Banco de Dados - Ciclo 3 - Webaula 19

FURB

WEBAULA 19

- Manipulação de Dados
- Referências

QUESTION BANKS

Manipulação de Dados

A manipulação de dados compreende as operações de inclusão, alteração e exclusão dos dados nas tabelas, ou seja, com os comandos da DML você poderá:

- adicionar novas linhas a uma tabela;
- modificar linhas existentes em uma tabela;
- remover linhas existentes de uma tabela.

Lembre-se que “linha” é sinônimo de “registro” em uma tabela de banco de dados. Há também em algumas literaturas a referência ao termo “tupla” como sendo usual ao se referenciar uma linha. Em alguns SGBDs, como é o caso do Oracle, por default, quando uma das operações acima é executada, automaticamente uma transação é criada.

Dica

Uma transação é uma unidade lógica que contempla um ou mais comandos DML. Uma transação pode ser finalizada implícita ou explicitamente. A mais comum é a forma explícita, através dos comandos COMMIT (confirmar operações) ou então ROLLBACK (desfazer as operações). Vamos abordar isso mais tarde!

Vamos iniciar pela operação de inserção de dados. A representação gráfica da Figura 44 ilustra a inserção de uma nova linha na tabela DEPT.

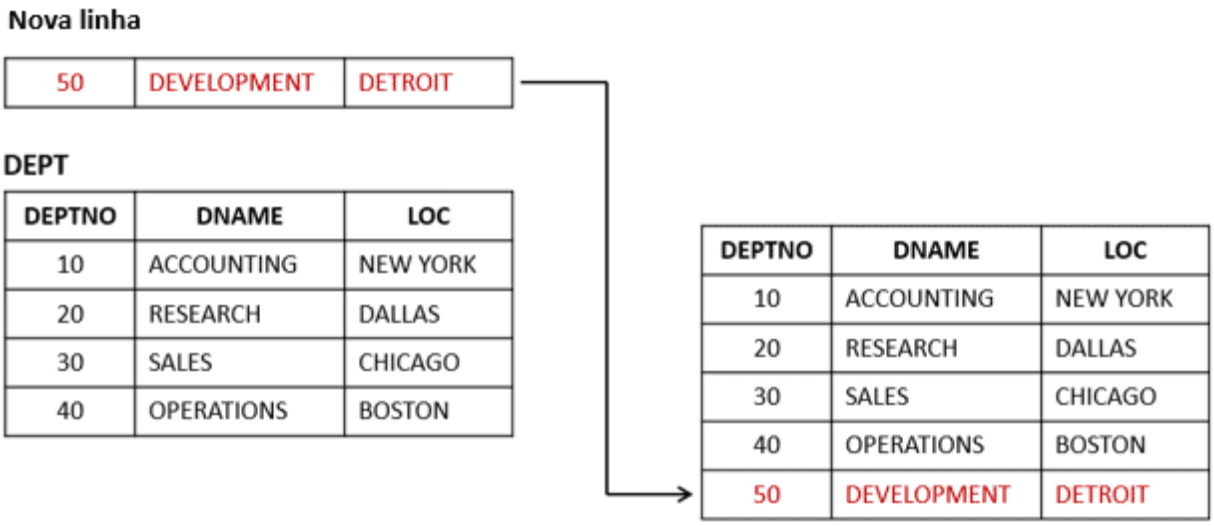

Figura 44

Para esta operação utilizamos o comando INSERT. O Quadro 30 apresenta a sintaxe deste comando.

Quadro 30 – Sintaxe do comando INSERT

```
INSERT INTO tabela
[ (coluna [, coluna...] ) ]

VALUES (valor [, valor...] ) ;
```

No quadro acima você pode observar as palavras reservadas INSERT INTO e em seguida o nome da tabela onde desejamos inserir a linha. Na sequência temos a possibilidade de informar em qual(is) coluna(s) queremos incluir valor(es). Neste caso, o nome da coluna a ser preenchida deve receber um valor correspondente a esta coluna. Para isso, temos a cláusula VALUES para indicar a sequência de

valores a serem incluídos. Por *default*, o comando INSERT aciona na tabela uma linha por vez, sendo que cada coluna deve apresentar um valor correspondente na respectiva cláusula Values.

