descomplica

Essencial Database

Linguagem SQL - Comandos DML e DQL

Professora: Lúcia Contente Mós

Tutor: Marcelo Estruc

Introdução

É através da linguagem SQL que interagimos com o SGBD. Essa linguagem é composta de categorias de comandos. São elas DDL (comandos de definição de estruturas), DML (comandos de manipulação de dados, DCL (comandos de controle) e DQL (comandos de consulta).

Conhecer os comandos da linguagem SQL é fundamental para acessar o SGBD, manipular estruturas de armazenamento e seus dados.

Objetivos da aula

- Entender e Implementar Comandos DML (Data Manipulation Language) Insert, Update, Delete;
- Consultar Campos da tabela, ou seja, realizar operações de projeção;
- Consultar Registros da tabela usando a cláusula where, ou seja, realizar operações de seleção.
- Entender e Desenvolver comandos da linguagem SQL;

Resumo

Comandos DML

Os comandos da categoria DML (Data Manipulation Language), servem para realizar adições, eliminações e atualizações em dados de um ou mais registros de uma ou mais tabelas de maneira concorrente. Os comandos da categoria DML são:

- Insert – Adição de linhas com dados na tabela
- Update – Atualização de dados na tabela
- Delete – Eliminação de dados da tabela
- Commit – Confirmação das transações, ou seja, salvamento dos dados
- Rollback – Desistência das transações, ou seja, desfazer a última transação

Inserção de Dados

A adição de linhas com dados é um comando DML.

Sintaxe:

INSERT INTO *tabela* [(*coluna* [*, coluna...*])] **VALUES** *(valor* [*, valor...*])*;*

Onde:

Tabela	é o nome da tabela
Coluna	é o nome da coluna a ser preenchida, a lista de colunas pode ser omitida, neste caso devem ser informados valores para todas as colunas.
Valor	é o valor correspondente para a coluna. Os valores de data e caractere devem ser informados entre aspas simples.

Para verificar a ordem default das colunas de uma tabela e o tipo de dado esperado utilize a instrução describe.

SQL> DESCRIBE dept

Name	**Null?**	**Type**
----------	**----------**	**----------**
DEPTNO	NUMBER(2)	NOT NULL
DNAME	VARCHAR2(14)	
LOC	VARCHAR2(13)	

Exemplo 1: Instrução completa para inserção de dados

```
SQL> INSERT INTO        dept (DEPTNO, DNAME, LOC)
  2  VALUES           (80, 'COMERCIAL', 'SP');
```

Exemplo 2:

```
SQL> INSERT INTO        dept
  2  VALUES           (90, 'FINANCEIRO', 'RJ');
```

Para inserir nulos, nos campos, pode-se executar os exemplos abaixo:

Exemplo 3:

```
SQL> INSERT INTO        dept (deptno, dname )
  2  VALUES           (55, 'RH');
```

A tabela dept contém as colunas deptno, dname e loc, conforme se observa, no exemplo anterior estão sendo inseridos valores apenas às colunas deptno e dname, neste caso a coluna loc irá conter NULL.

Exemplo 4:

```
SQL> INSERT INTO        dept
  2  VALUES           (45, 'DIRETORIA', NULL);
```

Adição de Linhas, com datas

Exemplo 5:

```
SQL> INSERT INTO emp
  2  VALUES      (3000,'JOSE','VENDEDOR',7782, '30/03/23',
  3        2800, NULL, 20);
```

Para visualizar os dados, execute o comando abaixo:

```
SQL> Select *
2    from emp;
```

Atualização de dados nos Registros das Tabelas

Sintaxe:

UPDATE *tabela*
SET *coluna* = *valor* [, *coluna* = *valor, ...*]
[WHERE *condição*];

onde:

Tabela	é o nome da tabela
Coluna	é o nome da coluna a ser preenchida, podem ser atualizados os dados de várias colunas
Valor	é o valor correspondente ou subconsulta para a coluna
Condição	é uma cláusula opcional e identifica as linhas a serem atualizadas de acordo com uma condição que pode ser composta por expressões de comparação ou subconsultas

Exemplo: Atualizar o número departamento para 30 do empregado com código 7900:

```
SQL> UPDATE      emp
  2  SET       deptno = 30
  3  WHERE     empno = 7900;
```

Quando a cláusula WHERE é omitida, todas as linhas da tabela sofrem a alteração.

