

5.2 Tutorial Query Planning (video relacionado: 5.7 Query planning. Contenido relacionado: Query planning in Amazon Redshift)

Trabajaremos sobre la misma [base de datos](#) que utilizamos para explicar las bases de datos columnares. En esta oportunidad el data Manager te ha pedido que le ayudes a analizar la escalabilidad de un requerimiento del equipo de ventas y que tan demandante podría ser esta tarea.

El requerimiento es:

Calcular el monto promedio (en miles) para cada categoría de decisión (F, D, C, A y X) en el contexto bancario según las siguientes reglas:

- Si el objetivo es verdadero, el nombre_tipo_contrato es 'Cash loans' y el monto del crédito es superior a 50 000, la categoría de decisión es 'F'.
- Si el destino es verdadero, el nombre_tipo_contrato es 'Revolving loans' y el monto del crédito es superior a 9.000, la categoría de decisión es 'D'.
- Si el objetivo es falso, el nombre_tipo_contrato es 'Revolving loans' y el monto del crédito es superior a 100 000, la categoría de decisión es 'C'.
- Si el destino es falso, el nombre_tipo_contrato es 'Cash loans' y el monto del crédito es superior a 120 000, la categoría de decisión es 'A'.
- Para cualquier otro caso que no cumpla con las condiciones anteriores, la categoría de decisión es 'X'.

Deberás analizar el costo en tiempo y en cantidad de \$ que involucraría este requerimiento.

En la próxima página podrás acceder a la solución.

❖ Solución:

1. Primero debemos generar la consulta para resolver el problema

```
```sql
SELECT x.decision as decision, AVG(x.cantidad)/1000 as Monto_K
FROM
 (SELECT target AS Moroso,
 name_contract_type AS Contrato,
 code_gender AS Genero,
 amt_credit AS Cantidad,
 CASE

 WHEN (target = true AND name_contract_type = 'Cash loans' AND
 amt_credit > 50000) THEN 'F'
 WHEN (target = true AND name_contract_type = 'Revolving loans'
 AND amt_credit > 9000) THEN 'D'
 WHEN (target = false AND name_contract_type = 'Revolving loans'
 AND amt_credit > 100000) THEN 'C'
 WHEN (target = false AND name_contract_type = 'Cash loans'
 AND amt_credit > 120000) THEN 'A'
 ELSE 'X'
 END AS Decision
 FROM fraude) AS x
GROUP BY decision
ORDER BY Monto_k
```
```

2. Ahora podemos colocar la palabra Explain al inicio para analizar la escalabilidad de la consulta:

```

explain SELECT x.decision as decision, AVG(x.cantidad)/1000 as Monto_K
FROM
  (SELECT target AS Moroso,
    name_contract_type AS Contrato,
    code_gender AS Genero,
    amt_credit AS Cantidad,
    CASE
      WHEN (target = true AND name_contract_type = 'Cash loans' AND amt_credit>50000)
      WHEN (target = true AND name_contract_type = 'Revolving loans' AND amt_credit> 9
      WHEN (target = false AND name_contract_type = 'Revolving loans' AND amt_credit >
      WHEN (target = false AND name_contract_type = 'Cash loans' AND amt_credit > 1200
      ELSE 'X'
    END AS Decision
  FROM fraude) AS x
GROUP BY decision
ORDER BY Monto_k

```

resultados 1 x

Enter a SQL expression to filter results (use Ctrl+Space)

QUERY PLAN

- XN Merge (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)
 - Merge Key: (avg(amt_credit) / 1000::double precision)
 - > XN Network (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)
 - Send to leader
 - > XN Sort (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)
 - Sort Key: (avg(amt_credit) / 1000::double precision)
 - > XN HashAggregate (cost=13837.99..14214.87 rows=10768 width=23)

Refresh Save Cancel Exportar datos ... 200 8

- Ahora podemos seleccionar todo el texto que sale como resultado y lo podemos copiar a un bloc de notas:

```

XN Merge (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)
Merge Key: (avg(amt_credit) / 1000::double precision)
-> XN Network (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)
  Send to leader
    -> XN Sort (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)
      Sort Key: (avg(amt_credit) / 1000::double precision)
        -> XN HashAggregate (cost=13837.99..14214.87 rows=10768 width=23)
          -> XN Seq Scan on fraude (cost=0.00..12300.44 rows=307511 width=23)|

