Centro Federal de Educação Tecnológica de Mato Grosso

UAB-CEFET-MT

CURSO DE TECNÓLOGO EM SISTEMAS PARA INTERNET
MODALIDADE A DISTÂNCIA

DISCIPLINA: FUNDAMENTOS DE BANCO DE DADOS

Professor Autor: Inara Aparecida Ferrer Silva

Unidade 7 – DML (DATA MANIPULATION LANGUAGE)

- **VISÃO GERAL DA UNIDADE**

O objetivo nesta unidade é especificamente auxiliar o aluno a compreender a segunda parte da linguagem SQL, chamada DML ou linguagem de manipulação de dados. Aqui são efetuadas as consultas e manipulações dos dados armazenados com os comandos DDL na linguagem SQL.

Objetivo Específico

Propomos, nesta unidade, levar você a:

- Entender e utilizar os comandos da linguagem DML- Linguagem de Manipulação de Dados .

7. LINGUAGEM SQL- DML (DATA MANIPULATION LANGUAGE)

Revisando o que já vimos temos que a linguagem SQL (Structured Query Language ou Linguagem de Consulta Estruturada) representa um conjunto de comandos responsáveis pela definição das tabelas, comandos e atualização dos dados em SGBD. É a linguagem padrão de definição e manipulação de banco de dados.

Neste momento já sabe o que é a linguagem SQL! Vamos seguir em frente.

Você também já conhece a primeira parte desta linguagem chamada DDL. Neste capítulo explicaremos os comandos básicos de manipulação de dados a DML.

Assim que as tabelas foram criadas, você pode começar a inserir dados no banco de dados, atualizá-los, alterar suas definições, eliminá-los e consultá-

los. Todas essas operações são denominadas como operações de manipulação de dados, a DML.

Os comandos de manipulação de dados discutidos neste capítulo são:

- INSERT
- UPDATE
- DELETE
- SELECT

Os comando utilizados seguem o padrão ANSI da SQL. O que deve ficar claro é que apesar de seguir um padrão pré-estabelecido temos pequenas diferenças de um fabricante para outro, nada que não consigamos compreender. Assim utilizaremos os comandos sem especificidades de cada fabricante.

UNIDADE VII- LINGUAGEM SQL- DML (DATA MANIPULATION LANGUAGE)

7. Linguagem SQL- Comandos DML
7.1 Comando Insert
7.2 Comando Update
7.3 Comando Delete
7.4 Comando Select

7-LINGUAGEM SQL- DML (DATA MANIPULATION LANGUAGE)

Como já citamos os comandos de manipulação de dados envolvem inserção de dados, consultas, atualizações nos dados inseridos e deleções nos dados da tabela.

Vamos começar com o comando insert, usado na adição de dados nas tabelas do BD. Com este comando nós vamos inserir dados em nossas tabelas.

7.1 –Comando INSERT

Objetivo:

Incluir um novo registro em uma tabela do Banco de Dados.

Sintaxe:

INSERT INTO <nome-tabela> (<nome-coluna>, <nome-coluna>)
VALUES (<relação dos valores a serem incluídos>);

Onde:

- <nome-tabela> - Representa o nome da tabela onde será incluída o registro.
- <nome-coluna> - Representa o nome da(s) coluna(s) terão conteúdo no momento da operação de inclusão.

Este comando pode ser executado de duas maneiras:

1) Se todos os campos da tabela tiverem conteúdo, é opcional especificar as colunas, entretanto a relação dos valores a serem incluídos deverão obedecer a mesma ordem (seqüência) utilizada na criação da tabela.

Exemplos item 1 anterior:

INSERT INTO aluno (codaluno,nome aluno, endereco)
VALUES (1, 'Marcio' , 'Rua 3 quadra 2');

INSERT INTO aluno
VALUES (2, ‘Julio’, ‘Rua Jacareí’);

INSERT INTO disciplina(coddisc, nomedisc)
VALUES (1, ‘matemática’);

7.2 – Comando UPDATE

Atualiza os dados de um ou um grupo de registros em uma tabela do Banco de Dados.

