[MAC0313]
Introdução aos Sistemas de Bancos de Dados
Aula 21
Linguagem SQL (Parte 8): Consultas Recursivas

Kelly Rosa Braghetto

DCC–IME–USP

26 de outubro de 2017

Consultas "não-convencionais"

- Existem alguns tipos de consultas que não conseguimos expressar usando a Álgebra Relacional
- Um caso comum são as consultas que envolvem **recursão**

Exemplo – Sequências de filmes

Vamos considerar a relação FILME_SEQUENCIA, que contém pares formados por um nome de filme e outro nome de filme que é a sua sequência imediata.

Filme	**Sequencia**
Rocky	Rocky II
Rocky II	Rocky III
Rocky III	Rocky IV

Podemos estar interessados em saber, para um dado filme, quais foram os seus "sucessores" (mesmo os não imediatos).

Consultas "não-convencionais"

Exemplo – Sequências de filmes

Podemos estar interessados em saber, para um dado filme, quais foram os seus "sucessores" (mesmo os não imediatos).

- É possível recuperar essa informação a partir da relação FILME_SEQUENCIA. **Mas como?**
- Também seria possível manter essa informação em uma relação, chamada, por exemplo, CONTINUACAO. Em CONTINUACAO, poderíamos ter tuplas que são pares (x,y), onde y é um filme que é uma continuação ("sucessor") de x. **Problema:** desperdício de espaço!

Consultas "não-convencionais"

Como obter CONTINUACAO a partir de FILME_SEQUENCIA

- Por meio de uma junção natural (operador * da álgebra relacional):

  $\pi_{\text{primeiro,terceiro}}($
  $\rho_{\text{FS1(primeiro,segundo)}}(\text{FILME\_SEQUENCIA}) *$
  $\rho_{\text{FS2(segundo,terceiro)}}(\text{FILME\_SEQUENCIA})$
  $)$

- O correspondente em SQL fica:

```
SELECT FS1.filme, FS2.sequencia
FROM FILME_SEQUENCIA AS FS1, FILME_SEQUENCIA AS FS2
WHERE FS1.sequencia = FS2.filme
```

- Problema: só obtém a sequência imediata de uma sequência imediata!

Consultas "não-convencionais"

Como obter CONTINUACAO a partir de FILME_SEQUENCIA

- Podemos obter i-ésima sequência de um filme fazendo uma junção de FILME_SEQUENCIA com ela mesma $i - 1$ vezes
- Fazendo a união das tuplas resultantes dessas junções, conseguimos todas as continuações de filmes até um **limite fixo**
- Entretanto, dessa forma não conseguimos definir uma "união infinita", que considere infinitas junções para gerar todas as possíveis continuações expressas na FILME_SEQUENCIA

O operador do ponto fixo

- Existe uma forma de expressar relações como a CONTINUACAO(x,y), que são construídas a partir de outras relações por meio de um processo infinito, porém regular
- Nós construímos uma equação na qual CONTINUACAO é descrita em termos de si mesma e de FILME_SEQUENCIA e então dizemos que o valor de CONTINUACAO é a menor relação (= ponto fixo mínimo) que satisfaz a equação
- Podemos usar o símbolo ϕ para indicar que o ponto fixo mínimo de uma equação deve ser considerado

O operador do ponto fixo

Exemplo do operador de ponto fixo aplicado a uma equação que descreve CONTINUACAO(x,y)

$$
\begin{aligned}
&\phi(\text{CONTINUACAO} = \\
&\quad \rho_{\text{FILME\_SEQUENCIA(x,y)}}(\text{FILME\_SEQUENCIA}) \cup \\
&\quad \rho_{\text{R(x,y)}}(\pi_{\text{filme,y}}(\\
&\qquad \text{FILME\_SEQUENCIA} \bowtie_{\text{sequencia=x}} \text{CONTINUACAO} \\
&\quad)) \\
&)
\end{aligned}
$$

Interpretação: "Um filme y é uma continuação de um filme x se ele é uma sequência imediata de x ou se ele é uma continuação de uma sequência imediata de x."

Calculando o ponto fixo

Para entender o significado do operador de ponto fixo, é preciso entender como ele pode ser calculado.

