Manipulando Dados

Curso de Introdução a Oracle 10g: SQL

Prof.: Marlon Mendes Minussi
marlonminussi@gmail.br

INSERT

- Sintaxe:

  INSERT INTO table [(column [, column…])]

  VALUES (value [, value…]);

- Você pode adicionar linhas novas para uma tabela executando um commando INSERT.
  - Table: é o nome da tabela
  - Column: é o nome da coluna da tabela que receberá os valores
  - Value: é o valor correspondente para a coluna
- Nota: este comando com a cláusula VALUES adiciona apenas uma linha de cada vez para a tabela.

Inserindo Novas Linhas

- Ex.:

  INSERT INTO paciente (cod_pac, nome, dt_nasc, rg, sexo, fone, est_civil,end) VALUES (1, 'PEDRO BIAL PEREIRA', '15-08-1964', 1234567890, 'M', '(21)31111587','Solteiro','RUA BARATA RIBEIRO, 320');

- Uma vez que você pode inserir uma linha nova que contenha valores para cada coluna da tabela, a lista de colunas não é obrigatória na cláusula INSERT.
- Porém, se você não utilizar a lista de colunas, os valores devem ser listados de acordo com a ordem default das colunas na tabela.
- Por questões de clareza, utilize a lista de colunas na cláusula INSERT. Coloque valores caractere a data entre aspas simples; não inclua valores numéricos entre aspas.

Inserindo Linhas com Valores Nulos

- Método implícito:

```
INSERT INTO atendimento
(cod_atendimento, cod_pac, cod_med, dt_atendimento, diagnostico, valor)
VALUES (1, 5, 1, ‘01-04-2011’, 'Estado regular', 100);
```

- Método explícito:

```
INSERT INTO atendimento
VALUES (1, 5, 1 , ‘01-04-2011’, 'Estado regular', 100, NULL);
```

Inserindo Linhas com Valores Nulos

Método	Descrição
Implícito	Omite a coluna da lista de colunas
Explícito	Especifique a palavra NULL na lista de cláusula VALUES. Especifique uma string vazia (‘ ‘) na lista da cláusula VALUES para valores do tipo caractere e data

- Tenha certeza que a coluna de destino permita valores nulos verificando o status da coluna Null? De o comando DESCRIBE do SQL*Plus.
- O Servidor Oracle automaticamente verifica todos os tipos de dados, intervalos de valores e regras de integridade de dados. Qualquer coluna que não é listada explicitamente recebe um valor nulo na linha nova.

Inserindo Valores Especiais

- Ex.:

  INSERT INTO atendimento

  VALUES (1, 5, 1 , ‘01-04-2011’, 'Estado regular', 100, NULL);

- Você pode utilizar pseudo colunas para entrar valores especiais em sua tabela.
- O exemplo acima armazena informações para o atendimento 5 na tabela ATENDIMENTO. Ele armazena a data e hora atual na coluna DT_ATENDIMENTO, utilizando a função SYSDATE.
- Você também pode utilizar a função USER quando inserir linhas em uma tabela. A função USER armazena o nome do usuário atual.

Confirmando Inserções para uma Tabela

- Ex.:

```
SELECT *
FROM atendimento
WHERE cod_atendimento = 1;
```

Inserindo Valores de Data Específicos

- Adicione um novo paciente:

  INSERT INTO paciente VALUES (2, 'BILL GATES', TO_DATE('OUT 28,55', 'MON DD, YY'), 7784512397, 'M', '85478965','Casado','Vale do Silício, CA, USA');
- O formato DD-MON-YY é normalmente utilizado para inserir um valor de data. Com este formato, lembre-se que o século fica sendo o século atual. Uma vez que a data também possui informação de hora, a hora default é meia-noite (00:00:00).
- Se uma data precisa ser entrada em outro século ou com uma hora especifica, utilize a função TO_DATE.
- O exemplo acima armazena informações para o paciente 2 na tabela PACIENTE. O valor da coluna DT_NASC fica sendo 28 de outubro de 1955.
- Se o formato CC for utilizado, o século pode então não ser o atual.

