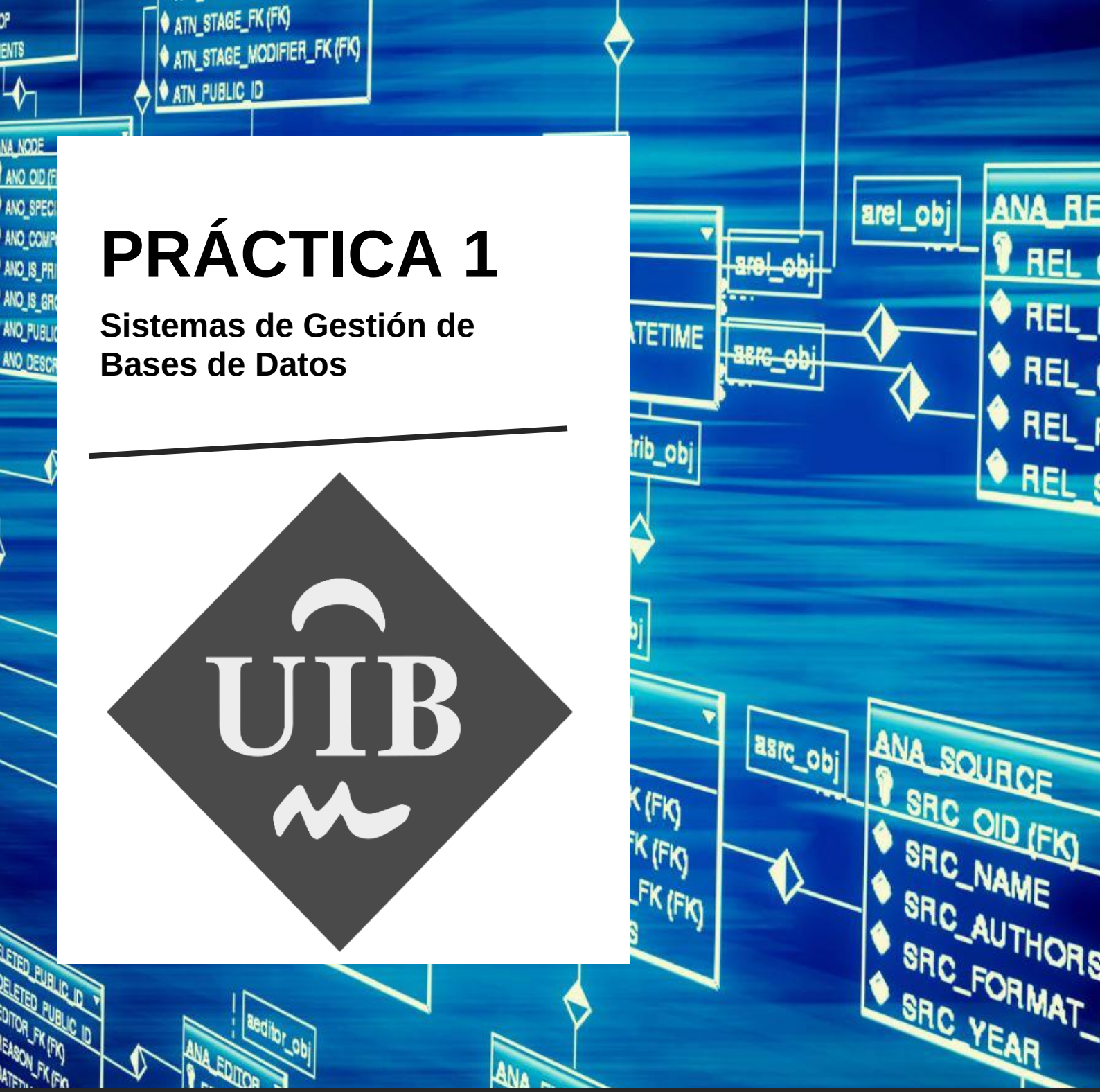


PRÁCTICA 1

Sistemas de Gestión de Bases de Datos

19 OCTUBRE

PRÁCTICA 1 – EJERCICIO 10
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 4 del examen cuya solución se encuentra en el archivo mvb135_5.sql y obtenemos lo siguiente:

INPUT:

```
SELECT ALI.referencia, ALI.nom, (COALESCE(QUILOSA.KG, 0) - COALESCE(QUILOSV.KG_VENUTS,
0)) AS STOCK
FROM Aliment ALI
    LEFT JOIN QUILOSALBARA QUILOSA
    ON QUILOSA.referencia = ALI.referencia
        LEFT JOIN QUILOSVENDA QUILOSV
        ON QUILOSV.referencia = ALI.referencia;
        -- ORDER BY ... (se puede añadir algun criterio que haga el resultado más le-
gible)
```

Para esta consulta, sin embargo, se han usado 2 vistas. Las vistas son las siguientes:

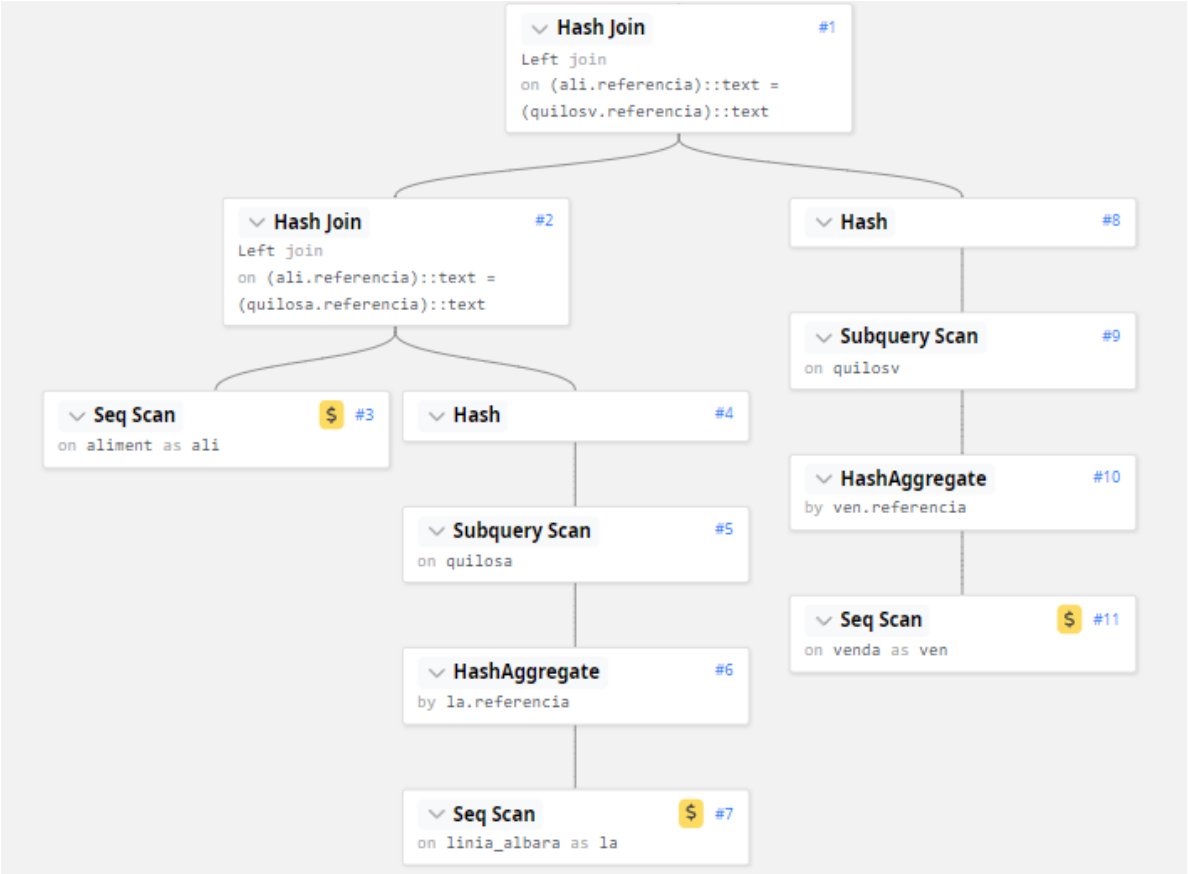
```
/*
Creamos una vista donde se vea, para cada venta, la referencia del producto vendido y
los kg vendidos
*/
CREATE VIEW QUILOSVENDA AS
    SELECT VEN.referencia, SUM(VEN.quilograms) AS KG_VENUTS
    FROM Venda VEN
    GROUP BY VEN.referencia;

/*
Creamos una vista donde se vea, para cada producto en el albaran, la referencia del
producto vendido y los kg
*/
CREATE VIEW QUILOSALBARA AS
    SELECT LA.referencia, SUM(LA.quilograms) AS KG
    FROM linia_albara LA
    GROUP BY LA.referencia;
```