Quando utilizamos o comando INSERT temos a possibilidade de listar os valores na ordem default em que as colunas foram criadas na tabela. Neste caso, para cada coluna da tabela deverá ser informado um valor ou uma indicação de nulo (*null*) para cada coluna. A outra possibilidade que temos é listar, opcionalmente, as colunas na cláusula INSERT em que efetivamente desejamos incluir valor. Vamos a exemplos para um melhor entendimento. O Quadro 31 apresenta uma sentença em que os nomes das colunas estão sendo informados e, conseguintemente, todas as colunas estão recebendo valores correspondentes.

Quadro 31 – Exemplo de sentença INSERT com colunas indicadas

```
INSERT INTO dept (deptno, dname, loc)

VALUES (50, 'DEVELOPMENT', 'DETROIT');
```

Já o exemplo apresentado no Quadro 32 omite a nomenclatura das colunas. Desta forma, necessariamente precisamos informar valores ou indicação de nulo (null) para todas as colunas na ordem em que elas foram criadas na tabela.

Quadro 32 – Exemplo de sentença INSERT com colunas omitidas

```
INSERT INTO dept

VALUES (50, 'DEVELOPMENT', 'DETROIT');
```

Dica

Coloque os valores de data e caractere entre apóstrofos (também conhecidos como aspas simples). Mas atenção: alguns SGBDs relacionais podem apresentar variações no uso de campos destes tipos de dados (data e caractere). Fique atento e não deixe de consultar a documentação, sempre que necessário.

Há situações em que vamos inserir dados em uma linha de uma tabela e não temos a necessidade de informar valores para todas as colunas. Neste caso, a(s) coluna(s) não selecionada(s) para receber valor deverá(ão) ser sinalizada(s) como nula(s) (*null*). Para isso, existem dois métodos que podemos utilizar. O primeiro é conhecido como método implícito. Neste, ao omitir a coluna cujo valor você não deseja inserir na lista de colunas, implicitamente a coluna receberá a informação de que está nula. O Quadro 33 apresenta uma sentença exemplificando este contexto.

Quadro 33 – Exemplo de sentença INSERT com coluna nula implícita

```
INSERT INTO    dept (deptno, dname )

VALUES         (60, 'MIS');
```

Observe que, no exemplo acima, ao omitirmos a coluna “loc” da tabela DEPT, implicitamente informamos que esta coluna (loc) é nula, ou seja, não há valor algum armazenado nela.

Porém, quanto utilizamos a estratégia de omitir as colunas no comando INSERT, necessariamente precisamos adotar o método explícito de indicação de nulidade de uma coluna. Para isso, é necessário que especifiquemos a palavra-chave NULL para o campo que se deseja ignorar (não informar um valor). O Quadro 34 apresenta um exemplo.

Quadro 34 – Exemplo de sentença INSERT com coluna nula explícita

```
INSERT INTO    dept
```

```
VALUES      (70, 'FINANCE' , NULL);
```

> **Dica**
>
> Para que você consiga omitir o preenchimento de uma coluna, ela não pode estar definida como NOT NULL ou fazer parte da chave primária da tabela.

Uma forma não muito usual, mas possível, é inserir dados numa tabela a partir de uma cópia realizada em outra tabela. Neste caso, costuma-se dizer que realizamos um INSERT a partir de uma consulta, ou melhor, uma subconsulta, visto que ela está inserida dentro de uma sentença principal – neste caso, o INSERT. Vejamos um exemplo através do Quadro 35.

Quadro 35 – Exemplo de sentença INSERT com subconsulta

```
INSERT INTO managers (id, name, salary, hiredate)
SELECT empno, ename, sal, hiredate
    FROM emp
    WHERE       job = 'MANAGER';
```

No exemplo acima podemos observar a inserção de uma ou mais linhas na tabela "managers". Sim, uma ou mais linhas, pois se o resultado da consulta interna (subconsulta) retornar várias linhas da tabela "emp", estas serão inseridas na tabela "managers".

Neste caso, cabe destacar dois aspectos:

- não use a cláusula VALUES, ou seja, utilize uma sub consulta;

- faça a correspondência do número de colunas na cláusula INSERT com o número de colunas na subconsulta, ou seja, exatamente o número de colunas retornadas na subconsulta deverá existir na tabela que receberá os valores das linhas.

