Comando EXPLAIN sem alteração no modelo e sem melhoria na QUERY 3

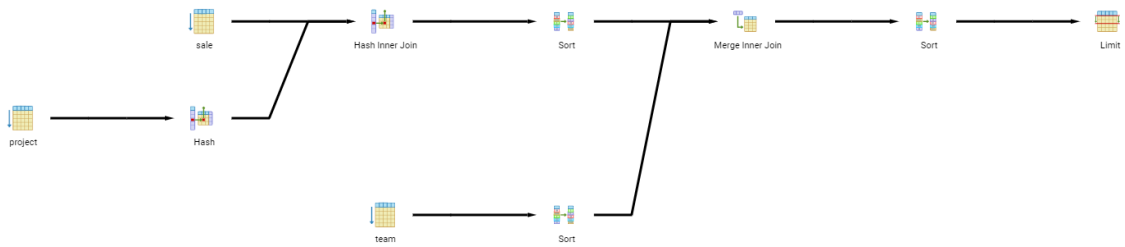
```
explain SELECT T.name_team, P.name_project, S.amount_sale_project
FROM team as T NATURAL JOIN sale as S NATURAL JOIN project as P
WHERE T.idt_project = P.idt_project AND P.idt_project = S.idt_project
ORDER BY S.amount_sale_project DESC
LIMIT 1
```

Consulta executada com o comando EXPLAIN

1.	→ Limit
2.	→ Sort
3.	→ Merge Inner Join
4.	→ Sort
5.	→ Hash Inner Join Hash Cond: (s.idt_project = p.idt_project)
6.	→ Seq Scan on sale as s
7.	→ Hash
8.	→ Seq Scan on project as p
9.	→ Sort
10.	→ Seq Scan on team as t

	QUERY PLAN text
1	Limit (cost=341.63..341.63 rows=1 width=96)
2	-> Sort (cost=341.63..358.58 rows=6780 width=96)
3	Sort Key: s.amount_sale_project DESC
4	-> Merge Join (cost=200.38..307.73 rows=6780 width=96)
5	Merge Cond: (s.idt_project = t.idt_project)
6	-> Sort (cost=117.01..119.83 rows=1130 width=72)
7	Sort Key: s.idt_project
8	-> Hash Join (cost=35.42..59.70 rows=1130 width=72)
9	Hash Cond: (s.idt_project = p.idt_project)
10	-> Seq Scan on sale s (cost=0.00..21.30 rows=1130 width=36)
11	-> Hash (cost=21.30..21.30 rows=1130 width=36)
12	-> Seq Scan on project p (cost=0.00..21.30 rows=1130 width=36)
13	-> Sort (cost=83.37..86.37 rows=1200 width=36)
14	Sort Key: t.idt_project
15	-> Seq Scan on team t (cost=0.00..22.00 rows=1200 width=36)

Plano de consulta executada com o comando EXPLAIN



Árvore de consulta executada com o comando EXPLAIN

```sql
/* depois: Query 3 seleciona o time que obteve a maior venda de projeto;
*   Decidimos manter com o objetivo principal da consulta, e, apenas adequamos ao novo modelo, substituindo o
* "ORDER BY" pelo "GROUP BY" e também adicionando o HAVING MAX
*
*/
SELECT T.name_team, P.name_project, MAX(S.amount_sale_project)
FROM team as T
NATURAL JOIN sale as S
NATURAL JOIN project as P
WHERE T.idt_project = P.idt_project AND P.idt_project = S.idt_project
GROUP BY T.name_team, P.name_project
HAVING MAX(S.amount_sale_project)>=ALL(
    SELECT amount_sale_project
    FROM sale
    );
```

