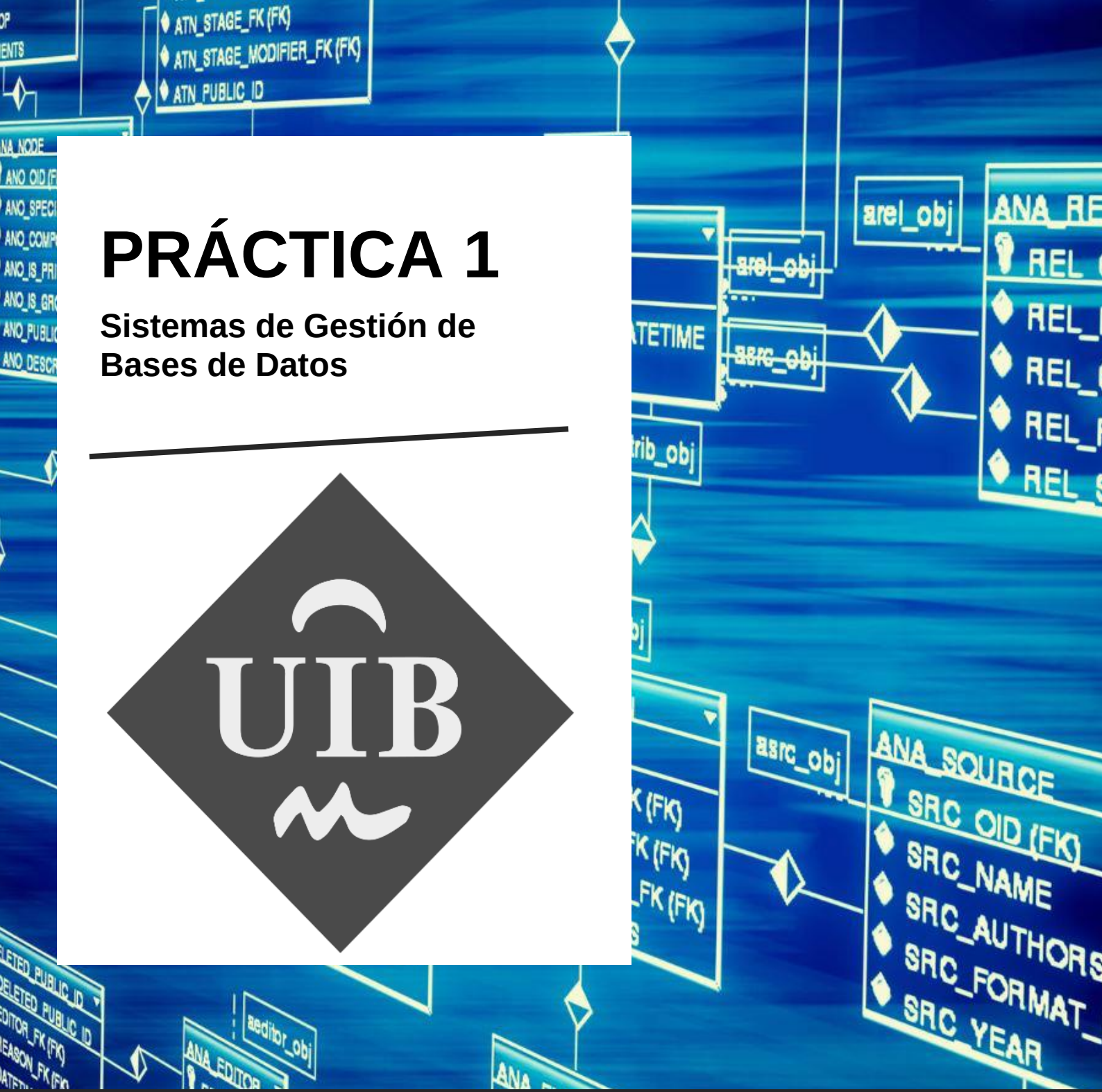


PRÁCTICA 1

Sistemas de Gestión de Bases de Datos

17 OCTUBRE

PRÁCTICA 1 – EJERCICIO 7
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 1 del examen cuya solución se encuentra en el archivo mvb135_2.sql y obtenemos lo siguiente:

INPUT:

```
EXPLAIN SELECT ALI.referencia, ALI.nom
FROM Aliment ALI
    JOIN Linia_ALbara LA
    ON LA.referencia = ALI.referencia
        JOIN Albara ALB
        ON ALB.codi = LA.codi_alb
            JOIN proveidor PRO
            ON PRO.nif = ALB.nif_pro
            AND PRO.nom = 'UIBFruita'
ORDER BY ALI.referencia ASC;
```

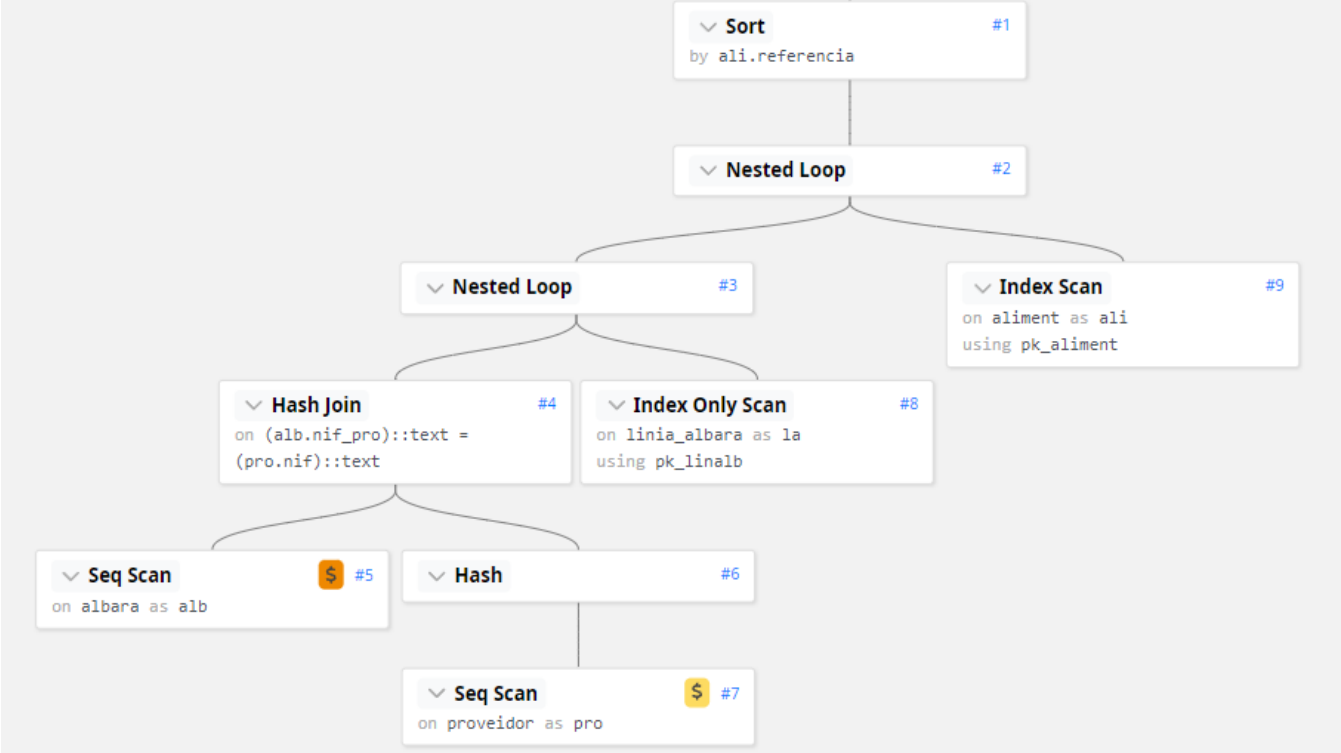
OUTPUT:

```
"Sort  (cost=38.71..38.72 rows=4 width=162)"
"  Sort Key: ali.referencia"
"  ->  Nested Loop  (cost=15.70..38.67 rows=4 width=162)"
"        ->  Nested Loop  (cost=15.55..37.79 rows=4 width=44)"
"              ->  Hash Join  (cost=15.40..36.53 rows=4 width=12)"
"                    Hash Cond: ((alb.nif_pro)::text = (pro.nif)::text)"
"                    ->  Seq Scan on albara alb  (cost=0.00..18.80 rows=880 width=52)"
"                    ->  Hash  (cost=15.38..15.38 rows=2 width=40)"
"                          ->  Seq Scan on proveidor pro  (cost=0.00..15.38 rows=2 width=40)"
"                                Filter: ((nom)::text = 'UIBFruita'::text)"
"              ->  Index Only Scan using pk_linalb on linia_albara la  (cost=0.15..0.27 rows=4 width=56)"
"                    Index Cond: (codi_alb = alb.codi)"
"        ->  Index Scan using pk_aliment on aliment ali  (cost=0.15..0.22 rows=1 width=162)"
"              Index Cond: ((referencia)::text = (la.referencia)::text)"
```