Método

1. Comece supondo que a relação R do lado esquerdo da equação está vazia
2. Repetidamente, calcule um novo valor para a relação R por meio da avaliação do lado direito da equação usando o valor antigo de R
3. Parar quando, depois de uma iteração, o valor novo e o antigo de R é o mesmo

Calculando o ponto fixo

Exemplo: Considere a relação FILME_SEQUENCIA com três tuplas:

Filme	**Sequencia**
Rocky	Rocky II
Rocky II	Rocky III
Rocky III	Rocky IV

Calculando o ponto fixo

Exemplo (cont.): Primeira rodada

- Na primeira rodada, assume-se que CONTINUACAO está vazia
- Assim, a junção na equação de ponto fixo devolve um conjunto vazio
- As únicas tuplas na nova CONTINUACAO virão do primeiro termo da união na equação, que é a relação FILME_SEQUENCIA
- Portanto, no final dessa rodada, CONTINUACAO é igual a FILME_SEQUENCIA

x	y
Rocky	Rocky II
Rocky II	Rocky III
Rocky III	Rocky IV

Calculando o ponto fixo

Exemplo (cont.): Segunda rodada

- Na segunda rodada, o primeiro termo da união dá as 3 tuplas de FILME_SEQUENCIA, e elas são incluídas na nova CONTINUACAO
- Para computar o segundo termo da união, usa-se como valor antigo para CONTINUACAO as tuplas do slide anterior
- A junção natural da antiga CONTINUACAO com FILME_SEQUENCIA resultará em 2 novas tuplas. A nova CONTINUACAO ficará como abaixo

x	y
Rocky	Rocky II
Rocky II	Rocky III
Rocky III	Rocky IV
Rocky	Rocky III
Rocky II	Rocky IV

Calculando o ponto fixo

Exemplo (cont.): Terceira rodada

- Na terceira rodada, o primeiro termo da união dá as 3 tuplas (...)
- Para computar o segundo termo da união, usa-se como (...)
- A junção natural da antiga CONTINUACAO com FILME_SEQUENCIA resultará em apenas 1 nova tupla:

x	y
Rocky	Rocky II
Rocky II	Rocky III
Rocky III	Rocky IV
Rocky	Rocky III
Rocky II	Rocky IV
Rocky	Rocky IV

Calculando o ponto fixo

Exemplo (cont.): Quarta rodada

- Na quarta rodada, não se obtém nenhuma tupla que já não estivesse na CONTINUACAO antiga
- Assim, a verdadeira relação CONTINUACAO definida por meio do cálculo de seu ponto fixo mínimo é a mostrada abaixo:

x	y
Rocky	Rocky II
Rocky II	Rocky III
Rocky III	Rocky IV
Rocky	Rocky III
Rocky II	Rocky IV
Rocky	Rocky IV

Consultas recursivas

- A abordagem usada pelo padrão SQL3 para consultas recursivas se baseia nas regras de *Datalog*
- A *Datalog* é uma linguagem lógica de consulta baseada na *Prolog*; ela é usada em **bancos de dados dedutivos**
- Em bancos de dados dedutivos, as tuplas nas relações sãos vistas como **fatos**
- Usando a *Datalog*, é possível especificar **regras**
- Uma máquina de inferência (= mecanismo de dedução) pode deduzir fatos novos do BD a partir dessas regras

Equações de ponto fixo em Datalog

- As expressões em álgebra relacional necessárias para definir equações de ponto fixo úteis costumam ser muito complicadas
- Frequentemente, é mais fácil representá-las como uma coleção de regras Datalog
- Mas é importante ressaltar que, na SQL, a notação para a definição de equações de ponto fixo é mais algébrica que lógica (porque isso está mais próximo dos demais comandos da SQL)

Exemplo: definição em Datalog para a relação CONTINUACAO

CONTINUACAO(X,Y) ← FILME_SEQUENCIA(X,Y)
CONTINUACAO(X,Y) ← FILME_SEQUENCIA(X,Z) AND
CONTINUACAO(Z,Y)

