

Banco De Dados

Conteúdo

1. **Comentários**
2. **Componentes da Linguagem SQL**
3. **DDL – Data Definition Language;**
4. **Criando Tabelas;**
5. **Criando Relações;**
6. **Criando Relações PK;**
7. **Criando Relações FK**
8. **Atividade Prática**
9. **Referências**

1. Comentários

A utilização de comentários em sentenças SQL faz-se utilizando a combinação ***/**** e ****/*** para delimitar **uma ou mais linhas** de texto considerado como um comentário.

***/* comentário
de múltiplas linhas */***

2. Componentes da Linguagem SQL

A Linguagem SQL é dividida nos seguintes componentes:

- 1) **DDL** → Data Definition Language;
- 2) **DML** → Data Manipulation Language;
- 3) **DQL** → Data Query Language;
- 4) **DCL** → Data Control Language

Data Definition Language

3. DDL –

Data Definition Language

Um esquema de banco de dados é especificado por um conjunto de definições, por meio da chamada Linguagem de Definição de Dados, ou **DDL** (do inglês, **Data Definition Language**).

3. DDL –

Data Definition Language

A compilação de comandos **DDL** é um conjunto de tabelas armazenadas em um arquivo chamado dicionário (ou diretório) de dados.

3. DDL –

Data Definition Language

Trata-se de um arquivo que contém metadados (dados sobre dados) e é sempre consultado antes que haja qualquer leitura ou modificação pelo banco de dados.

3. DDL –

Data Definition Language

Os comandos **DDL** permite a criação dos componentes do banco de dados como tabelas, índices etc.

3. DDL –

Data Definition Language

A “Linguagem de definição de dados” abrange comandos para:

Create table, **Alter table** e **Drop table**:

comandos para a definição de esquemas de relações (criação e modificação) e exclusão de relações.

Create domain:

criação de domínio dos valores associados a atributos.

3. DDL –

Data Definition Language

Portanto os principais
comandos **DDL** são:

CREATE TABLE;

ALTER TABLE;

DROP TABLE;

CREATE INDEX;

ALTER INDEX;

DROP INDEX;

4. Criando Tabelas (1/8)

Definição de Esquema em SQL:

```
create table r  
(A1D1, A2D2, ..., AnDn,  
<regras de integridade1>  
...,  
<regras de integridadek>)
```

onde:

- a) *r* é o nome da relação;
- b) cada *A*_{*i*} é o nome de um atributo no esquema da relação *r*;
- c) *D*_{*i*} é o tipo de domínio dos valores no domínio dos atributos *A*_{*i*}.

4. Criando Tabelas (2/8)

As regras de integridade (***constraint***) permitidas:

- ***primary key*** (chave primária)
- ***foreing key*** (chave estrangeira)

4. Criando Tabelas (3/8)

```
create table depto (  
    cdDepto integer not null,  
    nmDepto varchar(35),  
    ramal integer,  
    constraint pk_depto primary  
key (cdDepto)  
)
```

4. Criando Tabelas (4/8)

```
create table cargo (  
    cdCargo      integer not null,  
    nmCargo      varchar(35),  
    vrSalario    int(11),  
  
    constraint pk_cargo primary  
key (cdCargo)  
)
```

4. Criando Tabelas (5/8)

```
create table func (  
    nrMatric    integer not null,  
    nmFunc      varchar(35),  
    dtAdm       date,  
    sexo        char,  
    cdCargo     integer,  
    cdDepto     integer,  
constraint pk_func primary key(nrMatric),  
constraint fk_func_cdCargo foreign key(cdCargo)  
    references cargo(cdCargo),  
constraint fk_func_cdDepto foreign key(cdDepto)  
    references depto(cdDepto)  
)
```


4. Criando Tabelas (6/8)

```
create table produto (  
    cdProduto      integer not null,  
    nomeProduto    varchar(35),  
    constraint pk_produto primary  
key (cdProduto)  
)
```

4. Criando Tabelas (7/8)

```
create table cliente (  
    cdCliente      integer not null,  
    nomeCliente    varchar(35),  
    constraint pk_cliente primary  
key (cdCliente)  
)
```

4. Criando Tabelas (8/8)

```
create table ficha (  
    cdCliente      integer not null,  
    cdProduto      integer not null,  
    saldo          integer,  
constraint pk_ficha primary key(cdCliente, cdProduto),  
  
constraint fk_ficha_cdCliente foreign key(cdCliente)  
    references cliente(cdCliente),  
constraint fk_ficha_cdProduto foreign key(cdProduto)  
    references produto(cdProduto)  
)
```

Criando Relações

Data

Definition

Language

5. Criando Relações

Os comandos **DDL** servem para:

- a) definir esquemas;
- b) remover relações;
- c) criar índices;
- d) e modificar esquemas de relação.

