

# NExT – Nova Experiência de Trabalho

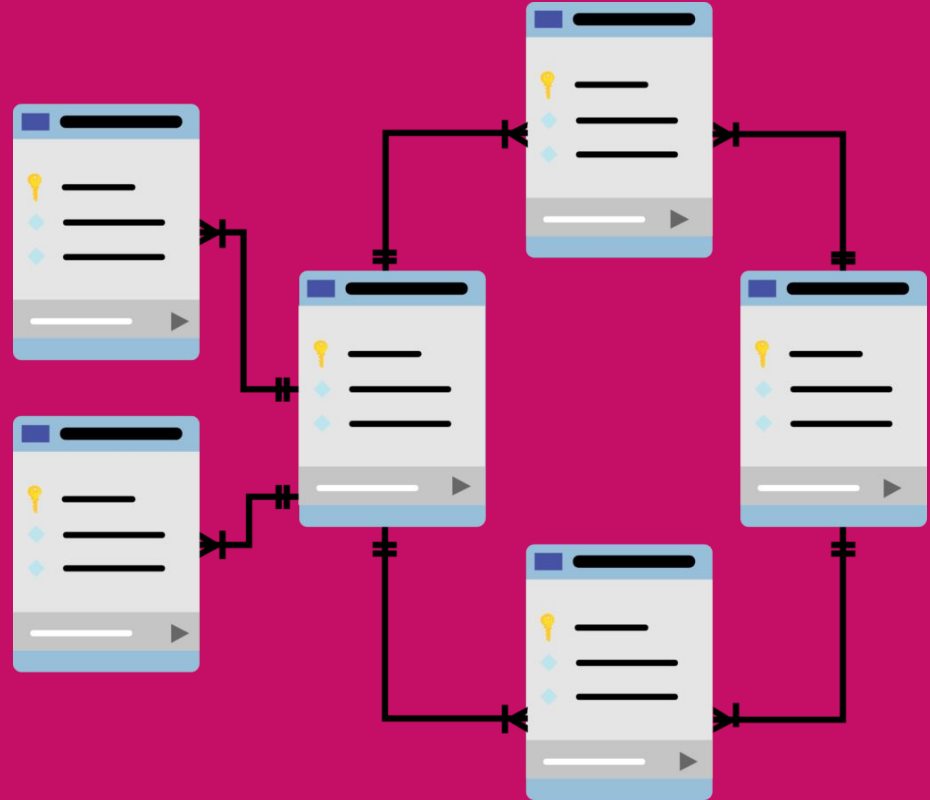
## Módulo – Banco de Dados

Gabrielle K. Canalle  
gkc@cesar.school

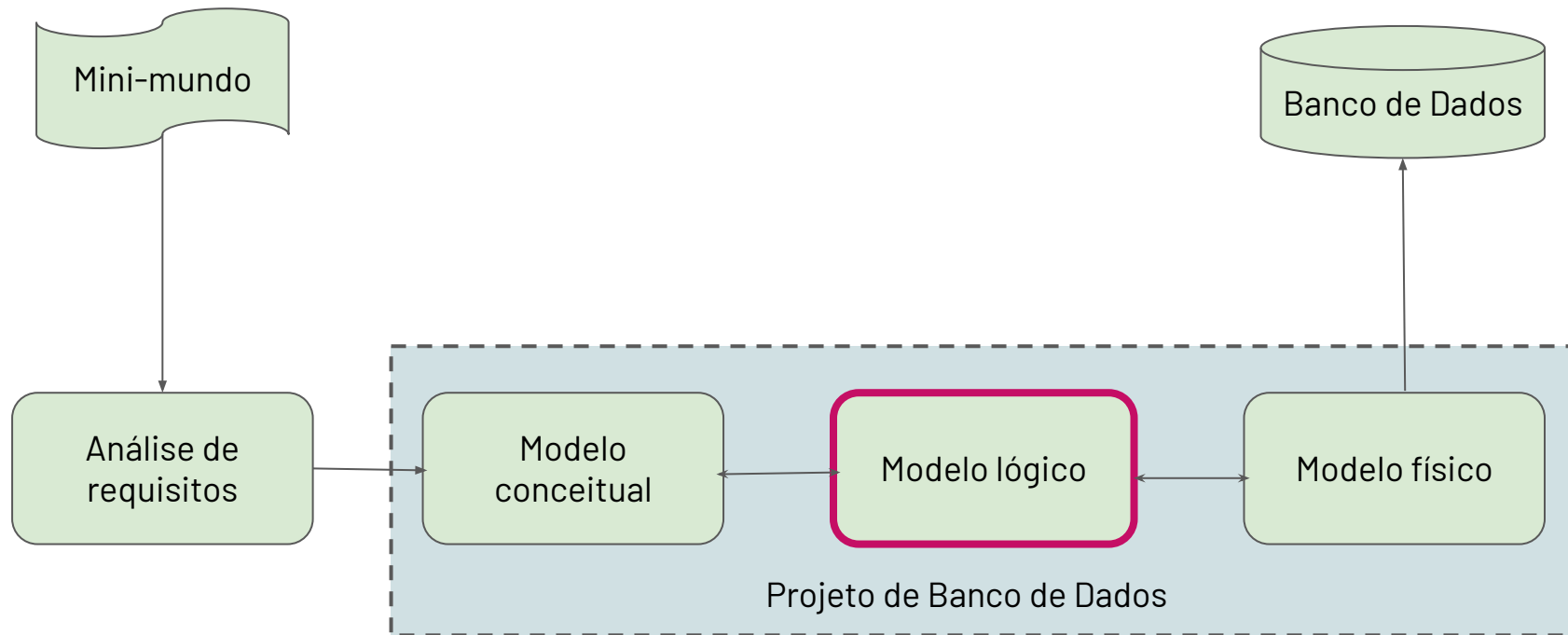
Natacha Targino  
ntrs@cesar.school



# Modelo Lógico



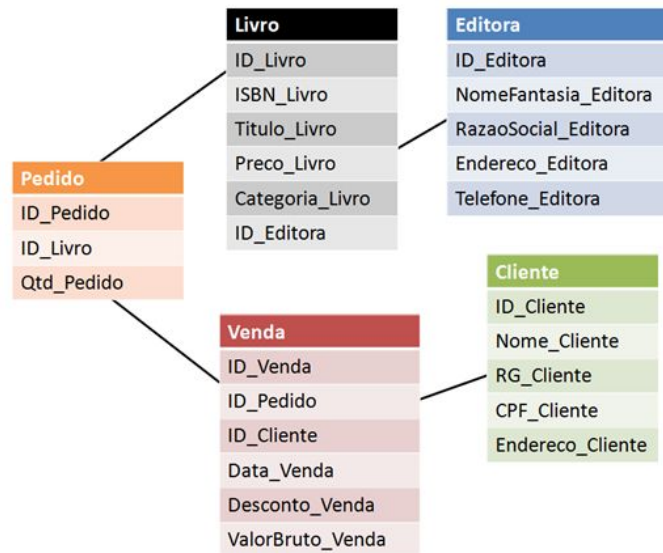
# Projeto de Banco de Dados



# Modelagem Lógica

## Abordagem Relacional

- O Banco de Dados Relacional é composto por tabelas ou relações, com **chaves primárias** e **estrangeiras**.



# Modelagem Lógica

## Tabela Livro

ID_Livro	ISBN	Título	Preço	Categoria	ID_Editora
00101	9786555442410	Ouvir, Agir e Encantar	41,70	Negócios e economia	3245
00102	9788588639171	Sistemas de Banco de Dados	232,99	Computação e Informática	2349
00103	9788576058816	Estruturas de Dados	159,38	Computação e Informática	2783

Chave Primária (PK)

Chave Estrangeira (FK)

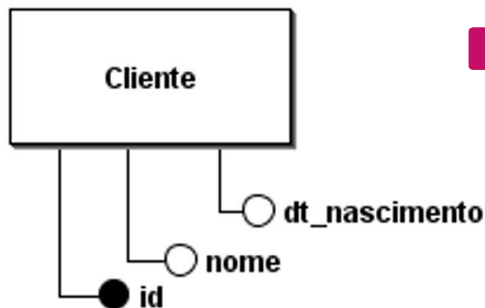
Linha  
ou  
Tupla  
ou  
Registro

# Transformação do Modelo ER para o Relacional

Modelo ER	Modelo Relacional
Entidade	Tabela (Relação)
Instância de Entidade	Linha (Tupla)
Atributo	Coluna (Campo)
Atributo Multivalorado	Tabela Auxiliar
Atributo Identificador	Chave
Atributo Composto	Várias Colunas
Relacionamento	Ligações (ou Tabela)

# Transformação Entidade e Atributos

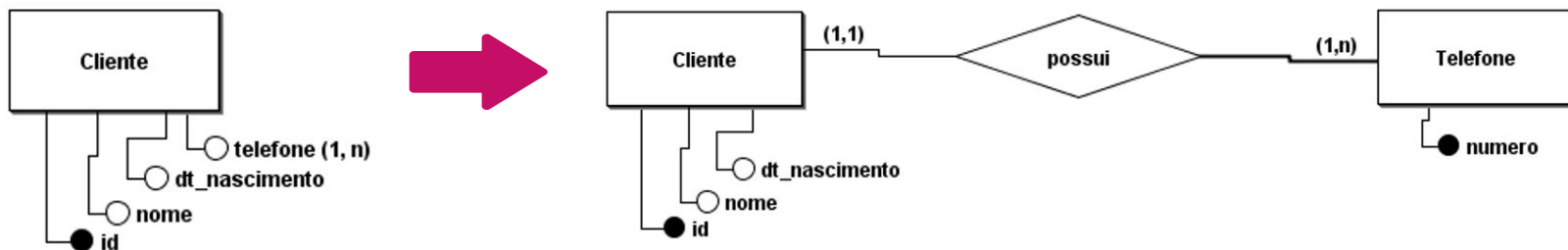
- Cada entidade é traduzida para uma **tabela**.
- Cada atributo (simples) da entidade define uma **coluna** da tabela.
- A coluna correspondente ao atributo identificador é **chave primária**