Notação Prolog/Datalog

Predicado

- **Predicado** – tem um nome exclusivo e um significado implícito (sugerido pelo seu nome), e um número fixo de argumentos
- Quando todos os argumentos de um predicado têm **valores constantes**, ele expressa que um certo **fato é verdadeiro**
- Quando o predicado contém **variáveis** como argumentos, então ele expressa uma **consulta** ou é parte de um regra
- Convenção (também usada na Prolog):
  - valores constantes em um predicado podem ser números ou cadeias de caracteres
  - as constantes que são cadeias de caracteres são iniciadas por letra minúscula
  - nomes de variáveis são iniciados por letra maiúscula

Notação Prolog/Datalog

Regra

- Uma **regra** tem a forma **cabeça :- corpo**, onde **:-** lê-se como "se e somente se"
- A **cabeça** é definida geralmente por **um único predicado**
- O **corpo** é definido **por um ou mais predicados**, também chamados de *premissas*
- Nos argumentos dos predicados de uma regra, geralmente aparecem tanto **variáveis** quanto **constantes**
- Uma regra especifica que, se uma atribuição de valores constantes às variáveis do corpo tornar todos os predicados do corpo da regra verdadeiros, então a cabeça da regra também será verdadeira usando a mesma atribuição de valores constantes às variáveis.
- Assim, uma regra permite gerar novos fatos, que são instanciações da cabeça da regra.

Notação Prolog/Datalog

Exemplo

- Fatos:

  FILME_SEQUENCIA(rocky,rocky ii)
  FILME_SEQUENCIA(rocky ii,rocky iii)
  FILME_SEQUENCIA(rocky iii,rocky iv)

- Consultas:

  FILME_SEQUENCIA(rocky,X)
  quais são as sequências do filme "rocky"?

  FILME_SEQUENCIA(X,rocky iii)
  quais são os filmes que têm como sequência o filme "rocky iii"?

Notação Prolog/Datalog

Exemplo – regras

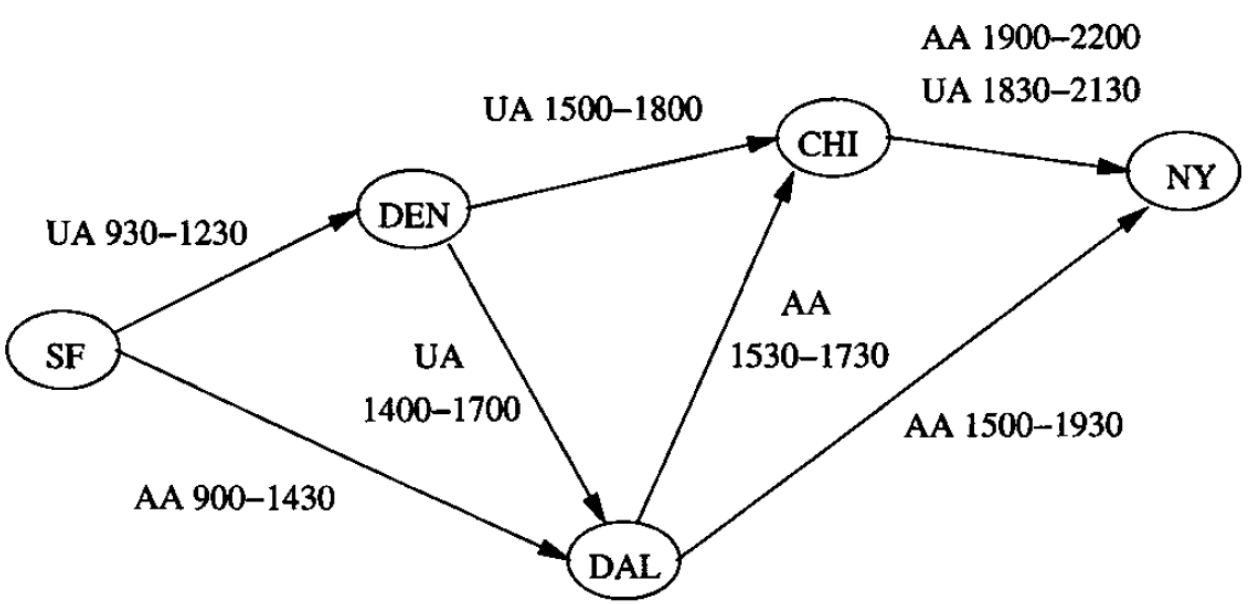
CONTINUACAO(X,Y) ← FILME_SEQUENCIA(X,Y)
CONTINUACAO(X,Y) ← FILME_SEQUENCIA(X,Z) AND
CONTINUACAO(Z,Y)

