

Restrições sobre Table's

Msc Denival Araújo dos Santos

SQL Constraints (Restrições)

Funcionamento

- Restrições são regras aplicadas em uma tabela (normalmente nas colunas), podendo ser especificadas no momento da criação da tabela (**CREATE**) ou após a tabela ser criada (**ALTER**).
- Principais:
 - **NOT NULL**
 - **PRIMARY KEY**
 - **FOREIGN KEY**
 - **UNIQUE**
 - **DEFAULT**
 - **CHECK**

SQL Constraints (Restrições)

NOT NULL

- Impõe a uma coluna **NÃO** aceitar valores **NULL** (não receber valor), o que torna o campo de preenchimento obrigatório.
- Há muitas situações em que a proibição de valores nulos é desejável. Um caso em particular é a proibição de valores nulos em uma chave primária. Se um usuário tentar salvar uma tupla sem informar a chave primária, o SGBD intercepta e emite um diagnóstico de erro.

Exemplo

```
create table pessoa(  
  id int not null,           // Não aceita valor NULL  
  nome varchar(100) null,   // aceita valor null  
  endereco varchar(50),     // Por default aceita NULL  
  primary key(id)  
);
```

SQL Constraints (Restrições)

Chave
Primária
(PK)

- Uma **PRIMARY KEY** identifica de forma única cada registro em uma tabela, ou seja, os valores armazenados devem sempre conter valores únicos e não ser NULL.
- Cada tabela deve ter apenas uma PRIMARY KEY (formada por um ou mais atributos).

Exemplo

```
create table pessoa(  
  id int not null,           // PK - valor de id é sempre unico  
  nome varchar(100) null,  
  cpf varchar(11) unique,  
  endereco varchar(50),  
  primary key(id)  
);
```

SQL Constraints (Restrições)

Chave
Estrangeira

(FK)

Exemplo

- O atributo definido com **FOREIGN KEY** (Chave estrangeira) em uma tabela aponta para uma chave primária em outra tabela.
- Ela é a chave de uma relação 1 para N, utilizada para identificação da tabela pai na tabela filho (relacionamento).
- **Observação:** No MySQL o uso de FOREIGN KEYS só é suportado pelas engine InnoDB e MariaDB.

```
create table pai (  
  id_pai int not null,  
  pai varchar(35),  
  primary key(id_pai)  
);
```

```
create table filho (  
  Id_filho int not null,  
  id_pai int,  
  filho varchar(35),  
  primary key(id_filho),  
  foreign key(id_pai) references pai(Id_pai)  
);
```

SQL Constraints (Restrições)

Chave
Estrangeira
(FK)

- **Observação:** Quando não se especifica o nome da Constraint, assume-se o valor padrão no formato **tabela_ibfk_1**, onde o 1 é o número da constraint. Para que o nome seja mais amigável é indicado colocar um nome na sua declaração.

Exemplo

```
create table cidade(  
    idcidade int not null,  
    nome varchar(35) not null,  
    estado_uf varchar(2) not null,  
    primary key(idcidade),  
    constraint fk_cidade_estado foreign key(estado_uf) references estado(uf)  
);
```

SQL Constraints (Restrições)

Valores
Duplicados
(UNIQUE)

- Identifica de forma única cada registro em uma tabela de um BD. As constraints UNIQUE e PRIMARY KEY garantem a unicidade de uma coluna ou conjunto de colunas. Ao declarar um atributo PRIMARY KEY automaticamente ele se tornará também UNIQUE, sendo que podemos ter várias constraints UNIQUE em uma tabela, mas apenas uma PRIMARY KEY.

Exemplo

```
create table pessoa(  
  id int not null,  
  nome varchar(100) null,  
  cpf varchar(11) unique, // cpf é unico  
  endereco varchar(50),  
  primary key(id)  
);
```


SQL Constraints (Restrições)

Padrão
(DEFAULT)

- A restrição **DEFAULT** é usada para inserir um valor padrão em uma coluna, e será acionado sempre que não for informado um valor para este para o atributo.

Exemplo

```
-- Criação da tabela
Create table Cidade(
    id_cidade int not null,
    nome varchar(50) null,
    uf varchar(2) Default "PI",
    primary key(id_cidade)
)
-- Povoar a tabela
insert into Cidade values(1,"Timon","MA");
insert into Cidade(id_cidade, nome) values(2,"Parnaíba");
-- Saída
```

|  id_cidade | nome | uf |
|---|----------|----|
| 1 | Timon | MA |
| 2 | Parnaíba | PI |

SQL Constraints (Restrições)

Checagem (CHECK)

- A restrição **CHECK** é um recurso de validação usado para restringir o intervalo de valores que pode ser usados em tabelas antes dos comandos Insert, Update serem aplicados no banco. Pode ser utilizado em uma ou mais colunas.

Exemplo

```
CREATE TABLE pessoaAdulta (  
    id int NOT NULL,  
    primeiroNome varchar(255) NOT NULL,  
    sobrenome varchar(255),  
    idade int,  
    cidade varchar(35),  
    CHECK (idade >= 18 and cidade = 'Parnaíba')  
);
```

```
insert into pessoaAdulta values(1, 'Denival', 'Santos', 46, 'Parnaíba');  
// Erro nas duas últimas linhas  
insert into pessoaAdulta values(3, 'Adão', 'Portela', 17, 'Parnaíba');  
insert into pessoaAdulta values(3, 'Adão', 'Portela', 35, 'Altos');
```

Referências bibliográficas

Básica

- SILBERSCHATZ, Abraham; KORTH, Henry F; SUDARSHAN, S. **Sistema de banco de dados**. Rio de Janeiro: Elsevier, 2012. 861 p. ISBN 978-85-352-4535-6.
- MACHADO, Felipe Nery Rodrigues. **Banco de dados: projeto e implementação**. 3. ed. São Paulo: Érica, 2014. 396 p. ISBN 978-85-365-0019-5.
- ELMASRI, Ramez; NAVATHE, Shamkant. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson, 2011. xviii, 788 p. ISBN 978-85-7936-085-5 (broch.).

Referências bibliográficas

Complementar

- BEIGHLEY, Lyn. **Use a cabeça SQL**. Rio de Janeiro: Alta Books, 2008. 454 p. ISBN 978-85-7608-210-1.
- DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Campus, 2000. xxiii, 803 p. ISBN 85-352-0560-8
- HEUSER, Carlos Alberto. **Projeto de banco de dados**. 6. ed. Porto Alegre (RS): Bookman, 2009. TEOREY, Toby J. et al. **Projeto e modelagem de banco de dados**. 2. ed. Rio de Janeiro: Elsevier, 2014.