**Cliente**(id\_cliente, nome, dt\_nasc)

# Transformação Atributos Multivalorados

**1º Solução:** Criar uma tabela para o atributo



**Cliente** (id\_cliente, nome, dt\_nasc)

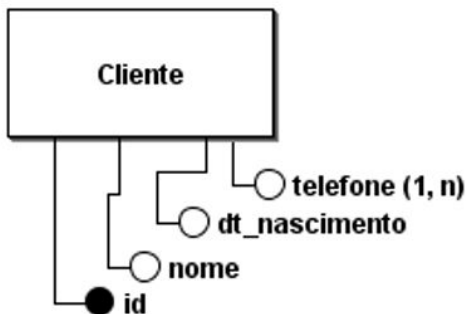
**Telefone** (id\_cliente, id\_numero, numero)

id\_cliente referencia Cliente



# Transformação Atributos Multivalorados

**2º Solução:** Criar colunas para valores do atributo

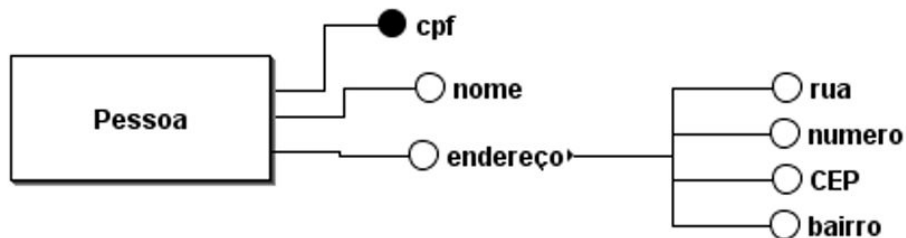


Indicada quando se sabe a quantidade máxima (\*\*E também não permite muitos valores)

**Cliente**(id\_cliente, nome, dt\_nasc, num\_telefone\_01, num\_telefone\_02)

# Transformação Atributos Compostos

## Criar colunas para as partes do atributo



**Pessoa**(cpf, nome, rua, numero, cep, bairro)

# Transformação de Relacionamentos

- Depende da cardinalidade das entidades envolvidas.
- Formas básicas de transformação:
  - **Tabela própria** para o relacionamento
  - **Colunas adicionais** dentro da tabela de entidade
  - **Fusão das entidades** em uma

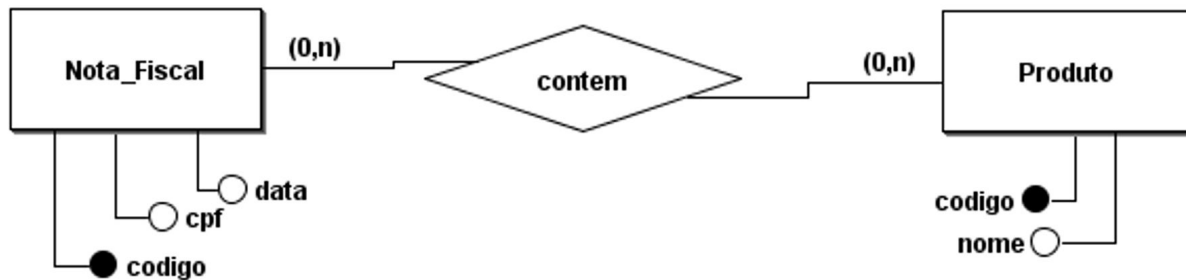
# Transformação de Relacionamentos

## Relacionamento N:N com ou sem Atributos

- **Criar sempre uma tabela própria**, as chaves estrangeiras das tabelas envolvidas formar a chave primária dessa nova tabela
  - Caso o relacionamento tenha **atributos**, são **criadas colunas** para esses atributos nessa nova tabela

# Transformação de Relacionamentos

## Relacionamento N:N SEM Atributos



**NotaFiscal**(cod\_nf, cpf, data)

**Produto**(cod\_pr, nome)

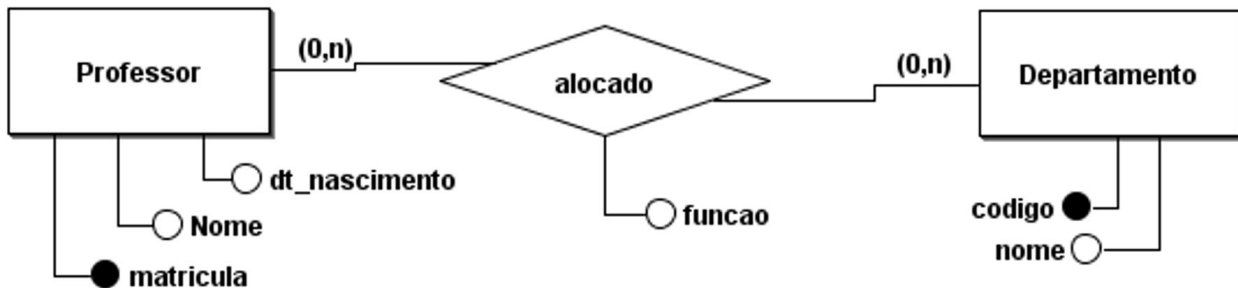
**Contem**(cod\_nf, cod\_pr)

cod\_nf referencia NotaFiscal

cod\_pr referencia Produto

# Transformação de Relacionamentos

## Relacionamento N:N COM Atributos



**Professor** (matricula, nome, dt\_nasc)

**Departamento** (codigo\_d, nome)

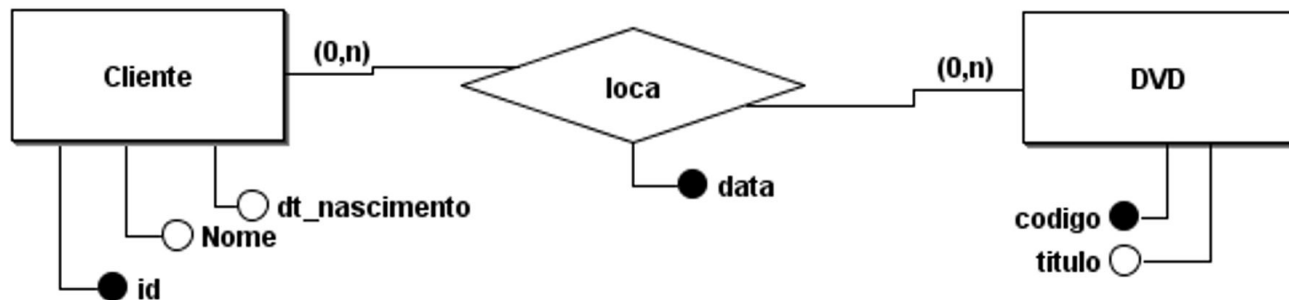
**Alocação** (matricula, codigo\_d, funcao)

matricula referencia Professor

codigo\_d referencia Departamento

# Transformação de Relacionamentos

## Relacionamento N:N COM Atributos



**Cliente** (id, nome, dt\_nasc)

**DVD** (codigo, titulo)

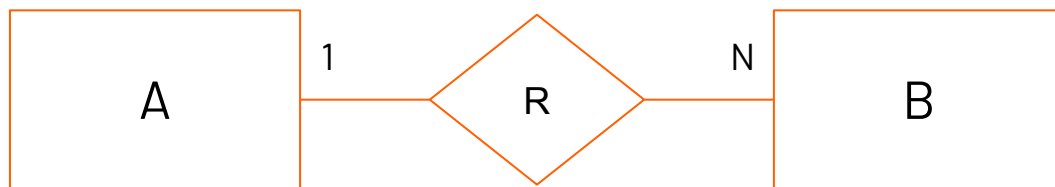
**Locação** (id, cod\_dvd, data)

id referencia Cliente

cod\_dvd referencia DVD(codigo)

# Transformação de Relacionamentos

## Relacionamento 1:N SEM Atributos

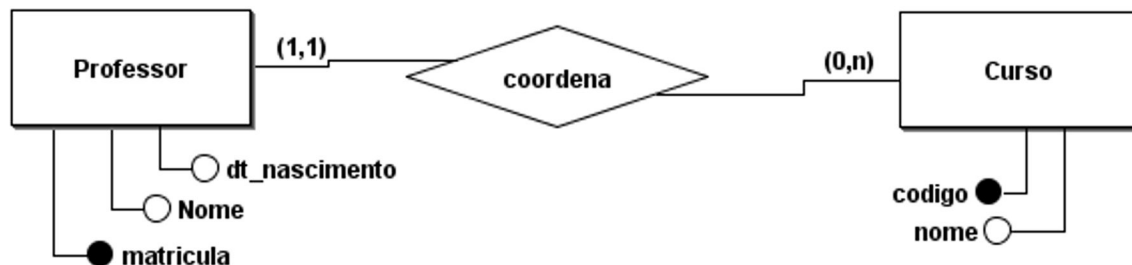


- **Adição de Coluna:** Acrescentar a chave primária da tabela A como chave estrangeira na tabela B
  - OBS.: a chave estrangeira na tabela B poderá ter ou não valor nulo, dependendo da opcionalidade



