EJERCICIOS TEMA 3

Ejercicio 1

Considerar las relaciones siguientes:

r_1(A,B,C)
r_2(C,D,E)
r_3(E,F)

donde r_1 tiene 1000 tuplas, r_2 tiene 1500 tuplas y r_3 750. Se pide:

a) Estimar el tamaño de la reunión $r_1 \bowtie r_2 \bowtie r_3$

Al final, el tamaño de la relación será el mismo sea cual sea el orden, debido a la propiedad de asociatividad y conmutatividad. Pero habrá que realizar primero la reunión que devuelva el menor número de tuplas. Teniendo esto en cuenta se tiene:

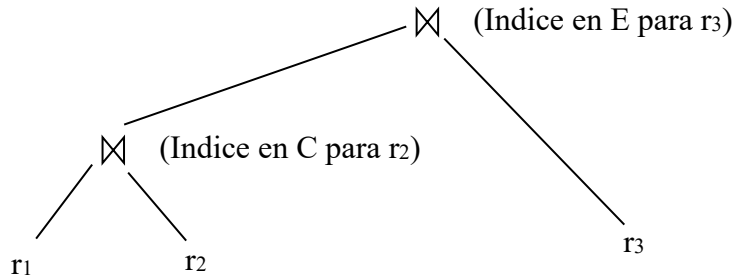
- Si se hace primero $r_1 \bowtie r_2$, como el campo de unión es C y es campo clave en la tabla r_2, entonces el número de tuplas es como máximo n_{r1} que son 1000.
- Si se hace primero $r_2 \bowtie r_3$, como el campo de unión es E y es campo clave en la tabla r_3, entonces el número de tuplas es como máximo n_{r2} que son 1500.

Entonces se haría primero la reunión $E_1=(r_1 \bowtie r_2)$ con 1000 tuplas.

Después $E_1 \bowtie r_3$, como E es clave primaria en r_3, como máximo una tupla de E_1 se unirá con una tupla de r_3, luego en total serán 1000 tuplas.

b) Diseñar una estrategia eficiente para el cálculo de la reunión.

Una estrategia eficiente sería crear un índice en el atributo C de la relación r_2 y en E para la relación r_3, con lo cuál el plan de evaluación sería:



c) Suponer que no hay claves primarias, y sean

V(C,r_1)=900
V(C,r_2)=1100
V(E,r_2)=50
V(E,r_3)=100

Estimar el tamaño de $r_1 \bowtie r_2 \bowtie r_3$ y diseñar una estrategia eficiente para calcular la reunión.

Para eso se calcula el número estimado de tuplas medio que surgen al realizar la reunión con cada tupla de la otra relación. Es necesario determinar la primera reunión:

- Si se hace primero $r_1 \bowtie r_2$, se aplica la expresión general, el número de tuplas devuelta será $n_{r1}*n_{r2}/\max\{V(C,r_1),V(C,r_2)\}$ = 1000*1500/max{900,1100}=1000*1500/1100=1000*15/11=1363.63 tuplas

- Si se hace primero $r_2 \bowtie r_3$, se aplica la expresión general, el número de tuplas devuelta será $n_{r2}*n_{r3}/\max\{V(E,r_2),V(E,r_3)\}$ = 1500*750/max{50,100}=1500*750/100=15*750=11250 tuplas

Luego se haría primero $E_1= r_1 \bowtie r_2$ con 1000*15/11 tuplas devueltas. El resultado se reúne con r_3, pero para ello hay que volver a estimar el V(E, E_1) que es el atributo de reunión con r_3. Para ello será V(E, E_1)=min{V(E,r_2),n_{E1}}=min{50,1000*15/11}=50.
Entonces el número de tuplas finales de E1 $\bowtie$ r3 será:
$n_{E1}*n_{r3}/\max\{V(E,E_1),V(E,r_3)\}$=(1000*15/11)*750/max{50,100}=(1000*15/11)*750/100=10227.27 tuplas

Buena estrategia es reunir r_1 con r_2 primero, ya que el tamaño intermedio está sobre el tamaño de r_1 o r_2. Se ha realizado la primera unión con las relaciones que tienen el mayor número de valores diferentes, es decir las que tienen menor selectividad para devolver menos tuplas.

Ejercicio 2

Suponer el siguiente esquema relacional de una facultad:

Asignaturas(<u>cod_as</u>,nom_as,creditos,cod_mat)
Mat_car(<u>cod_mat</u>,<u>cod_car</u>,creditos_min)
Carreras(<u>cod_car</u>,nom_car,año)

Además se tienen los siguientes datos:

	Asignaturas	**Mat_car**	**Carreras**
Tuplas	1300	600	54
Indices 1ºs	Cod_as 2 niveles		Cod_car 1 nivel
Indices 2ºs	Cod_mat 3 niveles		
Fr	50	60	30
Observaciones	400 Materias distintas Créditos: entre 6 y 30 Distribución uniforme	400 materias distintas 54 carreras distintas Distribución uniforme	A cada nombre de carrera Le corresponden 3 carreras diferentes

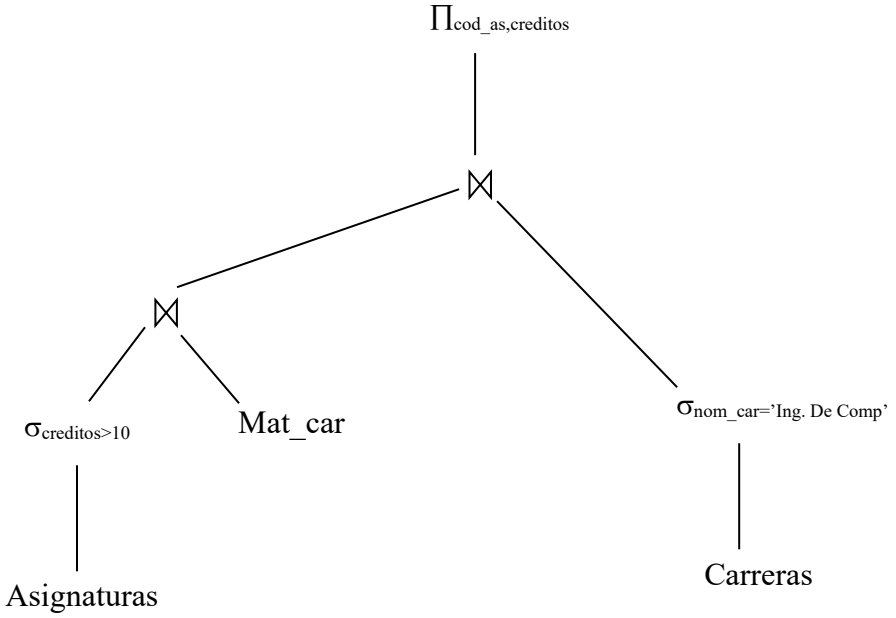
Se pide:

a) Obtener una consulta en el álgebra relacional para mostrar el cod_as, creditos donde se cumpla que los créditos > 10 y nom_car=”Ing. De Comp.”
b) Dar un plan lógico de la consulta, aplicando la heurística y calculando los tamaños intermedios.
c) Elegir una implementación para el primer join y calcular su coste estimado, sabiendo que cada tupla de carreras $\bowtie$mat_car tiene un fr=20 (factor de bloque)

a) Las operaciones involucradas serán la proyección, la reunión natural y la selección. Así que una posible consulta sería:

$$\Pi_{cod\_as,creditos}\left(\left((\sigma_{creditos>10}(Asignaturas) \bowtie Mat\_car \right) \bowtie (\sigma_{nom\_car='Ing.\ De\ Comp'}(carreras))\right)$$

b) La consulta expresada en forma de árbol se muestra a continuación:



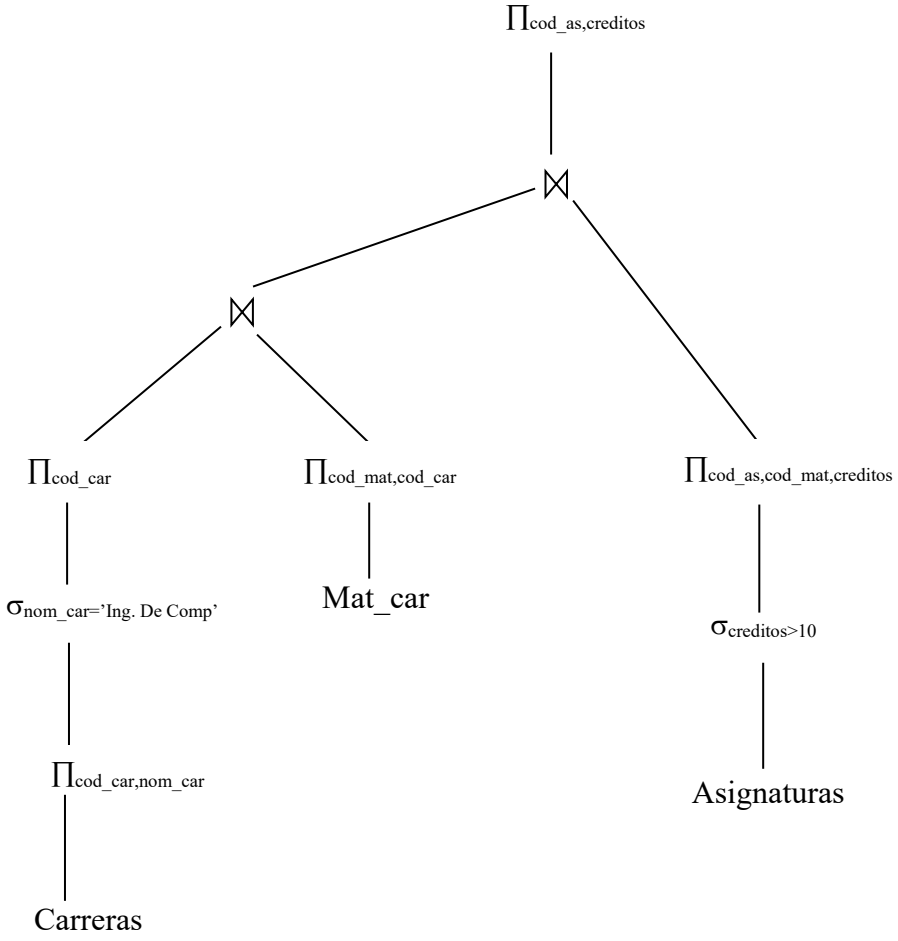
El tamaño estimado de cada operación en tuplas es el siguiente:

T(carreras)=54 tuplas
$T(\sigma_{nom\_car='Ing.\ De\ Comp'})$=3 tuplas
T(Mat_car)=600 tuplas
T(Asignaturas)=1300
$T(\sigma_{creditos>10})$=1300*1/25*(30-10)=1300*20/25=1040 tuplas
La nueva estadística a la salida de la selección para cod_mat será:
$\min\{V(cod\_mat,asignaturas),n(\sigma_{creditos>10})\}$=min(400,1040)=400
T(1ª reunión)=1040*600/max{400,400}=1560 tuplas
A la entrada de la segunda reunión, la estadística de cod_car de la rama de la izquierda se mantiene ya que debería de ser min{V(cod_car,mat_car),numero_tuplas primera reunión)=min{54,1560}=54
T(2ª reunión)=1560*3/max{3,54}=86.66 →87 tuplas

Para obtener el plan aplicando la heurística:

- Hacer las selecciones cuanto antes y primero las más restrictivas.
- Proyecciones cuanto antes.
- Hacer primero las reuniones que produzcan un menor número de tuplas.
- Hacer proyecciones para reducir tamaños intermedios de las tuplas, aunque no vengan en la consulta original.

Luego a la vista de los tamaños estimados de tuplas para cada operación, y aplicando la heurística se obtiene:



Los tamaños son los siguientes si se intercambia el orden de los joins, ya que la selección de nom_car va a devolver muy pocoas tuplas.

T(carreras)=54 tuplas
$T(\sigma_{nom\_car='Ing.\ De\ Comp'})$=3 tuplas, ya que cada nombre de carrera le corresponden 3 carreras diferentes, luego V(nom_car,carreras)=54/3=18.
T(mat_Car)=600
T(1ª reunión)=3*600/max{3,54}=33.3333 →34
T(asignaturas)=1300
$T(\sigma_{creditos>10})$=1300/25*(30-10)=1040
A la entrada de la segunda reunión el atributo cod_mat que viene de la selección de créditos, la nueva estadística será min{400,1040)=min{400,1040}=400
T(2ª reunión)=34*1040/max{400,400}=88.4

c) La primera reunión $\sigma_{nom\_car='Ing.\ De\ Comp'}$ ⋈ Mat_car se puede implementar con un lazo anidado sin índice, ya que Mat_car no tiene índices disponibles y el índices de carreras se pierde después de la selección. Para el peor caso de que la memoria sólo tiene 3 bloques, el coste asociado será:

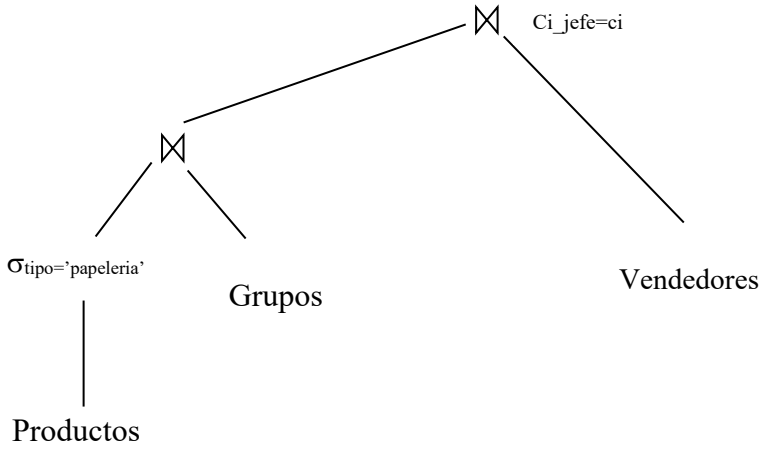
Coste = $b_r + b_r * b_s$ = 1+1*10 = 11 bloques
$b_r = \lceil 3/50 \rceil = 1$
$b_s = \lceil 600/60 \rceil = 10$
Si hubiese que grabar en disco esta operación, el coste sería: $11+\lceil 34/20 \rceil = 13$ bloques.

Ejercicio 3

Dado el siguientes esquema relacional

Productos(tipo,<u>nroprod</u>,nrogrupo)
Grupos(<u>nrogrupo</u>,ci_jefe)
Vendedores(<u>ci</u>,nrofrupo,salario)

Y el siguiente plan lógico de consulta:



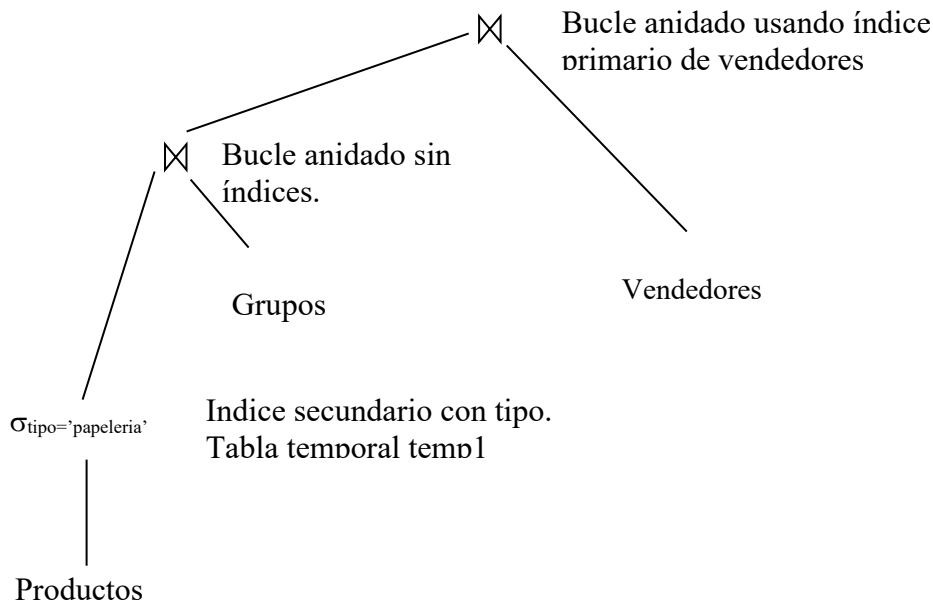
Y considerando que el resultado de la selección se debe de guardar en una tabla temporal temp1 y el resultado del primer join se debe de guardar en una tabla temporal temp2.

a) Dar el plan físico que le parezca mejor para ese plan lógico.
b) Calcular los tamaños de temp1 y temp2.
c) Calcular el coste total del plan.

Datos:

	Productos	**Grupos**	**Vendedores**	**Temp2**
tuplas	2000	50	10000	
Indice 1º			Ci Nivel 2	
Indice 2º	Tipo Nivel 2		Salario Nivel 3	
fr	20	10	5	5
Observaciones	200 tipos distintos			

a)



b)La selección : $\sigma_{tipo='papeleria'}$ tiene un tamaño de $n_{productos}/V(tipo,Productos)=2000/200=10$ tuplas que corresponderá a la tabla temp1

La primera reunión tendrá un tamaño esperado de 10 tuplas como máximo que el atributo de reunión es clave de la tabla grupos (No hay estadística para ese campo de la tabla Grupos). Corresponderá a la tabla temp2.

c) El coste total al ejecutar el plan será para el peor caso de que la memoria tenga 3 bloques:

Coste ($\sigma_{tipo='papeleria'}$)= Número de niveles del árbol del índice secundario + número registros a recuperar en bloques (desordenados) + coste de grabar la salida = $2 + 10 + \lceil 10/20 \rceil = 12 + 1$ (coste grabar salida) = 13 bloques, debido a la utilización del índice secundario, la recuperación de los 10 valores y el volcado del resultado a la tabla temp1.

Coste(temp1 ⋈ grupos)=$b_{temp1}+b_{temp1}*b_{grupos}$+coste grabar= $1 + 1*\lceil 50/10 \rceil + \lceil 10/5 \rceil = 1+5+2=8$, debido a la utilización de un bucle anidado por bloques.

Coste(temp2 ⋈ vendedores)= $n_r*C+b_r=10(2+1)+2=2+30=32$, debido a la utilización de un bucle anidado con índice. C es el coste de recuperar con el índice de 2 niveles un campo clave, luego C=2+1=3

Coste total= 13+8+32=53 bloques.

