

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

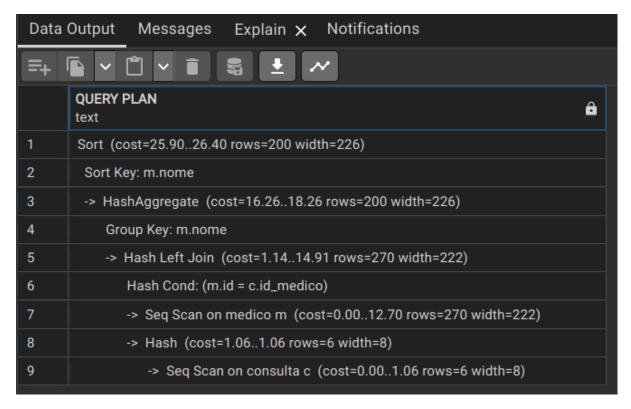
FROM medico m

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

GROUP BY m.nome

ORDER BY nome_medico;

Plano de execução:



--Questão 1 otimizada

```
EXPLAIN SELECT
```

```
m.nome AS nome_medico,
(

SELECT COUNT(*)

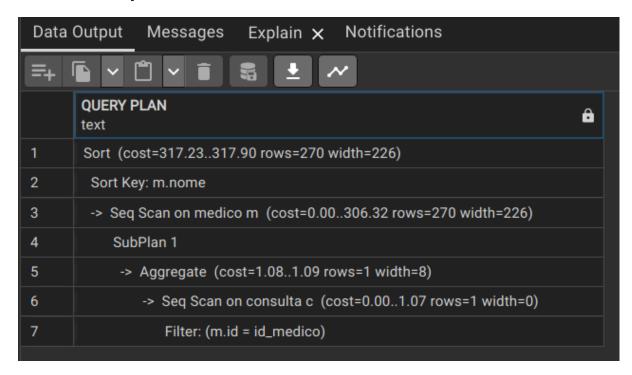
FROM consulta c

WHERE m.id = c.id_medico
) AS quantidade_consultas
```

FROM medico m

ORDER BY nome_medico;

Plano de execução:



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

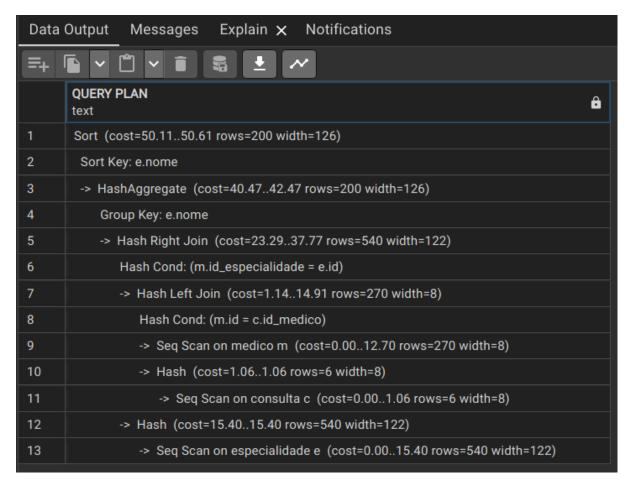
LEFT JOIN consulta c ON m.id = c.id_medico

GROUP BY

e.nome

ORDER BY

nome_especialidade;



