MARIA DB ESSENTIALS

O que é Maria DB?

É um banco de dados relacional: todas as informações são relacionadas em torno de um problema que deve ser resolvido; relacionamento 1...1, relacionamento 1...N e relacionamento de N... M.

Transacional: uma sequência de processos, que quando executadas formam apenas uma ação; ACID

1. Atomicidade: todo processo deve ter um fim,
2. Consistência: todos os processos devem ser executados obedecendo todas as regras e restrições impostas,
3. Isolamento: nenhuma transação pode afetar outra em andamento,
4. Durabilidade: [ou persistência]: toda informação escrita no repositório só pode ser desfeita/refeita por outra transação.

Normalizado: nas 5 formas normais tende a ser extremamente dinâmico, porém com perda de desempenho.

CARACTERÍSTICA DO MARIA DB

Implementação C e C++,

Multiplataforma,

Open Source,

Aceita várias linguagens de programação,

Comparação mariaDB e MySQL,

Repositório

ESCOLHA SEU STAGE

TIPOS DE AMBIENTE

Produção - Serviço do MariaDB no cPanel,

Homologação (sandbox) - Localhost,

Serviço dedicado.

INSTALAÇÃO NO UBUNTU

Instalação feita através do terminal seguindo o passo a passo do site: Produtos e Ferramentas MariaDB baixa | MariaDB

Após selecionar o SO, clicar no link MariaDB Package repository

Abra o terminal e utilize as linhas de comando da página do MariaDB.

Após instalação, execute o comando: mysql –u root –p

CREATE DATABASE teste;

SHOW DATABASES;

USE teste;

ESCOLHA SEU APLICATIVO

PARAMETRIZANDO ESTAÇÃO

TESTANDO CONEXÃO - telnet nome do domínio 3306

Teste conexão: telnet localhost 3306 -> Deve ser habilitado no Windows adicionar ou remover componentes do Windows.

COMPREENDENDO O PROCESSO

EXECUTANDO IMPORT

EXECUTANDO EXPORT

Marcar Personalizado – estrutura

TABELAS CAMPOS E ATRIBUTOS

TIPOS DE DADOS INTEIROS

TIPO DE DADOS [NÚMEROS REAIS]

TIPO DE DADOS [TEXTOS]

TIPOS DE DADOS [DATAS E HORAS]

TIPOS DE DADOS [OTHERS] -> tipos geométricos de dados, autoincremente e null.

OVERVIEW DDL

Linguagem de definição de dados (Data Definition language)

TRABALHANDO COM CREATE [DATABASE] -> CRIANDO BANCO DE DADOS!

```
CREATE DATABASE mod_essentials DEFAULT CHARACTER SET 'utf8' DEFAULT COLLATE = 'utf8_general_ci';
```

```
CREATE DATABASE IF NOT EXISTS mod_essentials DEFAULT CHARACTER SET 'utf8' DEFAULT COLLATE =
'utf8_general_ci'; ----->>>> O “IF NOT EXISTS” verifica se o banco já existe
```

```
USE mod_essentials; --->>>> Podemos utilizar o BD
```

TRABALHANDO COM CREATE [TABLE]

```
CREATE TABLE teste(
    teste_id INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY COMMENT 'Campo para
armazenar o ID',
    teste_nome VARCHAR(255) NOT NULL,
    teste_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP()
)
```

MODIFICANDO COM ALTER

```
ALTER DATABASE mod_essentials DEFAULT COLLATE = ' ';
```

```
ALTER TABLE teste MODIFY COLUMN teste_name VARCHAR(100) NOT NULL;
```

```
ALTER TABLE teste ADD COLUMN teste_descricao TEXT AFTER teste_nome;
```
--->>> Acrescentou uma coluna após a coluna teste_nome.

```
ALTER TABLE teste ADD COLUMN teste_title TEXT FIRST teste_nome;
```
---->> Inseriu uma coluna antes da coluna teste_nome.

```
ALTER TABLE teste DROP COLUMN if EXISTS teste_titulo;
```
--->>> Excluiu a coluna teste_titulo da tabela.

