

FAQI

FACULDADE QI BRASIL

**AQUI TEMOS O VERDADEIRO
BANCO DE DADOS**



BA DUM TSS



Banco de Dados

Contatos

- ▣ Silvio.viegas@qi.edu.br
- ▣ Silvio Cesar Viegas



Modelo Relacional

- Introduzido por Codd em 1970
- Modelo com uma sólida base formal
 - Conceitos Simples
 - relações, atributos, tuplas
- Não considera aspectos físicos de armazenamento, acesso e desempenho
- **Base para a maioria dos SGBDs que dominam o mercado**

Modelo Relacional - Características

- Organização dos dados

- conceitos do modelo: atributo, relação, chave, ...

Exemplo: Tupla

Tupla 1



Nome	CPF	DataNasc
Renata	01035	12/11/1980
Vânia	02467	03/07/1976
Maria	01427	20/02/1985

Atributo: Nome
Valor: Renata

Exemplo: Tupla

Tuplas



Nome	CPF	DataNasc
Renata	01035	12/11/1980
Vânia	02467	03/07/1976
Maria	01427	20/02/1985

Exemplo Entidade

Cabeçalho

Aluno

Nome	CPF	Endereço	DataNasc
Renata	01035	Rua das Flores, 210	12/11/1980
Vânia	02467	Capote Valente, 35	03/07/1976
Maria	01427	São Diego 310/34	20/02/1985

Exemplo Relação

Aluno

Nome	CPF	Endereço	DataNasc
Renata	01035	Rua das Flores, 210	12/11/1980
Vânia	02467	Capote Valente, 35	03/07/1976
Maria	01427	São Diego 310/34	20/02/1985

Corpo

Relembrando...

- ▣ Entidade ou Relação é uma Tabela
- ▣ Relacionamento **pode ser** uma Tabela
- ▣ Atributo é um Campo (coluna da tabela)
- ▣ Tupla é uma linha da tabela
- ▣ Valor é o dado de um atributo em si

Chave

- Conjunto de um ou mais atributos de uma relação
 - Chave Primária (primary key) - PK
 - Chave Candidata
 - Chave Alternativa
 - Chave Estrangeira (foreign key) - FK

Chave Primária

- Primary key (PK)

- atributo(s) cujo (conjunto de) valor(es) identifica(m) unicamente uma tupla em uma relação
- **Unicidade de valores** na coluna que compõe a chave

Chave Primária (PK)

Aluno

Nome	Mat	Endereço	DataNasc
Renata	01035	Rua das Flores, 210	12/11/1980
Vânia	02467	Capote Valente, 35	03/07/1976
Maria	01427	São Diego 310/34	20/02/1985

Qual(is) atributo(s) representam unicamente uma tupla?

Chave Primária (PK)

Aluno

Nome	Mat.	Endereço	DataNasc
Renata	01035	Rua das Flores, 210	12/11/1980
Vânia	02467	Capote Valente, 35	03/07/1976
Maria	01427	São Diego 310/34	20/02/1985

Qual(is) atributo(s) representam unicamente uma tupla?

Mat.

Chave Primária (PK)

Aluno

Nome	CPF	Endereço	DataNasc
Renata	701034263890	Rua das Flores, 210	12/11/1980
Vânia	693529876987	Capote Valente, 35	03/07/1976
Maria	347685784432	São Diego 310/34	20/02/1985

Aluno(CPF, Nome, Endereço, DataNasc)



Chave Primária (PK) - Composta

Alocação (Cod_Projeto, Cod_Func, DataIni, Tempo)

- Um funcionário pode estar em mais de um projeto
- Normalmente são atributos formados a partir de Relacionamentos

Chave Primária (PK)

Alocação (Cod_Projeto, Cod_Func, DataIni, Tempo)

Chave primária composta



Chave Candidata

- Possui as mesmas propriedades que a chave primária

Aluno

Nome	Matrícula	CPF	DataNasc
Renata	01035	701034263890	12/11/1980
Vânia	02467	693529876987	03/07/1976
Maria	01427	347685784432	20/02/1985

Chave Candidata

- Possui as mesmas propriedades que a chave primária

Aluno

Nome	Matrícula	CPF	DataNasc
Renata	01035	701034263890	12/11/1980
Vânia	02467	693529876987	03/07/1976
Maria	01427	347685784432	20/02/1985



Chaves candidatas

Chave Candidata

- Qual escolher para Chave Primária?
- Escolhe-se para chave primária aquela com o atributo único ou menor número de caracteres

Nome	Matrícula	CPF	DataNasc
Renata	01035	701034263890	12/11/1980
Vânia	02467	693529876987	03/07/1976
Maria	01427	347685784432	20/02/1985



Chave Candidata

Chave Primária

Nome	Matrícula	CPF	DataNasc
Renata	01035	701034263890	12/11/1980
Vânia	02467	693529876987	03/07/1976
Maria	01427	347685784432	20/02/1985

Chave Alternativa

Chave Primária

Nome	Matrícula	CPF	DataNasc
Renata	01035	701034263890	12/11/1980
Vânia	02467	693529876987	03/07/1976
Maria	01427	347685784432	20/02/1985

Chave alternativa

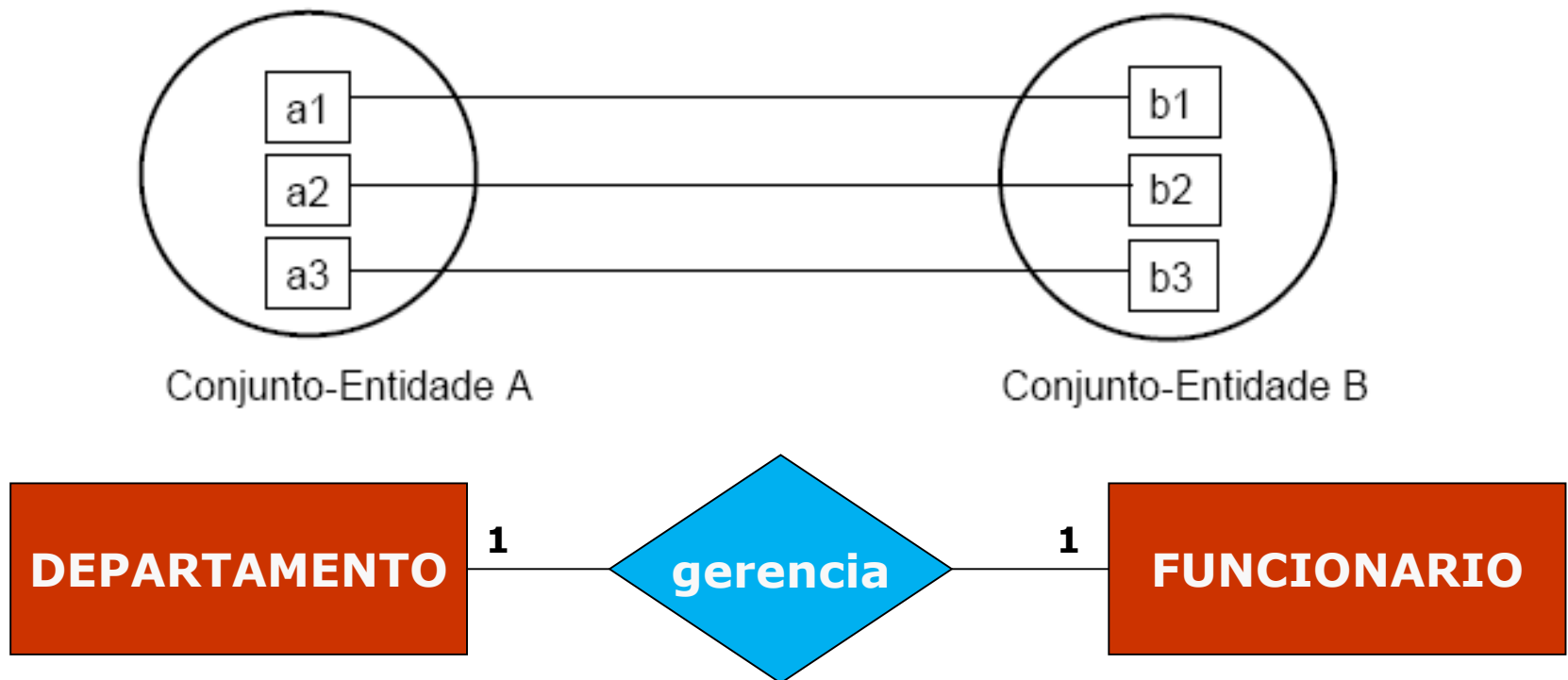
Chave alternativa: chave candidata que não é primária

