

# BANCO DE DADOS

Linguagem SQL – DML

# LINGUAGEM SQL - DML

Estudaremos como manipular dados em um Banco de Dados, ou seja, como realizar buscas, inserir dados, atualizar dados, apagar dados.

A instrução utilizada para consultas no banco de dados é a **SELECT**, para realização de inserções de novos dados utilizamos o **INSERT**, para remoção de linhas contendo dados utilizamos o comando **DELETE**, e o **UPDATE**, para atualizações nos dados.

- **INSERT** - INSERÇÃO
- **UPDATE** - ATUALIZAÇÃO
- **DELETE** - APAGAR
- **SELECT** - SELECIONAR

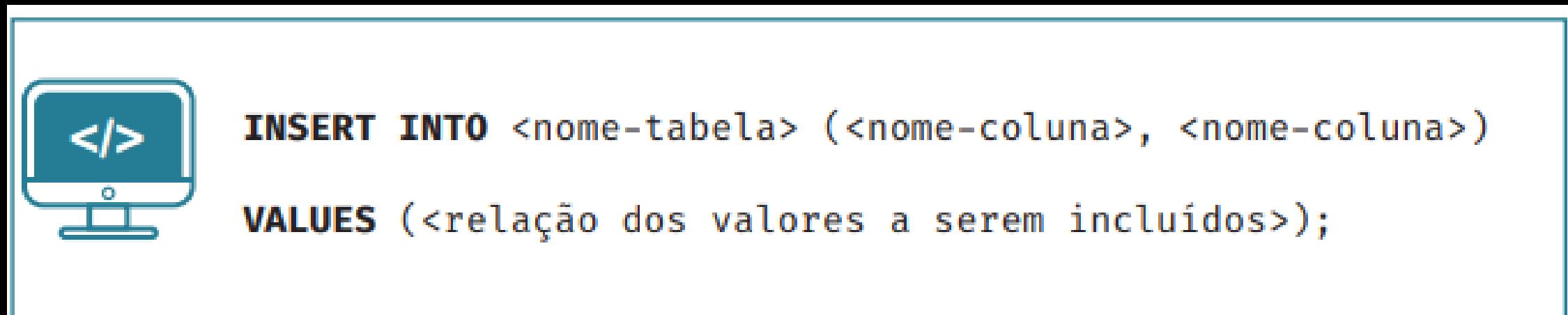
02

# COMANDO INSERT

Vamos começar com o comando **INSERT**, usado para adicionar dados nas tabelas do Banco de Dados. Com este comando vamos inserir dados em nossas tabelas.

Função:

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



03

# COMANDO INSERT

Onde cada campo representa:

<nome-tabela> o nome da tabela que será incluída como um novo registro.

<nome-coluna> o nome de uma ou mais colunas que terão o conteúdo no momento da operação de inclusão.

Podemos omitir os nomes das colunas, se a ordem dos valores que iremos inserir corresponder a mesma ordem em que foram criados. Caso contrário, é necessário informar todos os campos da tabela.

04

# COMANDO INSERT

Para facilitar o entendimento da linguagem vejamos como exemplo adicionar os dados de um Funcionário:

Funcionário

Código: 01

Nome: João

Endereço: Rua Dom Bosco

05

# COMANDO INSERT

Para inserir os dados deste novo funcionário no sistema de banco de dados da empresa utilizariamos o seguinte comando:



```
INSERT INTO FUNCIONARIO  
  (CODIGO_func, NOME_func, ENDEREÇO)  
VALUES (1, "João", "Rua Dom Bosco");
```

# COMANDO INSERT

Seguindo o exemplo do modelo relacional criado na anteriormente podemos inserir os dados de uma banda e logo em seguida de um músico:



```
INSERT INTO BANDA (codigo_banda, nome_banda)  
VALUES (1,"FOREVER");
```

