



longchunks,smallcode ,230 0 00 0 0,

O Elefante arborícola

Consultas recursivas com expressões comuns de tabelas

Leandro Guimarães Faria Corcete DUTRA

¹Comunidade Brasileira de PostgreSQL

²ArsData

III Conferência brasileira de PostgreSQL (2009)

▶ Expanda sua mente!



- ▶ Expanda sua mente!
- ▶ ...com:

- ▶ Expanda sua mente!
- ▶ ...com:
 - ▶ árvores,



- ▶ Expanda sua mente!
- ▶ ...com:
 - ▶ árvores,
 - ▶ grafos,



- ▶ Expanda sua mente!
- ▶ ...com:
 - ▶ árvores,
 - ▶ grafos,
 - ▶ recursão!

Besteirol

- ▶ Bobagens:

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados
 - ▶ Prevayler

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados
 - ▶ Prevayler
 - ▶ LDAP

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados
 - ▶ Prevayler
 - ▶ LDAP
 - ▶ ...embora possam ter seus usos

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados
 - ▶ Prevayler
 - ▶ LDAP
 - ▶ ...embora possam ter seus usos
- ▶ Mas:

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados
 - ▶ Prevayler
 - ▶ LDAP
 - ▶ ...embora possam ter seus usos
- ▶ Mas:
 - ▶ SQL não é perfeito

Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados
 - ▶ Prevayler
 - ▶ LDAP
 - ▶ ...embora possam ter seus usos
- ▶ Mas:
 - ▶ SQL não é perfeito
 - ▶ Ainda não temos solução ideal



Besteirol

- ▶ Bobagens:
 - ▶ NoSQL
 - ▶ SGBDs OO
 - ▶ SGBDs multivalorados
 - ▶ Prevayler
 - ▶ LDAP
 - ▶ ...embora possam ter seus usos
- ▶ Mas:
 - ▶ SQL não é perfeito
 - ▶ Ainda não temos solução ideal
 - ▶ SQL é bom o suficiente

Problemas aparentes

- ▶ Dificuldade de aprendizado

Problemas aparentes

- ▶ Dificuldade de aprendizado
 - ▶ Não OO



Problemas aparentes

- ▶ Dificuldade de aprendizado
 - ▶ Não OO
 - ▶ Não parecido com C, Java &c



Problemas aparentes

- ▶ Dificuldade de aprendizado
 - ▶ Não OO
 - ▶ Não parecido com C, Java &c
 - ▶ Computacionalmente incompleto



Problemas aparentes

- ▶ Dificuldade de aprendizado
 - ▶ Não OO
 - ▶ Não parecido com C, Java &c
 - ▶ Computacionalmente incompleto
- ▶ Não lidar com dados 'ricos' ou 'desestruturados'



Problemas aparentes

- ▶ Dificuldade de aprendizado
 - ▶ Não OO
 - ▶ Não parecido com C, Java &c
 - ▶ Computacionalmente incompleto
- ▶ Não lidar com dados 'ricos' ou 'desestruturados'
- ▶ Não lidar com hierarquias



Problemas aparentes

- ▶ Dificuldade de aprendizado
 - ▶ Não OO
 - ▶ Não parecido com C, Java &c
 - ▶ Computacionalmente incompleto
- ▶ Não lidar com dados 'ricos' ou 'desestruturados'
- ▶ Não lidar com hierarquias
- ▶ ...



Problemas aparentes

- ▶ Dificuldade de aprendizado
 - ▶ Não OO
 - ▶ Não parecido com C, Java &c
 - ▶ Computacionalmente incompleto
- ▶ Não lidar com dados 'ricos' ou 'desestruturados'
- ▶ Não lidar com hierarquias
- ▶ ...
- ▶ Mentiras! :-)

Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO
 - ▶ SQL não é única implementação relacional possível



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO
 - ▶ SQL não é única implementação relacional possível
 - ▶ Dados ricos ou desestruturados são ortogonais (tipos)



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO
 - ▶ SQL não é única implementação relacional possível
 - ▶ Dados ricos ou desestruturados são ortogonais (tipos)
- ▶ SQL é computacionalmente completo



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO
 - ▶ SQL não é única implementação relacional possível
 - ▶ Dados ricos ou desestruturados são ortogonais (tipos)
- ▶ SQL é computacionalmente completo
 - ▶ desde o ISO SQL:2008...