Chave Estrangeira

- Foreign Key (**FK**)
- Atributo(s) de uma relação, cujos valores devem obrigatoriamente aparecer na **chave primária** de uma **relação** (da mesma ou de outra)
- Implementa o relacionamento em um BD relacional
- Advêm da cardinalidade dos relacionamentos

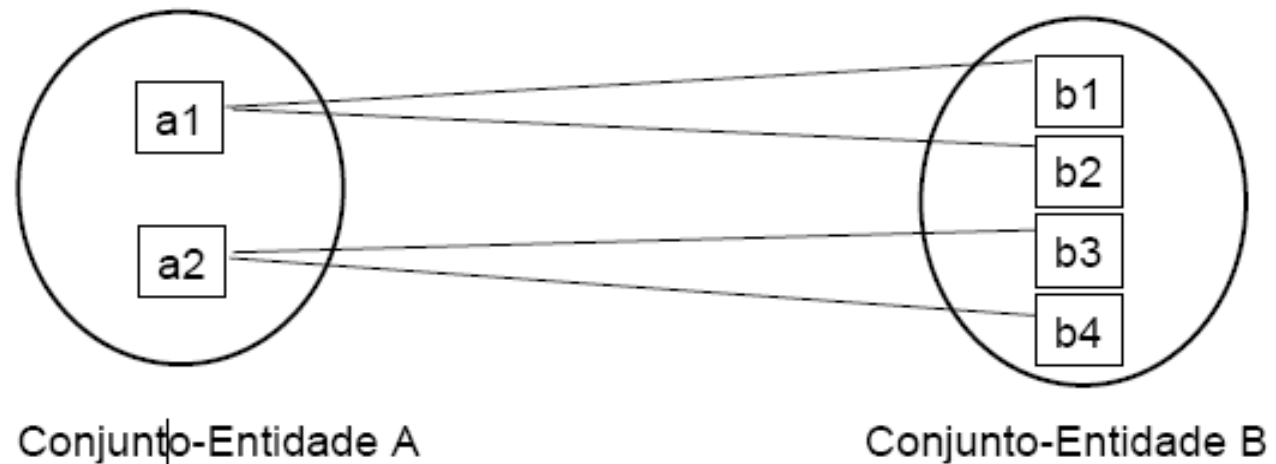
Chave Estrangeira (FK)

- Um-para-um. Uma entidade em A está associada no máximo a uma entidade em B e uma entidade em B está associada no máximo a uma entidade em A



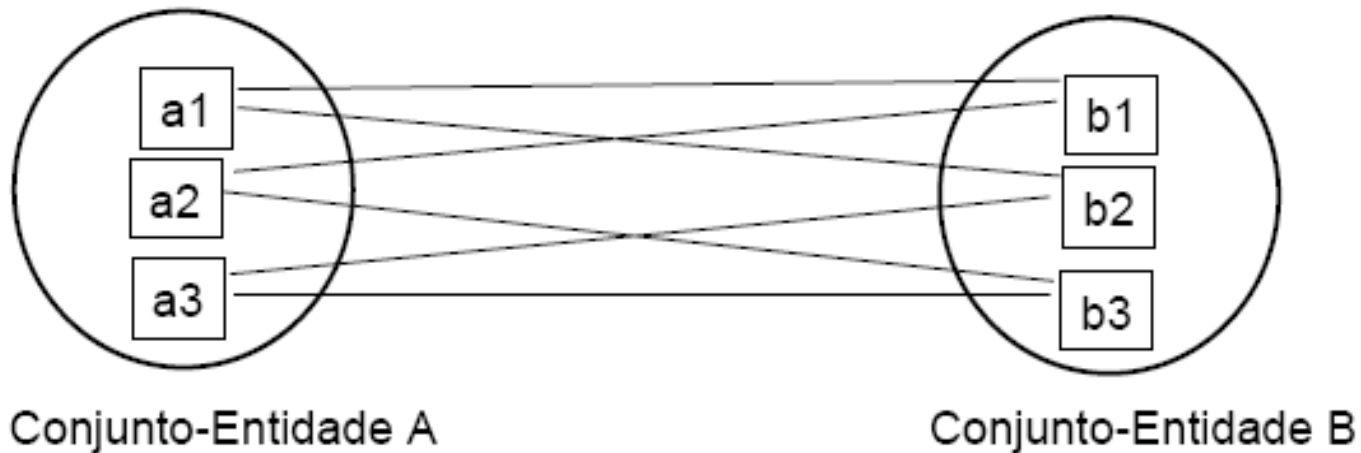
Chave Estrangeira (FK)

- Um-para-muitos. Uma entidade em A está associada a qualquer número de entidades em B, enquanto uma entidade em B está associada no máximo a uma entidade em A



Chave Estrangeira (FK)

- **Muitos-para-muitos.** Uma entidade em A está associada a qualquer número de entidades em B, e uma entidade em B está associada a qualquer número de entidades em A.



Chave Estrangeira (FK)


Nome	Matrícula	CPF	Curso
Renata	01035	701034263890	1
Vânia	02467	693529876987	2
Maria	01427	347685784432	1



Chave Estrangeira (FK)

Nome	Matrícula	CPF	Cod_c
Renata	01035	701034263890	1
Vânia	02467	693529876987	2
Maria	01427	347685784432	1

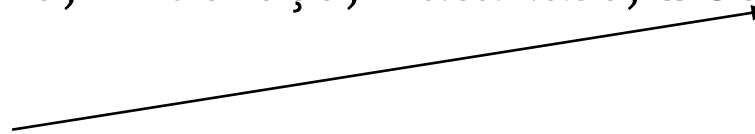
Cod_c	Descrição
1	Ciência da Computação
2	Administração de Empresas
3	Ciências Jurídicas e Sociais



Chave Estrangeira (FK)

Aluno(CPF, Nome, Endereço, DataNasc, #Cod_c)

Curso (Cod_c, Descrição)



Chave Estrangeira (FK)

Funcionário (CodFunc, Nome, Endereço, #Cod_Chefe)



A diagram showing a horizontal line with a vertical arrow pointing up to the 'CodFunc' column header and a vertical line ending in a crow's foot symbol pointing to the 'Cod_Chefe' column header, indicating a foreign key relationship.

