Banco de Dados Avançados

Consultas avançadas em Banco de Dados

Aula 10

Profa. Ma. Fabiana A. Rodrigues

EDUCAÇÃO METODISTA

Funções

UCASE() converte o texto em maiúscula.

Exemplo de código:

```SQL
SELECT UCASE('texto em minúsculas');
```

Saída:

```
TEXTO EM MAIÚSCULAS
```

A função **UCASE()** é idêntica à função **UPPER()**.

LCASE() converte o texto em minúscula.

Exemplo de código:

```SQL
SELECT LCASE('ALÔ');
```

Saída:

```
alô
```

A função **LOWER()** é idêntica à função **LCASE()**.

Exemplo de como a função LCASE() pode ser usada em uma consulta:

```SQL
SELECT
  LCASE(nome) AS nome_minúsculo
FROM
  clientes;
```

EDUCAÇÃO METODISTA

Funções

NOW() retorna a data e hora atual. Exemplo de código:

```sql
SELECT NOW();
```

Saída:

```
2023-10-14 14:15:20
```

CURRENT_USER() retorna o nome do usuário que está executando a consulta. Exemplo de código:

```sql
SELECT CURRENT_USER();
```

Saída:

```
[nome do usuário]
```

CURDATE() retorna a data corrente. Exemplo de código:

```sql
SELECT CURDATE();
```

Saída:

```
2023-10-14
```

DATE_FORMAT() formata uma data ou hora em um formato especificado. Exemplo de código:

```sql
SELECT DATE_FORMAT(CURDATE(), 'dd/mm/yyyy');
```

Saída:

```
14/10/2023
```

CURTIME() retorna a hora corrente. Exemplo de código:

```sql
SELECT CURTIME();
```

Saída:

```
14:15:20
```

TIME_FORMAT() formata uma hora em um formato especificado. Exemplo de código:

```sql
SELECT TIME_FORMAT(CURTIME(), 'hh:mm:ss');
```

Saída:

```
14:15:20
```

Funções

YEAR() extrai o ano de uma data.
Exemplo de código:

SQL

```
SELECT YEAR(CURDATE());
```

Saída:

```
2023
```

MONTH() extrai o mês de uma data.
Exemplo de código:

SQL

```
SELECT MONTH(CURDATE());
```

Saída:

```
10
```

DAY() extrai o dia de uma data. Exemplo de código:

SQL

```
SELECT DAY(CURDATE());
```

Saída:

```
14
```

HOUR() extrai a hora de uma hora.

MINUTE() extrai os minutos de uma hora.

SECOND() extrai os segundos de uma hora.

EDUCAÇÃO METODISTA

Funções

Exemplo:

```
SELECT CURDATE() AS "Data Atual", CURTIME() AS "Hora atual", NOW() AS "Este momento";
```

```
SELECT id_cliente, nome, sobrenome, fone, documento, valor_receber, CURDATE() AS "Data Atual", CURTIME() AS "Hora atual", NOW() AS "Este momento"
FROM clientes
RIGHT JOIN contas_receber
ON clientes.id_cliente = contas_receber.id_receber;
```

EDUCAÇÃO METODISTA

Funções

PERIOD_DIF()

A função **PERIOD_DIF()** em SQL é usada para calcular a diferença entre dois períodos. Os períodos são representados em formato YYMM ou YYYYMM.
A sintaxe da função **PERIOD_DIF()** é a seguinte:

```
SQL

PERIOD_DIF(period1, period2)
```

Onde:

- **period1:** É o primeiro período.
- **period2:** É o segundo período.

O resultado da função PERIOD_DIF() é um número inteiro que representa a diferença entre os dois períodos em meses. O valor pode ser positivo ou negativo. Por exemplo, a seguinte consulta retornará a diferença entre os períodos 202301 e 202303:

```
SQL

SELECT PERIOD_DIF(202301, 202303);
```

Saída:

```
2
```

EDUCAÇÃO METODISTA

Funções

Uma função é muito parecida com uma *stored procedure*. **A principal diferença está no fato que uma função retornar um valor e a *stored procedure* não.** Outra diferença é a maneira como uma função pode ser chamada. Por retornar um valor, ela pode ser chamada através de um comando SELECT e também usada em cálculos como outra função qualquer.