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO
 - ▶ SQL não é única implementação relacional possível
 - ▶ Dados ricos ou desestruturados são ortogonais (tipos)
- ▶ SQL é computacionalmente completo
 - ▶ desde o ISO SQL:2008...
 - ▶ ...e o PostgreSQL, desde a versão 8.4...



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO
 - ▶ SQL não é única implementação relacional possível
 - ▶ Dados ricos ou desestruturados são ortogonais (tipos)
- ▶ SQL é computacionalmente completo
 - ▶ desde o ISO SQL:2008...
 - ▶ ...e o PostgreSQL, desde a versão 8.4...
 - ▶ ...graças à recursividade (SQL99) e funções de janela



Falsos problemas

- ▶ Forma mais simples já imaginada de lidar com dados
 - ▶ Modelo relacional é ortogonal com OO
 - ▶ SQL não é única implementação relacional possível
 - ▶ Dados ricos ou desestruturados são ortogonais (tipos)
- ▶ SQL é computacionalmente completo
 - ▶ desde o ISO SQL:2008...
 - ▶ ...e o PostgreSQL, desde a versão 8.4...
 - ▶ ...graças à recursividade (SQL99) e funções de janela
 - ▶ Recursividade nos dá hierarquia, grafos &c!

¿Soluções para hierarquias?

- ▶ Processamento externo



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade

¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)
 - ▶ Caro na escrita



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)
 - ▶ Caro na escrita
 - ▶ Complicado



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)
 - ▶ Caro na escrita
 - ▶ Complicado
 - ▶ Estúpido

¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)
 - ▶ Caro na escrita
 - ▶ Complicado
 - ▶ Estúpido
- ▶ Oracle CONNECT BY



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)
 - ▶ Caro na escrita
 - ▶ Complicado
 - ▶ Estúpido
- ▶ Oracle CONNECT BY
 - ▶ Antigo, e implementado no contrib

¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)
 - ▶ Caro na escrita
 - ▶ Complicado
 - ▶ Estúpido
- ▶ Oracle CONNECT BY
 - ▶ Antigo, e implementado no contrib
 - ▶ Incomum, e obsoleto no Oracle 11gr2



¿Soluções para hierarquias?

- ▶ Processamento externo
 - ▶ Caro em recursos — escalabilidade
 - ▶ Não relacional
- ▶ Enumeração de caminhos materializados
- ▶ Funções & procedimentos armazenados
- ▶ Conjuntos aninhados (nested sets)
 - ▶ Caro na escrita
 - ▶ Complicado
 - ▶ Estúpido
- ▶ Oracle CONNECT BY
 - ▶ Antigo, e implementado no contrib
 - ▶ Incomum, e obsoleto no Oracle 11gr2
- ▶ Recursividade: expressões comuns de tabelas.

Recursividade

- ▶ Solução correta para o problema da hierarquia



Recursividade

- ▶ Solução correta para o problema da hierarquia
- ▶ Solução genérica: grafos &c



Recursividade

- ▶ Solução correta para o problema da hierarquia
- ▶ Solução genérica: grafos &c
- ▶ Interessante: conjuntos de Mandelbrot, caixeiro viajante &c



Recursividade

- ▶ Solução correta para o problema da hierarquia
- ▶ Solução genérica: grafos &c
- ▶ Interessante: conjuntos de Mandelbrot, caixeiro viajante &c
- ▶ Relacional (na medida do possível em SQL)



Recursividade

- ▶ Solução correta para o problema da hierarquia
- ▶ Solução genérica: grafos &c
- ▶ Interessante: conjuntos de Mandelbrot, caixeiro viajante &c
- ▶ Relacional (na medida do possível em SQL)
- ▶ Necessária: computação sem recursividade é manca

Recursividade com expressões comuns de tabelas

- ▶ Padrão ISO SQL:1999

Recursividade com expressões comuns de tabelas

- ▶ Padrão ISO SQL:1999
 - ▶ pronunciado como a solução desde 1997 ao menos

Recursividade com expressões comuns de tabelas

- ▶ Padrão ISO SQL:1999
 - ▶ renunciado como a solução desde 1997 ao menos
 - ▶ relacional, ao menos em espírito e na prática

Recursividade com expressões comuns de tabelas

- ▶ Padrão ISO SQL:1999
 - ▶ renunciado como a solução desde 1997 ao menos
 - ▶ relacional, ao menos em espírito e na prática
 - ▶ veloz, econômico...