> **Dica**
>
> No exemplo, observe que serão inseridos os dados dos empregados (tabela EMP) cuja atividade é “manager” (gerente). Algo similar você teve a oportunidade de conhecer na seção em que tratamos da criação e gerenciamento de tabelas.

Há situações em que necessitamos modificar o valor de uma ou mais colunas, seja em apenas uma linha ou mesmo em um conjunto de linhas de uma tabela. A representação gráfica a seguir (Figura 45) ilustra a modificação do valor da coluna DEPTNO no registro do funcionário "CLARK" da tabela EMP.

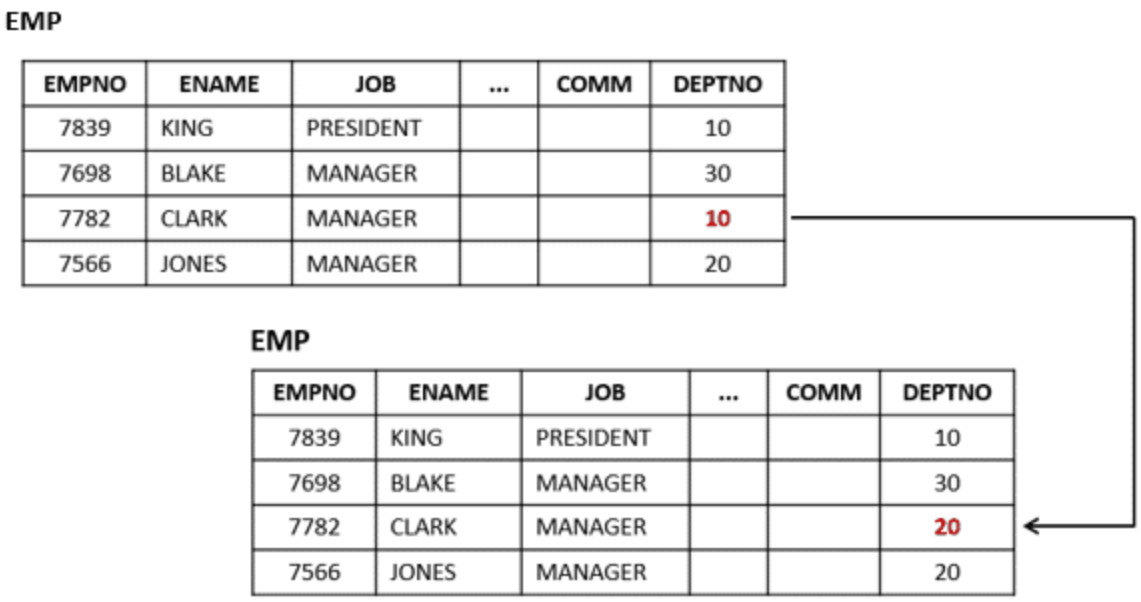

Figura 45

No exemplo acima, a coluna correspondente à linha do empregado CLARK (empno = 7782), que inicialmente continha o valor 10, passou para o valor 20. Para promovermos tal modificação na coluna utilizamos o comando UPDATE. O Quadro 36 apresenta a sintaxe do comando UPDATE.

Quadro 36 – Sintaxe do comando UPDATE

```
UPDATE      nome_tabela
SET          coluna = valor [, coluna = valor, ...]
[WHERE      condição];
```

Observe na sintaxe apresentada que após a palavra reservada UPDATE é necessário informar o nome da tabela em que se pretende modificar valor(es). Em seguida temos a palavra reservada SET, cuja função é definir qual(is) coluna(s), com o respectivo valor, será(ão) modificada(as). Por último temos a cláusula WHERE, cuja indicação sinaliza para opcionalidade (uso dos colchetes na sintaxe).

De fato, o uso da cláusula WHERE é opcional, mas atenção, pois a sua ausência significa que todos os registros serão afetados com a modificação pretendida.

Dica

Na maioria das vezes em que utilizamos o comando UPDATE o(s) campo(s) que compõe(m) a chave primária da tabela é(são) utilizado(s) na cláusula WHERE. Isso impede uma alteração em massa de linhas que não desejamos, mas que por alguma razão podem satisfazer a condição apresentada.