  Para testar: select to_char(dt_nasc, 'DD/MM/YYYY/CC HH24:MI:SS') from paciente;

Copiando Linhas a partir de Outra Tabela

- Ex.:

```
INSERT INTO pessoas (id, nome, cidade, estado)
SELECT id, nome, cidade, estado
FROM tclientes
WHERE estado IN ('RS', 'RJ');
```

- Você pode utilizar o comando INSERT para adicionar linhas para uma tabela onde os valores são derivados de outras tabelas. Ao invés de utilizar a cláusula VALUES, utilize uma sub-consulta.

Copiando Linhas a partir de Outra Tabela

- Sinaxe:

  INSERT INTO table [column (, column)] subquery;

- onde:

  table: é o nome da tabela

  column: é o nome da coluna da tabela que receberá valores

  subquery: é uma sub-consulta que retorna linhas para uma tabela

- O número de colunas e seus tipos de dados na lista da cláusula INSERT deve corresponder ao número de valores e seus tipos de dados na sub-consulta.

UPDATE

- Sintaxe:

```
UPDATE  table
SET     column = value [, column = value]
 [WHERE condition];
```

- Você pode modificar linhas existentes utilizando o comando UPDATE.

table: é o nome da tabela

column: é o nome da coluna a aterar

value: é o valor correspondente ou sub-consulta para a coluna

condition: identifica as linhas a serem atualizadas; é composto de nomes de colunas, expressões, constantes, sub-consultas e operadores de comparação.

UPDATE

- Confirme a operação de atualização examinando a tabela e exibindo as linhas atualizadas.

- Nota: em geral, utilize a chave primaria para identificar uma única linha. Utilizar outras colunas pode causar a atualização indesejada de varias linhas.
  - Por exemplo, identificar uma única linha da tabela ATENDEIMENTOS através da data do atendimento é perigoso porque mais de um atendimento pode ter a mesma data.

Alterando Linhas em uma Tabela

- Linhas especificas são modificadas quando você utiliza uma clausula WHERE.
- Ex.:

```
UPDATE atendimento
SET desconto = valor * 0.15
WHERE     cod_atendimento = 1;
```

- Todas as linhas da tabela são modificadas se você omitir a cláusula WHERE.
- Ex.:

```
UPDATE atendimento
SET desconto = valor * 0.15;
```

- Nota: para desfazer as modificações utilize o comando ROLLBACK

Alterando Linhas em uma Tabela

- O comando UPDATE modifica linhas específicas, se a cláusula WHERE for especificada. O exemplo acima altera todos os descontos para 15% do valor do total de cada atendimento.
- Se você omitir a cláusula WHERE, todas as linhas da tabela serão modificadas.
- Verifique as modificações:

```
Select * from atendimento;
```

Atualizando com Sub-consultas Multiple-Column

Ex.:

```
UPDATE atendimento
SET   (desconto, valor) =
      (SELECT desconto,valor
      FROM  atendimento
      WHERE cod_atendimento = 4)
WHERE  cod_atendimento = 3;
```

- Sub-consultas multiple-column podem ser implementadas na cláusula SET de um comando.

DELETE

- Sintaxe:

  DELETE FROM table

  [WHERE concition];

- Você pode remover linhas utilizando o comando DELETE.

  table é o nome da tabela.

  condition identifica as linhas a serem apagadas; é composto de nomes de colunas, expressões, constantes, sub-consultas e operadores de comparação.

Removendo Linhas de uma Tabela

- Linhas específicas são removidas quando você utiliza a cláusula WHERE.
- Ex.:

```
DELETE FROM atendimento
WHERE cod_atendimento = 1;
```

- Todas as linhas da tabela são removidas se você omitir a cláusula WHERE.
- Ex.:

```
DELETE FROM atendimento
```

Removendo Linhas com Base nos valores de outra Tabela

- Ex.:

```
DELETE FROM exame
WHERE desconto < (SELECT desconto
                  FROM plano_saude
                  WHERE cod_plano=1);
```

- Você pode utilizar sub-consultas para remover linhas de uma tabela baseado em valores de outra tabela. O exemplo acima remove todos os itens de exames, onde o desconto do exame seja inferior a o desconto do plano de saúde de código igual a 1.