BANCO DE DADOS

AULA 4

Prof. Lucas Rafael Filipak

CONVERSA INICIAL

Segue a apresentação da aula com a estrutura de contéudos que vão ser trabalhados:

1. *Data Manipulation Language* (DML):
   1.1 Inserir registros.
2. Selecionar registros:
   2.1 Restringir consultas;
   2.2 Ordenar consultas.
3. Outros comandos de restrição de consulta;
4. Editar e apagar registros:
   4.1 Editar registros;
   4.2 Apagar registros.

O objetivo deste encontro é aprender as instruções/comandos da linguagem SQL que fazem parte do grupo de manipulação de dados (DML), compreendendo suas sintaxes e aplicações. Os comandos da categoria DML são divididos em quatros grandes grupos de comandos: INSERT, SELECT, UDPADE e DELETE. Nesta aula, vão ser contemplados os comandos dos quatro grupos.

TEMA 1 – *DATA MANIPULATION LANGUAGE* (DML)

Com visto anteriormente, os comandos SQL são classificados em três categorias, e a DML é a categoria responsável para fazer a manipulação dos dados. A manipulação consiste em comandos de quatro grandes grupos.

1. Inserir (INSERT);
2. Selecionar (SELECT);
3. Atualizar (UPDATE);
4. Apagar (DELETE).

A Figura 1 representa os principais comandos de manipulação de dados, conforme o padrão ANSI/ISO.

Figura 1 – Comandos de manipulação

Comando	Função
INSERT INTO	Inserir um novo registro na tabela de dados.
DELETE FROM	Apagar um ou mais registros de uma tabela de dados.
UPDATE	Permitir que os dados de um registro sejam atualizados.
SELECT FROM	Selecionar um conjunto de registros a partir de uma condição e retorná-los ao usuário.

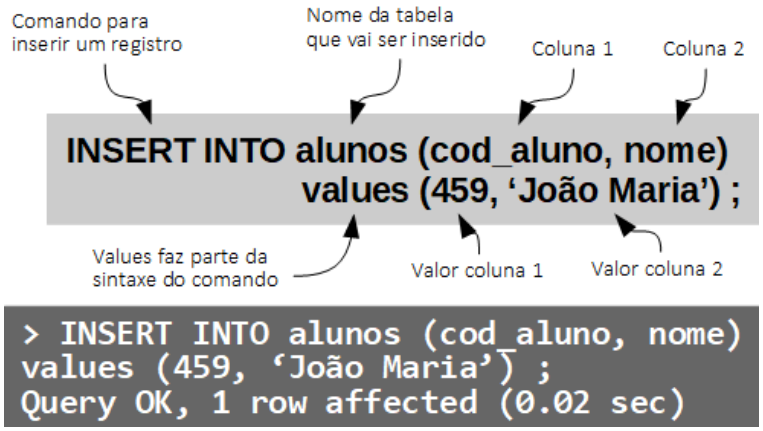
Fonte: Alves, 2014, p. 128.

Esses são os comandos principais, mas eles podem ter variações (cláusulas, intervalos e outros comandos) que podem modificar o resultado dos dados.

1.1 Inserir registros

Foi visto antes como criar o banco de dados e como criar, modificar e deletar as tabelas. Neste momento serão trabalhados os comandos da categoria DML. O primeiro passo é inserir os dados nas tabelas.

Figura 2 – Exemplo do uso do comando INSERT



O comando INSERT (INSERT INTO) é utilizado para inserir os dados nas tabelas. Note que na Figura 2 há dois blocos delimitados pelos parênteses que são separados pela palavra *values*. No primeiro bloco, são declaradas as colunas da tabela nas quais se pretende inserir os dados, sendo estas declaradas exatamente como estão definidas na estrutura da tabela. Após o comando *values*, encontra-se o segundo bloco delimitado pelos parênteses, em que são informados os dados a serem inseridos na tabela, exatamente na ordem em que foram declaradas as colunas da tabela no primeiro bloco.