Eliminação de Registros da Tabela

Sintaxe:

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

Tabela: é o nome da tabela

Condição: Está cláusula é opcional e identifica as linhas a serem eliminadas de acordo com uma condição que pode ser composta por expressões de comparação ou subconsultas

Exemplo:

SQL> **DELETE** FROM dept
2 **WHERE** dname = 'FINANCIERO';

Quando a cláusula WHERE é omitida, todas as linhas da tabela são eliminadas.

Comando Select

Este comando serve para fazer consultas aos dados da tabela.

Exemplo do Comando Select

Select *
From emp;

Sintaxe do comando

SELECT é uma lista de uma ou mais colunas
DISTINCT suprime os itens duplicados
* seleciona todas as colunas
coluna seleciona a coluna nomeada

apelido fornece cabeçalhos diferentes às colunas selecionadas
FROM *tabela*especifica a tabela contendo as colunas

Normas para uso dos Comandos da Linguagem SQL

Não há diferença no processamento dos commandos entre maiúsculas e minúsculas.
Os commandos podem ter uma ou mais linhas.
As palavras dos comandos não podem ser abreviadas.
Fica mais fácil entender os commandos quando usa-se linhas separadas .

Para mostrar alguns campos da tabela, é necessário nomear quais campos devem ser listados, como no exemplo abaixo.

Select deptno, dname
From dept;

Operadores aritméticos

Os operadores aritméticos podem ser utilizados em qualquer cláusula de uma instrução SQL com exceção da cláusula FROM.

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão

Tabela 1 - Operadores Aritméticos - Fonte: Goya, 2013.

Exemplo: Mostrar o nome de todos os empregados da tabela emp, os seus salário e salários acrescidos de 300,00

SELECT ename, sal, sal+300

FROMemp;

Resultado:

ENAME	SAL	SAL+300
KING	5000	5300
BLAKE	2850	3150
CLARK	2450	2750
JONES	2975	3275
MARTIN	1250	1550
ALLEN	1600	1900
...		

Perceba, que foi gerada uma nova coluna, somente para exibição dos dados, como se fosse uma simulação, com a operação aritmética do salário acrescido de 300. As regras matemáticas e a ordem de precedência continuam em vigor para qualquer expressão que seja construída, conforme exemplo abaixo.

Exemplo:

```
SQL> SELECT ename, sal, sal + ((sal * 30) / 100)
2  FROM       emp;
```

Utilize, os parênteses, para interferir e/ou isolar a ordem de precedência da execução das operações.

Operadores de comparação

São utilizados para estabelecer uma relação de comparação entre valores ou expressões. O resultado desta comparação é sempre um valor lógico (booleano) verdadeiro ou falso. Estes operadores podem ser utilizados em comparações que envolvem apenas uma linha.

Operador	Descrição	Operador	Descrição
=	Igual	<=	menor ou igual a
<>	diferente	between v1 and v2	Entre dois valores (inclusive)
>	maior do que	in (lista de valores)	compara a coluna aos valores de uma lista
<	menor do que	like	Vincula um padrão de caractere
>=	maior ou igual a	is null	É um valor nulo

Tabela 2 - Operadores de Comparação - Fonte: Goya, 2013.

Operador Between: Os valores especificados com o operador BETWEEN são inclusivos, deve-se especificar primeiro o limite inferior. A instrução SELECT abaixo retorna as linhas da tabela EMP para qualquer funcionário cujo salário esteja entre US$1.000 e US$1.500.

```
SQL> SELECT       ename, sal
  2  FROM     emp
  3  WHERE  sal BETWEEN 1000 AND 1500;
```

Operador IN: Para verificar a relação de valores com uma determinada lista, use o operador IN. Pode ser utilizado com qualquer tipo de dados. O exemplo seguinte retorna uma linha da tabela EMP para qualquer funcionário cujo nome estiver incluído na lista de nomes na cláusula WHERE:

```
SQL> SELECT       empno,  ename,  mgr, deptno
2       FROM          emp
3       WHERE         ename IN ('FORD' , 'ALLEN');
```

Se forem utilizados caracteres ou datas na lista, eles devem estar entre aspas simples ('').