Criando Relações

Primary Key

PRIMARY KEY – Chave Primária

Chaves primárias são restrições criadas em uma tabela do banco de dados para identificar a linha da tabela.

Elas não permitirão que possam ocorrer valores duplicados, ou seja, que a mesma linha seja inserida novamente.

Isso só ocorrerá caso o valor da chave primária for alterado.

PRIMARY KEY – Chave Primária

Conforme Angelotti (2010, p. 78), “A Primary Key é utilizada para identificar a linha da tabela, é a chave primária, **ela não pode se repetir e também não aceita valor nulo**”.

PRIMARY KEY – Chave Primária

Suas principais características são:

- Não pode ser nula;
- Identificará a linha;
- Terá um índice definido;
- Poderá ser referenciada em outra tabela, a fim de definir integridade referencial.

PRIMARY KEY – Chave Primária

Suas principais características são:

- Pode ser simples (possui um único campo) ou composta (possui dois ou mais campos);
- Se a chave primária é composta, os valores de cada campo podem se repetir, mas não a combinação desses valores.
- Toda **chave primária** criará um objeto chamado **Constraint** (**restrição**).

PRIMARY KEY – Chave Primária

Toda chave primária criará um objeto chamado **Constraint** (restrição).

Essa restrição pode ser criada junto ao comando **CREATE TABLE**.

Caso a tabela já exista e seja necessária a criação da chave primária, deve-se utilizar o comando **ALTER TABLE** com a condição **ADD CONSTRAINT**.

5. Criando Relações (6/14)

PRIMARY KEY – Chave Primária

Toda chave primária criará um objeto chamado **Constraint** (**restrição**).

As sintaxes são assim definidas: sintaxe de criação da chave primária junto ao comando **CREATE TABLE** e **ALTER TABLE** e sintaxe de exclusão de uma chave primária.

5. Criando Relações (7/14)

PRIMARY KEY – Chave Primária

➤ Criação com o comando **CREATE TABLE**.

```
1.  CREATE TABLE table_name (  
2.      column1 datatype,  
3.      column2 datatype,  
4.      column3 datatype,  
5.      column4 datatype,  
6.      .....  
7.      CONSTRAINT key_name PRIMARY KEY  
      (columns1, column2, ....)  
8.  );
```

5. Criando Relações (8/14)

PRIMARY KEY – Chave Primária

➤ Criação com o comando **ALTER TABLE**.

```
1. ALTER TABLE table_name
2. ADD CONSTRAINT key_name PRIMARY KEY
   (columns1, column2, . . . .)
3. ;
```

5. Criando Relações (9/14)

PRIMARY KEY – Chave Primária

➤ Exclusão com o comando **ALTER TABLE**.

```
1. ALTER TABLE table_name  
2. DROP CONSTRAINT key_name;
```

PRIMARY KEY – Chave Primária

Vamos tomar como exemplo a criação da **tabela Pessoa** conforme a imagem no próximo slide.

5. Criando Relações (11/14)

PRIMARY KEY – Chave Primária

Pessoa *

	Nome da Coluna	Tipo de Dados	Permitir Nulos
	PessoaID	int	<input checked="" type="checkbox"/>
	Nome	varchar(255)	<input checked="" type="checkbox"/>
	Endereco	varchar(255)	<input checked="" type="checkbox"/>
	Telefone	varchar(255)	<input checked="" type="checkbox"/>
	cidade	varchar(255)	<input checked="" type="checkbox"/>

Crie a tabela **sem a chave primária** no campo **PessoaID**.