Figura 3 – Exemplo do uso do comando INSERT

```
INSERT INTO alunos (cod_aluno, nome)
values (459, 'João Maria') ;
```

Na Figura 3 é possível visualizar a ordem de atribuição de valores. A coluna *cod_aluno* da tabela *alunos* recebe o dado 459. A coluna *nome* da tabela *alunos* recebe o dado "João Maria". Isso se dá pela ordem de escrita no comando INSERT, a primeira coluna do primeiro bloco de parênteses é *cod_aluno* e a primeira coluna do segundo bloco de parênteses é 459. Outras informações importantes:

- A ordem de escrita das colunas no comando INSERT não precisa ser na ordem que eles aparecem na estrutura da tabela;
- Os dados que vão ser inseridos que forem do tipo texto são obrigados a usar aspas simples;
- Não é preciso informar todas as colunas da tabela no comando INSERT (somente os definidos como NOT NULL).

Outra prática para a execução do comando INSERT é omitir o nome das colunas. Observe a Figura 4.

Figura 4 – Exemplo do uso do comando INSERT omitindo as colunas

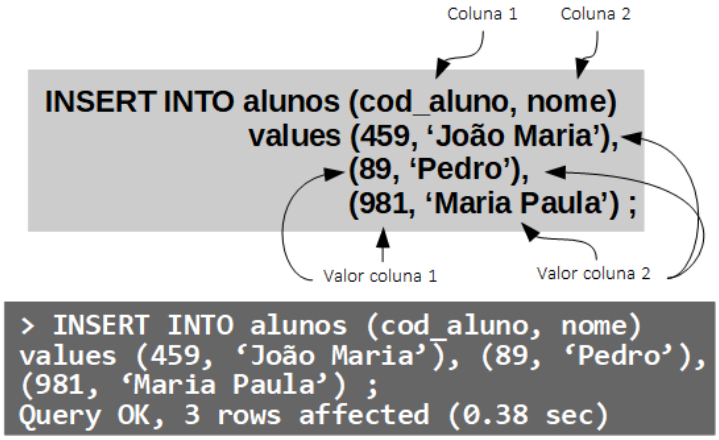
```
INSERT INTO alunos values (459, 'João Maria') ;
```

Com a omissão das colunas no comando INSERT, deve-se prestar atenção a dois pontos importantes:

1. A ordem dos dados é exatamente a ordem em que as colunas estão definidas na tabela;
2. Todas as colunas da tabela devem estar informadas, ainda que o valor seja nulo.

O procedimento de omissão das colunas no comando INSERT geralmente é utilizado quando se pretende realizar uma inclusão com dados para todas as colunas da tabela. O comando INSERT também permite que sejam inseridos vários registros, ao mesmo tempo (aproveitando o mesmo comando INSERT), em uma mesma tabela.

Figura 5 – Exemplo do uso do comando INSERT com vários registros



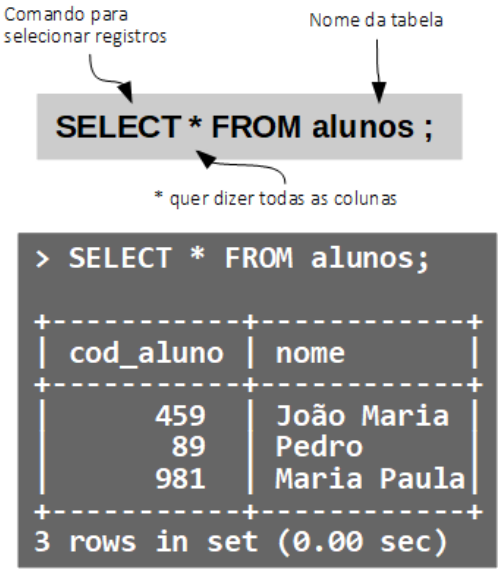
O comando da Figura 5 executou o comando INSERT, realizando a inclusão de três registros na tabela *alunos*. Podemos observar que na mensagem "Query OK" de execução do comando SQL, existem três linhas afetadas, em virtude da inclusão de três registros.

TEMA 2 – SELECIONAR REGISTROS

Uma das funções mais importantes do banco de dados é, além de armazenar os dados, recuperar (consultar) os dados, permitindo, assim, que o usuário, além de consultar esses dados, consiga também tomar decisões em cima das consultas ou do cruzamento dos dados entre duas ou mais tabelas. A Figura 6 traz um exemplo da seleção mais simples que podemos fazer em uma tabela.