Ejercicio 4

a) Sentencia SQL:

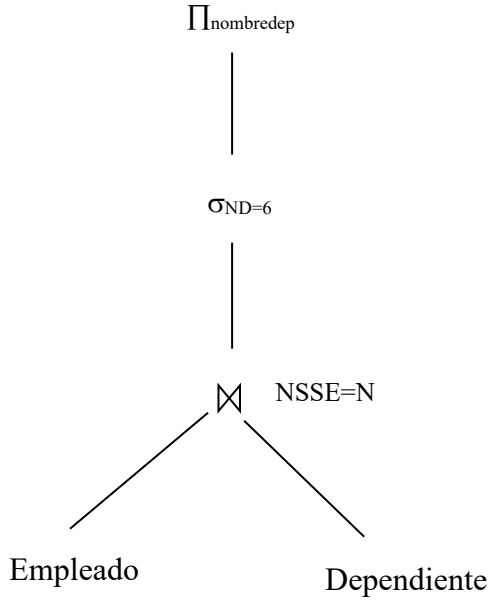
```
Select dependiente.nombredep
From dependiente
        inner join empleado
on dependiente.nsse=empleado.nss
```

where empleado.nd=6;

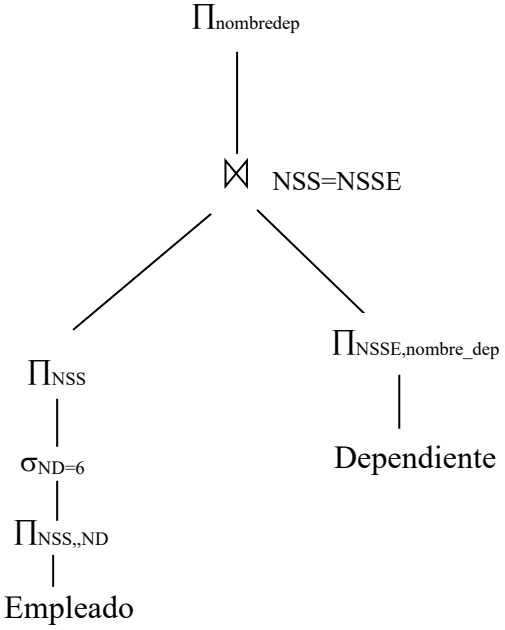
Se traslada al algebra relacional:

$$\Pi_{nombredep}\,(\ (\sigma_{ND=6}\,(empleado \bowtie_{nss=nsse} dependiente\ )$$

El árbol será el siguiente:



Optimizando, desplazando la selección e introduciendo proyecciones:



Se estiman el tamaño de cada operación en tuplas y bloques:

Empleado: Longitud del registro, L_R=50+1+100+4+8+50+1+4+4+1=223 bytes
Número de registros, n_R= 560.
Factor de bloques, f_R=⌊2048 / 223⌋ = 9 reg/bloque
Número de bloques, b_R=⌈560/9⌉ = 63 bloques

$\prod_{NSS,,ND}$, L_R=5 bytes , n_R=560 registros , f_R = 409 reg/bloque , b_R= 2 bloques
$\sigma_{ND=6}$, L_R=5 bytes , n_R=nempleado/V(ND,Empleado)=560/5 = 112 registros , f_R = 409 reg/bloque , b_R= 1 bloques
$\prod_{NSS}$, L_R=4 bytes , n_R= 112 registros , f_R = 512 reg/bloque , b_R= 1 bloques

Dependiente: Longitud del registro, L_R= 4+150+1+8+2=165 bytes
Número de registros, n_R= 2240.
Factor de bloques, f_R= $\lfloor 2048 / 165 \rfloor$ = 12 reg/bloque
Número de bloques, b_R= $\lceil 2240/12 \rceil$ = 187 bloques
$\prod_{NSSE,nombre\_dep}$ L_R=154 bytes , n_R=2240 registros , f_R = 13 reg/bloque , b_R= 173 bloques

Join NSS=NSSE , el número de registros será nr*ns/max{V(A,r),V(A,s)). Para eso hay que obtener la nueva estadística de la selección que para el campo NSS será min{V(NSS,Empleado),n$\sigma_{ND=6}$ }=min{560,112}=112
Entonces para el join será: 112*2240/max{112,560}=112*2240/560=448 tuplas.
L_R=4+150+4=158 bytes. f_R=12 registros/bloque , b_R=38 bloques a la salida del Join.

$\prod_{nombredep}$, L_R=150 bytes , n_R=448 registros , f_R = 13 reg/bloque , b_R= 35 bloques

Coste asociado a la operación de Join. Hay una memoria de 8 Kb , luego hay 1 bloque para una entrada, 1 bloque para la salida y 2 bloques para la otra entrada. Utilizamos el bucle anidado por bloques (siempre se puede usar) y tenemos 4 bloques de memoria de RAM, con coste: C= $\lceil br/(M-2) \rceil$ * bs +br . Tomando br como 1 ó 173 se obtiene:
br=1, C= $\lceil 1/2 \rceil$ * 173 +1=174
br=173, C= $\lceil 173/2 \rceil$ * 1 +173=260

Si se encauza una de las ramas, C= $\lceil br/(M-2) \rceil$ * bs.
br=1, C= $\lceil 1/2 \rceil$ * 173 =173
br=173, C= $\lceil 173/2 \rceil$ * 1 =87 , luego se encauza la rama de 173 bloques y se materializa el bloque de la otra rama.

Coste total = 63 (lectura empleado) +1 (coste materializar) + 187 (lectura dependiente) + 87 (coste del join) = 338 bloques, sin considerar que la salida se escribe en el disco.

Se puede ver que para este caso, hay un bloque de entrada en el Join que se puede mantener en memoria y por tanto el coste de la operación de Join sería cero. Por lo tanto, el coste total se reduciría a las lecturas de las dos relaciones: 63+187=250 bloques.

La consulta final será:

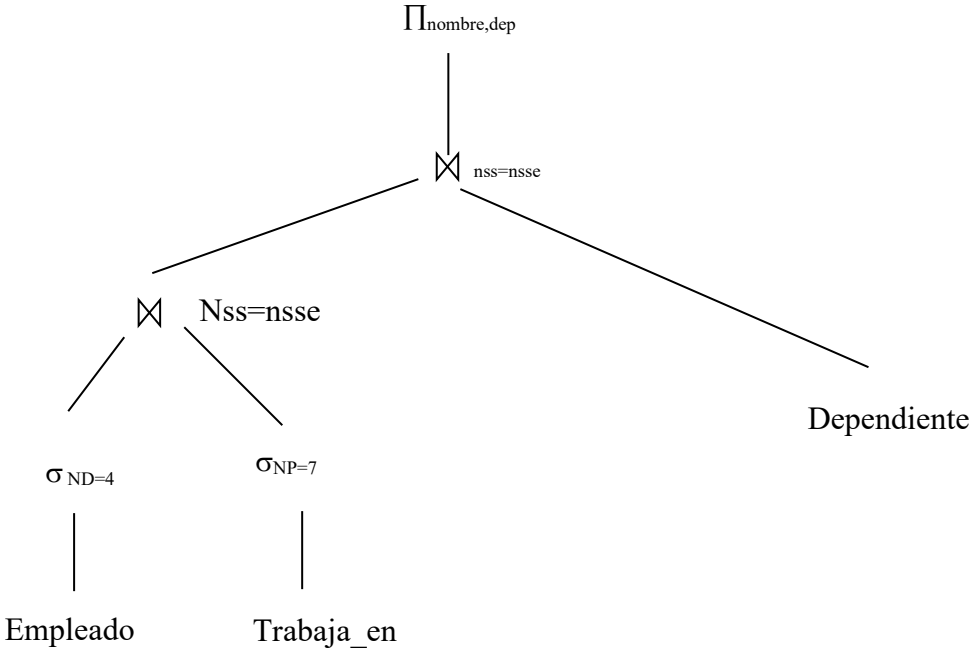
```
Select dependiente.nombredep
From
        Select NSS from
                Select NSS,ND from empleado
                Where nd=6
Inner join
        Select NSSE,Nombre_dep from
        Dependiente
On dependiente.NSSE=empleado.NSS;
```

b) La consulta es la siguiente:

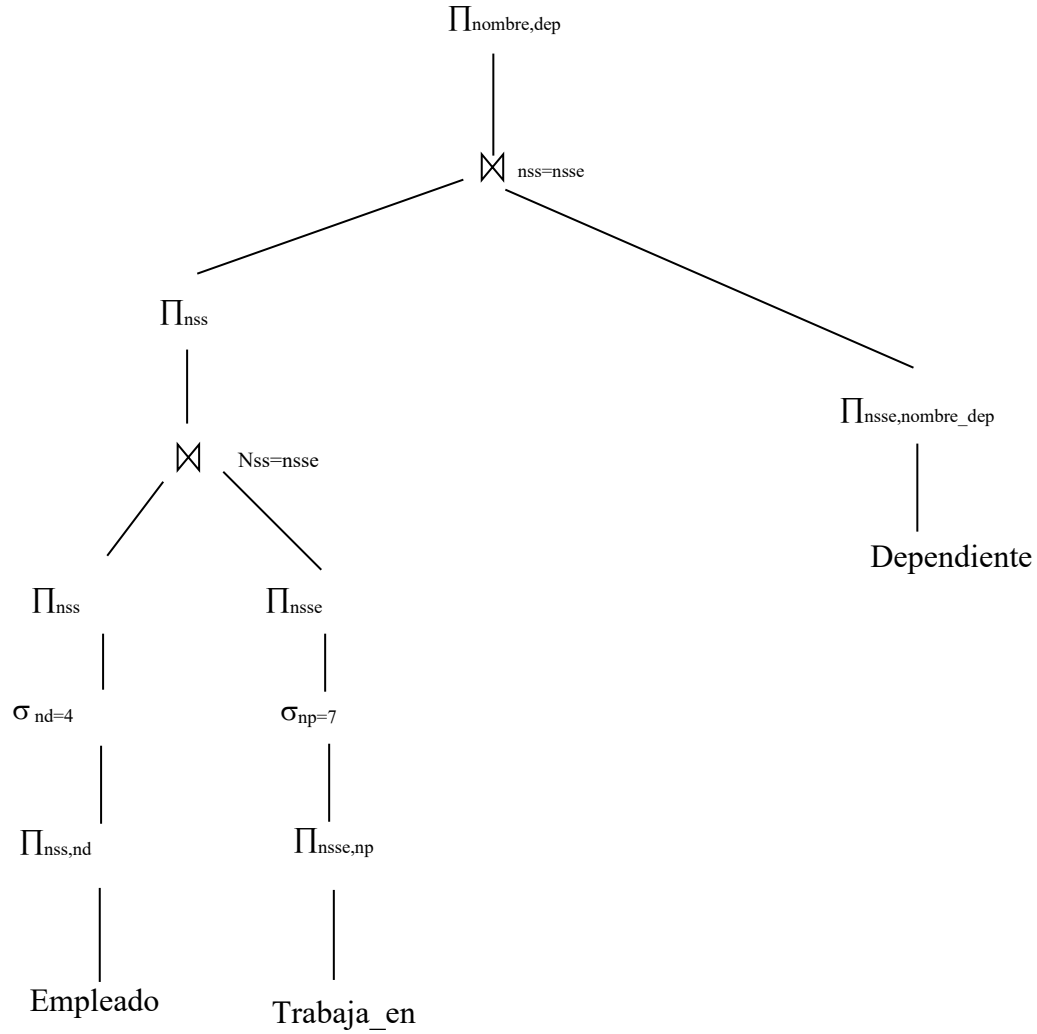
```
Select dependiente.nombredep
```

```
From dependiente
Inner join empleado
        Inner join trabaja_en
        On empleado.nss=trabaja_en.nsse
On dependiente.nsse=empleado.nss
Where empleado.nd=4 and trabaja_en.np=7;
```

El árbol de esta consulta sería:



El árbol optimizado sería el siguiente:



La rama empleado es la misma que la consulta a) por lo que proporcionaba 112 registros, 1 bloque al final de la última proyección y 63 bloques de lectura de la relación empleado. Notar que se ha realizado primero la reunión que tiene las dos entradas que proceden de las salidas de las selecciones.

Trabaja_en: Longitud del registro, L_R=4+4+4=12 bytes
Número de registros, n_R= 18500.
Factor de bloques, $f_R = \lfloor 2048 / 12 \rfloor = 170$ reg/bloque
Número de bloques, $b_R = \lceil 18500/170 \rceil = 109$ bloques

$\Pi_{NSSE,,NP}$, L_R=8 bytes , n_R=18500 registros , f_R = 256 reg/bloque , b_R= 73 bloques
$\sigma_{NP=7}$, Hay 3590 proyectos de la tabla proyectos. De promedio habrá nr/V(A,r)=18500/3590 = 5 personas por proyecto.
Π_{NSSE} , L_R=4 bytes , n_R= 5 registros , f_R = 512 reg/bloque , b_R= 1 bloques

Primer Join: Se aplica la fórmula general y se tiene nr*ns/max{V(A,r),V(A,s)}=112*5/max{112,5}=5 registros. A la salida de la selección ND=4, la nueva estadística vale min{560,112}=112 valores diferentes y para a selección de NP=7, sería el min{560,5}=5 valores diferentes.
L_R=8 bytes , n_R= 5 registros , f_R = 256 reg/bloque , b_R= 1 bloques
Π_{NSS} , L_R=4 bytes , n_R= 5 registros , f_R = 512 reg/bloque , b_R= 1 bloques

La rama dependiente es igual que la del apartado a) , donde había 2240 registros, 187 bloques de la tabla dependiente y 173 bloques de entrada al segundo join.

Segundo join: Para el segundo join se aplica la expresión general nr*ns/max{V(A,r),V(A,s)}. En la rama de la izquierda entran 5 tuplas (salida del primer join), luego su nueva estadística será el min{560,5}=5 valores diferentes y la de la rama de la derecha se mantiene el valor inicial de la estadística de 560 valores diferentes. Entonces el número de tuplas del segundo join serían: 5*2240/max{5,560}=5*2240/560=20 tuplas

L_R=158 bytes , n_R= 20 registros , f_R = 12 reg/bloque , b_R= 2 bloques

Π_{nombre_dep} , L_R=150 bytes , n_R= 20 registros , f_R = 13 reg/bloque , b_R= 2 bloques

Notar que el primer join que se realiza es aquél que produce un menor número de tuplas. En caso contrario habría que reordenar el árbol.
Para el primer join, el coste asociado no tiene ya que sólo se necesitan 3 bloques de memoria y como hay 4 no hay coste asociado.
Para el segundo join, ocurre lo mismo, cabe en memoria una de las entradas, luego el coste del join será cero.
En caso de no encauzar, la rama que habría que materializar sería la de la izquierda:

br=1, C=$\lceil 1/2 \rceil$ * 173 =173
br=173, C=$\lceil 173/2 \rceil$ * 1=87 , luego se materializa bs=1

Coste total = Coste de leer empleado + coste de leer trabaja en + coste de leer dependiente = 63+109+187=359 bloques, obviando la escritura de la salida de datos.

La consulta final será:

```
Select dependiente.nombre_dep
From
        Select nsse,nombre_dep from dependiente
        Inner join
                Select nss from
                        Select nss,nd from empleado
                        Where nd=4
                Inner join
                        Select nsse from
                                Select nsse,np from trabaja_en
                                Where np=7
                On empleado.nss=trabaja_en.nsse
        On dependiente.nsse=empleado.nss;
```

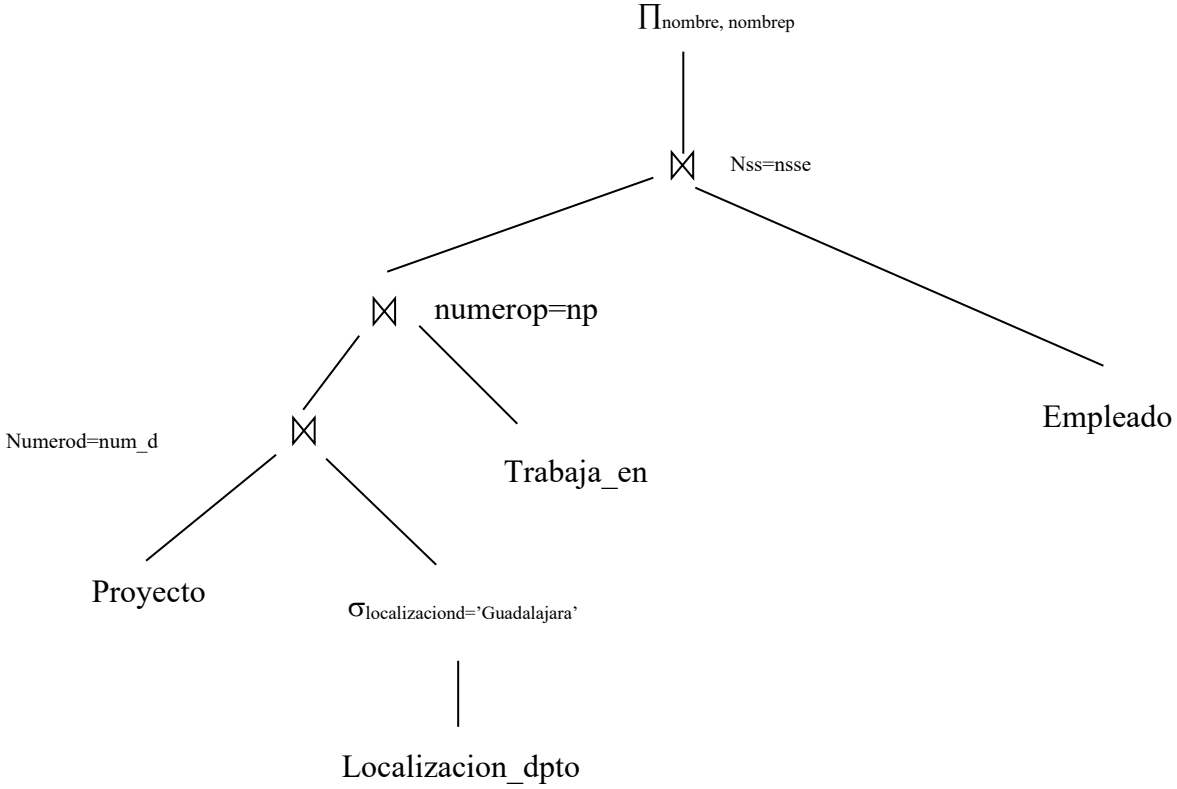
Notar que en este caso sólo hemos utilizado los índices como información estadística. En un caso general, se deberían de utilizar si existen y además el coste es menor.

c) La consulta es la siguiente

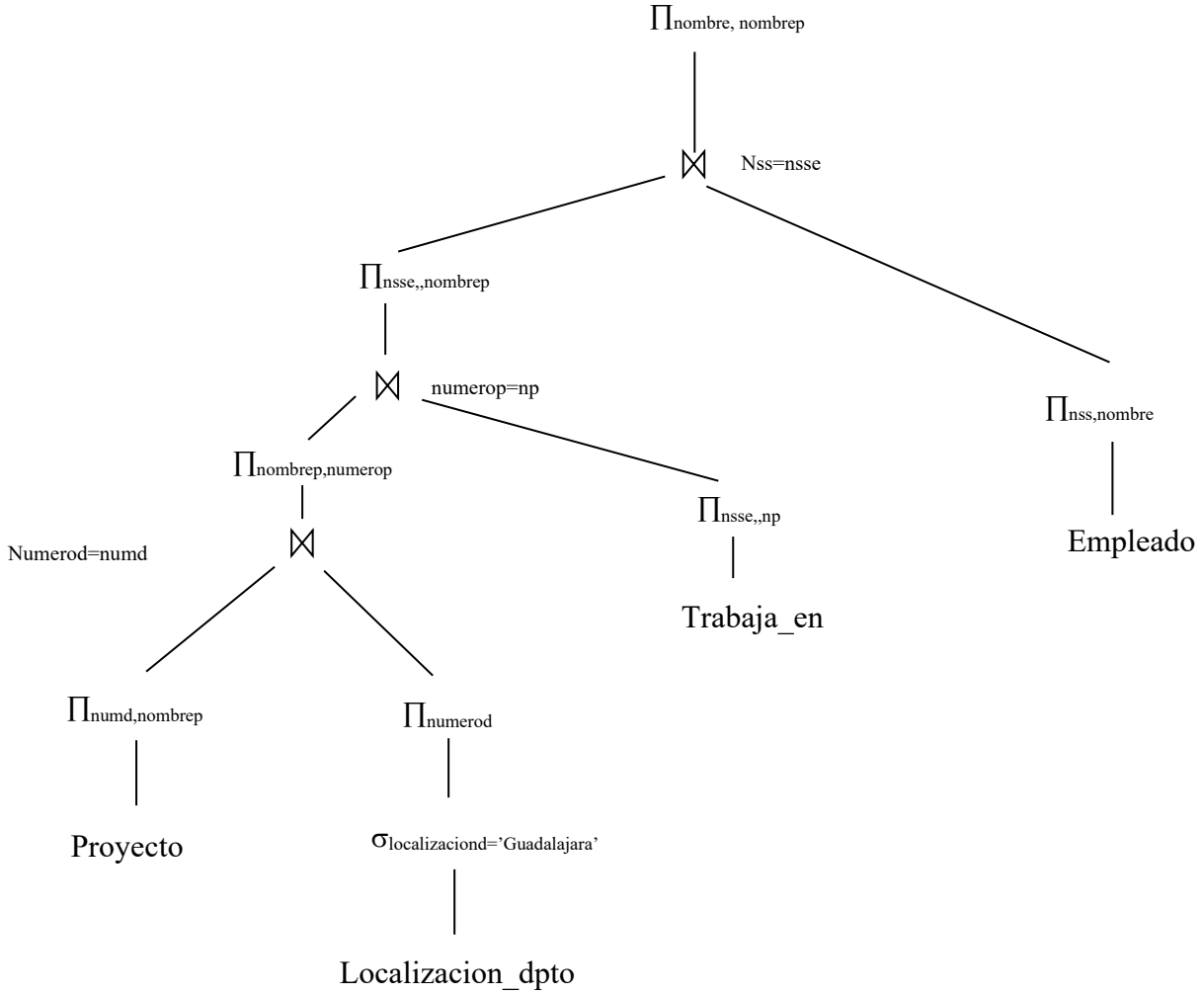
```
Select empleado.nombre,proyecto,nombrep
From empleado
        Inner join trabaja_en
        On empleado.nss=trabaja_en.nsse
Inner join
        Proyecto
        Inner join localizacion_dpto
        On proyecto.num_d=localizacion_dpto.numerod
On trabaja_en.np=proyecto.numerop
```

Where localizacion_dpto.localizaciond=’Guadalajara’;

Un primer árbol puede ser el siguiente:



Se introducen las proyecciones:



Se estima el número de tuplas de cada operación:

Tabla proyecto : 3590
$\Pi_{numd,nombrep}$: 3590
Tabla localización_dpto: 6 registros
$\sigma_{localizaciond='Guadalajara'}$: $n_r/V(A,r)=6/6=1$ registro
$\Pi_{numerop}$: 1 registro
Primer join: expresión general $n_r*n_s/max\{V(A,r),V(A,s)\}=3590*1/max\{6,1\}=3590/6=598.33=598=$ 598 tuplas, ya que para la rama de la izquierda se mantiene la estadística, y para la rama de la derecha $min\{6,1\}=1$
$\Pi_{nombrep,numerop}$: 598 tuplas
Tabla Trabaja_en 18500 registros
$\Pi_{nsse,,np}$: hay 18500 registros
Segundo join: aplicamos la expresión general $n_r*n_s/max\{V(A,r),V(A,s)\}=598*18500/max\{598,3590\}=3081.61=3082$ tuplas, ya que para la rama de la izquierda se la estadística es de $min\{3560,598\}$ y la rama de la derecha tiene 3590 proyectos diferentes.
$\Pi_{nsse,,nombrep}$: habrá 3082 registros
Tabla Empleado: 560 registros
$\Pi_{nss,nombre}$ habrá 560 registros

Tercer join: la expresión general es $n_r*n_s/\max\{V(A,r),V(A,s)\}=3082*560/\max\{560,560\}=3082$ tuplas, ya que la rama de la izquierda tiene una estadística de min{560,3082} y la rama de la derecha tiene 560 empleados diferentes.
$\prod_{nombre,\ nombrep}$ habrá 3082 registros.

