COSTO DE CADA OPERADOR

Para esta selección, con cuales bloques debería quedarme?

Pero tendríamos que leer todo, pasar por todos los registros para saber si coincide el bloque o no. Esto se conoce como un *file scan,* que es básicamente leer toda la tabla para saber si tengo que actualizar algo o no.

Para mejorar esto, podemos usar ***índices.***

Indices de árbol B+

Hace que la info este en los nodos hojas nada mas, todos los otros nodos son claves.

Costo de búsqueda por igualdad

- Altura de índice
- Acceso a los datos de la/s fila/s devueltas
  - Estimación de cantidad de filas
    - n(R) / V(Ai,R)
  - Si el archivo está ordenado por la clave de búsqueda, es menos de 1 acceso por fila
    - Índice de clustering
    - n(R) / (V(Ai,R) * F(R)) = B(R) / V(Ai,R)

Resumen de selección

$\sigma_{condición}(R)$

- Siempre se puede utilizar el método de file scan
  - Evaluar la condición en los datos
- Si alguna condición es de igualdad o rango, y está como AND a nivel principal, puede convenir aprovechar el índice y evaluar las otras condiciones al acceder a los datos
  - Puede convenir usar varios índices! (más complejo y raro)

Ejercicios guía

2. *(Uso de índices)* Dado el siguiente esquema de base de datos que guarda información de los clientes morosos de una empresa y de los abogados que llevan adelante los casos:
   - `Clientes(cod_cliente, nombre, domicilio, deuda, matricula_abogado)`
   - `Abogados(matricula, nombre, telefono)`

   encontramos que la siguiente consulta SQL se realiza de forma muy frecuente:

```
SELECT a.nombre, a.telefono, c.nombre, c.deuda
FROM Abogados a, Clientes c
WHERE a.matricula = c.matricula_abogado
AND c.cod_cliente = x;
```

   en donde x es un valor variable entre consultas. ¿Cuál de las siguientes alternativas de creación de índices propondría con el objetivo de agilizar las consultas? Justifique su respuesta.
   - Un índice por `Abogados.matricula` y otro por `Clientes.matricula_abogado`
   - Un índice compuesto por `(Abogados.nombre, Clientes.nombre)`
   - Un índice por `Abogados.matricula` y otro por `Clientes.cod_cliente`
   - Un índice compuesto por `(Cliente.cod_cliente, Cliente.matricula_abogado)`
   - Un índice por `Abogados.nombre` y otro por `Clientes.nombre`

Tenemos un join entre abogados y clientes y una condición fija sobre cliente. Vamos a ordenar las respuestas de la mas conveniente a la menos

1. Abogados.matricula y clientes.cod_cliente: Nos ayuda en ambas condiciones, en el join (el abogados.matricula) y en la selección estática también tendré un índice.

2. (clientes.cod_cliente, cliente.matricula_abogado): La selección estática se agiliza, y la condición de join también

3. Abogados.matricula y clientes.matricula_abogado: Para el join solo vamos a usar uno.

4. Abogados.nombre y clientes.nombre: No nos ayuda en nada tener un índice en una proyeccion porque no me sirve para nada cuando tengo que hace el join en el where´

5. (Abogados.nombre, clientes.nombre)

3. *(Costo de la junta con índice)* Dados los siguientes esquemas de relación sobre los clientes de una empresa y las facturas de las ventas realizadas:
   - `Clientes(cod_cliente, nombre, apellido)`
   - `Facturas(numero_factura, cod_cliente, fecha_factura, monto)`

   se propone construir un índice de árbol por código de cliente *(cod_cliente)* en la tabla *Facturas* para mejorar el costo de resolución de la junta natural en términos de accesos a disco.
   a) Calcule el nuevo costo de la junta natural suponiendo que el índice a construir fuera de agrupamiento *(clustering)*.
   b) Calcule el nuevo costo de la junta natural en caso que el índice a construir no fuera de agrupamiento.

   Asuma en ambos casos que un índice de árbol tendría una altura de 3, y considere la siguiente información de catálogo. Muestre todos los cálculos involucrados.

CLIENTES	FACTURAS
n(Clientes) = 1500	n(Facturas)=45000
B(Clientes) = 150	B(Facturas)=9000
	V(cod_cliente, Facturas) = 1500

Podrían no haberme dado el dato de V(cod_clientes, Facturas) pero la variabilidad se sacaba fácil porque sabemos que cod_clientes es clave foránea de Clientes, y que además es clave primaria ahí y que tenemos 1500 filas (únicas) en clientes, así que esta implícito en n(Clientes) por ser clave foránea en Facturas y clave primaria de Clientes

a.

