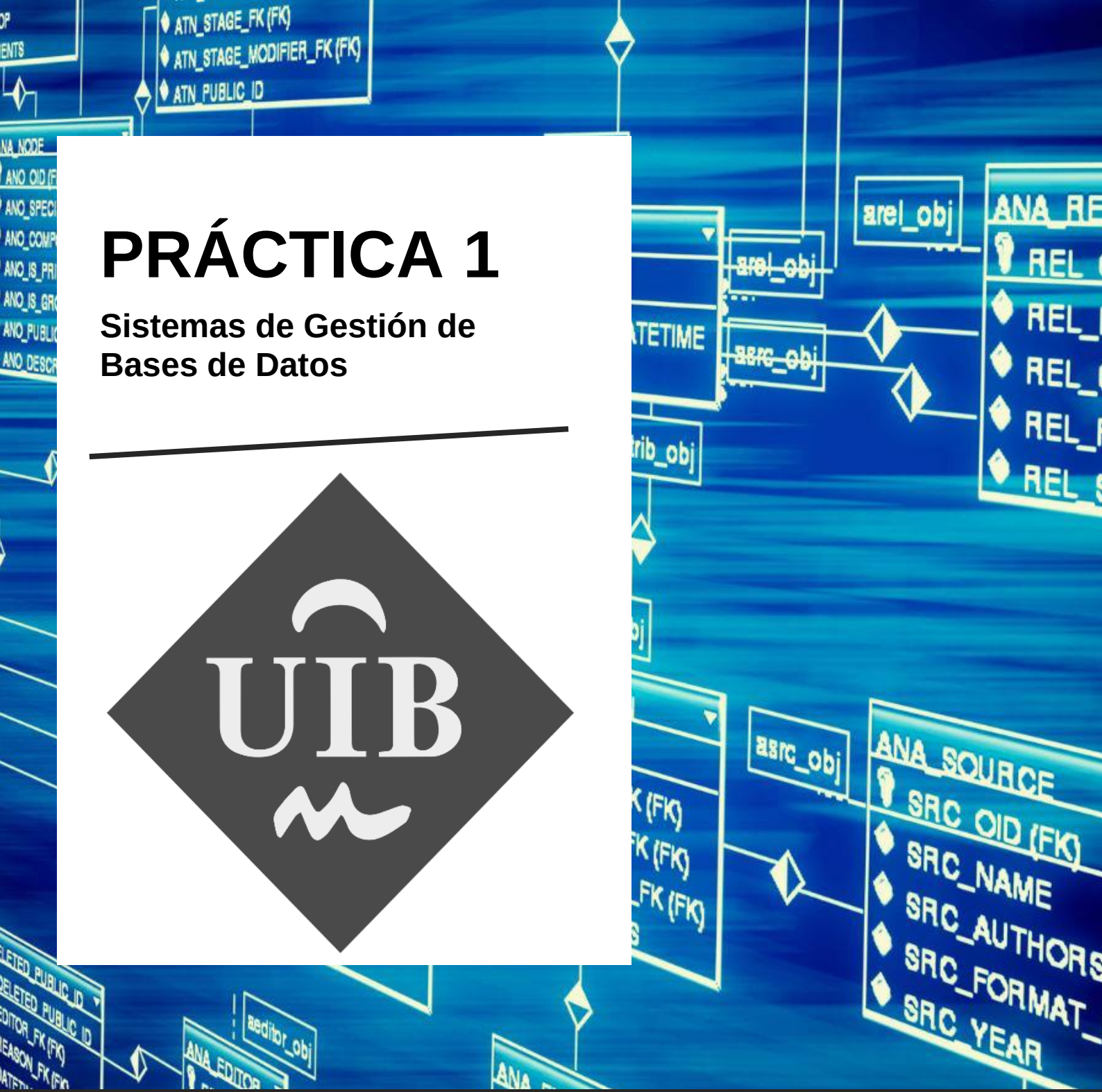


PRÁCTICA 1

Sistemas de Gestión de Bases de Datos

18 OCTUBRE

PRÁCTICA 1 – EJERCICIO 9
Mario Ventura Burgos 43223476-J
Grado en Ingeniería Informática (GIN 3)
CURSO 2023-2024

1. CÓDIGO Y RESULTADO

Abriremos la herramienta de Query Tool en pgAdmin (aunque también se puede hacer en sql shell) y añadimos "EXPLAIN" al inicio de la consulta 3 del examen cuya solución se encuentra en el archivo mvb135_4.sql y obtenemos lo siguiente:

