Correção da avaliação, Funções SQL

Prof. Sérgio Luiz Rodrigues

A instrução MySQL INSERT INTO

- A **INSERT INTO** instrução é usada para inserir novos registros em uma tabela. Colocando os valores entre ‘ ’ se não for numérico.
- Sintaxe
  - **INSERT INTO** *table_name*
  - (*column1*, *column2*, *column3*, ...) **VALUES**
  - (*value1*, *value2*, *value3*, ...);
- Outra maneira, mas mantendo a ordem:
  - INSERT INTO *table_name* VALUES (*value1*, *value2*, *value3*, ...);

A instrução MySQL SELECT

- A instrução **SELECT** é usada para **selecionar** dados de um banco de dados.
- Os dados retornados são armazenados em uma tabela de resultados, chamada conjunto de resultados.
  - **Sintaxe SELECT:**
  - SELECT column1, column2, …
  - FROM table_name;
- Aqui, column1, column2, … são os nomes dos campos da tabela da qual você quer selecionar dados.

Operadores na cláusula WHERE

Operador	Descrição	Exemplo
=	Igual	SELECT * FROM produtos WHERE preco = 18;
>	Maior que	SELECT * FROM produtos WHERE preco > 18;
<	Menor que	SELECT * FROM produtos WHERE preco < 18;
>=	Maior ou igual	SELECT * FROM produtos WHERE preco > =18;
<=	Menor ou igual	SELECT * FROM produtos WHERE preco < =18;
<>	Diferente.	SELECT * FROM produtos WHERE preco < >18;
BETWEEN	Entre um certo intervalo	SELECT * FROM produtos WHERE preco BETWEEN 50 AND 60;
LIKE	Procurar um padrão	SELECT * FROM clientes WHERE cidade LIKE 's%';
IN	múltiplos valores possíveis para uma coluna	SELECT * FROM clientes WHERE cidade IN ('Paris','London');

Operadores AND, OR e NOT do MySQL

- A **WHERE** cláusula pode ser combinada com os operadores **AND**, **OR** e **NOT**.
- Os operadores **AND** e **OR** são usados para filtrar registros com base em mais de uma condição:
- O operador **AND** exibe um registro se todas as condições separadas por forem VERDADEIRAS.
- O **OR** operador exibe um registro se qualquer uma das condições separadas por **OR** for **VERDADEIRA**.
- O **NOT** operador exibe um registro se a(s) condição(ões) **NÃO FOR VERDADEIRA**(S).
- Sintaxe **AND**
  - **SELECT** column1, column2, …
  - **FROM** table_name
  - **WHERE** condition1 **AND** condition2 **AND** condition3 …;

O operador LIKE

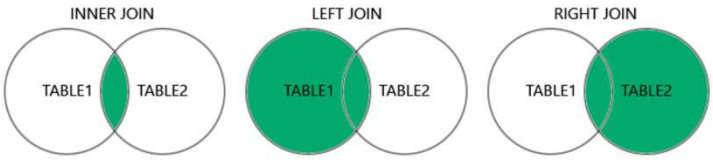
- O **LIKE** é usado em uma cláusula **WHERE** para procurar um padrão especificado em uma coluna.
- Existem dois curingas frequentemente usados em conjunto com o operador **LIKE** :O sinal de porcentagem **(%)** representa zero, um ou vários caracteres
- O sinal de sublinhado **(_)** representa um único caractere
- O sinal de porcentagem e o sublinhado também podem ser usados em combinações!
- Exemplo
- **SELECT * FROM** clientes **where** nome like 'a%';

A palavra-chave MySQL ORDER BY

- A **ORDER BY** palavra-chave é usada para classificar o conjunto de resultados em ordem crescente ou decrescente.
- A **ORDER BY** palavra-chave classifica os registros em ordem crescente por padrão.
- Para classificar os registros em ordem decrescente, use a DESC palavra-chave.
- Sintaxe **ORDER BY:**
  - **SELECT** *column1*, *column2, …*
    FROM *table_name*
    ORDER BY *column1, column2, …* ASC | DESC;

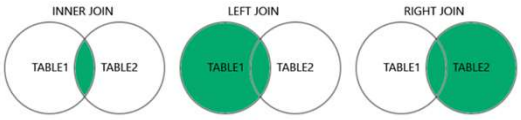
JOINs

- Nas aulas anteriores vimos que as tabelas podem ser relacionadas através de uma coluna através da **chave Estrangeira**.
- **Mas pra que servem essas relações?**
- Com essas relações, conseguimos utilizar informações de uma tabela em outra tabela. Será muito útil, por exemplo, pra gente descobrir o nome do produto vendido (na tabela de Pedidos) fazendo essa busca lá na tabela de Produtos.
- Essas relações serão criadas por meio do que chamamos de **JOIN's**.
- A tradução literal dessa palavra é "juntar", "unir". Os **JOINs** vão nos permitir fazer exatamente isso: juntar as nossas tabelas, de forma a complementar as informações umas das outras



