sus notas
    - 100,000 alumnos
    - 2,000,000 de notas, cada una de un alumno
    - El join devuelve 2,000,000 de tuplas
      - No hace falta usar la fórmula pero devolvería lo mismo
  - Cuántos registros entran por bloque?
    - 100 registros de alumno por bloque
    - 200 notas por bloque
    - $\frac{1}{F(alumnos \bowtie notas)} = \frac{1}{F(alumnos)} + \frac{1}{F(notas)}$
    - F($alumnos \bowtie notas$) = 66
    - B($alumnos \bowtie notas$) = 30,303

Ejemplo completo

Ejemplo completo

- Para la siguiente consulta:
  - ```
    SELECT nf.padron, nf.codigo, nf.numero
           , nf.nota
    FROM alumnos a, notasf nf
    WHERE a.padron = nf.padron
    AND a.secundaria = "ILSE"
    ```
  - Para las notas finales de los alumnos cuya secundaria es “ILSE”, mostrar su padrón, el código y número de la materia y la nota final
- Generar un árbol de consulta optimizado
- Estimar el costo de la operación completa
    ```

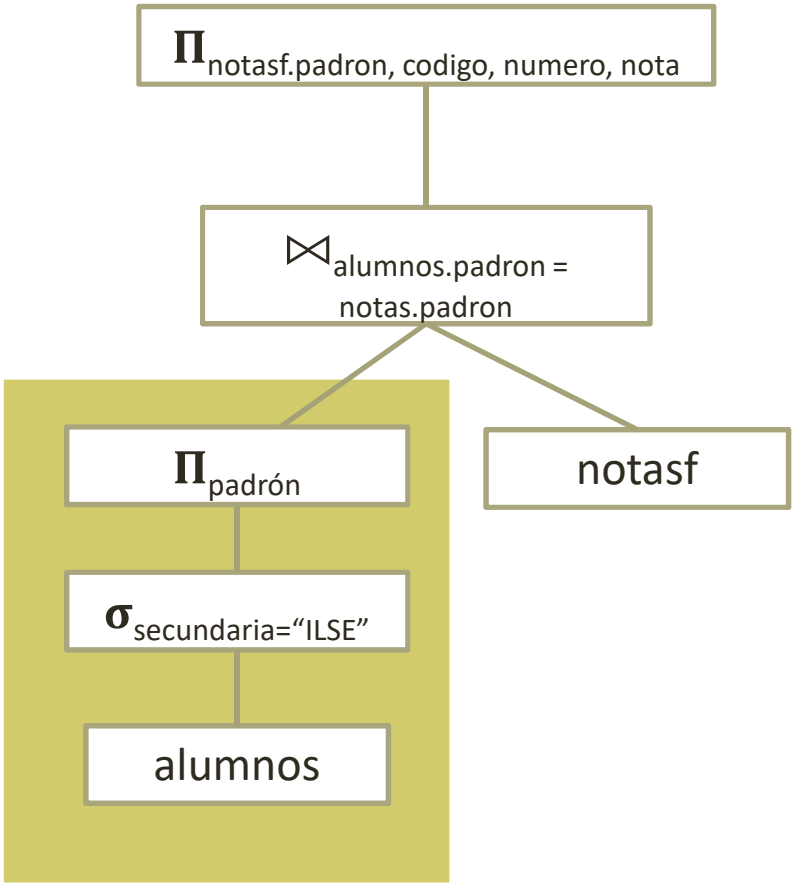
Árbol de la consulta



Ejemplo final - Datos

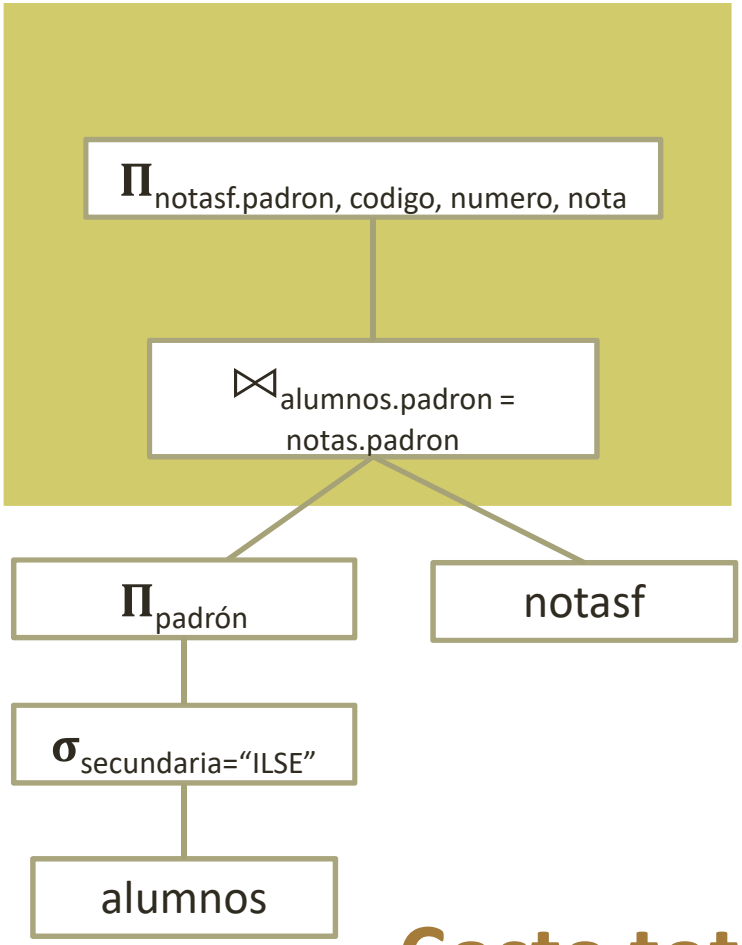
- alumnos(padron, nombre, secundaria, dni)
  - B(alumnos) = 1,000
  - n(alumnos) = 100,000
  - V(secundaria, alumnos) = 500
  - Índice I2 por secundaria Height(I2) = 3
  - Bloques de 8192 bytes
    - Header de 192 bytes, 8000 bytes para registros
