## ITB - INSTITUTO TECNOLÓGICO DE BARUERI

### LECO – LINGUAGEM ESTRUTURADA DE CONSULTA

**PROFESSORA VIVIANE** 



### **CONSTRAINTS**

Constraints são objetos utilizados com a finalidade de estabelecer regras referentes à integridade e à consistência nas colunas das tabelas pertencentes a um sistema de banco de dados. Isso é importante para planejar e criar tabelas, pois devemos garantir a integridade dos dados presentes nas colunas e identificar os valores validos para tais dados.

Tipos de integridade	Tipos de constraint
Chave Primária	Constraint Primary Key
Chave Estrangeira	Constraint Foreign Key e Constraint References
Chave Primária Secundária ou Chave Única	Constraint Unique
Regras de Validação	Constraint Check

## **CONSTRAINTS**

#### Regras de constraints

Cada tipo de constraint possui suas próprias regras. Vejamos, na tabela a seguir, quais são elas:

Tipos de constraints	Descrição
Constraint Primary Key (Chave Primária)	Uma coluna que é definida como chave primaria não pode aceitar valores nulos. Em cada tabela, pode haver somente uma constraint de chave primária.
Constraint Foreing Key (Chave Estrangeira)	Valores colunas podem ser definidas como chave estrangeira. No entanto, para que uma coluna seja definida dessa forma, é preciso que ela já tenha sido definida como chave primária em outra tabela. As colunas definidas como chave estrangeira podem aceitar valores nulos, e os datatypes das colunas relacionadas devem ser iguais.
Constraint Unique (Chave unica)	Várias colunas de uma tabela podem ser definidas como chave única e, ainda, aceitar valores nulos.
Constraint Check	Diversas colunas de uma tabela podem ser definidas como constraint check. Essas colunas podem aceitar valores nulos, mais isso depende das regras que são determinadas para elas.

Incluindo a constraint primary key em um campo na criação da tabela

## Sintaxe:

```
CREATE TABLE <nome da tabela>
(<nome do campo1> <tipo de dado> [ NULL | NOT NULL ],
  <nome do campo2> <tipo de dado> [ NULL | NOT NULL ],
...

CONSTRAINT <nome da constraint> PRIMARY KEY (<nome do campo>)
);
```

**EXEMPLO:** 

--Cria a tabela Autor □ CREATE TABLE AUTOR. (COD AUTOR DECIMAL(38), NOME VARCHAR2 (60) NOT NULL, DT NASCIMENTO DATE, BIOGRAFIA LONG, NACIONALIDADE VARCHAR2 (30) NOT NULL, CONSTRAINT PK AUTOR PRIMARY KEY (COD AUTOR) AT 屋 Saída do Script 🛚 🗴 Tarefa concluída em 0,222 segundos Table AUTOR criado.

CREATE TABLE < nome da tabela >

Incluindo a constraint primary key em mais de um campo (chave primária composta) na criação da tabela

## Sintaxe:

Table LIVRO AUTOR criado.

**EXEMPLO**:

```
--Criando a tabela associativa entre Livro e Autor com chave primária composta
    CREATE TABLE LIVRO AUTOR
      (COD LIVRO DECIMAL (38),
      COD AUTOR DECIMAL(38),
      CONSTRAINT PK LIVRO AUTOR PRIMARY KEY (COD AUTOR, COD LIVRO),
      CONSTRAINT FK LIVRO FOREIGN KEY (COD LIVRO) REFERENCES LIVRO (COD LIVRO),
      CONSTRAINT FK AUTOR FOREIGN KEY (COD AUTOR) REFERENCES AUTOR (COD AUTOR)
AT
Saída do Script X
                 Tarefa conduída em 0, 192 segundos
```

Incluindo a constraint primary key em um campo na alteração da tabela

## Sintaxe:

ALTER TABLE <nome da tabela>
ADD CONSTRAINT <nome da constraint> PRIMARY KEY (<nome do campo>);

