Consulta 1 (original)

```
SELECT A.CPF, A.Nome, B.Nome
FROM Funcionarios A, Clientes B, Aluguel C, Funcionarios D
WHERE A.CPF = B.CPF
  AND B.CPF = C.CPF_Cliente
  AND B.Sexo = "M"
  AND C.ValorPagar > 50
  AND A.CPF = D.CPF_Supervisor;
```

1. Árvore de Consulta Inicial (Não Otimizada)

a) Parsing da Esquerda para a Direita

A ordem da cláusula FROM é: Funcionarios A → Clientes B → Aluguel C → Funcionarios D

Árvore de consulta (não otimizada):

π A.CPF, A.Nome, B.Nome

|

σ B.Sexo = "M" AND C.ValorPagar > 50 AND A.CPF = B.CPF AND B.CPF = C.CPF_Cliente AND A.CPF = D.CPF_Supervisor

|

⋈ (A × B × C × D)

b) Parsing da Direita para a Esquerda

Ordem da cláusula FROM (reversa): Funcionarios D → Aluguel C → Clientes B → Funcionarios A

Árvore de consulta (não otimizada):

π A.CPF, A.Nome, B.Nome

|

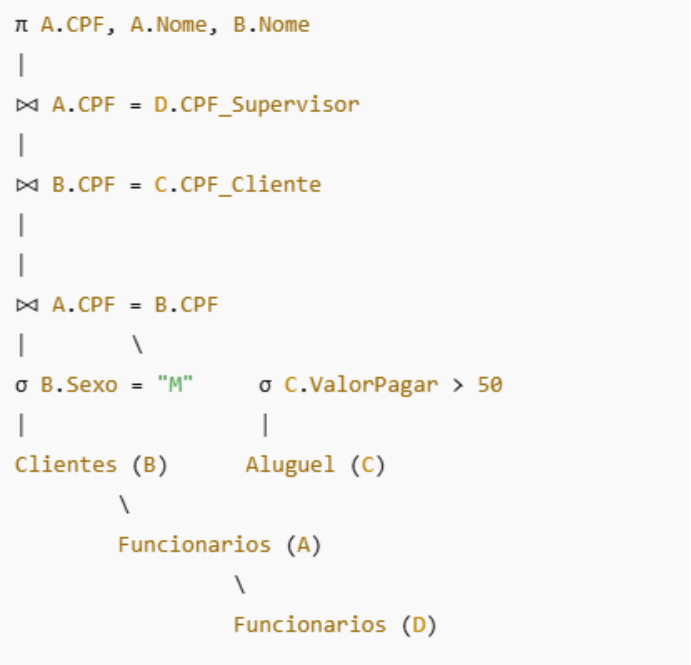
σ B.Sexo = "M" AND C.ValorPagar > 50 AND A.CPF = B.CPF AND B.CPF = C.CPF_Cliente AND A.CPF = D.CPF_Supervisor

|

⋈ (D × C × B × A)

2. Árvore de Consulta Otimizada

Árvore otimizada:



3. Consulta Reescrita Otimizada

a) Parsing da Esquerda para a Direita

SELECT A.CPF, A.Nome, B.Nome

FROM Funcionarios A

JOIN Clientes B ON A.CPF = B.CPF

JOIN Aluguel C ON B.CPF = C.CPF_Cliente

JOIN Funcionarios D ON A.CPF = D.CPF_Supervisor

WHERE B.Sexo = "M"

AND C.ValorPagar > 50;

b) Parsing da Direita para a Esquerda

SELECT A.CPF, A.Nome, B.Nome

FROM Funcionarios D

JOIN Funcionarios A ON A.CPF = D.CPF_Supervisor

JOIN Aluguel C ON A.CPF = C.CPF_Funcionario

JOIN Clientes B ON A.CPF = B.CPF AND B.CPF = C.CPF_Cliente

WHERE B.Sexo = "M"

AND C.ValorPagar > 50;

4. Plano de Execução Otimizado

Bloco = 2KB (2048B); ponteiro = 16B

σ B.Sexo = "M"

- Clientes: 100.000 registros, 1/2 devem ser homens → 50.000 resultados.
- Registro ≈ 16+11+200+16+12+1 = 256B → 8 registros por bloco.
- Acesso total ≈ 100.000 / 8 = 12.500 blocos.

σ C.ValorPagar > 50

- Aluguel: 20.000.000 registros. Vamos assumir 25% têm valor > 50 → 5.000.000.
- Registro ≈ 12+10+24+12+11 = 69B → 29 por bloco → ≈ 689.655 blocos.

Junção A.CPF = B.CPF

- Índice secundário em `B.CPF`.
- 50.000 registros de B → acessos via índice → desprezível custo de busca.

Junção B.CPF = C.CPF_Cliente

- Índice secundário em `CPF_Cliente` de C.
- 50.000 buscas → cada uma atinge o ISAM → vamos estimar 2 blocos por busca → 100.000 blocos.

Junção A.CPF = D.CPF_Supervisor

- Funcionarios: 3.500 registros. Registro ≈ 11+160 = 171B → 11 por bloco → 319 blocos.
- Para cada A, procurar subordinado em D. Supondo poucos subordinados → 2 blocos por A.
- 50.000 registros de A → 100.000 blocos.