# Transformação de Relacionamentos

## Relacionamento 1:N SEM Atributos



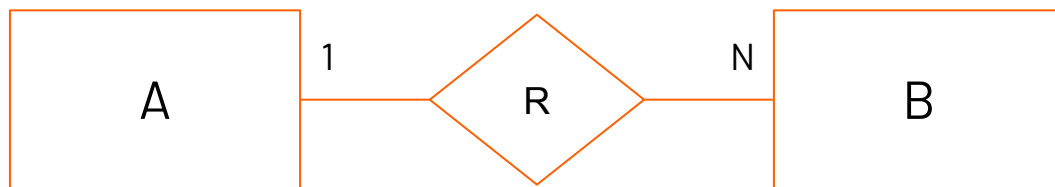
**Professor** (matricula, nome, dt\_nasc)

**Curso** (codigo, nome, coordenador)

coordenador referencia Professor(matricula)

# Transformação de Relacionamentos

## Relacionamento 1:N COM Atributos

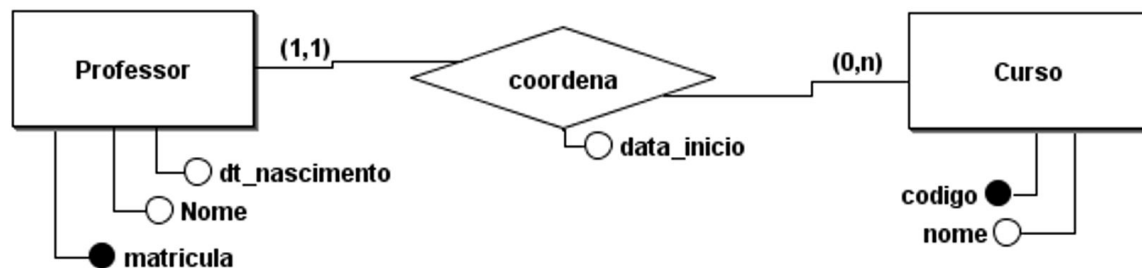


mais  
comum

- **1º opção:** migrar os atributos do relacionamento para a tabela B
- **2º opção:** criar uma nova tabela para o relacionamento

# Transformação de Relacionamentos

## Relacionamento 1:N COM Atributos



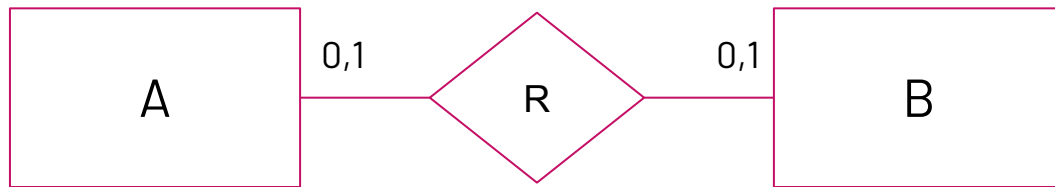
**Professor**(matricula, nome, dt\_nasc)

**Curso**(codigo, nome, coordenador, data\_inicio\_coor)

coordenador referencia Professor(matricula)

# Transformação de Relacionamentos

## Relacionamento 0,1:0,1 SEM Atributos



- **1º opção:** Acrescentar a chave primária da tabela A como chave estrangeira da tabela B
- **2º opção:** Acrescentar a chave primária da tabela B como chave estrangeira da tabela A

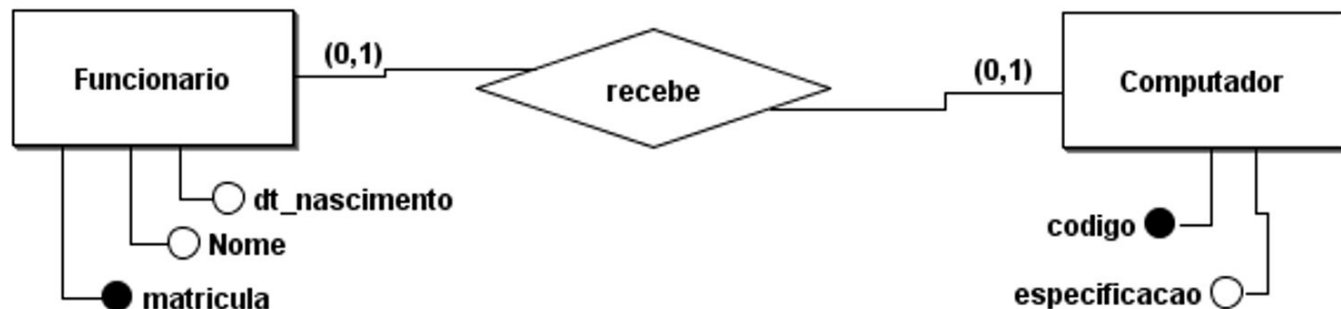
# Transformação de Relacionamentos

## Relacionamento 0,1:0,1 SEM Atributos

- Critérios para Escolha:
  - **1º Ver qual tabela nasce antes:** se A surge primeiro, então, migrar a chave primária de A para B
  - **2º Analisar qual entidade será mais manipulada, em nível de acesso:** se a tabela A será mais manipulada, colocar a chave primária de B nela
  - **3º Para desempate, observar qual a maior chave:** deverá ser migrada a menor

# Transformação de Relacionamentos

## Relacionamento 0,1:0,1 SEM Atributos



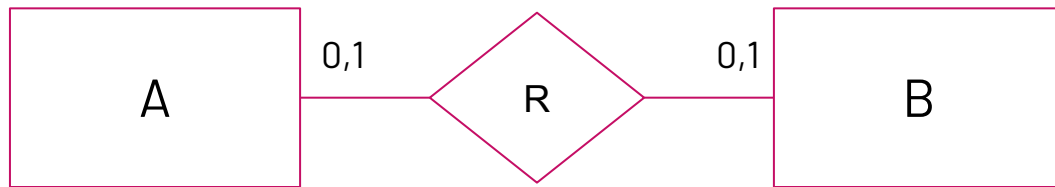
**Computador** (codigo, especificacao)

**Funcionario** (matricula, nome, dt\_nasc, codigo\_comp)

codigo\_comp referencia Computador(codigo)

# Transformação de Relacionamentos

## Relacionamento 0,1:0,1 COM Atributos

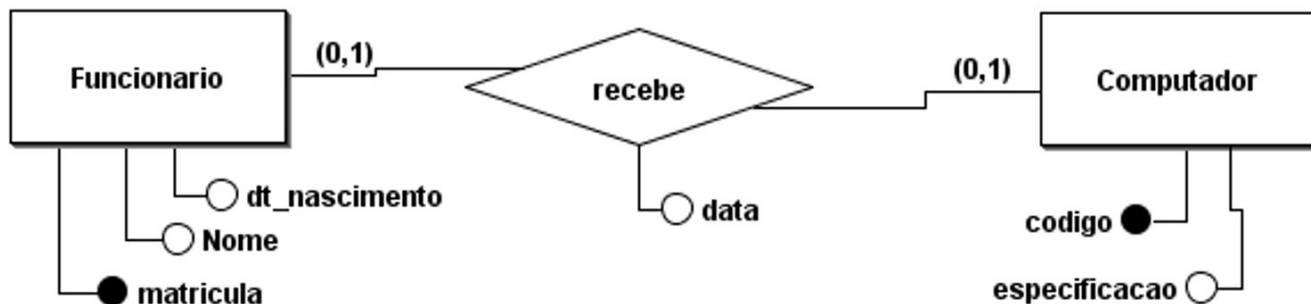


mais  
comum

- **1º opção:** migrar os atributos do relacionamento para uma das tabelas
- **2º opção:** Criar uma nova tabela, agregando as chaves estrangeiras de A e B com os atributos do relacionamento

# Transformação de Relacionamentos

## Relacionamento 0,1:0,1 COM Atributos



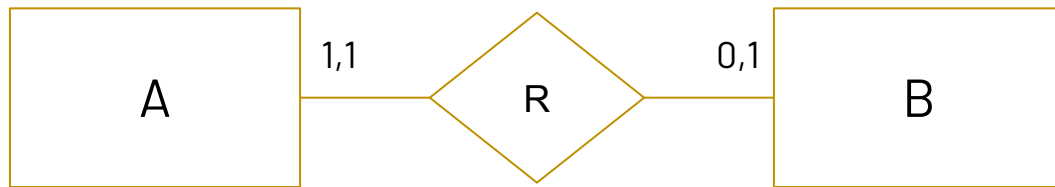
**Computador**(codigo, especificacao)

**Funcionario**(matricula, nome, dt\_nasc, cod\_comp, dt\_comp)  
cod\_comp referencia Computador(codigo)



# Transformação de Relacionamentos

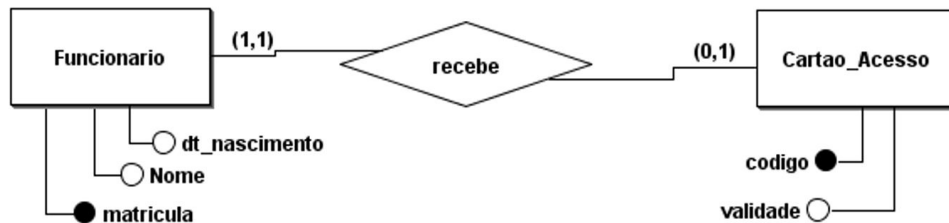
## Relacionamento 1,1:0,1



- **Fusão das tabelas**
- **Adição de Coluna:** Acrescentar a chave primária da tabela A como chave estrangeira da tabela B

# Transformação de Relacionamentos

## Relacionamento 1,1:0,1



- **Esquemas Relacionais possíveis:**

a) **Funcionario**(matricula, nome, dt\_nasc)

