

# Banco de Dados I

## Modelo Conceitual

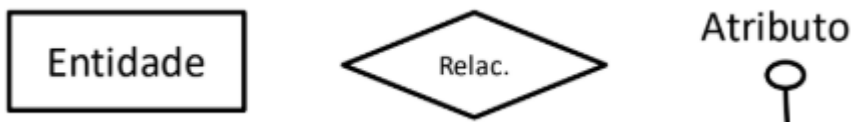
Descreve a estrutura de um BD de uma forma mais próxima da percepção dos usuários

Independente de aspectos de implementação

Representa a estrutura de um banco de dados sem considerar um SGBD específico

## Modelo Entidade-Relacionamento

### Três conceitos básicos



### Entidade

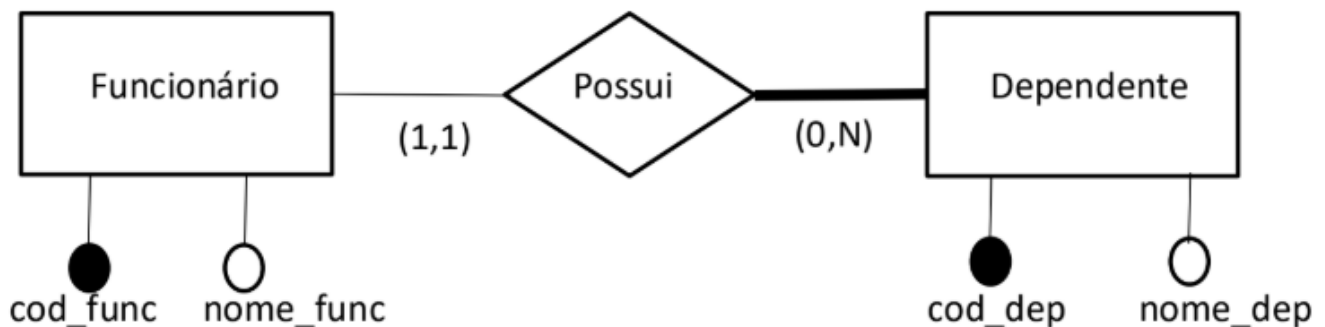
“Conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações no banco de dados”

Podendo ser:

- Algo que existe fisicamente, que pode ser tocado
- Algo que existe conceitualmente (Abstrato)
- Podem ser eventos

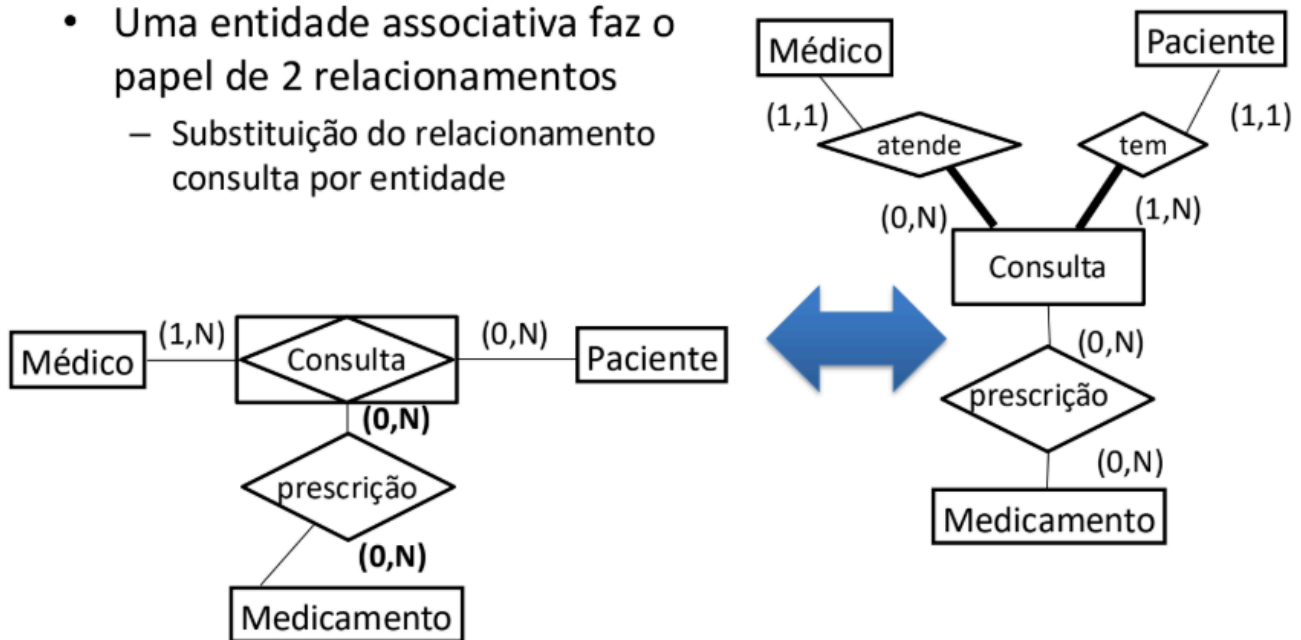
### Entidade Fraca

Existem entidades que apenas existem em função de outras



### Entidade Associativa

- Uma entidade associativa faz o papel de 2 relacionamentos
  - Substituição do relacionamento consulta por entidade



## Relacionamento

Conjunto de associações entre ocorrências/instâncias de entidades  
Cada ocorrência da entidade que participa de um relacionamento desempenha um Papel

Úteis sobretudo nos auto-relacionamentos (relacionamento unário)

## Grau de Relacionamento

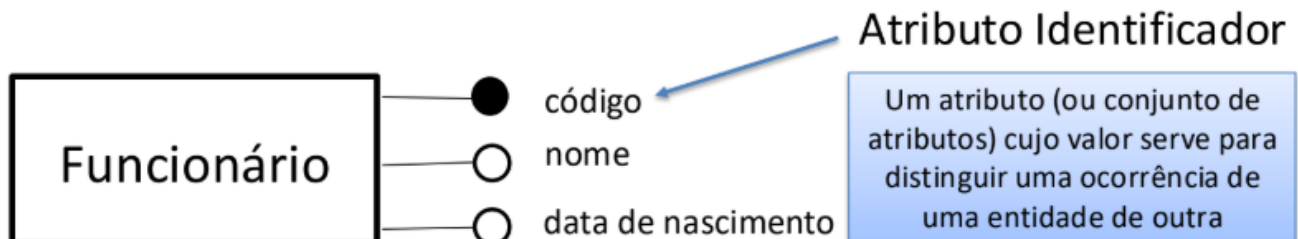
Número de (instâncias de) entidades que participam do relacionamento

## Cardinalidades

Expressar o número de ocorrências/instâncias às quais outra ocorrência/instância pode ser associada através de um conjunto de relacionamentos

- (1:1)
- (1:N)
- (N:N)

