



Universidade do Vale do Itajaí
Escola Politécnica
Curso de Ciência da Computação – Campus Itajaí
Banco de Dados II
Otimização

Trabalho - M1

Acadêmicos: Matheus Baron Lauritzen e Gustavo Baron Lauritzen

23/04/2024

▪ Questão 1:

-- Questão 1 não otimizada

```
EXPLAIN SELECT m.nome AS nome_medico, COUNT(c.id) AS quantidade_consultas
```

```
FROM medico m
```

```
LEFT JOIN consulta c ON m.id = c.id_medico
```

```
GROUP BY m.nome
```

```
ORDER BY nome_medico;
```

Plano de execução:

Data Output		Messages	Explain	×	Notifications
<div><div><div><div>≡+</div><div>📄</div><div>▼</div><div>📋</div><div>▼</div><div>🗑️</div><div>🗄️</div><div>⬇️</div><div>📈</div></div></div></div>					
	QUERY PLAN				🔒
	text				
1	Sort (cost=25.90..26.40 rows=200 width=226)				
2	Sort Key: m.nome				
3	-> HashAggregate (cost=16.26..18.26 rows=200 width=226)				
4	Group Key: m.nome				
5	-> Hash Left Join (cost=1.14..14.91 rows=270 width=222)				
6	Hash Cond: (m.id = c.id_medico)				
7	-> Seq Scan on medico m (cost=0.00..12.70 rows=270 width=222)				
8	-> Hash (cost=1.06..1.06 rows=6 width=8)				
9	-> Seq Scan on consulta c (cost=0.00..1.06 rows=6 width=8)				

Plano de execução:

Data Output		Messages	Explain X	Notifications
	QUERY PLAN			
	text			
1	Sort (cost=317.23..317.90 rows=270 width=226)			
2	Sort Key: m.nome			
3	-> Seq Scan on medico m (cost=0.00..306.32 rows=270 width=226)			
4	SubPlan 1			
5	-> Aggregate (cost=1.08..1.09 rows=1 width=8)			
6	-> Seq Scan on consulta c (cost=0.00..1.07 rows=1 width=0)			
7	Filter: (m.id = id_medico)			

Explicação da otimização:

Nesta consulta, substituímos o GROUP BY por uma subconsulta que conta o número de consultas para cada médico. Isso elimina a necessidade de agrupar os resultados, o que resultou em um plano de execução mais simples e eficiente.

- Questão 2:

-- Questão 2 não otimizada

EXPLAIN SELECT

e.nome AS nome_especialidade,

COUNT(c.id) AS quantidade_consultas

FROM

especialidade e

LEFT JOIN medico m ON e.id = m.id_especialidade

LEFT JOIN consulta c ON m.id = c.id_medico

GROUP BY

e.nome

ORDER BY

nome_especialidade;

Plano de execução:

Data Output

Messages

Explain

X

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

QUERY PLAN

text

🔒

1	Sort (cost=50.11..50.61 rows=200 width=126)
2	Sort Key: e.nome
3	-> HashAggregate (cost=40.47..42.47 rows=200 width=126)
4	Group Key: e.nome
5	-> Hash Right Join (cost=23.29..37.77 rows=540 width=122)
6	Hash Cond: (m.id_especialidade = e.id)
7	-> Hash Left Join (cost=1.14..14.91 rows=270 width=8)
8	Hash Cond: (m.id = c.id_medico)
9	-> Seq Scan on medico m (cost=0.00..12.70 rows=270 width=8)
10	-> Hash (cost=1.06..1.06 rows=6 width=8)
11	-> Seq Scan on consulta c (cost=0.00..1.06 rows=6 width=8)
12	-> Hash (cost=15.40..15.40 rows=540 width=122)
13	-> Seq Scan on especialidade e (cost=0.00..15.40 rows=540 width=122)

-- Questão 2 otimizada

EXPLAIN SELECT

```

e.nome AS nome_especialidade,
(
  SELECT COUNT(*)
  FROM medico m
  JOIN consulta c ON m.id = c.id_medico
  WHERE m.id_especialidade = e.id
) AS quantidade_consultas

```

FROM

```

especialidade e

```










ORDER BY

```

nome_especialidade;

```

Plano de execução:

Data Output	Messages	Explain	×	Notifications
				
				
	QUERY PLAN text			
1	Sort (cost=7882.06..7883.41 rows=540 width=126)			
2	Sort Key: e.nome			
3	-> Seq Scan on especialidade e (cost=0.00..7857.55 rows=540 width=126)			
4	SubPlan 1			
5	-> Aggregate (cost=14.51..14.52 rows=1 width=8)			
6	-> Nested Loop (cost=0.00..14.51 rows=1 width=0)			
7	Join Filter: (m.id = c.id_medico)			
8	-> Seq Scan on medico m (cost=0.00..13.38 rows=1 width=4)			
9	Filter: (id_especialidade = e.id)			
10	-> Seq Scan on consulta c (cost=0.00..1.06 rows=6 width=4)			

Explicação da otimização:

Nesta consulta, utilizamos uma subconsulta para contar diretamente o número de consultas para cada especialidade. Isso elimina a necessidade de múltiplas junções e agrupamentos, o que ajudou a simplificar o plano de execução.