Sintaxe:

UPDATE <nome-tabela>
SET <nome-coluna> = <novo conteúdo para o campo>
WHERE <condição>

Onde:

<nome-tabela> - Representa o nome da tabela cujo conteúdo será alterado.

<nome-coluna> - Representa o nome da(s) coluna(s) terão seus conteúdos alterados com o novo valor especificado.

<condição> - Representa a condição para a seleção dos registros que serão atualizados. Este seleção poderá resultar em um ou vários registros. Neste caso a alteração irá ocorrer em todos os registros selecionados. Quando não se especifica a condição tem-se a atualização de todos os dados da tabela.

Exemplos:

UPDATE aluno
SET endereco = ‘Rua 15 quadra 1’
WHERE endereço=’Rua 3 quadra 2’;

UPDATE disciplina

SET nomedisc=’ informática basica’

WHERE nomedisc=’introducao a informatica’;

UPDATE professor

SET salario=(salário * 0,20)+salario;

Obs: o item anterior atualiza o salário de todos os professores da tabela professor.

7.3 – Comando DELETE

Objetivo:

Deletar um ou um grupo de registros em uma tabela do Banco de Dados.

Sintaxe:

DELETE FROM <nome-tabela>

WHERE <condição>

Onde:

<nome-tabela> - Representa o nome da tabela cujos registros serão deletados.

< condição> - Representa a condição para a deleção dos registros. Esta seleção poderá resultar em um ou vários registros apagados .

Este exemplo apaga todos os dados da tabela aluno:

DELETE FROM aluno;

Este exemplo apaga a disciplina cujo código =1:

DELETE FROM disciplina

WHERE coddisc=1;

Este exemplo apaga as disciplinas cuja carga horária é menor que 20 horas semanais:

DELETE FROM disciplina

WHERE cargahoraria <’20horas’;

7.4 –Comando SELECT

Objetivo:

Selecionar um conjunto de registros em uma ou mais tabelas que atenda a uma determinada condição definida pelo comando. A condição é opcional, podem-se selecionar todos os registros sem nenhuma condição.

Sintaxe:

SELECT *

FROM <nome-tabela>

WHERE <condição>

Onde:

<nome-tabela> - Representa o nome da(s) tabela(s) que contem as colunas que serão selecionadas ou que serão utilizadas para a execução da consulta.

<condição> - Representa a condição para a seleção dos registros. Este seleção poderá resultar em um ou vários registros.

< nome-coluna> - Representa a(s) coluna(s) cujos resultados são grupados para atender à consulta.

Exemplos:

Selecione todos os alunos da tabela aluno:

SELECT *

FROM aluno;

Selecione os alunos que residem a ‘Rua 1 bairro cpa’:

SELECT *
FROM aluno
WHERE endereço=‘Rua 1 bairro cpa’**;**

Selecione o nome das disciplinas cuja carga horária é maior que ‘40 horas’:

SELECT nomedisc, cargahoraria
FROM disciplina
WHERE cargahoraria >’40 horas’;

Para a WHERE têm-se algumas cláusulas específicas. São elas:

DISTINCT <nome-coluna>
GROUP BY <nome-coluna>
HAVING <condição>
ORDER BY <nome-campo> **ASC/ DESC**

Onde:

DISTINCT - Opção que mostra os valores obtidos na seleção eliminando as repetições.

GROUP BY – Mostra os campo(s) agrupados pela coluna especificada nesta cláusula para atender a consulta. A opção having estabelece uma condição para GROUP BY, só pode ser utilizado associada a este comando :

HAVING - Especifica uma condição para seleção de um grupo de dados. Esta opção só é utilizada combinada com a opção GROUP BY.

ORDER BY - Esta opção quando utilizada apresenta o resultado da consulta ordenado de forma crescente (ASC) ou decrescente (DESC) pelos campos definidos.