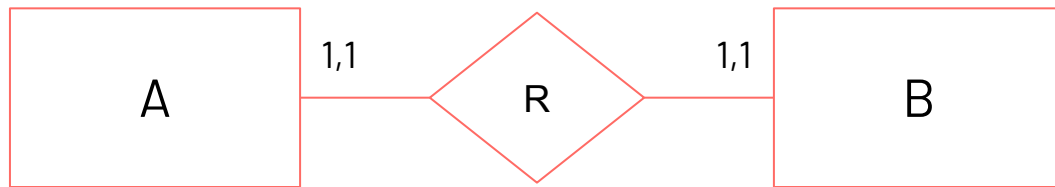
**Cartao\_Acesso**(codigo, validade, matricula)

matricula referencia Funcionario

b) **Funcionario**(matricula, nome, dt\_nasc, codigo\_ct, validade\_ct)

# Transformação de Relacionamentos

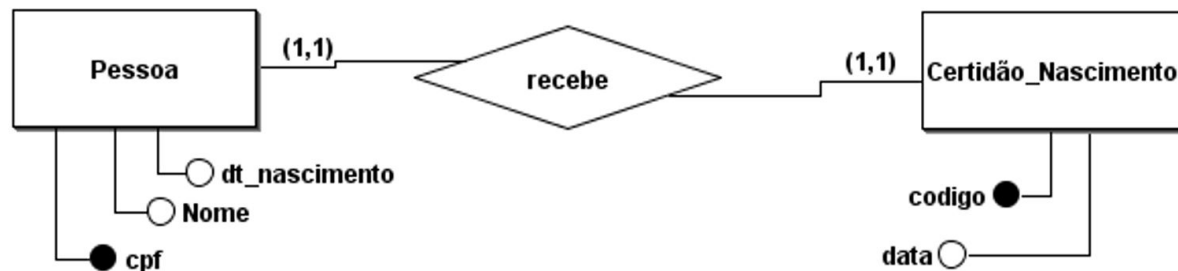
## Relacionamento 1,1:1,1



- **Fusão das tabelas**

# Transformação de Relacionamentos

## Relacionamento 1,1:1,1



**Pessoa**(cpf, nome, dt\_nasc, cod\_certidão, data\_certidão)

# Transformação de Relacionamentos

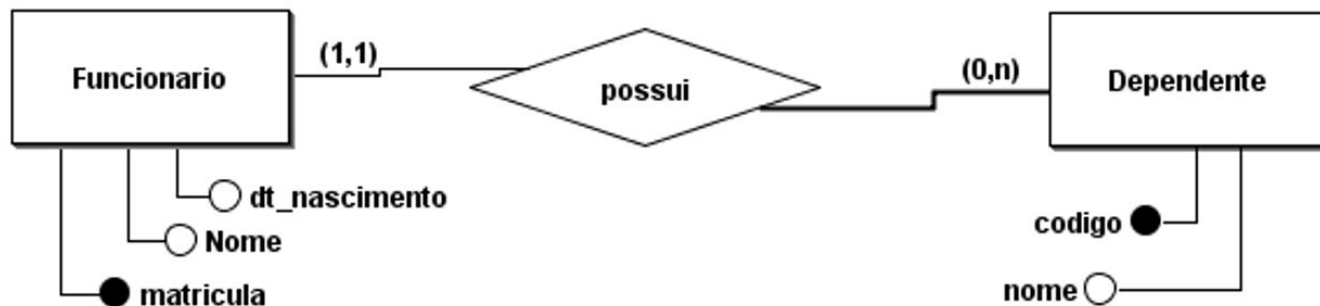
## Resumo

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
<b>Relacionamentos 1:1</b>			
	±	✓	×
	±	±	✓
	±	±	✓
<b>Relacionamentos 1:n</b>			
	±	✓	×
	±	✓	×
	±	✓	×
	±	✓	×
<b>Relacionamentos n:n</b>			
	✓	×	×
	✓	×	×
	✓	×	×

✓ Alternativa preferida  
 ± Pode ser usada – 1ª opção  
 × Não usar  
 ± Pode ser usada – 2ª opção

**Fonte:** Heuser, C. A. Projeto de Banco de Dados (6ª edição)

# Entidade Fraca/Relacionamento Identificador



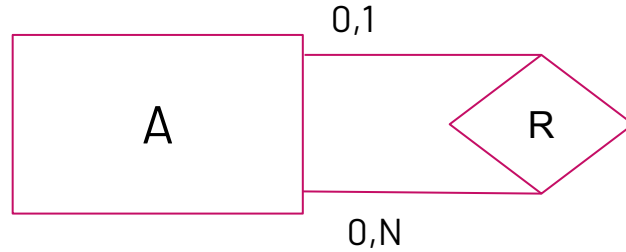
**Funcionario** (matricula, nome, dt\_nasc)


**Dependente** (matricula, codigo, nome)

matricula referencia Funcionario

A chave primária  
da entidade fraca  
é **Composta**

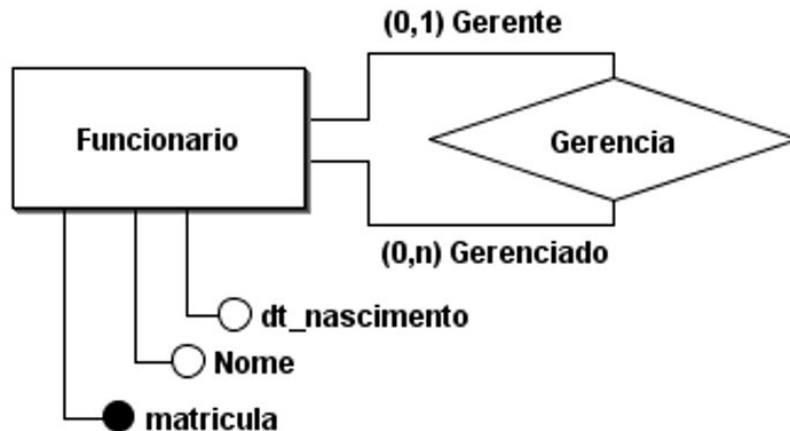
# Derivação de Auto Relacionamentos



- Etapas:
  - **1º:** Considere como se fosse um relacionamento entre duas entidades, ou seja: 
  - **2º:** Considere as regras de derivação, usando no papel das entidades A e B, a mesma entidade A

# Derivação de Auto Relacionamentos

## Auto Relacionamento 1:N

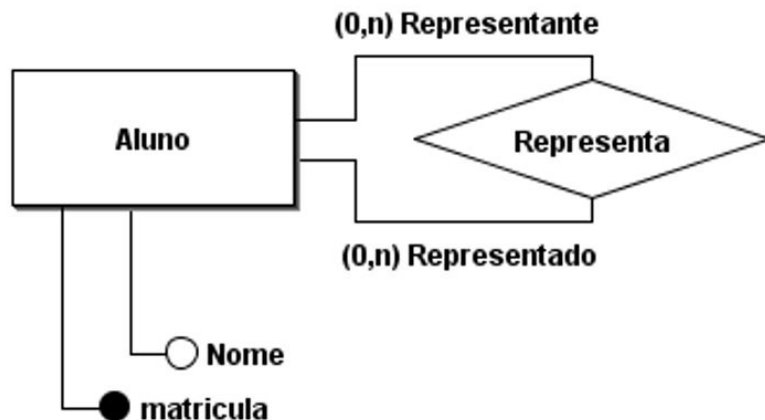


**Funcionario**(matricula, nome, dt\_nasc, mat\_gerente)  
mat\_gerente referencia Funcionario(matricula)



# Derivação de Auto Relacionamentos

## Auto Relacionamento N:N



**Aluno** (matricula, nome)

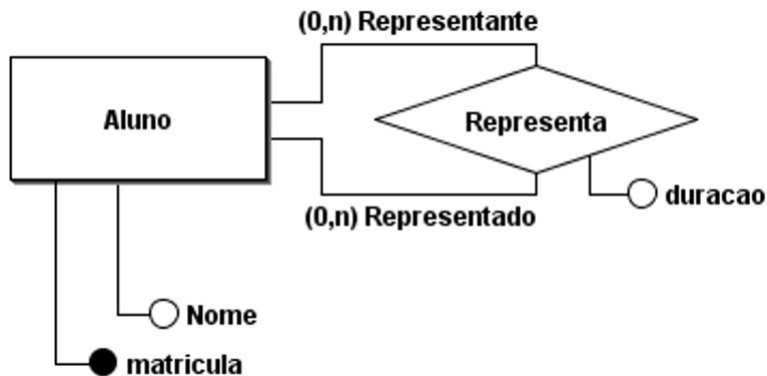
**Representacao** (mat\_representante, mat\_representado)

mat\_representante referencia Aluno(matricula)

mat\_representado referencia Aluno(matricula)

# Derivação de Auto Relacionamentos

## Auto Relacionamento N:N COM atributo



**Aluno** (matricula, nome)

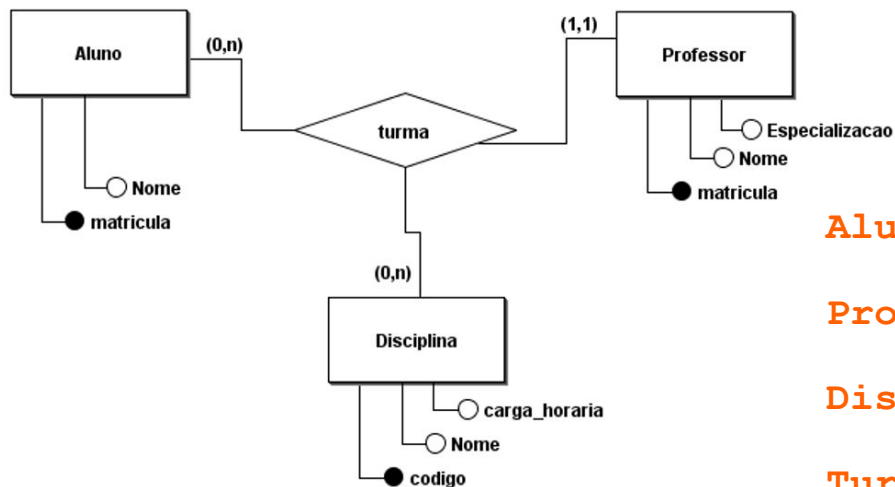
**Representacao** (mat\_representante, mat\_representado, duracao)

mat\_representante referencia Aluno(matricula)

mat\_representado referencia Aluno(matricula)

