Comando EXPLAIN sem alteração no modelo e sem melhoria QUERY 4

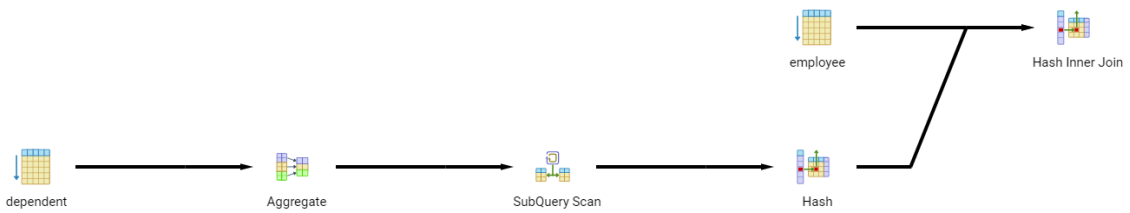
```
explain SELECT E.name_employee
FROM Employee E
LEFT JOIN (SELECT idt_employee, COUNT (*) as qtDependent FROM Dependent GROUP BY idt_employee) D ON E.idt_employee
 WHERE D.qtDependent > 2
```

Consulta executada com o comando EXPLAIN

1.	→ Hash Inner Join Hash Cond: (e.idt_employee = d.idt_employee)
2.	→ Seq Scan on employee as e
3.	→ Hash
4.	→ Subquery Scan
5.	→ Aggregate Filter: (count(*) > 2)
6.	→ Seq Scan on dependent as dependent

	QUERY PLAN text
1	Hash Join (cost=32.01..51.11 rows=67 width=32)
2	Hash Cond: (e.idt_employee = d.idt_employee)
3	-> Seq Scan on employee e (cost=0.00..17.20 rows=720 width=36)
4	-> Hash (cost=31.17..31.17 rows=67 width=4)
5	-> Subquery Scan on d (cost=28.00..31.17 rows=67 width=4)
6	-> HashAggregate (cost=28.00..30.50 rows=67 width=12)
7	Group Key: dependent.idt_employee
8	Filter: (count(*) > 2)
9	-> Seq Scan on dependent (cost=0.00..22.00 rows=1200 width=4)

