9) Alterando dados nas tabelas

9.1) Adicionando dados com Insert

Inserção de dados

A tabela recém-criada não contém dados. A primeira ação a ser realizada para o banco de dados ter utilidade é inserir dados. Conceitualmente, os dados são inseridos uma linha de cada vez. É claro que é possível inserir mais de uma linha, mas não existe maneira de inserir menos de uma linha por vez. Mesmo que se conheça apenas o valor de algumas colunas, deve ser criada uma linha completa.

Para criar uma linha é utilizado o comando INSERT. Este comando requer o nome da tabela, e um valor para cada coluna da tabela. Por exemplo, considere a tabela produtos do Capítulo 5:

```
CREATE TABLE produtos (
    cod_prod   integer,
    nome       text,
    preco      numeric
);
```

Um exemplo de comando para inserir uma linha é:

```
INSERT INTO produtos VALUES (1, 'Queijo', 9.99);
```

Os valores dos dados são colocados na mesma ordem que as colunas se encontram na tabela, separados por vírgula. Geralmente os valores dos dados são literais (constantes), mas também são permitidas expressões escalares.

A sintaxe mostrada acima tem como desvantagem ser necessário conhecer a ordem das colunas da tabela. Para evitar isto, as colunas podem ser relacionadas explicitamente. Por exemplo, os dois comandos mostrados abaixo possuem o mesmo efeito do comando mostrado acima:

```
INSERT INTO produtos (cod_prod, nome, preco) VALUES (1, 'Queijo', 9.99);
INSERT INTO produtos (nome, preco, cod_prod) VALUES ('Queijo', 9.99, 1);
```

Muitos usuários consideram boa prática escrever sempre os nomes das colunas.

Se não forem conhecidos os valores de todas as colunas, as colunas com valor desconhecido podem ser omitidas. Neste caso, estas colunas são preenchidas com seu respectivo valor padrão. Por exemplo:

```
INSERT INTO produtos (cod_prod, nome) VALUES (1, 'Queijo');
```

```
INSERT INTO produtos VALUES (1, 'Queijo');
```

A segunda forma é uma extensão do PostgreSQL, que preenche as colunas a partir da esquerda com quantos valores forem fornecidos, e as demais com o valor padrão.

Para ficar mais claro, pode ser requisitado explicitamente o valor padrão da coluna individualmente, ou para toda a linha:

```
INSERT INTO produtos (cod_prod, nome, preco) VALUES (1, 'Queijo', DEFAULT);
INSERT INTO produtos DEFAULT VALUES;
```

> **Dica:** Para realizar "cargas volumosas", ou seja, inserir muitos dados, consulte o comando *COPY*. Este comando não é tão flexível quanto o comando INSERT, mas é mais eficiente.

Fonte: http://pgdocptbr.sourceforge.net/pg80/dml.html

9.2) Adicionando dados com Select

O exemplo abaixo insere algumas linhas na tabela filmes a partir da tabela temp_filmes com a mesma disposição de colunas da tabela filmes:

```
INSERT INTO filmes SELECT * FROM temp_filmes WHERE data_prod < '2004-05-07';
```

9.3) Modificando dados com Update

Atualização de dados

A modificação dos dados armazenados no banco de dados é referida como atualização. Pode ser atualizada uma linha, todas as linhas, ou um subconjunto das linhas da tabela. Uma coluna pode ser atualizada separadamente; as outras colunas não são afetadas.

Para realizar uma atualização são necessárias três informações:

1. O nome da tabela e da coluna a ser atualizada;
2. O novo valor para a coluna;
3. Quais linhas serão atualizadas.

Lembre-se que foi dito no Capítulo 5 que o SQL, de uma maneira geral, não fornece um identificador único para as linhas. Portanto, não é necessariamente possível especificar diretamente a linha a ser atualizada. Em vez disso, devem ser especificadas as condições que a linha deve atender para ser atualizada. Somente havendo uma chave primária na tabela (não importando se foi declarada ou não), é possível endereçar uma linha específica com confiança, escolhendo uma condição correspondendo à chave primária. Ferramentas gráficas de acesso a banco de dados dependem da chave primária para poderem atualizar as linhas individualmente.

Por exemplo, o comando mostrado abaixo atualiza todos os produtos com preço igual a 5, mudando estes preços para 10:

```
UPDATE produtos SET preco = 10 WHERE preco = 5;
```

Este comando pode atualizar nenhuma, uma, ou muitas linhas. Não é errado tentar uma atualização que não corresponda a nenhuma linha.