Recursividade com expressões comuns de tabelas

- ▶ Padrão ISO SQL:1999
 - ▶ renunciado como a solução desde 1997 ao menos
 - ▶ relacional, ao menos em espírito e na prática
 - ▶ veloz, econômico...
 - ▶ e (relativamente) simples



Recursividade com expressões comuns de tabelas

- ▶ Padrão ISO SQL:1999
 - ▶ renunciado como a solução desde 1997 ao menos
 - ▶ relacional, ao menos em espírito e na prática
 - ▶ veloz, econômico...
 - ▶ e (relativamente) simples
- ▶ Implementação no PostgreSQL

Recursividade com expressões comuns de tabelas

- ▶ Padrão ISO SQL:1999
 - ▶ renunciado como a solução desde 1997 ao menos
 - ▶ relacional, ao menos em espírito e na prática
 - ▶ veloz, econômico...
 - ▶ e (relativamente) simples
- ▶ Implementação no PostgreSQL
 - ▶ Comitê ISO SQL, YOSHIYUKI Asaba, ISHII Tatsuo, Jeff DAVIS, Gregory STARK, Tom LANE, David FETTER, comunidade japonesa



Expressões de tabelas: WITH

Sintaxe:

```
WITH
apelido
[(lista_tributos)]
  AS
(consulta) [, ...]
  SELECT
lista_projeção
  FROM
apelido [, ...] ...
;
```

A ordem em que aparecem os apelidos é essencial: um apelido somente pode ser invocado *após* ter sido definido.





Exemplo simples

```
WITH subconsulta
  AS
  (SELECT random () AS aleatório
   FROM generate_series (1, 3)
  )
SELECT aleatório
  FROM subconsulta
UNION
SELECT aleatório
  FROM subconsulta
;
0.249934420455247
0.869007148314267
0.181567670311779
```



Exemplo complexo

```
CREATE TABLE
pedidos
(
  região CHAR (1),
  produto CHAR (1),
  dia DATE,
  quantidade NUMERIC,
  valor MONEY,
CONSTRAINT cp_pedidos PRIMARY KEY
( região ,
  produto ,
    dia )
)
;
```



Exemplo

Valores:

```
INSERT INTO
pedidos
(região , produto , dia , unidades , valor)
VALUES
('A' , 'a' , '2009 10 23' , '1' , '1' ) ,
('A' , 'a' , '2009 10 24' , '1' , '1' ) ,
('A' , 'b' , '2009 10 23' , '1' , '1' ) ,
('A' , 'b' , '2009 10 24' , '1' , '1' ) ,
('B' , 'a' , '2009 10 23' , '10' , '100' ) ,
('B' , 'a' , '2009 10 24' , '10' , '100' ) ,
('B' , 'b' , '2009 10 23' , '10' , '100' ) ,
('B' , 'b' , '2009 10 24' , '10' , '100' )
;
```



```
WITH vendas_região AS
(SELECT região , SUM (valor) AS totais
FROM pedidos
GROUP BY região),
melhores_regiões AS
(SELECT região
FROM vendas_região
WHERE totais > (SELECT SUM(totais)/10 FROM vendas
SELECT região , produto ,
SUM(quantidade) AS unidades ,
SUM(valor) AS valor
FROM pedidos
WHERE região IN
(SELECT região FROM melhores_regiões)
GROUP BY região , produto;
```



região	produto	unidades	valor
B	a	20	€200,00
B	b	20	€200,00

Como seria essa consulta sem a expressão de tabelas com a palavra chave WITH? Dois níveis de aninhamento de SELECT.



Mecanismo de recursão

▶ Partes:



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial
 - ▶ Passo de recursão (consulta)



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial
 - ▶ Passo de recursão (consulta)
 - ▶ Condição terminal



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial
 - ▶ Passo de recursão (consulta)
 - ▶ Condição terminal
- ▶ Se não...



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial
 - ▶ Passo de recursão (consulta)
 - ▶ Condição terminal
- ▶ Se não...
 - ▶ GNU Hurd



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial
 - ▶ Passo de recursão (consulta)
 - ▶ Condição terminal
- ▶ Se não...
 - ▶ GNU Hurd
 - ▶ Gnu Não é Unix



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial
 - ▶ Passo de recursão (consulta)
 - ▶ Condição terminal
- ▶ Se não...
 - ▶ GNU Hurd
 - ▶ Gnu Não é Unix
 - ▶ Hird of Unix Replacing Daemons