Como segue no exemplo:

```
SQL> SELECT  empno,  ename,  mgr, deptno
  2       FROM    scott.emp
  3   WHERE  JOB IN ('PRESIDENT', 'MANAGER');
```

Operador LIKE: É possível consultar registros que associem um padrão de caracteres usando o operador LIKE. A associação de um padrão de caracteres refere-se a uma consulta. Dois símbolos podem ser utilizados para construir a string de pesquisa.
A instrução SELECT abaixo retorna o nome do funcionário da tabela EMP para qualquer funcionário cujo nome começa com "J". Note o "J". maiúsculo. Os nomes iniciados com "j" não retornarão.

Exemplo:

```
SELECT      ename
FROM        SCOTT.emp
WHERE       ENAME LIKE 'J%';
```

Operador IS NULL: Mostra a correspondência com nulos. Um nulo significa que não há valor, ou seja, ausência de valor. Assim, é impossível verificar a correspondência com a operadora igual (=) pois um nulo, não pode ser igual ou diferente de qualquer valor. O exemplo abaixo mostra o nome, a função e o número do departamento de todos os empregados que não ganham comissão.

```
select ename, job, deptno
from scott.emp
where comm IS NULL;
```

Operadores lógicos

Um operador lógico junta o resultado de duas condições de componente para gerar um resultado único com base nesses componentes ou inverter o resultado para uma condição única. Três operadores lógicos estão disponíveis na linguagem SQL:

Operador	Descrição
AND	Retorna verdadeiro se todas as expressões envolvidas na operação forem verdadeiras
OR	Retorna verdadeiro se pelo menos uma expressão envolvida na operação for verdadeira
NOT	se o resultado da expressão for verdadeira retorna falso, caso contrário retorna verdadeiro

Tabela 3- Operadores Lógicos - Fonte: GOYA,2013

Estes operadores são utilizados para combinar expressões que serão utilizadas na cláusula **WHERE,** seja em consultas ou nas instruções de alteração e exclusão de dados.

Exemplo utilizando o operador AND: Mostrar o código, nome, cargo e salário de todos os empregados que possuem salário superior ou equivalente a 1100 e cargo equivalente a SALESMAN.

```
SELECT empno, ename, job, sal
FROM   scott.emp
WHERE  sal >= 1100 AND job = 'SALESMAN';
```

Resultado:

```
EMPNO     ENAME     JOB          SAL
7876      ADAMS     SALESMAN          1100
7934      MILLER    SALESMAN          1300
2 rows selected.
```

O operador AND funciona quando as duas condições forem verdadeiras, caso contrário, ou seja se uma das condições não for verdadeira, não serão selecionadas as linhas.
No exemplo abaixo, as duas condições devem ser verdadeiras para cada registro a ser selecionado. Assim, um funcionário que possua a função CLERK ***e*** receba mais de US$1.100 será selecionado.

Nas pesquisas que envolvem dados alfanuméricos existe distinção entre maiúsculas e minúsculas e estes valores devem estar entre aspas simples. No caso do exemplo acima não será retornada nenhuma linha se SALESMAN não estiver em letra maiúscula.

Exemplo utilizando o operador OR: Mostrar o código, nome, cargo e salário de todos os empregados que possuem salário superior ou equivalente a 1100 e cargo igual a SALESMAN

```
SQL> SELECT empno, ename, job, sal
2  FROM   emp
3  WHERE  sal>=1100 OR     job='SALESMAN';
```

Resultado:

```
EMPNO     ENAME      JOB            SAL
7839            KING       PRESIDENT      5000
7698      BLAKE      MANAGER        2850
7782      CLARK      MANAGER        2450
7566      JONES      MANAGER        2975
7654      MARTIN     SALESMAN       1250
...
```

O operador OR funciona quando pelo menos uma das condições for verdadeira. Assim, um funcionário que possua a função SALESMAN *ou* que receba mais de $1.100 será selecionado.