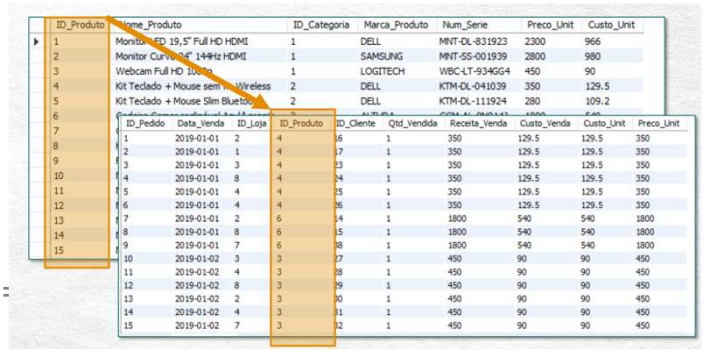
JOINs

- **INNER JOIN**: Retorna registros que possuem valores correspondentes em **ambas** as tabelas
- **LEFT JOIN**: Retorna todos os registros da tabela da **esquerda** e os registros correspondentes da tabela da direita
- **RIGHT JOIN**: Retorna todos os registros da tabela da **direita** e os registros correspondentes da tabela da esquerda



INNER JOIN: Estrutura

- **SELECT**
- Tabela_A.coluna1,
  Tabela_A.coluna2,
  Tabela_A.coluna3,
  Tabela_B.coluna4
- **FROM**
- Tabela_A
- **INNER** JOIN Tabela_B
- **ON**
  Tabela_A.id_chave_estrangeira =
  Tabela_B.id_chave_primaria



Funções SQL

Prof. Sérgio Luiz Rodrigues

Funções SQL

- As funções de SQL existem para facilitar a manipulação dos dados armazenados
- São ferramentas projetadas para uma tarefa única e bem definida.
- As funções são chamadas dentro de uma consulta SQL pelo seu nome:
- # Algumas recebem argumento, outras não.
- # Todas elas retornam um valor.

Funções SQL

Categoria de Função	Descrição
Matemáticas	Funções usadas para realizar cálculos matemáticos específicos, como as funções trigonométricas e outras.
Cadeia de Caracteres	Funções que realizam manipulação de cadeia de caracteres, tais como: localizar padrões dentro de cadeias de caracteres, inserir caracteres, concatenar cadeias de caracteres, descobrir o comprimento, converter para maiúsculo e minúsculo.
Data/Hora	Funções utilizadas para retornar informações sobre data/hora correntes, formatar data e hora como cadeia de caracteres, realizar cálculos baseado em horários, etc.
Agregação	Funções utilizadas para fazer agrupamento.
Formatação	Funções utilizadas para retornar informações formatadas.

Comandos Avançados: Funções Matemáticas - **ABS**

- No MySQL, existem várias **funções matemáticas** que você pode usar para realizar cálculos e operações em seus dados, vamos ver as mais comuns:
- Função **ABS()**, que é utilizada para retornar o valor absoluto (**positivo**) de um número.
- Exemplo:
- SELECT ABS(-10), ABS(0), ABS(10);

Funções Matemáticas - ROUND

- **ROUND**() no MySQL é utilizada para arredondar um número para um número especificado de casas decimais.
- Se nenhum número específico de casas decimais for fornecido, ela arredondará o número para o inteiro mais próximo.
- Vamos entender melhor como funciona:
- Sintaxe da função ROUND():ROUND(valor, casas_decimais)
- Ex:

```
SELECT  ROUND(3.14159, 2) AS arredondamento,
ROUND(52.36956) AS Arredondado,
ROUND(52.36956, 3) AS Arredondado2;
```

Funções Matemáticas - CEILING

- A função **CEILING**() no MySQL é usada para arredondar valores para cima, ou seja, para o menor inteiro que é maior ou igual a um número especificado.
- **Sintaxe**:
- A sintaxe básica da função CEILING() é a seguinte: **CEILING**(**numero**)
- Onde: numero: O valor decimal que você deseja arredondar para cima.
- Exemplo:

```
select
CEILING(4.3) AS nota01,
ceiling(4.7) AS nota02;
```

Funções Matemáticas - FLOOR

- A função **FLOOR**() no MySQL é muito útil para arredondar valores para baixo, ou seja, obter o maior número inteiro que é menor ou igual a um número decimal.
- **Sintaxe**:
- A sintaxe básica da função FLOOR() é a seguinte:
- **FLOOR**(**numero**)
- Onde: numero: O valor decimal que você deseja arredondar para baixo
- Exemplo:

SELECT
FLOOR(4.7) AS nota01,
floor(4.2) as nota02;

Funções Matemáticas – POWER ou Pow

- A função **POWER**() no MySQL é usada para encontrar o valor de um **número elevado à potência** de outro número.
- Ela retorna o resultado de elevar o número base (X) à potência do expoente (Y).
- Em outras palavras, é uma maneira de calcular **X^Y**.
- **Sintaxe**: A sintaxe básica da função POWER() é a seguinte:
- POWER(X, Y) Onde: X: Especifica o número base. Y: Especifica o número do expoente.
- Exemplo:

```
select
    POWER(2, 3) AS potencia1,
    power(5,2) as potencia2,
    pow(5,2) as potencia3,
    power(5, 5) as potencia4;
```

Funções Matemáticas - SQRT

- A função **SQRT**() no MySQL é usada para calcular a raiz quadrada de um número.
- Ela retorna o resultado da operação de raiz quadrada, ou seja, o valor positivo cujo quadrado é igual ao número especificado.
- **Sintaxe**: A sintaxe básica da função SQRT() é a seguinte: **SQRT**(**numero**)
- Onde: numero: O valor do qual você deseja calcular a raiz quadrada. Deve ser um número não negativo

```
select
    SQRT(16) AS raiz_quadrada;
```

Funções Matemáticas - HEX()

- **HEX():** Converte um valor **decimal** para sua representação **hexadecimal**.
- Por exemplo, **HEX**(255) resulta em 'FF'.

```
select
    hex(1),
    hex(10),
    hex(11),
    hex(12),
    hex(13),
    hex(14),
    hex(15),
    hex(255);
```

operações matemáticas

```
select 10+20 as soma,
10-20 as subtração,
10*20 as multiplicação,
10/20 as divisão,
(2+3)*5 as combinação,
22 % 5 as resto_da_divisão;
```

Variáveis

- As variáveis no MySQL são ferramentas poderosas que permitem armazenar valores temporariamente e reutilizá-los em consultas ou procedimentos. Elas facilitam a organização do código, melhoram a manutenção e podem ser usadas para cálculos, atribuição de valores e muito mais.
- Declarando Variáveis: Para declarar uma variável no MySQL, utilizamos a sintaxe: **SET @nome_variavel** = **valor**;

```
-- variáveis
SET @varquantidade =10;
SET @varpreco = 10.90;
SET @varReceita = @varquantidade * @varpreco;

select @varReceita as Receita_total;
```

Variáveis

- Exemplo 2:

```
SET @nota1 = 10;
SET @nota2 = 05;
SET @nota3 = 06;
SET @nota4 = 07;
SET @total = @nota1 + @nota2 + @nota3 + @nota4;

select @total as TOTAL_aluno;
```

Funções de Data e Hora

- As funções de **data e hora** no MySQL são recursos poderosos que permitem a manipulação e formatação de valores relacionados a datas e horas em bancos de dados.
- Elas facilitam cálculos, extrações de componentes de data, formatação para exibição e muito mais.
- Vou apresentar algumas das principais funções, juntamente com exemplos

Funções de Data e Hora - NOW()

- **NOW**():-- Retorna a data e hora atuais do sistema.–
- Exemplo:

SELECT NOW() as AGORA;

Funções de Data e Hora - CURRENT_DATE() ou CURDATE

- -- **CURRENT_DATE():**
- É um sinônimo de **CURDATE**(), ou seja, tem a mesma função.

```
SELECT
    CURRENT_DATE AS data_atual,
    CURRENT_TIME AS hora_atual,
    CURRENT_TIMESTAMP AS data_hora_atual;
```

Funções de Data e Hora - CURRENT_DATE() ou CURDATE

- Para que serve a **CURDATE**()?
- **Comparar datas**: Você pode usar CURDATE() para comparar datas em suas tabelas

```
-- exemplo - encontrar os clientes com mais de 18 anos
SELECT * FROM funcionarios WHERE DATEDIFF(CURDATE(), data_nasc) > 2006-10-06
order by data_nasc desc;
```

Funções de Data e Hora - EXTRACT

- -- **EXTRACT**() - Extrai componentes específicos de uma data (como dia, mês ou ano).

```
SELECT
    EXTRACT(YEAR FROM CURRENT_DATE) AS ano_atual;
```

Funções de Data e Hora - DATEDIFF()

- -- **DATEDIFF**():
- Calcula a diferença entre duas datas (em dias, meses ou anos).–
- Exemplo
- (calculando a diferença em dias entre duas datas):

```
SELECT DATEDIFF('2024-10-06', '2024-09-20') AS DIFEREÇA_DATA;
```

```
-- exemplo 2
select datediff('2024-10-06', '1967-01-12') as idade;
```

Funções de Data e Hora - TIMESTAMPDIFF

- A função **TIMESTAMPDIFF**() permite calcular a diferença entre duas datas em termos de anos, meses, dias, horas, etc.
- Para calcular a idade em anos, usaremos o seguinte exemplo:

```
SELECT TIMESTAMPDIFF(YEAR, '1967-01-12', CURDATE()) AS Idade;
```

- Nesse exemplo: '1967-01-12' é a data de nascimento da pessoa. **CURDATE**() retorna a data atual. **YEAR** especifica que queremos a diferença em anos.