Ao executar o comando apresentado na Figura 6, são buscados todos os registros da tabela *alunos*, como também são retornadas todas as colunas existentes na tabela (*cod_aluno* e *nome*) em virtude de ter sido utilizado o símbolo * na consulta.

Figura 6 – Comando SELECT



Pode-se também escolher as colunas que aparecerão na consulta. A Figura 7 traz o comando SELECT informando as colunas que vão ser visualizadas. No exemplo, foi buscada apenas a coluna *nome*, mas podem ser buscadas outras colunas da tabela em que a consulta está sendo realizada.

Figura 7 – Exemplo do uso do comando SELECT

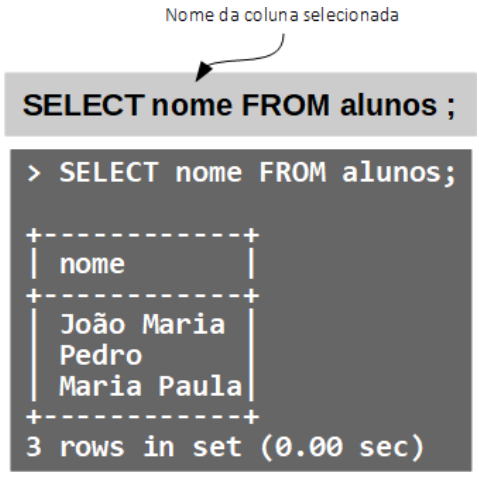


Figura 8 – Exemplo do uso do comando SELECT escolhendo as colunas

```
SELECT nome, idade, sexo FROM funcionarios ;
```

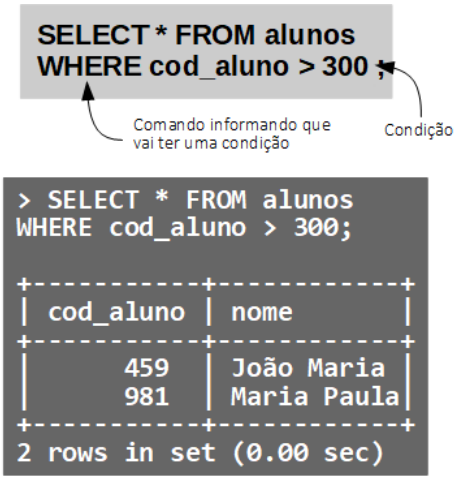
No exemplo da Figura 8, o comando SQL é um exemplo em que são pesquisadas apenas algumas colunas da tabela, sendo retornados apenas os registros das colunas *nome*, *idade* e *sexo* da tabela *funcionários*. O resultado

será apresentado na ordem informada no SELECT: primeiro o nome, depois a idade e, por último, o sexo.

2.1 Restringir consultas

A cláusula WHERE pode ser utilizada nos comandos SELECT, UPDATE e DELETE. É um comando usado para filtrar os registros, delimitando a pesquisa do comando a características específicas de um dado ou relacionamento. O WHERE informa que a seleção vai depender de uma condição, que deve ser declarada logo após a palavra WHERE.

Figura 9 – Exemplo do uso do comando SELECT com condição



Na Figura 9, a cláusula WHERE definiu que a consulta buscaria apenas os registros da tabela *alunos* em que o *cod_aluno* fosse maior que 300. Note que foi usado um operador relacional “>” de modo a especificar a delimitação da busca dos dados. A Tabela 1 mostra os operadores relacionais.

Tabela 1 – Operadores condicionais

>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
=	Igual
<>	Diferente

Para os próximos exemplos, a Figura 10 será utilizada como tabela base para as consultas.

Figura 10 – Tabela fornecedores

cod	nome	idade	sexo	funcao
1	Augusto	30	M	pintor
2	Paula	23	F	gerente
3	Maria	45	F	recepção
4	César	36	M	pintor
5	João Vitor	18	M	auxiliar

Para elaborar um comando SQL que retorne apenas os registros de fornecedores do sexo feminino (F), cadastrados na tabela *fornecedores,* é importante identificar a coluna a ser utilizada para restringir os dados pesquisados por meio da cláusula WHERE. A coluna *sexo* é onde consta essa informação, armazenando M para os homens e F para as mulheres, então a coluna *sexo* vai ser utilizada.

