Banco de Dados

Aulas Práticas - Laboratório 2

Profª Cristina Verçosa Pérez Barrios de Souza
cristina.souza@pucpr.br

Tópicos

- Base de Dados
- Criando Tabelas
- Povoando Tabelas
- Integridade de dados

Tabelas

- Definição
  - Coleção de dados sobre uma *entidade* específica (pessoal, local, coisa), que possui número discreto de *atributos* nomeados (ex. quantidade, tipo)
  - Geralmente, são relacionadas com outras tabelas

Tipos de Dados para Armazenamento

Dados Temporais		
Tipo	Formato padrão	Valores permitidos
Date	AAAA-MM-DD	1000-01-01 a 9999-12-31
Datetime	AAAA-MM-DD HH:MI:SS	1000-01-01 00:00:00 a 9999-12-31 23:59:00
Timestamp	AAAA-MM-DD HH:MI:SS	1970-01-01 00:00:00 a 2037-12-31 23:59:00
Year	AAAA	1901 a 2155
Time	HHH:MI:SS	-838:59:59 a 838:59:59

Dados de Texto Não-Binário	
Tipo de texto	Numero máximo de bytes
Tinytext	255
Text	65.535
MediumText	16.777.215
LongText	4.294.967.295
Varchar	65.535
Char	255

Dados de Texto Binário	
Tipo de texto	Numero máximo de bytes
Tinyblob	255
Blob	65.535
Mediumblob	16.777.215
Longblob	4.294.967.295
Varbinary	65.535
Binary	255

Dados Numéricos Inteiros		
Tipo	Escopo com sinal	Escopo sem sinal
Tinyint	-128 a 127	0 a 255
Smallint	-32.768 a 32.767	0 a 65.535
Mediumint	-8.388.608 a 8.388.607	0 a 16.777.215
Int	-2.147.483.648 a 2.147.483.647	0 a 4.294.967.295
Bigint	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	0 a 18.446.744.073.709.551.615

Dados Numéricos (Bit e Boolean)	
Tipo	Numero máximo de bytes
bit	1
bool ou boolean	1

Dados Numéricos de Ponto Flutuante e Ponto Fixo	
Tipo	Escopo numérico
Float(p,e)	-3,402823466E+38 a -1,175494351E-38 e de 1.175494351E-38 a 3,402823466E+38
Double(p,e)	-1,7976931348623157E+308 a -2,2250738585072014E-308 e de 2,2250738585072014E-308 a 1.7976931348623157E+308
Decimal(p,e)	-1,7976931348623157E+308 a -2,2250738585072014E-308 e de 2,2250738585072014E-308 a 1.7976931348623157E+308

PRÁTICA FORMATIVA

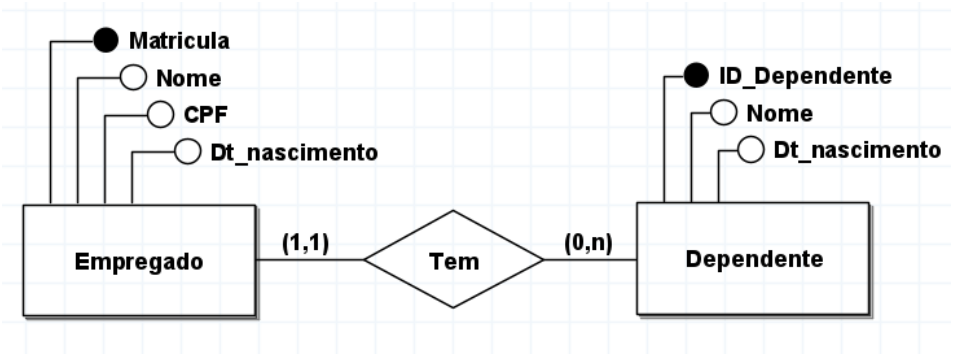
Trabalho em equipe

1. Realize os exercícios indicados.

Formativa: Construa os modelos

› Na ferramenta **brModelo**, construa o seguinte modelo entidade-relacionamento (MER), e seu respectivo modelo lógico (relacional):

Diagrama Conceitual = MER



No modelo conceitual, a chave estrangeira (FK) fica implícita! (a FK não aparece)

Diagrama Lógico = Modelo Relacional
Tipos de Dados das Entidades

Empregado
Matricula: INT
Nome: VARCHAR(100)
CPF: VARCHAR(15)
Dt_nascimento: DATE

Dependente
ID_Dependente: INT
Nome: VARCHAR(100)
Dt_nascimento: DATE

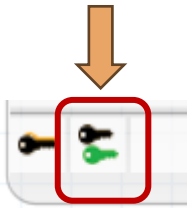
* FK do relacionamento precisa aparecer no diagrama.