OUTPUT:

```
Hash Left Join  (cost=58.65..76.13 rows=420 width=170)
  Hash Cond: ((ali.referencia)::text = (quilosv.referencia)::text)
  ->  Hash Left Join  (cost=29.70..45.01 rows=420 width=170)
        Hash Cond: ((ali.referencia)::text = (quilosa.referencia)::text)
        ->  Seq Scan on aliment ali  (cost=0.00..14.20 rows=420 width=162)
        ->  Hash  (cost=27.20..27.20 rows=200 width=52)
              ->  Subquery Scan on quilosa  (cost=23.20..27.20 rows=200 width=52)
                    ->  HashAggregate  (cost=23.20..25.20 rows=200 width=52)
                          Group Key: la.referencia
                          ->  Seq Scan on linia_albara la  (cost=0.00..18.80 rows=880 width=48)
  ->  Hash  (cost=26.45..26.45 rows=200 width=52)
        ->  Subquery Scan on quilosv  (cost=22.45..26.45 rows=200 width=52)
              ->  HashAggregate  (cost=22.45..24.45 rows=200 width=52)
                    Group Key: ven.referencia
                    ->  Seq Scan on venda ven  (cost=0.00..18.30 rows=830 width=48)
```

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 11 nodos, representados por pgAdmin (o, en su defecto, sql shell) mediante las 15 líneas obtenidas como output. Analicemos el resultado nodo por nodo:

1. **HASH LEFT JOIN (nodo #1)** → Las líneas 1 y 2 del plan de ejecución, representadas en el nodo raíz del árbol de la imagen, representan que la consulta empieza con una operación de Hash Left Join. Esta operación une la tabla Aliment (ALI) con la vista ***QUILOSV***. Esta vista representa, para cada venta, la referencia del producto vendido y la suma de kilogramos vendidos. Esta operación de unión se hace bajo una condición que se especifica en la línea 2: **(ali.referencia)::text = (quilosv.referencia)::text**, es decir, la columna referencia en ALI debe ser igual a la columna referencia en la vista QUILOSV.
   El costo estimado de la operación es de **58.65..76.13**, y se espera una cantidad estimada de **420** filas de anchura de **170** bytes.

2. **HASH LEFT JOIN (nodo #2**) → Este segundo nodo, formado por la tercera y la cuarta línea del plan de ejecución, representa otra operación de Hash Left Join. Se hace lo mismo que en el nodo anterior, pero con la diferencia de que esta vez se unen las tablas Aliment (ALI) y la vista *QUILOSALBARA (QUILOSA)*. La vista *QUILOSA* representa la suma de kilogramos en albaranes para cada producto.
   La condición de unión en este caso también consiste en la igualación de las columnas referencia en cada una de estas tablas: ALI.referencia = QUILOSA.referencia.
   El costo estimado de la operación y la cantidad de filas estimadas y anchura es:
   cost=29.70..45.01 rows=420 width=170

3. **SEQ SCAN (nodo #3)** → La quinta línea del Query Plan viene representada en este nodo. Se realiza un escaneo secuencial en la tabla Aliment (ALI), que se aplica sin filtro, con coste **cost=0.00..14.20 rows=420 width=162**, y que pretende obtener las filas de alimentos.

4. **HASH (nodo #4)** → La línea número 6. Este nodo representa una operación de Hash para construir una tabla hash basada en la subconsulta de la vista *QUILOSALBARA* (*QUILOSA*). El coste estimado de esta operación es **27.20 .27.20** unidades, y se espera una cantidad de **200** filas con una anchura de **52** bytes.