- As duas regras `CONTINUACAO(X,Y)` representam o fato de que "X é um continuação de Y", por meio de uma sequência direta ou indireta.
- A segunda regra com cabeça `CONTINUACAO(X,Y)` é uma **regra recursiva**, pois em seu corpo há um predicado que é igual ao predicado da cabeça da regra.
- Quando há duas (ou mais) regras com uma mesma cabeça, isso equivale a dizer que o predicado da cabeça é verdadeiro se qualquer um dos dois corpos for verdadeiro.
  Isso equivale à operação de **OU lógico**.

Exemplos mais complexos envolvendo recursão

Estudo de caminhos em grafos

Grafo com os voos de duas companhias aéreas – a UA e a AA



Esquema da relação VOO:

`Voos(Companhia, De, Para, HPartida, HChegada)`

Exemplos mais complexos envolvendo recursão

Estudo de caminhos em grafos

- Questão recursiva mais simples: "Para quais pares de cidade (X,Y) é possível ir da cidade X para a cidade Y pegando 1 ou mais voos?"

  As regras a seguir definem a relação ALCANCA, que contém esses pares de cidade:

  ALCANCA(X,Y) $\leftarrow$ VOO(A,X,Y,P,C)
  ALCANCA(X,Y) $\leftarrow$ ALCANCA(X,Z) AND ALCANCA(Z,Y)

Exemplos mais complexos envolvendo recursão

Estudo de caminhos em grafos

- Questão recursiva mais complexa: "Para quais pares de cidade (X,Y) é possível ir da cidade X para a cidade Y pegando 1 ou mais voos, mas com a restrição de que o tempo de espera em uma conexão seja de pelo menos uma hora?"

  As regras a seguir definem a relação CONECTA(X,Y,P,C). Elas dizem que podemos pegar um ou mais voos para partir de X no horário P e chegar em Y no horário C, sendo que, para cada conexão, haverá um tempo mínimo de 1 hora:

  CONECTA(X,Y,P,C) ← VOO(A,X,Y,P,C)
  CONECTA(X,Y,P,C) ← CONECTA(X,Z,P,T1) AND
  CONECTA(Z,Y,T2,C) AND
  T1 <= T2 - '01:00'

Negação em regras recursivas

- Às vezes, é necessário envolver negação em regras que envolvem recursão
- Há duas formas de se fazer: de forma segura e de forma não segura
- De forma geral, só é considerado apropriado usar a negação em situações onde ela não aparece dentro da operação de ponto fixo

Negação em regras recursivas

Exemplo de negação + recursão segura

- Queremos encontrar os pares de cidade (x,y) no grafo de voos tais que a companhia UA tem voos conectando x a y (talvez passando por mais de uma cidade), mas que a companhia AA não tem.

  A resposta para essa consulta é a relação SOMENTE_UA, definida a seguir:

$$\begin{aligned}
&\text{UA_ALCANCA}(X,Y) \leftarrow \text{VOO}(ua,X,Y,P,C)\\
&\text{UA_ALCANCA}(X,Y) \leftarrow \text{UA_ALCANCA}(X,Z)\ \text{AND}\\
&\qquad\qquad\qquad\qquad\ \text{UA_ALCANCA}(Z,Y)\\
&\text{AA_ALCANCA}(X,Y) \leftarrow \text{VOO}(aa,X,Y,P,C)\\
&\text{AA_ALCANCA}(X,Y) \leftarrow \text{AA_ALCANCA}(X,Z)\ \text{AND}\\
&\qquad\qquad\qquad\qquad\ \text{AA_ALCANCA}(Z,Y)\\
&\text{SOMENTE_UA}(X,Y) \leftarrow \text{UA_ALCANCA}(X,Y)\ \text{AND}\\
&\qquad\qquad\qquad\qquad\ \text{NOT AA_ALCANCA}(X,Y)
\end{aligned}$$