El índice se encuentra en la tabla de Facturas. Como tenemos la tabla indexada, aprovechamos esta situación y realizamos el cálculo de costo por ***método de único loop.***

Con un índice de clustering, la formula de calculo de costo, es la siguiente:

$$cost(R * S) = B(S) + n(S) \cdot Height\big(I(A,R)\big) + \left\lfloor \frac{n(R)}{V(A,R)\ .\ F(R)} \right\rfloor$$

La tabla indexada (R) en este caso es Facturas:

$$cost(Fac * Cli) = B(Cli) + n(Cli) \cdot (Height\big(I(A,Fac)\big) + \left\lfloor \frac{n(Fac)}{V(A,Fac)\ .\ F(Fac)} \right\rfloor)$$

Para calcular F(Facturas) podemos hacer: F(R) = n(R)/B(R) = 45.000/9000

$$cost(Fac * Cli) = 150 + 1500 \cdot \left(3 + \left\lfloor \frac{45.000\ .\ 9.000}{1500\ .\ 45.000} \right\rfloor\right) = \mathbf{13.650}$$

b. Ahora, sin usar indice de agrupamiento, calculamos el costo con un índice secundario. No puede ser primario porque cod_clientes no es clave primaria:

$$cost(Fac * Cli) = B(Cli) + n(Cli) \cdot Height\big(I(A,Fac)\big) + \left\lfloor \frac{n(Fac)}{V(A,Fac)} \right\rfloor = 150 + 1500 * 3 + \frac{45.000}{1500} = \mathbf{49.650}$$

Es mucho menos costoso aplicar el índice de clustering en este caso. En el primero tuve 6 accesos a bloques (9000/1500) y en el otro tengo 30, eso ya me aumenta mucho el costo.

7. *(Junta hash GRACE)* La red social *Barcando* conecta a personas que frecuentan bares. Para ello almacena las siguientes relaciones:

- `Visitas(nombre_usuario, nombre_bar, cantidad)`
- `Usuarios(nombre_usuario, localidad, fecha_alta)`
- `Bares(nombre_bar, dirección, ciudad, teléfono)`
- `Amistades(nombre_usuario_1, nombre_usuario_2)`

Se desea invitar a conectarse a aquellas personas que hayan visitado bares en común. Como primer paso, se calcula la siguiente junta por igual con el método de junta *hash GRACE*:

$$Encuentros \leftarrow Visitas \bowtie_{Visitas1.nombre\_bar=Visitas2.nombre\_bar} Visitas$$

Para la tabla de *Visitas* poseemos las siguientes estadísticas:

VISITAS
n(Visitas) = 5.000.000
B(Visitas) = 200.000
V(nombre_usuario, Visitas) = 500.000
V(nombre_bar, Visitas) = 50.000

a) Estime la cardinalidad del resultado de esta junta en términos de cantidad de tuplas.
b) Estime el costo de la operación en términos de cantidad de accesos a disco.
c) Si elijo $m = 100$ particiones para la función de *hash*, ¿cuántos bloques de espacio en memoria debería tener disponibles para poder realizar la operación de junta?
d) Si a continuación se quisiera hacer la proyección $\pi_{nombre\_usuario1,nombre\_usuario2}(Encuentros)$, ¿cree que la misma podría hacerse en *pipeline* (es decir, procesando una a una las tuplas a la salida de la junta) ó sería necesario materializar el resultado? Justifique su respuesta.

Nota: Asumimos que el esquema de la relación resultante de la junta es `Encuentros(nombre_usuario1, nombre_bar1, cantidad1, nombre_usuario2, nombre_bar2, cantidad2)`.

a. ***Estimación de cardinalidad***

b. ***Estimación de costo – método hash Grace de junta***

Elijo N tal que pueda hashear y luego cargar 1 particion a la vez

En este caso, como estoy haciendo una junta consigo misma, en vez de tener que hacer la lectura de dos tablas, tengo la lectura de una única tabla, entonces para el hasheo tengo solo 2B(R) en vez de 2B(R) + 2B(S).

Luego el costo de procesamiento continua con leer cada una de la tablas una vez mas, pero como solo tengo una tabla, tengo una sola lectura mas, y finalmente el costo resulta

Cost(Visitas * Visitas) = 3 B(V) = 3 * 5.000.000

c. Por un lado necesito que M >= N+1

Necesitaría 101. N= m, no se porque

PIPELINING

Pipelining - S⋈σ(R)

- Una junta aplicada al resultado de una selección puede ir ejecutándose para cada bloque que devuelve la selección
  - Se evita grabar resultado intermedio
  - Se evita re-leer B(σ(R)) en el join
    - Loops anidados ⌈ B(σ(R)) / (M-2) ⌉ * B(S)
    - Hash Grace: 2*B(σ(R)) + 3*B(S)
    - Loop simple: n(σ(R)) * Costo_Index_Scan