## Atributo

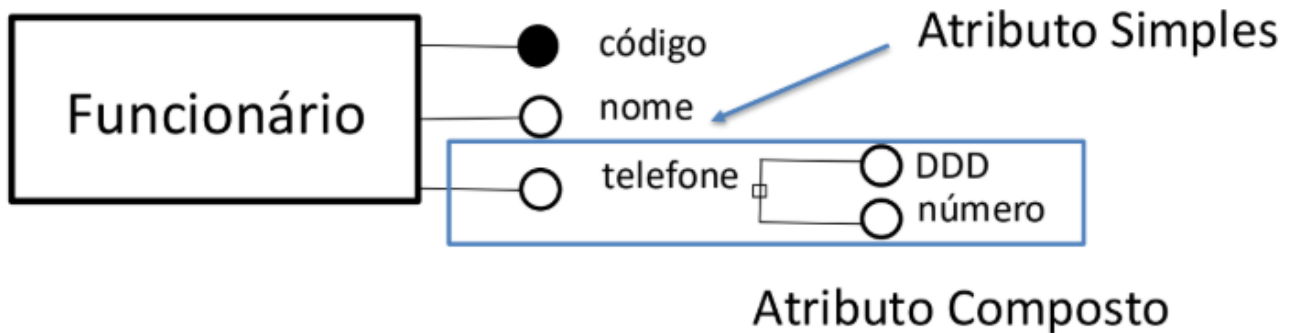


Propriedades que descrevem uma entidade

Exemplo:

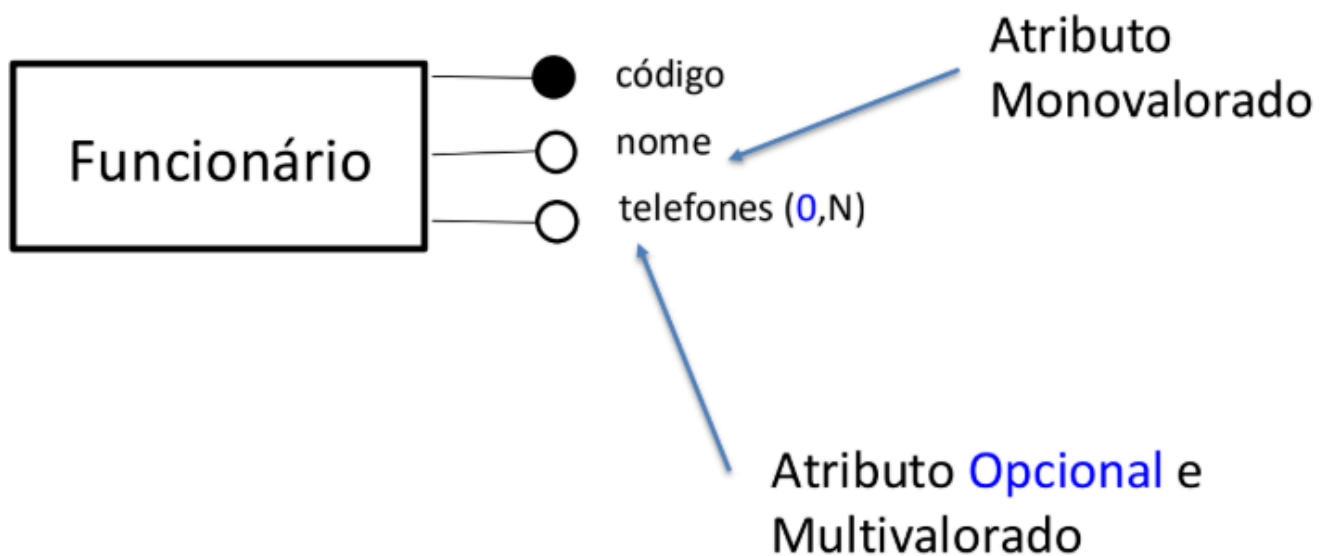
- Entidade: Funcionário
- Atributos: código, nome, data de nascimento

### Atributo Composto

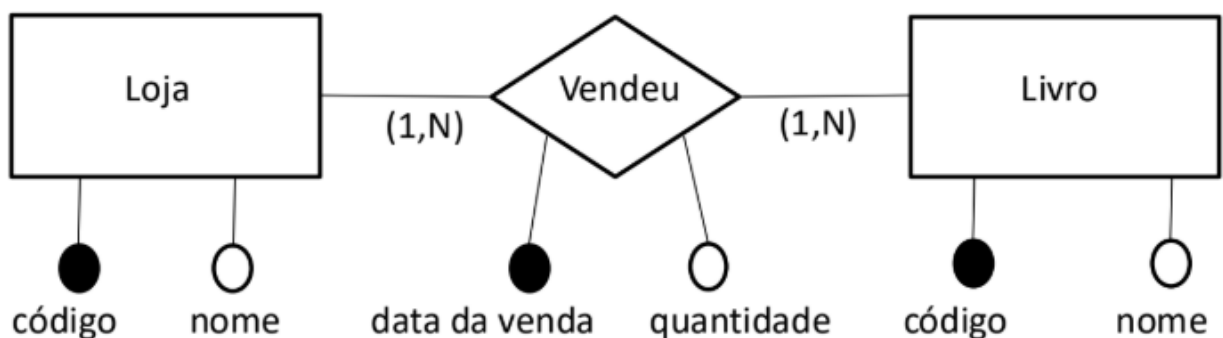


### Atributo Multivalorado

Atributos multivalorados e compostos são atributos complexos



- Atributo de Relacionamento

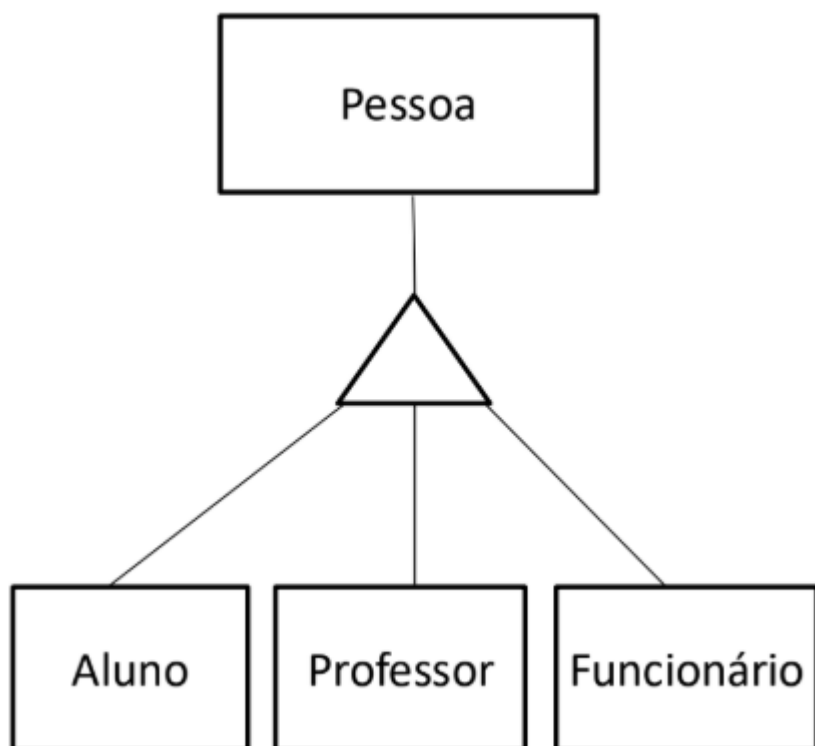


### Generalização (Especialização)

É um relacionamento de classificação entre uma entidade mais geral e outra mais específica

