

# Banco de Dados

## Laboratório 2

Prof<sup>a</sup> Cristina Verçosa Pérez Barrios de Souza

[cristina.souza@pucpr.br](mailto: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



**NEW SKILLS**



Loading...

# Prática FORMATIVA

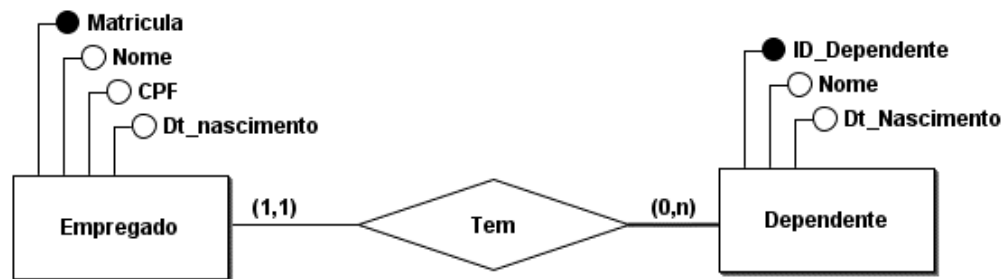
Trabalho INDIVIDUAL: realize os exercícios indicados.



# Formativa: Construa os modelos

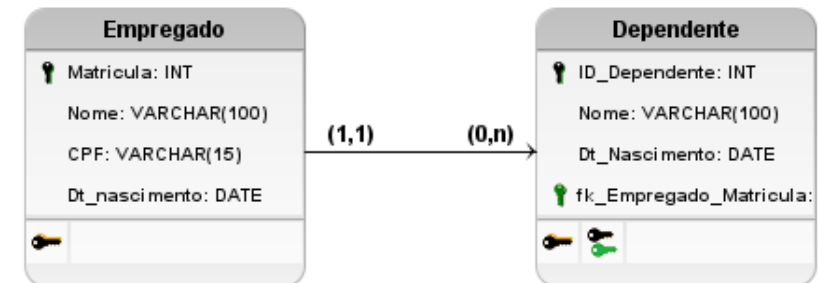
- › Na ferramenta **brModelo**, construa o seguinte **modelo entidade-relacionamento (MER)**, e seu respectivo **modelo lógico (relacional)**:

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



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

\* FK do relacionamento precisa aparecer no diagrama.

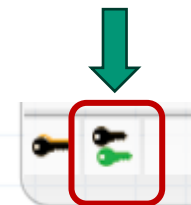


# Formativa: Implemente o modelo lógico

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



Dê duplo clique na chave estrangeira de Dependente:



Editor de campos

IR chave estrangeira

Tabela selecionada: 2 - Dependente

1 - IR chave estrangeira

Adicionar nova chave

+ Adicionar campo

☐ IR nomeada Nome:

Tabela de origem: 2 - Empregado

Integridade referencial: 1 - IR chave primária

Ligações: 1 - Empregado ((1,1)) <-- Dependente ((0,n))

☐ ID\_Dependente Origem: Selecione ☒ Chave primária ☐ Chave estrangeira ☐ Único

☐ Nome Origem: Selecione ☐ Chave primária ☐ Chave estrangeira ☐ Único

☐ Dt\_nascimento Origem: Selecione ☐ Chave primária ☐ Chave estrangeira ☐ Único

☒ fk\_Empregado\_M... Origem: Matricula ☐ Chave primária ☒ Chave estrangeira ☐ Único

FOREIGN KEY (fk\_Empregado\_Matricula)  
REFERENCES Empregado (Matricula)  
ON DELETE CASCADE;

FECHAR



# 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)



```
1. /* Lógico_1: */
2.
3. CREATE TABLE Empregado (
4.     Matricula INT PRIMARY KEY,
5.     Nome VARCHAR(100),
6.     CPF VARCHAR(15),
7.     Dt_nascimento DATE
8. );
9.
10. CREATE TABLE Dependente (
11.     ID_Dependente INT PRIMARY KEY,
12.     Nome VARCHAR(100),
13.     Dt_nascimento DATE,
14.     fk_Empregado_Matricula INT
15. );
16.
17. ALTER TABLE Dependente ADD CONSTRAINT FK_Dependente_2
18. FOREIGN KEY (fk_Empregado_Matricula)
19. REFERENCES Empregado (Matricula)
20. ON DELETE CASCADE;
```