CodFunc	Nome	Endereço	Cod_Chefe
1	Maria	Av. Joaquim 2	4
2	João	Oscar Freire, 10	3
3	Pedro	Anita Garibaldi, 12	1
4	Carla	Carlos Gomes, 50	2

Revisando...

- Chave Primária (PK)
- Chave candidata
- Chave alternativa
- Chave Estrangeira (FK)

Funcionalidades de um SGBD



Linguagens de SGBD

- DDL (Data Definition Language)
 - Linguagem de definição de dados
 - Especifica o **esquema** do BD

ALUNO

Nome	NumerodoAluno	Turma	Curso_Hab
------	---------------	-------	-----------

CURSO

NomedoCurso	NumerodoCurso	Creditos	Departamento
-------------	---------------	----------	--------------

PRE_REQUISITO

NumerodoCurso	NumerodoPre_requisito
---------------	-----------------------

DISCIPLINA

Identificador_Disciplina	NumerodoCurso	Semestre	Ano	Instrutor
--------------------------	---------------	----------	-----	-----------

RELATORIO_DE_NOTAS

NumerodoAluno	Identificador_Disciplinas	Nota
---------------	---------------------------	------

SQL - Criação de Tabelas

- Comando *Create table* (sintaxe)

```
CREATE TABLE <tabela>  
  
(  
  
<descrição das colunas>,  
  
<descrição das chaves>  
  
);
```

SQL - Criação de Tabelas

- Comando *Create table* (mais exemplos)

CREATE TABLE cliente

(
cod_cliente number(5),
nome varchar2(70) NOT NULL,
sexo char(1),
cidade varchar2(70),
estado char(2),
telefone number(7),
PRIMARY KEY (cod_cliente)

Tipo de dado 'number'

Tamanho de dado '5'

9 9 9 9 9

(valor máximo a ser representado)

Restrições de dado 'PRIMARY KEY'

Linguagens de SGBD

- DML (Data Manipulation Language)
 - Linguagem de manipulação de dados
 - Manipulação as **instâncias** dos Banco de Dados

ALUNO	Nome	Numero	Turma	Curso_Hab
	Smith	17	1	CC
	Brown	8	2	CC

CURSO	NomedoCurso	NumerodoCurso	Creditos	Departamento
	Introdução à Ciência da Computação	CC1310	4	CC
	Estruturas de dados	CC3320	4	CC
	Matemática Discreta	MAT2410	3	MATH
	Banco de dados	CC3380	3	CC

DISCIPLINA	IdentificadordeDisciplina	NumerodoCurso	Semestre	Ano	Instrutor
	85	MAT2410	Segundo Semestre	98	King
	92	CC1310	Segundo Semestre	98	Anderson
	102	CC3320	Primeiro Semestre	99	Knuth
	112	MAT2410	Segundo Semestre	99	Chang
	119	CC1310	Segundo Semestre	99	Anderson
	135	CC3380	Segundo Semestre	99	Stone

SQL - Inserção de Dados

□ Comando *Insert* (exemplos)

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(1,'sal','kg',0.85);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(13,'ouro','g',6.18);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(87,'cano','m',1.97);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(77,'papel','m',1.05);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(30,'açúcar','sac',0.99);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(45,'madeira','m',0.25);
```

SQL - Alteração de Tabelas

□ Comando *Alter table* (exemplos)

- Para adicionar a coluna *data_nascimento* na tabela cliente

ALTER TABLE cliente **ADD** (data_nascimento date);

- Para excluir a coluna *data_nascimento* na tabela cliente

ALTER TABLE cliente **DROP** (data_nascimento);

- Para modificar o tamanho da coluna *telefone* na tabela telefones_vendedor

ALTER TABLE telefones_vendedor

MODIFY (telefone number(9));

SQL - Atualização de Dados

□ Comando *Update* (exemplos)

```
UPDATE cliente  
SET     nome = 'Esdras Ricardo'  
WHERE  cod_cliente = 17;
```

```
UPDATE vendedor  
SET     salario_fixo = 400  
WHERE  cod_vendedor = 29;
```

```
UPDATE produto  
SET     unidade = 'kg', valor_unitario = 6.38  
WHERE  cod_produto = 13;
```

Se eu quiser atualizar **mais** de um campo ao mesmo tempo?

Linguagens de SGBD

□ Linguagem de Consulta

- Porção da linguagem de manipulação que envolve o resgate de informações

```
select * from pessoa;
```

CODIGO	NOME	NASCIMENTO	TELEFONE
1	Almir Moura	06/05/83	8132365858
21	Marcos Cardoso	25/04/82	8187698585
41	Marcelo dos Santos	25/07/85	-

```
select * from pessoa where nome like 'Mar%';
```

CODIGO	NOME	NASCIMENTO	TELEFONE
21	Marcos Cardoso	25/04/82	8187698585
41	Marcelo dos Santos	25/07/85	-