```

O en su defecto ponerlo en un json:

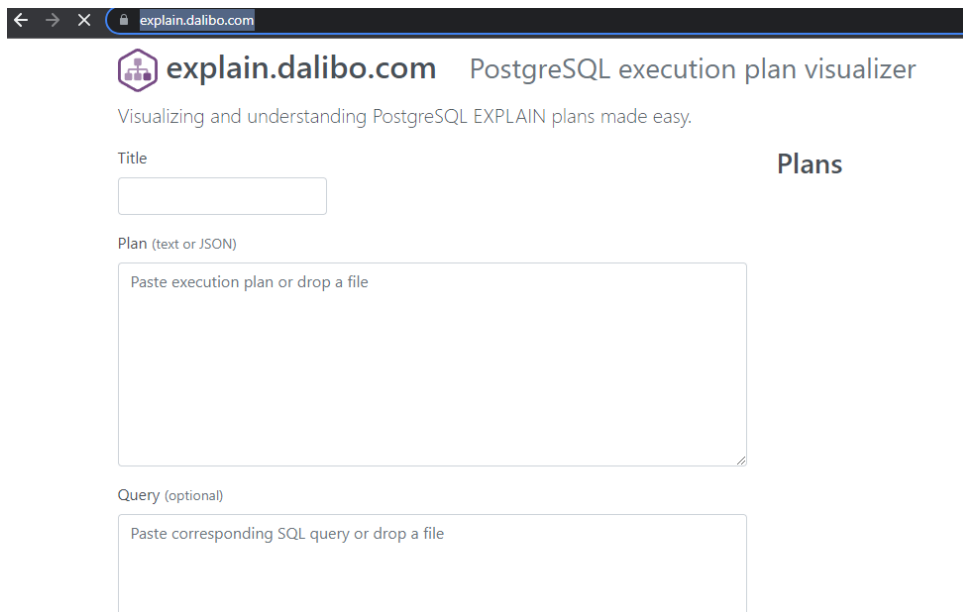
```

XN Merge (cost=1000000014936.03..1000000014962.95 rows=10768
width=23)
  Merge Key: (avg(amt_credit) / 1000::double precision)
    -> XN Network (cost=1000000014936.03..1000000014962.95
rows=10768 width=23)
      Send to leader

```

-> XN Sort (cost=10000000014936.03..10000000014962.95
rows=10768 width=23)
Sort Key: (avg(amt_credit) / 1000)::double precision)
-> XN HashAggregate (cost=13837.99..14214.87 rows=10768
width=23)
-> XN Seq Scan on fraude (cost=0.00..12300.44 rows=307511
width=23)

4. Luego podemos ir a la pagina explain.dalibo.com:



The screenshot shows the web interface of explain.dalibo.com, a PostgreSQL execution plan visualizer. The browser's address bar shows the URL. The page has a header with the site logo and name, followed by the tagline "PostgreSQL execution plan visualizer" and a subtitle "Visualizing and understanding PostgreSQL EXPLAIN plans made easy." Below the header, there are three main input sections: 1. "Title" with a text input field. 2. "Plan (text or JSON)" with a large text area containing the placeholder "Paste execution plan or drop a file". 3. "Query (optional)" with a text area containing the placeholder "Paste corresponding SQL query or drop a file". To the right of the "Title" and "Plan" sections, there is a "Plans" button.

5. Podemos copiar todo el contenido del bloc de notas en la sección Plan. También podemos ponerle un nombre al plan que analizaremos e.g Prueba Query:

Visualizing and understanding PostgreSQL EXPLAIN plans made easy.

Title

Pruebas Query

Plan (text or JSON)

```
XN Merge (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)
  Merge Key: (avg(amt_credit) / 1000::double precision)
  -> XN Network (cost=1000000014936.03..1000000014962.95 rows=10768
width=23)
    Send to leader
    -> XN Sort (cost=1000000014936.03..1000000014962.95 rows=10768
width=23)
      Sort Key: (avg(amt_credit) / 1000::double precision)
```

Query (optional)

Paste corresponding SQL query or drop a file

6. Bajamos y damos click en submit:

Submit

Sample Plans ^

7. Luego presionamos confirm:

Submit plan?