# Formativa: Faça a inserção

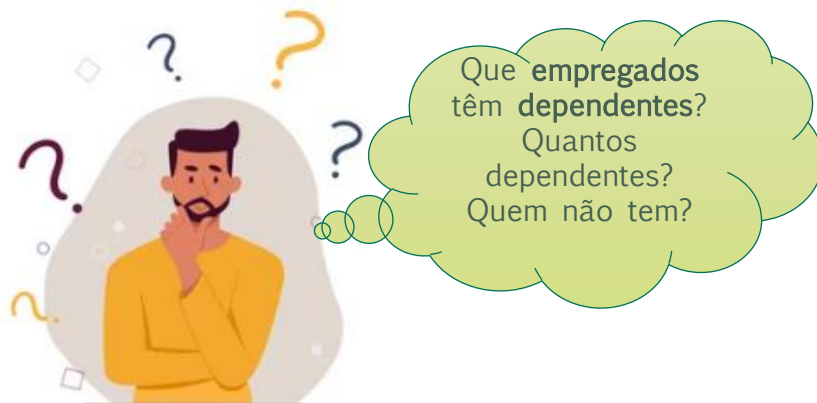
- › Com as tabelas **Empregado** e **Dependente** construídas no MySQL, faça a inserção dos valores:

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 registros de uma tabela com todos os registros de outra tabela

### Empregado

- 123 = Maria
- 234 = Rafael
- 345 = Antônio

X

- 1 = Tiaguinho
- 2 = Aninha
- 3 = Joãozinho

### Dependente

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

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

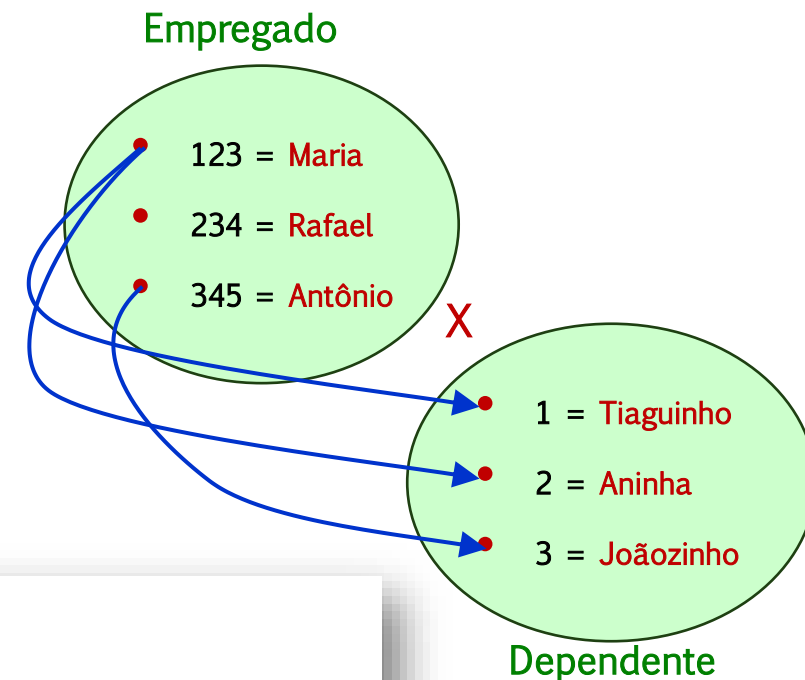
	Matricula	Nome	CPF	Dt_nascimento	ID_Dependente	Nome	Dt_nascimento	fk_Empregado_Matricula
▶	345	Antônio Castro	345678901	2000-08-11	1	Tiaguinho	2020-10-12	123
	234	Rafael Santos	234567890	1985-05-20	1	Tiaguinho	2020-10-12	123
	123	Maria Silva	123456789	1990-02-10	1	Tiaguinho	2020-10-12	123
	345	Antônio 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
```

	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;`



PK



FK



## NEW SKILLS



execute

# Prática SOMATIVA

### Trabalho EM EQUIPE :

1. Realize os **exercícios indicados**.
2. Salve o **número** do exercício e seu **resultado** (**imagens** da prática realizada) em um arquivo **Word**.
3. Após todos os exercícios, **salve os exercícios em arquivo PDF**.
4. Entregue o **PDF**.



## Prática 2.1

### CRIAÇÃO DE DATABASE

› Use o código abaixo:

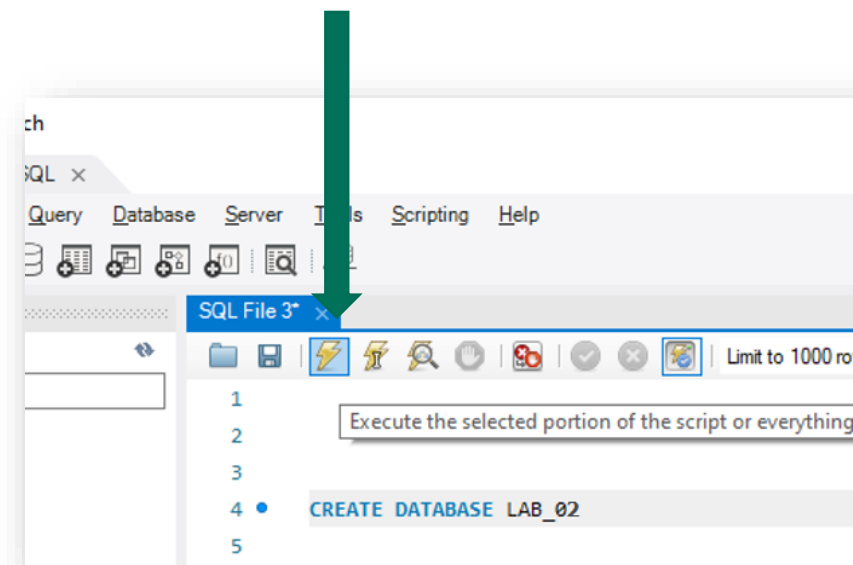
```
CREATE DATABASE LAB_02;  
  