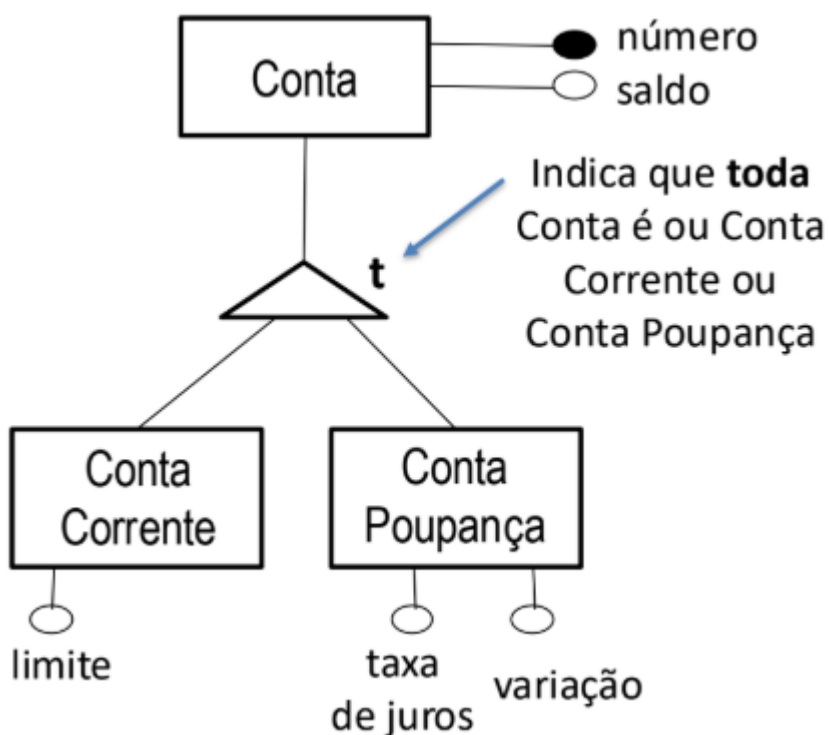
A entidade mais geral é denominada entidade de nível superior (**superclasse**) e a mais específica de entidade de nível inferior (**subclasse**)

As propriedades da superclasse são herdadas pela subclasse → **Herança**



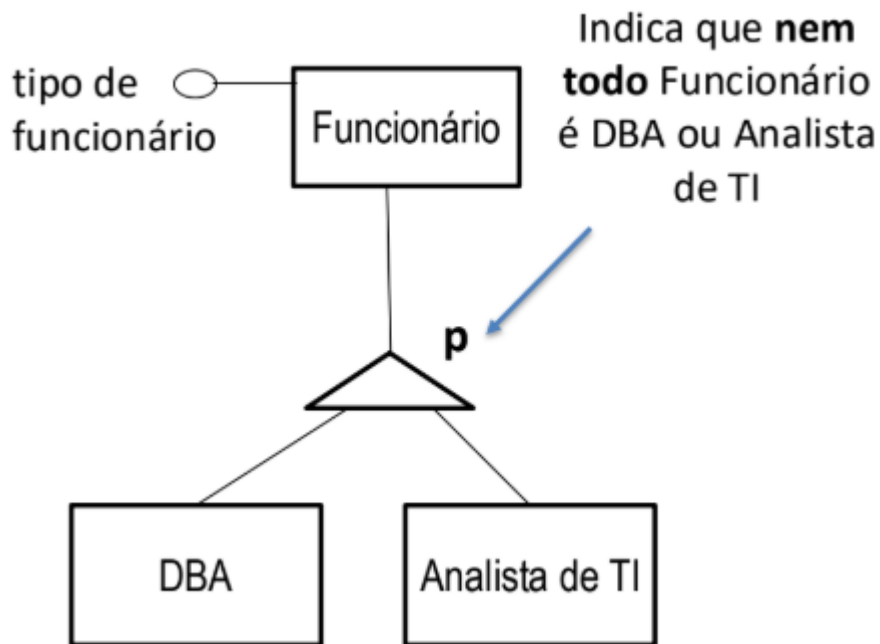
### Especialização Total

Para cada ocorrência da entidade genérica existe **sempre** uma entidade especializada



### Especialização Parcial

**Nem toda** ocorrência da entidade genérica possui uma ocorrência correspondente em uma entidade especializada










### Especialização Exclusiva

**Exclusiva (x)**: ocorrência de uma entidade genérica em apenas uma entidade especializada

### Especialização Compartilhada

**Compartilhada (c)**: Ocorrência de uma entidade genérica pode aparecer em **várias** entidades especializadas

Conceito	Símbolo									
Entidade										
Relacionamento										
Atributo										
Atributo identificador										
Relacionamento identificador										
Generalização/ especialização	<div></div> <table><tr><th></th><th>Total (t)</th><th>Parcial (p)</th></tr><tr><td>Exclusiva (x)</td><td>xt</td><td>xp</td></tr><tr><td>Compartilhada (c)</td><td>ct</td><td>cp</td></tr></table>		Total (t)	Parcial (p)	Exclusiva (x)	xt	xp	Compartilhada (c)	ct	cp
	Total (t)	Parcial (p)								
Exclusiva (x)	xt	xp								
Compartilhada (c)	ct	cp								
Entidade associativa										

## Modelo ER - Símbolos [Heuser, 2009]

Entidade fraca

17

## Construção do Modelo

### Transformando Relacionamento N:N para Entidade

1. O relacionamento N:N é representado como uma entidade
2. A entidade criada é relacionada às entidades que originalmente participavam do relacionamento
3. A entidade criada tem como identificador:
  - Os relacionamentos com as entidades que originalmente participavam do relacionamento
  - Os atributos que eram identificadores do relacionamento original (caso o relacionamento original tivesse atributos identificadores)
4. A cardinalidade da entidade criada em cada relacionamento de que participa é (1,1)
5. As cardinalidades das entidades que eram originalmente associadas pelo relacionamento transformado em entidade são transcritas ao novo modelo conforme mostrado no exemplo inicial

### Entidade vs. Atributo

Se objeto está vinculado a outros objetos:

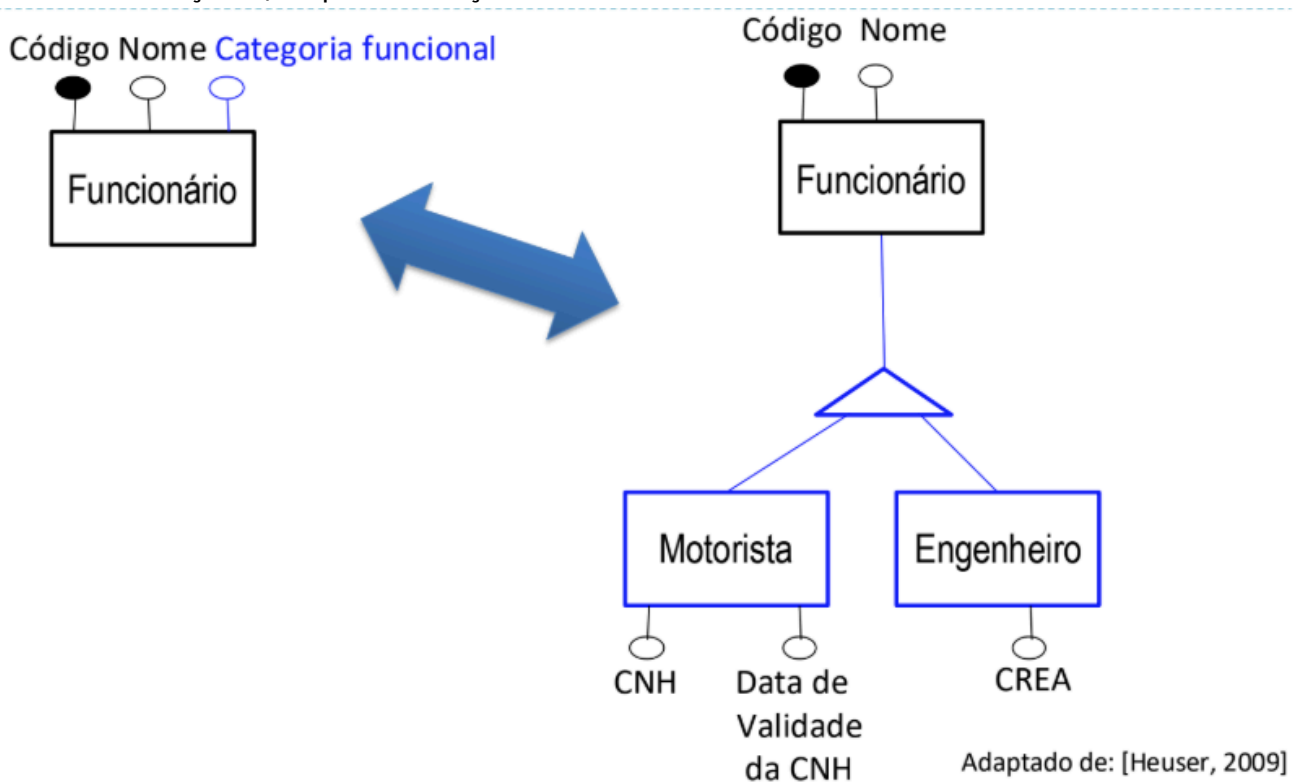
- Deve ser modelado como entidade  
Caso contrário:
- Pode ser modelado como atributo  
Conjunto de valores de um determinado objeto é fixo (domínio fixo):
- Pode ser modelado como atributo  
Existem transações no sistema que alteram o conjunto de valores do objeto (domínio variável):
- Não deve ser modelado como atributo

## Atributo vs. Especialização/Generalização

Especialização deve ser usada quando:

As classes especializadas de entidades possuem propriedades particulares:

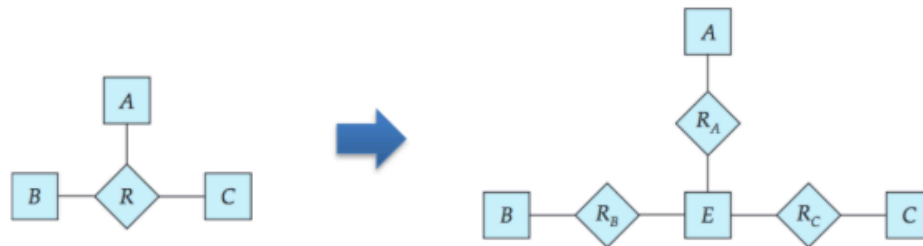
- Atributos
- Relacionamentos
- Generalizações/especializações



## Conversão de relacionamentos não binários para a forma binária

- Substituir  $R$  entre entidades  $A$ ,  $B$  e  $C$  por uma entidade  $E$ , e três relacionamentos:

1.  $R_A$ , relacionando  $E$  e  $A$
  2.  $R_B$ , relacionando  $E$  e  $B$
  3.  $R_C$ , relacionando  $E$  e  $C$
- Criar um atributo identificador especial para  $E$
  - Adicionar quaisquer atributos de  $R$  para  $E$
  - Para cada relacionamento  $(a_i, b_i, c_i)$  em  $R$ , criar
    1. uma nova ocorrência  $e_i$  na entidade  $E$
    2. adiciona  $(e_i, a_i)$  para  $R_A$
    3. adiciona  $(e_i, b_i)$  para  $R_B$
    4. adiciona  $(e_i, c_i)$  para  $R_C$



## Modelo Relacional

O modelo relacional representa um banco de dados como um conjunto de relações

**Relação**  $\leftrightarrow$  **Tabela** (de valores)

coluna (atributo)		nome do campo (nome do atributo)
<i>ID_Professor</i>	<i>Nome</i>	<i>CPF</i>
1	Silva	11111111111
2	Souza	22222222222
3	Costa	33333333333

linha (tupla)

valor do campo  
(valor do atributo)

Obs.: **Domínio** = Conjunto de valores que pode aparecer em cada coluna

## Relação $R$

$$R (A_1, A_2, A_3, \dots, A_n)$$

Onde:

- $R$  é o nome da relação
- $A_1, A_2, \dots, A_n$  é uma lista de atributos
- $N$  é o grau da relação

Exemplo:



Professor (ID\_Professor, Nome, CPF)

## Chave Primária (PK)

Seja  $K \subset R$

K é uma super chave de R se os valores de K são suficientes para identificar um única tupla de cada possível relação de  $r(R)$

Super-chave K é uma chave candidata se K é mínima

Uma das chaves candidatas é selecionada para ser **chave primária**

Chave(s) candidata(s) não selecionada(s) → chave(s) alternativa(s)