# COMANDO INSERT

Código Musico: 1

Código Banda: 1

Nome Musico: Ananias

Tempo Experiência: 3 anos

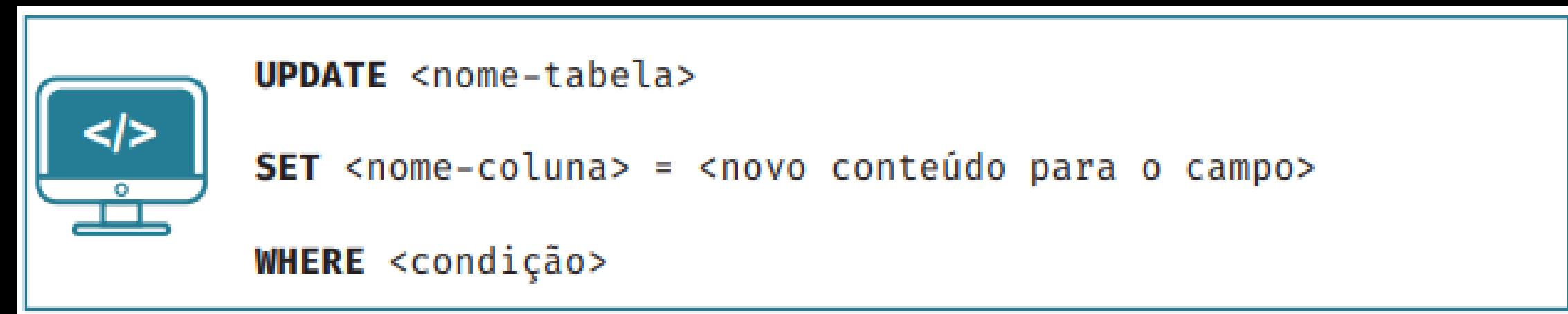


```
insert into MUSICO (codigo_musico, Código_bandaFK,  
                    nome_musico, tempo_experiencia)  
values (1,1,"Ananias", "3 anos");
```

08

# COMANDO UPDATE

Após inserir os dados no banco, pode ser necessário realizar atualizações, por exemplo: um funcionário trocou de endereço, ou está trabalhando em outro projeto. Para estes casos, utilizamos o comando de atualização de Banco UPDATE.



# COMANDO UPDATE

Cada campo representa:

<nome-tabela>- o nome da tabela que terá seus dados alterados.

<nome-coluna>- o nome de uma ou mais colunas que terão o conteúdo alterado.

<condições>- a condição para a seleção dos registros que serão atualizados.

Podemos indicar uma condição para que um ou vários registros possam ser alterados, quando não se especifica, todos os dados são alterados.

# COMANDO UPDATE

Atualize o endereço do funcionário João



```
UPDATE Funcionario  
SET endereço = "Rua Pedro de Araújo"  
WHERE nome_func = "João";
```

# COMANDO UPDATE

Caso a empresa queira atualizar o salário de todos os funcionários do projeto GEOCIÊNCIAS em 10%. Utilizariamos o seguinte código:



```
UPDATE Funcionario  
SET SALARIO = SALARIO * 1.1  
WHERE nome_projeto = "GEOCIENCIAS";
```

# COMANDO UPDATE

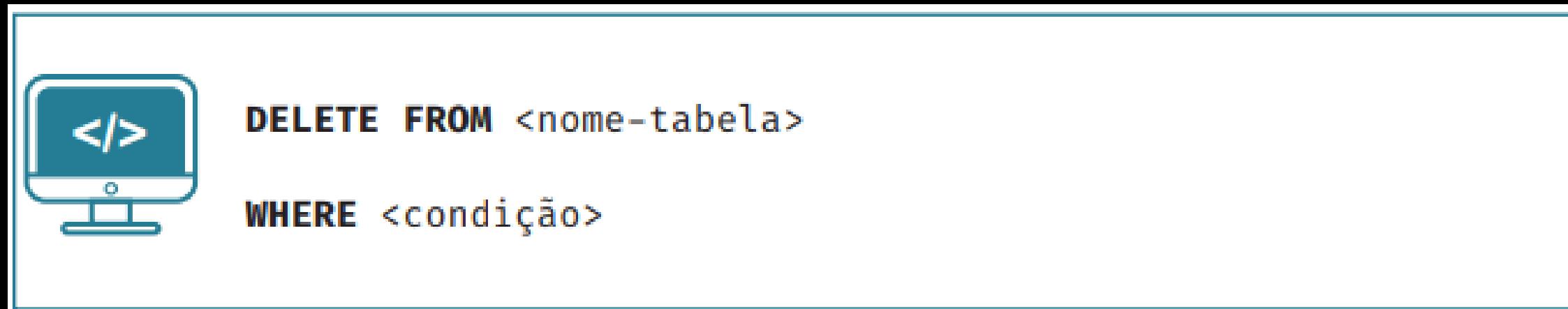
Atualize o tempo de experiência do músico Ananias para 4 anos:



```
UPDATE musico  
SET tempo_experiencia = "4 anos"  
WHERE nome_musico = "Ananias";
```