Para saber todas as informações sobre as funções do SGBD, podemos utilizar o comando ***show function status*** e ***show procedure status,*** para os procedimentos.

Funções

- **OR REPLACE** – Essa opção recria a função mantendo os privilégios previamente concedidos.
- **Lista de parâmetros** – Se mais um parâmetro for usado pela função devem ser separados por vírgula. Um parâmetro deve ser definido com a cláusula IN e inicializado como uma variável.
- **Declarações** – nessa seção são declaradas constantes variáveis e até mesmo outras procedures e funções locais.
- **Comandos** – nessa seção são colocados os comandos que serão executados pela procedure.
- **Valor_da_Função** – É uma constante ou variável que contém o valor retornado pela função.

```
CREATE [OR REPLACE] FUNCTION <nome_da_function>
                [ ( lista de parâmetros ) ]
                              RETURN Tipo_de_Dado IS
   [ declarações ]
BEGIN
   Comandos
   RETURN Valor_da_Função;
END [nome_da_function];
```

EDUCAÇÃO METODISTA

Funções

(exemplo 01)

Criando a função:

```
DELIMITER $
 CREATE FUNCTION func() RETURNS CHAR(100)
 BEGIN
  RETURN "ESSA É A MINHA PRIMEIRA FUNÇÃO";
 END
 $
```

Chamando a função:

```
SELECT func();
```

EDUCAÇÃO METODISTA

Funções

(exemplo 02)

Criando a função:

```
CREATE FUNCTION fn_teste (a DECIMAL(10,2), b INT)
RETURNS INT
RETURN a * b;
```

Invocando a função:

```
SELECT fn_teste(2.5, 4) AS Resultado;
```

Funções

(excluindo uma função)

Para excluir uma função usamos o commando **DROP FUNCTION**, de acordo com a sintaxe a seguir:

Sintaxe

```
DROP FUNCTION nome-da-função [IF EXISTS] nome_warning
```

EDUCAÇÃO METODISTA

Funções

Operador Like

Operador LIKE é utilizado para buscar por uma determinada *string* dentro de um campo com valores textuais.

Texto: Nesse caso, serão retornados todos os registros que contêm no campo buscado exatamente o "texto" informado no filtro. O funcionamento aqui é equivalente a utilizar o operador de igualdade (=);

%texto%: Serão retornados os registros que contêm no campo buscado o "texto" informado. Por exemplo, podemos buscar os nomes que contêm "Santos", ou que contêm uma sílaba ou letra específica. O registro com nome "Luis da Silva", por exemplo, contém o termo "da", então atenderia ao filtro '%da%';

Funções

Operador Like

%texto: Serão retornados os registros cujo valor do campo filtrado termina com o "texto" informado. O %, nesse caso, indica que pode haver qualquer valor no começo do campo, desde que ele termine com o "texto". Por exemplo, o registro com nome "Luis da Silva" atenderia ao filtro '%Silva';

texto%: Serão retornados os registros cujo valor do campo filtrado começa com o "texto" informado. Dessa vez, o % indica que após o "texto" pode haver qualquer valor. Por exemplo, o registro com nome "Luis da Silva", atenderia ao filtro 'Luis%'.

Funções

Operador Like

operador LIKE é utilizado para buscar por uma determinada *string* dentro de um campo com valores textuais.

Exemplo:

```
select count(*) from clientes  where email like
"%gmail%";
```

O que São Gatilhos?

- Na linguagem SQL, os gatilhos (triggers) são procedimentos armazenados especiais.
- São desencadeados automaticamente pelo Sistema de Gerenciamento de Banco de Dados (SGBD) em resposta a eventos específicos.

A Mecânica dos Gatilhos

- Os gatilhos são armazenados como objetos independentes na base de dados.
- São semelhantes a procedimentos (*stored procedures*), mas sua execução é acionada pelo SGBD em resposta a eventos do banco de dados.
- Não aceitam argumentos e são executados automaticamente.

Casos de Uso Comuns

- Gatilhos são usados para manter a consistência dos dados em resposta a eventos em tabelas específicas.
- Exemplo: Registro de alterações de clientes para manter um histórico.