Proyecto: L_R=258 bytes, n_R= 3590 , f_R= 7 reg/bloque , b_R= 513 bloques
$\prod_{numd,nombrep}$ L_R=108 bytes, n_R= 3590 , f_R= 18 reg/bloque , b_R= 200 bloques
Localización_dpto: L_R=154 bytes, n_R= 6 , f_R= 13 reg/bloque , b_R= 1 bloques
$\sigma_{localizaciond='Guadalajara'}$: igual que el anterior, n_R= 1 , f_R= 13 reg/bloque , b_R= 1 bloques
$\prod_{numerop}$: L_R=4 bytes, n_R= 1 , f_R= 512 reg/bloque , b_R= 1 bloques

Primer join:

br=1, C=⌈1/2⌉ * 200 =200
br=200, C=⌈200/2⌉ * 1=100 , luego se materializa bs=1, que equivale a la rama localizacion_dpto. Notar que cabe en RAM una de las entradas y por lo tanto el coste del join sería cero.

$\prod_{nombrep,numerop}$: L_R=104 bytes, n_R= 598 , f_R= 19 reg/bloque , b_R= 32 bloques

Tabla Trabaja_en, L_R=12 bytes, n_R= 18500 , f_R= 170 reg/bloque , b_R= 109 bloques
$\prod_{nsse,,np}$: L_R=8 bytes, n_R= 18500 , f_R= 256 reg/bloque , b_R= 73 bloques

Segundo join:

br=32, C=⌈32/2⌉ * 73 =1168
br=73, C=⌈73/2⌉ *32=1184 , se materializa la más pequeña y se encauza la más grande, luego esta es la mejor opción ya que tendría un coste de 1184 + 32 =1216, sin embargo la otra tendría 1241 bloques.

$\prod_{nsse,,nombrep}$: L_R=104 bytes, n_R= 3082 reg. , f_R= 19 reg/bloque , b_R= 163 bloques

Tabla Empleado: L_R=223 bytes, n_R= 560 , f_R= 9 reg/bloque , b_R= 63 bloques
$\prod_{nss,nombre}$ L_R=54 bytes, n_R= 560 , f_R= 37 reg/bloque , b_R= 16 bloques

Tercer join: 3082 registros a la salida

br=163, C=⌈163/2⌉ * 16 =1312 , Es la mejor opción. Se materializa la rama de 16 bloques (la más pequeña), se encauza la de 163 (la más grande)
br=16, C=⌈16/2⌉ *163=1304

$\prod_{nombre,\ nombrep}$ L_R=150 bytes, n_R= 3082 , f_R= 13 reg/bloque , b_R= 30 bloques

El coste total será: Lectura de la tabla proyecto + lectura de la tabla Localizacion_dpto + lectura de la tabla trabaja_en + materializar rama de la izquierda + coste segundo join + lectura de la tabla empleado + materializar la rama de la izquierda + coste tercer join = 513 + 1 + 109 + 32 + 1184 + 63 + 16 + 1312=3230 bloques

La consulta optimizada será:

```
Select empleado.nombre,proyecto.nombrep
From
    Select nss,nombre
    From empleado
```

```
Inner join
        Select nsse,np
        From trabaja_en
        Inner join
                Select nombrep,numerop
                From
                        Select numd,nombrep,numerop
                        From proyecto
                        Inner join
                                Select numerod
                                From localizacion_dpto
                                Where localizaciond=’Guadalajara’
                        On proyecto.numd=localizacion_dpto.numerod
        On trabaja_en.np=proyecto.numerop
On empleado.nss=trabaja_en.nsse;
```

Notar que en este caso sólo hemos utilizado los índices como información estadística. En un caso general, se deberían de utilizar si existen y además el coste es menor.

Ejercicio 5

Sea el siguiente esquema relacional representando sucursales de un banco, cuentas y clientes. Se sabe que el banco cuenta con una sola sucursal por ciudad, teniendo sucursales en las 19 capitales del país:

Sucursales(ciudad,nombre_sucursal, fecha_inauguración, dirección, teléfono)
Cuentas(numero_cuenta, ci_titular, ci_asociado, nombre_sucursal, tipo_cta, saldo)
Clientes(ci,nombre,domicilio,teléfono)

Se tiene la siguiente información sobre las relaciones:

Sucursales:
- Ciudad: 40 bytes
- Nombre_sucursal: 40 bytes
- Fecha_inauguración: 8 bytes
- Dirección: 50 bytes
- Teléfono: 4 bytes
- Hay 40.000 registros.

Cuentas:
- Numero_cuenta: 12 bytes
- Ci_titular: 4 bytes
- Ci_asociado: 4 bytes
- Nombre_sucursal: 50 bytes
- Tipo_cta: 2 bytes
- Saldo: 4 bytes
- Los saldos se encuentran uniformemente distribuidos en el rango 1 – 1.000.000
- Hay 100.000 registros.

Clientes:
- Ci: 4 bytes
- Nombre: 50 bytes
- Domicilio: 50 bytes
- Teléfono: 4 bytes
- Hay 100.000 registros

El tamaño de bloque del disco es de 2 KB y la memoria tiene un tamaño de 12 KB. Se pide:

a) Construir una consulta donde se muestren los nombres y el domicilio de los clientes titulares que tengan cuentas con un salario > 500.000 euros y la ciudad de la sucursal sea distinta de Montevideo. Ordenar los resultados ascendentemente por el nombre del cliente. Expresarla en SQL y álgebra relacional.

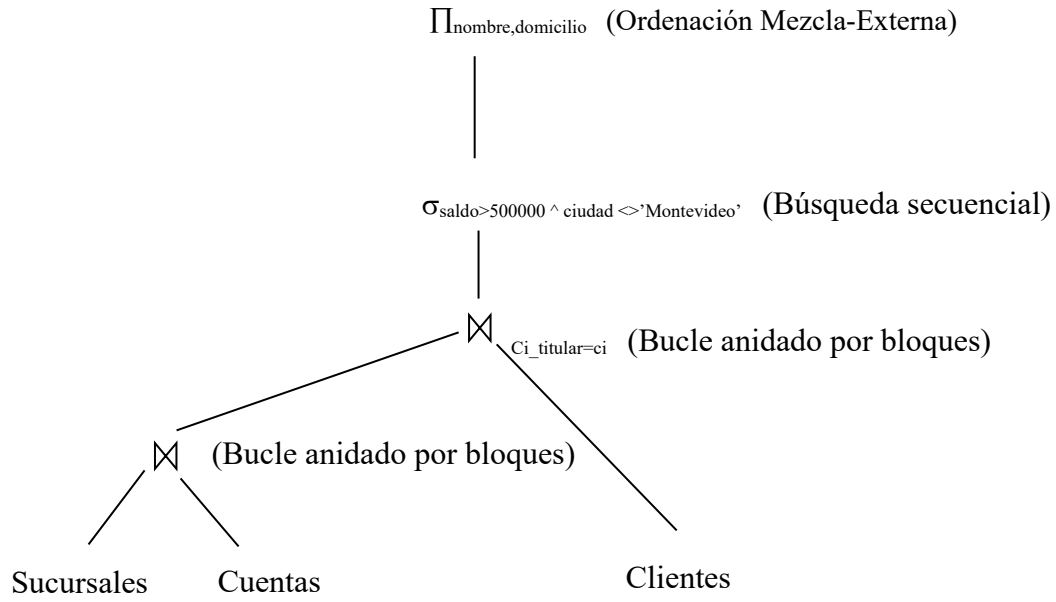
En SQL:

```
SELECT nombre,domicilio
FROM sucursales,cuentas,clientes
WHERE saldo > 500000 AND ciudad<>'Montevideo' AND
sucursales.nombre_sucursal=cuentas.nombre_sucursal AND ci_titular=ci
ORDER BY nombre ASC;
```

En el álgebra relacional:

$\Pi_{nombre,domicilio}$ ($\sigma_{saldo>50000 \wedge sucursales.nombre\_sucursak<>'Montevideo'}$ (sucursales $\bowtie$ cuentas $\bowtie$ clientes))

En forma de árbol podría ser:



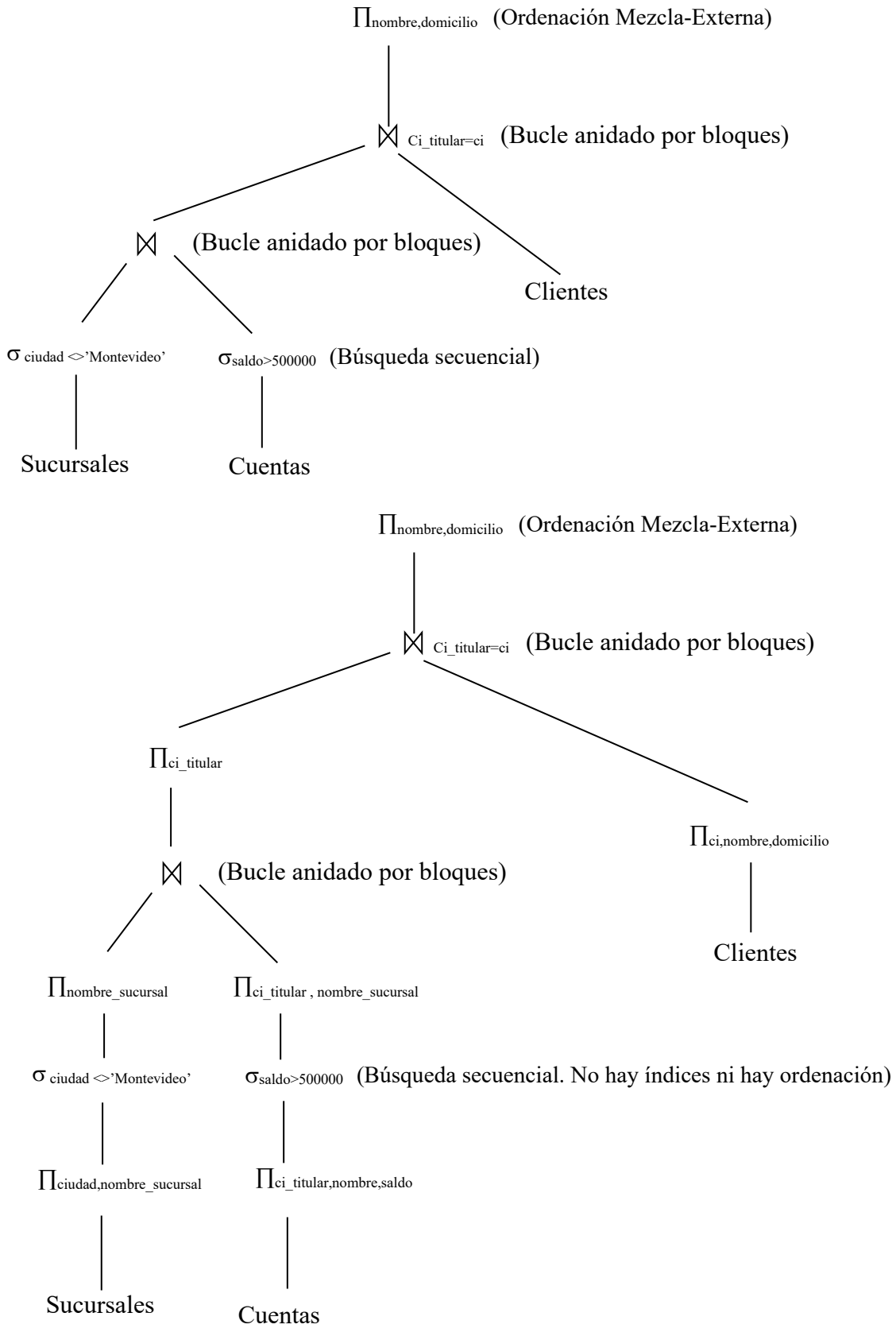
b) Construir un plan lógico mejorado para la consulta anterior utilizando la optimización heurística. Indicar los pasos intermedios realizados.

Para la optimización basada en la heurística:

1. Descomponer las selecciones conjuntivas en secuencia de operaciones sencillas.
2. Desplazar las operaciones de selección hacia la parte inferior del árbol.
3. Operaciones de reunión y selección que producirán menor tamaño se realizan primero.
4. Sustituir operaciones de X seguidas de σ_θ por operaciones de reunión.
5. Dividir las listas de atributos de proyección y llevarlas hacia abajo, creando nuevas si es necesario

6. Identificar subárboles cuyas operaciones pueden encauzarse y ejecutarlas utilizando encauzamiento.

Partiendo del árbol anterior, se obtiene lo siguiente:



Para este árbol se estiman el número de tuplas que se produce en cada operación, para así ver si hay que cambiar el orden de alguna rama:

T(sucursal)=40.000 registros
$T(\sigma_{ciudad <> 'Montevideo'})=n_r*N_v/V(ciudad,sucursal)=40.000*18/19=37895$ registros
T(cuentas)=100.000 registros
$T(\sigma_{saldo>500000})=100.000*500.000/1.000.000=50.000$ registros

T(1ª reunión (sucursal,cuenta)). La expresión general es $n_r*n_s/max(V(A,r),V(A,s)\}=37.895*50.000/max\{37.895,40.000\}=47.368.75=47.369$ tuplas, ya que en la rama de la izquierda el número de valores diferentes es min{37.895,40.000} y en la rama de la derecha es min{40.000,50.000}.

T(2ª reunión (cuenta,(sucursal,cuenta))). Si se aplica la expresión general será 47.369*100.000/max{47369,100.000}=47.369 tuplas, ya que en la rama de la izquierda la nueva estadística es de min{47369,100.000} y en la rama de la derecha 100.000.

Si se hace al revés:

T(1ª reunión (cuenta,cliente)). Aplicando la expresión general sería 100.000*50.000/max{50.00,100.000}=50.000 ya que la rama de la izquierda tiene una nueva estadística de min{50.000,100.000} y la rama de la derecha de 100.000

T(2ª reunión (sucursal,(cuenta,cliente))). Aplicando la expresión general de la reunión se tiene 37.895*50.000/max{37.895,40.000}=47.369 tuplas, ya que la estadística de la rama de la izquierda es de min{40.000,37.895} y la rama de la derecha el min{40.000,50.000}, luego se mantiene.

Luego nos quedamos con la ordenación de los joins anteriores.Dejamos el árbol como está y el árbol definitivo sobre el cuál se va a calcular el coste es el anterior, donde hay que darse cuenta que una de las ramas de cada reunión se puede encauzar.

c) Calcular el coste total asociado al plan anterior.

Para calcular el coste total, hay que tener en cuenta el plan físico que se propone y que viene dado por lo que hay disponible, así como las ramas que se pueden encauzar y además la memoria que se encuentra disponible.

<u>Rama 1</u>

Sucursales: 40.000 registros
Lregistro=40+40+8+50+4=142 bytes
N registros/ Bloque = $\lfloor 2*1024 / 142 \rfloor$ = 14 registros / bloque
Nº total bloques = $\lceil 40000 / 14 \rceil$ = 2858 bloques
$\Pi_{ciudad,nombre\_sucursal}$: 40.000 registros
Lregistro=40+40=80 bytes
N registros/ Bloque = $\lfloor 2*1024 / 80 \rfloor$ = 25 registros / bloque
Nº total bloques = $\lceil 40000 / 25 \rceil$ = 1600 bloques
$\sigma_{ciudad <> 'Montevideo'}$: produce 37895 registros

$\Pi_{,nombre\_sucursal}$: 37895 registros

Lregistro=40 bytes
N registros/ Bloque = $\lfloor 2*1024 / 40 \rfloor$ = 51 registros / bloque
Nº total bloques = $\lceil 37895 / 51 \rceil$ = 744 bloques

Rama 2

Cuentas: 100.000 registros

Lregistro=12+4+4+50+2+4=76 bytes
N registros/ Bloque = $\lfloor 2*1024 / 76 \rfloor$ = 26 registros / bloque
Nº total bloques = $\lceil 100000 / 26 \rceil$ = 3847 bloques

$\Pi_{ci\_titular,nombre\_sucursal,saldo}$: 100.000 registros

Lregistro=4+50+4=58 bytes
N registros/ Bloque = $\lfloor 2*1024 / 58 \rfloor$ = 35 registros / bloque
Nº total bloques = $\lceil 100000 / 35 \rceil$ = 2858 bloques

$\sigma_{saldo>500.000}$: produce 50.000 registros

$\Pi_{,ci\_titular,nombre\_sucursal}$: 50.000 registros

Lregistro=4+50=54 bytes
N registros/ Bloque = $\lfloor 2*1024 / 54 \rfloor$ = 37 registros / bloque
Nº total bloques = $\lceil 37895 / 37 \rceil$ = 1352 bloques

Rama 3

Cliente: 100.000 registros

Lregistro=4+50+50+4=108 bytes
N registros/ Bloque = $\lfloor 2*1024 / 108 \rfloor$ = 18 registros / bloque
Nº total bloques = $\lceil 100000 / 18 \rceil$ = 5556 bloques

$\Pi_{ci,nombre,,domicilio}$: 100.000 registros

Lregistro=4+50+50=104 bytes
N registros/ Bloque = $\lfloor 2*1024 / 104 \rfloor$ = 19 registros / bloque
Nº total bloques = $\lceil 100000 / 19 \rceil$ = 5264 bloques

Rama 1ª reunión

Reunión: 47.369 registros

Lregistro=4+50=54 bytes
N registros/ Bloque = $\lfloor 2*1024 / 54 \rfloor$ = 37 registros / bloque
Nº total bloques = $\lceil 47369 / 37 \rceil$ = 1281 bloques

$\Pi_{ci\_titular}$: 47.369 registros

Lregistro=4 bytes
N registros/ Bloque = $\lfloor 2*1024 / 4 \rfloor$ = 512 registros / bloque
Nº total bloques = $\lceil 47369 / 512 \rceil$ = 93 bloques

Rama 2ª reunión

Reunión: 47.369 registros

Lregistro=4+50+50=104 bytes
N registros/ Bloque = $\lfloor 2*1024 / 104 \rfloor$ = 19 registros / bloque
Nº total bloques = $\lceil 47369 / 19 \rceil$ = 2494 bloques

$\Pi_{nombre,domicilio}$: 47.369 registros

Lregistro=50+50 bytes

N registros/ Bloque = ⌊2*1024 / 100⌋ = 20 registros / bloque
Nº total bloques = ⌈ 47369 / 20 ⌉ = 2369 bloques

Sólo queda calcular el coste de las operaciones (reunión, selección y ordenamiento) y ver cuál de las ramas se va a materializar. Una de las ramas de cada reunión se puede encauzar y como al final hay que realizar una ordenación, es necesario materializar antes de ordenar.