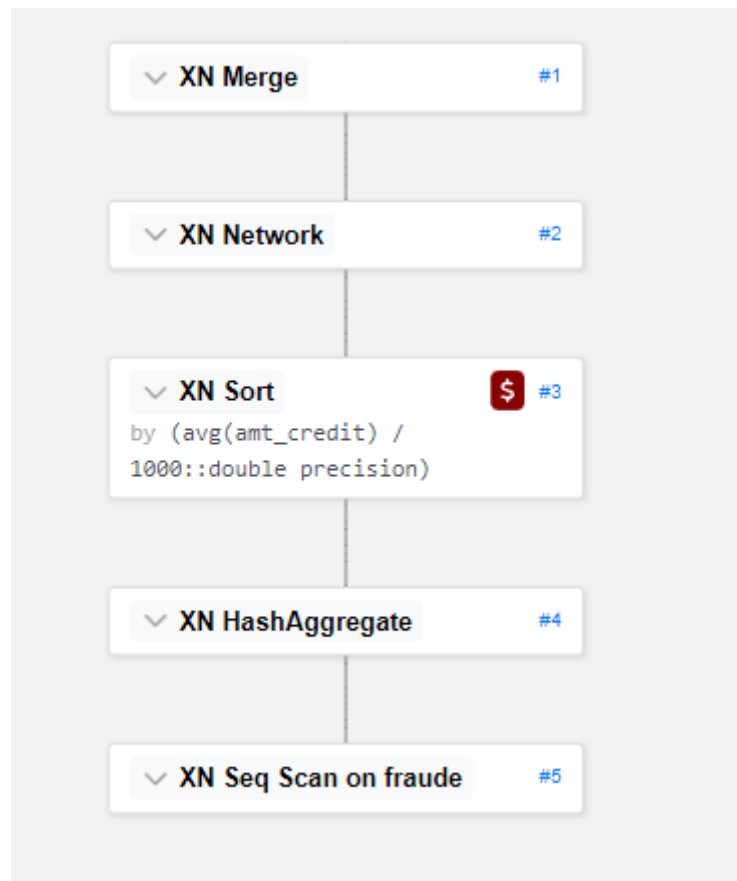
The plan will be sent to the server and stored in a database.
See the [data retention policy](#) for more info.

Cancel

☐ Don't ask me again

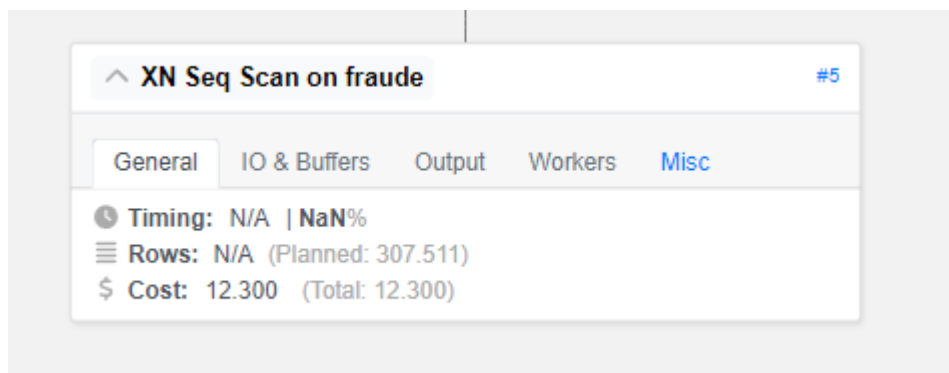
Confirm

8. Obtendremos una salida como ésta:



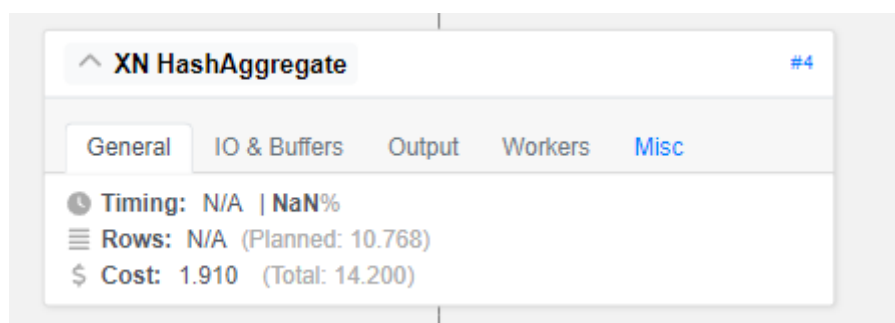
9. El orden de ejecución es de abajo hacia arriba. Presionamos en XN Seq Scan on fraude.