    - Registros de 80 bytes cada uno
    - Padrón ocupa 4 bytes
- notasf(padron, codigo, numero, nota)
  - B(notasf) = 10,000
  - n(notasf) = 2,000,000
  - Índice I3 por padrón Height(I3) = 3
  - V(padron,notasf) = 100,000

Árbol de la consulta



- Proyección de padrón no agrega costo extra
  - Pipelining
- Costo de selección: 3 + 100,000 / 500
  - C1= 203
- Devuelve 200 alumnos
- Se proyecta su padrón
  - 800 bytes
  - La salida es 1 bloque

Árbol de la consulta



- Proyección final no agrega costo extra
  - Pipelining
- Ya se tiene en memoria el único bloque devuelto por la proyección de padrón
  - Pipelining
- Costo Join: 200 * (3 + 20) = 4600
  - Menor que otros costos
  - B(notasf) = 10,000

Costo total: 203 + 4600 = 4803

Ejercicio de final

Ejercicio de final

La UFA dispone de una base de datos con información sobre 10000 estrellas y 50000 observaciones realizadas por sus socios. La base de datos cuenta con las siguientes tablas:

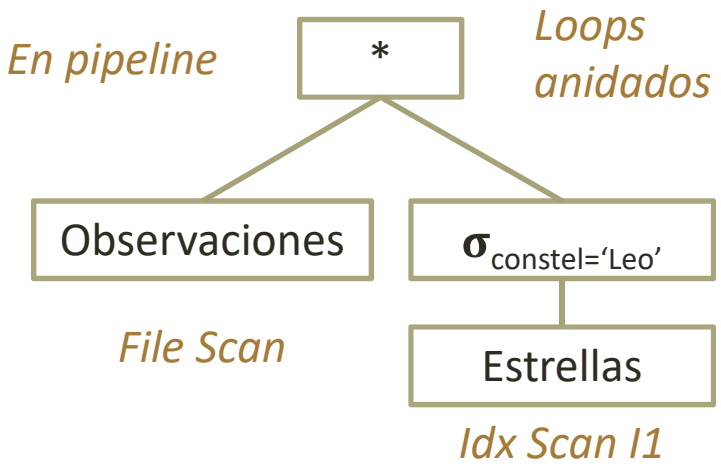
- Estrellas (nombre_estrella, constelación, declinación, ascensión)
- Observaciones (nro_socio, fecha, nombre_estrella, velocidad, intensidad)

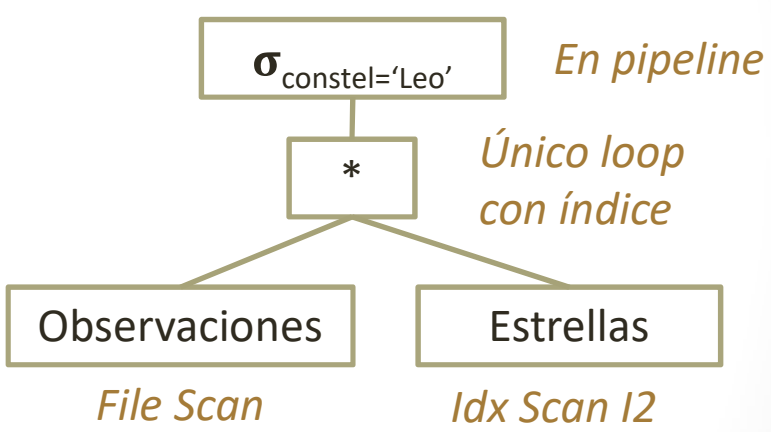
También se cuenta con los siguientes dos índices de tipo árbol:

- I1(constelación, Estrellas):Índice de clustering (agrupamiento) por el atributo constelación.