# Derivação de Relacionamento com grau maior que 2

## Relacionamentos Ternários - N:N:1



Uma tabela é criada para o relacionamento Ternário

**Aluno** (mat\_alu, nome)

**Professor** (mat\_prof, nome, especializacao)

**Disciplina** (codigo, nome, carga\_horaria)

**Turma** (mat\_alu, codigo, mat\_prof)

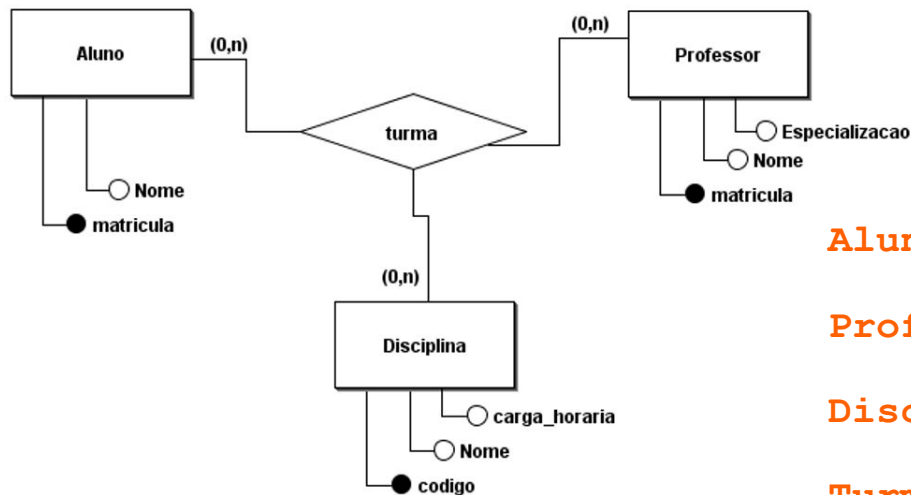
mat\_alu referencia Aluno

codigo referencia Disciplina

mat\_prof referencia Professor

# Derivação de Relacionamento com grau maior que 2

## Relacionamentos Ternários - N:N:N



Uma tabela é criada para o relacionamento Ternário

**Aluno** (mat\_alu, nome)

**Professor** (mat\_prof, nome, especializacao)

**Disciplina** (codigo, nome, carga\_horaria)

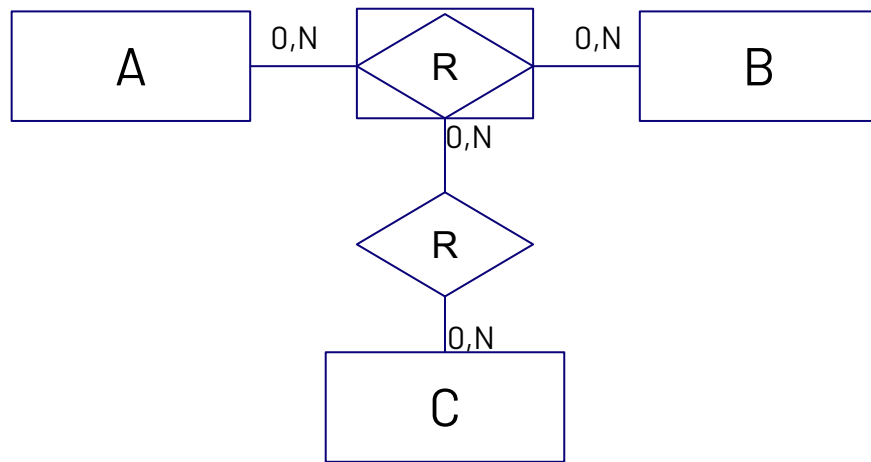
**Turma** (mat\_alu, codigo, mat\_prof)

mat\_alu referencia Aluno

codigo referencia Disciplina

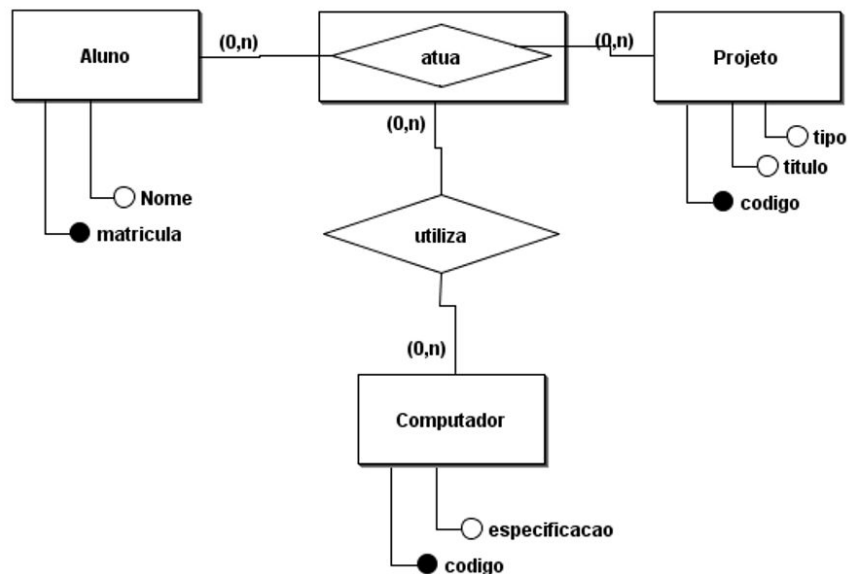
mat\_prof referencia Professor

# Derivação de Agregação/Entidade Associativa



- Etapas:
  - **1º:** Criar uma tabela referente ao relacionamento entre as tabelas A e B
  - **2º:** Analisar o relacionamento dessa tabela criada e a tabela C, para a relação essas duas tabelas considere as regras de derivação vistas

# Derivação de Agregação/Entidade Associativa



**Aluno**(matricula, nome)

**Projeto**(cod\_proj, titulo, tipo)

**Computador**(cod\_comp, especificacao)

**Atuacao**(mat, cod\_proj)

mat referencia Aluno(matricula)

cod\_proj referencia projeto

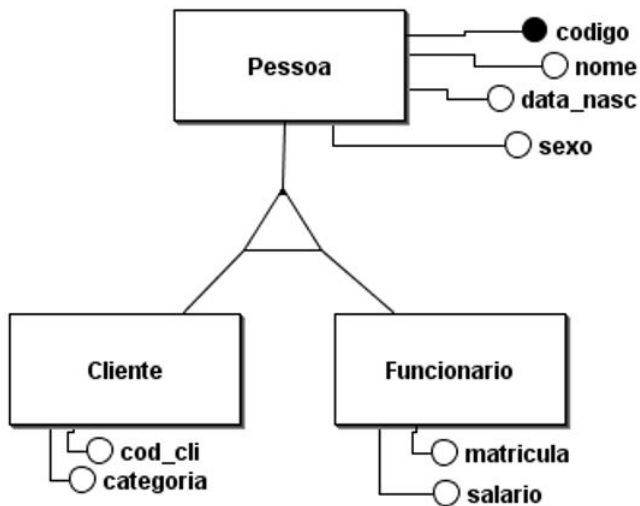
**Uso**(mat, cod\_proj, cod\_comp)

mat,cod\_proj referencia Atuacao

cod\_comp referencia Computador

# Derivação de Estruturas de Generalização/Especialização

- 1ª opção: Criar uma tabela para a entidade generalizada e outra tabela para cada entidade especializada



**Pessoa** (codigo, nome, data\_nasc, sexo)

**Cliente** (codigo, cod\_cli, categoria)

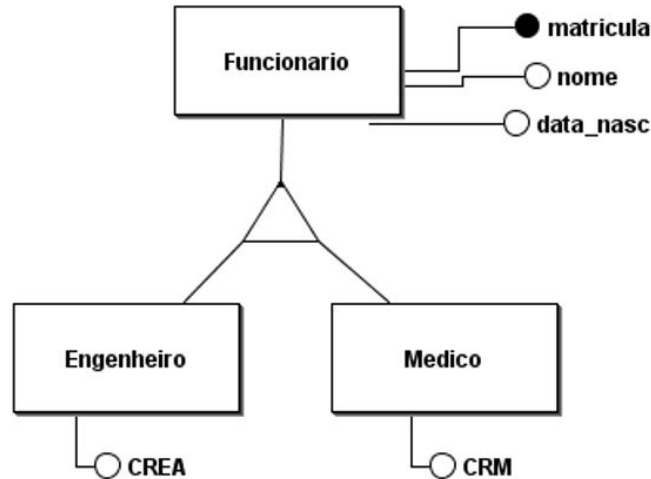
codigo referencia Pessoa

**Funcionario** (codigo, mat, salario)

codigo referencia Pessoa

# Derivação de Estruturas de Generalização/Especialização

- **2ª opção: Criar somente uma tabela para a entidade generalizada e migrar todos os atributos e relacionamentos para essa tabela**