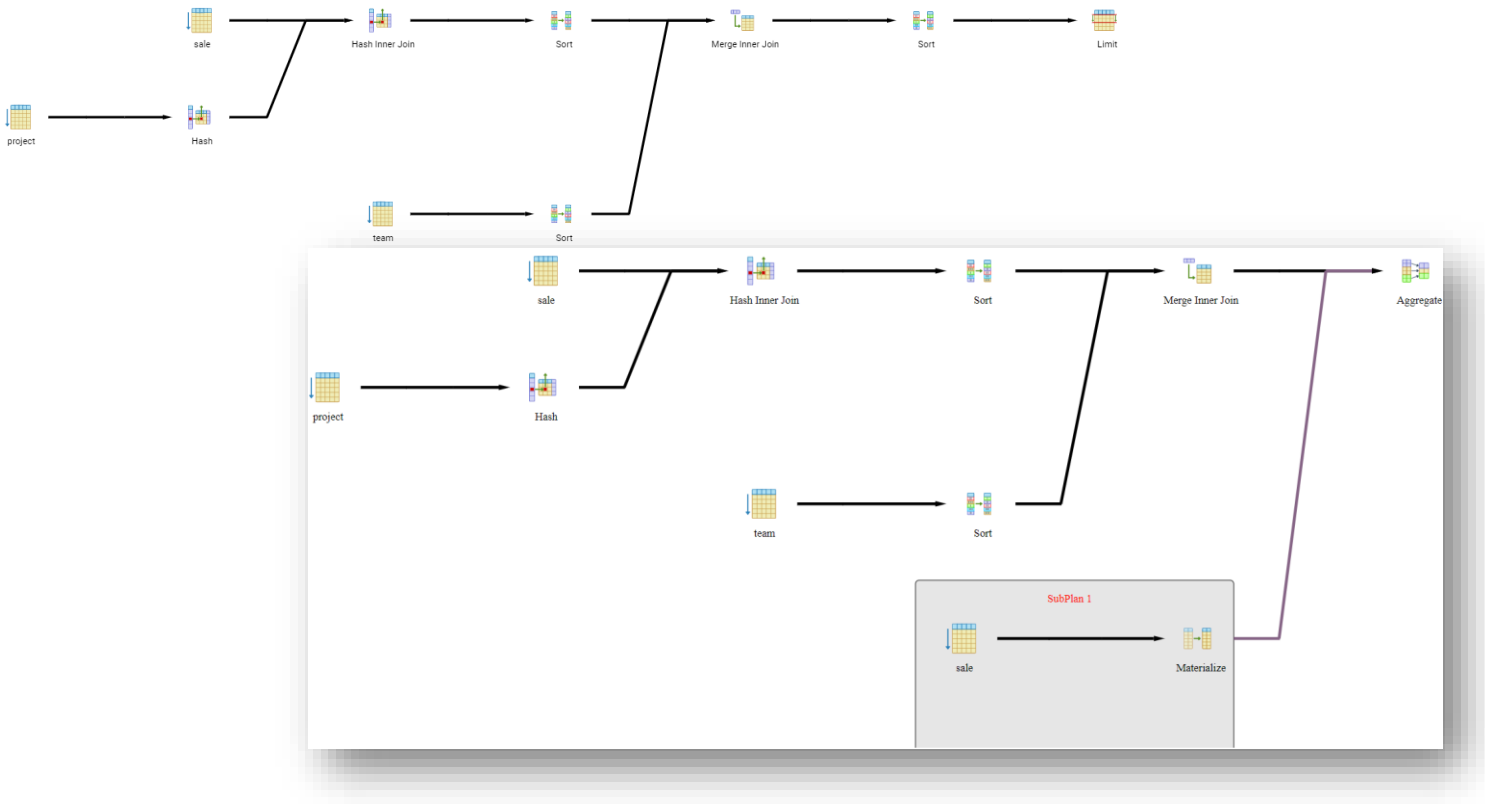
Consulta alterada e executada com o comando EXPLAIN