▪ Questão 3:

-- Questão 3 não otimizada

EXPLAIN SELECT

TO_CHAR(dia, 'DD/MM') AS dia_mes,

(SELECT MAX(duracao)

FROM (SELECT EXTRACT(EPOCH FROM (data_hora_fim - data_hora_inicio)) AS duracao

FROM consulta c

WHERE DATE_TRUNC('day', c.data_hora_inicio) = dia) AS duracoes) AS maior_duracao

FROM

(SELECT DISTINCT DATE_TRUNC('day', data_hora_inicio) AS dia FROM consulta) AS dias

ORDER BY

dia_mes;

Plano de execução:

Data Output		Messages	Explain X	Notifications
<div> </div>				
	QUERY PLAN			
	text			
1	Sort (cost=7.90..7.92 rows=6 width=64)			
2	Sort Key: (to_char(dias.dia, 'DD/MM':text))			
3	-> Subquery Scan on dias (cost=1.09..7.83 rows=6 width=64)			
4	-> HashAggregate (cost=1.09..1.16 rows=6 width=8)			
5	Group Key: date_trunc('day':text, consulta.data_hora_inicio)			
6	-> Seq Scan on consulta (cost=0.00..1.07 rows=6 width=8)			
7	SubPlan 1			
8	-> Aggregate (cost=1.10..1.11 rows=1 width=32)			
9	-> Seq Scan on consulta c (cost=0.00..1.09 rows=1 width=16)			
10	Filter: (date_trunc('day':text, data_hora_inicio) = dias.dia)			

-- Questão 3 otimizada

EXPLAIN SELECT

TO_CHAR(data_hora_inicio, 'DD/MM') AS dia_mes,

MAX(EXTRACT(EPOCH FROM (data_hora_fim - data_hora_inicio))) AS maior_duracao

FROM

consulta

GROUP BY

TO_CHAR(data_hora_inicio, 'DD/MM')

ORDER BY

```
dia_mes;
```

Plano de execução:

Data Output

Messages

Explain ×

Notifications

QUERY PLAN

text

1

GroupAggregate (cost=1.15..1.30 rows=6 width=64)

2

Group Key: (to_char(data_hora_inicio, 'DD/MM'::text))

3

-> Sort (cost=1.15..1.17 rows=6 width=48)

4

Sort Key: (to_char(data_hora_inicio, 'DD/MM'::text))

5

-> Seq Scan on consulta (cost=0.00..1.07 rows=6 width=48)

Explicação da otimização:

Nesta consulta, estamos formatando a data de data_hora_inicio para exibir apenas o dia e o mês no formato 'DD/MM' usando a função TO_CHAR. Depois, utilizamos a função EXTRACT para calcular a diferença entre data_hora_fim e data_hora_inicio, convertendo o resultado para segundos utilizando EXTRACT(EPOCH FROM ...), e em seguida, usamos a função MAX para encontrar a maior duração para cada dia. Assim otimizamos a consulta com um código sql menor e mais aninhado, deixando o plano de execução menor.

▪ Questão 4:

--Questão 4 não otimizada

EXPLAIN SELECT

COUNT(*) AS quantidade_consultas,

e.nome AS nome_especialidade,

m.nome AS nome_medico

FROM

consulta c

INNER JOIN

medico m ON c.id_medico = m.id

INNER JOIN

especialidade e ON m.id_especialidade = e.id











GROUP BY

e.nome, m.nome

ORDER BY

LIMIT 1;

Plano de execução:

Data Output		Messages	Explain X	Notifications
		        		
	QUERY PLAN text 			
1	Limit (cost=17.21..17.21 rows=1 width=344)			
2	-> Sort (cost=17.21..17.22 rows=6 width=344)			
3	Sort Key: (count(*)) DESC			
4	-> GroupAggregate (cost=17.06..17.18 rows=6 width=344)			
5	Group Key: e.nome, m.nome			
6	-> Sort (cost=17.06..17.07 rows=6 width=336)			
7	Sort Key: e.nome, m.nome			
8	-> Nested Loop (cost=1.28..16.98 rows=6 width=336)			
9	-> Hash Join (cost=1.14..14.91 rows=6 width=222)			
10	Hash Cond: (m.id = c.id_medico)			
11	-> Seq Scan on medico m (cost=0.00..12.70 rows=270 width=226)			
12	-> Hash (cost=1.06..1.06 rows=6 width=4)			
13	-> Seq Scan on consulta c (cost=0.00..1.06 rows=6 width=4)			
14	-> Index Scan using especialidade_pkey on especialidade e (cost=0.15..0.35 rows=1 width=122)			
15	Index Cond: (id = m.id_especialidade)			

-- Questão 4 otimizada

EXPLAIN SELECT

```
(
    SELECT COUNT(*)
    FROM consulta c
    WHERE c.id_medico = m.id
) AS quantidade_consultas,
e.nome AS nome_especialidade,
m.nome AS nome_medico
```

