13) Entendendo a Herança de tabelas

Vamos criar duas tabelas. A tabela capitais contém as capitais dos estados, que também são cidades. Por conseqüência, a tabela capitais deve herdar da tabela cidades.

```
CREATE TABLE cidades (
    nome        text,
    populacao   float,
    altitude    int     -- (em pés)
);

CREATE TABLE capitais (
    estado      char(2)
) INHERITS (cidades);
```

Neste caso, as linhas da tabela capitais *herdam* todos os atributos (nome, população e altitude) de sua tabela ancestral, cidades. As capitais dos estados possuem um atributo adicional chamado estado, contendo seu estado. No PostgreSQL uma tabela pode herdar de zero ou mais tabelas, e uma consulta pode referenciar tanto todas as linhas de uma tabela, quanto todas as linhas de uma tabela mais todas as linhas de suas descendentes.

> **Nota:** A hierarquia de herança é, na verdade, um grafo acíclico dirigido. [1]

Por exemplo, a consulta abaixo retorna os nomes de todas as cidades, incluindo as capitais dos estados, localizadas a uma altitude superior a 500 pés:

```
SELECT nome, altitude
    FROM cidades
    WHERE altitude > 500;

   nome    | altitude
-----------+----------
 Las Vegas |     2174
 Mariposa  |     1953
 Madison   |      845
```

Por outro lado, a consulta abaixo retorna todas as cidades situadas a uma altitude superior a 500 pés, que não são capitais de estados:

```
SELECT nome, altitude
    FROM ONLY cidades
    WHERE altitude > 500;

   nome    | altitude
-----------+----------
 Las Vegas |     2174
 Mariposa  |     1953
```

O termo "ONLY" antes de cidades indica que a consulta deve ser executada apenas na tabela cidades, sem incluir as tabelas descendentes de cidades na hierarquia de herança. Muitos comandos mostrados até agora — SELECT, UPDATE e DELETE — suportam esta notação de "ONLY".

> **Em obsolescência:** Nas versões anteriores do PostgreSQL, o comportamento padrão era não incluir as tabelas descendentes nos comandos. Descobriu-se que isso ocasionava

muitos erros, e que também violava o padrão SQL:1999. Na sintaxe antiga, para incluir as tabelas descendentes era necessário anexar um * ao nome da tabela. Por exemplo:

```
SELECT * FROM cidades*;
```

Ainda é possível especificar explicitamente a varredura das tabelas descendentes anexando o *, assim como especificar explicitamente para não varrer as tabelas descendentes escrevendo "ONLY". A partir da versão 7.1 o comportamento padrão para nomes de tabelas sem adornos passou a ser varrer as tabelas descendentes também, enquanto antes desta versão o comportamento padrão era não varrer as tabelas descendentes. Para ativar o comportamento padrão antigo, deve ser definida a opção de configuração SQL_Inheritance como desativada como, por exemplo,

```
SET SQL_Inheritance TO OFF;
```

ou definir o parâmetro de configuração sql_inheritance.

Em alguns casos pode-se desejar saber de qual tabela uma determinada linha se origina. Em cada tabela existe uma coluna do sistema chamada tableoid que pode informar a tabela de origem:

```
SELECT c.tableoid, c.nome, c.altitude
FROM cidades c
WHERE c.altitude > 500;

 tableoid |   nome    | altitude
----------+-----------+----------
   139793 | Las Vegas |     2174
   139793 | Mariposa  |     1953
   139798 | Madison   |      845
```

Se for tentada a reprodução deste exemplo, os valores numéricos dos OIDs provavelmente serão diferentes. Fazendo uma junção com a tabela "pg_class" é possível mostrar o nome da tabela:

```
SELECT p.relname, c.nome, c.altitude
FROM   cidades c, pg_class p
WHERE  c.altitude > 500 AND c.tableoid = p.oid;

 relname  |   nome    | altitude
----------+-----------+----------
 cidades  | Las Vegas |     2174
 cidades  | Mariposa  |     1953
 capitais | Madison   |      845
```

Uma tabela pode herdar de mais de uma tabela ancestral e, neste caso, possuirá a união das colunas definidas nas tabelas ancestrais (além de todas as colunas declaradas especificamente para a tabela filha).

Uma limitação séria da funcionalidade da herança é que os índices (incluindo as restrições de unicidade) e as chaves estrangeiras somente se aplicam a uma única tabela, e não às suas descendentes. Isto é verdade tanto do lado que faz referência quanto do lado que é referenciado na chave estrangeira. Portanto, em termos do exemplo acima:

- Se for declarado cidades.nome como sendo UNIQUE ou PRIMARY KEY, isto não impedirá que a tabela capitais tenha linhas com nomes idênticos aos da tabela cidades e, por padrão,

estas linhas duplicadas aparecem nas consultas à tabela cidades. Na verdade, por padrão, a tabela capitais não teria nenhuma restrição de unicidade e, portanto, poderia conter várias linhas com nomes idênticos. Poderia ser adicionada uma restrição de unicidade à tabela capitais, mas isto não impediria um nome idêntico na tabela cidades.

- De forma análoga, se for especificado cidades.nome REFERENCES alguma outra tabela, esta restrição não se propagará automaticamente para a tabela capitais. Neste caso, o problema poderia ser contornado adicionando manualmente a restrição REFERENCES para a tabela capitais.
- Especificar para uma coluna de outra tabela REFERENCES cidades(nome) permite à outra tabela conter os nomes das cidades, mas não os nomes das capitais. Não existe um maneira boa para contornar este problema.

Estas deficiências deverão, provavelmente, serem corrigidas em alguma versão futura, mas enquanto isso deve haver um cuidado considerável ao decidir se a herança é útil para resolver o problema em questão.

Notas

[1] *Grafo*: uma coleção de vértices e arestas; *Grafo dirigido*: um grafo com arestas unidirecionais; *Grafo acíclico dirigido*: um grafo dirigido que não contém ciclos. FOLDOC - Free On-Line Dictionary of Computing (N. do T.)