## Restrições de integridade de valores Null

Especifica se a um atributo é permitido ter valores Null

## Chave Estrangeira

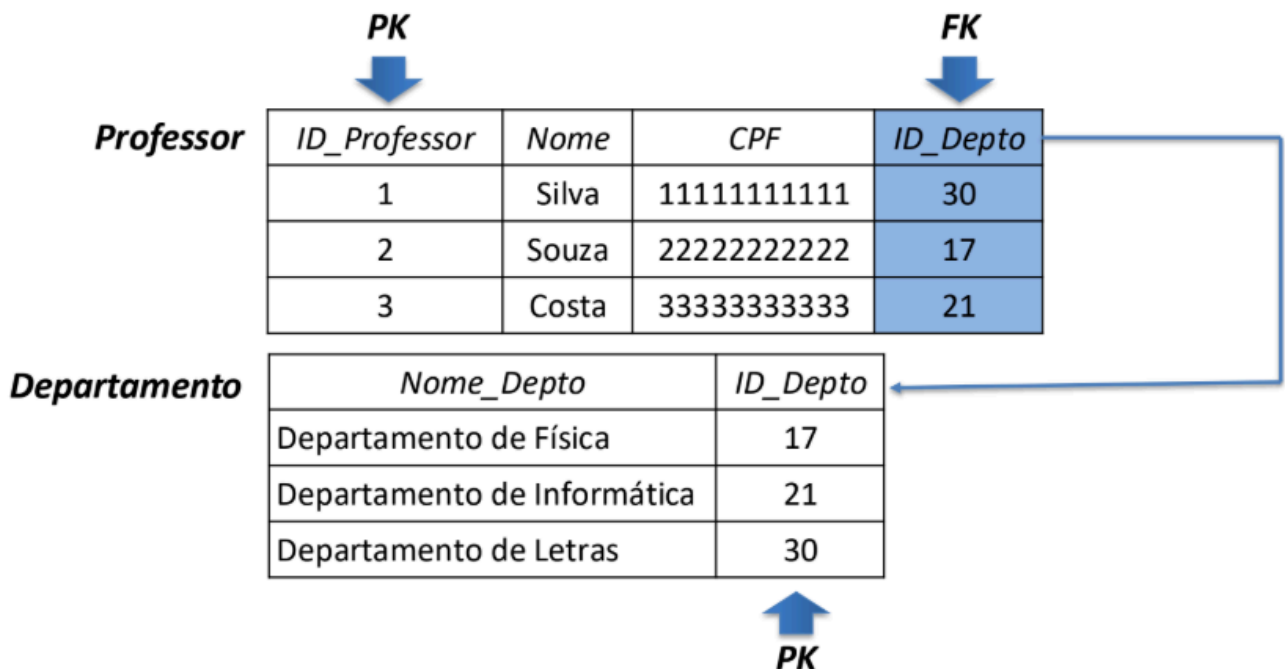
$$R_1[FK] \rightarrow R_2[PK]$$

Onde:

- PK é a chave primária
- FK é a chave estrangeira

Então, para qualquer tupla  $t_1$  de  $R_1$ :

$t_1[FK] = t_2[PK]$ , onde  $t_2$  é uma tupla de  $R_2$  ou  $t_1[FK]$  é Null



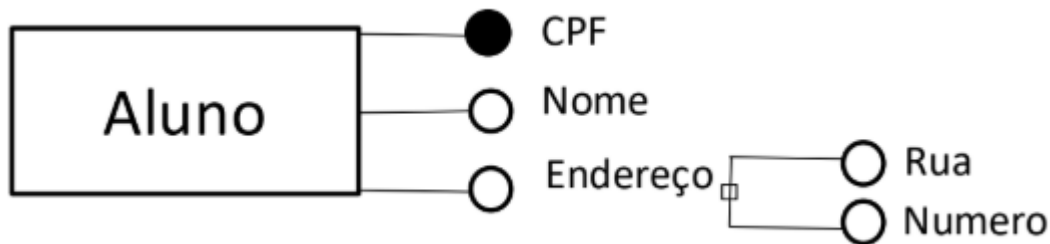
## Esquema Lógico

Departamento (Nome\_Depto, ID\_Depto)  
PK(ID\_Depto)

```
Professor (ID_Professor, Nome, CPF, ID_Depto)
  PK(ID_Professor)
  FK(ID_Depto) ref Departamento(ID_Depto)
```

## Mapeamento ER → Relacional

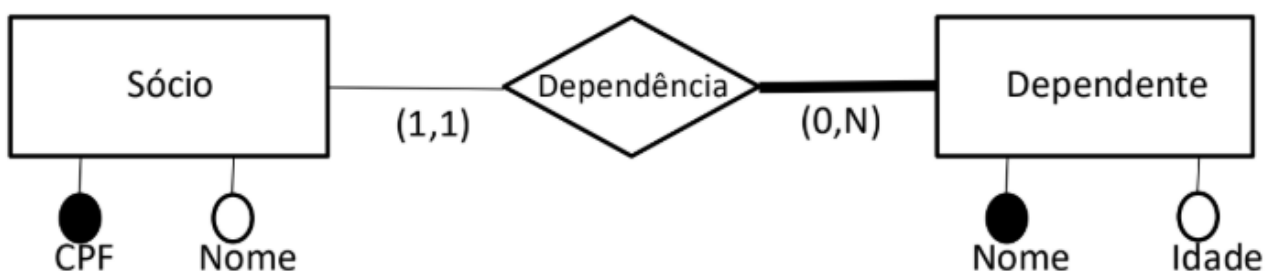
### Entidades Fortes



1. Criar uma relação: `Aluno()`
2. Para todo Atributo Simples criar um atributo: `Aluno(Nome, CPF)`
3. Para atributos compostos, criar vários atributos simples: `Aluno(Nome, CPF, Rua, Numero)`
4. Criar uma chave primária

```
Aluno(Nome, CPF, Rua, Numero)
  PK(CPF)
```

### Entidades Fracas



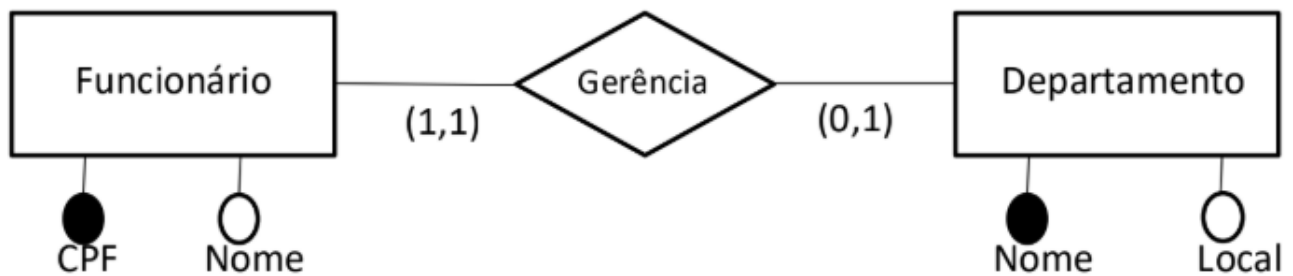
1. Criar uma relação `Dependente()`
2. Para todo Atributo Simples criar um atributo: `Dependente(Nome, idade)`
3. Para atributos compostos, criar vários atributos simples
4. Criar uma FK apontando para PK da Entidade Forte

```
Dependente(Nome, Idade, SocioCPF)
  FK(SocioCPF) ref Socio(CPF)
```