-- 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

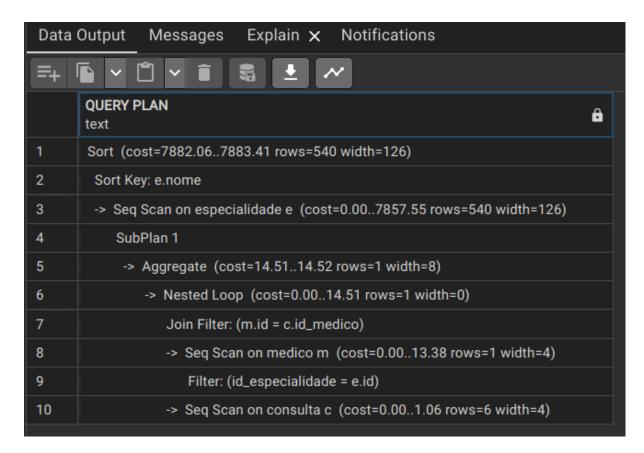
) AS quantidade_consultas

FROM

especialidade e

ORDER BY

nome_especialidade;
```



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

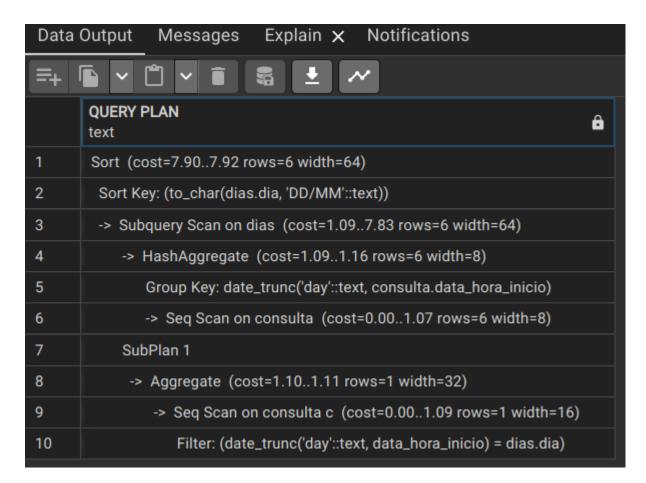
WHERE DATE_TRUNC('day', c.data_hora_inicio) = dia) AS duracoes) AS maior_duracao

FROM

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

ORDER BY

dia_mes;



-- 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

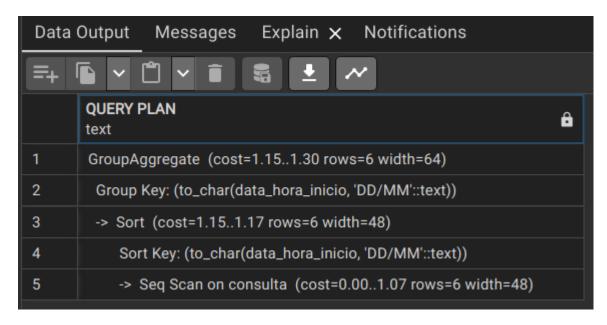
consulta

GROUP BY

TO_CHAR(data_hora_inicio, 'DD/MM')

ORDER BY

dia_mes;



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

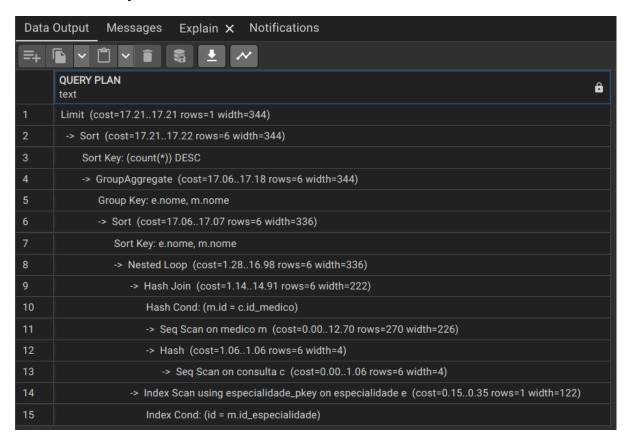
GROUP BY
e.nome, m.nome

ORDER BY
```

```
quantidade_consultas DESC
```

LIMIT 1;

Plano de execução:



```
-- Questão 4 otimizada
```

ORDER BY

```
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

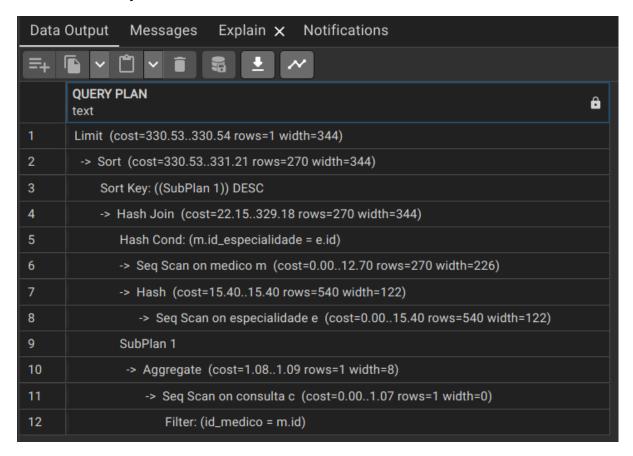
medico m

INNER JOIN

especialidade e ON m.id_especialidade = e.id
```

LIMIT 1;

Plano de execução:



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

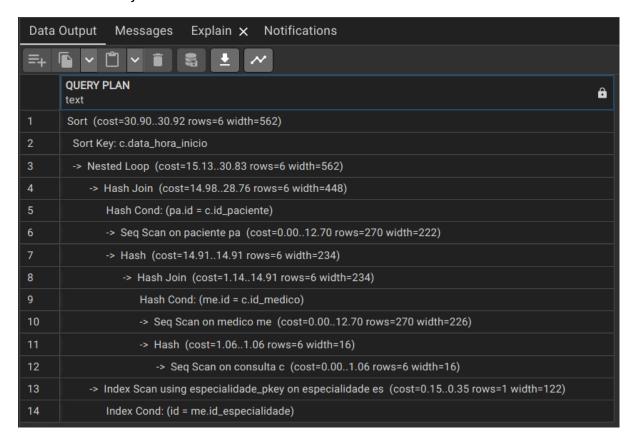
especialidade es ON me.id_especialidade = es.id

INNER JOIN

paciente pa ON c.id_paciente = pa.id

ORDER BY

data:
```



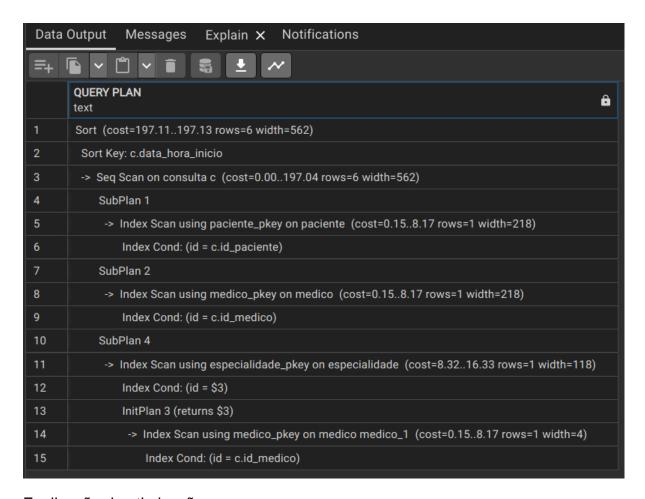
```
-- Questão 5 otimizada

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;
```



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.