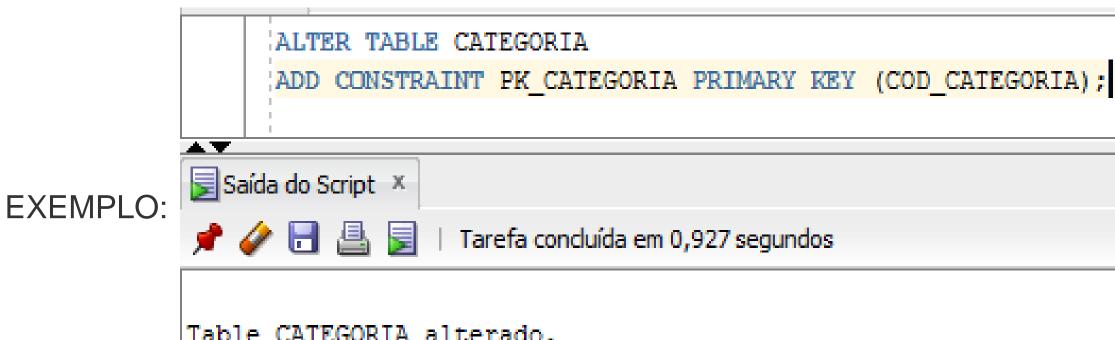


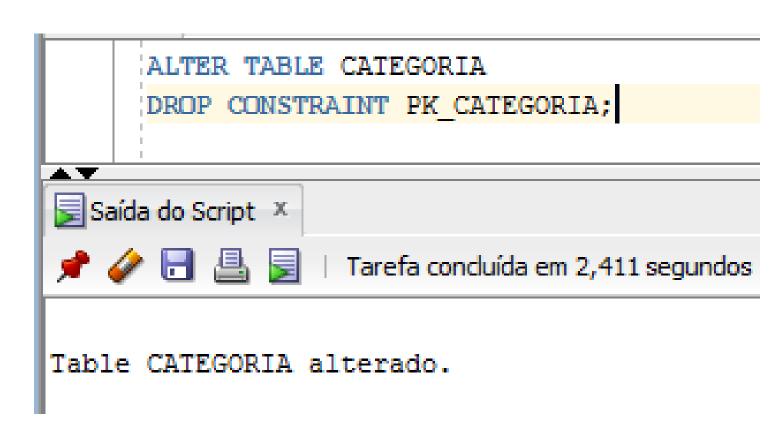
Table CATEGORIA alterado.

Removendo a constraint primary key

## Sintaxe:

ALTER TABLE <nome da tabela>
DROP CONSTRAINT <nome da constraint>;

Obs.: Só será possível excluir a PK, caso não tenha nenhuma FK vinculada em outra tabela.



**EXEMPLO**:

(<nome do campo1> <tipo de dado> [ NULL | NOT NULL ],

CREATE TABLE < nome da tabela >

Incluindo a constraint foreign key em um campo na criação da tabela

## Sintaxe:

```
<nome do campo2> <tipo de dado> [ NULL | NOT NULL ],
...
CONSTRAINT <nome da constraint> FOREIGN KEY (<nome do campo>) REFERENCES <nome da tabela de referência que possui a PK para fazer o relacionamento> (<nome do campo que é PK na tabela de referência>)
```

Table LIVRO criado.

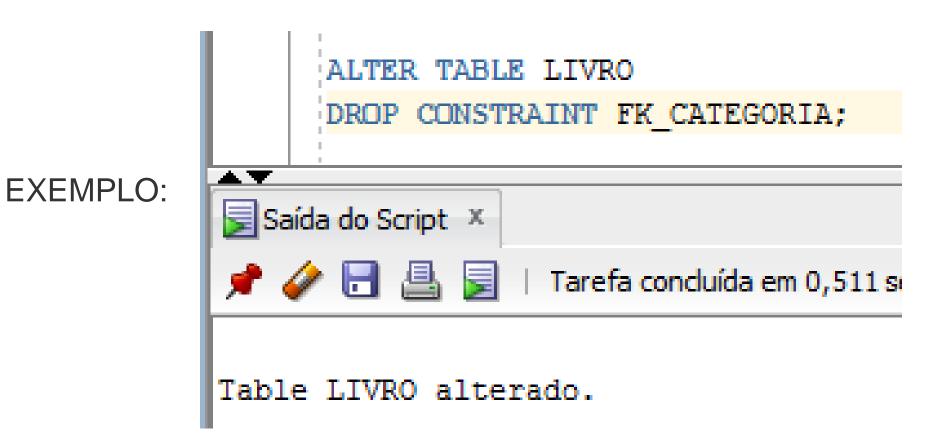
**EXEMPLO**:

```
--Cria a tabela Livro relacionando com a tabela Categoria
   CREATE TABLE LIVRO
     (COD LIVRO INT,
     TITULO VARCHAR2 (80) NOT NULL,
     ANO VARCHAR2 (4) NOT NULL,
     COD CATEGORIA INT,
     CONSTRAINT PK_LIVRO PRIMARY KEY (COD_LIVRO),
     CONSTRAINT FK_CATEGORIA FOREIGN KEY (COD_CATEGORIA) REFERENCES CATEGORIA (COD_CATEGORIA)
     1);
Saída do Script X
📌 🧼 🔡 볼 📘 | Tarefa concluída em 0,128 segundos
```

Removendo a constraint foreign key

Sintaxe:

ALTER TABLE <nome da tabela>
DROP CONSTRAINT <nome da constraint>;



