RESTRIÇÕES SQL

Prof.Me.Joice Wolfrann

RESTRIÇÕES

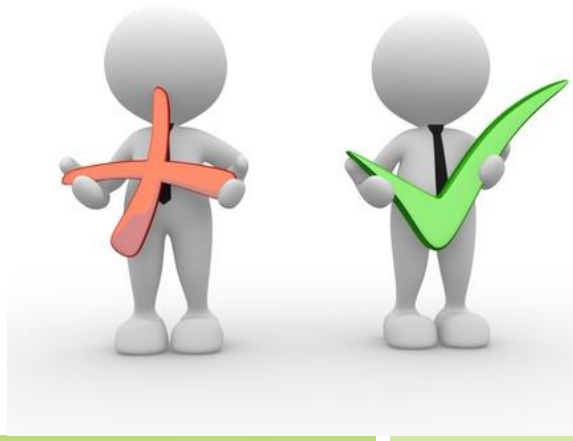
- Tipos de dados são uma forma de limitar os dados que podem ser armazenadas em uma tabela.

- Entretanto para muitas aplicações não é suficiente.
  - Uma coluna contendo um preço de um produto pode provavelmente aceitar somente valores positivos.
  - Não há nenhum tipo de dado padrão que aceite somente números positivos.

- Pode-se ainda precisar criar uma coluna restrita em relação ao valor de outra coluna.
- Em uma tabela contendo informação de produto deve existir somente uma linha para cada número de produto.

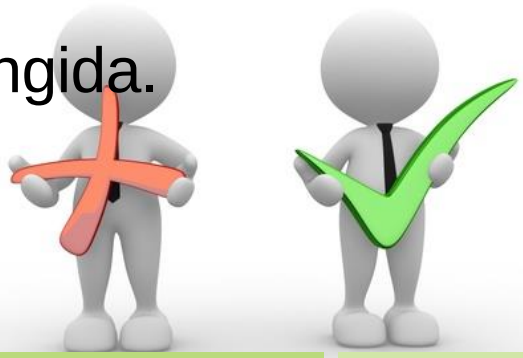
RESTRIÇÕES

- SQL permite definir restrições em colunas e tabelas.
- Restrições permitem um maior controle dos dados nas tabelas.
- Se um usuário tentar armazenar dado em uma coluna que viola a restrição, um erro é levantado.



RESTRIÇÕES CHECK

- Uma restrição CHECK é o tipo mais genérico de restrição.
- Permite que se especifique qual valor em uma certa coluna deve satisfazer uma expressão booleana.
- A restrição CHECK é inserida depois do tipo do dado.
- Consiste na palavra-chave CHECK seguida de uma expressão em parênteses.
- A expressão deve envolver a coluna restringida.



RESTRIÇÕES CHECK

```
CREATE TABLE Produto (
        produto_num integer,  nome
        text,
        preco numeric CHECK (preco > 0)
);
```

RESTRIÇÕES CHECK

- É possível dar um nome específico para a restrição.
- Utilizar a **palavra-chave CONSTRAINT (I)** seguida do **identificador (II)** e a definição da **restrição (III)**.
- É importante para tornar os possíveis erros mais claros.
- É importante para facilitar a alteração de uma restrição específica.

```
CREATE TABLE Produto (
        produto_num integer,  nome
        text,
        preco numeric CONSTRAINT preco_positivo CHECK (preco > 0)
);
```

RESTRIÇÕES CHECK

- Uma restrição CHECK pode se referir a várias colunas:

```
CREATE TABLE Produto (
    produto_num integer,  nome
    text,
    preco numeric,
    preco_desconto numeric,
    CONSTRAINT preco_positivo CHECK (preco > 0),
    CONSTRAINT preco_desconto_positivo CHECK (preco_desconto > 0),
    CONSTRAINT desconto_valido CHECK (preco > preco_desconto)
);
```

RESTRIÇÕES NOT NULL

- Uma restrição NOT NULL especifica que uma coluna não deve assumir valor NULL.

```
CREATE TABLE Produto (
        produto_num integer NOT NULL,
        nome text NOT NULL,
        preco numeric NOT NULL CONSTRAINT preco_positivo CHECK (preco > 0),
        preco_desconto numeric CONSTRAINT preco_desconto_positive CHECK
(preco_desconto > 0),
        CONSTRAINT desconto_valido CHECK (preco > preco_desconto)
);
```

RESTRIÇÕES UNIQUE

A restrição UNIQUE garante que o dado inserido em um campo deve ser único se comparado as demais tuplas da tabela.

```
CREATE TABLE Produto (
        produto_num integer UNIQUE,
        nome text NOT NULL UNIQUE,
        preco numeric NOT NULL CONSTRAINT preco_positivo CHECK (preco > 0),
        preco_desconto numeric CONSTRAINT preco_desconto_positive CHECK
(preco_desconto > 0),
        CONSTRAINT desconto_valido CHECK (preco > preco_desconto)
);
```

PRIMARY KEYS

- Define a chave primária da tabela.
- Uma chame primária (PRIMARY KEY) é uma combinação de uma restrição UNIQUE e uma NOT NULL, **mas**:
  - Primary Key automaticamente cria um índice (btree) na (grupo de) coluna(s).
  - Uma tabela deve ter, no mínimo, uma chave primária.

```
CREATE TABLE Produto ( produto_num
    integer PRIMARY KEY, nome text NOT
    NULL UNIQUE,
    preco numeric NOT NULL CONSTRAINT preco_positivo CHECK (preco > 0),
    preco_desconto numeric CONSTRAINT preco_desconto_positive CHECK (preco_desconto > 0),
    CONSTRAINT desconto_valido CHECK (preco > preco_desconto)
);
```

PRIMARY KEYS

- Chaves podem ser compostas.
- A sintaxe, nesse caso, será:

```
CREATE TABLE Exemplo (
        a integer,
        b integer,
        c integer,
        PRIMARY KEY (a, c)
);
```

FOREIGN KEYS

- Restrições FOREIGN KEY especifica que os valores em uma coluna (ou grupo de colunas) deve corresponder aos valores existentes em algum outro campo (geralmente de outra relação).

- Garantir INTEGRIDADE REFERENCIAL

```
CREATE TABLE Produto (
        produto_num integer PRIMARY KEY,
        nome text,
        preco numeric
);
```

```
CREATE TABLE Pedido (
        pedido_num integer PRIMARY KEY,
        produto_num integer REFERENCES produto (produto_num) ,  qtd
        integer
);
```

FOREIGN KEYS

- Uma FOREIGN KEY pode restringir e referenciar um grupo de colunas.
- **SEMPRE O NÚMERO E O TIPO DAS COLUNAS DEVEM SER CORRESPONDENTES NAS DUAS TABELAS.**
- Exemplo:

```
CREATE TABLE Exemplo (
        a integer PRIMARY KEY,
        b integer,
        c integer,
        FOREIGN KEY (b, c) REFERENCES outro_exemplo (b1, c1)
);
```