Mecanismo de recursão

- ▶ Partes:
 - ▶ Condição inicial
 - ▶ Passo de recursão (consulta)
 - ▶ Condição terminal
- ▶ Se não...
 - ▶ GNU Hurd
 - ▶ Gnu Não é Unix
 - ▶ Hird of Unix Replacing Daemons
 - ▶ Hurd of Interfaces Representing Depth



Expressões recursivas de tabelas

```
WITH RECURSIVE
  apelido      temporário
[      (
atributos
)]
AS
( termo_não_recursivo      inicial
  UNION [ALL]
  termo_recursivo      iteração & término
)
consulta      resultado
;
```



Exemplo simplicíssimo, do manual

```
WITH RECURSIVE
t (n)      tabela recursiva
  AS
(VVALUES (1)      inicial
 UNION
 SELECT n + 1      iteração
  FROM t
 WHERE n < 100     término
)
SELECT SUM (n)     resultado
  FROM t
;
505
```



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Incluir tuplas no resultado da tabela recursiva (RECURSIVE: 1)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Incluir tuplas no resultado da tabela recursiva (RECURSIVE: 1)
4. Incluir tuplas numa tabela temporária (1)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Incluir tuplas no resultado da tabela recursiva (RECURSIVE: 1)
4. Incluir tuplas numa tabela temporária (1)
5. Enquanto a tabela temporária for não vazia:



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Incluir tuplas no resultado da tabela recursiva (RECURSIVE: 1)
4. Incluir tuplas numa tabela temporária (1)
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substituir a referência recursiva pela tabela temporária (1)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Inclui tuplas no resultado da tabela recursiva (RECURSIVE: 1)
4. Inclui tuplas numa tabela temporária (1)
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária (1)
 - 5.2 Avalia o termo recursivo ($n + 1 = 2$)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Inclui tuplas no resultado da tabela recursiva (RECURSIVE: 1)
4. Inclui tuplas numa tabela temporária (1)
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária (1)
 - 5.2 Avalia o termo recursivo ($n + 1 = 2$)
 - 5.3 Elimina duplicados, inclui numa tabela intermediária (2)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Inclui tuplas no resultado da tabela recursiva (RECURSIVE: 1)
4. Inclui tuplas numa tabela temporária (1)
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária (1)
 - 5.2 Avalia o termo recursivo ($n + 1 = 2$)
 - 5.3 Elimina duplicados, inclui numa tabela intermediária (2)
 - 5.4 Substitui a tabela temporária pela intermediária (2)



Avaliação de consulta recursiva

1. Avaliar o termo não recursivo (inicial: 1)
2. Eliminar tuplas duplicadas (exceto ALL)
3. Inclui tuplas no resultado da tabela recursiva (RECURSIVE: 1)
4. Inclui tuplas numa tabela temporária (1)
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária (1)
 - 5.2 Avalia o termo recursivo ($n + 1 = 2$)
 - 5.3 Elimina duplicados, inclui numa tabela intermediária (2)
 - 5.4 Substitui a tabela temporária pela intermediária (2)
 - 5.5 Esvazia a intermediária, inclui a temporária na recursiva



Exemplo mais interessante, do manual

```
CREATE TABLE
partes
(
  subparte CHAR (1),
  parte CHAR (1),
  quantidade NUMERIC,
CONSTRAINT cp_partes PRIMARY KEY
  ( parte ,
    subparte)
)
```



Exemplo

Valores:

```
INSERT INTO
```

```
partes
```

```
(subparte , parte , quantidade)
```

```
VALUES
```

```
('a' , 'A' , '1' ),
```

```
('b' , 'A' , '1' ),
```

```
('a' , 'B' , '10' ),
```

```
('b' , 'B' , '10' ),
```

```
('x' , 'b' , '2' ),
```

```
('x' , 'a' , '2' )
```

```
;
```



Exemplo mais interessante, do manual

```
WITH RECURSIVE partes_inclusas
    (subparte, parte, quantidade) AS
(SELECT subparte, parte, quantidade
 FROM partes
 WHERE parte = 'B'
 UNION
 SELECT p.subparte, p.parte, p.quantidade
 FROM partes_inclusas AS i,
 partes AS p
 WHERE p.parte = i.subparte)
SELECT subparte, SUM (quantidade) AS total
 FROM partes_inclusas
 GROUP BY subparte;
```



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: a, B, 10 e b, B, 10)



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: a, B, 10 e b, B, 10)
2. Elimina tuplas duplicadas



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: a, B, 10 e b, B, 10)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: a, B, 10 e b, B, 10)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva
4. Inclui tuplas numa tabela temporária