Vejamos este comando em detalhe: Primeiro aparece a palavra chave UPDATE seguida pelo nome da tabela. Como usual, o nome da tabela pode ser qualificado pelo esquema, senão é procurado no caminho. Depois aparece a palavra chave SET, seguida pelo nome da coluna, por um sinal de igual, e pelo novo valor da coluna. O novo valor da coluna pode ser qualquer expressão escalar, e não apenas uma constante. Por exemplo, se for desejado aumentar o preço de todos os produtos em 10% pode ser utilizado:

```
UPDATE produtos SET preco = preco * 1.10;
```

Como pode ser visto, a expressão para obter o novo valor pode fazer referência ao valor antigo. Também foi deixada de fora a cláusula WHERE. Quando esta cláusula é omitida, significa que todas as linhas da tabela serão atualizadas e, quando está presente, somente as linhas que atendem à condição desta cláusula serão atualizadas. Deve ser observado que o sinal de igual na cláusula SET é uma atribuição, enquanto o sinal de igual na cláusula WHERE é uma comparação, mas isto não cria uma ambigüidade. Obviamente, a condição da cláusula WHERE não é necessariamente um teste de igualdade, estão disponíveis vários outros operadores (consulte o Capítulo 9), mas a expressão deve produzir um resultado booleano.

Também pode ser atualizada mais de uma coluna pelo comando UPDATE, colocando mais de uma atribuição na cláusula SET. Por exemplo:

```
UPDATE minha_tabela SET a = 5, b = 3, c = 1 WHERE a > 0;
```

9.4) Removendo dados com Delete

Exclusão de dados

Até aqui foi mostrado como adicionar dados a tabelas, e como modificar estes dados. Está faltando mostrar como remover os dados que não são mais necessários. Assim como só é possível adicionar dados para toda uma linha, uma linha também só pode ser removida por inteiro da tabela. Na seção anterior foi explicado que o SQL não fornece uma maneira para endereçar diretamente uma determinada linha. Portanto, a remoção das linhas só pode ser feita especificando as condições que as linhas a serem removidas devem atender. Havendo uma chave primária na tabela, então é possível especificar exatamente a linha. Mas também pode ser removido um grupo de linhas atendendo a uma determinada condição, ou podem ser removidas todas as linhas da tabela de uma só vez.

É utilizado o comando DELETE para remover linhas; a sintaxe deste comando é muito semelhante a do comando UPDATE. Por exemplo, para remover todas as linhas da tabela produtos possuindo preço igual a 10:

```
DELETE FROM produtos WHERE preco = 10;
```

Se for escrito simplesmente

```
DELETE FROM produtos;
```

então todas as linhas da tabela serão excluídas! Dica de programador.

9.5) Removendo dados com Truncate

TRUNCATE -- esvazia a tabela

Sinopse

```
TRUNCATE [ TABLE ] nome
```

Descrição

O comando TRUNCATE remove rapidamente todas as linhas da tabela. Possui o mesmo efeito do comando DELETE não qualificado (sem WHERE), mas como na verdade não varre a tabela é mais rápido. É mais útil em tabelas grandes.

Parâmetros

nome

O nome (opcionalmente qualificado pelo esquema) da tabela a ser truncada.

Observações

O comando TRUNCATE não pode ser usado quando existe referência de chave estrangeira de outra tabela para a tabela. Neste caso a verificação da validade tornaria necessária a varredura da tabela, e o ponto central é não fazê-la.

O comando TRUNCATE não executa nenhum gatilho ON DELETE definido pelo usuário, porventura existente na tabela.

Exemplos

Truncar a tabela tbl_grande:

```
TRUNCATE TABLE tbl_grande;
```

Compatibilidade

Não existe o comando TRUNCATE no padrão SQL.

Manipulação de registros

INSERT insere um registro por vez.

Para inserir vários registros de cada vez usar:

INSERT INTO tabela (campo1, campo2) SELECT campo1, campo2 FROM tabela2;

INSERT INTO tabela SELECT * FROM tabela2;

Operadores Lógicos

... WHERE (c1=3 AND c2=5) OR c3=6;

- Não há limite na quantidade de operadores usados;
- Os operadores são avaliados da esquerda para a direita;
- É muito importante usar parênteses: legibilidade e garantem o que queremos. Sem eles pode ficar obscuro.