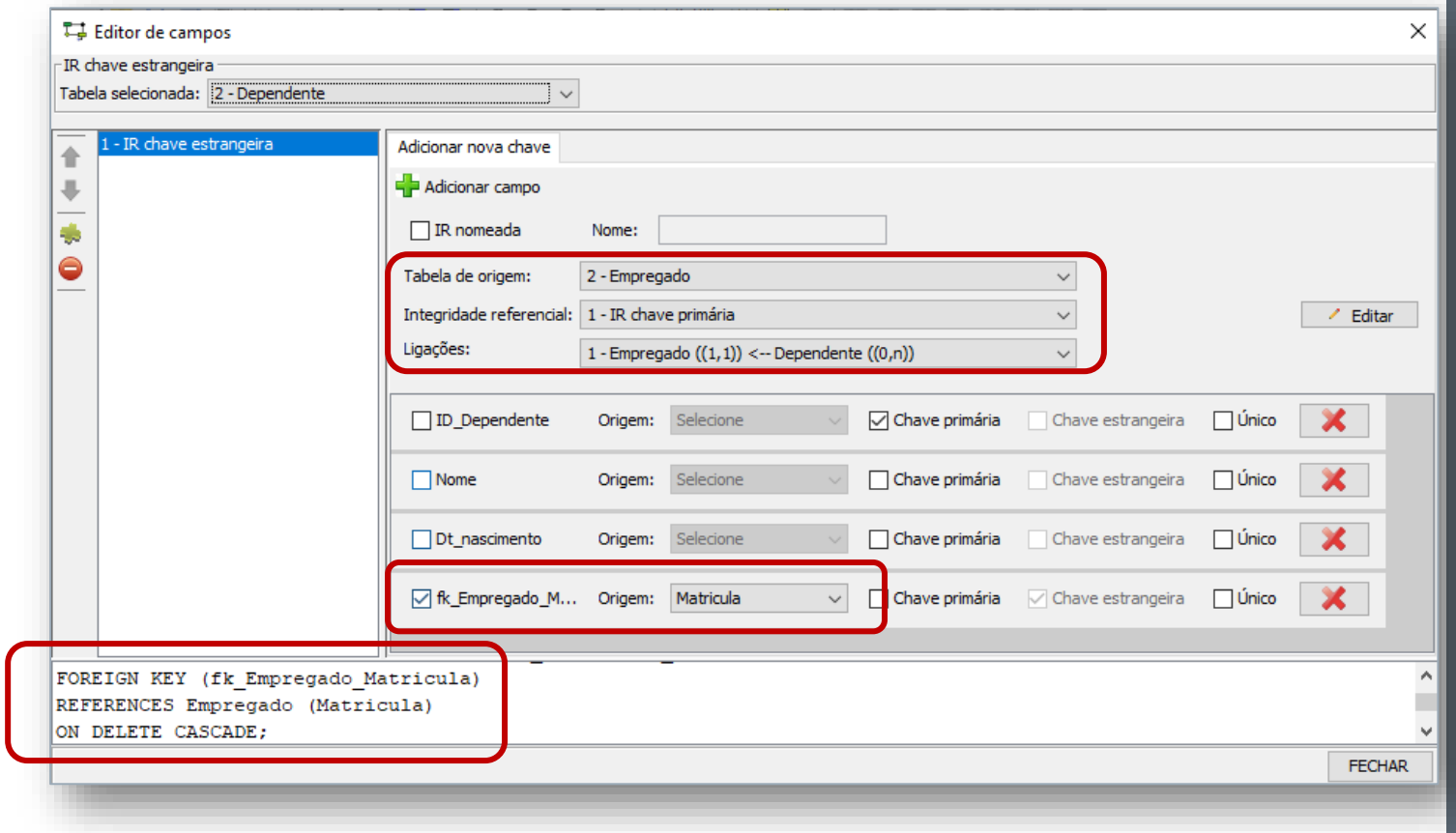
No modelo relacional, a chave estrangeira (FK) fica explícita! (a FK tem que aparecer no modelo)