Para las reuniones hay que contar que hay una memoria de 12 Kb y como el bloque es de 2 Kb, entonces eso indica que hay 6 bloques disponibles para realizar la operación. En ese caso, el coste de realizar la reunión es de:

⌈ br / (M-2) ⌉*bs+br

Si además una de la entrada se encauza, entonces el coste de leer una de las entradas desaparece y el coste asociado es: ⌈ br / (M-2) ⌉*bs

Lo que falta por decidir es en la operación de reunión, cuál será br y cuál bs. Para ello se calculan los posibles costes de cada operación de reunión:

1ª reunión: hay dos entradas de 744 y 1352 bloques:
- Si br=744, coste= ⌈ 744 / (6-2) ⌉*1352=251.472 bloques
- Si br=1352, coste= ⌈ 1352 / (6-2) ⌉*744=251.472 bloques

Generalmente se encauza la rama de 1352 bloques (la de mayor bloques) que debe de ser br y se materializa la de 744 (la de menor bloques)

2ª reunión: hay dos entradas de 93 y 5264 bloques:
- Si br=93, coste= ⌈ 93 / (6-2) ⌉*5264=126.336 bloques
- Si br=5264, coste= ⌈ 5264 / (6-2) ⌉*93=122.388 bloques

Se elige como br (relación externa) la rama de 5264 bloques que además se encauza y se materializa la rama de 93 bloques.

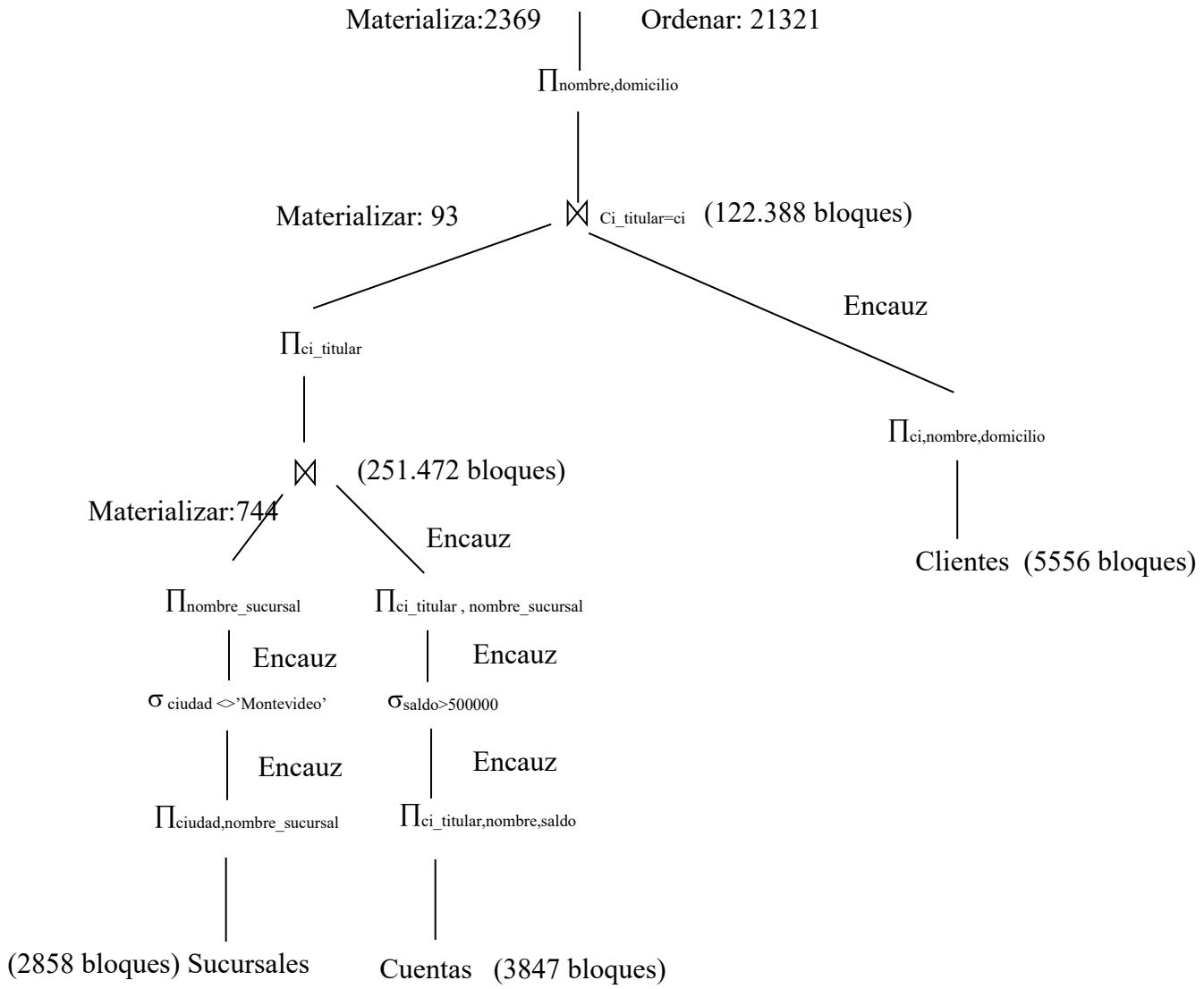
Y por último sólo queda ordenar utilizando el algoritmo de mezcla-externa (no hay memoria suficiente) la proyección (nombre,domicilio) de 2369 bloques, que además debe de estar materializada, siendo el coste:

$br*[2*\lceil \log_{M'-1} (br / M) \rceil+1]$

Para esta memoria, hay 6 bloques, de los cuales 5 se dejan para las secuencias de entrada y otro para la secuencia de salida. Se podrán formar secuencias de 6 bloques de tamaño. El coste asociado será:

$2369*[2*\lceil \log_{5} \lceil 2369 / 6 \rceil \rceil+1] = 2369*(2*4+1)=21.329$ bloques.

Luego el árbol final con encauzamiento y materialización será:



El coste total será sumar cada uno de los costes: Leer sucursales + materializar 744 + leer cuentas + coste del primer join + materializar 93 bloques + leer clientes + coste del segundo join + ordenar la salida = 2858+744+3847+251.472 +93+5556+122388+2369+21321=410.648 bloques

d) Escribir la consulta SQL del plan lógico mejorado.

```
SELECT nombre,domicilio
FROM cliente
        SELECT ci,nombre,domicilio
        FROM cliente
        INNER JOIN
                SELECT ci_titular
                FROM
                        SELECT ci_titular,nombre_sucursal
                        FROM  (SELECT ci_titular,nombre_sucursal,saldo
                                FROM cuentas) AS t
                        WHERE t.saldo > 500000
```

```
            INNER JOIN
                SELECT nombre_sucursal
                FROM (SELECT ciudad,nombre_sucursal
                        FROM sucursales) AS s
                WHERE s.ciudad<>'Montevideo'
            ON sucursales.nombre_sucursal=cuentas.nombre_sucursal
        ON sucursales.ci_titular=clientes.ci
ORDER BY cliente.nombre ASC;
```

e) Indicar los cambios que se podrían introducir que podrían beneficiar el coste total del plan anterior.

La memoria es muy pequeña en relación a la cantidad de tuplas que hay que procesar, luego la solución sería aumentar el tamaño de la memoria.

Ejercicio 6

a) La consulta sería:

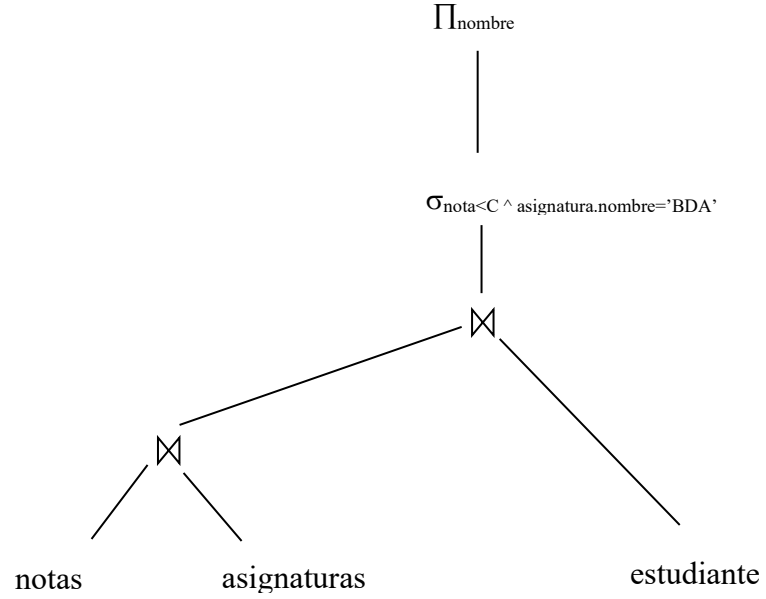
```
Select nombre
From estudiante,nota,asignatura
Where nota <'C' and asignatura.nombre='BDA' and estudiante.dni=nota.dni and
nota.id_asig=asignatura.id_asig;
```

En el álgebra relacional:

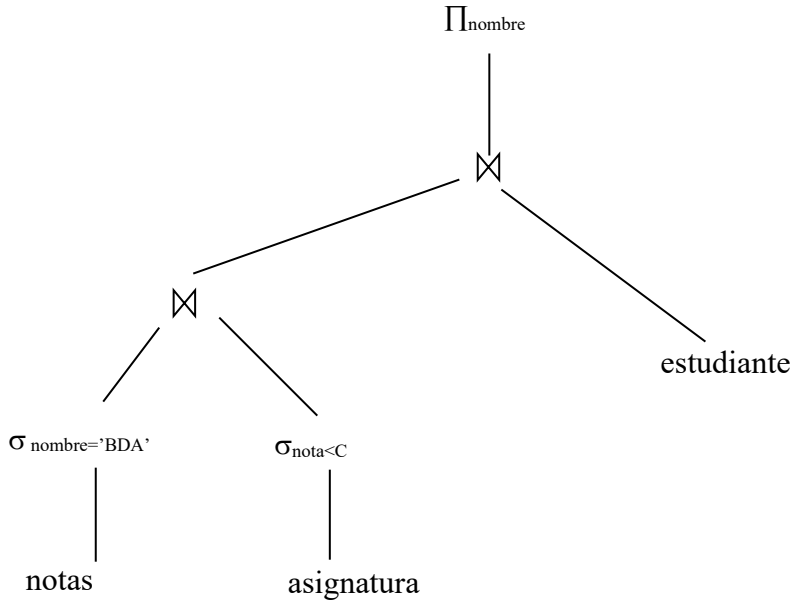
$$\Pi_{nombre}\,(\,\sigma_{nota<C\ \wedge\ asignatura.nombre='BDA'}\,(estudiante \bowtie notas \bowtie asignatura))$$

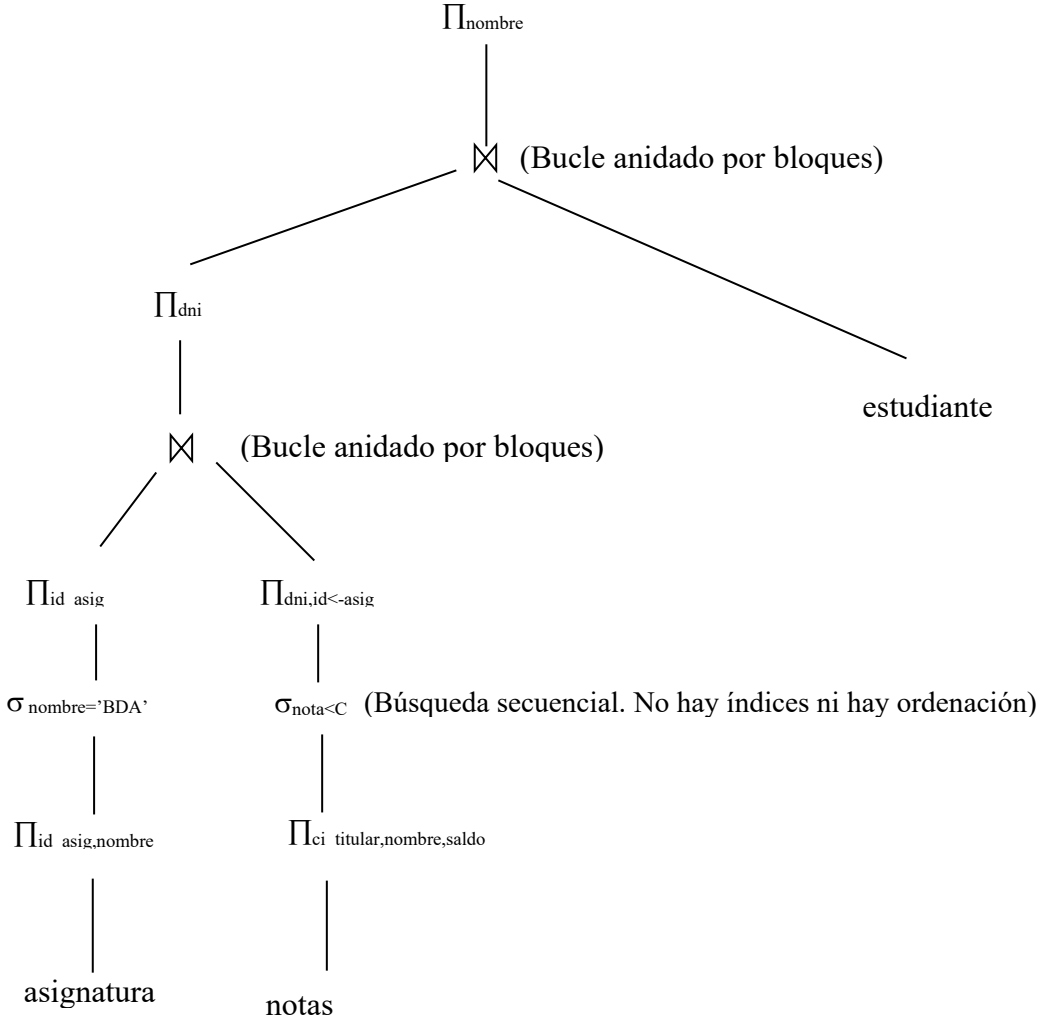
b)

En forma de árbol podría ser:



Aplicando heurística se obtiene lo siguiente:





Plan físico: para las selecciones se realiza búsqueda secuencial y para las reuniones bucle anidado por bloques. No hay índices ni ordenación de ficheros.

Para este árbol se estiman el número de tuplas que se produce en cada operación, para así ver si hay que cambiar el orden de alguna rama:

T(asignatura)=5000 registros
$T(\sigma_{nombre='BDA'})$=1 registro
T(notas)=600.000 registros (40000*15)
$T(\sigma_{notas<C})$=0.5*600.000+0.25*600.000=450.000 registros

T(1ª reunión). Aplicando la expresión general se tiene nr*ns/max{V(A,r),V(A,s)} = 1*450000/max{1,5000}=90 tuplas, ya que en la rama de la izquierda el número de asignaturas diferentes es 1, y por la derecha entre min{5000,450.000}.

T(2ª reunión). Aplicando la expresión general se tiene 90*40.000/max{90,40000}=90 tuplas, ya que en la rama de la izquierda sería la estadística min{90,40000} y por la derecha serían 40.000 estudiantes.

Dejamos el árbol como está y el árbol definitivo sobre el cuál se va a calcular el coste es el anterior, donde hay que darse cuenta que una de las ramas de cada reunión se puede encauzar.

c) Calcular el coste total asociado al plan anterior.

Para calcular el coste total, hay que tener en cuenta el plan físico que se propone y que viene dado por lo que hay disponible, así como las ramas que se pueden encauzar y además la memoria que se encuentra disponible. Como es el peor caso, debe de haber 3 bloques de memoria para la operación de reunión.

Rama 1

Asignatura: 5.000 registros
Lregistro=20+40+40=100 bytes
N registros/ Bloque = $\lfloor 1024 / 100 \rfloor$ = 10 registros / bloque
Nº total bloques = $\lceil 5000 / 10 \rceil$ = 500 bloques
$\Pi_{id\_asig,nombre}$: L_R=60 bytes, n_R= 5000 , f_R= 17 reg/bloque , b_R= 295 bloques
$\sigma_{nombre='GABD'}$: produce 1 registros
$\Pi_{,id\_asig}$: L_R=20 bytes, n_R= 1 , f_R= 51 reg/bloque , b_R= 1 bloques

Rama 2

Notas: 15x40000=600000 registros
Lregistro=20+20+40=80 bytes
N registros/ Bloque = $\lfloor 1024 / 80 \rfloor$ = 12 registros / bloque
Nº total bloques = $\lceil 600000 / 12 \rceil$ = 7500 bloques
$\sigma_{nota<C}$: produce 450.000 registros
$\Pi_{,dni,id\_asig}$: L_R=40 bytes, n_R= 450000 , f_R= 25 reg/bloque , b_R= 18000 bloques

Primera reunión: Si se encauza una rama, coste=br*bs =18000*1=18000 bloques, luego se encauzaría la rama de 18000 bloques y se materializaría la rama de 1 bloque. Pero se puede ver que una rama entra completamente en RAM, luego el coste del primer join es cero.

$\Pi_{,dni}$: L_R=20 bytes, n_R= 90 , f_R= 51 reg/bloque , b_R= 2 bloques

Rama 3

Estudiante: 40.000 registros

Lregistro=20+40=60 bytes
N registros/ Bloque = $\lfloor 1024 / 60 \rfloor$ = 17 registros / bloque
Nº total bloques = $\lceil 40000 / 17 \rceil$ = 2353 bloques

Segunda reunión: Si se encauza una rama, coste=br*bs=2353*2=4706 bloques, se encauza la rama de la izquierda ya que la rama de ,a derecha ya está materializada al ser una relación de la base de datos.

Π_{nombre}: L_R=40 bytes, n_R= 90 , f_R= 25 reg/bloque , b_R= 4 bloques

El coste total será sumar cada uno de los costes: Leer asignatura + leer notas + materializar 2 bloques + leer estudiante + coste del segundo join = 500+7500+2+2353+4706=15061 bl.

d) El coste viene dado principalmente por la escasez de memoria. Habría que aumentar la memoria para la operación de reunión.