2. ANÁLISIS DEL QUERY PLAN

El output obtenido nos muestra el plan de ejecución, que describe cómo se ejecuta la consulta paso a paso. El plan de ejecución está organizado en una estructura jerárquica (fácil representación mediante un árbol) que muestra las operaciones que se realizan y los costos asociados a cada paso. Esta estructura jerárquica se puede entender mejor si, en lugar de usar las tabulaciones que PostgreSQL nos devuelve para entender el nivel de profundidad de cada nodo en el árbol, representamos gráficamente nosotros mismos este árbol con cada uno de los nodos obtenidos.

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 9 nodos, representados por pgAdmin (o, en su defecto, sql shell) mediante las 14 líneas obtenidas como output. Analicemos el resultado nodo por nodo:

1. **SORT** → Las líneas 1 y 2 del Query Plan: Al analizar la primera línea nos damos cuenta de que el ***start-up cost*** es de 38,71 unidades computacionales, el ***total cost*** es de 38,72 (es decir, se estima un coste de 38,72), se estima un número de filas resultante (***ouput rows***) de 4, y se estima un ancho de filas (***average row size***) de 162 bytes.
   La segunda línea, por otro lado, nos informa de que se van a ordenar los resultados en función de la columna ALI.referencia de la tabla Aliment en orden ascendente (ASC).

2. **NESTED LOOP (nodo #2)** → La línea 3 del Query Plan. Esta línea nos informa de que el ordenamiento del nodo 1 es precedido con un bucle anidado que unirá los resultados de operaciones anteriores. Representa la operación de JOIN entre la tabla Aliment y la tabla Linia_Albara. Es el paso final de la consulta en este caso.
   Se estima un **start-up cost** con valor de **15,70** y un **total cost** de **38.67** (coste estimado de 38,67 unidades computacionales). Además, se espera obtener 4 filas **(rows=4)** y, nuevamente, un ancho de filas de 162 bytes **(width=162)**.

3. **NESTED LOOP (nodo #3)** → La cuarta línea del Query Plan. Es otro bucle anidado que precede al bucle anidado del nodo 2. La función es la misma: unir los resultados de operaciones anteriores. Dado que hay varios JOIN, se han anidado varias operaciones de "Nested Loop" para unir las tablas en la consulta original. En concreto, este "Nested Loop" tendrá lugar dentro del "Nested Loop" del nodo 2 (tercera línea del Query Plan) y representa la operación de JOIN entre las tablas Linia_Albara y Albara.
   Los valores obtenidos en este caso son: **cost=15.55..37.79 rows=4 width=44**