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: a, B, 10 e b, B, 10)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva
4. Inclui tuplas numa tabela temporária
5. Enquanto a tabela temporária for não vazia:



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: a, B, 10 e b, B, 10)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva
4. Inclui tuplas numa tabela temporária
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária (b; x)



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: a, B, 10 e b, B, 10)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva
4. Inclui tuplas numa tabela temporária
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária (b; x)
 - 5.2 Avalia o termo recursivo (parte b, subparte da B; x, da b)



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: $a, B, 10$ e $b, B, 10$)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva
4. Inclui tuplas numa tabela temporária
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária ($b; x$)
 - 5.2 Avalia o termo recursivo (parte b , subparte da $B; x$, da b)
 - 5.3 Elimina duplicados, inclui numa tabela intermediária (b, x)



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: $a, B, 10$ e $b, B, 10$)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva
4. Inclui tuplas numa tabela temporária
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária ($b; x$)
 - 5.2 Avalia o termo recursivo (parte b , subparte da $B; x$, da b)
 - 5.3 Elimina duplicados, inclui numa tabela intermediária (b, x)
 - 5.4 Substitui a tabela temporária pela intermediária (b, x)



Avaliação de consulta recursiva

1. Avalia o termo não recursivo (esquerda: $a, B, 10$ e $b, B, 10$)
2. Elimina tuplas duplicadas
3. Inclui tuplas no resultado da tabela recursiva
4. Inclui tuplas numa tabela temporária
5. Enquanto a tabela temporária for não vazia:
 - 5.1 Substitui a referência recursiva pela tabela temporária (b, x)
 - 5.2 Avalia o termo recursivo (parte b , subparte da B ; x , da b)
 - 5.3 Elimina duplicados, inclui numa tabela intermediária (b, x)
 - 5.4 Substitui a tabela temporária pela intermediária (b, x)
 - 5.5 Esvazia a intermediária, inclui a temporária na recursiva (b, bx)

Outras aplicações

- ▶ Conjunto de Mandelbrot

Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'

Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias

Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias
 - ▶ Estrutura organizacional

Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias
 - ▶ Estrutura organizacional
 - ▶ ...



Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias
 - ▶ Estrutura organizacional
 - ▶ ...
- ▶ Árvores quaisquer



Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias
 - ▶ Estrutura organizacional
 - ▶ ...
- ▶ Árvores quaisquer
 - ▶ Árvore de produto (BoM, bill of materials)



Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias
 - ▶ Estrutura organizacional
 - ▶ ...
- ▶ Árvores quaisquer
 - ▶ Árvore de produto (BoM, bill of materials)
 - ▶ Genealógica



Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias
 - ▶ Estrutura organizacional
 - ▶ ...
- ▶ Árvores quaisquer
 - ▶ Árvore de produto (BoM, bill of materials)
 - ▶ Genealógica
 - ▶ ...



Outras aplicações

- ▶ Conjunto de Mandelbrot
- ▶ Problema 'do caixeiro viajante'
- ▶ Quaisquer hierarquias
 - ▶ Estrutura organizacional
 - ▶ ...
- ▶ Árvores quaisquer
 - ▶ Árvore de produto (BoM, bill of materials)
 - ▶ Genealógica
 - ▶ ...
- ▶ Grafos...



Conclusão

► Dúvidas?

Leandro Guimarães Faria Corcete DUTRA
ArsData
leandro.gfc.dutra@gmail.com



Conclusão

- ▶ Dúvidas?
- ▶ Perguntas?

Leandro Guimarães Faria Corcete DUTRA
ArsData
leandro.gfc.dutra@gmail.com



Conclusão

- ▶ Dúvidas?
- ▶ Perguntas?
- ▶ Críticas?

Leandro Guimarães Faria Corcete DUTRA
ArsData
leandro.gfc.dutra@gmail.com



Conclusão

- ▶ Dúvidas?
- ▶ Perguntas?
- ▶ Críticas?
- ▶ Sugestões?

Leandro Guimarães Faria Corcete DUTRA
ArsData
leandro.gfc.dutra@gmail.com



Conclusão

- ▶ Dúvidas?
- ▶ Perguntas?
- ▶ Críticas?
- ▶ Sugestões?
- ▶ Pepinos, controvérsias, protestos, atentados &c?

Leandro Guimarães Faria Corcete DUTRA

ArsData

leandro.gfc.dutra@gmail.com