Exemplos utilizando cláusula order by:

```
SELECT nomedisc, cargahoraria
FROM disciplina
ORDER BY nomedisc;
```

Obs: Por default (padrão) a cláusula order by já ordena de modo ascendente, se você quiser em ordem descendente especifique após o nome do atributo a palavra DESC. Ex:

```
SELECT nomedisc, cargahoraria
FROM disciplina
ORDER BY nomedisc DESC;
```

Exemplo da cláusula GROUP BY e GROUP BY COM HAVING:

```
SELECT nomealuno, bairro, endereco
FROM aluno
GROUP BY bairro;
```

```
SELECT nomedisc, cargahoraria
FROM disciplina
GROUP BY  cargahoraria
HAVING cargahoraria>'20 horas';
```

Com a cláusula distinct abaixo tem-se a busca dos funcionários que possuem salários distintos, ou melhor, sem repetição.

Este exemplo seleciona os funcionários com seus salários sem repetição

SELECT nomefuncionario, DISTINCT salario
FROM Funcionário;

Este exemplo seleciona os funcionário com funções distintas
SELECT nomefuncionario, DISTINCT funcao
FROM funcionario;

CONECTORES LÓGICOS NO COMANDO SQL	
Utilizados para enumerar duas ou mais condições na condição WHERE do comando SELECT.	
OPERADOR	SIGNIFICADO
AND	"E" LÓGICO
OR	"OU" LÓGICO
NOT	NEGAÇÃO
PREDICADOS EM SQL:	
Um predicado é uma condição que pode ser calculada e produzirá um valor verdadeiro, falso ou ignorado. São utilizados na condição WHERE do comando SELECT.	
COMPARAÇÃO	=(IGUAL), <> (DIFERENTE), < (MENOR), > (MAIOR), <=(MENOR OU IGUAL) ,>= (MAIOR OU IGUAL)
INTERVALO	BETWEEN, NOT BETWEEN
DENTRO DE CONJUNTO DE VALORES	IN, NOT IN
COMPARAR CARACTERES	LIKE, NOT LIKE

Exemplos com operadores e predicados:

1)Selecione o nome e o salário dos funcionários que ganham mais do que R$ 400,00 reais.

```
SELECT  nomefuncionario, salario
FROM Funcionário
WHERE salário >400;
```

2)Selecione o nome e o salário dos funcionários que ganham R$ 400,00 reais ou mais.

```
SELECT  nomefuncionario, salario
FROM Funcionário
WHERE salário >=400;
```

3)Selecione o nome e o salário dos funcionários que menos que R$ 400,00 reais.

```
SELECT  nomefuncionario, salario
FROM Funcionário
WHERE salário <400;
```

4)Selecione o nome , a função e o salário dos funcionários que ganham mais de R$ 400,00 reais **e** a função é auxiliar administrativo.

```
SELECT  nomefuncionario, salario
FROM Funcionário
WHERE salário >400 and função =’auxiliar administrativo’;
```

Obs: O operador lógico and (E) só será verdadeiro, ou seja, só mostrará os resultados se as duas condições forem satisfeitas. Para o mesmo exemplo acima se modificásemos o enunciado dizendo que a seleção seria dos

funcionários com salário maior que R$ 400,00 **ou** função auxiliar administrativo. O resultado segue na cláusula abaixo:

```
SELECT  nomefuncionario, salario
FROM Funcionário
WHERE salário >400 or função =’auxiliar administrativo’;
```

```
SELECT  nomefuncionario, salario
FROM Funcionário
WHERE salário <400 or função <>’auxiliar administrativo’;
```

Obs: O operador lógico or (OU) só será verdadeiro, ou seja, só mostrará os resultados se uma ou outra condição for satisfeita, não precisa necessariamente ter as duas condições satisfeitas ao mesmo tempo. Se encontrar funcionários que ganhem mais do que R$400,00 ele mostrará registros na tela, também se encontrar só auxiliares administrativos mostrará registros. Isto não impede de mostrar registros que satisfaçam as duas condições, o operador OU se tiver uma condição verdadeira é considerado verdadeiro.