DELETANDO COM TRUNCATE

```
TRUNCATE TABLE teste; --->>> Apaga todos os dados da tabela
```

```
DELETE FROM teste;
```
---->>> Apaga todos os dados da tabela

RENAME DE TABELA

RENAME TABLE teste TO teste_geral;

RENAME TABLE teste_geral TO usuarios, teste2 TO produtos; -->>> Renomeando duas tabelas

REMOVENDO ESTRUTURA COM DROP

DROP TABLE usuario; -->> Apaga a tabela do banco de dados

OVERVIEW DML (CRUD) - DATA MANIPULATION LANGUAGE (LINGUAGEM DE MANIPULAÇÃO DE DADOS)

EXECUTANDO INSERT

```
INSERT INTO tbl_users ( user_name, user_lastname, user_created, user_updated, user_years_old, user_type, user_birth) VALUES ('Fábio', 'Freitas', DEFAULT, DEFAULT, '39', 'customer', 1982-06-26);
```

INSERT INTO tbl_users SET user_name = 'Fábio1', user_lastname = 'Freitas1'; --> alterando somente nome e sobre nome

SELECIONANDO REGISTRO COM SELECT/WHERE

SELECT * FROM tbl_users; -> mostra a tabela inteira.

SELECT user_id, user_name, user_lastname FROM tbl_users; -> Trans informações especificas.

```
SELECT
user_id,
   UPPER(user_name),
   user_lastname,
   DATE_FORMAT(user_created, '%d/%m/%Y %H:%i:%s') user_created_brl
   FROM tbl_users
   -- JOIN
```

WHERE user_years_old IS NOT NULL; -> trazendo as informações com a data em formado BR e filtrando os resultados que esteja com a idade nula.

WHERE user_years_old BETWEEN 23 AND 25; -> filtra as informações com idade entre 23 e 25 anos.

```
INSERT INTO tbl_usuarios (usuario_name, usuario_idade) SELECT
UPPER(user_name),
   user_years_old
```

FROM tbl_users

WHERE user_years_old > 23; ->Inserte trabalhando com select para migrar informação de uma tabela para outra.

ATUALIZANDO COM UPDATE

UPDATE tbl_users SET user_name = 'Robson', user_lastname = 'V. Leite'

WHERE user_id = 1 ---->>>> Atualizando informações do usuário de ID = 1.

UPDATE tbl_users SET user_name = 'Robson', user_lastname = 'V. Leite'

WHERE user_id = 1 --->>>Alterando id do usuário

```
UPDATE tbl_users u, tbl_usuarios u2
SET
        u.user_name = 'Robson',
    u2. usuario_nnome = 'Robson'
  WHERE u.user_id = u2. usuario_id
        AND u.user_id = 1; ----->>> unindo o registro de duas tabelas através do id do usuário.
```

EXCLUINDO COM DELETE

DELETE FROM tbl_users WHERE user_type IS NULL; -->>> Deletando usuário com campos setado como NULL.

OVERVIEW TCL - TOOL COMMAND LANGUAGE (LINGUAGEM DE COMANDOS E FERRAMENTAS)

BEGIN, COMMIT E ROLLBACK

EXEMPLOS PRÁTICOS DE TRANSAÇÃO

BEGIN; -> abre uma nova transação

COMMIT; -> escreve a transação dentro do BD e encerra a transação.

Exemplo:

BEGIN; Inicia uma transação

COMMIT; certo, pode salvar as informações no BD

UPDATE tbl_users SET user_name = 'Teste'; -> Alterou o nome dos usuários da tabela para teste

SELECT * FROM tbl_users; -> mostrou o resultado.

EXEMPLO PRÁTICO COM ROLLBACK

Descantando as alterações realizada na transação anterior!