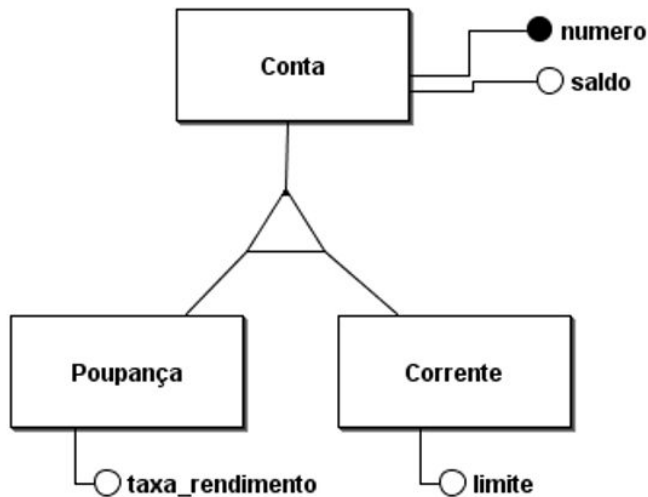


**Funcionario** (matricula, nome, data\_nasc, CREA, CRM)



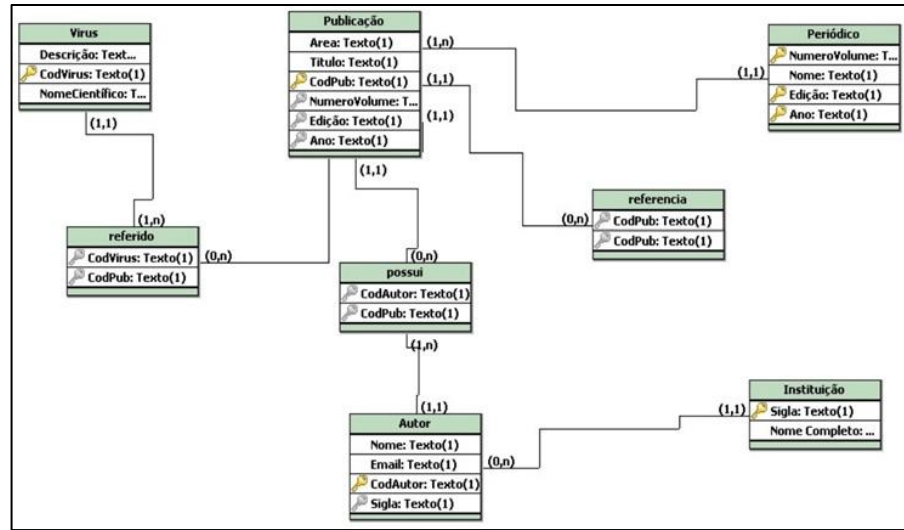
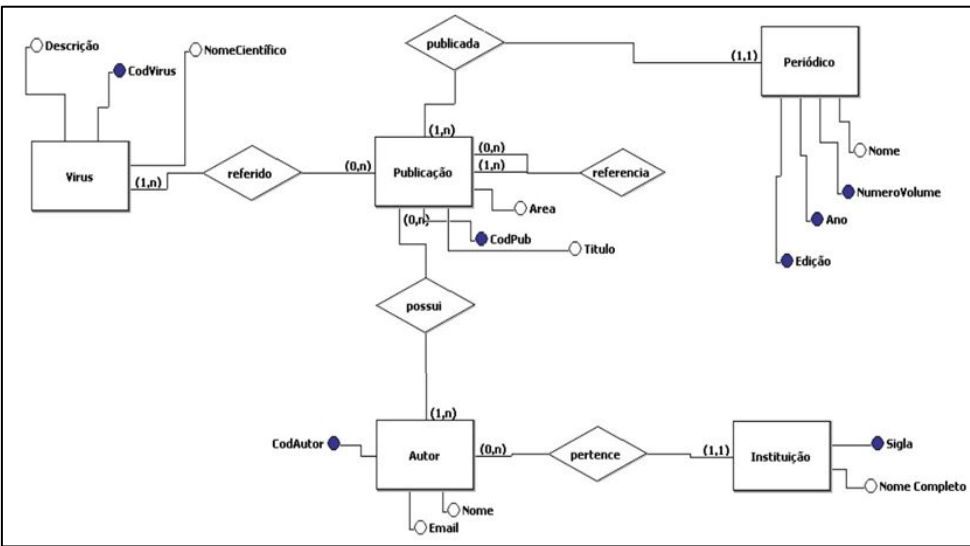
# Derivação de Estruturas de Generalização/Especialização

- **3ª opção: Criar somente tabelas para as entidades especializadas** e migrar todos os atributos e relacionamentos generalizados para cada uma dessas tabelas

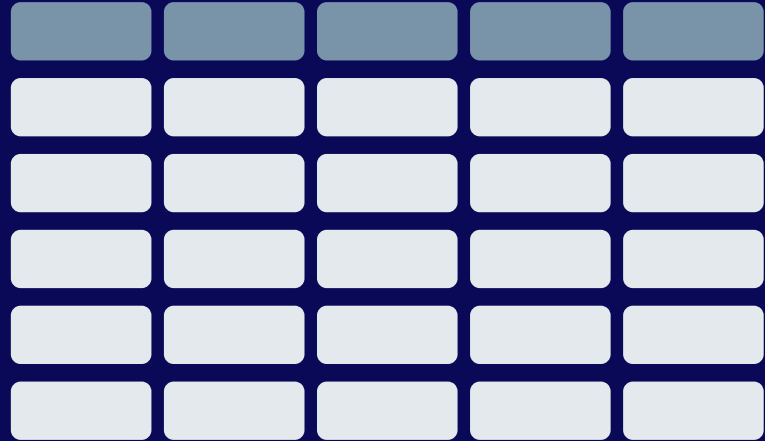


**Conta\_Poupança** (num, saldo, taxa)

**Conta\_Corrente** (num, saldo, limite)



# Normalização



# Normalização

- É o processo de **decomposição** de um esquema de relação em outros esquemas de relação
- Os esquemas resultantes devem **preservar a semântica original** (restrições de integridade, dados e relacionamentos)
- Tabelas normalizadas representam de maneira melhor uma realidade modelada e atendem ao modelo relacional.

# Normalização

- **Objetivos:**

- Eliminar redundâncias
- Minimizar anomalias de inserção, remoção e atualização
- Garantir que as dependências entre os dados façam sentido
- Obtenção de um modelo ER

- **Benefícios:**

- O espaço de armazenamento dos dados diminui
- A descrição do BD será imediata
- A tabela pode ser atualizada com maior eficiência
- A reversão para o modelo conceitual é possível

# Normalização

**1FN:** Uma tabela está na primeira forma normal, quando ela

**NÃO CONTÉM tabelas aninhadas**

(Tabela Aninhada: **Grupo repetido** ou **coluna não atômica** ou **coluna multi-valorada**, que não apresenta valores atômicos)

# Normalização: 1FN

Cod_Disciplina	Nome_Disciplina	Cod_Prof	Nome_Prof	Créditos	Horas
IDC09	Banco de Dados I	IP032	Natacha Targino	06	100
		IP059	Gabrielle Canalle		
IDC12	Banco de Dados II	IP098	João Vicente	04	67
		IP059	Gabrielle Canalle		
		IP044	Vinícius Filho		
IDC07	Programação	IP098	João Vicente	04	67
IDC16	Estrutura de Dados	IP032	Natacha Targino	06	100
		IP029	Mariana Rachel		

**Disciplina** (Cod\_Disciplina, Nome\_Disciplina, (Cod\_Prof, Nome\_Prof),  
Créditos, Horas)

# Normalização: 1FN

Cod_Disciplina	Nome_Disciplina	Créditos	Horas
IDC09	Banco de Dados I	06	100
IDC12	Banco de Dados II	04	67
IDC07	Programação	04	67
IDC16	Estrutura de Dados	06	100

Cod_Disciplina	Cod_Prof	Nome_Prof
IDC09	IP032	Natacha Targino
IDC09	IP059	Gabrielle Canalle
IDC12	IP098	João Vicente
IDC12	IP059	Gabrielle Canalle
IDC12	IP044	Vinícius Filho
IDC07	IP098	João Vicente
IDC16	IP032	Natacha Targino
IDC16	IP029	Mariana Rachel

**Disciplina** (Cod\_Disciplina, Nome\_Disciplina, Créditos, Horas)

**Disciplina\_Prof** (Cod\_Disciplina, Cod\_Prof, Nome\_Prof)

Cod\_Disciplina referencia Disciplina



# Normalização

**2FN:** Uma tabela está na segunda forma normal, quando ela **está na 1NF** e todo atributo não-chave é plenamente dependente da chave primária, ou seja,

**NÃO Contém Dependência Funcional Parcial**

**(Dependência Funcional Parcial:** Ocorre quando uma coluna **depende** apenas de **parte** de uma **chave primária composta**)

# Normalização: 2FN

Cod_Disciplina	Cod_Prof	Nome_Prof
IDC09	IP032	Natacha Targino
IDC09	IP059	Gabrielle Canalle
IDC12	IP098	João Vicente
IDC12	IP059	Gabrielle Canalle
IDC12	IP044	Vinícius Filho
IDC07	IP098	João Vicente
IDC16	IP032	Natacha Targino
IDC16	IP029	Mariana Rachel

**Disciplina\_Prof** (Cod\_Disciplina, Cod\_Prof, Nome\_Prof)

Cod\_Disciplina referencia Disciplina

# Normalização: 2FN

Cod_Prof	Nome_Prof
IP032	Natacha Targino
IP059	Gabrielle Canalle
IP098	João Vicente
IP044	Vinícius Filho
IP029	Mariana Rachel