Ejercicio 7

a) Primero calcularemos la lista de tuplas de R que ***sí*** concuerdan con alguna tupla de S:
Una reunión natural de R y S (R⋈S) contiene las tuplas en las que ***sí*** concuerdan los atributos comunes, además de los atributos no comunes de las dos relaciones. Para obtener únicamente los atributos de R, que llamaremos r, deberemos hacer una proyección de éstos:

$$U = \Pi_r (R \bowtie S)$$

Ahora solo queda eliminar de toda la relación R, las tuplas anteriores (las que sí concuerdan):

$$T = R - U$$

$$\boxed{T = R - \Pi_r (R \bowtie S)}$$

b) Dado que la expresión final es de la forma **T = R – U**, y aplicando la fórmula estudiada (**conservadora**) del tamaño de una resta: **tam(r – u) = tam(r)**, resulta finalmente y simplemente: **tam(T) = tam(R)** .
¿Aún así, podría obtenerse una estimación "mejor"? Esta estimación consistiría en realizar el cálculo basado en la expresión de álgebra relacional completa. La estimación conservadora presupone el caso peor, que es aquel en el que tam(U) = 0. El caso contrario, no muy acertado, seria presuponer que el tam(U) es máximo. Finalmente, la nueva estimación (**ajustada**) del tamaño de T seria aquella en la que el tamaño de U toma un valor intermedio:

tam(T) = tam(R) - ½ tam(U) . Dado que no conocemos si R y S tienen algún atributo común que sea clave, usaríamos la expresión más general para las reuniones naturales. Con todo ello queda:

Relación	**Tamaño (conservador)** **tam(T) = tam(R)**	**Tamaño (ajustado)** **tam(T) = tam(R) – ½ tam(U)**
R	N_R	N_R
S	-	N_S
$R \bowtie S$	-	$(N_R * N_S) / \max (V(A,R), V(A,S))$
U	0	$(N_R * N_S) / \max (V(A,R), V(A,S))$
T = R - …	$\mathbf{N_R}$	$\mathbf{N_R – ½ (N_R * N_S) / \max (V(A,R), V(A,S))}$

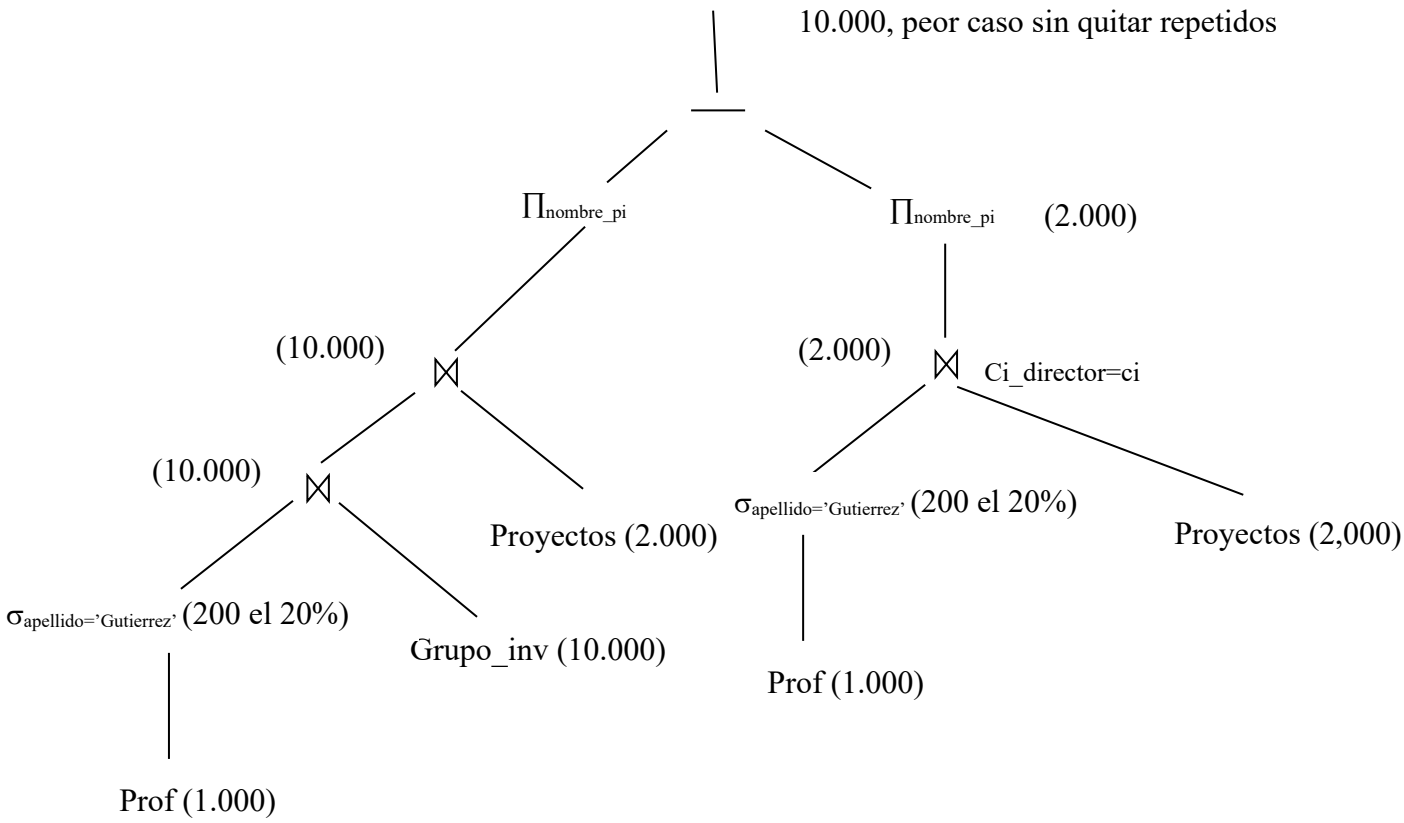
Ejercicio 8

Aunque existen diferentes consultas SQL que sean solución del problema, la más simple consistiría en seguir literalmente el enunciado: **(... nombre de...) QUE NO (... sean...)**. Con ello quedaría:

```
SELECT nombre_pi
FROM proyectos, grupo_inv, prof
WHERE apellido = 'Gutierrez' AND proyectos.n_proyi = grupo_inv.n_proyi AND ci_inv = ci
        EXCEPT
SELECT nombre_pi
FROM proyectos, prof
WHERE apellido = 'Gutierrez' AND ci_director = ci;
```

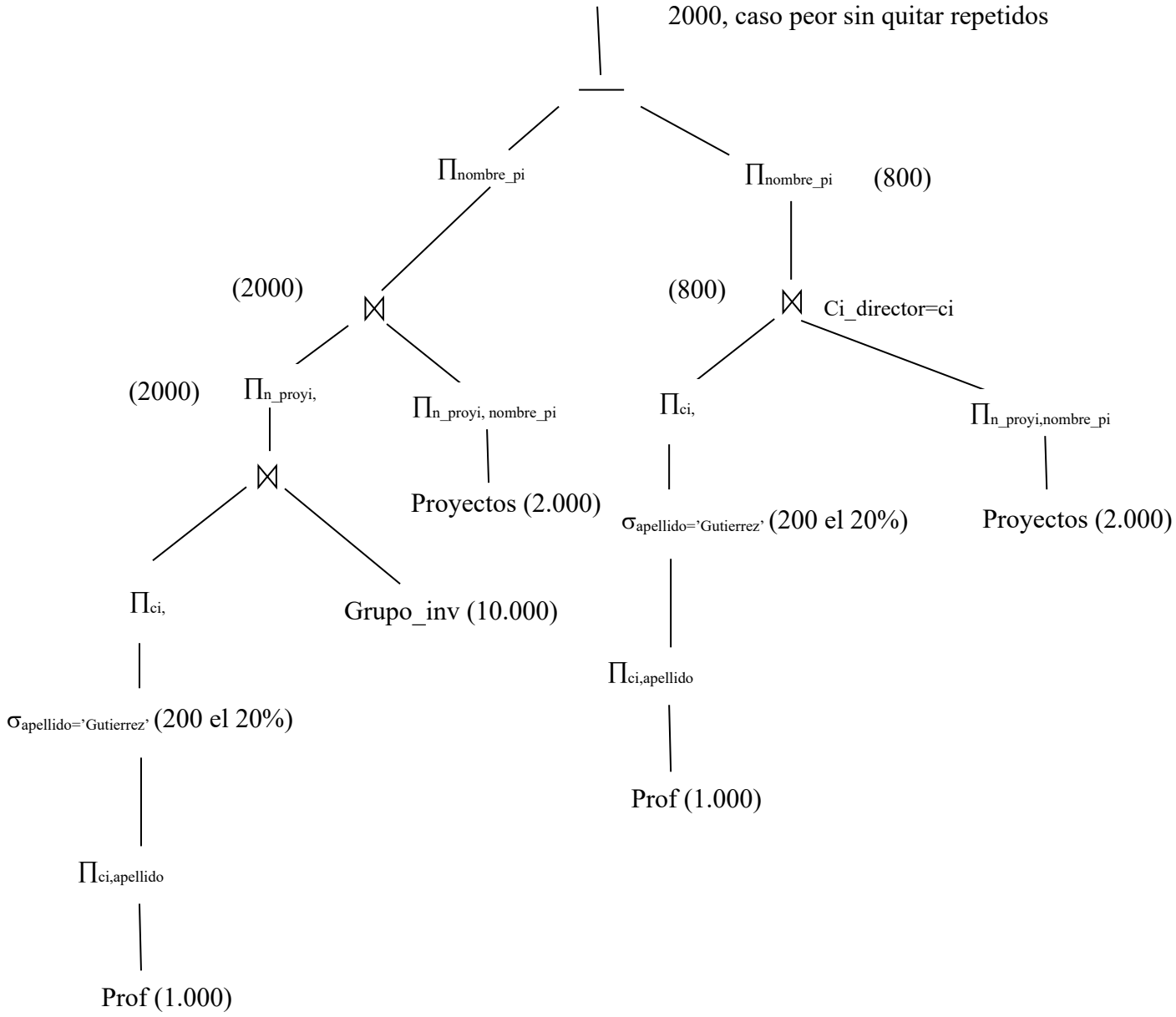
b) Optimizar la consulta anterior, indicando el plan lógico y físico final que se aplicaría.

Un posible árbol asociado a la consulta anterior podría ser:



Para la optimización se introducen proyecciones adicionales y después se ve si se puede encauzar alguna rama. Hay que notar que la resta no es conmutativa, luego parece lógico hacer primero la reunión con la rama de la selección que con la que no la tiene y la segunda rama de la resta tiene que ser así.

T(Prof)=1000 reg
T($\sigma_{apellido='Gutierrez'}$)=nr/V(apellido,Prof)=1000*0.2=200 reg
T(Grupo_inv)=10000 reg
T(primer join)= nr*ns/max{V(A,r),V(A,s)}=200*10000/max{200,1000}=2000 tuplas, ya que la rama de la izquierda tiene una estadística de min{200,1000} y la rama de la derecha de 1000 profesionales.
T(Proyectos)=2000 reg
T(segundo join) = nr*ns/max{V(A,r),V(A,s)}=2000*2000/max{2000,2000}=2000 tuplas ya que la rama de la izquierda tiene una estadística de min(2000,2000) y la rama de la derecha tiene una estadística de 2000 proyectos diferentes.
T(tercer join) = nr*ns/max{V(A,r),V(A,s)}= 200*2000/max{200,500}=800 reg.
T(resta) sería como máximo el número de registros de la rama de la izquierda = 2000 tuplas.



La selección se hace mediante lectura secuencial.
Las reuniones mediante bucle anidado por bloques.
Y el operador diferencia mediante bucle anidado por bloques.

c) Calcular el coste asociado al plan anterior.

Se calcula el coste de esta consulta. Para ello se determina el número de tuplas y bloques de cada operación básica:

Prof: Longitud del registro, $L_R = 4 + 20 + 20 + 20 = 64$ bytes
Número de registros, $n_R = 1.000$
Factor de bloques, $f_R = \lfloor 2.048 / 64 \rfloor = 32$ reg/bloque
Número de bloques, $b_R = \lceil 1000/ 32 \rceil = 32$ bloques

$\prod_{ci,apellido}$: L_R = 24 bytes, n_R = 1.000 registros, f_R = 85 reg/bloque, b_R = 12 bloques
$\sigma_{apellido='Gutierrez'}$: Peor caso 20% de 1.000 registros = 200 registros.
$\prod_{ci}$: L_R = 4 bytes, n_R = 200 registros, f_R = 512 reg/bloque, b_R = 1 bloque

Grupo_inv: L_R = 8 bytes, n_R = 10.000 registros, f_R = 256 reg/bloque, b_R = 40 bloques

Primer Join: L_R = 12 bytes, n_R = 2000 registros, f_R = 170 reg/bloque , b_R = 12 bloques
$\prod_{n\_proyi}$: L_R = 4 bytes, n_R = 2000 registros, f_R = 512 reg/bloque, b_R = 4 bloques

Proyectos: L_R = 48 bytes, n_R = 2.000 registros, f_R = 42 reg/bloque, b_R = 48 bloques
$\prod_{n\_proyi,nombre\_pi}$: L_R = 24 bytes, n_R = 2.000 registros, f_R = 85 reg/bloque, b_R = 24 bloques

Segundo Join: L_R = 24 bytes, n_R = 2000 registros, f_R = 85 reg/bloque, b_R = 24 bloques
$\prod_{,nombre\_pi}$ (rama izquierda): L_R = 20 bytes, n_R = 2000 registros, f_R = 102 reg/bloque, b_R = 20 bloques

Tercer Join: L_R = 28 bytes, n_R = 800 registros, f_R = 73 reg/bloque, b_R = 11 bloques
$\prod_{,nombre\_pi}$ (rama derecha): L_R = 20 bytes, n_R = 800 registros, f_R = 103 reg/bloque, b_R = 8 bloques

Coste de las operaciones:

Generalmente se puede encauzar una rama, luego el coste será: $\lceil b_r / (M - 2) \rceil * b_s$, donde M son los bloques de Memoria disponibles, que en este caso son 24 Kb / 2 Kb = 12

Primer Join:
Si b_r = 1, C = $\lceil 1 / 10 \rceil * 40$ = 40 bloques
Si b_r = 40, C = $\lceil 40 / 10 \rceil * 1$ = 4 bloques
Luego se materializa la rama de la izquierda, donde el coste será 1, y se encauza la rama de la derecha de 40 bloques. Pero M=12, luego cabe una rama completa, luego el coste del join será de 40 bloques ya que se lee la rama de la derecha del disco.

Segundo Join:
Si b_r = 4, C = $\lceil 4 / 10 \rceil * 24$ = 24 bloques
Si b_r = 24, C = $\lceil 24 / 10 \rceil * 4$ = 12 bloques
Se materializaría la rama de 4 bloques y se encauzaría la de la derecha 24 bloques, pero como M=10 y una entrada vale 4 bloques, el coste del segundo join es cero.
Tercer join:
Si b_r = 1, C = $\lceil 1 / 10 \rceil * 8$ = 8 bloques
Si b_r = 24, C = $\lceil 8 / 10 \rceil * 1$ = 3 bloques
Luego se materializa la rama de la izquierda 1 bloque y se encauza la de la derecha 24 bloques. Pero se puede ver que la de la izquierda se puede mantener en RAM luego el coste del tercer join es cero.

Para la operación diferencia se realiza como un bucle anidado y si se utiliza toda la memoria disponible para realizar esa operación, se puede ver que la rama de la derecha no cabe en RAM luego hay que esperar a que se reciba entera y por lo tanto 8 bloques hay que materializar. Si se encauza la rama de la izquierda que son 20 bloques, entonces el coste será:

Si b_r = 20, C = $\lceil 20 / 10 \rceil * 8$ = 16 bloques

Luego el coste sería:

Coste = 32 (leer prof) + 40 (coste del primer join (notar que la lectura de Grupo_inv está incluida en el join)) + 48 (leer proyectos) + 32 (leer prof) + 48 (leer proyectos) + 8 (materializar la entrada rama derecha de la resta)+16 (coste operación diferencia) = **224 bloques**

Hay que notar que hay dos join cuyo coste es cero porque una entrada se puede guardar en RAM y otra encauzar.

d) Que cambios serían convenientes introducir para reducir el coste asociado al plan anterior.

Algunos ejemplos de mejoras:
- Para este caso lo que se podría hacer sería reutilizar expresiones ya calculadas, como por ejemplo la rama que tiene que ver con la tabla prof se repite tanto en la subconsulta de la izquierda como de la derecha, luego se podría materializar el resultado y utilizarlo posteriormente.
- Por supuesto un aumento de memoria para las operaciones mejoraría el coste de la consulta.

Ejercicio 9

Para las demostraciones, se pueden usar las definiciones formales o alguna propiedad del operador. Para refutar una afirmación, basta con mostrar un contraejemplo.

(a) ρ_S – **SI**
ρ_S (R) renombra una relación, dándole un nuevo nombre S, y manteniendo las tuplas y los atributos originales. Volver a renombrar la relación a S mediante este operador, ya no modificaría ninguna característica de la relación en absoluto: ρ_S (ρ_S (R)) = ρ_S (R)

(b) σ_C – **SI**
$\forall$ C1, C2 $\rightarrow$ σ_{C1} (σ_{C2} (R)) = $\sigma_{C1 \wedge C1}$ (R)
Haciendo C1 = C2 = C, se obtiene: σ_C (σ_C (R)) = $\sigma_{C \wedge C}$ (R) = σ_C (R)

(c) π_L – **SI**
$\forall$ L1 $\subseteq$ L2 $\rightarrow$ π_{L1} (π_{L2} (R)) = π_{L1} (R)
Haciendo L1 = L2 = L, se obtiene directamente: π_L (π_L (R)) = π_L (R)

(d) τ_A (ordenación de un conjunto de tuplas por un atributo A) – **NO**
Por ejemplo, considere la relación R (A,B) con dos tuplas: { (1,11), (1, 22) }. El operador τ_A (R) puede devolver las dos tuplas en cualquier orden. Al ordenarlas otra vez por A puede cambiar el orden.

Ejercicio 10

(a) $T(\sigma_{A=7}(R)) = \dfrac{T(R_1)}{V(R_1, A)}$

$= \dfrac{100}{8} =$ **12,5** (R_1 se refiere a las tuplas con valores de A entre 1 y 10)

(b) $T(\sigma_{A=17}(R)) = \dfrac{T(R_2)}{DOM(R_2, A)}$

$= \dfrac{200}{10} =$ **20** (R_2 se refiere a las tuplas con valores de A entre 11 y 20)

(c) $T(\sigma_{A>17}(R)) = \sum_i n_i \times \dfrac{T(R_i)}{DOM(R_i, A)}$

$$= 3 \times \frac{200}{10} + 10 \times \frac{300}{10} + 10 \times \frac{400}{10} = 60 + 300 + 400 = \mathbf{760}$$

(n_i se refiere al número de valores diferentes que cumplen la condición en cada subrango, y R_i se refiere a las tuplas con valores de A en cada subrango, i = 1, ..., 4)

(d) Se puede tratar cada subrango del histograma i = 1, …, 4 como una relación en sí misma, con:

$$T(R_i \bowtie S_i) = \frac{T(R_i) \times T(S_i)}{\max\{V(R_i, A), V(S_i, A)\}} = \frac{T(R_i) \times T(S_i)}{V(R_i, A)}$$

$$T(R \bowtie S) = \frac{100^2}{8} + \frac{200^2}{5} + \frac{300^2}{10} + \frac{400^2}{10} = 1.250 + 8.000 + 9.000 + 16.000 = \mathbf{34.250}$$

Ejercicio 11

(a) El mejor plan (menor número de I/Os) es ejecutar $R_1 \bowtie R_3$ primero, y después ejecutar la reunión del resultado anterior con R_2.

(b) **Número de I/Os: 18.200**
El coste (en I/Os) es: 3 × (1.000 + 3.000) (para ejecutar $R_1 \bowtie R_3$), más 100 (para escribir el resultado de la primera reunión en cajones), más 2 × 2.000 (para crear los cajones de R_2), más 100 + 2.000 (para leer los cajones de la reunión final). Todo esto suma un total de 18.200 bloques.

Ejercicio 12

Se puede resolver razonando, sin necesidad de recordar la fórmula de reunión asociativa (*hash*)

(a) **Plan 1**

Op	Coste
1.1	$T(P) = \frac{T(R)}{V(R,B)} = \frac{60.000}{12} = \mathbf{5.000}$
1.2	$\frac{T(P) \times T(S)}{\max\{V(P,B), V(S,B)\}} = \frac{5.000 \times 30.000}{\max(1,5)} =$ **30.000.000**

Coste: 5.000+30.000.000 = **30.005.000**

Plan 2

Op	Coste
2.1	$T(P) = \frac{T(R)}{V(R,B)} = \frac{60.000}{12} = \mathbf{5.000}$
2.2	$T(Q) = \frac{T(S)}{V(S,B)} = \frac{30.000}{5} = \mathbf{6.000}$
2.3	$\frac{T(S) \times T(Q)}{\max\{V(Q,B), V(Q,B)\}} = \frac{5.000 \times 6.000}{\max(1,1)} =$ **30.000.000**

Coste: 5.000+6.000+30.000.000 = **30.011.000**

Según lo anterior, el Plan 1 es el mejor plan.