```
select nome, telefone from pessoa where nascimento = '06/05/83';
```

NOME	TELEFONE
Almir Moura	8132365858

Exercícios

1) Associe corretamente de acordo com as linguagens de SGBD

a) DML

b) DDL

() Permite a alteração das instâncias do BD

() Permite definir um esquema de BD

() Permite alterar um esquema de BD

() Permite inserir uma instância no BD

Exercícios (1/2)

1) Associe corretamente de acordo com as linguagens de SGBD

a) DML

b) DDL

() Permite a alteração das instâncias do BD

() Permite definir um esquema de BD

() Permite alterar um esquema de BD

() Permite inserir uma instância no BD

Exercícios (2/2)

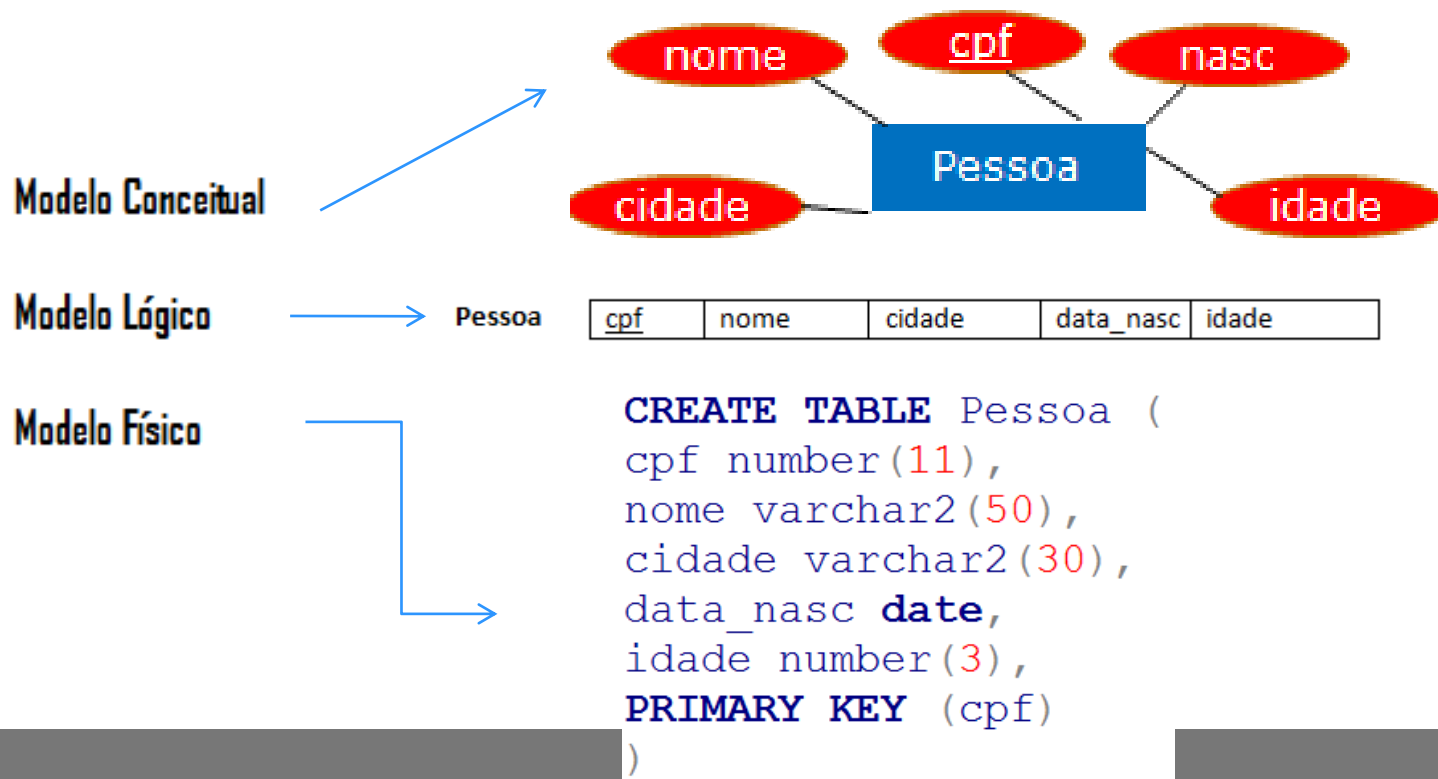
2) Como são classificados os usuários de um SGBD?

Modelagem de Dados



Modelagem de Dados

- Objetivo -> transformar aspectos do Mundo Real em um Modelo de Dados Formal (que pode ser gráfico ou textual)
 - Exemplo: Uma pessoa tem um cpf, nome, cidade, data de nascimento e idade (MUNDO REAL)



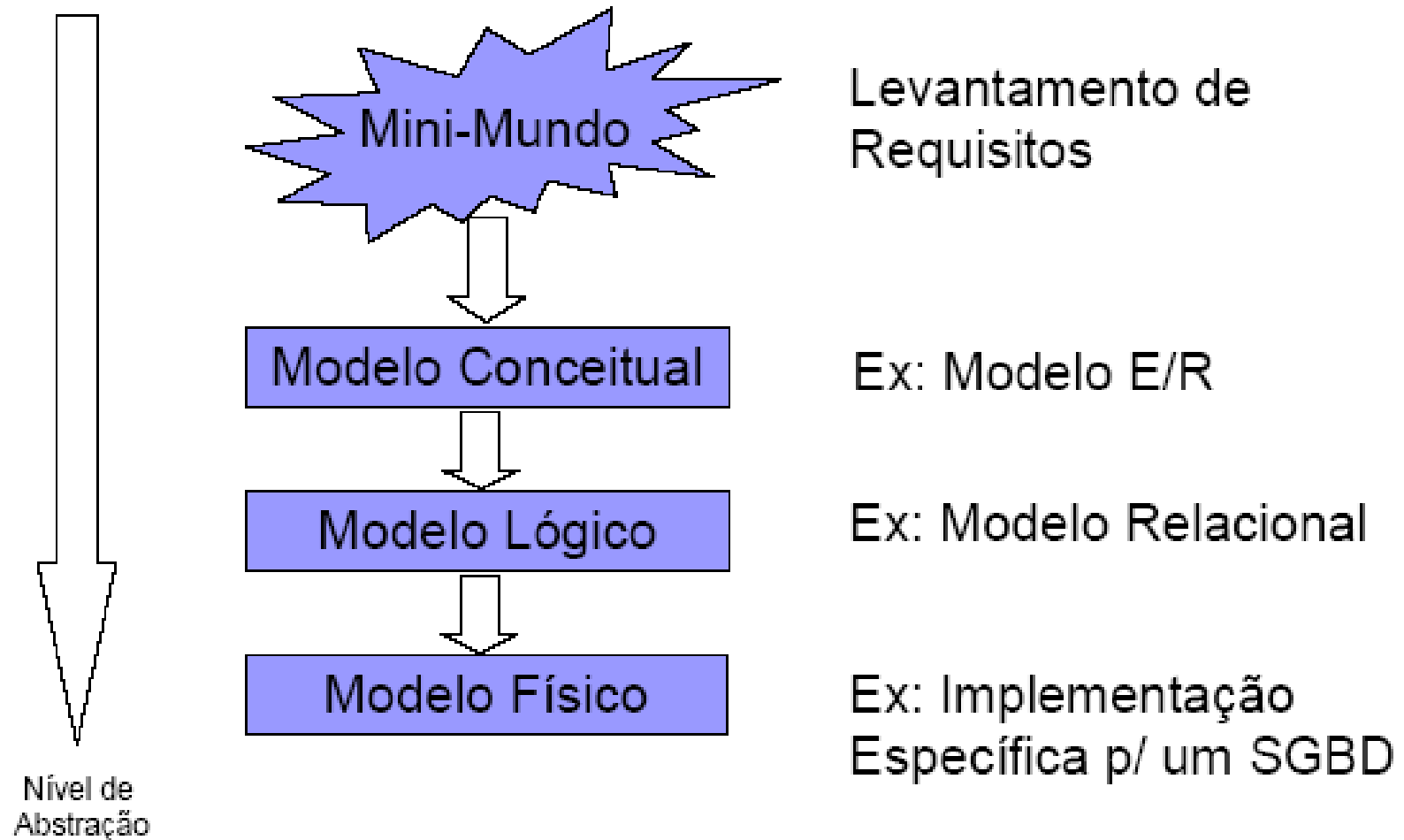
Modelo Conceitual



Modelo Entidade- Relacionamento



Modelagem de Dados



Introdução

- ▣ O modelo E-R é uma **Modelagem Conceitual** de Dados.