Negação em regras recursivas

Exemplo de negação + recursão segura

- No exemplo do slide anterior, computa-se os pontos fixos de UA_ALCANCA e de AA_ALCANCA independentemente (usando o método descrito anteriormente)
- Depois disso, calcula-se a relação SOMENTE_UA
- A negação envolvida na definição da relação não atrapalha o seu cálculo, já que ela não interfere no cálculo dos pontos fixos

Negação em regras recursivas

Exemplo de negação + recursão **não** segura

- Considere uma relação unária R (= com 1 só atributo) que contém uma só tupla (0)
- Considere duas relações unárias P e Q, definidas sobre R da seguinte maneira:

  P(X) $\leftarrow$ R(X) AND NOT Q(X)
  Q(X) $\leftarrow$ R(X) AND NOT P(X)
- Essas regras dizem que um elemento X em R ou está em P, ou está em Q. Mas X não pode estar em P e Q ao mesmo tempo. Isso equivale às equações:
  P = R - Q
  Q = R - P

Negação em regras recursivas

Exemplo de negação + recursão **não** segura (cont.)

- Como R contém somente a tupla (0), sabemos que essa tupla tem que estar ou em P, ou em Q. Mas onde está o (0)?
- A tupla não pode não estar em nenhuma das duas relações (P e Q), porque nesse caso as equações não seriam satisfeitas:
  P = R - Q implicaria que $\emptyset = \{(0)\} - \emptyset$ (falso!)
- Se fizermos P = $\{(0)\}$ enquanto Q = $\emptyset$, então temos uma solução para as duas equações:
  P = R - Q implicaria que $\{(0)\} = \{(0)\} - \emptyset$ (verdade!)
  Q = R - P implicaria que $\emptyset = \{(0)\} - \{(0)\}$, (também verdade!)
- Mas se fizermos Q = $\{(0)\}$ enquanto P = $\emptyset$, também temos uma solução para as duas equações

Negação em regras recursivas

Exemplo de negação + recursão **não** segura (cont.)

- Assim, temos duas soluções:
  a) $P = \{(0)\}$ $\quad Q = \emptyset$
  a) $P = \emptyset$ $\quad Q = \{(0)\}$
- Ambas são minimais – se removemos qualquer tupla de uma relação, as relações resultantes não satisfazem mais as regras
- Portanto, não é possível decidir entre os pontos fixos mínimos (a) e (b)
- Isso significa que somos incapazes de responder à seguinte questão simples: "P(0) é verdade?"

Recursão estratificada

- **Conclusão:** a ideia definir o significado de regras recursivas por meio do método do ponto fixo mínimo não funciona quando temos negação e recursão "intrincadas"
- Como não há um método alternativo que seja um consenso, convenciona-se que só devemos usar negações em recursões quando essas negações forem **estratificadas**.
- Quando uma negação é estratificada, existe um algoritmo capaz de computar um ponto fixo mínimo que está de acordo com a nossa intuição sobre o quê as regras significam

Recursão estratificada

Método de detecção – construção do grafo de dependência das relações

1. Desenhe um grafo nos quais os nós correspondem às relações definidas por regras
2. Desenhe um arco do nó A para o nó B se uma regra com A na cabeça possui um B negado no seu corpo. Rotule esse arco com um sinal de '-', para indicar que ele é um arco negativo
3. Desenhe um arco do nó A para o nó B se uma regra com cabeça A possui um B não negado em seu corpo. Esse arco não deve ter o sinal '-' como rótulo

Recursão estratificada

Método de detecção (cont.)

- Se o grafo de dependências possui um ciclo que contém um ou mais arcos negativos, então a recursão é **não estratificada**
- No caso contrário, a recursão é estratificada
- É possível agrupar as relações no grafo em **estratos**
- O estrato de uma relação A é o maior número de arcos negativos em um caminho começando por A
- Se a recursão é estratificada, podemos calcular as relações na ordem de seus estratos, começando pelo menor estrato
- Esta estratégia produz um dos pontos fixos mínimo das regras