Diferença entre Stored Procedures e Gatilhos

- Os gatilhos são um tipo especial de procedimento armazenado.
- A principal diferença: os gatilhos são acionados automaticamente em resposta a eventos, enquanto os stored procedures são chamados manualmente.

Sintaxe da Trigger

```
CREATE TRIGGER nome_do_gatilho
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON nome_da_tabela
FOR EACH ROW
BEGIN
  -- Lógica da trigger
END;
```

https://www.devmedia.com.br/mysql-basico-triggers/37462

Sintaxe da Trigger

1. **CREATE TRIGGER nome_do_gatilho:** Isso define o início da criação da trigger. O nome_do_gatilho é o nome que você escolhe para a sua trigger. É uma convenção usar nomes descritivos que indiquem o propósito da trigger.

1. **{BEFORE | AFTER}:** Indica se a trigger deve ser acionada antes ou depois da ação específica. Você pode escolher entre "BEFORE" (antes) ou "AFTER" (depois) da ação que desencadeia a trigger.

1. **{INSERT | UPDATE | DELETE}:** Especifica o tipo de ação do banco de dados que acionará a trigger. Pode ser "INSERT" (inserção de dados), "UPDATE" (atualização de dados) ou "DELETE" (exclusão de dados).

1. **ON nome_da_tabela:** Indica em qual tabela a trigger será aplicada. A trigger responderá a eventos ocorridos nesta tabela.

1. **FOR EACH ROW:** Indica que a trigger será acionada para cada linha afetada pela ação específica (por exemplo, para cada linha inserida, atualizada ou excluída).

1. **BEGIN ... END:** Dentro deste bloco, você define a lógica da trigger. Este é o código que será executado quando a trigger for acionada. Você pode usar SQL para realizar operações desejadas, como atualizar outras tabelas, inserir registros em uma tabela de histórico, etc.

PRÁTICA

TRIGGERS

EDUCAÇÃO METODISTA

Uma Stored Procedure é um conjunto de comandos SQL que executa uma tarefa específica. Ao contrário de triggers, que são acionadas automaticamente, uma procedure requer ativação a partir de um programa ou manualmente pelo usuário.

Benefícios das Stored Procedures

Reusabilidade de Código: uma stored procedure pode ser utilizada por vários usuários e em diferentes contextos, seja em scripts SQL, triggers ou aplicações.

Portabilidade: Stored procedures são altamente portáteis, permitindo uma fácil transferência entre sistemas.

Aprimoramento de Desempenho: armazenar a lógica da aplicação no banco de dados reduz o tráfego de rede em ambientes cliente/servidor, resultando em maior eficiência.

Manutenção Centralizada: ao manter o código no servidor, qualquer alteração feita é imediatamente refletida para todos os usuários, eliminando a necessidade de gerenciar versões distribuídas da aplicação.

Sintaxes

Sintaxe de Criação:

```
CREATE PROCEDURE nome_procedimento (parâmetros) AS
BEGIN
  -- declarações
END;
```

Invocando a Procedure:

```
CALL nome_procedimento (parâmetros);
```

Removendo uma Stored Procedure:

```
DROP PROCEDURE nome_procedimento;
```

Sintaxes

Stored Procedure

```sql
-- Exemplo de uma Stored Procedure para inserir um novo produto e atualizar o estoque
CREATE PROCEDURE InserirProduto(IN pNome VARCHAR(255), IN pPreco DECIMAL(10, 2), IN pQuantidade INT)
BEGIN
    -- Inserir um novo produto na tabela Produtos
    INSERT INTO Produtos (Nome, Preco, Quantidade) VALUES (pNome, pPreco, pQuantidade);

    -- Atualizar o estoque total
    UPDATE EstoqueTotal SET Quantidade = Quantidade + pQuantidade WHERE ProdutoID = LAST_INSERT_ID();
END;
```

CREATE PROCEDURE InserirProduto(IN pNome VARCHAR(255), IN pPreco DECIMAL(10, 2), IN pQuantidade INT): Esta linha cria uma nova stored procedure chamada "InserirProduto". Ela aceita três parâmetros de entrada: "pNome" (nome do produto), "pPreco" (preço do produto) e "pQuantidade" (quantidade em estoque).

BEGIN: A cláusula **BEGIN** marca o início do corpo da stored procedure, onde todas as ações a serem executadas são definidas.