- ▣ O objetivo é obter resultados e esquemas puramente conceituais sobre a essência de um sistema.
- ▣ Três noções básicas: **Entidades**, **Atributos** e **Relacionamentos**

Modelo Entidade-Relacionamento

- ▣ Modelo baseado na percepção do mundo real, que consiste em um conjunto de objetos básicos chamados **entidades** e nos **relacionamentos** entre esses objetos
- ▣ Facilita o projeto de banco de dados, possibilitando a especificação da estrutura lógica geral do bd

Introdução

▣ Entidades

- **Objetos** que existem no mundo real com uma identificação distinta e com um significado próprio.
- São “coisas” que existem no negócio, ou ainda, descrevem o negócio em si.



CLIENTE



FUNCIONÁRIO



CONTA (de
banco)

Introdução

▣ Atributos

- Todo objeto para ser uma entidade possui atributos e seus valores.
- Vamos imaginar um funcionário de uma empresa. **O que caracteriza o funcionário?**

Introdução



▣ Atributos

• Entidade: Funcionário

MATRÍCULA	NOME	DATA DE ADMISSÃO
0001	José	25/05/2003
0002	Maria	01/09/2003
0003	João	11/10/2003
0004	Pedro	19/11/2003

← ATRIBUTOS

Introdução

□ Relacionamento

- Associação entre uma ou mais entidades.

- Exemplos:

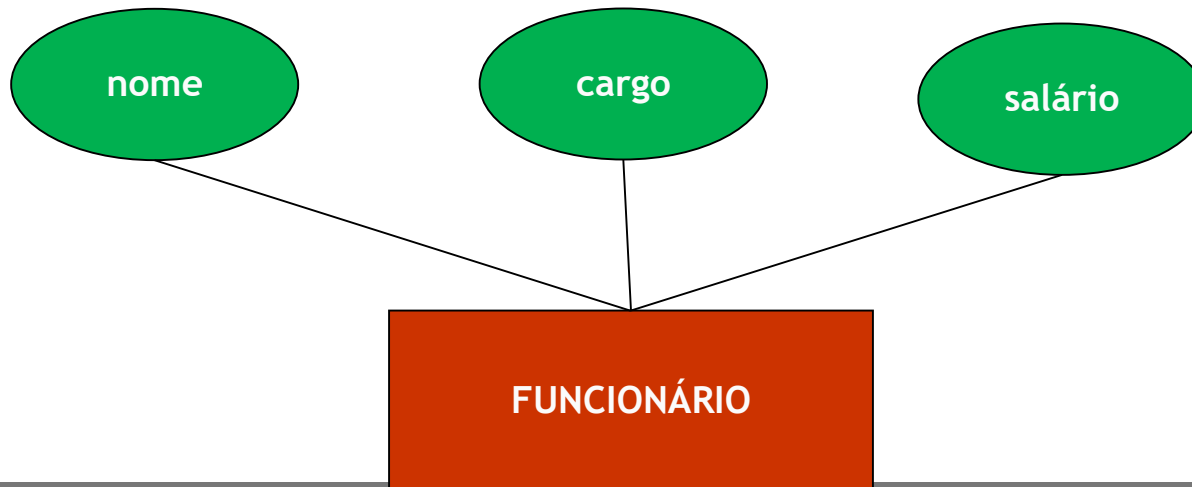