EJERCICIO DE FINAL

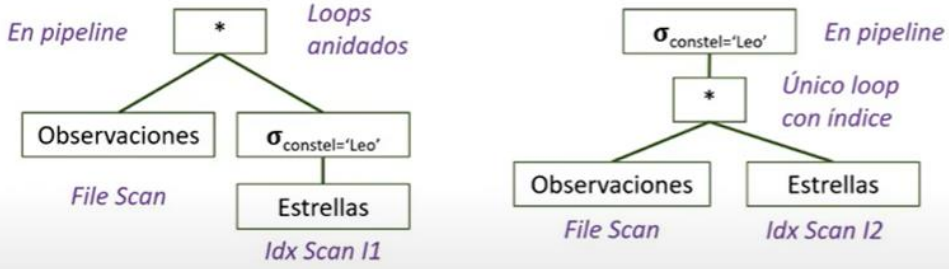
La UFA dispone de una base de datos con información sobre 10000 estrellas y 50000 observaciones realizadas por sus socios. La base de datos cuenta con las siguientes tablas:

- 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.

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.

a. Tenemos los siguientes datos:

OBSERVACIONES	ESTRELLAS
n(Observaciones) = 50,000	n(Estrellas) = 10,000
B(Observaciones) = 5,000	B(Estrellas) = 1,000
V(nombre_estrella, Observaciones) = 10,000	V(Constelacion, Estrellas) = 10
	H(I1(constelación) = 1 (CLUST)
	H(I2(nombre_estrella) = 4

Para el PLAN A primero analizamos el costo de la selección. Luego de eso hago la 2da etapa, del join.

El calculo de costo la selección se realiza teniendo en cuenta un ***índice de clustering***.

$$cost(S_5) = Height(I(A_i, R)) + \left\lceil \frac{n(R)}{V(A_i, R) \cdot F(R)} \right\rceil$$

Con F(Estrellas) = n(Es)/B(Es) = 10

Luego **Cost = 1 + (1.000/10) = 101**

Ahora tengo que hacer el join. Sin embargo, acá tengo un pipeline. Como antes del join vino una selección, se nos presenta el caso en que la junta recibe bloques en memo generados por otro operador.

Para estimar la cantidad de filas de la selección, vamos a usar su Variabilidad.

Pero ahora el n(Est) no es el mismo, porque realice la selección. **Si o si mantiene el factor de bloque, porque es selección.**

Para estimar el n(Est) usamos la variabilidad. Como tenemos V(Est) = 10, si el atributo se distribuye equitativamente, puedo asumir que tengo $n_{seleccion} = 10.000/10 = 1.000$

Con eso, voy a ocupar 100 bloques. $B_{selecc} = 1000/10 = 100$. **Es la misma proporción.**

Por otro lado, la F(Est) se mantiene siempre.

Ahora, Partimos al join con esos nuevos valores.

En este caso, utilizamos la formula del método elegido (en este caso loop anidado) pero nos ahorramos una lectura de tabla porque tenemos en memoria el bloque levantado, no lo bajamos a disco después de la selección.

Para calcularlo, debería utilizar la siguiente formula:

El costo del método es entonces: cost(R ∗ S) = min(B(R)+B(R)·B(S), B(S)+B(R)·B(S)), con S=Estrellas, como me ahorro una de las lecturas de S me queda en:

Cost_total(Obvs * Estrellas) = B(Estrellas) + B(Observaciones) * $B_{seleccion}$(Estrellas) - B(Estrellas) = **B(Observaciones) * $B_{seleccion}$(Estrellas) = 5.000 * 100 = 500.000**

Luego, sumo el costo de la selección inicial y resulta en:

Costo_total = 500.000 + 101 = 500.101

Para el PLAN B

Tenemos primero una junta por único loop con un índice compuesto por una clave primaria. La fórmula es la siguiente:

$$cost(R * S) = B(S)+n(S)\cdot(Height(I(A,R))+1)$$

El índice está en la tabla de Estrellas, así que los cálculos quedan:

$$cost(Est * Obvs) = 5.000 + 50.000\cdot(4+1) = 255.000$$

Luego, tenemos un pipeline con una selección. Como recibe bloques en memoria generados por otro operador, genera costo 0

Por lo tanto tenemos un **Costo_total = 255.000**

<u>En este sentido, es mucho mas conveniente en tema de costos el plan B</u>

b. Podríamos poner un indice adicional en observaciones, teniendo en cuenta el primer plan. La primera etapa se mantiene, con el mismo indice, y en el join podría poder un indice, que tiene una altura aproximadamente de 4/5 como en estrellas.

Suponiendo un nuevo indice de altura H = 5, sobre el atributo de junta, tendríamos un indice primario y deberíamos aplicar una formula de único loop:

$$cost(R * S) = B(S)+n(S)\cdot(Height(I(A,R))+1)$$

osea:

$$cost(Est * Obvs) = 100 + 1.000\cdot(5+1) = 6100$$

luego el costo total sumando el costo de 101 de la primera selección resulta en: **costo total: 6201**