INSERT INTO Produtos (Nome, Preco, Quantidade) VALUES (pNome, pPreco, pQuantidade);: Esta instrução SQL insere um novo registro na tabela "Produtos" com os valores fornecidos nos parâmetros "pNome", "pPreco" e "pQuantidade".

UPDATE EstoqueTotal SET Quantidade = Quantidade + pQuantidade WHERE ProdutoID = LAST_INSERT_ID();: Esta instrução SQL atualiza o estoque total na tabela "EstoqueTotal". Ela aumenta a quantidade existente em estoque com base na quantidade fornecida no parâmetro "pQuantidade". O **LAST_INSERT_ID()** é uma função que retorna o ID do último registro inserido na tabela "Produtos", garantindo que o estoque total seja atualizado para o produto correto.

END;: A cláusula END marca o final do corpo da stored procedure.

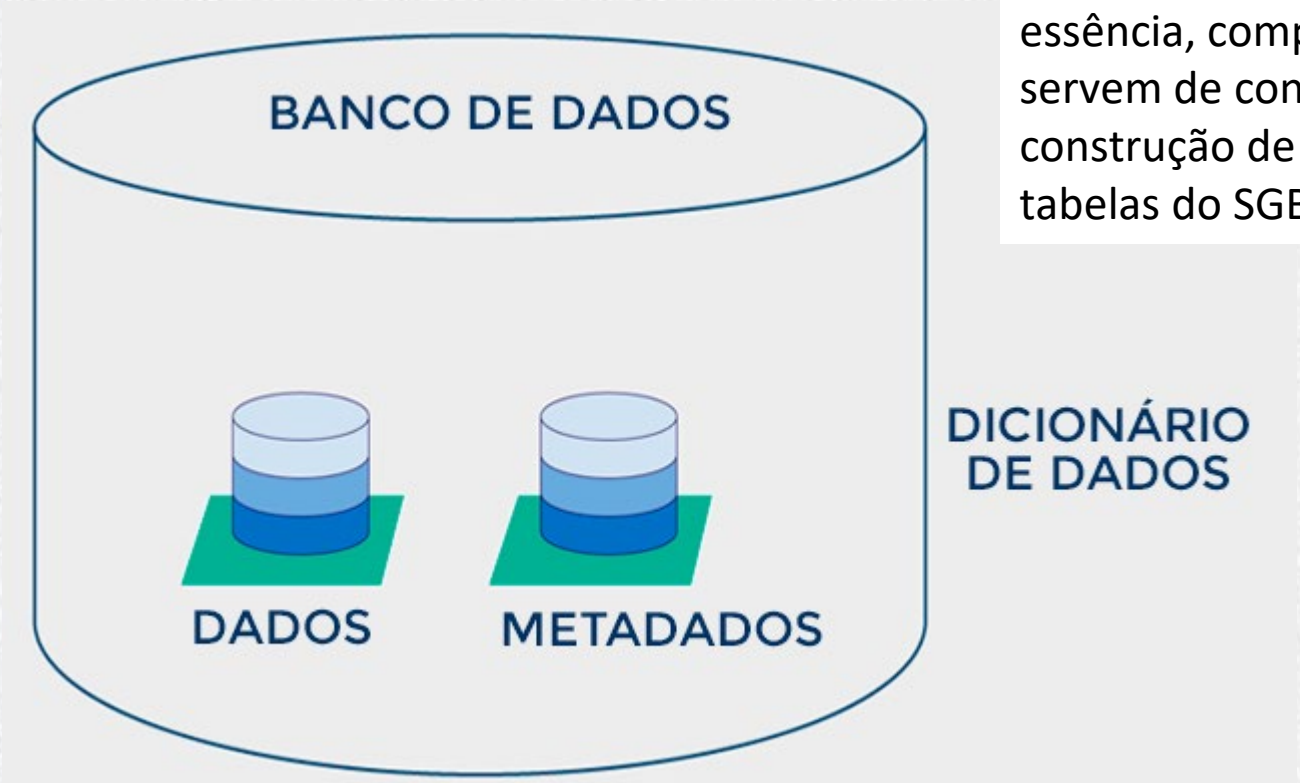
PRÁTICA

Stored Procedure

EDUCAÇÃO METODISTA

Dicionário de Dados

É uma estrutura que registra as características definidas para os dados que serão armazenados no banco de dados.