# COMANDO DELETE

Quando utilizamos a requisição para remover um dado do banco utilizamos a instrução DELETE. Essa requisição é descrita da mesma forma de uma consulta vejamos a sintaxe:



<nome-tabela> - o nome da tabela que terá seus dados deletados.

<condição> - a condição para apagar os dados da tabela, que pode ser um ou vários registros.

# COMANDO DELETE

Como exemplo, imagine que a mesma empresa que contratou o João para execução de um projeto, ao finalizar este, não necessitou mais dos serviços de João. Dessa forma os dados de João devem ser excluídos do Banco de Dados.



```
DELETE FROM funcionario  
WHERE nome_func = "João" AND codigo_func=1;
```

# COMANDO DELETE

Outro exemplo seria o músico Ananias decidiu sair da Banda FOREVER, pois irá seguir carreira solo. Os dados dele devem ser retirados do banco de dados da seguinte forma:



```
DELETE FROM musico  
WHERE nome_musico = "Ananias" AND codigo_bandaFK=1;
```

# COMANDO DELETE

Utilizamos o operador “AND” para selecionar o funcionário João, pois é necessário informar também qual é o código único deste funcionário, já que no banco de dados podemos ter vários funcionários com o mesmo nome, porém somente um único funcionário terá o código 1 (chave primária); o mesmo caso se aplica para o músico, pois podemos ter vários músicos com o nome Ananias, porém queremos retirar apenas aquele que está na Banda Forever que possui código 1.

# COMANDO CREATE

A linguagem SQL oferece os seguintes operadores lógicos:

CONECTORES LÓGICOS	
Utilizados para enumerar duas ou mais condições na condição	
OPERADOR	SIGNIFICADO
AND	“E” LÓGICO
OR	“OU” LÓGICO
NOT	NEGAÇÃO

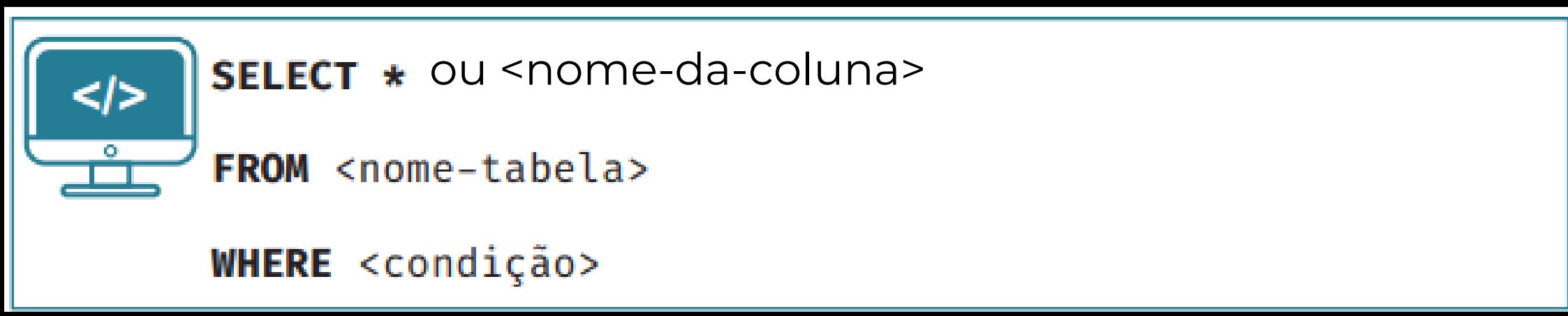
# COMANDO SELECT

Vimos como inserir, como atualizar e apagar dados do banco de dados. Agora veremos a clausula mais utilizada em um sistema de banco de dados: **a consulta!**

Sistemas de empresas, de banco e sites de modo geral necessitam realizar buscas no banco de dados de forma periódica, portanto é muito importante entender como realizar buscas em um banco de dados.