PRIMARY KEY – Chave Primária

Agora vamos utilizar o comando de modificação da tabela **ALTER TABLE**, adicionando a constraint **PRIMARY KEY** na coluna **PessoaID**.

```
ALTER TABLE Pessoa ADD  
CONSTRAINT PK_PESSOA PRIMARY KEY  
(PessoaID) ;
```

PRIMARY KEY – Chave Primária

```
ALTER TABLE Pessoa ADD CONSTRAINT PK_PESSOA PRIMARY  
KEY (PessoaID) ;
```

O comando acima **criará a chave primária** na **tabela Pessoa**, tomando o campo **PessoaID** como **chave primária** da **tabela Pessoa**.

PRIMARY KEY – Chave Primária

```
ALTER TABLE Pessoa ADD CONSTRAINT PK_PESSOA PRIMARY  
KEY (PessoaID) ;
```

Deve-se atentar-se para que o campo **PessoaID** tem que ser declarado com **NOT NULL**, pois uma das condições da chave primária é que o campo não pode receber valores nulos.

FOREIGN KEY - FK


Chave Estrangeira

FOREIGN KEY – Chave Estrangeira


Esta restrição **é utilizada para manter o relacionamento entre duas tabelas em um banco de dados.**

Esse relacionamento é imposto através de campos contidos nas duas tabelas.

FOREIGN KEY – Chave Estrangeira

A decorative graphic consisting of three overlapping, curved, grey shapes on the left side of the text box.

Chave estrangeira garante um dos principais conceitos do banco de dados que é a integridade referencial, onde um campo primário de uma tabela Pai se relaciona com um outro campo de uma tabela Filho.

A decorative graphic consisting of three overlapping, curved, grey shapes on the right side of the text box.

FOREIGN KEY – Chave Estrangeira

Algumas de suas características são:

- O campo de relação da tabela Pai deverá ser a chave primária da tabela Pai.
- As colunas de relação, necessariamente, precisam ser do mesmo tipo.
- Uma chave estrangeira poderá compor uma chave primária.

FOREIGN KEY – Chave Estrangeira

Algumas de suas características são:

- **Nem toda chave estrangeira** será uma chave primária.
- Pode ter o valor nulo.
- **Toda chave estrangeira.** irá criar um objeto chamado **Constraint**

FOREIGN KEY – Chave Estrangeira

A **chave estrangeira** pode ser criada junto ao comando **CREATE TABLE**, ou pelo comando **ALTER TABLE** em tabelas já existentes.

FOREIGN KEY – Chave Estrangeira

Para a exclusão, assim como na chave primária, devemos utilizar o comando **ALTER TABLE** com a condição **DROP CONSTRAINT**.

FOREIGN KEY – Chave Estrangeira

Toda chave estrangeira irá criar um objeto chamado **Constraint** (**restrição**).

Sintaxe para criação de chave
estrangeira em tabela inexistente, junto ao
comando **CREATE TABLE**.

FOREIGN KEY – Chave Estrangeira

Sintaxe para **criação de chave estrangeira** em **tabela inexistente**, junto ao comando **CREATE TABLE**.

```
CREATE TABLE table_name (  
column1 datatype,  
column2 datatype,  
column3 datatype,  
.....  
CONSTRAINT key_name FOREIGN KEY column1  
REFERENCES table_ref (column));
```

FOREIGN KEY – Chave Estrangeira

Sintaxe de **criação de chave estrangeira** em **tabelas já existentes** utilizando o comando **ALTER TABLE**.

```
ALTER TABLE table_name ADD CONSTRAINT  
key_name FOREIGN KEY column REFERENCES  
table_ref (column);
```

FOREIGN KEY – Chave Estrangeira

Sintaxe de **exclusão de uma chave estrangeira**

```
ALTER TABLE table_name DROP CONSTRAINT  
key_name;
```

FOREIGN KEY – Chave Estrangeira

O próximo slide mostra duas tabelas: a **Tabela Estado** e a **Tabela Cidade**.

Para que seja criado um relacionamento entre as duas tabelas, **devemos levar a chave primária da tabela Estado** para se relacionar com a coluna de mesmo tipo de dados da **tabela Cidade**, pois para cada estado temos várias cidades, e para cada cidade temos somente um estado.