5. **SUBQUERY SCAN (nodo #5)** → La séptima línea del plan de ejecución representa, en este nodo, una nueva búsqueda secuencial en una subconsulta, en la vista *QUILOSALBARA* (*QUILOSA*). Esta subconsulta escanea la tabla Linia_Albara y realiza una operación de "Hash Aggregate" (nodo 6 que explicaremos a continuación) para calcular la suma de quilogramos en albaranes. Esta operación en la vista *QUILOSA* tiene un coste de **23.20..27.20** unidades de costo, y estimamos obtener una cantidad de **200** filas de **52** bytes de anchura.

6. **HASH AGGREGATE (nodo #6)** → La octava y novena línea del plan de ejecución vienen representadas en este nodo. Este nodo forma parte de la subconsulta descrita previamente, y en él, se hace una operación de agregación “Hash Aggregate” para obtener la suma de quilogramos (función SUM(LA.quilograms) de la vista *QUILOSALBARA*) en la tabla Linia_Albara y agrupar los resultados por referencia (línea 9: ***Group key: la.referencia***).
   El coste de todo esto es de **23.20..25.20** unidades. Se esperan **200** filas de anchura **52** bytes.

7. **SEQ SCAN (nodo #7)** → La décima línea del Query Plan representa, en este séptimo nodo, un escaneo secuencial sobre la tabla Linia_Albara. Este escaneo es la última operación de esta subconsulta, y tiene un coste estimado de **0.00..18.80**. Se espera obtener una cantidad de **880** filas con una anchura de **48** bytes.

8. **HASH (nodo #8)** → La undécima línea del Query Plan. Dado que hay otra subconsulta en la otra vista (ya no estamos dentro de la subconsulta descrita anteriormente), se repite el mismo proceso descrito antes. Se realiza otra operación de "Hash" para construir una tabla basada en la subconsulta de la vista QUILOSVENDA (QUILOSV). Esta vista permite obtener, para cada venta, la referencia del producto y los kg vendidos de cada uno.
   El coste de esta operación de Hash es: **cost=26.45..26.45 rows=200 width=52**

9. **SUBQUERY SCAN (nodo #9)** → La línea 12 del plan de ejecución. Representa, al igual que en la subconsulta anterior, una búsqueda secuencial en la subconsulta de la vista QUILOSVENDA (QUILOSV). En esta subconsulta se escanea la tabla Venda y, al igual que en el caso anterior, los nodos de nivel inferior harán uso de una operación de Hash Aggregate y un escaneo secuencial (nodos 10 y 11 respectivamente).
   El coste en este nodo es: **cost=22.45..26.45 rows=200 width=52**

10. **HASH AGGREGATE (nodo #10)** → La decimotercera y decimocuarta línea. Estas dos líneas representan en este nodo el uso de una operación de agregación de tipo hash (subconsulta de la vista *QUILOSVENDA QUILOSV)*. Esto se hace para obtener la suma de quilogramos vendidos (función SUM(VEN.quilograms) en la vista *QUILOSVENDA*) en la tabla Venda. Los resultados se agrupan por referencia, tal y como viene establecido en la línea 14 (***Group key: ven.referencia***).
    El coste de estas operaciones es de 22.45..24.45 unidades, y se estiman 200 filas con 52 bytes de anchura.

11. **SEQ SCAN (nodo #11)** → Este último nodo (línea 15 del Query Plan) representa un escaneo secuencial en la tabla Venda. Es la última operación de esta segunda subconsulta y se le estima un coste y cantidad de filas de: **cost=0.00..18.30 rows=830 width=48**

En conclusión, esta consulta hace uso de funciones de Hash para realizar las uniones (Hash Join y hash), y también hace uso de una agregación de tipo hash para obtener las sumas de kg totales de alimentos. Las búsquedas en las tablas se realizan mediante escaneos secuenciales (Seq Scan) y se tienen 2 subconsultas (Subquery Scan), una para la vista *QUILOSVENDA* y otra para la vista *QUILOSALBARA*.

El uso de vistas, en este caso, no causa un impacto en el coste total de la consulta (coste máximo estimado de 76.13), aunque si que aumenta la complejidad de la consulta principal, ya que la consulta principal termina haciendo uso de subconsultas. Si estas subconsultas dentro de las vistas tuviesen una complejidad mayor, es probable que el uso de estas vistas sí que tuviera una influencia directa en el coste máximo estimado de la consulta principal.