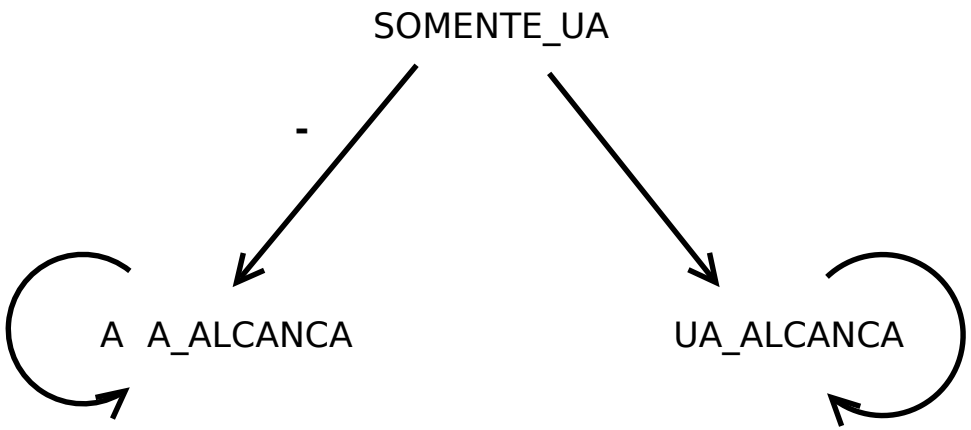
Recursão estratificada

Exemplo – grafo de dependência da regra segura

UA_ALCANCA(X,Y) ← VOO(ua,X,Y,P,C)
UA_ALCANCA(X,Y) ← UA_ALCANCA(X,Z) AND
UA_ALCANCA(Z,Y)

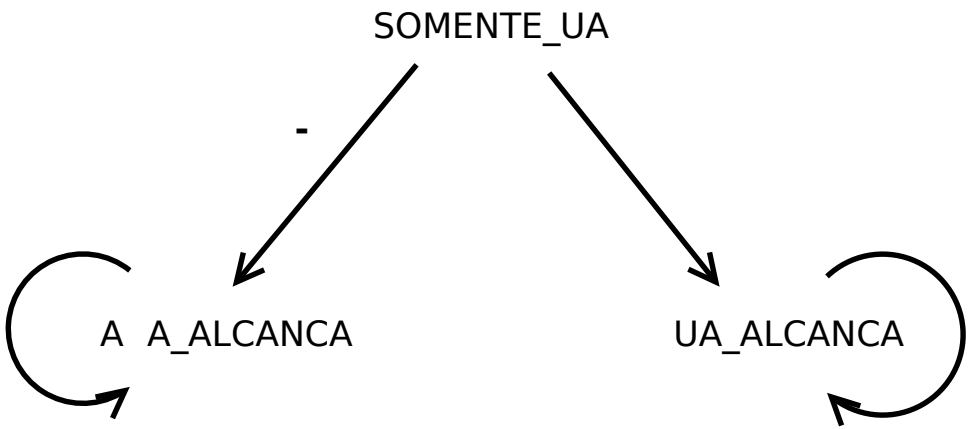
AA_ALCANCA(X,Y) ← VOO(aa,X,Y,P,C)
AA_ALCANCA(X,Y) ← AA_ALCANCA(X,Z) AND
AA_ALCANCA(Z,Y)

SOMENTE_UA(X,Y) ← UA_ALCANCA(X,Y) AND
NOT AA_ALCANCA(X,Y)



Recursão estratificada

Exemplo – grafo de dependência da regra segura

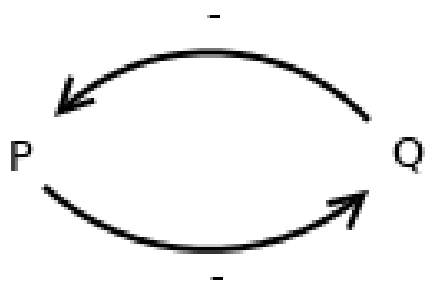


- SOMENTE_UA está no estrato 1, porque possui caminhos com 1 arco negativo partindo dela
- UA_ALCANCA e AA_ALCANCA estão no estrato 0 (e portanto devem ser avaliadas antes do início da avaliação de SOMENTE_UA)

Recursão estratificada

Exemplo – grafo de dependência da regra não segura

P = R - Q
Q = R - P



- Há um ciclo com arcos negativos no grafo
- Portanto, as regras não estão estratificadas