Formativa: Implemente o modelo lógico

- Na ferramenta **brModelo**, certifique-se que o relacionamento entre **Empregado** e **Dependente** foi construído corretamente:

Dê **duplo clique** na chave estrangeira de **Dependente**:





Formativa: Implemente o modelo lógico

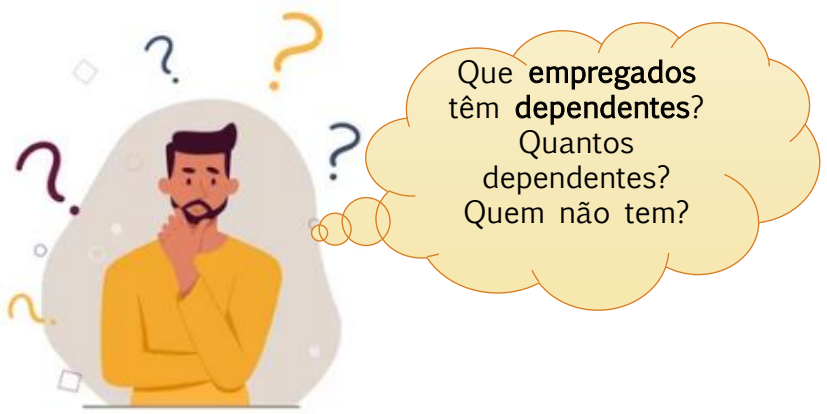
- Na ferramenta **brModelo**, gere o modelo físico (SQL)
- Crie um database **LAB02_Formativa** que irá manter as tabelas **Empregado** e **Dependente**
- Execute o SQL para criar as tabelas indicadas

Modelo Físico (SQL)

```sql
/* Lógico_1: */

CREAT TABLE Empregado (
    Matricula INT PRIMARY KEY,
    Nome VARCHAR(100),
    CPF VARVHAR(15),
    Dt_nascimento DATE
);

CREAT TABLE Dependente (
    ID_Dependente INT PRIMARY KEY,
    Nome VARCHAR(100),
    Dt_nascimento DATE,
    fk_Empregado_Matricula INT
);

ALTE TABLE Dependente ADD CONSTRAINT FK_Dependente_2
    FOREIGN KEY (fk_Empregado_Matricula)
    REFERENCES Empregado (Matricula)
    ON DELETE CASCADE;
```

Formativa: Faça a inserção

- Com as tabelas **Empregado** e **Dependente** construídas no MySQL, faça a inserção dos valores:



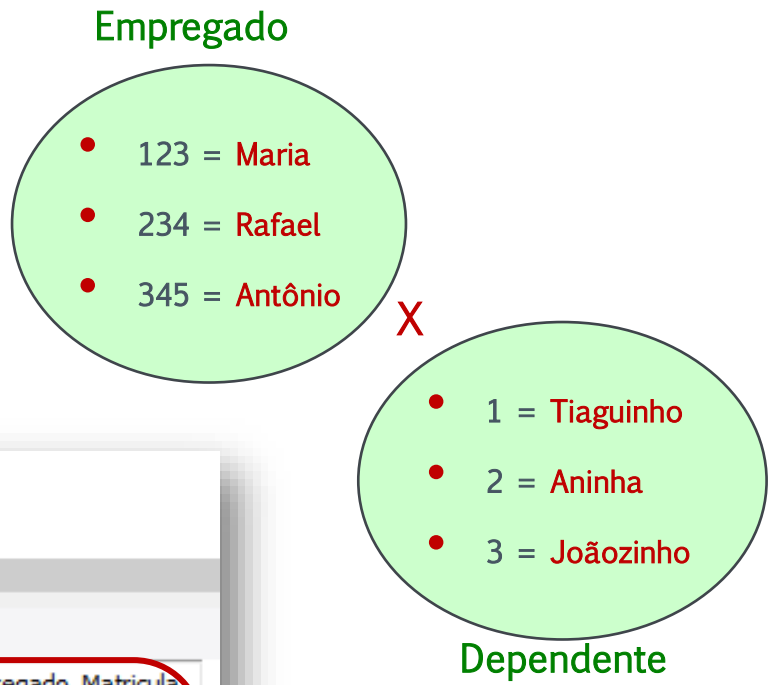
Tabelas preenchidas

Matricula	Nome	CPF	Dt_nascimento
123	Maria Silva	123456789	1990-02-10
234	Rafael Santos	234567890	1985-05-20
345	Antônio Castro	345678901	2000-08-11

ID_Dependente	Nome	Dt_nascimento	fk_Empregado_Matricula
1	Tiaguinho	2020-10-12	123
2	Aninha	2018-05-03	123
3	Joãozinho	2015-12-01	234

Formativa: Faça consultas

- **Produto Cartesiano**
  - Problema: relaciona todos os registos de uma tabela com todos os registros de outra tabela




```
23 •  SELECT * FROM Empregado, Dependente;
24
```

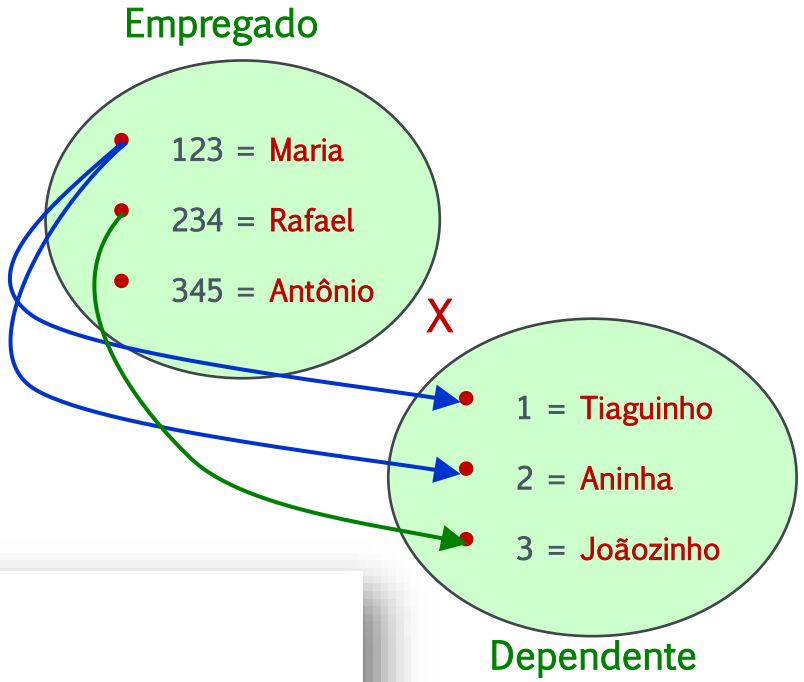
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Matricula	Nome	CPF	Dt_nascimento	ID_Dependente	Nome	Dt_nascimento	fk_Empregado_Matricula
345	Antônio Castro	345678901	2000-08-11	1	Tiaguinho	2020-10-12	123
234	Rafael Santos	234567890	1985-05-20	1	Tiaguinho	2020-10-12	123
123	Maria Silva	123456789	1990-02-10	1	Tiaguinho	2020-10-12	123
345	Antônio Castro	345678901	2000-08-11	2	Aninha	2018-05-03	123
234	Rafael Santos	234567890	1985-05-20	2	Aninha	2018-05-03	123
123	Maria Silva	123456789	1990-02-10	2	Aninha	2018-05-03	123
345	Antônio Castro	345678901	2000-08-11	3	Joãozinho	2015-12-01	234
234	Rafael Santos	234567890	1985-05-20	3	Joãozinho	2015-12-01	234
123	Maria Silva	123456789	1990-02-10	3	Joãozinho	2015-12-01	234

O Produto Cartesiano sozinho mistura todos os dados, criando informação INCORRETA!!!