Cod_Disciplina	Cod_Prof
IDC09	IP032
IDC09	IP059
IDC12	IP098
IDC12	IP059
IDC12	IP044
IDC07	IP098
IDC16	IP032
IDC16	IP029

**Prof** (Cod\_Prof, Nome\_Prof)

**Disciplina\_Prof** (Cod\_Disciplina, Cod\_Prof)

Cod\_Disciplina referencia Disciplina

Cod\_Prof referencia Prof

# Normalização

**3FN:** Uma tabela está na segunda forma normal, quando ela **está na 2NF** e nenhum atributo não-chave é transitivamente dependente da chave primária ou seja,

**NÃO Contém Dependência Funcional Transitiva**

**(Dependência Funcional Transitiva:** Ocorre quando uma coluna, além de depender da chave primária da tabela, **depende de outra coluna** ou conjunto de colunas da tabela)

# Normalização: 3FN

Cod_Disciplina	Nome_Disciplina	Créditos	Horas
IDC09	Banco de Dados I	06	100
IDC12	Banco de Dados II	04	67
IDC07	Programação	04	67
IDC16	Estrutura de Dados	06	100

**Disciplina** (Cod\_Disciplina, Nome\_Disciplina, Créditos, Horas)

# Normalização: 3FN

Créditos	Horas
06	100
04	67

Cod_Disciplina	Nome_Disciplina	Créditos
IDC09	Banco de Dados I	06
IDC12	Banco de Dados II	04
IDC07	Programação	04
IDC16	Estrutura de Dados	06

**Creditos**(creditos, horas)

**Disciplina**(Cod\_Disciplina, Nome\_Disciplina, Créditos)

Creditos referencia Creditos

# Normalização

Antes

Cod_Disciplina	Nome_Disciplina	Cod_Prof	Nome_Prof	Créditos	Horas
IDC09	Banco de Dados I	IP032	Natacha Targino	06	100
		IP059	Gabrielle Canalle		
IDC12	Banco de Dados II	IP098	João Vicente	04	67
		IP059	Gabrielle Canalle		
		IP044	Vinícius Filho		
IDC07	Programação	IP098	João Vicente	04	67
IDC16	Estrutura de Dados	IP032	Natacha Targino	06	100
		IP029	Mariana Rachel		

X

Depois

Cod_Disciplina	Nome_Disciplina	Créditos
IDC09	Banco de Dados I	06
IDC12	Banco de Dados II	04
IDC07	Programação	04
IDC16	Estrutura de Dados	06

Cod_Prof	Nome_Prof
IP032	Natacha Targino
IP059	Gabrielle Canalle
IP098	João Vicente
IP044	Vinícius Filho
IP029	Mariana Rachel

Cod_Disciplina	Cod_Prof
IDC09	IP032
IDC09	IP059
IDC12	IP098
IDC12	IP059
IDC12	IP044
IDC07	IP098
IDC16	IP032
IDC16	IP029

Créditos	Horas
06	100
04	67

# Linguagem SQL

Structured Query  
Language

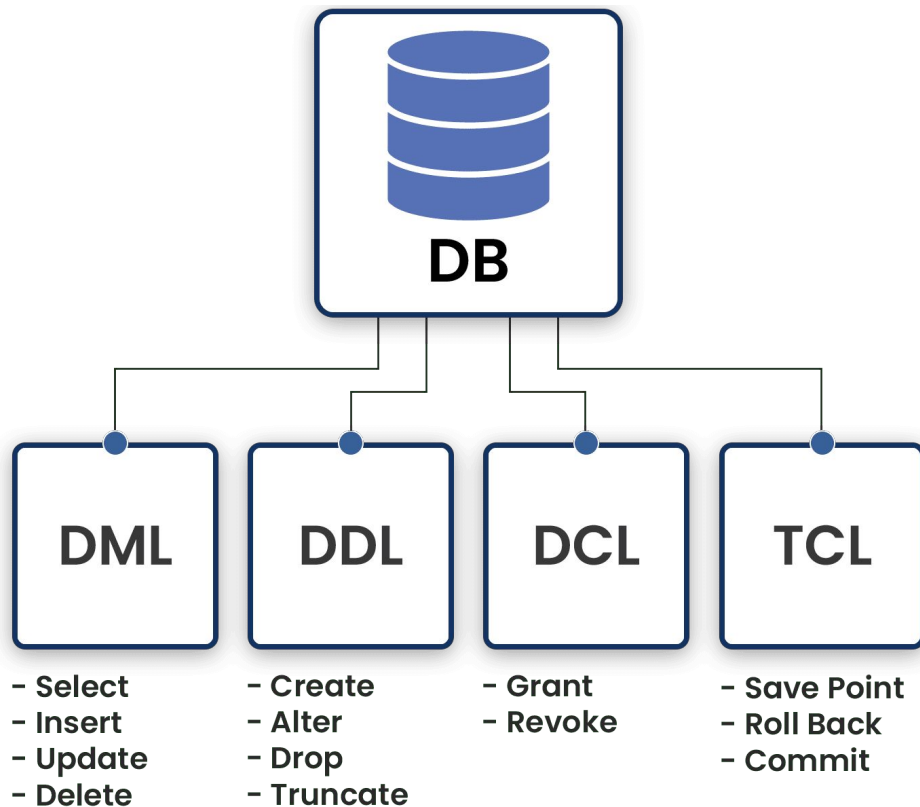




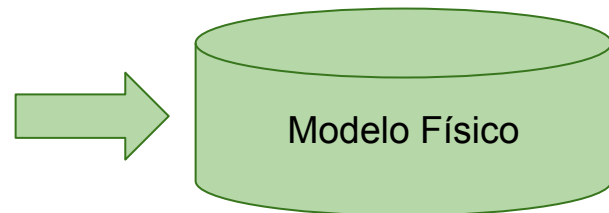
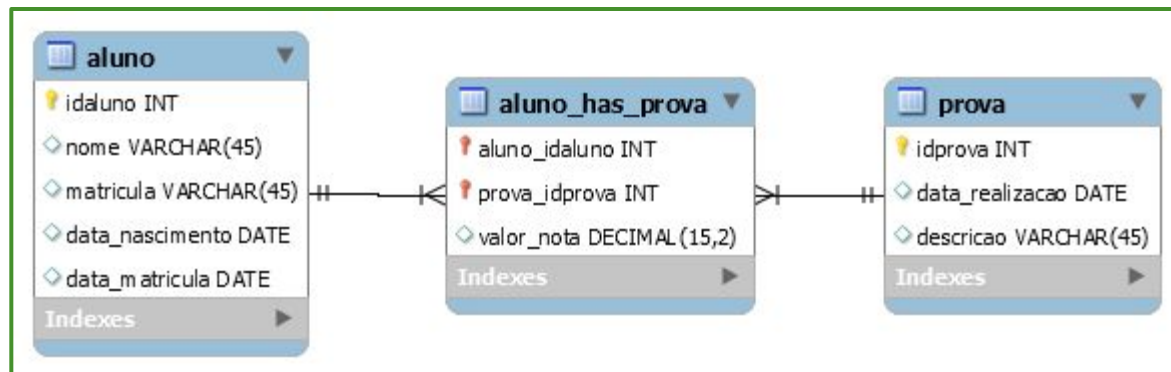
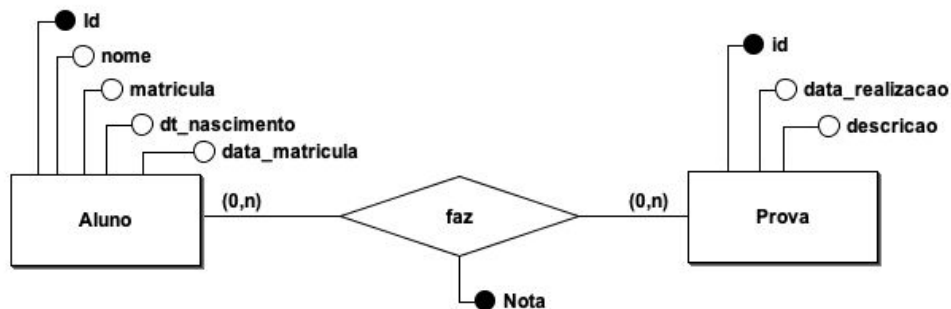
# Linguagem SQL

- SQL - **S**tructured **Q**uery **L**anguage
- Padrão para bd's relacionais
- Linguagem de Consulta Estruturada
  - Apesar do QUERY no nome, não é apenas de consulta (definição e manipulação)
- Fundamentada no modelo relacional
  - Álgebra Relacional
    - união, interseção, seleção, junção...
- Inicialmente denominada SEQUEL (Structured English Query Language) evoluiu desde sua criação

# Linguagem SQL



# Do lógico para o físico...



**DDL**

# Data Definition Language (DDL)

Define os comandos utilizados para criação (**CREATE**) de esquemas, tabelas, views, índices, atualização dessas estruturas (**ALTER**), assim como a remoção (**DROP**), renomeação (**RENAME**), remoção de todas as linhas de uma tabela sem condição (**TRUNCATE**)

# Data Definition Language (DDL)

## Restrições de integridade

Garantem que as mudanças realizadas no banco não resultem em perda de consistência

- **Integridade de Domínio:**
  - O valor de um campo deve obedecer à definição de valores admitidos para o domínio da coluna
  - Domínios: número inteiro, número real, alfanumérico, data, etc.
- **Integridade de Nulo:**
  - Especifica se o valor de um campo pode ser nulo
  - Campos que compõem a PK não podem ser nulos
- **Integridade de Chave:**
  - Define que os valores de chave primária e alternativa devem ser únicos

# Tipos de dados no MySQL

## MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 2147483647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