Plano de consulta executada com o comando EXPLAIN



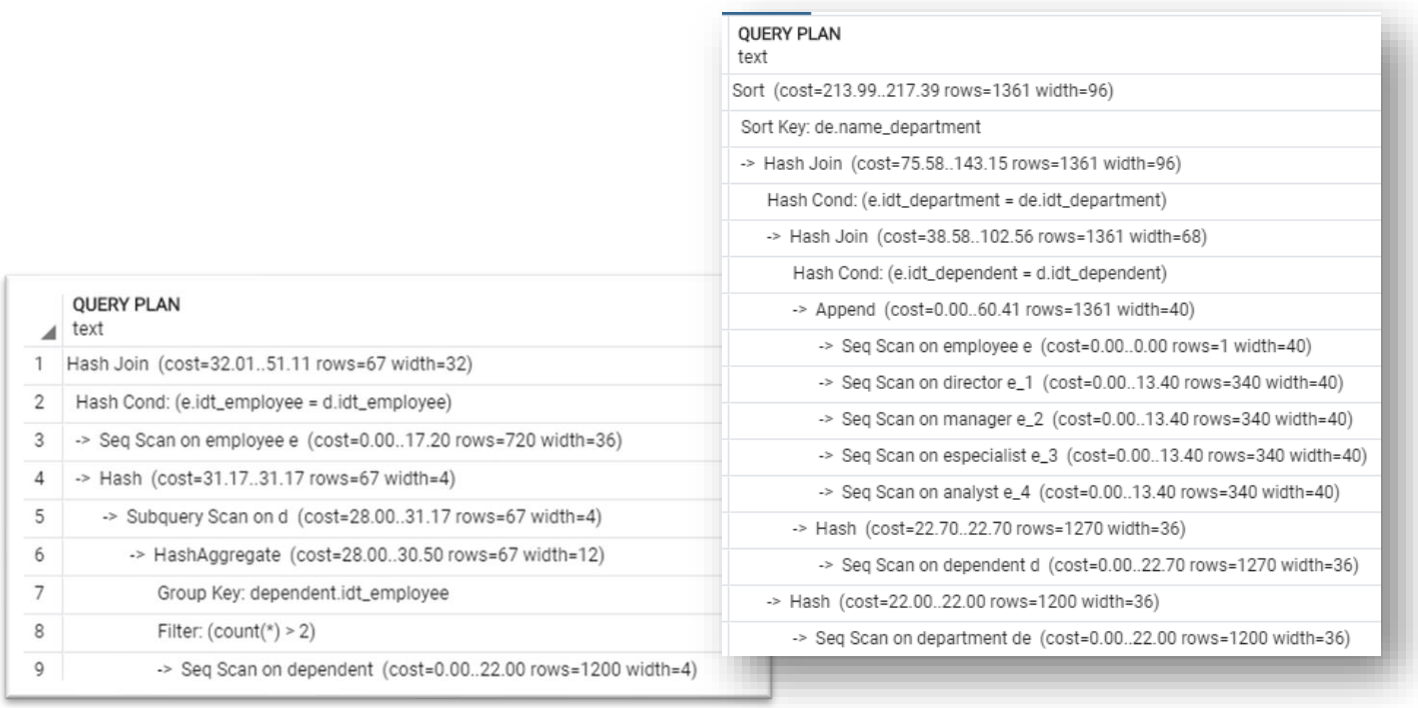
Árvore de consulta executada com o comando EXPLAIN

O plano de execução acima é feito na consulta que seleciona o empregado que possui mais de dois dependentes. Apresenta a busca sequencial na tabela principal “Employee”, e possui uma função de agregação na “SubQuery” através da cláusula “WHERE”. Se a tabela tiver um volume maior de dados a busca sequencial não seria uma vantagem, aqui ela conta com a ajuda da cláusula “WHERE” o que diminui o custo dessa busca, mas ainda assim ,no SGBD PostgreSQL, com base nas estatísticas que o comando “EXPLAIN” nos retorna, torna-se difícil validar se realmente esse custo irá diminuir, e, como comentado acima, se essas tabelas tivessem um volume de dados muito grande o processo seria ainda seria muito custoso e demoraria muito tempo para ser executado.

Comando EXPLAIN com alteração no modelo e sem melhoria na QUERY 4

```
/* depois: Query 4 seleciona o nome do dependente, o nome do empregado e
* o nome do departamento ordenado pelo nome do departamento.
*   Apos a alteracao no artefato A, ao adicionar a heranca na tabela "Employee", o
* postgres nao permitia nenhum objeto que possuisse a chave estrangeira da tabela
* "Employee". Sendo assim, agora nao e mais possivel um funcionario possuir mais do
* que um dependente. O valores do campo idt_dependent, para os registros dessa mesma tabela,
* podem variar (sequenciais e 'null' e deve possuir o valor correspondente ao idt_dependente
* inserido na tabela "Dependent".
*/
explain SELECT D.name_dependent, E.name_employee, DE.name_department
FROM Dependent AS D
INNER JOIN Employee AS E ON E.idt_dependent = D.idt_dependent
INNER JOIN Department AS DE ON DE.idt_department = E.idt_department
ORDER BY DE.name_department
```