4. **HASH JOIN** → Las líneas 5 y 6 del Query Plan. Este nodo representa una unión hash entre las tablas Albara (ALB) y Proveidor (PRO). Esto se hace utilizando el campo nif_pro en Albara y nif en Proveidor, tal y como se puede ver en la línea 6. "Hash Cond" se refiere a la condición de unión que se utiliza para realizar la unión hash entre dos tablas, es decir, especifica qué columnas se utilizan como clave para la operación de unión hash.
   Todo esto sucede dentro del "Nested Loop" del nodo 3, que precede al Hash Join. PostgreSQL utiliza una tabla hash para optimizar la unión.
   En este nodo, el coste estimado es de 36,53 **(cost=15.40..36.53)** y se estima obtener 4 filas con una anchura de 12 bytes **(rows=4 width=12)**.

5. **SEQ SCAN (nodo #5)** → La línea 7 del Query Plan. Este nodo representa la extracción de filas en la tabla Albara mediante una búsqueda secuencial en ella. El coste de esto se estima que sea **cost=0.00..18.80**. En este caso, se puede apreciar un aumento significativo en el número de filas que se espera obtener. Se estima que se obtendrán un total de 880 filas con una anchura de 52 bytes **(rows=880 width=52)**, cantidad que contrasta con el resto de los nodos ya que, hasta ahora, todos los ellos tenían el valor de **rows=4**, indicando que se esperaban 4 filas.

6. **HASH** → La octava línea del Query Plan. En este nodo se realiza una construcción de tabla Hash. Los valores de costo estimado, número de filas y anchura son los siguientes: **cost=15.38..15.38 rows=2 width=40**

7. **SEQ SCAN (nodo #7)** → Las líneas 9 y 10 del plan de ejecución. Este nodo es el de "mayor profundidad" en el árbol. Al igual que el nodo 5, representa un recorrido secuencial, pero en este caso se hace sobre la tabla Proveidor y estableciendo un filtro en la columna nom (línea 10 del Query Plan: Filter). Se buscan las filas en las que se cumple con la condición de que el valor de la columna nom sea "*UIBFruita*", es decir, se buscan todos los proveedores de nombre *UIBFruita*.
   Todo esto sucede dentro del Hash explicado anteriormente, por tanto, se unen las filas obtenidas de la tabla Proveidor con las obtenidas en la tabla Albara.
   El costo estimado de esta operación es de **15.38 unidades**, y se espera obtener **2 filas** con **40 bytes** de width

8. **INDEX ONLY SCAN** → Las líneas 11 y 12. En este nodo, se realiza un escaneo utilizando un índice (de ahí que se llame Index Only Scan) en la tabla Linia_albara. La condición de este índice es que el campo codi_alb en la tabla Albara ALB sea igual al campo codi en la tabla Linia_albara LA. Para ello, hace uso de la CONSTRAINT "pk_linalb" definida cuando se crearon las tablas, en la que se establece que la PRIMARY KEY de la tabla Linia_Albara es (codi_alb, referencia)
   Todo esto sucede dentro del "Nested Loop" del nodo 3 y, por tanto, se contempla la condición evaluada en ese loop.
   El coste estimado es el siguiente: **cost=0.15..0.27 rows=4 width=56**

9. **INDEX SCAN** → Las líneas 13 y 14 del plan de ejecución. Se hace un escaneo en la tabla Aliment utilizando la CONSTRAINT pk_aliment. Esta se utiliza para igualar el campo referencia en Aliment (ALI) y el campo referencia en Linia_Albara (LA). Esto sucede dentro del Nested Loop del nodo 2
   El coste estimado es el siguiente: **cost=0.15..0.22 rows=1 width=162**

El último paso será el Sort, que ordenará el resultado obtenido en función del valor de la referencia del alimento con criterio Ascendente (ORDER BY ALI.referencia ASC).

PostgreSQL ha utilizado operaciones como unión hash, exploración secuencial y escaneo de índice para realizar la consulta, y se ha especificado en qué tablas, en qué orden y con qué objetivo se ha aplicado cada una de estas operaciones. Además, se ha especificado la jerarquía de este plan de ejecución y se ha representado en forma de árbol para facilitar su comprensión.