6. Criando Relações (12/15)

PRIMARY KEY – Chave Primária

TabCidade *			
	Nome da Coluna	Tipo de Dados	Permitir Nulos
	cidID	int	<input type="checkbox"/>
	cidNome	varchar(200)	<input type="checkbox"/>
	cid_estID	int	<input type="checkbox"/>


TabEstado *			
	Nome da Coluna	Tipo de Dados	Permitir Nulos
	estID	int	<input type="checkbox"/>
	estUF	char(2)	<input type="checkbox"/>

6. Criando Relações (13/15)

PRIMARY KEY – Chave Primária

Para que o relacionamento seja feito entre as duas tabelas, devemos relacionar o campo **estID** da **tabela Estado** com o campo **cid_estID** da **tabela Cidade**, através do comando SQL:

Nome da Coluna	Tipo de Dados	Permitir Nulos
 cidID	int	<input type="checkbox"/>
cidNome	varchar(200)	<input type="checkbox"/>
cid_estID	int	<input type="checkbox"/>

Nome da Coluna	Tipo de Dados	Permitir Nulos
 estID	int	<input type="checkbox"/>
estUF	char(2)	<input type="checkbox"/>

6. Criando Relações (14/15)

PRIMARY KEY – Chave Primária

	Nome da Coluna	Tipo de Dados	Permitir Nulos
🔑	cidID	int	<input type="checkbox"/>
	cidNome	varchar(200)	<input type="checkbox"/>
	cid_estID	int	<input type="checkbox"/>

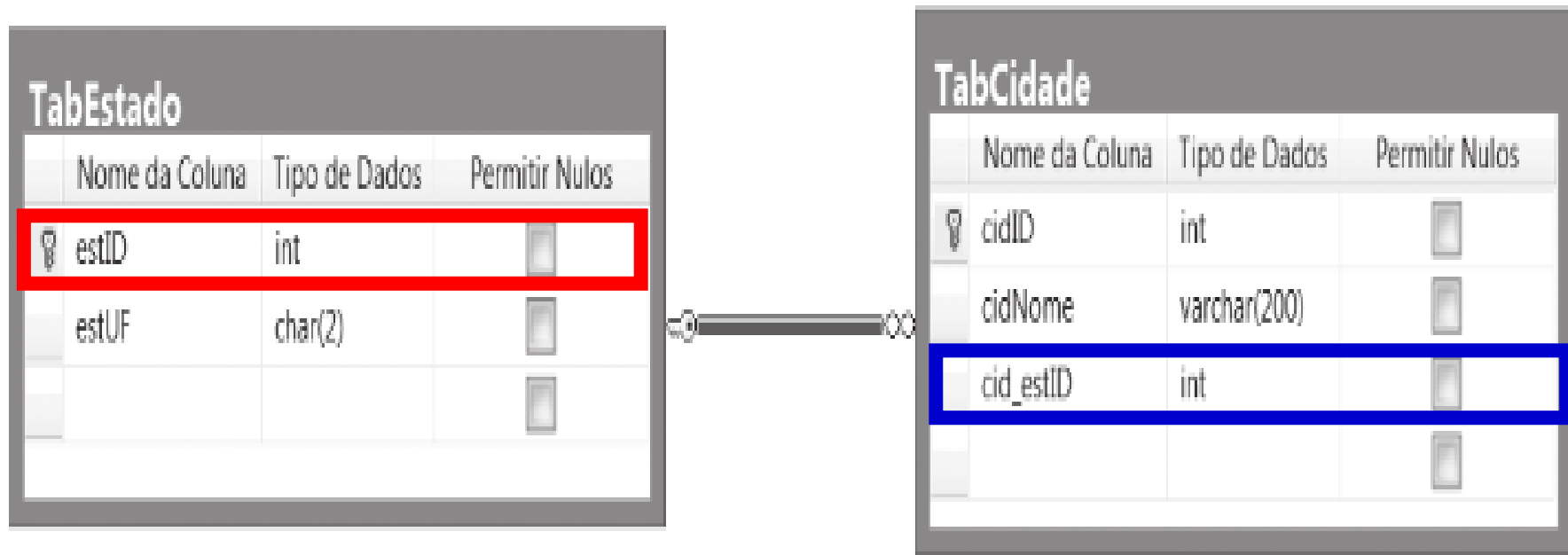
```
ALTER TABLE TabCidade ADD CONSTRAINT  
FK_Estado_Cidade  
FOREIGN KEY (cid_estID) REFERENCES TabEstado  
(estID) ;
```