# COMANDO CREATE

A estrutura básica do comando de busca de banco de dados SELECT consiste em três cláusulas.



# COMANDO SELECT

Em que cada um dos campos corresponde:

- SELECT: seleciona os atributos desejados na consulta;
- < \* > : significa que todos os atributos da tabela selecionada serão retornados; Caso seja especificado aqui os nomes dos atributos ou expressões ou constantes ou funções deve ser conectadas por operadores aritméticos e parênteses.
- < nome-tabela>- o nome da(s) tabela(s) que possui as colunas que serão selecionadas na consulta.
- < condições>- condição para selecionar os dados, podendo retornar um ou vários dados.
- < nome-coluna> - caso não seja utilizado o <\*> devemos especificar os nomes das colunas que queremos retornar os dados.

# COMANDO SELECT

Quando trabalhamos com consultas, é normal que seja utilizado expressões para retornar os dados solicitados pelo usuário. Temos os seguintes operadores para relacionar duas expressões em SQL

Operador	Significado
=	Igual
!=	Diferente
<>	Diferente
>	Maior
!>	Não maior (menor ou igual)
<	Menor
!<	Não menor (maior ou igual)
>=	Maior ou igual
<=	Menor ou igual

# COMANDO SELECT

Veja a seguir como realizar uma consulta para obter os nomes dos funcionários utilizando para isso a tabela FUNCIONARIO:



```
SELECT nome_func  
FROM FUNCIONARIO;
```

# COMANDO SELECT

Quando precisamos recuperar os dados de mais de uma coluna, devemos simplesmente declarar os nomes das colunas separados por vírgula. Vejamos a seguir, por exemplo, como obter os nomes e endereço de todos os funcionários:



```
SELECT nome_func, endereço  
FROM FUNCIONARIO;
```

# COMANDO SELECT

Caso, queiramos recuperar apenas os funcionários cujo salário é igual a R\$1000,00, utilizariamos o seguinte comando:



```
SELECT *
FROM FUNCIONARIO
WHERE SALARIO = 1000;
```

Observe que neste caso, apenas faríamos uma consulta e verificaríamos quais são os funcionários com o salario igual a R\$1000,00.

# COMANDO SELECT

Podemos utilizar a cláusula WHERE de diferentes formas, e colocar vários tipos de sentenças que irão depender dos critérios utilizados para busca e podem ser utilizados quaisquer operadores lógicos ou aritméticos.

Para a WHERE, têm-se outras cláusulas específicas que podem ser utilizadas para aprimorar as buscas. São elas:

- DISTINCT <nome-da-coluna>
- GROUP BY <nome-da-coluna>
- HAVING <condição>
- ORDER BY <nome-do-campo> (ASC/DESC)

# COMANDO SELECT

Veremos agora exemplos de cada uma dessas cláusulas, utilizando a tabela Funcionário. Vamos buscar em nosso banco de dados todos os nomes de projetos sem repetições. Para isso utilizariamos o seguinte comando:



```
SELECT DISTINCT nome_projeto  
FROM FUNCIONARIO;
```

# COMANDO SELECT

Agora imagina que no sistema da Empresa seja necessária a realização de uma agenda de funcionários em ordem alfabética, dessa forma, utilizariamos o seguinte comando:



```
SELECT nome_func  
FROM FUNCIONARIO  
ORDER BY nome_func ASC ;
```

# COMANDO SELECT

Da mesma forma em que é possível ordenar os nomes dos funcionários em ordem alfabética também é possível ordenar os nomes dos projetos em ordem decrescente:



```
SELECT nome_proj  
FROM FUNCIONARIO  
ORDER BY nome_proj DESC ;
```

# COMANDO SELECT

Outra cláusula comumente usada em consultas de banco de dados é o agrupamento de dados, onde podemos verificar e agrupar, por exemplo, os funcionários que estão trabalhando com uma carga horária superior a 30 horas semanais:



```
SELECT nome_func  
FROM FUNCIONARIO  
GROUP BY carga_horaria  
HAVING carga_horaria>'30 horas';
```