BEGIN;

COMMIT;

ROLLBACK; -> Encerra a transação

UPDATE tbl_users SET user_name = 'Fábio' WHERE user_id = 1;

SELECT * FROM tbl_users;

FUNÇÕES NATIVAS

FUNÇÕES PARA STRINGS

```
SELECT
    UPPER(user_name) user_name_epper, -->Deixa em caixa alta
    UCASE(user_name) user_name_ucase, -->>Deixa em caixa alta
    LOWER(user_name) user_name_lower, -->>Deixa em caixa baixa
    LCASE(user_name) user_name_lcase, -->>Deixa em caixa baixa
    CHAR_LENGTH(user_name) char_length_campo, -->>Retorna quantidade de caracteres
    LEFT(user_name, 4) user_name_left, -->>retorna 4 caracteres começando da esquerda
    RIGHT(user_name, 4) user_name_right, -->>retornar 4 caracteres començando da direita
    SUBSTR(user_name, 2) user_name_substring,
    SUBSTRING(user_name FROM 2) user_name_substring_2,
    SUBSTRING(user_name, 2, CHAR_LENGTH(user_name) -1) user_name_sbstring_3,
    SUBSTRING(user_name FROM 2 FOR CHAR_LENGTH(user_name) -1) user_name_sbstring_4,
    user_name

    FROM tbl_users;
```

```
SELECT
   LTRIM(user_name) user_name_ltrim, -->>Limpa todos os espaços a esquerda da String
    RTRIM(user_name) user_name_ltrim, -->Limpa os espaços a direita da String
```

```
    RTRIM(LTRIM(user_name)) user_name_fulltrim, -->Limpando espaços em ambos os lados
    TRIM(user_name) user_name_trim, -->Faz a mesma coisa da linha a cima
    TRIM(TRIM(BOTH '123' FROM user_name)) user_name_trim2, --> limpando espaço e números na String de ambos os lados
    user_name
    FROM tbl_users;
SELECT
        //Campo de 50 caracteres com X antes da string
 LPAD(TRIM(TRIM(BOTH '123' FROM user_name)), 50, 'x') user_name_trim,
//Completando com X após a string
RPAD(TRIM(TRIM(BOTH '123' FROM user_name)), 50, 'x') user_name_trim,
   user_name
   FROM tbl_users;
CONCAT() -->> concatena os campos



FUNÇÃO PARA DATA/HORA
SELECT
        CURRENT_TIMESTAMP,
        NOW(),
        CURRENT_TIME,
        CURTIME(),
        CURRENT_DATE,
        CURDATE()
SELECT
         user_name,
         DAY(user_created),
         MONTH(user_created),
         YEAR(user_created),
         HOUR(user_created),
         MINUTE(user_created),
```

```
SECOND(user_created)

FROM tbl_users


SELECT
user_name,
    user_created,
    ADDDATE(user_created, INTERVAL -1 MONTH),
    DATE_ADD(user_created, INTERVAL -1 MONTH),
    SUBDATE(user_created, INTERVAL -1 MONTH),
    DATE_SUB(user_created, INTERVAL -1 MONTH),
    ADDTIME(user_created, '05:00:00'),
    SUBTIME(user_created, '01:00:00'),
    DATEDIFF(CURRENT_DATE, user_created), -- Descobrir a quanto tempo o cadastro foi inserido no BD
    TIMEDIFF(CURRENT_TIME, TIME(user_created)), -- Descobrir a quantas horas o cadastro foi feito no BD
    TIMESTAMPDIFF(DAY, user_created, CURRENT_TIMESTAMP), -- Há quantos dias o cadastro foi realizado no BD
    DATE_FORMAT(user_created, '%d/%m/%Y'), -- Mostra a data que o cadastro foi feito no BD
    CONCAT('Usuário cadastrado em: ', DATE_FORMAT(user_created, '%d/%m/%Y às %H:%i:%s'), '.')
    FROM tbl_users


                                    OPERADORES

SELECT
user_id,
    user_name,
    user_years_old as user_years_old_valid,
    user_years_old + 5,
    user_years_old + user_years_old,


    user_years_old - 5,
```

```
    user_years_old - user_years_old,

    user_years_old / 5,
    user_years_old / user_years_old,

    user_years_old * 5,
    user_years_old * user_years_old,

    user_years_old DIV 5,

    user_years_old MOD 5 -- Resto

    FROM tbl_users;
```