(b) Primero leemos R en memoria (B(R) = 60.000/10 = **6.000** I/Os) y ejecutamos [1.1]. A continuación ejecutamos [1.2] de la siguiente forma: Construimos los cajones para P (la relación intermedia resultado de la selección sobre R) y los escribimos a disco. Como T(P) = T(R)/12, escribir los cajones de P requerirán B(P) = B(R)/12 = **500** I/Os. A continuación leemos S (B(S) = 30.000/30 = 1.000 I/Os), construimos los cajones, y escribimos los cajones a disco (en total **2 × 1.000** I/Os de disco). Finalmente, leemos P y S en memoria, un cajón de cada relación cada la vez, y ejecutamos la reunión (**500 + 1.000** I/Os de disco).

El número total resultante de I/Os de disco es de:

$$6.000 + (500 + 2 \times 1.000) + (500 + 1.000) = \mathbf{10.000}.$$

(c) Primero leemos R en memoria (B(R) = 60.000/10 = **6.000** I/Os) y ejecutamos [2.1]. A continuación leemos S en memoria (B(S) = 30.000/30 = **1.000** I/Os) y ejecutamos [2.2]. Finalmente, ejecutamos [2.3] como sigue. Construimos los cajones de P (la relación intermedia resultado de la selección sobre R) y los escribimos a disco (**500** I/Os). De forma análoga, construimos los cajones de Q (la relación intermedia resultado de la selección sobre S) y los escribimos a disco (B(Q) = B(S)/5 = **200** I/Os). Finalmente leemos P y Q en memoria, un cajón de cada uno cada vez, y ejecutamos la reunión (**500 + 200** I/Os de disco).

El número total resultante de I/Os de disco es de:

$$6.000 + 1.000 + 2 \times (500 + 200) = \mathbf{8.400}.$$

(d) Según el apartado (a) el menor número de tuplas resultantes (y por lo tanto el mejor plan) corresponde al Plan 1. Sin embargo, el mejor plan según el número de I/Os (o bloques) de disco es el Plan 2. La razón de esta paradoja, reside en que los costes en tuplas dependen sólo de las operaciones involucradas, mientras que los costes en I/Os dependen de los algoritmos empleados para ejecutar esas operaciones y del tamaño o longitud de cada tupla.

Ejercicio 13

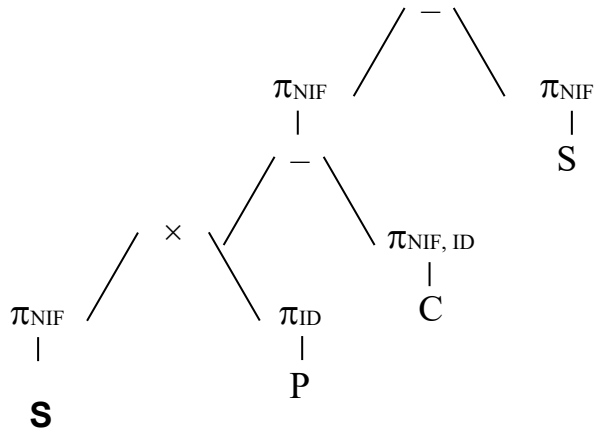
(a) Esta consulta, muy común en realidad, es en esencia una simple **división**. Sin necesidad de recordar la expresión se puede deducir como:

$R' = \pi_{NIF}$ (Suministradores) $\times \pi_{ID}$ (Piezas) [Todos Suministr. con todas Piezas, disponibles o no]

$R'' = R' - \pi_{NIF, ID}$ (Catalogo) [Suministr. con alguna Pieza **no** disponible]

$R''' = \pi_{NIF}$ (Suministradores) $- \pi_{NIF}$ (R'') [Suministr. que tienen todas las Piezas disponibles]

(b) El diagrama con el árbol resultante es:



No existe optimización alguna, pues el diagrama ya está optimizado. Razones:

- No existen condiciones de selección en esta consulta
- Las proyecciones incluyen los atributos mínimos desde el inicio

(c) Para calcular el coste asociado en el peor caso se calculan cada una de las tuplas que se generan de cada operación. No hay índices en las tablas y se utiliza el bucle anidado por

bloques en el peor caso, asimismo, como estamos en el peor caso, no hay encauzamiento y se materializa.

Suministradores: Longitud del registro, $L_R = 20 + 20 + 8 = 48$ bytes
Número de registros, $n_R = 100$.
Factor de bloques, $f_R = \lfloor 1.024 / 48 \rfloor = 21$ reg/bloque
Número de bloques, $b_R = \lceil 100 / 21 \rceil = 5$ bloques
$\prod_{NIF}$: $L_R = 8$ bytes, $n_R = 100$ registros, $f_R = 128$ reg/bloque, $b_R = 1$ bloque

Piezas: Longitud del registro, $L_R = 20 + 20 + 8 = 48$ bytes
Número de registros, $n_R = 2.000$.
Factor de bloques, $f_R = \lfloor 1.024 / 48 \rfloor = 21$ reg/bloque
Número de bloques, $b_R = \lceil 2.000 / 12 \rceil = 96$ bloques
$\prod_{ID}$: $L_R = 8$ bytes, $n_R = 2.000$ registros, $f_R = 128$ reg/bloque, $b_R = 16$ bloques

Catálogo: $L_R = 36$ bytes, $f_R = 28$ registros/bloque, $b_R = 143$ bloques
$\prod_{NIF, ID}$: $L_R = 16$ bytes, $n_R = 4.000$ registros, $f_R = 64$ reg/bloque, $b_R = 63$ bloques

El producto cartesiano se hace por medio de bucles anidados por bloques:
$n_R = 2.000 \times 100 = 200.000$ registros
Coste asociado: br * bs + br = 1 * 16 + 1 = 17
$L_R = 16$ bytes, $f_R = 64$ reg/bloque, $b_R = 3125$ bloques

Primera operación diferencia: se hace por medio de bucles anidados por bloques:
$n_R = 200.000$ registros, que será el peor caso.
Coste asociado: br * bs + br = 3125 * 63 + 63 = 196938 bloques
$L_R = 16$ bytes, $f_R = 64$ reg/bloque, $b_R = 3125$ bloques.

$\prod_{NIF}$ (-): $L_R = 8$ bytes, $n_R = 200.000$ registros, $f_R = 128$ reg/bloque, $b_R = 1563$ bloques

Segunda operación diferencia: se hace por medio de bucles anidados por bloques:
$n_R = 200.000$ registros, que será el peor caso.
Coste asociado: br * bs + br = 1 * 1563 + 1 = 1563 bloques

Coste total = 5 (lectura **S**uministradores) +1 (grabar $\prod_{NIF}$) + 5 (lectura **S**uministradores) + 1 (grabar $\prod_{NIF}$) + 96 (lectura **P**iezas) + 16 (grabar $\prod_{ID}$) + 17 (Coste de x) + 3125 bloques (grabar salida x) + 143 (lectura **C**atalogo) + 63 (grabar $\prod_{NIF, ID}$) + 196938 (Coste de primera diferencia) + 3125 (grabar salida primera diferencia) + 3125 (leer salida primera diferencia) + 1563 (grabar $\prod_{NIF}$ (-)) + 1563 (coste segunda diferencia) + 1563 (grabar salida de la consulta)= **211349 bloques**

(d) Aparentemente, la única mejora directa consistiría en almacenar el resultado de la operación π_{NIF} (Suministradores), dado que se utiliza dos veces en la consulta.

- El ahorro mínimo sería el coste de la lectura de la relación S (5 bloques), más el coste de la escritura del resultado de la operación π_{NIF} (1 bloque).

Otra posible mejora consistiría en encauzar ciertas operaciones en vez de materializarlas.

Ejercicio 14

(c) *Justifique* cual es el mejor plan para calcular la consulta. **[2 p]**

El mejor plan es aquel cuyo ***resultado intermedio*** genere menos ***bloques***, es decir: $(R_1 \bowtie R_3) \bowtie R_2$, puesto que, en este caso, $(R_1 \bowtie R_3)$ genera sólo 100 bloques.

(d) ¿Cuántas operaciones de I/O se necesitan para este mejor plan?. *Explique* brevemente cómo ha calculado su respuesta. **[3 p]**

$(R_1 \bowtie R_3)$ Coste operación: 3 $(b_{R1} + b_{R3})$ = 3 * (3.200 + 1.600) = **14.400** bloques

$S_{1,3}$ Coste materializar (según enunciado) salida = **100** bloques

$S_{1,3} \bowtie R_2$ Coste operación: 3 $(b_{S1,3} + b_{R2})$ = 3 * (100 + 800) = **2.700** bloques

Coste total = 14.400 + 100 + 2.700 = 17.200 bloques

(e) Calcule cual sería el valor mínimo de bloques de memoria que permite, según el enunciado, que "*... hay suficiente memoria solamente para efectuar las operaciones de reunión...*". *Justifíquelo*. **[4 p]**

- Entrada: Es suficiente con **1 bloque** (no es preciso utilizar mas bloques)
- Operación de reunión:
  - Construcción de cajones: Para R_1 necesitamos tener en memoria de forma simultánea tantos bloques como cajones (N) determine la función *hash*, es decir, tantos como valores distintos tenga: **N bloques**

    Una vez acabada la tarea de construir los cajones de R_1, para R_3 se pueden reutilizar los mismos N bloques (recordar que la función *hash* es la misma para las 2 relaciones, por lo que el rango de salida es el mismo)
  - Lectura de los cajones: Es suficiente con 2 bloques, uno por cada relación de entrada. Como se ha terminado la fase de construcción de cajones, se pueden reutilizar 2 de los N+1 bloques anteriores
- Salida: Es suficiente con 1 bloque. Serviría también 1 de los N+1 anteriores

Bloques necesarios = N + 1 $(N+1 \geq 2+1)$

(f) Suponga ahora que puede disponer de la memoria adicional que desee. Calcule cual sería el nuevo coste de la operación de reunión, y el número de bloques mínimos necesarios en esta situación. *Justifíquelo*. **[6 p]**

El coste mínimo sería aquel en el que pudiéramos evitar las escrituras/lecturas para guardar resultados intermedios:

- Relaciones de salida
- Escritura/lectura de cajones en la fase de construcción de éstos

En estas condiciones el coste mínimo se reduciría a:

$(R_1 \bowtie R_3)$ Coste operación: $(b_{R1} + b_{R3})$ = (3.200 + 1.600) = **4.800** bloques

$S_{1,3}$ **0** bloques (se encauza)

$S_{1,3} \bowtie R_2$ Coste operación: $(b_{S1,3} + b_{R2})$ = (0 + 800) = **800** bloques

Es decir, el coste de las operaciones es simplemente el coste de las lecturas

Coste total = 4.800 + 0 + 800 = 5.600 bloques

El número de bloques de memoria utilizados sería ahora:

- Entrada: Es suficiente con **1 bloque** (dado que la entrada y la salida no son simultáneas, veremos que lo podremos tomar de los bloques de salida)
- Operación de reunión:
  - Construcción de cajones: Para R_1 necesitamos tener en memoria de forma simultánea **todos** los bloques (b_{R1}) de esta relación: **3.200 bloques**

    Una vez acabada la tarea de construir los cajones de R_1, para R_3 se necesita, de forma simultánea con lo anterior, tener en memoria **todos** los bloques (b_{R3}) de esta relación: **1.600 bloques**
- Salida: para encauzar $S_{1,3}$ se necesita, de forma simultánea con los bloques de los cajones finales, tener en memoria **todos** los bloques ($b_{S1,3}$) de esta relación: **100 bloques**
- Para $S_{1,3} \bowtie R_2$, procederemos igual, pero ahora ya podremos reutilizar los bloques de construcción de los cajones de R_1 y de R_3, por lo que no se necesitará ninguno adicional

Bloques necesarios = 3.200 + 1.600 + 100 = 4.900

Ejercicio 15

1. ¿Se podría estimar el número de tuplas esperado en la consulta $\sigma_{MARCA=HONDA}(R)$? Si es así, determinar ese valor, si no es así, justificarlo **[2 p]**

   No se podría calcular (en general) debido a que para realizar **estimaciones** se necesita conocer ciertas estadísticas básicas sobre los datos contenidos en la tabla, previa su recolección y almacenamiento en el catálogo. Pero se podría dar una estimación si se conociese el tamaño del archivo en bloques. Conocido el número de bloques y el factor de bloques (el tamaño de los campos si que están definidos) se puede estimar cuantas tuplas habría en ese archivo ocupando todos sus bloques.

2. ¿Qué estadísticas podría tener almacenada la base de datos en el catálogo del sistema si se decide recolectarlas? ¿Cuáles serían los valores de estas estadísticas? **[3 p]**

   Según lo visto en teoría, se tendría las siguientes estadísticas básicas:

   T (R) = 9

   V (MARCA, R) = 3

   V (MODELO, R) = 7

   V (PUERTAS, R) = 2

3. Considerar de nuevo la consulta $\sigma_{MARCA=HONDA}(R)$. ¿Cuál sería el número de tuplas estimado por el optimizador de consultas utilizando las estadísticas recolectadas? **[2 p]**

$$T(\sigma_{MARCA=HONDA}(R)) = \frac{T(R)}{V(MARCA,R)} = \frac{9}{3} = 3$$

4. Considerar la consulta $\sigma_{MARCA=HONDA \wedge MODELO=SUV}(R)$. ¿Cuál sería el número de tuplas estimado por el optimizador de consultas utilizando las estadísticas? **[2 p]**

$$T(\sigma_{MARCA=HONDA \wedge MODELO=SUV}(R)) = \frac{T(R)}{V(MARCA,R) \times V(MODELO,R)} = \frac{9}{3 \times 7} = \frac{3}{7}$$

5. Considerar la consulta $\sigma_{MARCA=HONDA \wedge PUERTAS=2}(R)$. ¿Cuál sería el número de tuplas estimado por el optimizador de consultas utilizando las estadísticas? **[3 p]**

$$T(\sigma_{MARCA=HONDA \wedge PUERTAS=2}(R)) = \frac{T(R)}{V(MARCA,R) \times V(PUERTAS,R)} = \frac{9}{3 \times 2} = \frac{3}{2}$$

6. De los resultados de las consultas (4) y (5) hay una que se acerca más a la realidad. ¿Cuál es, y a qué puede ser debido? **[3 p]**

   La fórmula de la pregunta 4 es más cercana a la realidad (el error es de aprox. ½ tupla, vs. un error de 1 ½ tuplas en la fórmula de la pregunta 5)

   La razón reside en que las formulas usadas para estimar los resultados de las selecciones suponen que los atributos son independientes entre sí. Pero en el caso de MARCA y PUERTAS, existe un alto grado de correlación en los valores. De hecho, en la relación R, todos los vehículos de la marca HONDA tienen 2 puertas.

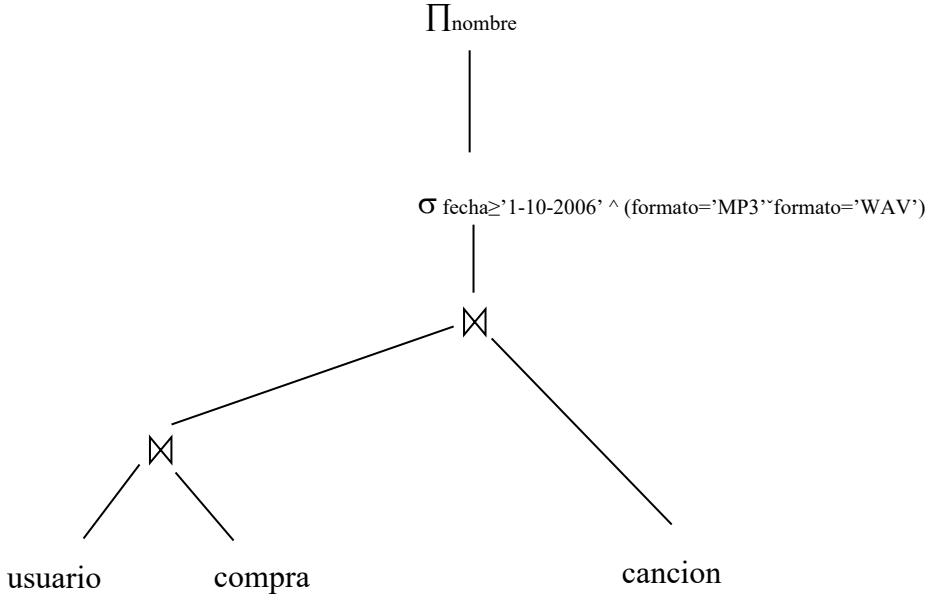
Ejercicio 16

1. Dar una expresión del algebra relacional para la consulta "Obtener el nombre de los clientes que han comprado alguna canción en formato WAV o MP3 en el cuarto trimestre del año" **[2 p]**

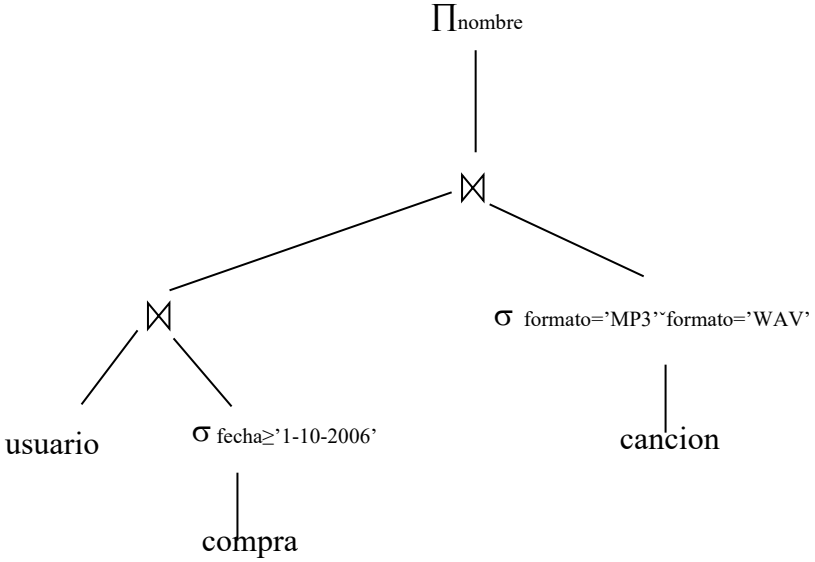
$$\Pi_{nombre}(\sigma_{fecha \geq '1\text{-}10\text{-}2006' \wedge (formato='MP3' \vee formato='WAV')}(usuario \bowtie compra \bowtie cancion))$$

2. Diseñar un plan lógico y un plan físico para la consulta anterior utilizando la optimización heurística, indicando el proceso seguido **[8 p]**

Partiendo de esa primera expresión, se puede obtener un primer árbol:



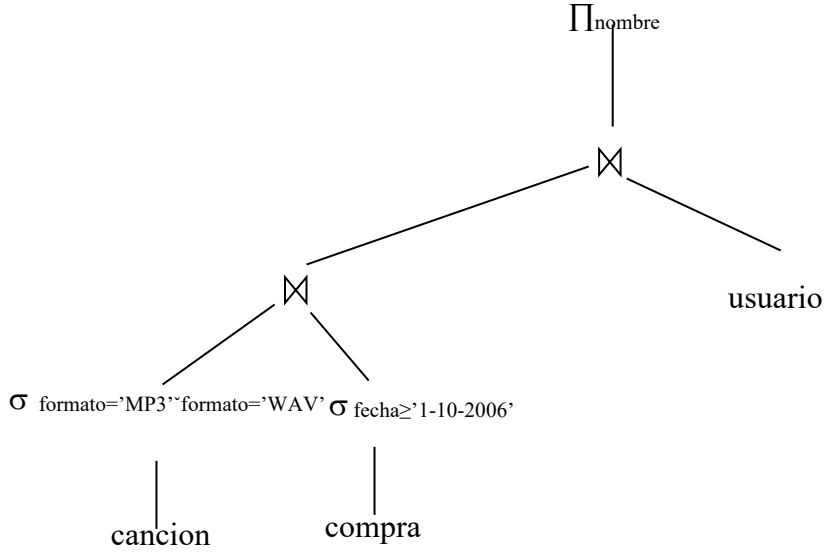
Se descomponen las selecciones y se bajan lo más posible en las ramas del árbol, quedando:



Hasta este paso puede haber dos ordenaciones diferentes del árbol dependiendo de si se hace la primera reunión con usuario y compra ó con canción y compra. Entonces hay que estimar cúales de ambos árboles producen menos tuplas en sus primeras operaciones. Para la ordenación del árbol anterior se estima el tamaño de las reuniones y de las selecciones:

T(canción)=100.000
T(σ formato='MP3'˅formato='WAV') = 2/20*100.000=10.000
T(usuario)=50.000
T(compra)=50.000
T(σ fecha≥'1-10-2006')=0.1*50.000=5.000
T(1ª reunión)=aplicando la expresión general serían 50000*5000/max{50000,5000}=5000 tuplas ya que la rama de la izquierda tiene una estadística de 50000 valores diferentes, y la rama de la derecha una estadística de min{50000,5000}
T(2ª reunión) = 5000*10000/max{5000,10000} ya que la rama de la izquierda tiene una estadística de min{20000,5000} y la rama de la derecha de min{100000,10000}

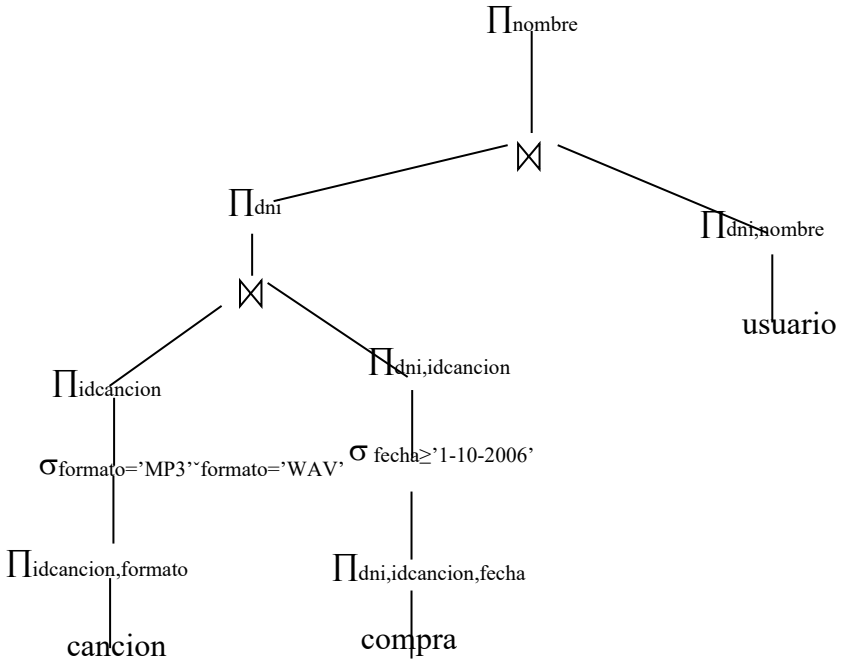
El segundo posible árbol sería:



Para esta posible ordenación, el número de tuplas estimado sería:

T(canción)=100.000
T($\sigma_{formato='MP3'\vee formato='WAV'}$) = 2/20*100.000=10.000
T(compra)=50.000
T($\sigma_{fecha\geq'1\text{-}10\text{-}2006'}$)=0.1*50.000=5.000
T(1ª reunión)=5000*10000/max{5000,10000}=5000 tuplas ya que la estadística de la izquierda tiene min{100000,5000} y al rama de la derecha de min{20000,10000}
T(usuario)=50.000
T(2ª reunión) = 5000*50000/max{5000,50000} ya que la rama de la izquierda tiene una estadística de min{50000,5000} y la rama de la derecha de 50.000

Se puede ver que ambos árboles produce el mismo número de tuplas estimadas en cada reunión, por lo que se elige como árbol definitivo el que realiza primero las selecciones en las ramas más bajas del árbol. Por último sólo falta añadir las proyecciones intermedias para eliminar los atributos que no intervienen en la consulta, quedando como árbol optimizado el siguiente:



Como no se dice en el enunciado si hay índices, ordenación de las tablas y la memoria está limitada a 12 KB para cada operación individual, se utiliza lectura secuencial y bucle anidado por bloques para las operaciones.

Calcular el coste total asociado al plan anterior bajo las condiciones comentadas anteriormente **[12 p]**

Para calcular el coste, vamos analizando cada una de las ramas su longitud de registro (L_R), su factor de bloque (f_R) y su número de bloques b_R

Rama 1

Canción: 100.000 registros
L_R=20+40+40+40=140 bytes
f_R=N registros/ Bloque = $\lfloor 1024 / 140 \rfloor$ = 7 registros / bloque
b_R=Nº total bloques = $\lceil 100.000 / 7 \rceil$ = 14.286 bloques
$\Pi_{idcancion,formato}$: L_R=60 bytes, n_R= 100.000 , f_R= 17 reg/bloque , b_R= 5883 bloques
$\sigma_{formato='MP3'\vee formato='WAV'}$: n_R= 10.000 registros , b_R= 589 bloques
$\Pi_{idcancion}$: L_R=20 bytes, n_R= 1 , f_R= 51 reg/bloque , b_R= 197 bloques

Rama 2

compra: 50.000 registros

L_R=20+20+40+40=120 bytes
N registros/ Bloque = ⌊1024 / 120⌋ = 8 registros / bloque
Nº total bloques = ⌈ 50.000 / 8 ⌉ = 6250 bloques

$\Pi_{dni,idcancion,fecha}$: L_R=80 bytes, n_R= 50.000 , f_R= 12 reg/bloque , b_R= 4167 bloques
$\sigma_{fecha \geq '1\text{-}10\text{-}2006'}$ **: produce 5.000 registros,** b_R= 417 bloques
$\Pi_{dni,idcancion}$: L_R=40 bytes, n_R= 5.000 , f_R= 25 reg/bloque , b_R= 200 bloques

Primera reunión: L_R=40 bytes, n_R= 5.000 , f_R= 25 reg/bloque , b_R= 200 bloques

Π_{dni}: L_R=20 bytes, n_R= 5.000 , f_R= 51 reg/bloque , b_R= 99 bloques

Rama 3

usuario: 50.000 registros
Lregistro=20+20+40+40=120 bytes
N registros/ Bloque = ⌊1024 / 120⌋ = 7 registros / bloque
Nº total bloques = ⌈ 50.000 / 7 ⌉ = 7143 bloques

$\Pi_{dni,nombre}$: L_R=60 bytes, n_R= 50.000 , f_R= 17 reg/bloque , b_R= 2941 bloques

Segunda reunión: L_R=60 bytes, n_R= 5.000 , f_R= 17 reg/bloque , b_R= 295 bloques

Sólo falta estimar el coste de ambas reuniones y decidir qué es lo que se encauza y se materializa.

Para las reuniones, si se encauza una de las ramas, el coste es $\lceil b_r / (M - 2) \rceil * b_s$, donde M son los bloques de Memoria disponibles, que en este caso son 12 Kb / 1 Kb = 10

Primer Join:
Si b_r = 197, C = ⌈ 197 / 10 ⌉ * 200 = 4000 bloques
Si b_r = 200, C = ⌈ 200 / 10 ⌉ * 197 = 3940 bloques, luego se encauza la rama que proporciona 200 bloques.

Segundo Join:
Si b_r = 99, C = ⌈ 99 / 10 ⌉ * 2941 = 29.410 bloques
Si b_r = 2941, C = ⌈ 2941 / 10 ⌉ * 99 = 29.205 bloques , luego se encauza la rama que proporciona 2941 bloques.

El coste total será: leer cancion + materializar 197 + coste 1er join + leer compra + materializar 99 + leer usuarios secuencialmente + coste 2º join = 14.286 + 197 + 3940 + 6250 + 99 +7143 + 29.205 = 46.864 bloques.

Coste (I/O) = 46.834 bloques

3. Indicar los cambios que se podrían introducir que podrían beneficiar el coste total del plan anterior, haciendo una **valoración aproximada** del beneficio **[3 p]**

   El coste se podría beneficiar aumentando el tamaño del bloque, aunque para un sistema implantado eso es difícil, asignar más memoria a las operaciones de reunión, y es posible que el coste se pudiese reducir si hubiese un índice sobre el atributo fecha y el atributo formato, aunque como requiera 1 acceso por cada valor la solución podría ser peor que una lectura secuencial.

Ejercicio 17

1. $\pi_{a,c} (\sigma_{b < 5} (R \bowtie S))$

   $\boldsymbol{\pi_{a,c} (\sigma_{b < 5} (R) \bowtie \sigma_{b < 5} (S))}$

2. $\pi_{b,c}(\sigma_{(a<b)\wedge(a<3)}(R \bowtie S))$

   $\boldsymbol{\pi_{b,c}(\sigma_{(a<b)\wedge(a<3)}(R) \bowtie S)}$

3. $\pi_{d}(\sigma_{c<5}(R \bowtie S \bowtie T))$

   $\boldsymbol{(\pi_{d}(R \bowtie \sigma_{c<5}(S) \bowtie T)}$

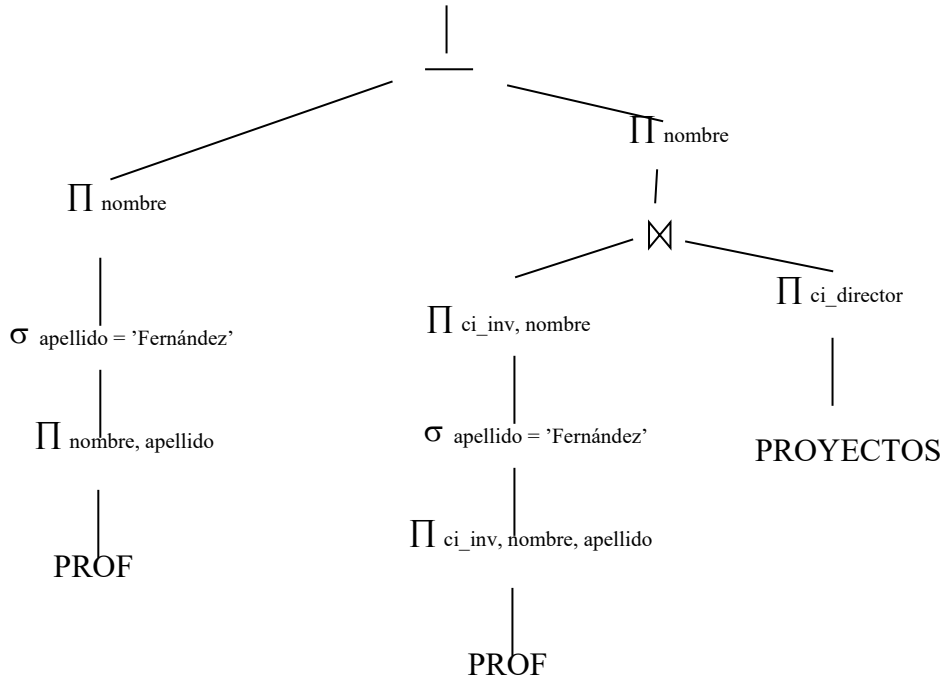
Ejercicio 18

1. Dar una expresión del álgebra relacional para la consulta: "Obtener el nombre de los profesionales que se apelliden Fernández y que **no** sean directores de proyecto".

   $\boldsymbol{\pi}_{\text{nombre}}(\sigma_{\text{apellido='Fernández'}}(\text{PROF})) - \boldsymbol{\pi}_{\text{nombre}}(\sigma_{\text{apellido='Fernández'}}(\text{PROF} \bowtie \text{PROYECTOS}))$

2. Optimizar la consulta anterior, indicando ***claramente***: (a) el plan lógico de la expresión anterior, (b) la heurística considerada, y (c) el plan físico final que se aplicaría.

   Aplicando la heurística: bajando las selecciones lo máximo posible, introduciendo proyecciones adicionales para eliminar atributos que no intervienen es suficiente, por lo que el árbol final sería:



   Como estamos en el peor caso de memoria posible (3 bloques para operaciones binarias), éstas se podrán realizar únicamente mediante el algoritmo de bucle anidado por bloques. Al no haber índices, las lecturas de las tablas se realizarán por lectura secuencial.

3. Calcular el coste asociado al plan anterior.

   Para calcular el coste, vamos analizando cada una de las ramas su longitud de registro (L_R), su factor de bloque (f_R) y su número de bloques (b_R)

Rama 1
PROF: 1.000 registros
L_R = Longitud del registro = 4 + 20 + 20 + 20 = 64 bytes
f_R = N registros/Bloque = $\lfloor$ 2.048 / 64 $\rfloor$ = 32 registros/bloque
b_R = Nº total bloques = $\lceil$ 1.000 / 32 $\rceil$ = 32 bloques
$\Pi_{\text{nombre, apellido}}$: L_R = 40 bytes, n_R = 1.000, f_R = 51 reg/bloque, b_R = 20 bloques
$\sigma_{\text{apellido='Fernández'}}$: n_R = 1.000/20 = 50 registros, b_R = 1 bloque
Π_{nombre}: L_R = 20 bytes, n_R = 50, f_R = 102 reg/bloque, b_R = 1 bloques

Rama 2
PROF: 1.000 registros
L_R = 4 + 20 + 20 + 20 = 64 bytes
f_R = $\lfloor$ 2.048 / 64 $\rfloor$ = 32 registros/bloque
b_R = $\lceil$ 1.000 / 32 $\rceil$ = 32 bloques
$\Pi_{\text{ci\_inv, nombre, apellido}}$: L_R = 64 bytes, n_R = 1.000, f_R = 46 reg/bloque, b_R = 22 bloques
$\sigma_{\text{apellido='Fernández'}}$: n_R = 1.000/20 = 50 registros, b_R = 2 bloques
$\Pi_{\text{ci\_inv, nombre}}$: L_R = 24 bytes, n_R = 50, f_R = 85 reg/bloque, b_R = 1 bloque

Rama 3
PROYECTOS: 30 registros
L_R = 4 + 4 + 20 + 20 = 48 bytes
f_R = $\lfloor$ 2048 / 48 $\rfloor$ = 42 registros/bloque
b_R = $\lceil$ 30 / 42 $\rceil$ = 1 bloque
$\Pi_{\text{ci\_director}}$: L_R = 4 bytes, n_R = 30, f_R = 512 reg/bloque, b_R = 1 bloque

Operación reunión
Número de registros = nr*ns/max{V(A,r),V(A,s)}=50*30/max{50,30}=30 ya que la estadística de la rama de la izquierda dice que hay 50 valores diferentes y en la rama de la derecha 30 valores diferentes
L_R = 4 + 4 + 20 = 28 bytes, f_R = 42 reg/bloque, b_R = 1 bloques

Proyección final
Π_{nombre}: b_R = 1 bloques.

La salida de la operación de diferencia genera 1 bloque, ya que ambas entradas ocupan un único bloque y la operación diferencia (en el mejor de los casos) lo que hace es eliminar las tuplas que coincidan.

Como las operaciones binarias ocupan cada una de sus entradas 1 bloque, no existe coste asociado a las operaciones binarias (reunión y diferencia). Entonces el coste total será:

leer PROF + leer PROF + leer PROYECTOS + grabar salida = 32 + 32 + 1 + 1 = 66 bloques

Coste =	**66**	**Bloques**

4. Justificar que cambios serían convenientes introducir para reducir el coste asociado al plan anterior, y **calcular** dicho ahorro.

Las diferentes opciones serían:

- El tema de incrementar memoria para las operaciones binarias no introduce ninguna mejora en absoluto... ¡entre otras cosas porque su coste es ya 0!
- Introducir índices en el atributo `apellido` de PROF no supone ventaja alguna, porque hay estimada una cantidad significativa de valores que son Fernández (que al no ser clave puede situar cada registro en un bloque de entrada diferente) y porque además la relación ocupa pocos bloques en disco.
- Lo que si se podría hacer sería utilizar la lectura de PROF y guardarla en memoria para alimentar las ramas 1 y 2 del árbol, con lo que se ahorrarían 32 bloques de la segunda lectura, quedando el coste final en sólo 34 bloques (casi la mitad).

Ahorro = 32 **Bloques**

Ejercicio 19

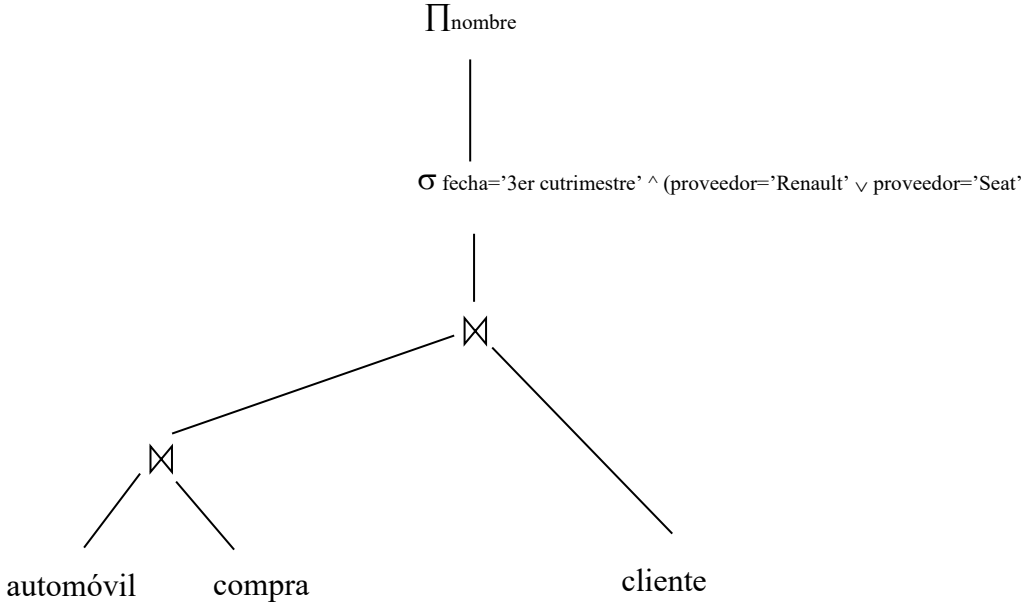
1. Considerar la consulta "Obtener el nombre de los clientes que han comprado un automóvil del proveedor Renault o SEAT en el tercer cuatrimestre del año". Expresarla en álgebra relacional.

$$\Pi_{\text{nombre}}\left(\sigma_{\text{fecha='3er trimestre'} \wedge (\text{proveedor='Renault'} \vee \text{proveedor='Seat'})}(\text{automóvil} \bowtie \text{compra} \bowtie \text{cliente})\right)$$

Esta consulta se basa en realizar dos reuniones para obtener la información necesaria en una tupla (automóvil que ha comprado un cliente determinado) y una selección para descartar las tuplas que no cumplen las condiciones.

2. (a) Describir inicialmente un árbol de consulta equivalente. (b) Proponer los criterios particulares de optimización **para este árbol**, siendo especialmente claros en los algoritmos y consideraciones aplicadas. (c) Describir finalmente el árbol optimizado resultante.

Para la consulta anterior se puede obtener este primer árbol:



Para optimizar este árbol, se pueden aplicar estas consideraciones:

- Descomponer la selección compleja en selecciones sencillas.
- Desplazar las selecciones sencillas cuanto más abajo en el árbol a la rama correspondiente.
- Ordenar las reuniones, haciendo primero las más restrictivas. Parece que es mejor primero aplicar las selecciones sobre las tablas automóvil y compra, después realizar la reunión y lo que salga, la reunión con cliente.
- Introducir proyecciones adicionales, teniendo especial cuidado con la utilización de los índices o no (Hay índice en una condición de selección y otro índice en una tabla que se reune).
- Decidir si se puede encauzar algunas operaciones.