USE LAB_02;
```

#### RESPONDA:

- Para que serve o comando SQL: **USE** ?
- Dê um exemplo de comando **DROP** para **eliminar totalmente** o database **LAB\_02** criado, execute o comando e verifique seu resultado, mostrando a **imagem**.
- 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:





# 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

### CRIAÇÃO DE TABELAS

› Execute o código abaixo:

(A)

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

(B)

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

#### RESPONDA:

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

› Adicionando **Chave Primária (PK)**





# Integridade

## › **Restrições** (*Constraints*)

- 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;
```

#### RESPONDA:

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



## Prática 2.4

- › 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;
```

### RESPONDA:

- a) O que significa **AUTO\_INCREMENT PRIMARY KEY** e para que ela serve?
- b) O que significa **GENERATED ALWAYS AS (SUBSTRING\_INDEX(nome, " ", 1))** e para que serve?
- c) Um **atributo derivado** é salvo em disco?



## Prática 2.5

- › 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;`

### RESPONDA:

- a) O que significa `STR_TO_DATE('20/02/1985', '%d/%m/%Y')` e para que este comando foi utilizado?
- b) O que significa `AS` e para que server? Ele pode ser omitido?
- c) 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 2.6

### › Integridade Referencial

- Em sua **database** de trabalho, execute:

#### RESPONDER:

- O que é e para que servem os comandos:
  - CHECK** (semestre BETWEEN 1 AND 2)
  - UNIQUE** (ano, semestre, id\_discip, id\_prof)
- Apresente em uma **imagem** os modelos conceitual (MER) e lógico (relacional), gerado pelo **brModelo**, 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 apenas aceita valores 1 e 2
  CONSTRAINT UN_Ofeta  UNIQUE (ano, semestre, id_discip, id_prof), -- Prof.Disc.Ano.Sem = exclusivo
  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 2.7

### › 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);
```

#### RESPONDA:

- a) Nos comandos passados, que comando não funcionou e como ele foi arrumado?
- b) Apresente em uma **imagem** todos registros de cada uma das tabelas **Professor**, **Disciplina** e **Turma**.
- c) Apenas vendo o conteúdo das tabelas, escreva que professores lecionam quais disciplinas e quando.





## Prática 2.8

### › Integridade Referencial – Buscando dados nas tabelas relacionadas

(A) `SELECT *`  
`FROM Turma, Professor, Disciplina -- produto cartesiano de 3 tabelas`

(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 -- com PK x Fk`

(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

#### RESPONDA:

- a) Explique cada um dos comandos passados e apresente **imagem** com seus **resultados** obtidos.
- b) Explique as diferenças entre **(A)** e **(B)**
- c) Apresente **imagem** com o **diagrama relacional (modelo lógico)**, gerado por **Engenharia Reversa** das tabelas Turma, Professor e Disciplina.



## Prática 2.9

### › 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 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);
```

#### RESPONDA:

- Nos comandos passados, que comando não funcionou e como ele foi arrumado?
- Exiba a **imagem** do 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](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/>