	Nome da Coluna	Tipo de Dados	Permitir Nulos
🔑	estID	int	<input type="checkbox"/>
	estUF	char(2)	<input type="checkbox"/>

6. Criando Relações (15/15)

PRIMARY KEY – Chave Primária

Após a execução deste comando, o relacionamento **será criado entre as duas tabelas** e o **diagrama ficará da seguinte maneira.**



PRATICANDO o APRENDIDO

Criando Relações

7. Exercício

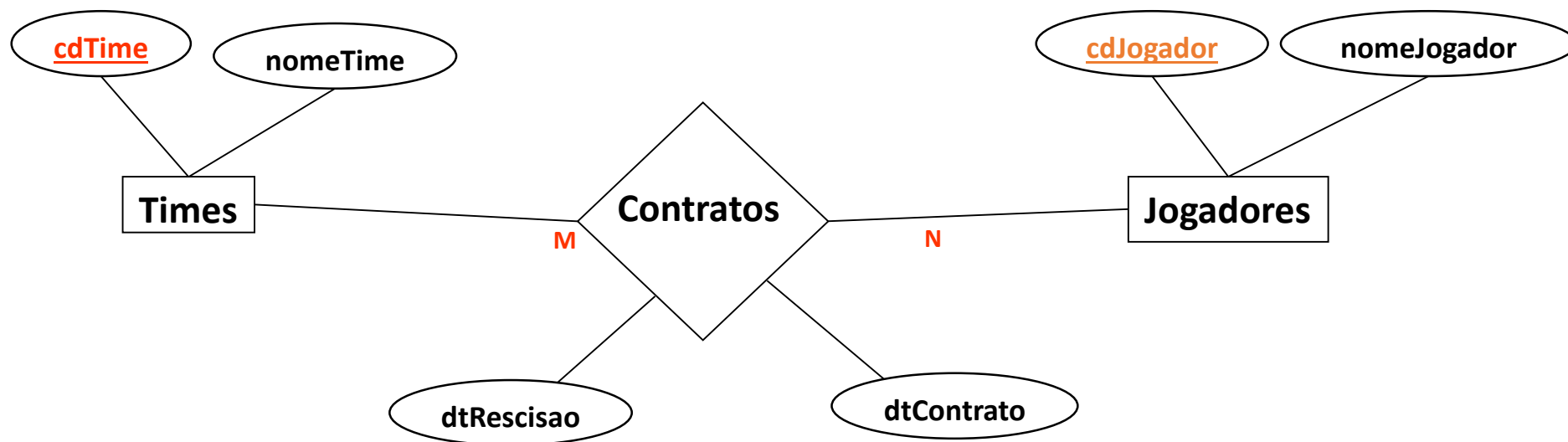
No Próximo slide copie o **M.E.R** e o **M.R**;

Crie um **Banco de Dados** e as:
TABELAS e seus **ATRIBUTOS**

ANTES DE RODAR O SCRIPT SQL
SALVE ELE NUM ARQUIVO BLOCO DE NOTAS

7. Exercício

Modelo Entidade Relacionamento (projeto conceitual)



7. Exercício

Modelo Relacional (projeto lógico)

Times (cdTime, nomeTime)

Jogadores (cdJogador, nomeJogador)

Contratos (cdTime, cdJogador, dtContrato, dtRescisao)

SOLUÇÃO DO EXERCÍCIO

8. SOLUÇÃO - Exercício

SOLUÇÃO - TIMES

8. SOLUÇÃO - Exercício

SOLUÇÃO - JOGADORES

8. SOLUÇÃO - Exercício

SOLUÇÃO - CONTRATOS

9. REFERENCIAS

Banco de Dados I

FERRARETO, Leonardo de Marchi ; NISHIMURA, Roberto Yukio. – Londrina : Editora e Distribuidora Educacional S.A., 2018. 136 p.

Sistema de Banco de Dados.

Abraham Silberschatz; Henry F. Korth; S. Sudarshan.
Capítulo 4: SQL - São Paulo: Makron Books, 3ª ed., 1999.

FIM