OPERADORES COMPARATIVOS

OPERADORES COMPARATIVOS (BETWEEN)

```
SELECT
user_name,
    user_years_old,
    user_birth
    FROM tbl_users
    WHERE DATE_FORMAT(user_birth, '%Y-%m') < '1995-02';
    WHERE user_years_old BETWEEN '20' AND '29'; -- ou OR também NOT BETWEEN
```

OPERADORES COMPARATIVOS (IS) -> Utilizado em tipos nulo booleam

```
SELECT * FROM tbl_users WHERE user_created IS NULL;

IS true
IS false
IS NULL
IS NOT NULL
```

```
UPDATE tbl_users SET user_created = NULL, user_years_old = null WHERE user_id = 1


SELECT
user_name,
    user_status,

    CASE
        WHEN user_status IS TRUE THEN 'Ativo'
      WHEN user_status IS FALSE THEN 'Ativo'
      ELSE 'Pendente' END as status_geral
    FROM tbl_users;
```

OPERADORES COMPARATIVOS (IN)

```
SELECT
user_name,
    user_lastname,
    user_type_id

    FROM tbl_users
    WHERE user_type_id IN (SELECT user_type_id FROM tbl_users_types WHERE user_type_group
= 'employee');
```

OPERADORES LÓGICOS

```
SELECT
user_name,
    user_lastname,
    user_type_id,
    user_status

    FROM tbl_users
    -- Retornar todos os usuários com tipo employee e o uauário seja marcado ativos ou data de
nacimento hoje
```

```
    WHERE user_type_id IN (SELECT user_type_id FROM tbl_users_types WHERE user_type_group
= 'employee')

                AND (user_status IS TRUE OR user_birth = CURRENT_DATE)

          OR user_type_id IN (SELECT user_type_id FROM tbl_users_types WHERE user_type_group
= 'provider');
```

FORMAS NORMAIS

1FN - MULTIVALORAÇÃO E SEGMENTAÇÃO

CONSULTAS X RESTRIÇÕES

```
SELECT *

FROM enderecos e

    INNER JOIN cidades c ON c.id_cidade = e.id_cidade
```

TEORIA DOS CONJUNTOS

```
SELECT *

FROM enderecos e

    INNER JOIN cidades c ON c.id_cidade = e.id_cidade
```

```
SELECT * FROM enderecos CROSS JOIN cidades
```

```
SELECT *

FROM enderecos

    INNER JOIN cidades USING (id_cidade)
```

```
SELECT *

FROM enderecos

    INNER JOIN cidades ON cidades.id_cidade = enderecos.id_cidade
```

CRIANDO CONSULTAS

```
SELECT *

FROM usuarios u

    LEFT JOIN enderecos e ON e.id_usuario = u.id_usuario
```

```
    LEFT JOIN cidades c ON c.id_cidade = e.id_cidade
```

JOINS X SUBQUERIES

```
-- Pedidos
SELECT
p.id_pedido,
    CONCAT(u.nome, ' ', u.sobrenome) nome_completo_usuario,
    prod.produto,
    prod.valor_unitario,
    p.quantidade,

    prod.valor_unitario * p.quantidade valor_total

FROM pedidos p
    INNER JOIN usuarios u ON u.id_usuario = p.id_usuario
    INNER JOIN produtos prod ON prod.id_produto = p.id_produto

SELECT
p.id_pedido,
    (SELECT CONCAT(u.nome, ' ', u.sobrenome) FROM usuarios u WHERE u.id_usuario =
p.id_usuario) nome_completo_usuario,
    (SELECT prod.produto FROM produtos prod WHERE prod.id_produto = p.id_produto) produto,
    (SELECT prod.valor_unitario FROM produtos prod WHERE prod.id_produto = p.id_produto)
valor_unitario,

    p.quantidade,
    (SELECT prod.valor_unitario FROM produtos prod WHERE prod.id_produto = p.id_produto) *
p.quantidade valor_unitario

    FROM pedidos p
```

DESENVOLVENDO APLICAÇÕES

SELECT PELA APLICAÇÃO

UPDATE PELA APLICAÇÃO