# Data Definition Language (DDL)

## Restrições de integridade

- Integridade de Domínio
- Integridade de Nulo
- Integridade de Chave

```
Create table cliente(  
  id_cliente int,  
  nome_cliente char(20) not null,  
  rua_cliente char(30),  
  cidade_cliente char(30),  
  primary key (id_cliente))
```



# Data Definition Language (DDL)

## Restrições de integridade

- **Integridade Referencial:**

- Os valores dos campos que aparecem em uma chave estrangeira (FK) devem aparecer na chave primária (PK) da tabela referenciada

- **Integridade Semântica:**

- Exemplos:
  - “Nenhum aluno pode estar matriculado em mais de um curso”
  - “A carga horária máxima de uma disciplina é de 120 horas”
- Pode ser implementada através de mecanismos como regras e triggers

# Data Definition Language (DDL)

## Restrições de integridade

- **Integridade Referencial**

- Utilizado para garantir que chaves estrangeiras estejam cadastradas nas suas tabelas de origem

ALUNO

Código	nome	telefone
<b>001</b>	João	22321122
<b>002</b>	Maria	77534423

MATRÍCULA

Código_aluno	Curso
<b>001</b>	Inglês
<b>001</b>	Alemão
<b>002</b>	Francês

Primary key → Foreign key

# Data Definition Language (DDL)

## Constraints

- Constraints são restrições
- Mecanismo capaz de implementar controles que garantam a consistência dos dados

<b>NOT NULL</b>	Informa que o campo em questão não pode ser nulo
<b>UNIQUE</b>	Indica que os valores na coluna não podem ser repetidos
<b>PRIMARY KEY</b>	Identifica a chave primária da tabela
<b>FOREIGN KEY</b>	Identifica uma chave estrangeira da tabela, criando um link entre as tabelas
<b>CHECK</b>	Determina uma regra de validação
<b>DEFAULT</b>	Define um valor padrão para uma coluna que deve ser aplicado

# Data Definition Language (DDL)

## Constraints - Sintaxe (criação)

- nome\_coluna tipo\_dado **NOT NULL**
- nome\_coluna tipo\_dado **UNIQUE**
- nome\_coluna tipo\_dado **PRIMARY KEY**
- nome\_coluna tipo\_dado **CHECK (condição)**
  - **EX.: constraint pessoa\_idade check (idade > 18)**
- nome\_coluna tipo\_dado **DEFAULT** <valor>
- **FOREIGN KEY** nome\_coluna **REFERENCES** <nomeTabela>(<nome\_coluna>)

## Eliminando uma constraint

- **ALTER TABLE** <nomeTabela> **DROP** <tipoConstraint> <nomeConstraint>;

# Data Definition Language (DDL)

## Criação do esquema

**CREATE SCHEMA** <nome do esquema>;

ou

**CREATE SCHEMA** <nome do esquema> **AUTHORIZATION** <usuário dono do esquema>;

- Nem todos os usuários são autorizados a criar esquemas ou elementos de esquema, como tabelas e colunas, por exemplo
  - Os privilégios para esse tipo de ação devem ser concedidos às contas de usuário pelo administrador do sistema ou DBA (administrador do banco de dados)

# Data Definition Language (DDL)

## Create

- O nome de uma tabela em um bd deve ser único
- O nome de uma coluna em uma tabela também deve ser único
  - podem existir colunas com o mesmo nome em tabelas diferentes.
- Nomeação da tabela:
  - Deve começar com uma letra;
  - Deve conter de 1 a 30 caracteres;
  - Não deve ser uma palavra reservada;
  - Não deve existir no esquema;

# Data Definition Language (DDL)

## Create - Sintaxe

**CREATE TABLE** [IF NOT EXISTS] <tabela>(  
    <nomeAtributo1> <tipoDado1> [CONSTRAINTS],  
    <nomeAtributo2> <tipoDado2> [CONSTRAINTS],  
    .....,  
    [CONSTRAINTS]);

FILME

cod	nome	diretor
001	Titanic	João
002	Matrix	José

```
create table filme(  
cod char(3) PRIMARY KEY,  
nome varchar(255) NOT NULL,  
diretor varchar(255) NOT NULL);
```

OU

```
create table filme(  
cod char(3) NOT NULL,  
nome varchar(255) NOT NULL,  
diretor varchar(255) NOT NULL,  
CONSTRAINT PK_filme PRIMARY KEY(cod));
```

# Data Definition Language (DDL)

## Create - Default

- Serve para especificar um valor default para uma coluna
- Podem ser atribuídos valores literais, expressões e até mesmo funções SQL
- O tipo de dados default deve ser igual ao especificado para a coluna.

```
Create table cliente(  
    id_cliente int PRIMARY KEY,  
    cpf char(11),  
    nome varchar(20),  
    cidade varchar(30) default 'Recife')
```



# Data Definition Language (DDL)

## Alter

- Permite alterar constraints/atributos de uma tabela ou adicionar novas constraints/atributos

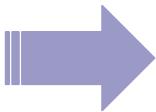
COMANDO	DESCRIÇÃO
ADD	Adiciona coluna ou restrição
MODIFY	Modifica definição de coluna
DROP <TIPOCONSTRAINT>	Apaga restrição
DROP COLUMN	Apaga coluna
RENAME TO	Altera nome da tabela
RENAME COLUMN	Altera nome da coluna

# Data Definition Language (DDL)

## Alter

**alter table** estudante **add** data\_alt default SYSDATE

Cod_cliente	Nome
001	Pedro
002	Maria
003	João



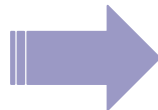
Cod_cliente	Nome	Data_alt
001	Pedro	14/10/2022
002	Maria	02/10/2022
003	João	05/10/2022

# Data Definition Language (DDL)

## Alter

**alter table** estudante **drop column** telefone;

Cod_cliente	Nome	Telefone
001	Pedro	99338822
002	Maria	98392922
003	João	90020332



Cod_cliente	Nome
001	Pedro
002	Maria
003	João

# Data Definition Language (DDL)

## Drop

- Permite a remoção de uma tabela do banco de dados
  - definição + dados

## Sintaxe:

**drop** table <nomeTabela>;

# Data Definition Language (DDL)

## Truncate

- Permite a remoção de todas as tuplas de uma tabela do banco de dados

### Sintaxe:

**truncate** table <nomeTabela>;

**DML**

# Data Manipulation Language (DML)

Define os comandos utilizados para manipulação de dados no banco

## Comandos:

- **INSERT**
- **UPDATE**
- **DELETE**

# Data Manipulation Language (DML)

## Insert

- Permite adicionar uma tupla em uma tabela
  - se a inserção for feita com os atributos na ordem correta, não é necessário especificar as colunas

## Sintaxe

```
INSERT INTO <nomeTabela> [listaColunas]
```

```
VALUES (<listaValoresAtômicos>);
```

```
INSERT INTO cliente VALUES (20194853, 'Maria', 30);
```

```
INSERT INTO cliente (cpf, nome, idade) VALUES (20194853, 'José', 28);
```

```
INSERT INTO cliente (cpf, nome, idade) VALUES (20194853, 'Paulo', 18), (20197432, 'João', 25);
```



# Data Manipulation Language (DML)

## Insert - **AUTO\_INCREMENT**

- Realiza a geração automática de números únicos para a chave primária.
- Utilizado no momento da criação da tabela

```
CREATE TABLE produto (  
  codigo smallint AUTO_INCREMENT primary key,  
  tipo VARCHAR(40),  
  preco DECIMAL(10,2));  
  
INSERT INTO produto (tipo, preco) VALUES("alimentação", 6);
```

# Data Manipulation Language (DML)

## Insert - **AUTO\_INCREMENT**

- Adicionando em tabela já criada

```
ALTER TABLE produto MODIFY codigo smallint AUTO_INCREMENT;
```

- Alterando o valor de um auto\_increment

```
ALTER TABLE produto AUTO_INCREMENT = 23;
```

# Data Manipulation Language (DML)

## Update

- Permite alterar valores de atributos de uma tabela com base em critérios especificados
  - Se não for especificada uma condição na cláusula WHERE o update será aplicado para todas as tuplas da tabela

## Sintaxe

UPDATE <nomeTabela>

SET <atributo> = <novoValor>

[WHERE <condição>];

# Data Manipulation Language (DML)

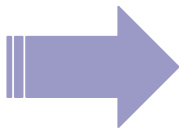
## Update

UPDATE cliente

SET telefone = 0101010101

WHERE cod\_cliente = 001;

<b>cod_cliente</b>	<b>nome</b>	<b>telefone</b>
001	Pedro	99338822
002	Maria	98392922
003	João	90020332



<b>cod_cliente</b>	<b>nome</b>	<b>telefone</b>
001	Pedro	0101010101
002	Maria	98392922
003	João	90020332

# Data Manipulation Language (DML)

## Delete

- Permite remover tuplas de uma relação
  - Se não for especificada uma condição na cláusula WHERE todas as tuplas serão excluídas

## Sintaxe

DELETE FROM <nomeTabela>

[WHERE <condição>];

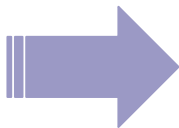
# Data Manipulation Language (DML)

## Delete

Delete from cliente

where nome = "Maria";

<b>cod_cliente</b>	<b>nome</b>	<b>telefone</b>
001	Pedro	99338822
002	Maria	98392922
003	João	90020332



<b>cod_cliente</b>	<b>nome</b>	<b>telefone</b>
001	Pedro	99338822
003	João	90020332

# Dúvidas?

Gabrielle K. Canalle  
gkc@cesar.school

Natacha Targino  
ntrsb@cesar.school