Interpretación: análisis secuencial de la tabla "fraude", lo que implica que la consulta está leyendo todas las filas de esta tabla. El costo estimado de este análisis se muestra 12300 unidades (**ahora bien ten presente este número porque si bien no es un costo real nos permite comparar qué operaciones son más demandantes**). El número de filas estimado para este escaneo es 307511.



ACLARACIÓN IMPORTANTE: El costo es acumulado en cada etapa no lo olvides (es decir cada sección no tiene un costo independiente)

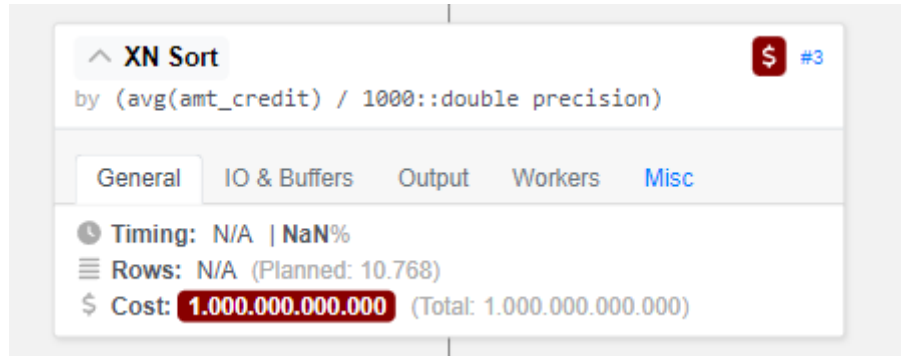
10. Luego podemos dar click en XN HashAggregate y veremos algo como esto
Interpretación: Después de escanear la tabla "fraude", los datos se pasan al paso HashAggregate. Este paso realiza operaciones de agregación con el GROUP BY. El costo estimado para este paso se muestra como 14200 (**apenas aumentó ligeramente**) y se espera producir 10.768 filas.



11. Ahora damos click en XN sort y veremos algo como esto:

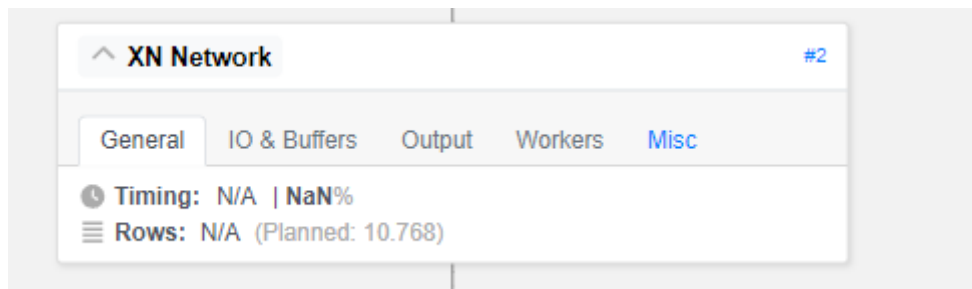
Interpretación: Siguiendo el paso HashAggregate, los datos se ordenan según la expresión "(avg(amt_credit) / 1000::double precision)" del paso anterior Como puedes ver hay un color especial (**es el paso más**

demandante). El costo estimado para esta operación aumenta hasta 10000000000000 y se realiza en las 10768 filas producidas por el paso anterior.:



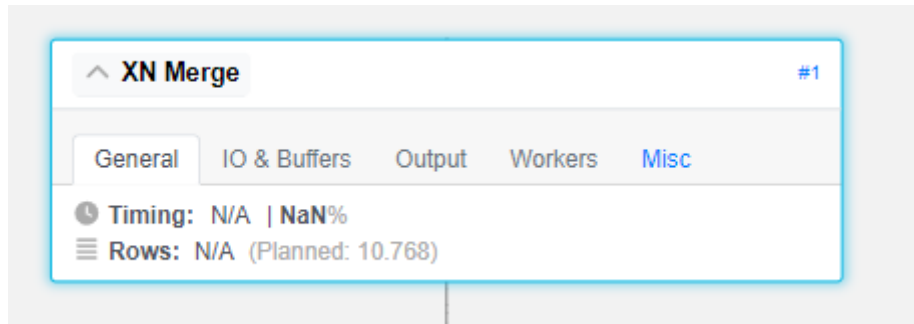
12. Ahora podemos analizar XN Network:

Interpretación: Luego, los datos ordenados se envían al nodo líder. En Redshift (**recuerdas el concepto de nodos líder y de cómputo, bueno acá es importante recordarlo**), y este paso implica transferir los datos de los nodos de cálculo al nodo líder para su posterior procesamiento. El costo no se mueve del acumulado ~10000000000000



13. Finalmente analizamos XN Merge

Interpretación: Finalmente, los datos recibidos por el nodo líder se fusionan a través de "(avg(amt_credit) / 1000::double precision)". El costo no se mueve del acumulado ~10000000000000



Anexo

Lo que hicimos en Amazon Redshift también se puede realizar en PostgreSQL

En postgres tenemos lo siguientes

```

sql
explain SELECT x.decision as decision, AVG(x.cantidad)/1000 as Monto_K
FROM
  (SELECT target AS Moroso,
    name_contract_type AS Contrato,
    code_gender AS Genero,
    amt_credit AS Cantidad,
    CASE
      WHEN (target = true AND name_contract_type = 'Cash loans' AND
amt_credit>50000) THEN 'F'
      WHEN (target = true AND name_contract_type = 'Revolving loans' AND
amt_credit> 9000) THEN 'D'
      WHEN (target = false AND name_contract_type = 'Revolving loans' AND
amt_credit > 100000) THEN 'C'
      WHEN (target = false AND name_contract_type = 'Cash loans' AND
amt_credit > 120000) THEN 'A'
      ELSE 'X'
    END AS Decision
  FROM fraude) AS x
GROUP BY decision
ORDER BY Monto_k

```

Seguro notarás que los resultados no son exactamente lo mismo que en Amazon Redshift, bueno tiene sentido ya que es un sistema análogo a OLTP y no OLAP:

```
"Sort (cost=12110.55..12139.87 rows=11728 width=40)"
"  Sort Key: ((avg(fraude.amt_credit) / '1000'::double precision))"
"  -> Finalize HashAggregate (cost=10907.39..11317.87 rows=11728 width=40)"
"    Group Key: (CASE WHEN (fraude.target AND
((fraude.name_contract_type)::text = 'Cash loans'::text) AND (fraude.amt_credit
> '50000'::double precision)) THEN 'F'::text WHEN (fraude.target AND
((fraude.name_contract_type)::text = 'Revolving loans'::text) AND
(fraude.amt_credit > '9000'::double precision)) THEN 'D'::text WHEN ((NOT
fraude.target) AND ((fraude.name_contract_type)::text = 'Revolving loans'::text)
AND (fraude.amt_credit > '100000'::double precision)) THEN 'C'::text WHEN
((NOT fraude.target) AND ((fraude.name_contract_type)::text = 'Cash loans'::text)
AND (fraude.amt_credit > '120000'::double precision)) THEN 'A'::text ELSE 'X'::text
END)"
"    -> Gather (cost=9324.11..10848.75 rows=11728 width=64)"
"      Workers Planned: 1"
"        -> Partial HashAggregate (cost=8324.11..8675.95 rows=11728 width=64)"
"          Group Key: CASE WHEN (fraude.target AND
((fraude.name_contract_type)::text = 'Cash loans'::text) AND (fraude.amt_credit
> '50000'::double precision)) THEN 'F'::text WHEN (fraude.target AND
((fraude.name_contract_type)::text = 'Revolving loans'::text) AND
(fraude.amt_credit > '9000'::double precision)) THEN 'D'::text WHEN ((NOT
fraude.target) AND ((fraude.name_contract_type)::text = 'Revolving loans'::text)
AND (fraude.amt_credit > '100000'::double precision)) THEN 'C'::text WHEN
((NOT fraude.target) AND ((fraude.name_contract_type)::text = 'Cash loans'::text)
AND (fraude.amt_credit > '120000'::double precision)) THEN 'A'::text ELSE 'X'::text
END"
"            -> Parallel Seq Scan on fraude (cost=0.00..7419.67 rows=180889
width=40)"
```