INPUT:

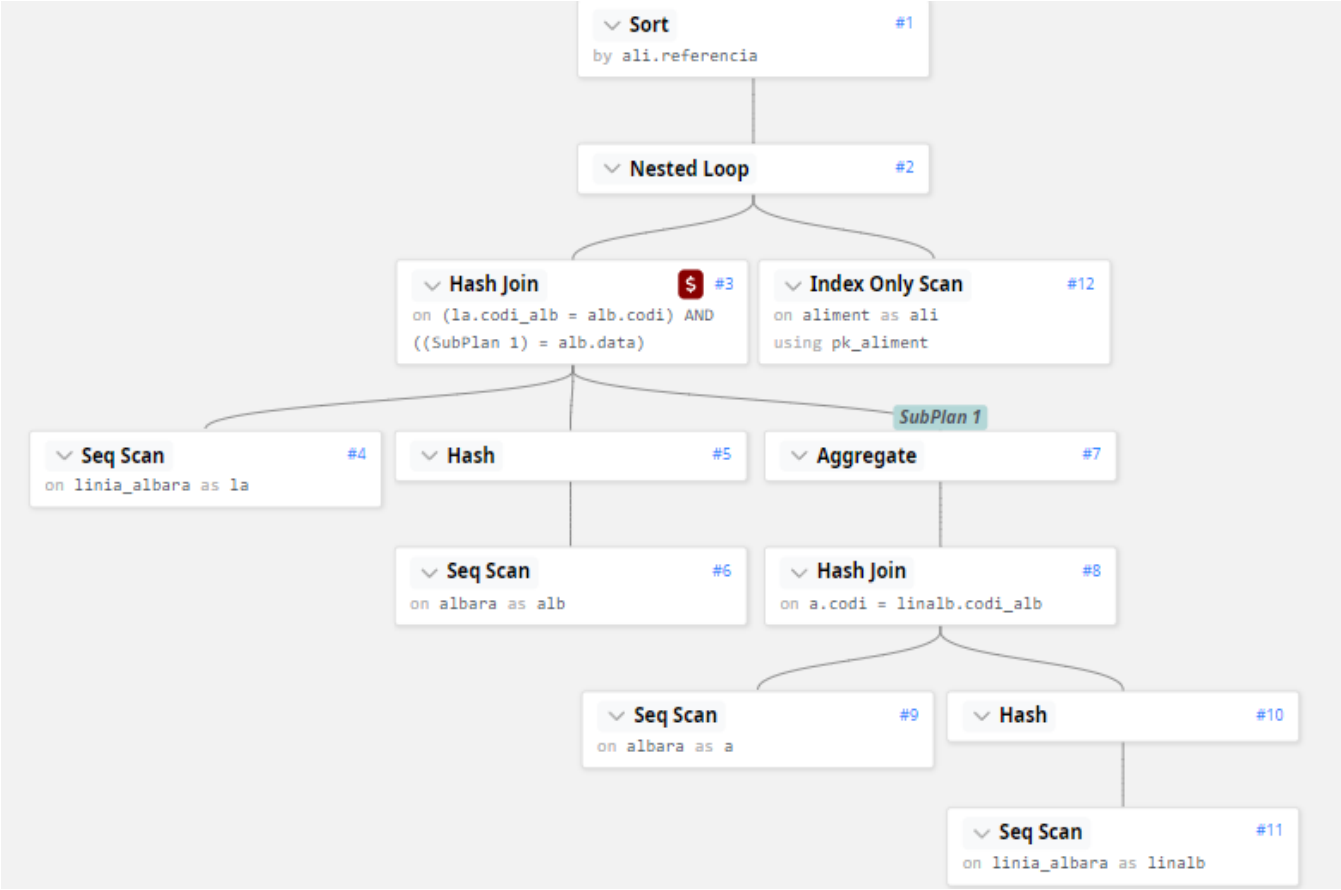
```
SELECT ALI.referencia, LA.preu
FROM Aliment ALI
    JOIN linia_albara LA
    ON LA.referencia = ALI.referencia
        JOIN Albara ALB
        ON ALB.codi = LA.codi_alb
WHERE ALB.data = (
        SELECT MAX(A.data)
        FROM ALBARA A
            JOIN linia_albara LINALB
            ON LINALB.codi_alb = A.codi
        WHERE LINALB.referencia = LA.referencia
    )
AND LA.preu IS NOT NULL
ORDER BY ALI.referencia; -- No se pide explícitamente pero el resultado será más legible
```