- aluno FAZ prova
- professor LECIONA disciplina
- cliente REALIZA pedido
- lojista VENDE roupa

Componentes do Diagrama E-R

- ▣ **Retângulos**: representam as entidades

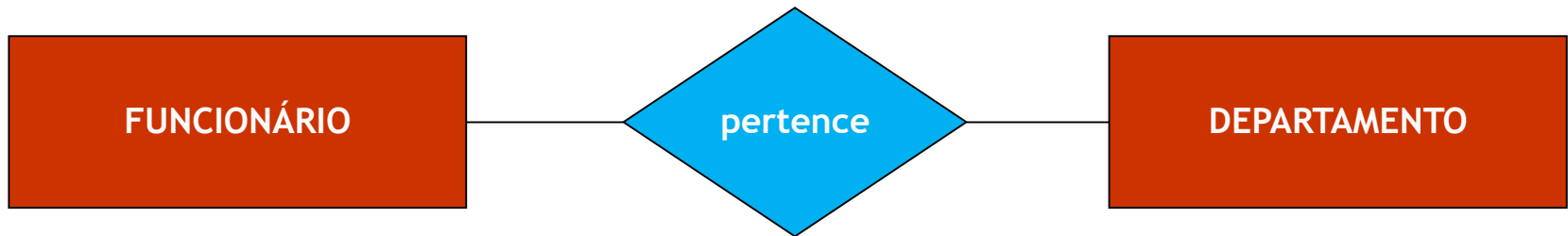


- ▣ **Elipses**: representam atributos



Componentes do Diagrama E-R

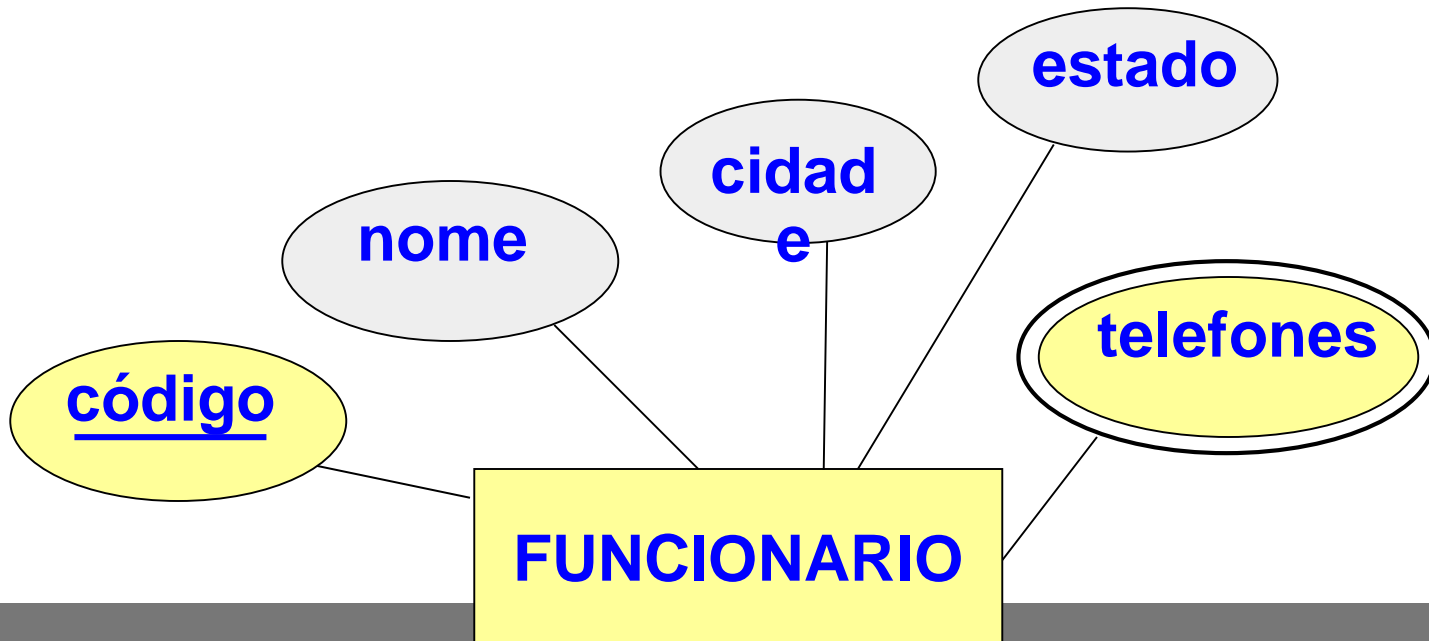
- ▣ **Losangos:** representam os relacionamentos



- ▣ **Linhas:** ligam atributos a entidades e entidade a relacionamentos

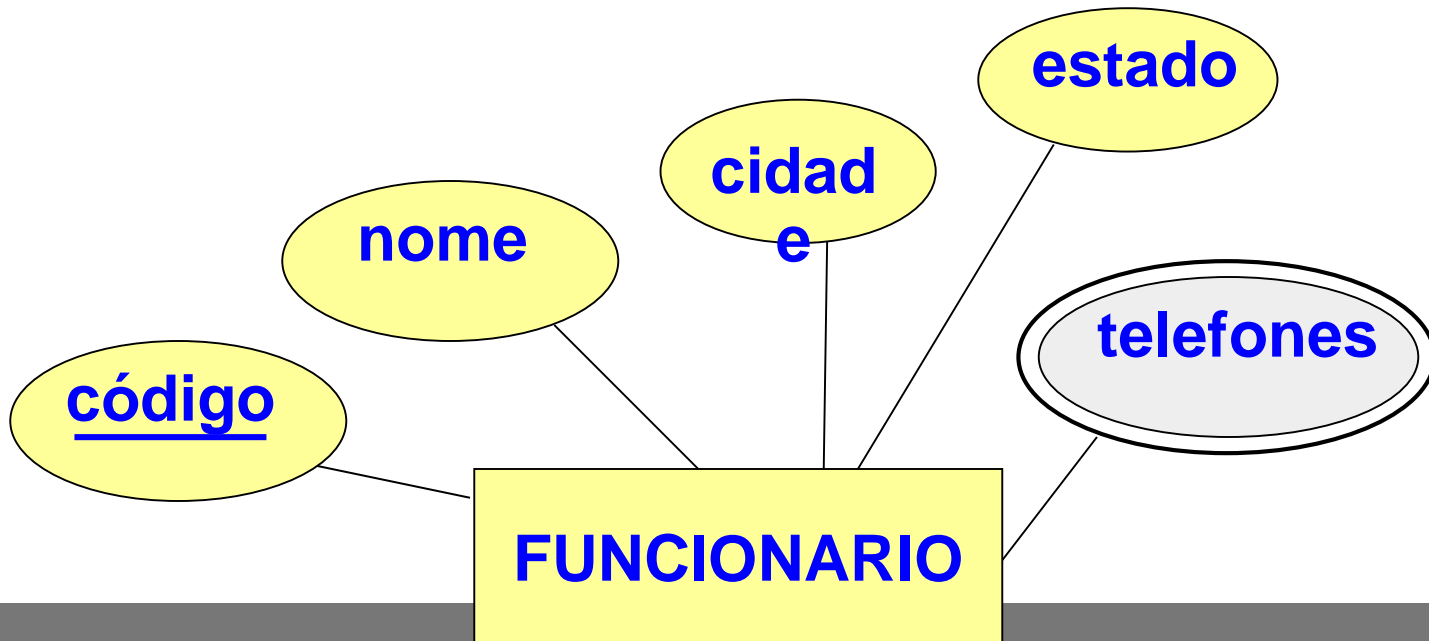
Tipos de Atributos

- ▣ Atributo **Monovalorado**: assume um único valor para cada elemento da entidade.
 - Exemplos: nome, cidade, estado



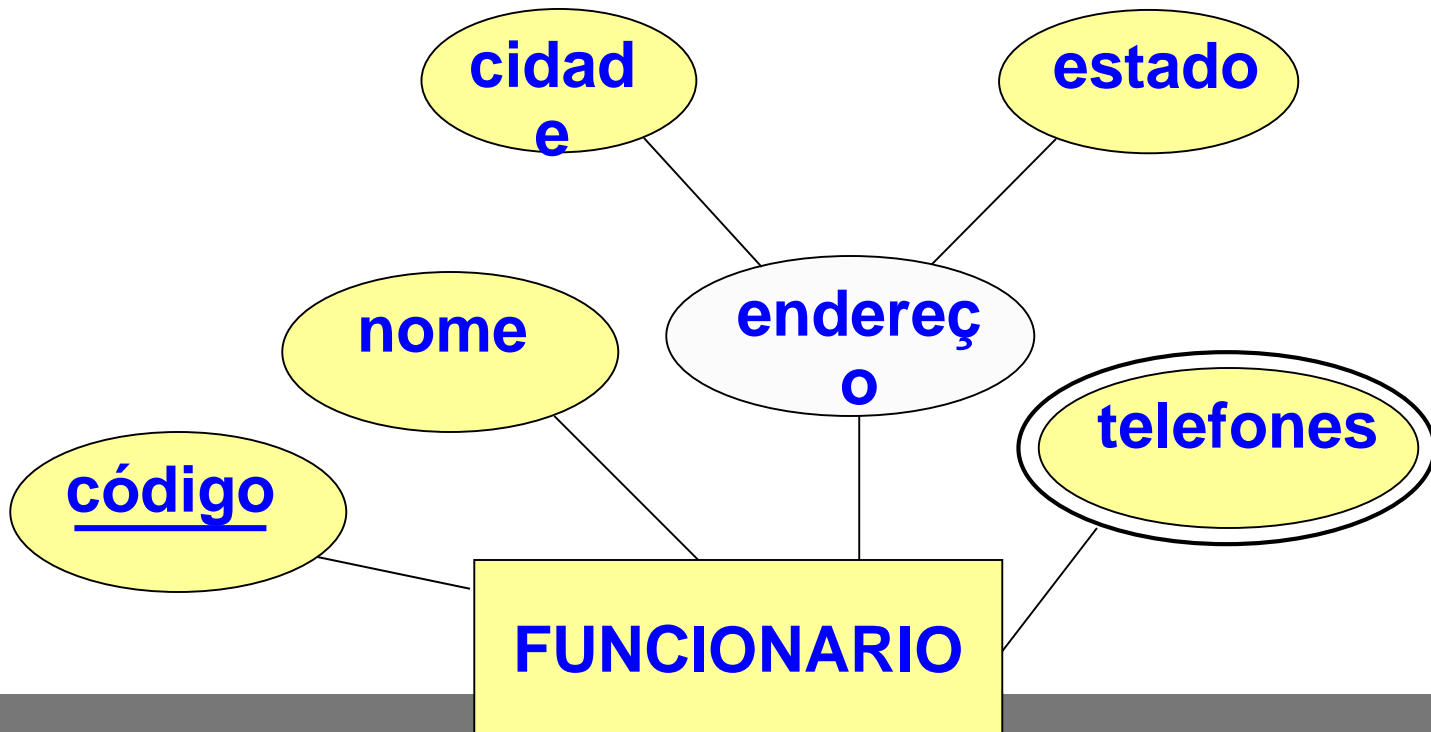
Tipos de Atributos

- ▣ Atributo **Multivalorado**: uma única entidade tem diversos (vários, múltiplos) valores para este atributo.
 - Exemplo: telefones



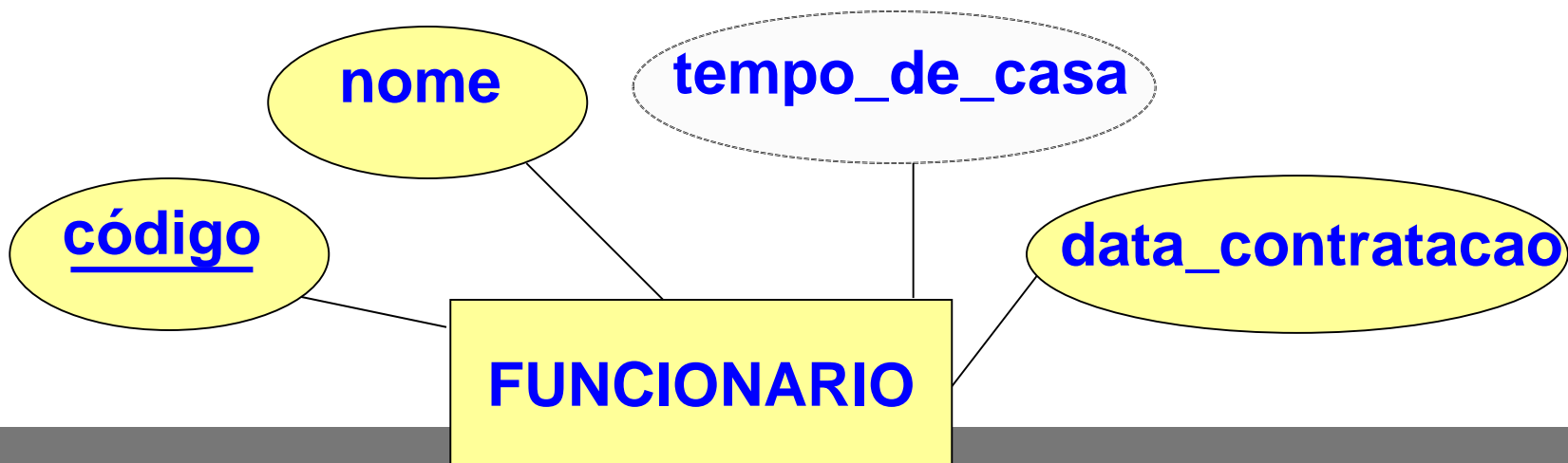
Tipos de Atributos

- ▣ Atributo **Composto**: formado por um ou mais sub-atributos.
 - Exemplo: endereço



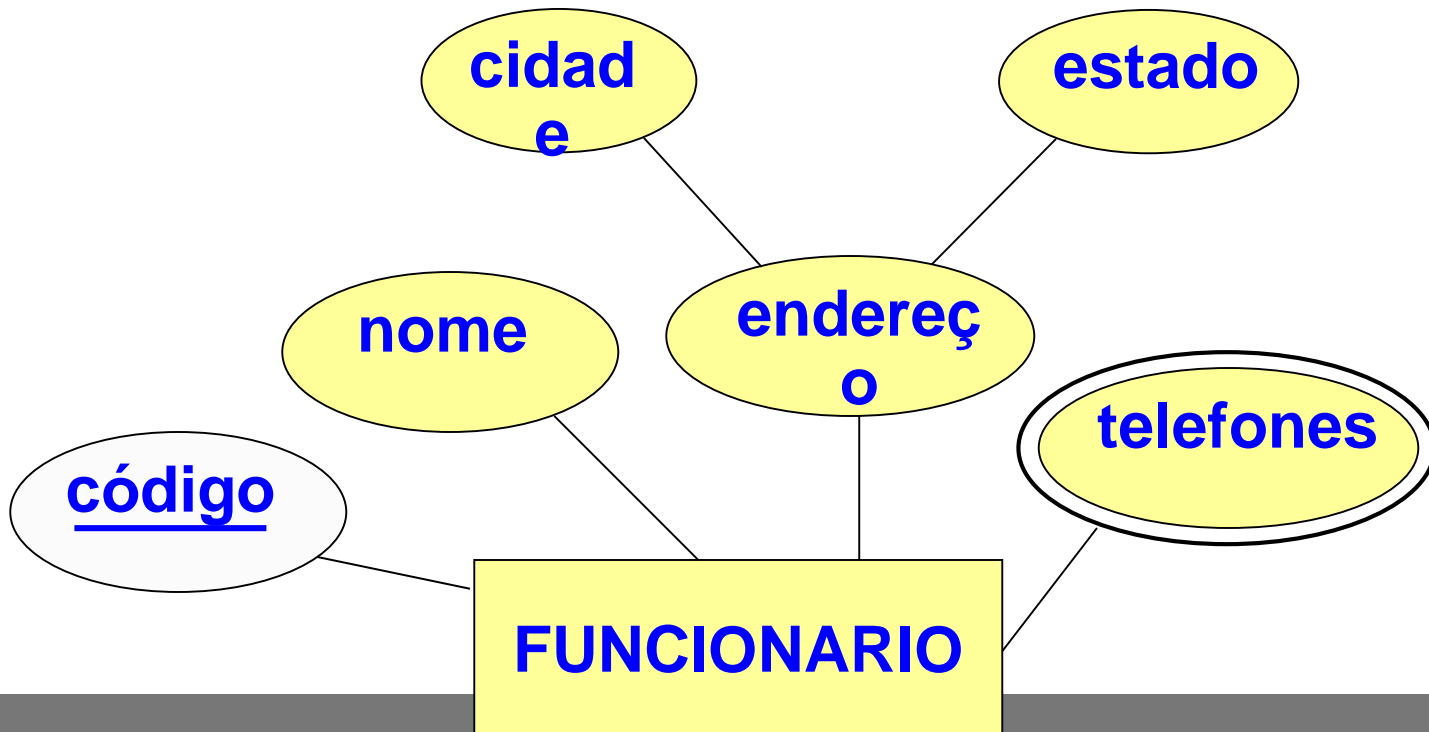
Tipos de Atributos

- ▣ **Atributo Derivado**: o valor deste tipo de atributo pode ser derivado de outros atributos a ele relacionados.