Consulta 2

```
SELECT A.Nome, C.Nome
FROM Filmes A, AtoresEmFilmes B, Atores C, Midias D
WHERE A.Codigo = B.CodFilme
  AND B.CodAtor = C.Codigo
  AND A.Genero = "Aventura"
  AND A.Codigo = D.CodFilme
  AND D.PrecoDiaria > 10
```

Parsing natural (esquerda → direita):

```
π A.Nome, C.Nome
  σ A.Genero = "Aventura" ∧ D.PrecoDiaria > 10
    ⋈ A.Codigo = D.CodFilme
      ⋈ B.CodAtor = C.Codigo
        ⋈ A.Codigo = B.CodFilme
          Filmes A, AtoresEmFilmes B, Atores C, Midias D
```

Parsing reverso (direita → esquerda):

```
π A.Nome, C.Nome
  σ A.Genero = "Aventura" ∧ D.PrecoDiaria > 10
    ⋈ A.Codigo = D.CodFilme
      ⋈ B.CodAtor = C.Codigo
        ⋈ A.Codigo = B.CodFilme
          Midias D, Atores C, AtoresEmFilmes B, Filmes A
```

Árvore de Consulta Otimizada

```
π A.Nome, C.Nome
  ⋈ A.Codigo = D.CodFilme
    σ D.PrecoDiaria > 10
      Midias D
```

⋈ A.Codigo = B.CodFilme

σ A.Genero = "Aventura"

Filmes A

⋈ B.CodAtor = C.Codigo

AtoresEmFilmes B

Atores C

Parsing natural (esquerda → direita):

```
SELECT A.Nome, C.Nome
FROM (
   SELECT * FROM Filmes WHERE Genero = 'Aventura'
) A
JOIN AtoresEmFilmes B ON A.Codigo = B.CodFilme
JOIN Atores C ON B.CodAtor = C.Codigo
JOIN (
   SELECT * FROM Midias WHERE PrecoDiaria > 10
) D ON A.Codigo = D.CodFilme
```

Parsing reverso (direita → esquerda):

```
SELECT A.Nome, C.Nome
FROM (
   SELECT * FROM Midias WHERE PrecoDiaria > 10
) D
JOIN (
   SELECT * FROM Filmes WHERE Genero = 'Aventura'
) A ON A.Codigo = D.CodFilme
JOIN AtoresEmFilmes B ON A.Codigo = B.CodFilme
JOIN Atores C ON B.CodAtor = C.Codigo
```

Consulta 3:

```
SELECT A.CPF, A.Nome, B.Nome
FROM Funcionarios A, Clientes B, Aluguel C, Pagamentos D
WHERE A.CPF = B.CPF
  AND C.ValorPagar > 100
  AND B.CPF = C.CPF_Cliente
  AND D.Valor < 50
  AND A.CPF_Supervisor IS NULL
  AND A.CPF = C.CPF_Funcionario
```

Parsing natural (esquerda → direita):

π A.CPF, A.Nome, B.Nome

  σ C.ValorPagar > 100 ∧ D.Valor < 50 ∧ A.CPF_Supervisor IS NULL

    ⋈ A.CPF = C.CPF_Funcionario

      ⋈ A.CPF = B.CPF

        ⋈ B.CPF = C.CPF_Cliente

          ⋈ C.ID_Midia = D.ID_Midia

            Funcionarios A, Clientes B, Aluguel C, Pagamentos D

Parsing reverso (direita → esquerda):

π A.CPF, A.Nome, B.Nome

  σ C.ValorPagar > 100 ∧ D.Valor < 50 ∧ A.CPF_Supervisor IS NULL

    ⋈ A.CPF = C.CPF_Funcionario

      ⋈ A.CPF = B.CPF

        ⋈ B.CPF = C.CPF_Cliente

          ⋈ C.ID_Midia = D.ID_Midia

            Pagamentos D, Aluguel C, Clientes B, Funcionarios A

Árvore de Consulta Otimizada

π A.CPF, A.Nome, B.Nome

⋈ A.CPF = C.CPF_Funcionario

σ A.CPF_Supervisor IS NULL

Funcionarios A

⋈ A.CPF = B.CPF

Clientes B

⋈ B.CPF = C.CPF_Cliente

⋈ C.ID_Midia = D.ID_Midia

σ D.Valor < 50

Pagamentos D

σ C.ValorPagar > 100

Aluguel C

Parsing natural (esquerda → direita):

```
SELECT A.CPF, A.Nome, B.Nome
FROM (
   SELECT * FROM Funcionarios WHERE CPF_Supervisor IS NULL
) A
JOIN Clientes B ON A.CPF = B.CPF
JOIN (
   SELECT * FROM Aluguel WHERE ValorPagar > 100
) C ON B.CPF = C.CPF_Cliente AND A.CPF = C.CPF_Funcionario
JOIN (
   SELECT * FROM Pagamentos WHERE Valor < 50
) D ON C.ID_Midia = D.ID_Midia
```

Parsing reverso (direita → esquerda):

```
SELECT A.CPF, A.Nome, B.Nome
FROM (
   SELECT * FROM Pagamentos WHERE Valor < 50
) D
JOIN (
   SELECT * FROM Aluguel WHERE ValorPagar > 100
) C ON C.ID_Midia = D.ID_Midia
JOIN Clientes B ON B.CPF = C.CPF_Cliente
JOIN (
   SELECT * FROM Funcionarios WHERE CPF_Supervisor IS NULL
) A ON A.CPF = B.CPF AND A.CPF = C.CPF_Funcionario
```