OUTPUT:

```
Sort  (cost=1904.29..1904.30 rows=4 width=48)
  Sort Key: ali.referencia
  ->  Nested Loop  (cost=32.15..1904.25 rows=4 width=48)
      ->  Hash Join  (cost=32.00..1903.37 rows=4 width=48)
          Hash Cond: ((la.codi_alb = alb.codi) AND ((SubPlan 1) = alb.data))
          ->  Seq Scan on linia_albara la  (cost=0.00..18.80 rows=876 width=60)
              Filter: (preu IS NOT NULL)
          ->  Hash  (cost=18.80..18.80 rows=880 width=16)
              ->  Seq Scan on albara alb  (cost=0.00..18.80 rows=880 width=16)
          SubPlan 1
            ->  Aggregate  (cost=42.18..42.19 rows=1 width=4)
                ->  Hash Join  (cost=21.05..42.17 rows=4 width=4)
                    Hash Cond: (a.codi = linalb.codi_alb)
                    ->  Seq Scan on albara a  (cost=0.00..18.80 rows=880 width=16)
                    ->  Hash  (cost=21.00..21.00 rows=4 width=12)
                        ->  Seq Scan on linia_albara linalb  (cost=0.00..21.00 rows=4 width=12)
                            Filter: ((referencia)::text = (la.referencia)::text)
      ->  Index Only Scan using pk_aliment on aliment ali  (cost=0.15..0.22 rows=1 width=44)
          Index Cond: (referencia = (la.referencia)::text)
```

2. ANÁLISIS DEL QUERY PLAN

Al igual que en el caso anterior, el plan de ejecución está organizado en una estructura jerárquica que se muestra mediante tabulaciones en el output. Sin embargo, se facilita la comprensión y se hace más legible si se representa en forma de árbol. El árbol obtenido para este plan de ejecución es el siguiente:



Como se puede apreciar, el árbol que representa el plan de ejecución cuenta con 12 nodos, representados por pgAdmin (o, en su defecto, sql shell) mediante las 19 líneas obtenidas como output. Analicemos el resultado nodo por nodo:

1. **SORT** → Las líneas 1 y 2 del plan de ejecución. El Query Plan comienza con una ordenación, aunque realmente esto será lo último que se haga en la consulta. Este nodo representa la operación de ordenación que se encarga de ordenar los resultados en función de la columna 'referencia' de la tabla Aliment (línea 2: **Sort Key: ali.referencia**).
   Cabe aclarar que realmente esta ordenación **no se pide explícitamente** en el enunciado de la consulta. Se añadió porque al probar la consulta en una base de datos de prueba se vio que los resultados carecían de cualquier tipo de orden, y el resultado se hacía más legible si al menos una de las dos columnas seleccionadas se ordenaba bajo algún criterio. Por tanto, teniendo en cuenta su coste (se estima un coste de **cost=1904.29..1904.30**) se podría eliminar esta

ordenación y la consulta funcionaría de igual manera en caso de ejecutarse, aunque no quedase ordenada.
Se espera obtener una cantidad de **rows=4** filas con una anchura de **width=48** bytes.

2. **NESTED LOOP** → La línea 3 del Query Plan. Este bucle anidado representa la operación de JOIN entre las tablas Aliment, Linia_Albara y Albara (realmente representa la unión del resultado de las operaciones que preceden a este Loop, que son el hash Join (nodo 3) del escaneo secuencial de las tablas linia_albara (nodo 4) y la tabla albara (nodo 6) y el escaneo de índice en la tabla aliment (nodo 12)), y precede a la ordenación del nodo 1.
Se espera un coste estimado, numero de filas y anchura de: **cost=32.15..1904.25 rows=4 width=48**

3. **HASH JOIN (nodo 3)** → Las líneas 4 y 5. Representa la unión entre las tablas Linia_Albara y Albara mediante una operación de unión de Hash. La condición de unión de estas dos tablas es: **Hash Cond: ((la.codi_alb = alb.codi) AND ((SubPlan 1) = alb.data))**, que es la línea 5 del plan de ejecución obtenido. Para esta operación se espera obtener un coste de **cost=32.00..1903.37** unidades, y una cantidad de filas y anchura de **rows=4 width=48** filas y bytes respectivamente.