Consultas SQL recursivas

Definição de consultas recursivas

- Forma geral do comando SQL:

  WITH R **AS** <definição de R> <consulta envolvendo R>

- "Ideia" do comando: definir uma relação temporária chamada R e então usar R em alguma outra consulta (cuja resposta será a resposta final para o comando todo)
- Na verdade, é possível definir várias relações depois da cláusula **WITH**; todas podem ser recursivas (e até mutualmente recursivas)
- Toda relação envolvida em uma recursão precisa ser precedida pela palavra-chave **RECURSIVE**

Consultas SQL recursivas

Exemplo simples

```
WITH
  RECURSIVE relacao_temp(n) AS (
    SELECT 1
    UNION
    SELECT n+1 FROM relacao_temp WHERE n < 5
  )
SELECT sum(n) FROM relacao_temp;
```

$\Rightarrow$ O resultado dessa consulta é o número 15.
A relação `relacao_temp` termina com as tuplas
{(1),(2),(3),(4),(5)}.

Consultas SQL recursivas

Exemplo: relação ALCANCA

```
WITH
  RECURSIVE Alcanca(de,para) AS (
    (SELECT de, para FROM Voos)
    UNION
    (SELECT R1.de, R2.para
     FROM Alcanca as R1, Alcanca as R2
     WHERE R1.para = R2.de)
  )
SELECT * FROM Alcanca;
```

A sintaxe acima é a apresentada nos livros de BD.

Recursões lineares

- O padrão SQL3 só aceita consultas envolvendo **recursões lineares**
- A consulta recursiva mostrada no slide anterior é inválida!
- Uma recursão é linear se na cláusula **FROM** de todas as relações definidas há **no máximo uma ocorrência** de uma relação que é mutualmente recursiva com a relação sendo definida.
- No caso mais comum, a relação sendo definida aparece ela própria uma vez na cláusula **FROM**

Consultas SQL recursivas

Exemplo: relação `Alcanca`

Em SQL, versão recursiva linear:

```
WITH
  RECURSIVE Alcanca(de,para) AS (
    (SELECT de, para FROM Voos)
    UNION
    (SELECT Voos.de, Alcanca.para
     FROM Voos, Alcanca
     WHERE Voos.para = Alcanca.de)
  )
SELECT * FROM Alcanca;
```

Consultas SQL recursivas

Exemplo: relação `Alcanca`

Em SQL, versão recursiva linear "alternativa":

```
WITH
  Pares AS (SELECT de, para FROM Voos),

  RECURSIVE Alcanca(de,para) AS (
    Pares
    UNION
    (SELECT Pares.de, Alcanca.para
     FROM Pares, Alcanca
     WHERE Pares.para = Alcanca.de)
  )
SELECT * FROM Alcanca;
```

Consultas SQL recursivas

Exemplo: relação `Alcanca`

Na sintaxe aceita no PostgreSQL:

```
WITH RECURSIVE
  Pares AS (SELECT de, para FROM Voos),

  Alcanca(de,para) AS (
    (SELECT * FROM Pares)
    UNION
    (SELECT Pares.de, Alcanca.para
     FROM Pares, Alcanca
     WHERE Pares.para = Alcanca.de)
  )
SELECT * FROM Alcanca;
```

- No PostgreSQL, a cláusula `RECURSIVE` é um modificador do `WITH`
- O PostgreSQL não entende `R UNION S`, mas entende `(SELECT * FROM R) UNION (SELECT * FROM S)`; onde `R` e `S` são duas relações quaisquer

Expressões problemáticas em consultas SQL recursivas

Exemplo de consulta inválida no SQL3 (recursão insegura!)

```
WITH
    RECURSIVE P(x) AS (
        (SELECT * FROM R)
        EXCEPT
        (SELECT * FROM Q)
    ),

    RECURSIVE Q(x) AS (
        (SELECT * FROM R)
        EXCEPT
        (SELECT * FROM P)
    )
SELECT * FROM P;
```

Referências Bibliográficas

- *A First Course in Database Systems* (1ª edição), Ullman e Widom. Prentice Hall, 1997.
  Capítulos 4 e 5
- *Sistemas de Bancos de Dados* (6ª edição), Elmasri e Navathe. Pearson, 2010.
  Capítulo 26