FROM

medico m

INNER JOIN

especialidade e ON m.id_especialidade = e.id

ORDER BY

quantidade_consultas DESC

LIMIT 1;

Plano de execução:

Data Output

Messages

Explain X

Notifications

QUERY PLAN

text

1	Limit (cost=330.53..330.54 rows=1 width=344)
2	-> Sort (cost=330.53..331.21 rows=270 width=344)
3	Sort Key: ((SubPlan 1)) DESC
4	-> Hash Join (cost=22.15..329.18 rows=270 width=344)
5	Hash Cond: (m.id_especialidade = e.id)
6	-> Seq Scan on medico m (cost=0.00..12.70 rows=270 width=226)
7	-> Hash (cost=15.40..15.40 rows=540 width=122)
8	-> Seq Scan on especialidade e (cost=0.00..15.40 rows=540 width=122)
9	SubPlan 1
10	-> Aggregate (cost=1.08..1.09 rows=1 width=8)
11	-> Seq Scan on consulta c (cost=0.00..1.07 rows=1 width=0)
12	Filter: (id_medico = m.id)

Explicação da otimização:

Estamos usando uma subconsulta para contar o número de consultas para cada médico na tabela consulta. Em seguida, juntamos os resultados com as tabelas “medico” e “especialidade” para obter o nome do médico e da especialidade correspondentes. Finalmente, ordenamos os resultados pelo número de consultas em ordem decrescente e limitamos o resultado a apenas 1 registro, que será o médico que realizou o maior número de consultas. Esta abordagem simplificou o plano de execução, já que evita a necessidade de junções adicionais e gregações de grupos.

▪ Questão 5:

--Questão 5 não otimizada

EXPLAIN SELECT

pa.nome AS paciente,

me.nome AS medico,

es.nome AS especialidade,

c.data_hora_inicio AS data

FROM

consulta c

INNER JOIN

medico me ON c.id_medico = me.id

INNER JOIN

especialidade es ON me.id_especialidade = es.id

INNER JOIN

paciente pa ON c.id_paciente = pa.id

ORDER BY

data;

Plano de execução:

Data Output Messages Explain X Notifications	
	QUERY PLAN text
1	Sort (cost=30.90..30.92 rows=6 width=562)
2	Sort Key: c.data_hora_inicio
3	-> Nested Loop (cost=15.13..30.83 rows=6 width=562)
4	-> Hash Join (cost=14.98..28.76 rows=6 width=448)
5	Hash Cond: (pa.id = c.id_paciente)
6	-> Seq Scan on paciente pa (cost=0.00..12.70 rows=270 width=222)
7	-> Hash (cost=14.91..14.91 rows=6 width=234)
8	-> Hash Join (cost=1.14..14.91 rows=6 width=234)
9	Hash Cond: (me.id = c.id_medico)
10	-> Seq Scan on medico me (cost=0.00..12.70 rows=270 width=226)
11	-> Hash (cost=1.06..1.06 rows=6 width=16)
12	-> Seq Scan on consulta c (cost=0.00..1.06 rows=6 width=16)
13	-> Index Scan using especialidade_pkey on especialidade es (cost=0.15..0.35 rows=1 width=122)
14	Index Cond: (id = me.id_especialidade)

-- Questão 5 otimizada

EXPLAIN SELECT

(

SELECT nome

```
FROM paciente
WHERE id = c.id_paciente
) AS paciente,
(
SELECT nome
FROM medico
WHERE id = c.id_medico
) AS medico,
(
SELECT nome
FROM especialidade
WHERE id = (
SELECT id_especialidade
FROM medico
WHERE id = c.id_medico
)
) AS especialidade,
c.data_hora_inicio AS data
FROM
consulta c
ORDER BY data;
```

Plano de execução:

Data Output Messages Explain X Notifications	
<div> <div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div> </div>	
	<div> <div>QUERY PLAN</div> <div>text</div> <div>🔒</div> </div>
1	Sort (cost=197.11..197.13 rows=6 width=562)
2	Sort Key: c.data_hora_inicio
3	-> Seq Scan on consulta c (cost=0.00..197.04 rows=6 width=562)
4	SubPlan 1
5	-> Index Scan using paciente_pkey on paciente (cost=0.15..8.17 rows=1 width=218)
6	Index Cond: (id = c.id_paciente)
7	SubPlan 2
8	-> Index Scan using medico_pkey on medico (cost=0.15..8.17 rows=1 width=218)
9	Index Cond: (id = c.id_medico)
10	SubPlan 4
11	-> Index Scan using especialidade_pkey on especialidade (cost=8.32..16.33 rows=1 width=118)
12	Index Cond: (id = \$3)
13	InitPlan 3 (returns \$3)
14	-> Index Scan using medico_pkey on medico medico_1 (cost=0.15..8.17 rows=1 width=4)
15	Index Cond: (id = c.id_medico)

Explicação da otimização:

Nesta consulta, substituímos as junções explícitas com subconsultas para buscar os nomes do paciente, médico e especialidade diretamente das tabelas correspondentes. Isso ajudou a reduzir a complexidade do plano de execução.