Formativa: Faça consultas

- **Produto Cartesiano com condição PK x FK**
  - Relaciona corretamente os registos que estão relacionados por suas respectivas **Chave Primária** (PK) e **Chave Estrangeira** (FK)




```
27 •   SELECT *
28     FROM Empregado AS E, Dependente AS D
29     WHERE E.Matricula = D.fk_Empregado_Matricula;
30
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Matricula	Nome	CPF	Dt_nascimento	ID_Dependente	Nome	Dt_nascimento	fk_Empregado_Matricula
123	Maria Silva	123456789	1990-02-10	1	Tiaguinho	2020-10-12	123
123	Maria Silva	123456789	1990-02-10	2	Aninha	2018-05-03	123
234	Rafael Santos	234567890	1985-05-20	3	Joãozinho	2015-12-01	234

Informação CORRETA!!!

Formativa: Faça consultas

- **Produto Cartesiano com condição PK x FK**
  - Podemos indicar quais campos (atributos) são necessários em uma consulta:

```
31 •   SELECT E.Matricula AS 'Matrícula Empregado', E.Nome AS Empregado,
32            D.Nome AS Dependente,
33            TIMESTAMPDIFF(YEAR, D.dt_nascimento, CURDATE()) AS 'Idade do Dependente'
34     FROM Empregado AS E, Dependente AS D
35     WHERE E.Matricula = D.fk_Empregado_Matricula;
36
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Matrícula Empregado	Empregado	Dependente	Idade do Dependente
123	Maria Silva	Tiaguinho	2
123	Maria Silva	Aninha	5
234	Rafael Santos	Joãozinho	7

Formativa: conclusões

- Nesta formativa, vimos como:
  - Usar o **Produto Cartesiano** para combinar os dados de mais de uma tabela.

    Exemplo: SELECT * FROM Empregado AS E, Dependente AS D

  - Usar **Chave Primária** (**PK**) e **Chave Estrangeira** (**FK**) para selecionar os dados que realmente têm relacionamento.

    Exemplo: SELECT * FROM Empregado AS E, Dependente AS D

    WHERE E.Matricula = D.fk_Empregado_Matricula;

PRÁTICA SOMATIVA

Trabalho em equipe para entrega.

1. Realize os exercícios indicados.
2. Indique qual o número do exercício e salve o resultado (comandos e respectivas imagens da prática realizada) em um arquivo Word.
3. Após todos os exercícios, salve o Word como PDF.
4. Entregue o PDF.

Prática 2.1: Criação de Database

CRIAÇÃO DE DATABASE

› Use o código abaixo:

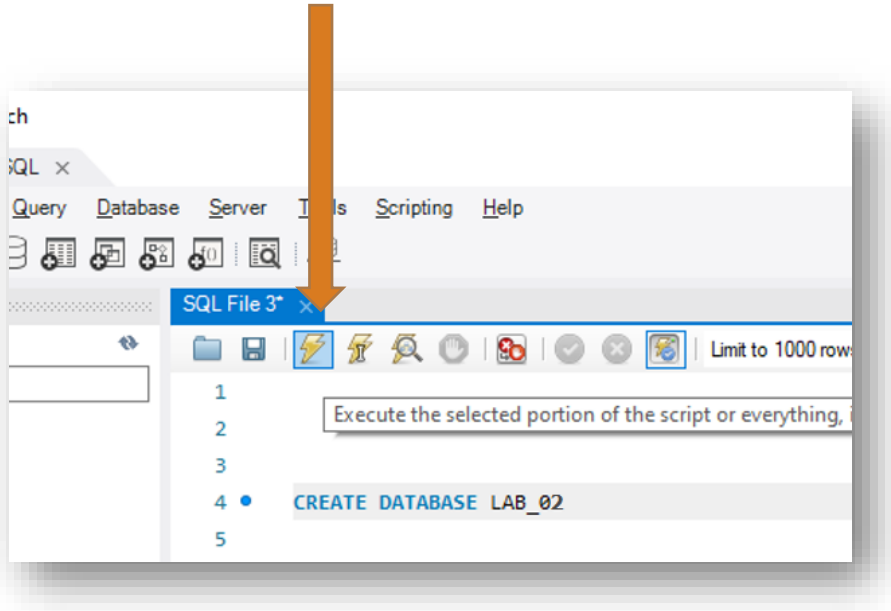
```
CREATE DATABASE LAB_02;

USE LAB_02;
```

2.1 - RESPONDER:

1. Para que serve o comando SQL: **USE**
2. Como devemos usar o comando **DROP** para **eliminar totalmente** o database **LAB_02** criado? Dê um exemplo e execute para verificar seu resultado.
3. Após eliminar o **LAB_02**, crie novamente este database para as próximas práticas.