Agora que você já sabe utilizar os operadores lógicos and, or vamos ver exemplos com os predicados between, not between, like, in e not in.

Neste exemplo pedimos a consulta dos alunos cujos nomes terminem com a letra ‘a’ e morem no bairro Alvorada.

```
SELECT nomealuno, bairro
FROM aluno
WHERE bairro=’Alvorada’ and nomealuno like ‘%a’;
```

Neste exemplo pedimos a consulta dos alunos, cujos códigos matrícula estão entre 2 and 10, e seu bairro comece com a letra C .

SELECT codaluno**,** nomealuno, bairro
FROM aluno
WHERE bairro LIKE 'C%' and codaluno BETWEEN 2 AND 10;

Outra opção para o mesmo exercício utilizando predicado IN:

SELECT codaluno**,** nomealuno, bairro
FROM aluno
WHERE bairro LIKE 'C%' and codaluno IN (2,,3,4,5,6,7,8,9,10);

Neste exemplo pedimos a consulta dos alunos cujos nomes comecem com a letra T e terminem com a letra 'a' , mas seus códigos de matricula não estejam entre 1 e 5.

SELECT codaluno**,** nomealuno
FROM aluno
WHERE nomealuno LIKE 'T%a' and codaluno NOT IN (1,2,3,4,5);

No exemplo citado anteriormente o operador % quer dizer qualquer número de caracteres entre as letras T e a. Quando utilizado no início ou no fim tem a mesma função na posição indicada.

No exemplo a seguir pedimos a seleção dos registros da disciplina cujo nome comece com as letra MA e tenha carga horária entre 20 e 50 horas

SELECT nomedisc, cargahoraria

FROM disciplina

WHERE (cargahoraria BETWEEN 20 and 50) and (nomedisc LIKE ‘MA%’)**;**

Exercícios de auto-avaliação:

1. **Imagine que as tabelas abaixo estejam preenchidas e resolva os exercícios a seguir:**

Tabela Cliente
Campos: codcli, nome, rua, cidade

Tabela Agencia
Campos: codagen (Chave primaria),nomeagen, cidadeagen,ativo

Tabela conta
Numeroconta (chave primaria), codagen (chave estrangeira), saldo

Tabela depositante
Codcli (chaveprimaria), numero conta (chave primaria), tabela que surgiu do relacionamento N:N entre cliente e conta

Faça os comandos da linguagem SQL para os itens abaixo:

1) Insira dois registros para cada tabela do exercícios acima.
2) Atualize a os dados da tabela cliente acrescentando bairro Quilombo onde bairro era igual a CPA.
3) Atualize o saldo das contas em 10% onde a cidade é Cuiabá.
4) Selecione todos os clientes que morem na cidade de Cuiabá.
5) Selecione os cliente que morem na cidade de Cuiabá e tenham o nome terminado com a letra ‘o’.
6) Selecione as agencia cuja ativo é maior que R$100.000,00 reais e cidade comece com a letra c.
7) Selecione todas as agências que não estão em Cuiabá .
8) Selecione todas as contas cujo saldo é maior ou igual a R$ 2000,00 reais.
9) Selecione todas as contas cujo saldo está entre R$ 100,00 e R$1000,00 reais.
10) Selecione todas as contas, agencia e saldo da tabela conta menos das agencias de código 145, 146, 148 e 149.
11) Apague todos os dados das tabelas acima.

BIBLIOGRAFIA

MACHADO, Nery R.;ABREU, Maurício P. **Projeto de Banco de Dados: uma visão prática.** São Paulo: Editora Érica, 1996.

SILBERCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados- Tradução Daniel Vieira. Rio de Janeiro. Editora Elsevier, 2006.**

MACHADO, Felipe N. R. **Banco de Dados: Projeto e Implementação.** São Paulo, Érica : 2004.

TOREY, Toby J.; LIGHTSTONE, S.; NADEAU, T. **Projeto e Modelagem de Banco de Dados – Tradução de Daniel Vieira**. Rio de Janeiro: Elsevier, 2007.