Figura 11 – Exemplo do uso do comando SELECT restringindo o sexo feminino (F)

```
SELECT * FROM fornecedores WHERE sexo = 'F' ;
```

cod	nome	idade	sexo	funcao
2	Paula	23	F	gerente
3	Maria	45	F	recepção

2 rows in set (1.02 sec)

Na Figura 9, o valor da coluna *cod_aluno* > 300 não estava entre aspas, pois é um número. Já na condição da pesquisa, apresentada na Figura 11, foi colocado entre aspas simples por se tratar de um dado do tipo caractere.

A cláusula WHERE pode conter mais de uma condição, e para isso utilizam-se os conectores lógicos (AND e/ou OR).

- AND – Todas as condições devem ser verdadeiras, para que o conjunto seja verdadeiro;
- OR – Se apenas uma das condições é verdadeira, o conjunto se torna verdadeiro.

Figura 12 – Exemplo do uso do operador AND

```
SELECT * FROM fornecedores WHERE sexo = 'M'
AND idade < 30 ;
```

```
+------+-----------+-------+------+------------+
| cod  | nome      | idade | sexo | funcao     |
+------+-----------+-------+------+------------+
|  5   |João Vitor |   18  |  M   | auxiliar   |
+------+-----------+-------+------+------------+
1 row in set (0.02 sec)
```

Na Figura 12, são selecionados apenas os registros que possuem a coluna *sexo* preenchida com o valor M e a coluna *idade* menor que 30. Como foi utilizado o conector AND, os dados pesquisados atendem às duas condições. Por esse motivo, o registro encontrado é de um fornecedor do sexo M e a sua idade é 18, ou seja, menor que 30.

Figura 13 – Exemplo do uso do operador OR

```
SELECT * FROM fornecedores WHERE sexo = 'F'
OR idade < 30 ;
```

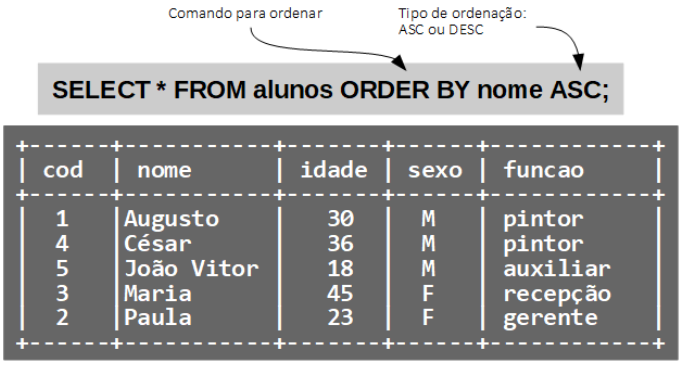
```
+------+-----------+-------+------+------------+
| cod  | nome      | idade | sexo | funcao     |
+------+-----------+-------+------+------------+
|  2   |Paula      |   23  |  F   | gerente    |
|  3   |Maria      |   45  |  F   | recepção   |
|  5   |João Vitor |   18  |  M   | auxiliar   |
+------+-----------+-------+------+------------+
3 rows in set (0.02 sec)
```

Com a utilização do conector OR na sentença da Figura 13, o comando SELECT traz os dados dos fornecedores que possuem o *sexo* igual a F (Feminino) ou a *idade* menor que 30. Analisando a Figura 13, o comando SELECT retornou os registros com o cod = 2 e o cod = 3 por se tratarem de registros do sexo feminino, mas também retornou o registro com o cod = 5, o "João Vitor", mesmo ele sendo do sexo masculino (coluna sexo = M). Isso ocorre porque a outra condição (idade < 30) é verdadeira. Observe que o conteúdo da coluna idade dele é 18. Lembrando que quando é utilizado o conector OR, apenas uma das condições precisa ser verdadeira.

2.2 Ordenar consultas

É possível ordenar a apresentação dos resultados buscados por meio de um comando SELECT.