EXECUTE O CÓDIGO

› Pressione F5 ou clique em Executar, como mostra a Figura:



Prática 2.2: Criando Tabelas

- **Colunas** / **Campos** / **Atributos** “**NULOS**”
  - Todos os tipos de dados podem ser declarados como
  - **NULL** permite entrada nula / vazia.
  - **NOT NULL** não permite entrada nula / vazia.

  - Recomendação: devemos **evitar** permissão **NULL** em colunas
    - O tratamento do NULL acrescenta uma lógica extra ao SGBD, que pode reduzir o desempenho da atualização

Prática 2.2: Criando Tabelas

CRIAÇÃO DE DATABASE

› Execute o código abaixo:

RESPONDER:

1. O que significa quando não indicamos que um campo (atributo) é **NOT NULL**?
2. Para que serve o comando SQL: **ALTER TABLE ... ADD CONSTRAIN**?
3. O que significa a restrição de **PRIMARY KEY**? Para que ela serve na prática?

(a)

```
CREATE TABLE Disciplina
(
id_discip   int          NOT NULL,
nome        varchar(50)  NOT NULL,
ementa      text,
creditos    int          NOT NULL,
periodo     int          NOT NULL
)
```

› Adicionando Chave Primária (PK)

(b)

```
ALTER TABLE Disciplina
ADD CONSTRAINT PRIMARY KEY (ID_discip);
```

Integridade

- **Restrições** (*Constrains*)
  - Forma fácil e poderosa de impor regras nas tabelas de um BD

- ***Constrain Primary Key*** (*restrição de chave primária*)
  - **Chave primária**: uma ou mais colunas que identificam unicamente uma linha/tupla/registro
  - Construída com a restrição **primary key**
  - Modelagem Correta:
    - **Sempre** declarar uma restrição de primary key por tabela
    - Se utilizar Chave Primária Composta, então todas as colunas que fazem parte da chave primária devem ser declaradas como **NOT NULL**

Prática 2.3: Povoando Tabelas

- Adiciona uma ou mais linhas a uma tabela ou exibição no SQL Server.
  - Em sua database de trabalho, execute:

(a)

```
INSERT INTO disciplina VALUES (1, 'Banco de Dados',NULL, 4, 2);

SELECT * FROM disciplina;
```

(b)

```
INSERT INTO disciplina VALUES (1, 'Redes', 'Básico de redes de computadores', 4, 3);

SELECT * FROM disciplina;
```

RESPONDER:

1. Dos comandos passados, que comando não funcionou e como ele foi arrumado?
2. Qual o comando para visualizar as inserções para ver se elas estão corretas?
3. Crie e execute um comando para inserir mais **5 registros / linhas** na tabela **disciplina**.
4. Apresente todos registros inseridos na tabela **disciplina**.

Prática 2.3: Povoando Tabelas

RESPONDER:

1. O que significa **AUTO_INCREMENT PRIMARY KEY** e para que ela serve?
2. O que significa **GENERATED ALWAYS AS (SUBSTRING_INDEX(nome, " ", 1))** e para que ele server.
3. Um **atributo derivado** é salvo em disco? Qual a sua utilidade?