Hasta este paso puede haber dos ordenaciones diferentes del árbol dependiendo de si se hace la primera reunión con cliente y compra ó con automóvil y compra. Entonces hay que estimar cúales de ambos árboles producen menos tuplas en sus primeras operaciones. Para la ordenación del árbol anterior se estima el tamaño de las reuniones y de las selecciones:

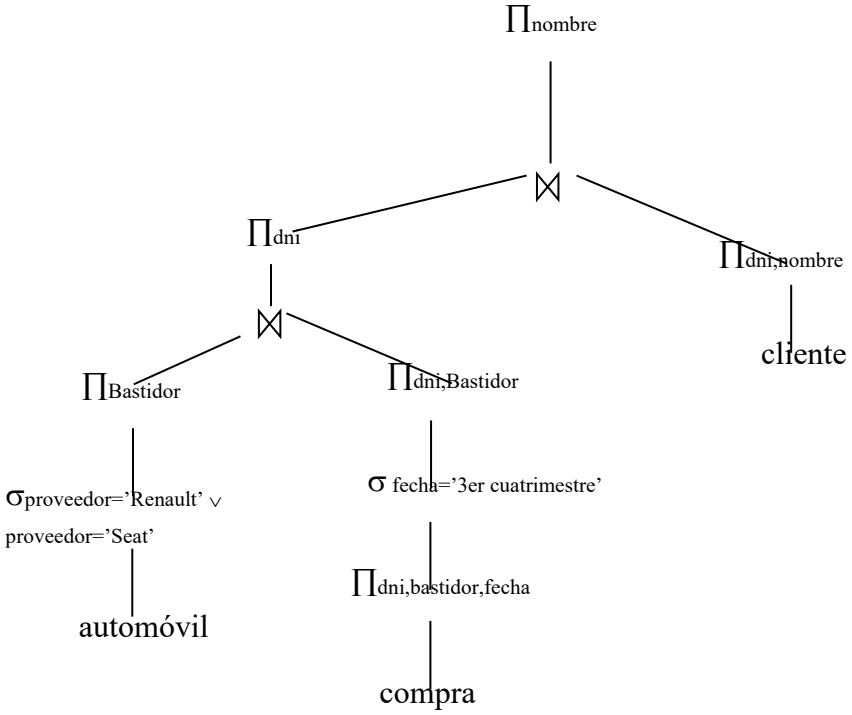
T(automóvil)=100.000 reg.
T($\sigma_{\text{prvveedor='Renault'} \vee \text{proveedor='Seat'}}$) = 2/500*100.000=400 reg.
T(cliente)=50.000
T(compra)=50.000
T($\sigma_{\text{fecha='3er cuatrimestre'}}$)=0.2*50.000=10.000
T(1ª reunión automóvil,compra)=nr*ns/max{V(A,r),V(A,s)}= 400*10.000/max{400,10000}=400 registros, ya que la estadística de la rama de la izquierda ha cambiado a 400 y la de la derecha a 10000.
T(2ª reunión) = 400*50000/max{400,50000}=400 registros, que son los números de valores diferentes de cada rama.

El segundo posible árbol sería, realizar primero la reunión de la rama compra con cliente y el resultado con la rama de automóvil. Para este caso:
T(cliente)=50.000
T(compra)=50.000
T($\sigma_{\text{fecha='3er cuatrimestre'}}$)=0.2*50.000=10.000

T(automóvil)=100.000
T($\sigma_{\text{prvveedor='Renault'} \vee \text{proveedor='Seat'}}$) = 2/500*100.000=400
T(1ª reunión cliente,compra)=50000*10000/max{50000,10000}=10000 reg.
T(2ª reunión con automovil) = 10000*400/max{10000,400}=400 reg.

Luego se puede ver que el primer árbol produce menos tuplas en el primer join. Además parece lógico realizar primero ambas selecciones para mezclar luego la rama que no tiene selecciones.

Introduciendo operaciones intermedias, salvo antes de las operaciones de lectura de las tablas que disponen índices hasta que se verifique si es mejor utilizarlos o no, el árbol queda:



Los algoritmos que se van a utilizar:

- Para la selección de proveedor:

Lectura secuencial: Longitud de registro de Automóvil: Lr=3*40+20=140 bytes. Factor de bloques: fr= $\lfloor 1024 / 140 \rfloor$= 7 reg/bloque. Numero de bloques: $b_R = \lceil 100.000 / 7 \rceil$ = 14.286 bloques

Lectura utilizando el índice: 2*(4+200)=408 bloques peor caso. Luego es mejor usar el índice y por lo tanto no es necesario introducir una proyección antes de la lectura de la tabla automóvil.

Para la rama de compra, una lectura secuencial. En la selección de fecha no hay índice disponible.

Lectura secuencial: Longitud de registro de Automóvil: Lr=2*40+2*20=120 bytes. Factor de bloques: fr= $\lfloor 1024 / 120 \rfloor$= 8 reg/bloque. Numero de bloques: $b_R = \lceil 50.000 / 8 \rceil$ = 6250 bloques

Para la primera reunión. No hay índice disponible, sólo hay 5 bloques de memoria disponibles, no hay ordenación de las relaciones de entrada y no se sabe si se puede hacer un hash join. Luego se utiliza un bucle anidado por bloques, donde una de las ramas se puede encauzar. Generalmente la más grande y la otra se materializa.

El coste de este primer join sería:

- La primera rama (Bastidor) entraría con: fr= $\lfloor 1024 / 20 \rfloor$= 51 reg/bloque. Numero de bloques: $b_R = \lceil 400 / 51 \rceil$ = 8 bloques
- La segunda rama (Bastidor,DNI) entraría con: fr= $\lfloor 1024 / 40 \rfloor$= 25 reg/bloque. Numero de bloques: $b_R = \lceil 10000 / 25 \rceil$ = 400 bloques

Si se encauza una de las ramas, es $\lceil b_r / (M - 2) \rceil * b_s$, donde M son los bloques de Memoria disponibles, que en este caso son 5.

Si $b_r = 8$, C = $\lceil 8 / 3 \rceil * 400 = 1200$ bloques

Si $b_r = 400$, C = $\lceil 400 / 3 \rceil * 8 = 1072$ bloques, luego se encauzaría la rama de compra

Para el segundo join. Se puede utilizar un bucle anidado por bloques o utilizar reunión con índice. La tabla cliente tiene un índice sobre la clave primaria DNI.

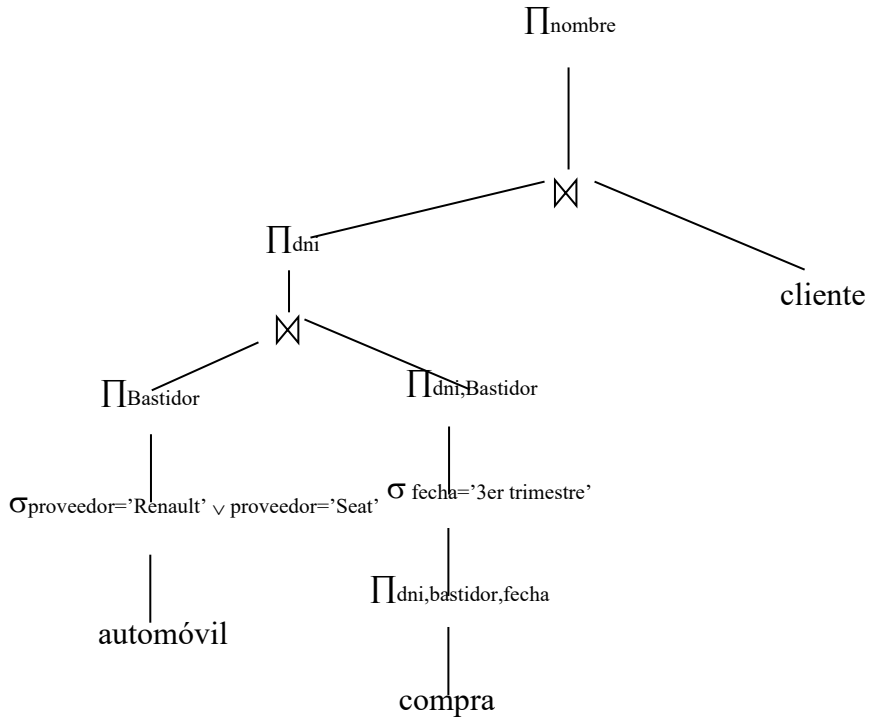
- Si se utiliza bucle anidado por bloques: La entrada de la primera rama es la salida del 1er join con la proyección, luego habría 400 tuplas, que ocuparían: fr= $\lfloor 1024 / 20 \rfloor = 51$ reg/bloque. Numero de bloques: $b_R = \lceil 400 / 51 \rceil = 8$ bloques. La otra rama, sería la lectura de cliente haciendo una proyección de los atributos nombre y dni. Luego ocuparía: fr= $\lfloor 1024 / 60 \rfloor = 17$ reg/bloque. Numero de bloques: $b_R = \lceil 50.000 / 17 \rceil = 2942$ bloques

Si se encauza una de las ramas, es $\lceil b_r / (M - 2) \rceil * b_s$, donde M son los bloques de Memoria disponibles, que en este caso son 5.

Si $b_r = 8$, C = $\lceil 8 / 3 \rceil * 2942 = 8826$ bloques

Si $b_r = 2942$, C = $\lceil 2942 / 3 \rceil * 8 = 7848$ bloques, luego se encauzaría la rama de automóvil y se materializaría la salida del primer join.

Si se utiliza el índice, se podría encauzar la entrada de la salida del primer join y buscar en el índice. Esa búsqueda vale (3+1) para el peor caso. Luego el coste del join sería: nr*Coste=400*4=1600, menor que el bucle anidado. Por lo tanto el árbol final con los algoritmos serían:



Con lectura utilizando el índice para la tabla automóvil, materializando la salida de la proyección de bastidor, lectura secuencial de compra, encauzamiento hasta la segunda entrada del primer join,

encauzamiento de la salida del primer join y entrada del segundo join y utilización del índice para el join.

3. Calcular los costes asociados al árbol optimizado anterior.

Según las explicaciones anteriores, el coste del árbol optimizado sería:

Coste: lectura de los datos de automóvil con el indice + lectura de la tabla compra + materializar salida de proyección de bastidor + coste del primer join + coste segundo join = 2*(4+200)+ 6250 + 8 + 1072 + 400*4 = 9338 bloques.

Coste = 9338 Bloques

4. Indicar las mejoras que se podrían realizar para ahorrar coste. En caso de haberlas, calcular ese ahorro. Si no hay mejoras, especificarlo.

En este caso se podría reducir el coste haciendo:

- Incrementando la memoria para reducir el coste del primer join. Si se tuviesen 8 +2 bloques, la entrada del primer join cabría en memoria y el coste del join sería cero: 8 + 1072
- Un índice en el atributo fecha podría reducir el coste. En este caso parece que no sea probable ya que hay pocos valores diferentes (sólo 3 los tres trimestres y utilizar el índice podría ser más costoso que leer secuencialmente la tabla compra).
- Si la tabla automóvil hubiese estado ordenada por proveedores, el factor de bloques de la tabla es 7 y entonces se podría recuperar los 400 registros con las siguientes búsquedas: 2*(4+29) = 66, donde esos 29 bloques serían los necesarios para recuperar secuencialmente los 200 registros de un proveedor. Luego el ahorro sería: 342 bloques

En estas condiciones, el ahorro sería: 8+1072+342=1404 bloques

Ahorro = 1404 Bloques

Ejercicio 20

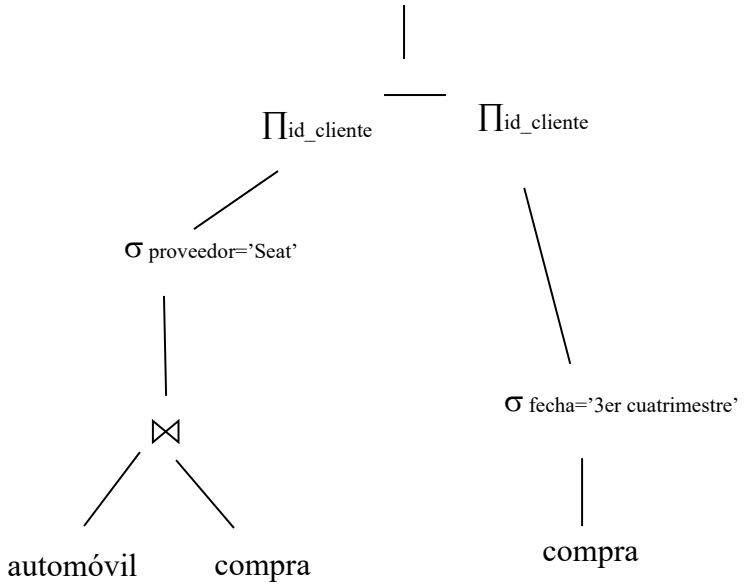
a) Considerar la consulta "Obtener el identificador de todos los clientes que han comprado un automóvil del proveedor SEAT y que además no hayan comprado ningún otro automóvil en el tercer cuatrimestre". No utilizar el operador ≠.

Una consulta expresada en el álgebra relacional que cumple con esta búsqueda sería:

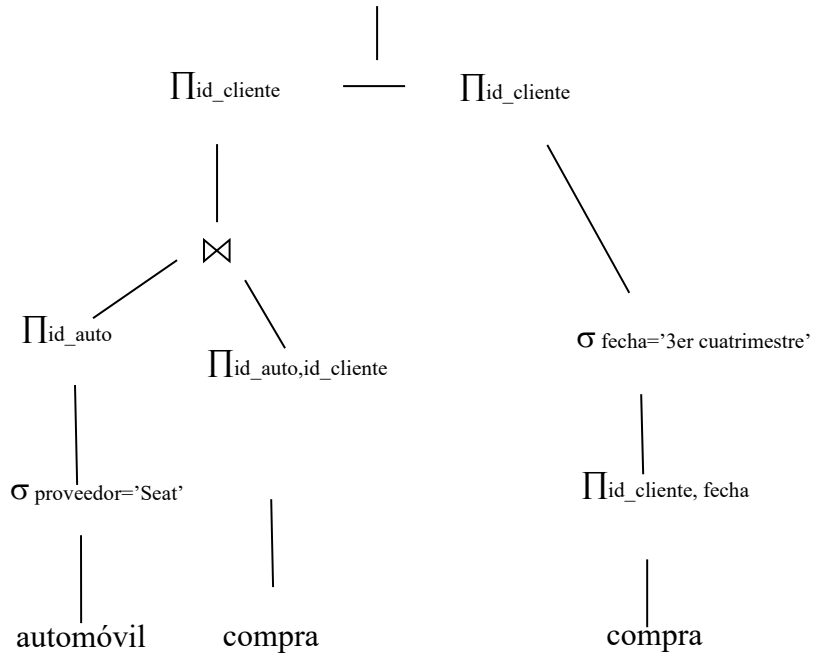
$$\Pi_{id\_cliente}(\sigma_{proveedor='Seat')}(automóvil \bowtie compra))) - \Pi_{id\_cliente}(\sigma_{fecha='3er\ cuatrimestre'}(compra))$$

b) (1) Describir inicialmente un árbol de consulta equivalente. (2) Proponer los criterios particulares de optimización **para este árbol**, siendo especialmente claros en los algoritmos y consideraciones aplicadas. (3) Describir finalmente el árbol optimizado resultante.

Inicialmente el árbol podría ser:



Lo que se podría hacer sería añadir proyecciones para eliminar los atributos que no se necesitan, salvo quizás en la rama automóvil ya que parece ser que la búsqueda binaria sobre el campo proveedor será más eficiente que la lectura secuencial. Y también descender las selecciones cuanto más abajo mejor Con eso, nos queda:



Determinamos el número de bloques y tuplas de cada una de las operaciones para ver que algoritmos podemos elegir.

Longitud de registro de Automóvil: Lr=4*40+20=180 bytes. Factor de bloques: fr= $\lfloor 1024 / 180 \rfloor$= 5 reg/bloque. Numero de bloques: $b_R = \lceil 120.000 / 5 \rceil$ = 24.000 bloques
Longitud de registro de Compra: Lr=2*40+2*20=120 bytes. Factor de bloques: fr= $\lfloor 1024 / 120 \rfloor$= 8 reg/bloque. Numero de bloques: $b_R = \lceil 90.000 / 8 \rceil$ = 11.250 bloques
$\sigma_{proveedor='Seat'}$, hay 500 proveedores, luego la selección producirá: 120.000*1/500= 240 registros.
$\sigma_{fecha='3er\ cuatrimestre'}$, hay un 30% en el tercer trimestre, luego serán: 0.3*90.000= 27.000 registros.
Para la operación de reunión, el número de tuplas a la salida será:
nr*ns/max{V(A,r),V(A,s)}=240*90000/max{240,90000}=240 reg porque la estadística de la rama de la izquierda ha cambiado a 240 y la de la derecha sigue siendo 90.000 vehículos diferentes.

Los algoritmos que se utilizarán serán:

- lectura secuencial en la tabla compra para la condición de selección en fecha, no hay ordenación ni índice.
- Para la tabla automóvil, si se realiza una búsqueda binaria: = $\lceil \log_2 (24.000) \rceil + \lceil 240/5 \rceil - 1 = 15 + 48 - 1 = 62$ bloques, mucho menor que la lectura secuencial de 24.000 bloques.
- Para la operación de reunión:
  - Si se utiliza bucle anidado, como estamos en la condición de mínima memoria, M=3. El número de bloques para cada entrada sería:

$\prod_{id\_auto}$, Lr=20 bytes. Factor de bloques: fr= $\lfloor 1024 / 20 \rfloor$= 51 reg/bloque. Numero de bloques: $b_R = \lceil 240 / 51 \rceil$ = 5 bloques
$\prod_{id\_auto,id\_cliente}$, Lr=40 bytes. Factor de bloques: fr= $\lfloor 1024 / 40 \rfloor$= 25 reg/bloque. Numero de bloques: $b_R = \lceil 90.000 / 25 \rceil$ = 3600 bloques

Luego encauzando una rama (mayor) y materializando otra (menor), coste br*bs=5*3600=18.000 bloques + 5 de materializar.

- Si se utiliza hash, el coste sería: 3(br+bs) materializando las dos entradas = 3(5+3600)=10815. Si no se graba en disco la entrada que se va a particionar, el coste sería: 2(br+bs)=2(5+3600)=7210 bloques.

Con la operación diferencia ocurriría lo mismo:
$\prod_{id\_cliente}$, Lr=20 bytes. Factor de bloques: fr= $\lfloor 1024 / 20 \rfloor$= 51 reg/bloque. Numero de bloques: $b_R = \lceil 27.000 / 51 \rceil$ = 530 bloques, de la rama de la derecha
$\prod_{id\_cliente}$, Lr=20 bytes. Factor de bloques: fr= $\lfloor 1024 / 20 \rfloor$= 51 reg/bloque. Numero de bloques: $b_R = \lceil 240 / 51 \rceil$ = 5 bloques, de la rama de la derecha, ya que el join produce como máximo 240 tuplas.

Esta claro que el hash join va a funcionar mejor para la operación diferencia ya que las relaciones son más grandes.

- Con bucle anidado: la resta no es conmutativa, luego se debe materializar la rama de la derecha siempre, y encauzar la rama de la izquierda siempre, coste br*bs=5*530=2650 bloques + 530 de materializar.
- Si se utiliza hash, el coste sería: 3(br+bs) materializando las dos entradas = 3(530+5)=1605. Si no se graba en disco la entrada que se va a particionar, el coste sería: 2(br+bs)=2(530+5)=1070 bloques.

Luego el hash join funcionaría mejor.

Lugo los algoritmos que se utilizarían sería:
Hash join para las dos operaciones binarias , lectura secuencial para la tabla compra y búsqueda binaria para la tabla automóvil.

c) Calcular los costes asociados al árbol optimizado anterior.

El coste saciado al árbol anterior sería. Sabiendo que las operaciones unarias las puedo encauzar y las entradas de las operaciones binarias también, (sólo me evito una lectura y escritura, pero tengo que grabar las particiones y volverlas a leer), el coste sería:

Coste: búsqueda secuencial automóvil + lectura secuencial compra + lectura secuencial compra + coste operación hash de reunión + coste hash operación diferencia = 62 + 11250 + 11250 + 2(5 + 3600) + 2(530+5)=30842 bloques

Coste = 30842 bloques

d) Determinar el número mínimo de bloques de memoria necesarios para poder reducir al máximo el coste estimado del árbol.

Para reducir al máximo el coste, implicaría que las operaciones de join y la resta no deberían de tener coste, lo que implica que sería necesario particionar y guardar una de las relaciones en memoria. Luego parece ser que 530 bloques serían necesarios para guardar una de las entradas en memoria para la resta (rama de la derecha), y para el primer join se necesitaría guardar la pequeña en ram 5 bloques. También la operación hash necesitaría 100 bloques + 1 de entrada + 1 de salida para poder funcionar