4. **SEQ SCAN (nodo #4)** → Las líneas 6 y 7. Este nodo representa un escaneo secuencial de la tabla Linia_albara (LA), utilizado para obtener todas las filas de esta tabla que cumplan con la condición establecida en la línea 7: el precio no puede ser nulo. Los valores de coste y número de filas / anchura estimados son: **cost=0.00..18.80 rows=876 width=60**

5. **HASH (nodo #5)** → La línea 8 del plan de ejecución representa la utilización de una operación de Hash para la unión de la tabla Linia_Albara (explorada secuencialmente en el nodo 4) con el resultado de la operación que precede a esta operación de Hash (nodo 6 que se explicará a continuación). Se estima un coste de **cost=18.80..18.80** unidades, una cantidad de **rows=880** filas estimadas y una anchura en bytes de **width=16.**

6. **SEQ SCAN (nodo #6)** → La novena línea del Query plan es otro escaneo secuencial, pero en este caso, en la tabla Albara (ALB). El resultado de este escaneo será usado por la operación de hash descrita en el nodo anterior.
El coste de estas operaciones es: **cost=0.00..18.80** y respecto a las filas se espera obtener **880** con una anchura de **16** bytes.

- **El inicio del sub-plan de ejecución *SubPlan 1* viene marcado por la línea 10 del plan de ejecución.**
- **Todos los nodos de ahora en adelante (excepto el nodo 12) y las operaciones que estos representan pertenecen a la sub-consulta *SubPlan1.***

7. **AGGREGATE** → La línea número 11 del plan de ejecución representa una agregación de los resultados de la subconsulta. Forma parte de la subconsulta anidada y, por tanto, el inicio del sub-plan de ejecución para la consulta en la que se busca, en la tabla Albara, el valor máximo para A.data (uso de la función MAX). El coste de esta operación es de **cost=42.18..42.19**, se espera obtener **1** fila con una anchura de **4** bytes.

8. **HASH JOIN (nodo #8)** → Las líneas 12 y 13 representan una operación de JOIN mediante Hast, con un coste de **cost=21.05..42.17** unidades. Se espera obtener **rows=4** filas con **width=4** bytes de anchura.
   Este nodo realiza una unión hash entre las tablas Albara (A) y Linia_Albara (LINALB). La condición de unión de este hash (Hash cond en la línea 13) consiste en la igualdad de las columnas A.codi y LINALB.codi_alb.

9. **SEQ SCAN (nodo #9)** → También pertenece a la subconsulta. Es la línea 14 del Query Plan y representa un escaneo secuencial en la tabla Albara. Esto tiene un coste de **cost=0.00..18.80.**
   Se estima una cantidad de filas y anchura en bytes de: **rows=880 width=16**

10. **HASH (nodo #10)** → La decimoquinta línea del plan de ejecución vuelve a ser, al igual que en el nodo 5, una operación de Hash. Esta vez la operación se aplica con el fin de poder unir la tabla Albara, escaneada previamente en el nodo 9, con la tabla Linia_Albara (sobre esta última aún se tiene que hacer un escaneo secuencial, que se hará en el nodo 11). El coste es de **21** unidades, y se esperan **4** filas con una anchura de **12** bytes.

11. **SEQ SCAN (nodo #11)** → Las líneas 16 y 17. Este nodo representa el escaneo secuencial sobre la tabla Linia_Albara (LINALB). En este escaneo se aplica un filtro en la columna referencia. Este filtro viene especificado en la línea 17 del plan de ejecución y es el siguiente: **Filter: ((referencia)::text = (la.referencia)::text)**, que es el filtro que viene de la expresión LINALB.referencia = LA.referencia
    El coste de esta operación es de **21** unidades, y se estiman **4** filas con una anchura de **12** bytes

12. **INDEX ONLY SCAN** → Por último, las líneas 18 y 19 de este plan de ejecución. Estas ya no pertenecen a la subconsulta y representan en el nodo 12 un escaneo utilizando un índice en la tabla Aliment. La condición de índice viene especificada en la línea 19 del plan de ejecución, y se basa en la igualdad de la columna 'referencia' en Aliment (ALI) con la columna 'referencia' en Linia_Albara (LA).
    Dado que esta operación está dentro del bucle anidado representado en el nodo 2, las condiciones de este Loop también se tienen en cuenta.
    El coste es realmente muy pequeño, tan solo **0,22** unidades de coste, y se estima que se obtendrá **1** fila con **44** bytes de anchura.