5. Criar uma PK composta pelo Atributo identificador e FK

```
Dependente(Nome, Idade, SocioCPF)
FK(SocioCPF) ref Socio(CPF)
PK(Nome, SocioCPF)
```

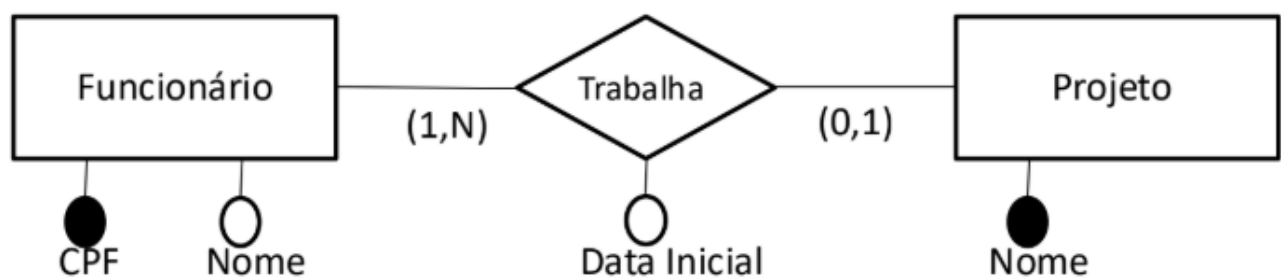
## Relacionamentos Binários 1:1



1. Criar uma FK na relação com participação total (*Todo departamento tem funcionário*)

```
Funcionário (Nome, CPF)
PK(CPF)
Departamento (Nome, Local, GerenteCPF)
FK(GerenteCPF) ref Funcionário(CPF)
PK(Nome)
```

## Relacionamentos Binários 1:N



1. Criar uma FK na relação com cardinalidade N

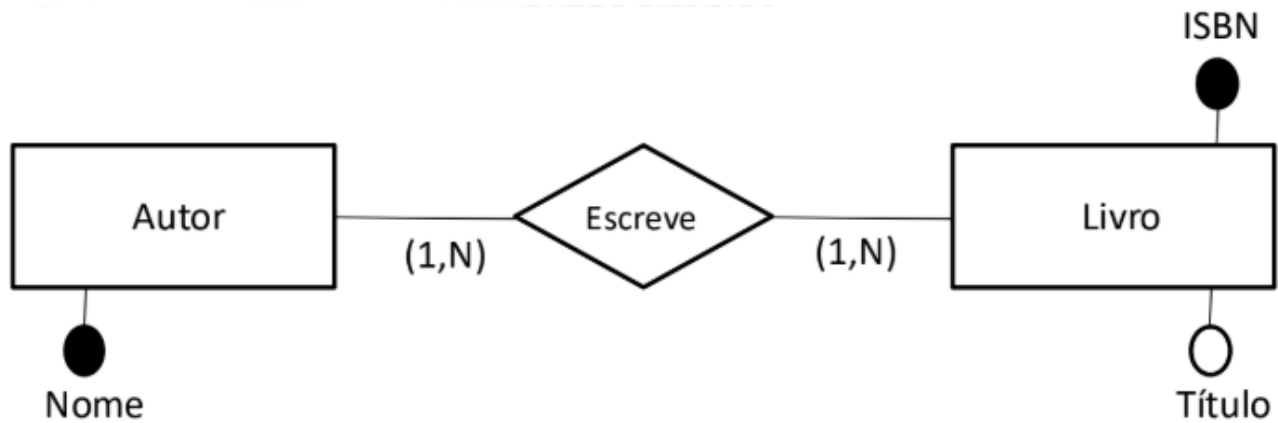
```
Funcionário (Nome, CPF, ProjetoNome)
PK(CPF)
FK(ProjetoNome) ref Projeto(Nome)
Projeto (Nome)
PK(Nome)
```

2. Criar todos os atributos do relacionamento, se houver

```
Funcionário (Nome, CPF, ProjetoNome, DataInicial)
PK(CPF)
```

```
FK(ProjetoNome) ref Projeto(Nome)
Projeto (Nome)
PK(Nome)
```

## Relacionamentos Binários N:N



### 1. Criar um novo Relacionamento

```
Autor (Nome)
    PK(Nome)

Livro (ISBN, Título)
    PK(ISBN)

Escreve ()
```

### 2. Criar FK das duas relações

```
Autor (Nome)
    PK(Nome)

Livro (ISBN, Título)
    PK(ISBN)

Escreve (AutorNome, LivroISBN)
    FK(AutorNome) ref Autor(Nome)
    FK(LivroISBN) ref Livro(ISBN)
```

### 3. Criar a PK (FK1 + FK2)

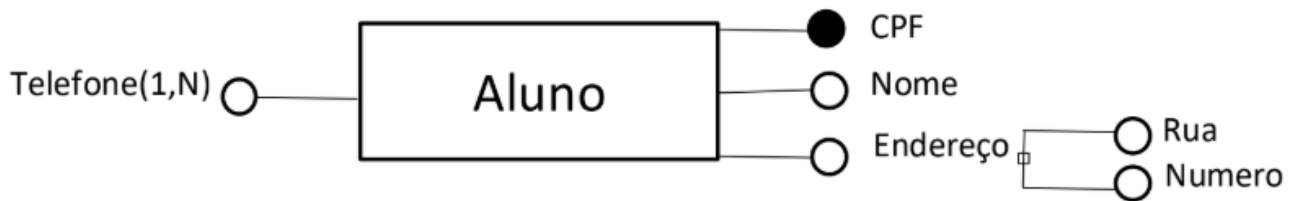
```
Autor (Nome)
    PK(Nome)

Livro (ISBN, Título)
    PK(ISBN)

Escreve (AutorNome, LivroISBN)
```

```
FK(AutorNome) ref Autor(Nome)
FK(LivroISBN) ref Livro(ISBN)
PK(AutorNome, LivroISBN)
```

## Atributos Multivalorados



1. Criar uma nova relação: `Telefone ()`
2. Criar Atributo(s) Simples: `Telefone (Telefone)`
3. Cria FK para a relação original

```
Aluno(CPF, Nome, Endereço)
  PK(CPF)

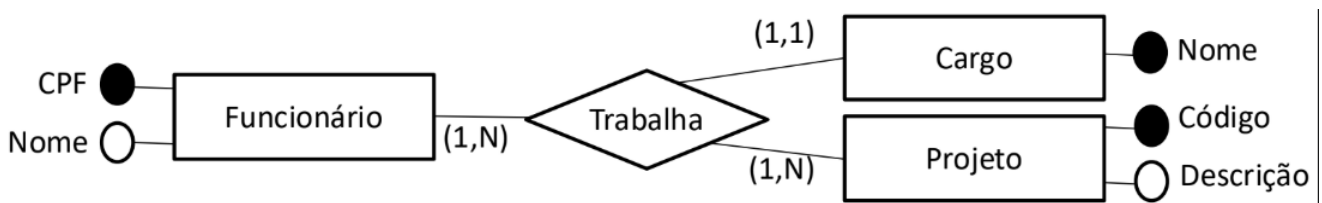
Telefone(Telefone, AlunoCPF)
  FK(AlunoCPF) ref Aluno(CPK)
```