Figura 14 – Exemplo do uso do comando ORDER BY ASC



Observe na Figura 14 que o comando ORDER BY foi adicionado na instrução SQL, seguido do nome da coluna na qual os dados devem ser ordenados, e também é possível acrescentar o tipo de ordenação. Há dois tipos de ordenação: ASC (ascendente) e DESC (decrescente), como pode ser visto na Tabela 2.

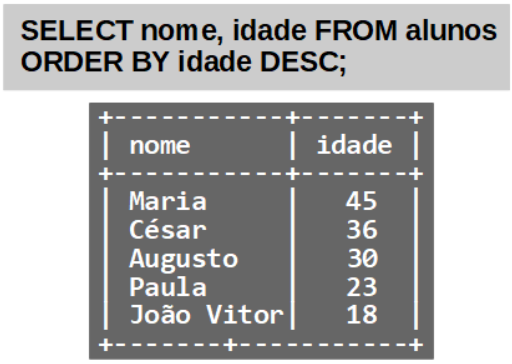
Tabela 2 – Ordem crescente e decrescente

ASC	Números	A – Z
	Texto	Menor para o maior

DESC	Números	Z – A
	Texto	Maior para o menor

A Figura 15 é um exemplo de utilização do comando ORDER BY. Note que a ordenação é pela coluna *idade* de maneira decrescente.

Figura 15 – Exemplo do uso do comando ORDER BY DESC



Em uma instrução SQL pode-se usar várias ordenações. Analisando a Figura 16 a ordem será primeira em ordem alfabética, do A – Z (nome ASC) e depois pela idade, do mais velho para o mais novo (idade DESC).

Figura 16 – Exemplo do uso do comando ORDER BY ASC E DESC

```
SELECT nome, idade FROM alunos
ORDER BY nome ASC, idade DESC;
```

Em uma instrução SQL pode-se usar várias ordenações. Analisando a Figura 16 a ordem será primeiro em ordem alfabética, do A – Z (nome ASC) e depois pela idade, do mais velho para o mais novo (idade DESC).

TEMA 3 – OUTROS COMANDOS DE RESTRIÇÃO DE CONSULTA

O comando BETWEEN seleciona dados em um intervalo de valores.

Figura 17 – Exemplo do uso do comando BETWEEN

```
SELECT * FROM alunos WHERE
idade BETWEEN 20 AND 35 ;
```

cod	nome	idade	sexo	funcao
1	Augusto	30	M	pintor
2	Paula	23	F	gerente

A consulta retornou todos os registros da tabela *alunos*, em que a *idade* está entre 20 e 35 anos. Outra forma de realizar esse tipo de pesquisa é utilizar os operadores relacionais. Observe na Figura 18.

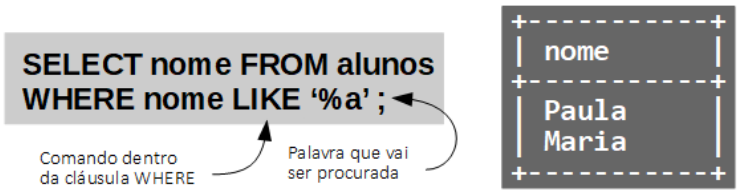
Figura 18 – Exemplo do uso do comando SELECT com operadores relacionais

```
SELECT * FROM alunos
WHERE idade >=  20 AND idade <= 35 ;
```

Na Figura 18 foram utilizados os operadores >= e <=, pois o comando BETWEEN, exemplificado na Figura 17, também interpreta que o número 20 e o número 30 fazem parte do intervalo da seleção.

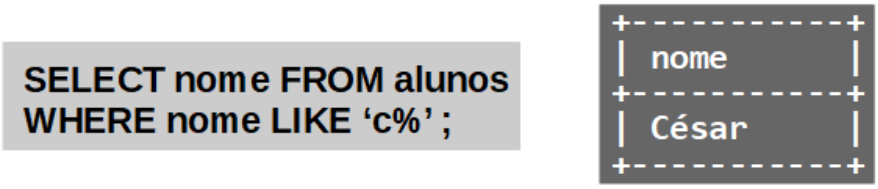
Também é possível realizar a pesquisa de dados que tenham um determinado conteúdo de texto. Para isso, utiliza-se o comando LIKE, que é utilizado nos quatros grandes grupos de comandos. Para ajudar na construção do padrão, o SQL possui dois caracteres coringas: % (porcentagem) e _ (*underline*).