Funções de String (cadeia de caracteres)

- As **funções de string** no MySQL são ferramentas essenciais para manipular dados de texto.
- Elas permitem realizar diversas operações em strings, como concatenação, conversão de tipos de dados, busca e substituição de substrings, entre outras.
- Vamos apresentar algumas das principais funções de string no MySQL:

Funções de String - Concat

- -- **Concat**
- A função **CONCAT**() no MySQL é usada para concatenar (ou seja, juntar) múltiplas strings em uma única string.
- Ela é especialmente útil quando você precisa combinar informações de diferentes colunas ou valores em uma consulta.
- Sintaxe: A sintaxe básica da função CONCAT() é a seguinte: CONCAT(string1, string2, string3,

```
select
    CONCAT('SQL', ' ', 'is', ' ', 'fun') AS concatenacao;
```

Funções de String - Concat

- -- **outro exemplo**, usando tabelas do emporio_turquinho
- use emporio_turquinho;

```
SELECT
concat (nome, ' ', sobrenome) as 'nome_completo'
from funcionarios
order by nome;
```

Funções de String - Trim

- A função **TRIM**() no MySQL é usada para remover espaços em branco de uma string.
- Ela é especialmente útil quando você precisa limpar dados ou eliminar caracteres indesejados no início ou no final de uma cadeia de caracteres.
- Sintaxe: A sintaxe básica da função TRIM() é a seguinte:
- **TRIM**(string) Onde: string: A string da qual você deseja remover espaços em branco.

```
select
    TRIM('  espacos  ') AS sem_espacos;
```

Funções de String - ROUND

- **ROUND**() no MySQL é utilizada para arredondar um número para um número especificado de casas decimais.
- Se nenhum número específico de casas decimais for fornecido, ela arredondará o número para o inteiro mais próximo.
- Sintaxe da função ROUND():
- **ROUND**(valor, casas_decimais)

```
SELECT
    ROUND(3.14159, 2) AS arredondamento,
    ROUND(52.36956) AS Arredondado,
    ROUND(52.36956, 3) AS Arredondado2;
```

funções de agregação

- As funções de agregação no MySQL são ferramentas essenciais para resumir e processar dados em uma única coluna de uma tabela, retornando um único valor como resultado.
- Elas nos permitem obter informações importantes, como o valor máximo, mínimo, média, soma ou contar a quantidade total de itens.
- Vamos apresentar algumas das principais funções de agregação:

funções de **agregação**

- **COUNT**: A COUNT() função retorna o número de linhas que correspondem a um critério especificado.
- **Sintaxe:** SELECT COUNT(column_name) FROM table_name WHERE condition

```
SELECT COUNT(*) AS 'Quantidade de Registros da Tabela Cliente'
 FROM CLIENTES;

select count(preco) as 'qde de produtos com preco =18'
from produtos
where preco = 18;

select count(preco) as 'qde de produtos com preco>18'
from produtos
where preco > 18;
```

funções de **agregação**

- **MAX**: Retorna o maior valor de um conjunto de valores
- **MIN**: Retorna o menor valor de um conjunto de valores
- **SUM**: Calcula a soma dos valores em uma coluna
- **AVG**: Calcula a média aritmética dos valores em uma coluna.

```
SELECT AVG(preco) as 'valor médio dos produtos'
FROM produtos;

SELECT max(preco) as 'maior valor dos produtos'
FROM produtos;

SELECT min(preco) as 'menor valor dos produtos'
from produtos;

select sum(preco) as 'soma dos precos'
from produtos;
```

funções de formatação

- Existem várias **funções de formatação** que você pode usar para manipular e apresentar dados de maneira mais legível.
- Vou apresentar algumas delas:

1. **DATE_FORMAT()**: Essa função é usada para formatar datas de acordo com um padrão específico. Você pode escolher o formato desejado para exibir datas.
2. Por exemplo:

```
SELECT DATE_FORMAT('2023-10-07', '%d/%m/%Y') AS DataFormatada;
-- Resultado: "07/10/2023"
```