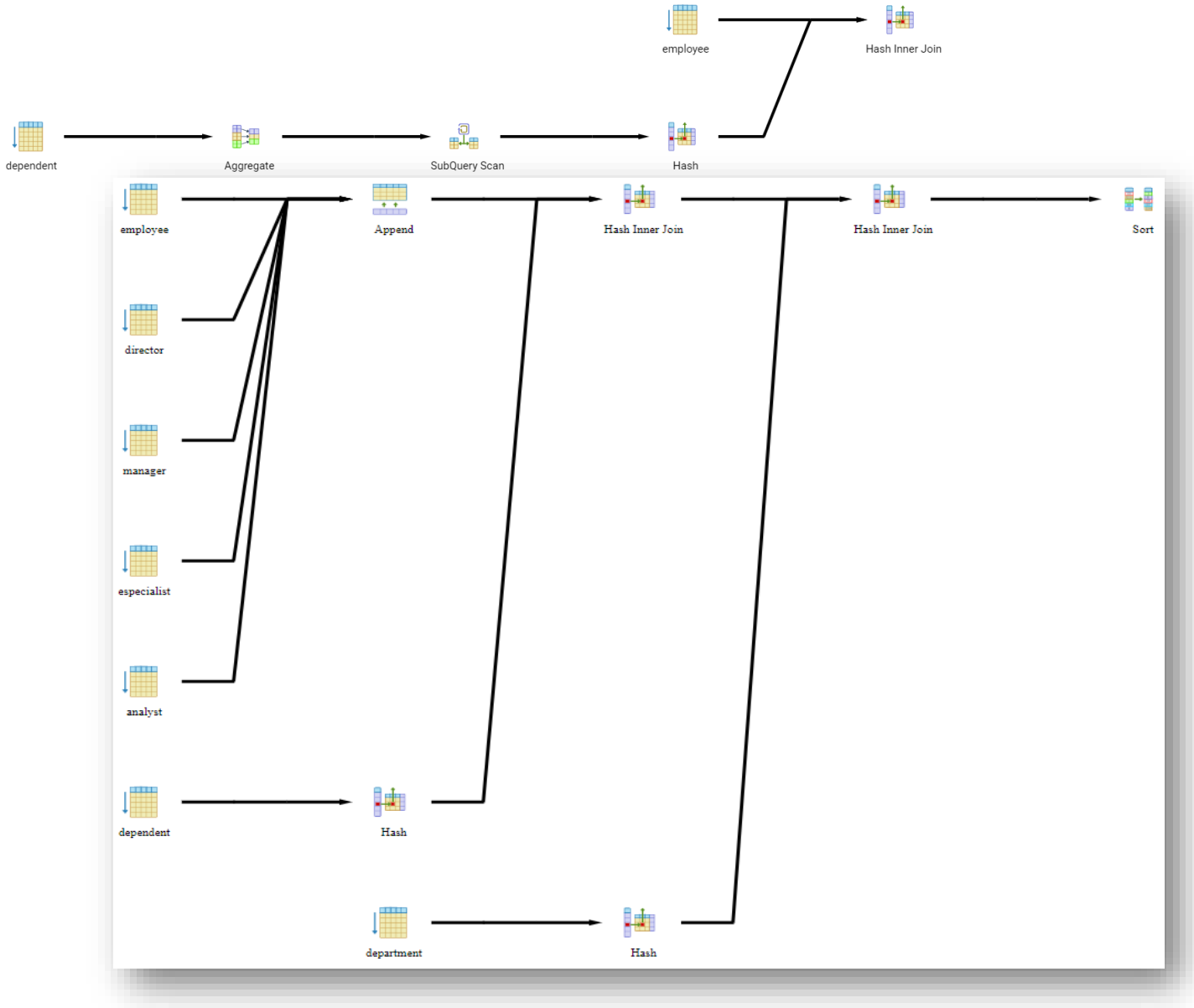
Consulta alterada e executada com o comando EXPLAIN