Vamos a um exemplo para um melhor entendimento. Observe que se trata do cenário apresentado na Figura 45, onde desejamos alterar o departamento de lotação do empregado “7782” (Clark). O Quadro 37 ilustra a sentença correspondente.

Quadro 37 – Exemplo de sentença UPDATE limitada

```
UPDATE      emp
SET          deptno = 20
 WHERE      empno = 7782;
```

Já o exemplo apresentado a seguir (Quadro 38) omite a cláusula WHERE, o que provoca uma alteração em massa, ou seja, todas as linhas da tabela para o valor do deptno (20).

Quadro 38 – Exemplo de sentença UPDATE ilimitada

```
UPDATE     emp
SET          deptno = 20 ;
```

Dica

Atenção! Não recomendamos a execução da sentença apresentada no Quadro 38, pois ela dificultará a execução dos demais exemplos que serão apresentados. O exemplo serviu apenas como ilustração.

Agora consideremos o desejo de modificar o departamento (deptno) de todos os empregados que atualmente estão vinculados ao departamento 10 e devem passar para o departamento 55. O Quadro 39 apresenta o comando a ser executado.

Quadro 39 – Sentença UPDATE gerando erro de integridade referencial

```
UPDATE      emp
SET          deptno = 55
WHERE        deptno = 10;
```

A execução da sentença acima deverá gerar uma mensagem de erro. Isso ocorre porque não existe o departamento 55 na tabela DEPT, e como há uma restrição de integridade referencial entre as tabelas EMP e DEPT, o SGBD relacional não permitirá a alteração, visto que somente valores existentes na tabela DEPT poderão ser utilizados na tabela EMP. O Quadro 40 apresenta um exemplo de mensagem de erro gerado no SGBD Oracle, o que deve ser similar nos demais SGBDs relacionais existentes no mercado.

Quadro 40 – Sentença UPDATE gerando erro de integridade referencial

```
UPDATE    emp
          *
ERROR at line 1:
ORA-02291: integrity constraint (USR. EMP_DEPTNO_FK)
Violated – parent key not found
```

Dica

Aqui há uma clara demonstração da importância da integridade referencial entre tabelas. É fundamental destacar que esta consistência é garantida pela existência das restrições de chave primária (PK) e *foreign key* (FK).

O terceiro comando do grupo DML é responsável pela funcionalidade de remoção de dados. Para melhor entendimento, observe a representação gráfica abaixo (Figura 46), que ilustra a remoção do departamento "DEVELOPMENT" da tabela *DEPT*.

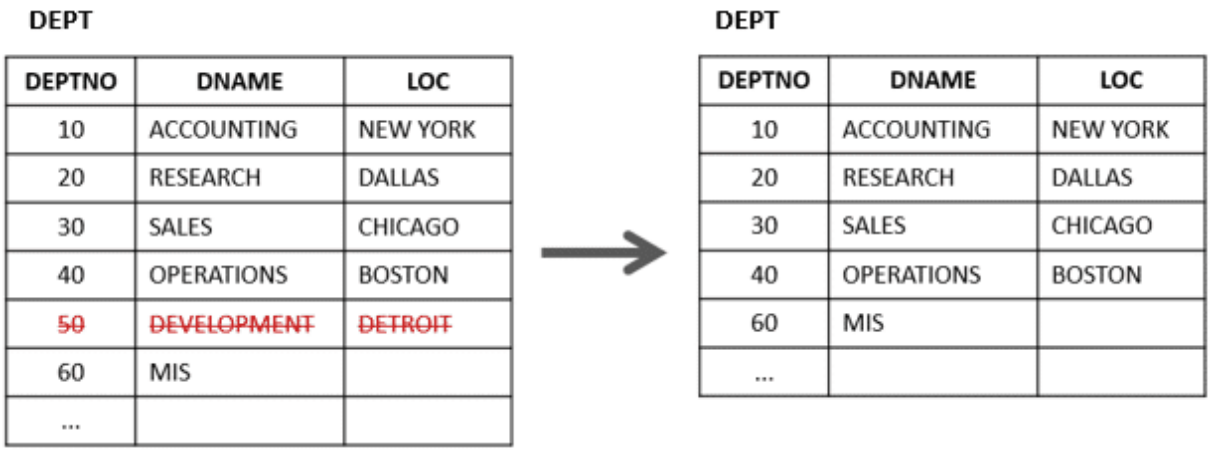

Figura 46

Para a execução de operações de remoção de dados em tabelas utilizamos o comando DELETE. A sintaxe do comando é apresentada no Quadro 41.