Figura 19 – Exemplo do uso do comando LIKE com % no início



Com a utilização do caractere % na frente da palavra (Figura 19), a consulta retorna todos os registros que contenham o caractere "a" no final do valor do dado da coluna consultada, não importando quantos caracteres existam antecedendo.

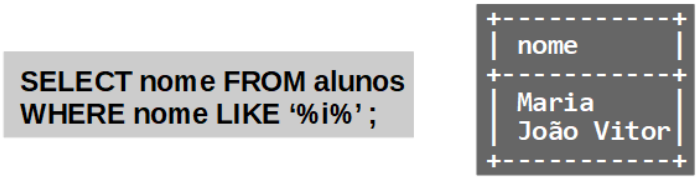
Figura 20 – Exemplo do uso do comando LIKE com % no final



O caractere % também pode ser utilizado ao final do padrão; a diferença é que a consulta retorna todos os registros que contenham o caractere "c" no início do valor do dado da coluna consultada, não importando quantos caracteres existam depois.

Outra maneira para sua utilização é colocar o caractere de % antes e depois do padrão, como na Figura 21. A consulta retornará os registros que contenham o caractere "i" em qualquer posição do dado da coluna consultada (nome).

Figura 21 – Exemplo do uso do comando LIKE com % no início e no final do padrão



O caractere _ indica o número de caracteres envolvidos na pesquisa. A Figura 22 é um exemplo de uma consulta que retorna todos os registros que contenham exatamente 5 caracteres na coluna *nome*, pois no comando LIKE foi informado 5 *underlines* (_ _ _ _ _).

Figura 22 – Exemplo do uso do comando LIKE com _

```
SELECT nome FROM alunos
WHERE nome LIKE '_ _ _ _ _';
```

```
+-----------+
| nome      |
+-----------+
| César     |
| Paula     |
| Maria     |
+-----------+
```

No exemplo da Figura 23, foram combinados os dois caracteres coringas (_ e %) e a consulta retornou os registros que tinham o caractere “a” na segunda posição do valor do dado da coluna consultada, não importando quantos caracteres existam depois.

Figura 23 – Exemplo do uso do comando LIKE com _ e %

```
SELECT nome FROM alunos
WHERE nome LIKE '_a%';
```

```
+-----------+
| nome      |
+-----------+
| Paula     |
| Maria     |
+-----------+
```

Assim como os operadores relacionais e o BETWEEN, o LIKE é mais um dos comandos que podem ser utilizados na cláusula WHERE. É importante frisar que é possível usar os comandos juntos (no mesmo WHERE).

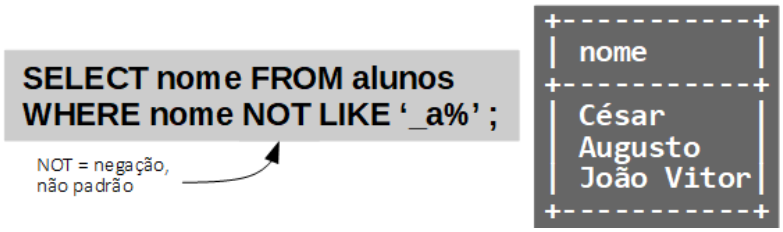
O SELECT da Figura 24 retorna o nome de todos os alunos registrados na tabela *alunos* que possuem a *idade* maior que 18 e têm o *sobrenome* iniciando com a letra F.

Figura 24 – Exemplo do uso do comando LIKE com operadores relacionais

```
SELECT nome FROM alunos
WHERE idade > 18
AND sobrenome LIKE 'f%';
```

Há ainda a combinação do operador NOT junto com o comando LIKE, formando o comando NOT LIKE. A instrução SQL vai retornar todos os registros que não fazem parte do padrão determinado. O comando SELECT executado na Figura 25 trouxe apenas os registros em que os nomes não têm o caractere “a” como segunda letra do nome.

Figura 25 – Exemplo do uso do comando NOT LIKE



Para finalizar o comando LIKE e suas combinações com os caracteres coringas, Bertol (2013) desenvolveu uma tabela com a utilização do comando LIKE. Observe a Figura 26.