	QUERY PLAN text
1	Hash Join (cost=32.01..51.11 rows=67 width=32)
2	Hash Cond: (e.idt_employee = d.idt_employee)
3	-> Seq Scan on employee e (cost=0.00..17.20 rows=720 width=36)
4	-> Hash (cost=31.17..31.17 rows=67 width=4)
5	-> Subquery Scan on d (cost=28.00..31.17 rows=67 width=4)
6	-> HashAggregate (cost=28.00..30.50 rows=67 width=12)
7	Group Key: dependent.idt_employee
8	Filter: (count(*) > 2)
9	-> Seq Scan on dependent (cost=0.00..22.00 rows=1200 width=4)

QUERY PLAN text
Sort (cost=213.99..217.39 rows=1361 width=96)
Sort Key: de.name_department
-> Hash Join (cost=75.58..143.15 rows=1361 width=96)
Hash Cond: (e.idt_department = de.idt_department)
-> Hash Join (cost=38.58..102.56 rows=1361 width=68)
Hash Cond: (e.idt_dependent = d.idt_dependent)
-> Append (cost=0.00..60.41 rows=1361 width=40)
-> Seq Scan on employee e (cost=0.00..0.00 rows=1 width=40)
-> Seq Scan on director e_1 (cost=0.00..13.40 rows=340 width=40)
-> Seq Scan on manager e_2 (cost=0.00..13.40 rows=340 width=40)
-> Seq Scan on especialist e_3 (cost=0.00..13.40 rows=340 width=40)
-> Seq Scan on analyst e_4 (cost=0.00..13.40 rows=340 width=40)
-> Hash (cost=22.70..22.70 rows=1270 width=36)
-> Seq Scan on dependent d (cost=0.00..22.70 rows=1270 width=36)
-> Hash (cost=22.00..22.00 rows=1200 width=36)
-> Seq Scan on department de (cost=0.00..22.00 rows=1200 width=36)

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 de execução para a consulta realizada na “Parte 1” do trabalho seleciona o empregado que possui mais de dois dependentes. Apresenta a busca sequencial na tabela principal “Employee”, e possui uma função de agregação na “SubQuery” (comentado anteriormente no relatório da “Query 3”) através da cláusula “WHERE”. Se a tabela tiver um volume maior de dados a busca sequencial não seria uma vantagem, aqui ela conta com a ajuda da cláusula “WHERE” o que diminui o custo dessa busca, mas ainda assim ,no SGBD PostgreSQL, com base nas estatísticas que o comando “EXPLAIN” nos retorna, torna-se difícil validar se realmente esse custo irá diminuir, e, como comentado acima, se essas tabelas tivessem um volume de dados muito grande o processo seria ainda seria muito custoso e demoraria muito tempo para ser executado.

Após a alteração do artefato A, e da inclusão da herança, o PostgreSQL não permitia a adição da chave estrangeira derivada do “Pai” de uma herança. Ou seja, “Dependent” já não poderia mais satisfazer uma das solicitações da “Parte 1” do trabalho, de possuir ao menos uma entidade-fraca. Nesse caso, realizamos ajustes para que a tabela “Team” continue cumprindo com o requisito.

Visto as alterações realizadas em consultas anteriores, tentamos alterar coisas já observadas anteriormente. Como na primeira consulta é realizado um filtro, derivado de uma “SUBQUERY”, também comentada anteriormente, optamos por modificações que tornassem a consulta simples e que tirasse características já observadas.

Feita as alterações, a segunda consulta retorna o nome do dependente, o nome do empregado e o nome do departamento (e ordenado por este último) por meio de junções (“joins”). Desse ela realiza uma busca sequencial (“seq scan”) por cada tabela que herda a tabela “Employee”, utilizando o campo “idt_employee”. Por mais que tenhamos tirado subconsultas e filtros, o custo diminuía conforme execução e é possível notar que o maior “impacto”, na segunda consulta em comparativo com a primeira consulta, foi o número de linhas afetadas. A janela também teve um aumento, provavelmente não tão significativo, visto que a quantidade de linhas afetadas mais que “triplicou”.