- I2(nombre_estrella, Estrellas): Índice secundario por el atributo clave nombre_estrella.

Ejercicio de final

Con el objetivo de encontrar todas las observaciones de estrellas que pertenecen a la constelación de 'Leo' , el SGBD construye los siguientes dos planes de ejecución:



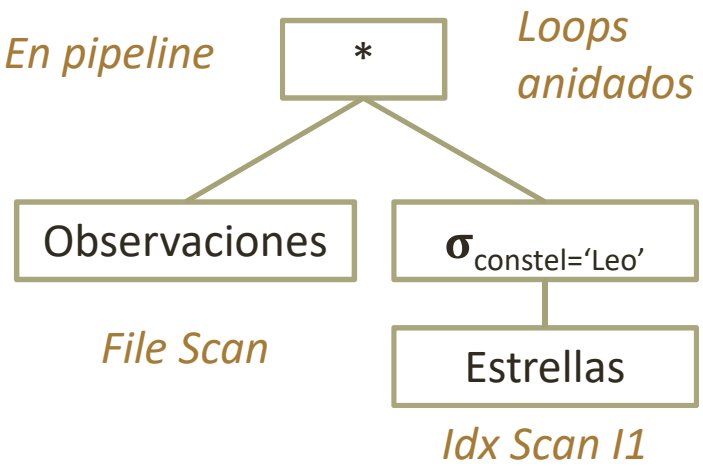


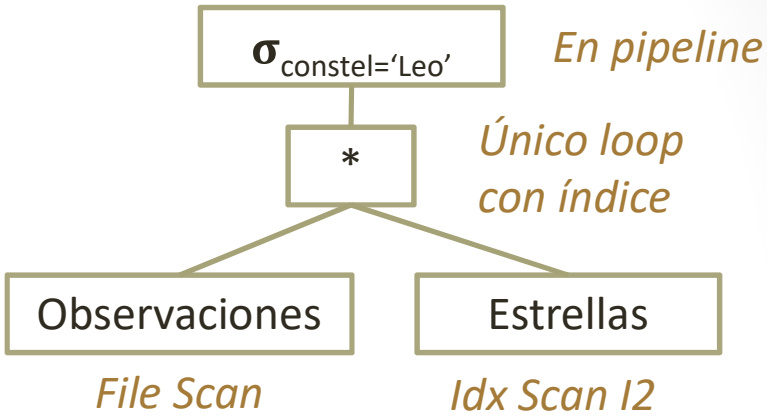
a) Estime el costo de cada plan en términos de cantidad de accesos a disco, y determine cuál de los dos es más conveniente en este sentido

b) Proponga la construcción de un índice adicional que permita planificar esta consulta de manera de utilizar dos índices, y grafique el plan de ejecución correspondiente.

Poner un índice para el nombre de las observaciones (tabla observaciones porque la tabla de estrellas ya tiene un índice)

Ejercicio de final





ESTRELLAS	OBSERVACIONES
n(Estrellas) = 10,000	n(Observaciones) = 50,000
B(Estrellas) = 1,000	B(Observaciones) = 5,000
V(Constelacion, Estrellas) = 10	V(nombre_estrella, Observaciones) = 10,000
H(I1(constelación) = 1 *(CLUST)*	OJO: el índice de clustering es un índice de ordenamiento; una misma tabla no puede tener 2 índices de clustering porque no puede estar ordenada por dos variables diferentes.
H(I2(nombre_estrella) = 4	