4. Criar PK (FK + Atributos)

```
Aluno(CPF, Nome, Endereço)
  PK(CPF)

Telefone(Telefone, AlunoCPF)
  FK(AlunoCPF) ref Aluno(CPF)
  PK(Telefone, AlunoCPF)
```

## Relacionamentos N-ários, N>2



1. Criar uma nova relação `Trabalha ()`
2. Criar Atributo(s) simples
3. Criar FK para todas as relações

Funcionário (CPF, Nome)

PK(CPF)

Cargo (Nome)

PK(Nome)

Projeto (Código, Descrição)

PK(Código)

Trabalha (FCPF, CNome, PCódigo)

FK(FCPF) ref Funcionário(CPF)

FK(CNome) ref Cargo(Nome)

FK(PCódigo) ref Projeto(Código)

#### 4. Criar PK com todas as relações que não sejam 1

Funcionário (CPF, Nome)

PK(CPF)

Cargo (Nome)

PK(Nome)

Projeto (Código, Descrição)

PK(Código)

Trabalha (FCPF, CNome, PCódigo)

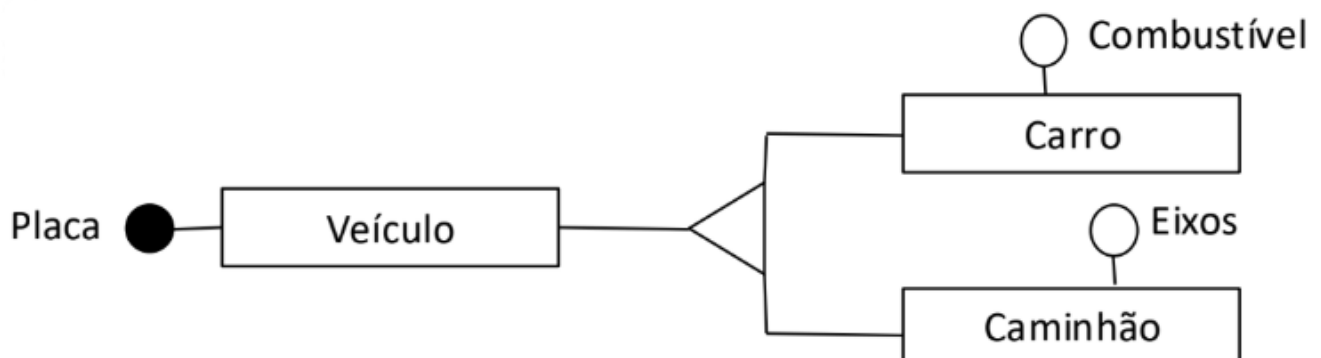
FK(FCPF) ref Funcionário(CPF)

FK(CNome) ref Cargo(Nome)

FK(PCódigo) ref Projeto(Código)

PK(FCPF, PCódigo)

## Mapeamento de Heranças



## Partição Única

```
Veículo (Placa, Combustível, Eixos, TipoVeículo*)  
    PK(Placa)
```

## Particionamento Vertical

```
Veículo (Placa)  
    PK(Placa)  
  
Carro (Combustível, VeículoPlaca)  
    FK(VeículoPlaca) ref Veículo(Placa)  
    PK(VeículoPlaca)  
  
Caminhão (Eixos, VeículoPlaca)  
    FK(VeículoPlaca) ref Veículo(Placa)  
    PK(VeículoPlaca)
```

## Particionamento Horizontal

```
Carro (Combustível, Placa)  
    PK(Placa)  
  
Caminhão (Eixos, Placa)  
    PK(Placa)
```

## Modelo Físico

### SQL - Struct Query Language

#### Tipos de Dados

Numérico (principais):

- integer/int, float, real, numeric(p,n)

Cadeia de caracteres:

- char(n), varchar(n), text

Dados binários:

- blob

Data/tempo:

- date, datetime, timestamp, time, year

Booleano:

- bool, boolean, tinyint(1)

# Criando Banco de Dados

```
create database nome_db
--ou
create schema nome_db
```

## Criando Tabela

```
create table r (A1 D1, A2 D2, ..., An Dn,
(integrity-constraint1),
...,
(integrity-constraintk));
```

$r$  é o nome da relação

Cada  $A_i$  é um nome de atributo no esquema da relação  $r$

$D_i$  é o tipo de dados dos valores no domínio do atributo  $A_i$

Exemplo:

```
create table Departamento (Nome_Depto varchar(50), ID_Depto numeric(5,0) );
```

## Restrições de Integridade (RIs)

```
-- Não Nulo
not null
-- Atributo(s) forma(m) uma chave candidata
unique(A1,...,An)
-- PK
primary key (A1, ..., An)
-- FK
foreign key (Am, ..., An) references r
```

Exemplo:

```
create table Departamento (
    Nome_Depto varchar(50) not null,
    ID_Depto numeric(5,0),-----
    primary key (ID_Depto) ); -----

create table Professor (
    ID_Professor numeric(5,0),-----
    Nome varchar(50) not null,
    CPF char(11), -----
    Salario numeric(8,2),
    ID_Depto numeric(5,0),
    unique (CPF), -----
    primary key (ID_Professor), -----
    constraint fk_depto_prof foreign key (ID_Depto) references
```



```
Departamento(ID_Depto) );  
-- foreign key (ID_Depto) references Departamento(ID_Depto) );
```

## Drop table

```
drop table r
```

## Alterar Tabela

```
alter table r add A D  
-- Onde A é o nome do atributo a ser adicionado na relação r e D é o domínio de A  
-- Todas as tuplas na relação são associados valores nulos como valor do novo atributo  
alter table r drop A  
-- Onde A é o nome do atributo da relação r a ser removido  
-- Remoção de atributos não é suportado por muitos SGBDs
```

## Exemplo

```
create table Departamento (  
    Nome_Depto varchar(50),  
    ID_Depto numeric(5,0) );  
  
alter table Departamento add Data_criacao date;  
  
alter table Departamento add primary key (ID_Depto);  
  
alter table Departamento drop primary key;
```

```
create table Professor (  
    ID_Professor numeric(5,0),  
    Nome varchar(50) not null,  
    CPF char(11),  
    Salario numeric(8,2),  
    ID_Depto numeric(5,0),  
    unique (CPF),  
    primary key (ID_Professor));  
  
alter table Professor add constraint fk_depto_prof foreign key (ID_Depto)  
references Departamento(ID_Depto);  
  
alter table Professor drop foreign key fk_depto_prof;
```

## Restrições de atributos e domínios

```
not null  
default <valor>
```

```
check <condição>
```

## Exemplo

```
ID_Depto int not null
check (ID_Depto>0 and ID_Depto<=99999)

semestre varchar(6) default 'Summer' check (semestre in ('Fall',
'Winter', 'Spring', 'Summer'));

create domain D_NUM as integer
check (D_NUM > 0 and D_NUM < 21);
ID_Depto D_NUM not null;
```