Comando EXPLAIN com alteração no modelo e sem melhoria na QUERY 3

	QUERY PLAN text
1	Limit (cost=341.63..341.63 rows=1 width=96)
2	-> Sort (cost=341.63..358.58 rows=6780 width=96)
3	Sort Key: s.amount_sale_project DESC
4	-> Merge Join (cost=200.38..307.73 rows=6780 width=96)
5	Merge Cond: (s.idt_project = t.idt_project)
6	-> Sort (cost=117.01..119.83 rows=1130 width=72)
7	Sort Key: s.idt_project
8	-> Hash Join (cost=35.42..59.70 rows=1130 width=72)
9	Hash Cond: (s.idt_project = p.idt_project)
10	-> Seq Scan on sale s (cost=0.00..21.30 rows=1130 width=36)
11	-> Hash (cost=21.30..21.30 rows=1130 width=36)
12	-> Seq Scan on project p (cost=0.00..21.30 rows=1130 width=36)
13	-> Sort (cost=83.37..86.37 rows=1200 width=36)
14	Sort Key: t.idt_project
15	-> Seq Scan on team t (cost=0.00..22.00 rows=1200 width=36)

	QUERY PLAN text
1	HashAggregate (cost=375.53..101397.53 rows=3390 width=96)
2	Group Key: t.name_team, p.name_project
3	Filter: (SubPlan 1)
4	-> Merge Join (cost=200.38..307.73 rows=6780 width=96)
5	Merge Cond: (s.idt_project = t.idt_project)
6	-> Sort (cost=117.01..119.83 rows=1130 width=72)
7	Sort Key: s.idt_project
8	-> Hash Join (cost=35.42..59.70 rows=1130 width=72)
9	Hash Cond: (s.idt_project = p.idt_project)
10	-> Seq Scan on sale s (cost=0.00..21.30 rows=1130 width=36)
11	-> Hash (cost=21.30..21.30 rows=1130 width=36)
12	-> Seq Scan on project p (cost=0.00..21.30 rows=1130 width=36)
13	-> Sort (cost=83.37..86.37 rows=1200 width=36)
14	Sort Key: t.idt_project
15	-> Seq Scan on team t (cost=0.00..22.00 rows=1200 width=36)
16	SubPlan 1
17	-> Materialize (cost=0.00..26.95 rows=1130 width=32)
18	-> Seq Scan on sale (cost=0.00..21.30 rows=1130 width=32)
19	JIT:
20	Functions: 29
21	Options: Inlining false, Optimization false, Expressions true, Deforming true

Comparação entre os planos de consultas antes x depois da alteração do modelo



Comparação entre as árvores de consultas antes x depois da alteração do modelo

O Plano feito partir da primeira consulta, realizada na “Parte 1” do trabalho, seleciona o time que obteve maior venda de projeto. Apresenta o “merge cond”, ou seja, a condição de mesclagem entre os campos através da chave “idt_project”. Faz uma busca sequencial, como já discutido esse processo é custoso, porém, como a consulta apresenta a cláusula “limit”, não é possível apresentar de fato o custo total dessa busca.

Após a alteração do modelo, a consulta dois, mantêm o mesmo objetivo da primeira consultam, porém, visto que a cláusula “LIMIT” foi um “bloqueador” na possibilidade de termos mais informações, optamos por não manter a cláusula de restrição. Em contra partida, utilizamos mais cláusulas, como a implementação de uma “SUBQUERY”, ou subconsulta, que trabalha junto com outra cláusula, o “HAVING MAX”, gerando um “SubPlan”, ou subplano.

O plano e a árvore para a segunda consulta teve um resultado que extraímos um aprendizado bastante interessante, o conceito de suplanos. As últimas cláusulas mencionadas no parágrafo anterior geraram um operador, que é chamado pelo menos no PostgreSQL de “SubqueryScan”, que é utilizado para satisfazer uma cláusula “UNION” e, por consequência, o subplano é utilizado para subconsultas.

A comparação entre ambas estruturas (primeira consulta x segunda consulta) nos gerou bastante impacto e, ao também realizarmos as comparações de custo, ficamos surpresos que, apesar de utilizarmos tais cláusulas junto com a situação de “falta de informação” que a cláusula “LIMIT” nos restringiu na primeira consulta, os custos para a segunda consulta não tiveram um impacto exorbitante, pelo contrário, manteve o padrão de também utilizar buscas sequenciais (“seq scan”); Tal padrão pode derivar-se dos ids, sequenciais, de tipos “SERIAL” que utilizamos no nosso banco.