Exemplo utilizando o operador NOT: Mostrar o nome e a função de todos os empregados que *não* possuem os cargos CLERK, MANAGER ou ANALYST.

```
SELECT ename, job
FROM   scott.emp
WHERE  job NOT IN ('CLERK','MANAGER','ANALYST');
```

Resultado:

```
ENAME      JOB
KING       PRESIDENT
MARTIN     SALESMAN
ALLEN      SALESMAN
TURNER     SALESMAN
WARD       SALESMAN
```

O operador NOT nega uma condição. E pode ser utilizado também com outros operadores da linguagem SQL, como BETWEEN, LIKE e NULL.

Exemplos:

```
... WHERE  job   NOT  IN ('SALESMAN', 'MANAGER')
... WHERE  sal   NOT  BETWEEN  25000 AND  5000
... WHERE  ename NOT  LIKE '%S%'
... WHERE  comm  IS   NOT  NULL
```

Conteúdo bônus

Salvando ou Desfazendo transações

Para o usuário uma transação parece uma simples operação, por exemplo transferir o dinheiro de uma conta corrente para outra o usuário informa os dados requeridos para que a operação de transferência seja realizada e recebe uma notificação de conclusão da operação. Por outro lado, para que a operação seja

realizada com sucesso, sem a ocorrência de falhas, uma série de operações devem ser realizadas, no caso do exemplo de transferência de valores de uma conta para outra de maneira bem simplificada:

- O dinheiro precisa ser debitado de uma conta, portanto uma operação de atualização de dados deve ser realizada nesta conta.
- O dinheiro precisa ser creditado em outra conta, portanto outra operação de atualização de dados deve ser realizada nesta outra conta.

Sendo assim, uma transação é um conjunto de operações DML que são realizadas para concluir uma determinada tarefa.

Para garantir a integridade dos dados é necessário que as transações assegurem:

- A consistência de dados.
- A atomicidade – todas as operações devem ser refletidas corretamente no banco ou então nenhuma das operações deverá ser realizada.
- A integridade da base de dados.

Quando não ocorrem falhas no processamento das operações de uma transação ela pode ser efetivada, neste caso as modificações são refletidas fisicamente no banco de dados. Enquanto isso não ocorre, as modificações são refletidas apenas em memória e podem ser desfeitas ou descartadas.
O controle de transações também permite que as alterações realizadas, possam ser visualizadas antes de se tornarem permanentes, e que as operações relacionadas logicamente possam ser agrupadas.
O controle das transações se dá através dos comandos commit e rollbback. A emissão de um commit salva as transações, isto é, efetiva as manipulações de dados realizadas nas tabelas. A emissão de um rollback desfaz as transações que ainda não tenham sido “comitadas”.
Uma transação inicia quando é emitido algum comando DML como Insert, Update ou Delete e é finalizada com um dos seguintes eventos:

- A emissão de uma instrução COMMIT ou ROLLBACK.

- A execução de uma instrução DDL ou DCL, nesse caso ocorre um commit implícito.
- Quando o usuário sai da aplicação ou houver uma falha no computador ou o sistema falhar.
- A saída normal da aplicação, sem emitir explicitamente COMMIT ou ROLLBACK.

Um ROLLBACK automático ocorre quando há uma finalização anormal da aplicação ou queda do sistema.

Exemplo usando commit: Alterar a tabela EMP e definir o número de departamento para o funcionário 7900 como 10 e depois confirmar a atualização.

```
UPDATE      scott.emp
SET   deptno = 10
WHERE       empno = 7900;
COMMIT;
```

Exemplo do comando rollback: Excluir todas as linhas da tabela employees

```
Delete
From hr.employees;
```

```
ROLLBACK;
```

Com o comando Rollback, a alteração provocada é desfeita e a versão original da tabela é restabelecida.

Referências Bibliográficas

DATE, C. J. **Introdução a sistemas de banco de dados**. Rio de Janeiro. Ed. Campus, 1991.

CHEN, Peter. **Modelagem de dados: a abordagem entidade-relacionamento para projeto lógico**. São Paulo: Makron Books, 1990.

MEDEIROS, L. F. **Banco de dados, princípios e práticas**, 1ª. ed., Ed. Intersaberes, 2013.

PUGA, S., França E., GOYA M., **Banco de dados: Implementação em SQL, PL/SQL e Oracle 11g**, Ed. Pearson, 2013.

ELMASRI R., NAVATHE, S. **Sistemas de Banco de Dados**, 4ª ed. Ed. Pearson, 2005.