## Restrições de integridade referencial

```
-- Remoção
on delete
cascade (propagação)
set null (substituição por nulos)
set default (substituição por um valor default)
-- Opção default: bloqueio (restrict)

-- As mesmas opções se aplicam à cláusula
on update (alteração)
```

## Select

```
select A1, A2, ..., An
from r1, r2, ..., rm
where P

-- Ai representa um atributo
-- ri representa uma relação
-- P é um predicado

select * from r
-- * denota todos os atributos
-- r representa uma relação
```

## Modificações do banco de dados

```
insert -- inserir

insert into Departamento
values ('Departamento de Informática', 21);
```

```

insert into Professor (Nome, CPF, ID_Professor)
values ('Costa', 33333333333, 3);

--
update -- alterar

update Professor
set Salario=10000, ID_Depto=21
where ID_Professor=3;

update Professor
set Salario=Salario*1.1
where ID_Depto in (select ID_Depto from Departamento
where Nome_Depto='Departamento de Informática');

--
delete -- remover

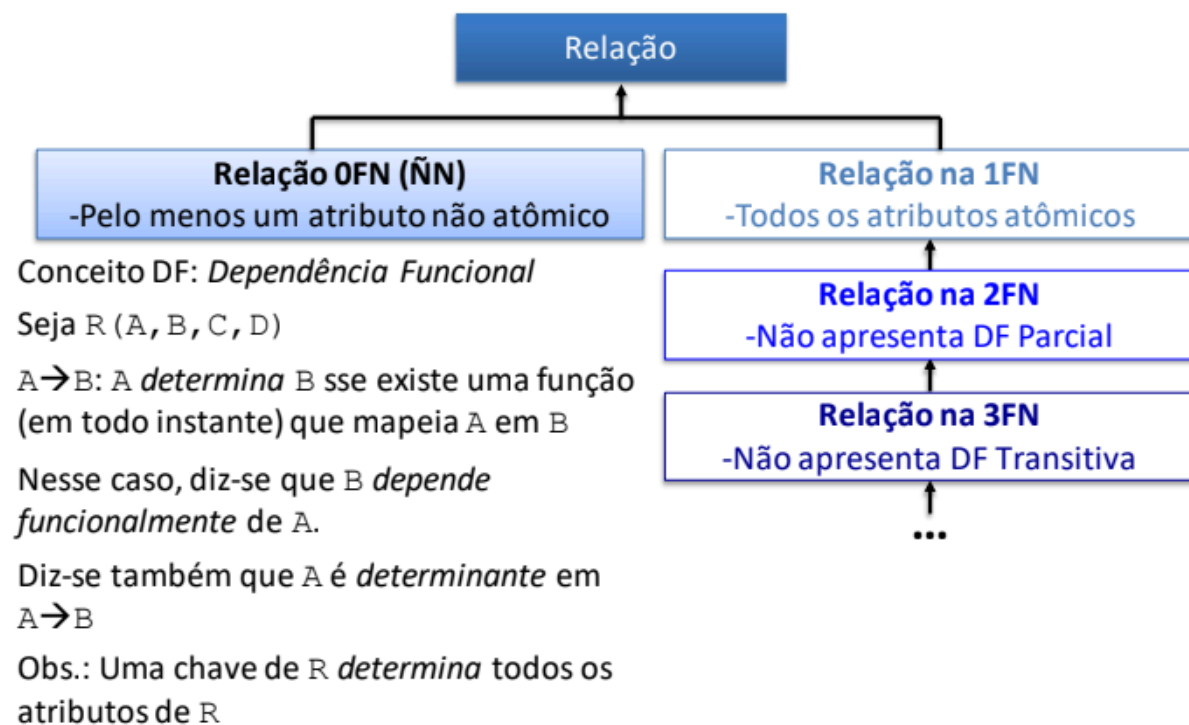
delete from Professor;

delete from Professor where ID_Professor=3;

delete from Professor where ID_Depto in
(select ID_Depto from Departamento
where Nome_Depto='Departamento de Informática');

```

## Normalização



## 0FN ou ÑN

Uma relação está na 0FN se ela apresentar algum atributo não atômico

**Atributos não-atômicos**  
(multivalorados)



### **Departamento**

<u>ID_Depto</u>	Nome_Depto	Tel_Secret_Depto	Disciplina		
			<u>ID_Disc</u>	Nome	Créditos
DCC	Departamento de Ciência da Computação	3938-3393	MAB605	Recuperação da Informação	4
			MAB112	Sistemas de Informação	4
			MAB120	Computação I	6
			MAB489	Banco de Dados I	4

## **1FN**

Uma relação está na 1FN se todos os seus atributos forem atômicos

### **Departamento (>=1FN)**

<u>ID_Depto</u>	Nome_Depto	Tel_Secret_Depto
DCC	Departamento de Ciência da Computação	3938-3393

### **Disciplina (>=1FN)**

<u>ID_Disc</u>	Nome	Créditos	<u>ID_Depto</u>
MAB605	Recuperação da Informação	4	DCC
MAB112	Sistemas de Informação	4	DCC
MAB120	Computação I	6	DCC
MAB489	Banco de Dados I	4	DCC

## **2FN**

Uma relação está na 2FN se ela estiver na 1FN e se ela **não** apresentar dependências funcionais (DFs) parciais da chave

Parcial = "de uma parte"

## AlunoMatriculaDisciplina

Ex. DFs parciais da chave

<u>DRE</u>	<u>Cod_Disc</u>	<u>Ano_Sem</u>	Nome_Aluno	Nome_Disc	Créditos	Nota	Data_nasc
------------	-----------------	----------------	------------	-----------	----------	------	-----------

1FN

## Aluno

<u>DRE</u>	Nome_Aluno	Data_nasc
------------	------------	-----------

## Disciplina

<u>Cod_Disc</u>	Nome_Disc	Créditos
-----------------	-----------	----------

## Matricula

<u>DRE</u>	<u>Cod_Disc</u>	<u>Ano_Sem</u>	Nota
------------	-----------------	----------------	------

## 3FN

Uma relação está na 3FN se ela estiver na 2FN e se ela **não** apresentar dependências funcionais (DFs) transitivas da chave

## DisciplinaDepartamento

Ex. DF transitiva da chave

<u>Cod_Disc</u>	Nome_Disc	Créditos	ID_Depto	NomeCoord_Depto
-----------------	-----------	----------	----------	-----------------

## Departamento

<u>ID_Depto</u>	NomeCoord_Depto
-----------------	-----------------

## Disciplina

<u>Cod_Disc</u>	Nome_Disc	Créditos	ID_Depto
-----------------	-----------	----------	----------

## Resumo

0FN



(Eliminar atributos multivalorados)

1FN



(Eliminar dependência funcional parcial)

2FN



(Eliminar dependência funcional transitiva)

3FN