Um dicionário de dados é, em essência, composto por tabelas que servem de consulta para a construção de todas as outras tabelas do SGBD.

EDUCAÇÃO METODISTA

Dicionário de Dados

Criação

No processo de desenvolvimento, na etapa de análise, quando se constrói o modelo conceitual de dados, se inicia a definição e, na etapa de projeto, se efetiva a formação, a partir da criação dos objetos no banco de dados.

Utilização

- ✓ Validação das definições dos objetos.
- ✓ Gestão do banco de dados.
- ✓ Verificação no atendimento a uma transação.
- ✓ Verificação do atendimento às regras de integridade, na finalização da transação.

Dicionário de Dados

Restauração

Realizada a partir das cópias de segurança.

Vantagens

- ✓ Minimiza erros na modelagens dos dados.
- ✓ Compõe a documentação do sistema, garantindo sua continuidade.
- ✓ Representa um identificador dos objetos de banco de dados para constante validação de recursos utilizados.

Dicionário de Dados

Informações complementares

É indicado o uso de padrão para o nome dos objetos e de suas identificações.

Cada SGBD tem uma estrutura própria para definição das informações do dicionário de dado

Informações catalogadas para objeto constraint

INFORMATION_SCHEMA TABLE_CONSTRAINTS

Constraint_Catalog	*Constraint_Schema*	*Table_schema*	*Table_name*	*Constraint_type*	*Enforced*
def	Information_schema	Table_constraints	aluno	primary key	yes
def	Information_schema	Table_constraints	aluno	foreign key	yes
def	Information_schema	Table_constraints	curso	primary key	yes

Dicionário de Dados

O dicionário de dados vem para realizar esta padronização básica e extremante importante, visto que, à medida que o banco de dados cresce e mais desenvolvedores são escalados para trabalhar no projeto, mais fora de padrão pode ficar o banco de dados como um todo.

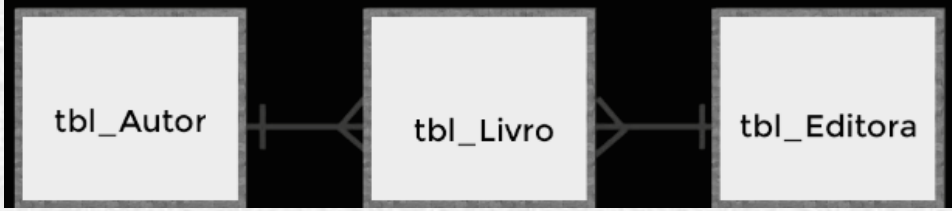
"(DDD) xxxxx-xxxx".

EDUCAÇÃO METODISTA

Dicionário de Dados

Exemplo de Dicionário de Dados

Tabela	Relacionamento	Nome do Relacionamento	Descrição
tbl_Livro	tbl_Autor	Escreve	Tabela para cadastro dos livros da coleção
	tbl_Editora	Publica	
tbl_Autor	tbl_Livro	Escreve	Tabela para cadastro dos autores dos livros
tbl_Editora	tbl_Livro	Publica	Cadastro de editoras



EDUCAÇÃO METODISTA

Dicionário de Dados

Exemplo de Dicionário de Dados - ATRIBUTOS

Tabela	Nome da Coluna	Tipo de Dados	Comprimento	Restrições	Valor Padrão	Descrição
tbl_Livro	ID_Livro	Inteiro	4 bytes	PK, NOT NULL	N/D	Número de identificação do livro, gerado automaticamente
	Nome_Livro	Caracteres	40 bytes	NOT NULL	N/D	N/D
	ID_Autor	Inteiro	4 bytes	FK	N/D	Nº de identificação do autor
	ID_Editora	Inteiro	4 bytes	FK	N/D	Nº de identificação da editora

EDUCAÇÃO METODISTA

Criação do Dicionário de Dados

A criação do dicionário de dados se dá a partir das instruções realizadas durante a criação e/ou alteração das estruturas do banco de dados. Na SQL, o comando responsável pela criação das estruturas de dados é o **CREATE**.

```
SHOW CREATE TABLE List;
```

EDUCAÇÃO METODISTA