Figura 26 – Explicação das expressões do comando LIKE

Expressão	Resultado
LIKE 'Juca%'	Qualquer string que inicie com Juca.
LIKE '%Silva'	Qualquer string que termine com Silva.
LIKE '%Santos%'	Qualquer string que tenha Santos em qualquer posição.
LIKE 'A_'	String de dois caracteres que tenha a primeira letra A e o segundo caractere seja qualquer outro.
LIKE '_A'	String de dois caracteres cujo primeiro caractere seja qualquer um e a última letra seja a letra A.
LIKE '_A_'	String de três caracteres cuja segunda letra seja A, independentemente do primeiro ou do último caractere.
LIKE '%A_'	Qualquer string que tenha a letra A na penúltima posição e a última seja qualquer outro caractere.
LIKE '_A%'	Qualquer string que tenha a letra A na segunda posição e o primeiro caractere seja qualquer outro caractere.
LIKE '___'	Qualquer string com exatamente três caracteres.
LIKE '___%'	Qualquer string com pelo menos três caracteres.
LIKE '%"%'	Qualquer string que tenha o caractere ' em qualquer posição.

Fonte: Bertol, 2013.

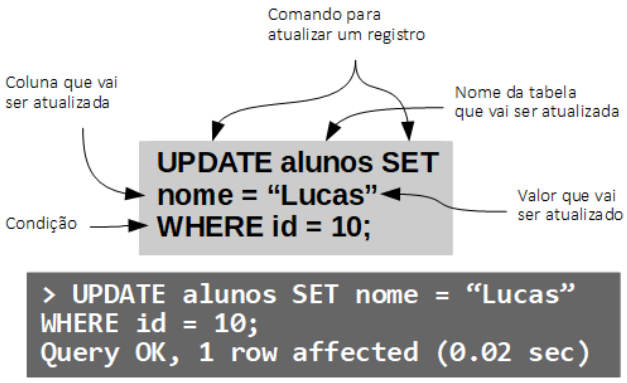
TEMA 4 – EDITAR E APAGAR REGISTROS

4.1 Editar registros

Após fazer a inserção dos dados nas tabelas, às vezes é necessário alterar ou atualizar informações. Para isso, é utilizado o comando UPDATE. A cláusula WHERE é de extrema importância, pois ela é a condição para que um registro seja alterado. Após a execução do UPDATE, não é possível desfazer as alterações.

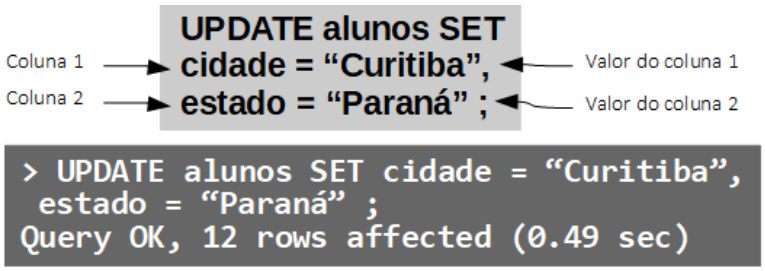
Na Figura 27, o comando está alterando o conteúdo da coluna *nome* para Lucas no registro que contém o *id* 10. Como foi especificado o *id*, a alteração na coluna *nome* se deu em apenas um registro.

Figura 27 – Exemplo do uso do comando UPDATE



O UPDATE pode atualizar uma ou várias colunas no mesmo comando, desde que as colunas sejam separadas por vírgulas. Observe a Figura 28.

Figura 28 – Exemplo do uso do comando UPDATE sem cláusula WHERE



Também é possível executar um comando UPDATE sem colocar a condição WHERE. Note que na Figura 28 foram atualizados 12 registros. Com isso, é possível concluir que a tabela *alunos* tinha 12 registros, pois o UPDATE atualizou todos eles. Na Figura 27, também foram alteradas duas colunas (cidade e estado) na mesma instrução SQL.