- Criando e povoando nova tabela.
  - Em sua database de trabalho, execute:

(c)

```
CREATE TABLE Professor (
id_prof         INT AUTO_INCREMENT PRIMARY KEY,  -- chave primária auto-incrementada
nome            VARCHAR(50) NOT NULL,
dt_nascimento   DATE,
apelido         VARCHAR(50) GENERATED ALWAYS AS (SUBSTRING_INDEX(nome, " ", 1)) --atributo derivado
)
```

(d)

```
SELECT DATE_FORMAT (curdate(), '%d/%m/%y') AS Data; -- retorna a data atual, formatada dia/mês/ano

INSERT INTO Professor (nome, dt_nascimento)
VALUES ('Maria das Flores', STR_TO_DATE('23/12/1990', '%d/%m/%Y')); -- converte string para data

-- Apresenta os dados da tabela
SELECT * FROM Professor;

SELECT nome, dt_nascimento AS 'Data de Nascimento',
TIMESTAMPDIFF(YEAR, dt_nascimento, CURDATE()) AS Idade
FROM Professor;
```

Prática 2.3: Povoando Tabelas

- Múltiplos INSERTs na tabela.
  - Em sua database de trabalho, execute:

(e)

```
INSERT INTO Professor (nome, dt_nascimento) VALUES
('José da Silva', STR_TO_DATE('20/02/1985', '%d/%m/%Y')),
('Paulo Soares', STR_TO_DATE('10/12/1995', '%d/%m/%Y')),
('Ana Rita', STR_TO_DATE('20/02/2000', '%d/%m/%Y'));

SELECT nome, dt_nascimento AS 'Data de Nascimento’,
TIMESTAMPDIFF(YEAR, dt_nascimento, CURDATE()) AS Idade
FROM Professor;
```

RESPONDER

1. O que significa **STR_TO_DATE('20/02/1985', '%d/%m/%Y’)** e para que este comando foi utilizado?
2. O que significa **AS** e para que server?
3. O que o comando **TIMESTAMPDIFF(YEAR, dt_nascimento, CURDATE())** está realizando?

Integridade

- ***Constrain Foreign Key***
  - Recurso de integridade referencial declarativa
  - **Chave Externa ou Estrangeira**: uma ou mais colunas em uma tabela cujos valores devem ser iguais a uma **primary key**
  - Construída com a restrição **foreign key**
  - Controle:
    - O SGBD restringe a **inserção** ou **modificação** de um registro em uma tabela que é referenciado em outra tabela
- ***Constrain Unique***
  - **Chave Candidata**: coluna exclusiva que pode vir a ser usada como chave primária
  - Construída com a restrição **unique**
  - Na prática:
    - Uma das colunas de valor exclusivo em uma tabela é definida como **primary key,** e
    - As outras colunas exclusivas são definidas como **unique**
- ***Constrain Check***
  - Imposição de **integridade de domínio**: garante que apenas entradas de tipos, valores ou intervalos definidos possam existir para uma determinada **coluna**
  - Construída com a restrição **check**

Prática 2.4: Integridade Referencial

- Criando nova tabela
  - Em sua database de trabalho, execute:

RESPONDER:

1. O que é e para que servem os comandos:
   a) **CHECK (semestre BETWEEN 1 AND 2)**
   b) **UNIQUE (ano, semestre, id_discip, id_prof)**
2. Apresente os modelos **conceitual** (**MER**) e **lógico** (**relacional**) das tabelas **Professor**, **Disciplina** e **Turma**.

(a)

```
CREATE TABLE Turma
(
id_turma    int AUTO_INCREMENT PRIMARY KEY,-- PK auto-incrementada:
ano         int NOT NULL,
semestre    int NOT NULL,
id_discip   int NOT NULL,
id_prof     int NOT NULL,
CONSTRAINT  CK_Sem    CHECK  (semestre BETWEEN 1 AND 2), -- semestre
CONSTRAINT  UN_Ofeta  UNIQUE (ano, semestre, id_discip, id_prof), -- Prof x Disc x Ano x Semestre
CONSTRAINT  FK_Prof   FOREIGN KEY (id_prof)   REFERENCES Professor (id_prof),  -- FK Professor
CONSTRAINT  FK_Discip FOREIGN KEY (id_discip) REFERENCES Disciplina(id_discip) -- FK Disciplina
);

SELECT * FROM Turma;
```

Prática 2.4: Integridade Referencial

- Povoando nova tabela
  - Em sua database de trabalho, execute:

(b)

```
INSERT INTO Turma (ano, semestre, id_prof, id_discip) VALUES
(2020, 1, 2, 2),
(2020, 2, 2, 2),
(2021, 1, 3, 1);

SELECT * FROM Turma;
```

(c)

```
INSERT INTO Turma (ano, semestre, id_prof, id_discip) VALUES
(2020, 1, 2, 2);
```

(d)

```
INSERT INTO Turma (ano, semestre, id_prof, id_discip) VALUES
(2020, 5, 2, 2);
```

RESPONDER:

1. Nos comandos passados, que comando não funcionou e como ele foi arrumado?
2. Apresente todos registros de cada uma das tabelas **Professor**, **Disciplina** e **Turma**.
3. Apenas vendo o conteúdo das tabelas, deduza que professores lecionam quais disciplinas e quando.

Prática 2.4: Integridade Referencial

- Buscando dados nas tabelas relacionadas
  - Em sua database de trabalho, execute:

(a)
```
SELECT *
FROM   Turma, Professor, Disciplina -- produto cartesiano
```

(b)
```
SELECT *
FROM   Turma AS t, Professor AS p, Disciplina AS d    -- prod. Cartesiano
WHERE  t.id_discip = d.id_discip AND p.id_prof = t.id_prof
```

(c)
```
SELECT t.ano, t.semestre, p.nome, d.nome
FROM   Turma AS t, Professor AS p, Disciplina AS d
WHERE  t.id_discip = d.id_discip AND
       p.id_prof = t.id_prof AND
       t.semestre = 1
```

(d)
```
SELECT t.ano, t.semestre, p.nome, d.nome
FROM   Turma AS t, Professor AS p, Disciplina AS d
WHERE  t.id_discip = d.id_discip AND p.id_prof = t.id_prof
ORDER  BY t.ano ASC, t.semestre DESC
```

(e)
```
SELECT p.nome, d.nome, t.ano
FROM   Turma AS t, Professor AS p, Disciplina AS d
WHERE  t.id_discip = d.id_discip AND
       p.id_prof = t.id_prof AND
       p.nome LIKE 'j%'  -- nome começa com j
```

AS: é um ***alias*** (apelido) para uma coluna ou tabela.

WHERE: indica a **condição** que será utilizada para trazer os dados selecionados.

ORDER BY: após o WHERE, indica a ordenação das linhas retornadas.
ASC: ordem ascendente
DESC: ordem descendente

LIKE: determina se uma cadeia de caracteres corresponde a um padrão especificado.
%: máscara para qualquer caractere de qualquer tamanho

RESPONDER:
1. Explique cada um dos comandos passados e seus **resultados** obtidos.
2. Explique as diferenças entre **(a)** e **(b)**

Prática 2.5: Integridade de Valores

- Povoando nova tabela criada com Regras de Negócio (constraints)
  - Em sua database de trabalho, execute:

(a)
```
CREATE TABLE Colaborador (
id_emp     INT         NOT NULL CONSTRAINT PK_emp  PRIMARY KEY CONSTRAINT ID_val CHECK (id_emp BETWEEN 0 AND 1000),
nome       VARCHAR(30) NOT NULL,
salario    FLOAT       NOT NULL CONSTRAINT SL_val  CHECK (salario >= 1000));
```

(b)
```
INSERT INTO Colaborador VALUES
(2000, 'Josué',1500.56);
```

(c)
```
INSERT INTO Colaborador (id_emp, salario) VALUES
(300, 3500.56);
```

(d)
```
INSERT INTO Colaborador VALUES
(400, 'Antônio',350.56);
```

RESPONDER:
1. Nos comandos passados, que comando não funcionou e como ele foi arrumado?
2. Exiba o conteúdo da tabela com as inserções corrigidas

Referência Bibliográfica

- Sistema de Banco de Dados
  - Abraham Silberschatz, Henry F. Korth, S. Sudaarshan

- Referência do SQL
  - Chapter 13 SQL Statements: https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html
  - W3Schools: https://www.w3schools.com/mysql/mysql_drop_db.asp

- Documentação Técnica do MySQL
  - MySQL 8.0 Reference Manual
  - https://dev.mysql.com/doc/refman/8.0/en/