e) (artefato do tipo texto) Projete 1 consulta que faça uso da estratégia de subconsulta (subquery). Se uma de suas consultas usadas nos itens anteriores atende a esse quesito, você pode usá-la. Reescreva-a sem usar essa estratégia. Execute as 2 consultas criadas e proceda com a análise dos custos e planos (os mesmos itens de relatórios e prints do item (b) devem ser expostos aqui para fins de comparação). O texto deve ser entregue em formato PDF. O código deverá ser entregue em formato ASCII (.txt) na plataforma adotada pelo professor da sua turma (e-disciplina) (ver observação em vermelho abaixo).

(os mesmos itens de relatórios e prints do item (b) fazer interpretação e relação com teoria

Consulta com subquery

"QUERY PLAN"
"Nested Loop (cost=0.00..15299.25 rows=185000 width=162)"
" Join Filter: (historico.valorpago > (SubPlan 1))"
" -> Seq Scan on historico (cost=0.00..17.50 rows=750 width=80)"
" -> Materialize (cost=0.00..21.10 rows=740 width=82)"
"   -> Seq Scan on produto (cost=0.00..17.40 rows=740 width=82)"
" SubPlan 1"
"   -> Result (cost=0.00..0.01 rows=1 width=32)"
"   One-Time Filter: ((NOT produto.devolvido) AND (produto.idproduto = historico.idproduto))"

Interpretação

One-Time Filter: ((NOT produto.devolvido) AND (produto.idproduto = historico.idproduto))"

Filtro produto.devolvido = false ^ produto.idproduto = historico.idproduto

Result (cost=0.00..0.01 rows=1 width=32)

Resultado output tem custo total estimado 0.00 e custo total 0.01

SubPlan 1

As linhas acima são o subplano 1

Seq Scan on produto (cost=0.00..17.40 rows=740 width=82)

Scan sequencial em produto de custo total estimado 0.00 e custo total 17.40

Materialize (cost=0.00..21.10 rows=740 width=82)

No materialize o output das ações seguintes são materializadas na memória com custo total estimado de 0.00 e custo total de 21.10

Seq Scan on historico (cost=0.00..17.50 rows=750 width=80)

Scan sequencial em produto de custo total estimado 0.00 e custo total 17.50

Join Filter: (historico.valorpago > (SubPlan 1))

Filtro historico.valorpago > output de SubPlan 1

Nested Loop (cost=0.00..15299.25 rows=185000 width=162)

Loop com custo total estimado de 0.00 e custo total de 15299.25

Consulta sem subquery

```
explain select * from produto
join historico
on produto.idproduto = historico.idproduto where produto.devolvido = false and
historico.valorpago > produto.valor;

"QUERY PLAN"
"Hash Join  (cost=22.02..41.49 rows=125 width=162)"
"  Hash Cond: (historico.idproduto = produto.idproduto)"
"  Join Filter: (historico.valorpago > produto.valor)"
"  ->  Seq Scan on historico  (cost=0.00..17.50 rows=750 width=80)"
"  ->  Hash  (cost=17.40..17.40 rows=370 width=82)"
"        ->  Seq Scan on produto  (cost=0.00..17.40 rows=370 width=82)"
"              Filter: (NOT devolvido)"
```

Interpretação

Filter: (NOT devolvido)
Filtro devolvido = false
Seq Scan on produto (cost=0.00..17.40 rows=370 width=82)
Scan sequencial em produto de custo total estimado 0.00 e custo total 17.40
Hash (cost=17.40..17.40 rows=370 width=82)
Armazenamento em hash de custo total estimado de 17.40 e custo total 17.40
Join Filter: (historico.valorpago > produto.valor)
Filtro de função join em historico.valorpago > produto.valor
Hash Cond: (historico.idproduto = produto.idproduto)
Condição Hash em historico.idproduto = produto.idproduto
Hash Join (cost=22.02..41.49 rows=125 width=162)
Join utilizando Hash com custo total estimado 22.02 e custo total 41.49

Podemos observar que a consulta com o subquery tem custo total e número de linhas maior do que o sem subquery, a disparidade do custo sem subquery foi muito menor que a com subquery

Relação

O SBGD realiza a distribuição do hash entre os buckets para otimização.