Quadro 41 – Sintaxe do comando DELETE

```
DELETE FROM       tabela
 [WHERE           condição];
```

Com o comando DELETE é possível remover mais de uma linha por vez, se necessário. Porém, fique atento à condição da cláusula WHERE, na qual a sintaxe apresentada encontra-se entre colchetes,

indicando que não é obrigatória sua utilização, pois a sua ausência significa que todas as linhas da tabela serão afetadas (neste caso, removidas).

Dica

Há SGBDs relacionais que não exigem a utilização da cláusula FROM no comando DELETE. Por padrão, todos a aceitam, daí sua utilização nos exemplos. Damos destaque a isso porque você poderá encontrar exemplos em que ela não é utilizada.

Passemos a um exemplo de utilização do comando DELETE. Observe o código da sentença apresentada no Quadro 42, onde o objetivo é remover o departamento 50. Aqui é importante destacar que o comando somente terá sucesso, ou seja, será executado, se o respectivo departamento (50) não estiver sendo utilizado na tabela EMP.

Quadro 42 – Exemplo de sentença do comando DELETE com cláusula WHERE

```
DELETE FROM     dept
WHERE           deptno = 50;
```

Dica

Na maioria das vezes o(s) campo(s) que compõe(m) a chave primária da tabela é(são) utilizado(s) na cláusula WHERE. Desta forma é possível garantir que apenas uma linha específica será atingida – neste caso, removida.

Agora, imagine a execução do código exemplo apresentado no Quadro 43. Naturalmente você irá concluir que todas as linhas na tabela serão removidas, pois foi omitida a cláusula WHERE.

Quadro 43 – Exemplo de sentença do comando DELETE omitindo cláusula WHERE

```
DELETE FROM      dept
```

> **Dica**
>
> Ao executar a sentença apresentada no Quadro 43, você vai perceber que o SGBD relacional não permitirá a exclusão das linhas, pois algumas das linhas afetadas estão sendo referenciadas na tabela EMP. Este é mais um exemplo da ação da integridade referencial.

Outra situação comum no uso do comando DELETE é quando ocorre uma tentativa de remoção de linhas e o SGBD relacional reporta um erro de restrição de integridade referencial. O Quadro 44 apresenta um código exemplo de uma sentença onde há a tentativa de remoção do departamento 10.

Quadro 44 – Exemplo de sentença do comando DELETE omitindo cláusula WHERE

```
DELETE FROM      dept
 WHERE           deptno = 10;
```

A execução da sentença apresentada acima no SGBD Oracle irá gerar a mensagem apresentada no Quadro 45. Isso ocorre porque não podemos remover uma linha que está sendo referenciada como chave estrangeira em outra tabela. No contexto, a tabela EMP apresenta uma chave estrangeira para a tabela DEPT, lembra-se?

Quadro 45 – Exemplo de sentença do comando DELETE omitindo cláusula WHERE

```
DELETE FROM dept
            *
ERROR at line 1:
ORA-02292 : integrity constraint (USR.EMP_DEPTNO_FK)
violated – child record found;
```

Dica

A execução da sentença SQL apresentada no Quadro 44 em outro SGBD relacional deve gerar mensagem de erro similar, desde que exista a indicação de regras de integridade referencial entre as tabelas envolvidas.

Encerramento

Nesta seção você teve a oportunidade de conhecer os três comandos do subgrupo DML encontradas nos SGBDs relacionais: INSERT, UPDATE E DELETE. Na próxima seção vamos abordar o principal comando da SQL: o comando SELECT. Iniciaremos pela parte da seleção básica de dados, avançando para recursos mais elaborados do próprio comando.

CONTINUE

Referências

Referências

ORACLE. Documentação de utilização do sistema (Versão 11g). Califórnia: Oracle Corp., 2017.

CARDOSO, Vírginia M. **Linguagem SQL**: fundamentos e práticas. São Paulo: Saraiva, 2009.

Referências de Imagens

Divisão de Modalidades de ensino (DME), Fundação Universidade Regional de Blumenau (FURB), 2019.