Incluindo a constraint unique em um campo na criação da tabela

## Sintaxe:

```
CREATE TABLE <nome da tabela>
(<nome do campo1> <tipo de dado> [ NULL | NOT NULL ],
  <nome do campo2> <tipo de dado> [ NULL | NOT NULL ],
...

CONSTRAINT <nome da constraint> UNIQUE (<nome do campo>)
);
```

# **EXEMPLO**: CREATE TABLE CLIENTE (COD\_CLIENTE DECIMAL(38), NOME VARCHAR2(70) NOT NULL, CPF VARCHAR2(11) NOT NULL, GENERO CHAR(1) NOT NULL, ATIVO CHAR(1), CONSTRAINT UQ\_CPF\_CLIENTE UNIQUE (CPF),

Incluindo a constraint unique em mais de um campo na criação da tabela

## Sintaxe:

```
CREATE TABLE <nome da tabela>
(<nome do campo1> <tipo de dado> [ NULL | NOT NULL ],
  <nome do campo2> <tipo de dado> [ NULL | NOT NULL ],
...

CONSTRAINT <nome da constraint> UNIQUE (<nome do campo1>, <nome do campo2>)
);
```

Incluindo a constraint unique em um campo na alteração da tabela

## Sintaxe:

ALTER TABLE <nome da tabela>
ADD CONSTRAINT <nome da constraint> UNIQUE (<nome do campo>);

**EXEMPLO**:

ALTER TABLE CLIENTE

ADD CONSTRAINT UQ\_CPF\_CLIENTE UNIQUE (CPF);

Removendo a constraint unique

Sintaxe:

ALTER TABLE <nome da tabela>
DROP CONSTRAINT <nome da constraint>;

Desativando a constraint unique

Sintaxe:

ALTER TABLE <nome da tabela>
DISABLE CONSTRAINT <nome da constraint>;

Ativando a constraint unique

Sintaxe:

ALTER TABLE <nome da tabela>
ENABLE CONSTRAINT <nome da constraint>;

Incluindo a constraint CHECK em um campo na criação da tabela

### Sintaxe:

```
EXEMPLO:
CREATE TABLE CLIENTE
(COD_CLIENTE DECIMAL(38),
NOME VARCHAR2(70) NOT NULL,
CPF VARCHAR2(11) NOT NULL,
GENERO CHAR(1) NOT NULL,
ATIVO CHAR(1),
CONSTRAINT CK_GENERO_CLIENTE CHECK (GENERO IN('M', 'F'))
```

Incluindo a constraint check em um campo na alteração da tabela

## Sintaxe:

ALTER TABLE <nome da tabela>
ADD CONSTRAINT <nome da constraint> CHECK (<nome do campo> <condição>)
[DISABLE];

### **EXEMPLO**:

ALTER TABLE CLIENTE

ADD CONSTRAINT CK\_GENERO\_CLIENTE CHECK (GENERO IN('M', 'F'));

Removendo a constraint check

Sintaxe:

ALTER TABLE <nome da tabela>
DROP CONSTRAINT <nome da constraint>;

Desativando a constraint check

Sintaxe:

ALTER TABLE <nome da tabela>
DISABLE CONSTRAINT <nome da constraint>;

Ativando a constraint check

Sintaxe:

ALTER TABLE <nome da tabela>
ENABLE CONSTRAINT <nome da constraint>;

# **OPÇÃO DEFAULT**

- A opção DEFAULT permite atribuir um valor por omissão a uma coluna da tabela. Isto significa que, se uma linha for inserida sem valor nessa coluna, será atribuído o valor definido em DEFAULT. Isto impede a ocorrência de valores NULL ou a ocorrência de erros, caso tenha sido especificado NOT NULL.
- Esta opção pode ser usada no comando CREATE TABLE e no comando ALTER TABLE.
- Os valores DEFAULT deverão ser do mesmo tipo de dados que a coluna e poderão ser obtidos por uma expressão, constante ou função (por exemplo SYSDATE ou USER).

## **OPÇÃO DEFAULT**

No exemplo abaixo, se não for inserido um valor para a coluna ATIVO, a base de dados assume 'SIM'.

```
CREATE TABLE CLIENTE (COD_CLIENTE DECIMAL(38), NOME VARCHAR2(70) NOT NULL, ATIVO CHAR(3) DEFAULT 'SIM' NOT NULL );
```

# **OPÇÃO DEFAULT**

Se a tabela fosse criada sem a opção DEFAULT na coluna ATIVO poderíamos usar o comando abaixo para adicionar esta opção:

ALTER TABLE CLIENTE MODIFY (ATIVO DEFAULT 'SIM');

Para remover a opção DEFAULT previamente definida usamos o comando abaixo:

ALTER TABLE CLIENTE MODIFY (ATIVO DEFAULT NULL);