E quando é possível utilizar o comando UPDATE sem cláusula WHERE? Preste atenção ao exemplo: a tabela *alunos* já estava criada, e com 12 alunos registrados. Foi observado que as colunas *cidade* e *estado* seriam de extrema importância, e com o comando ALTER TABLE foram adicionadas essas duas colunas na tabela. A tabela, no entanto, já tinha 12 registros cadastrados que, após a inclusão das duas colunas, ficaram com essas colunas vazias (sem dados). É utilizado o comando UPDATE, da Figura 28, para atualizar (preencher

as colunas vazias) todos os registros que já estavam cadastrados de uma única vez.

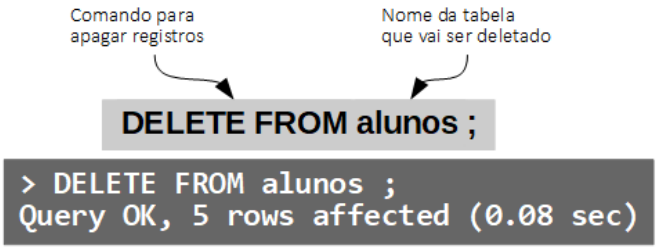
4.2 Apagar registros

Até agora, nesta aula, foram demonstrados os comandos para inserir, atualizar e consultar os dados que estão registrados em uma tabela. O último grupo de comandos é o DELETE. Sua função é apagar os registros dentro das tabelas. Antes de mostrar sua sintaxe e exemplos, algumas regras gerais do comando DELETE.

- O DELETE não pode ser utilizado para apagar um dado em uma coluna específica.
- É utilizado para apagar um ou mais registros, dependendo da cláusula WHERE.
- Pode ser utilizado para apagar todos os dados da tabela, bastando para isso não informar condição.

O comando executado na Figura 29 apaga todos os dados da tabela *alunos*. Quando for executar uma instrução DELETE, seja cuidadoso, pois os registros deletados são excluídos permanentemente, isto é, não poderão ser resgatados.

Figura 29 – Exemplo do uso do comando DELETE sem cláusula WHERE



A utilização da cláusula WHERE é essencial para direcionar o comando DELETE para um grupo de registros, ou mesmo para um único registro. Observe na Figura 29.

O comando da Figura 30 apaga os registros da tabela *alunos* que possuem o ID menor que 5 (1, 2, 3 e 4), não importando quantos registros a tabela possui.

Figura 30 – Exemplo do uso do comando DELETE com cláusula WHERE

O comando da Figura 31 apaga os registros da tabela *alunos* que possuem o valor da coluna *id* menor que 5 e têm o conteúdo da coluna *situacao* igual a “D”. Como foi utilizado o operador AND, só vão ser apagados os registros que tiverem as duas condições ao mesmo tempo.

Figura 31 – Exemplo do uso do comando DELETE com operador relacional AND

```
DELETE FROM alunos WHERE id < 5 AND situacao = 'D';
```

Também é possível utilizar o operador OR como exemplificado na Figura 32. Após a execução do comando, serão apagados todos os registros que possuem *idade* menor que 18 e os registros que possuem *idade* maior que 60. Note que o mesmo aluno não poderá ter as duas idades ao mesmo tempo, mas como foi utilizado o conector OR, os registros apagados devem ter, no mínimo, uma das duas condições verdadeiras.

Figura 32 – Exemplo do uso do comando DELETE com operador relacional OR

```
DELETE FROM alunos WHERE idade < 18 OR idade > 60 ;
```

FINALIZANDO

Nesta aula foi apresentada a categoria DML, que possui quatro comandos principais: INSERT, SELECT, UPDATE e DELETE, e foram explanados os comandos dessa categoria. Foi feita a inserção de dados nas tabelas e depois feita a consulta aos dados já inseridos. O comando SELECT é um dos mais importantes, e é o que possui mais comandos – por isso foi dado mais atenção a ele. Depois foi mostrado o comando UPDATE, para fazer as alterações nos registros das tabelas e, por último, o comando DELETE foi utilizado para apagar os registros inseridos.

REFERÊNCIAS

ALVES, W. P. **Banco de dados**. São Paulo: Érica, 2014.

BERTOL, O. F. Operador LIKE em consultas SQL. **Dev Media**, 2013. Disponível em: <https://www.devmedia.com.br/operador-like-em-consultas-sql-no-delphi/26995>. Acesso em: 6 set. 2019.