 - Exemplo: tempo de casa



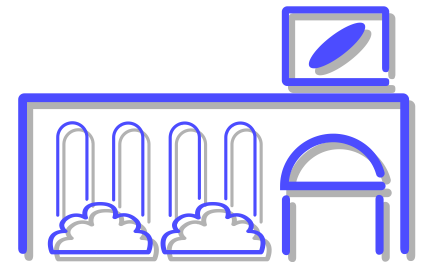
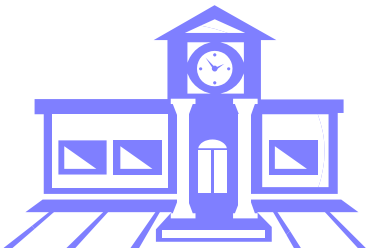
Tipos de Atributos

- ▣ Atributo **Chave**: identificador de uma entidade (*também conhecido como atributo **determinante***)
 - Exemplo: código



Introdução

- Mini-mundo (Especificação dos UC's)
 - A base para a especificação da estrutura conceitual do banco de dados.
 - Principais necessidades de uma organização (banco, hospital, escola, supermercado)
 - Normalmente são substantivos.



Estudo de Caso: Um Supermercado

- O Supermercado possui vários **funcionários**. Esses funcionários são identificados por um *código*. Ainda serão registrados nesses funcionários o *nome* de cada um, o *cargo* que eles ocupam, o *salário* e o *telefone*.

Estudo de Caso: Um Supermercado

- ❑ O Supermercado ainda deseja guardar as informações de seus **departamentos**. Eles devem possuir um *código* para identificá-los, um *nome*, uma *localização* e os *telefones* disponíveis do mesmo para contato.

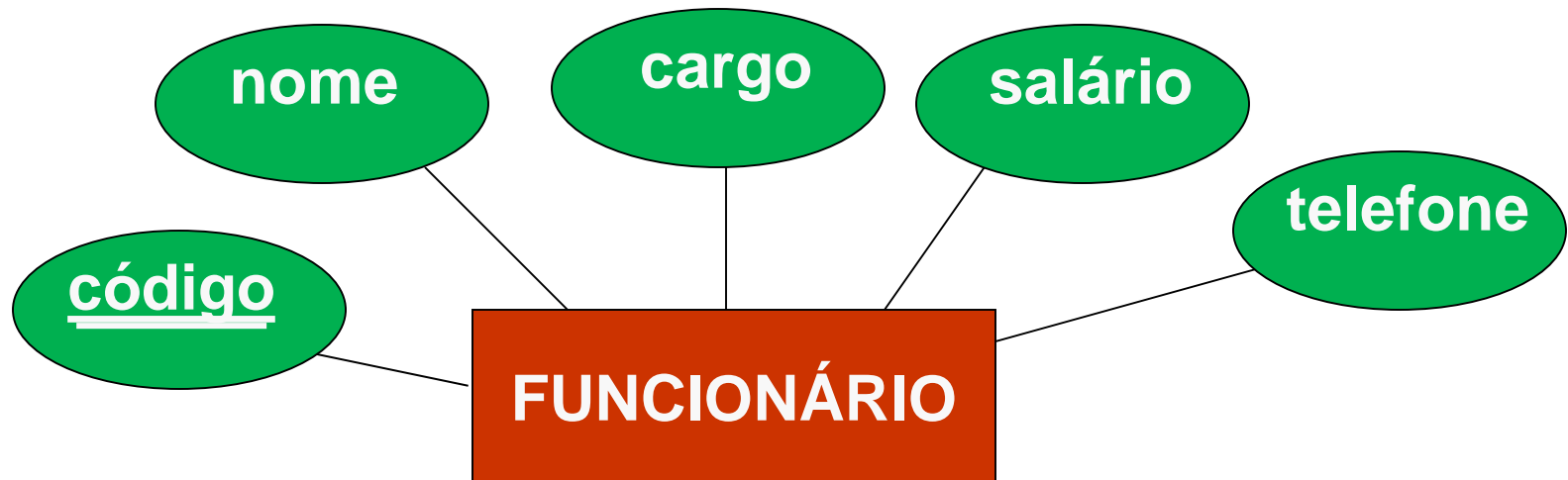
Estudo de Caso: Um Supermercado

Próximo Passo: Identificar as entidades e seus atributos

Estudo de Caso: Um Supermercado

▣ Relembrando:

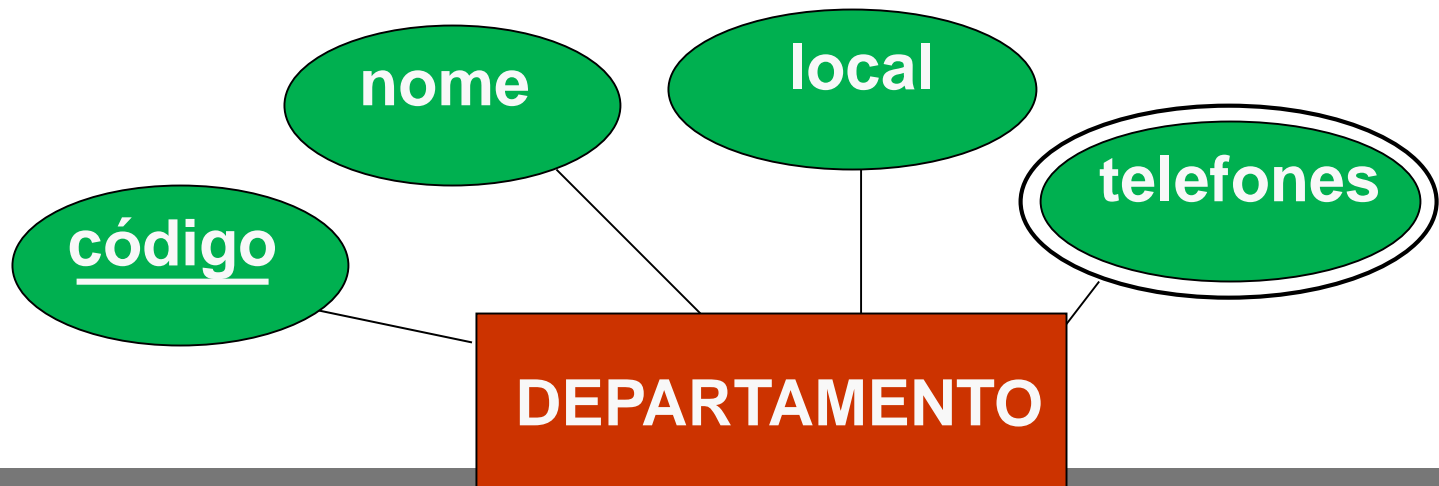
- O Supermercado possui vários **funcionários**. Esses funcionários são identificados por um *código*. Ainda serão registrado nesses funcionários o *nome* de cada um, o *cargo* que eles ocupam, o *salário* e o *telefone*.



Estudo de Caso: Um Supermercado

▣ Relembrando:

- O Supermercado ainda deseja guardar as informações de seus **departamentos**. Eles devem possuir um *código* para identificá-los, um *nome*, uma *localização* e os *telefones* disponíveis do mesmo para contato.



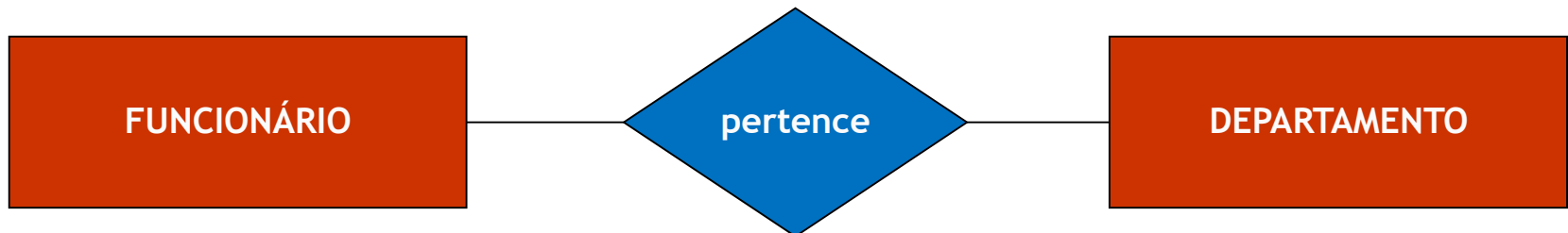
Estudo de Caso: Um Supermercado

Passo seguinte: Identificar os relacionamentos

Estudo de Caso: Um Supermercado

□ Relembrando:

- Um **FUNCIONARIO** pode **pertencer** a no máximo um **DEPARTAMENTO**, e cada **DEPARTAMENTO** pode ter um ou mais **FUNCIONARIOS** associados.



Estudo de Caso: Um Supermercado

□ Relembrando:

- Um **FUNCIONARIO** pode **atender** um ou mais **CLIENTES**, e cada **CLIENTE** pode ser atendido por um ou mais **FUNCIONARIOS**.



Estudo de Caso: Um Supermercado

□ Relembrando:

- Um **CLIENTE** pode **comprar** uma ou mais **PRODUTOS**, e cada **PRODUTO** pode ser comprado por um ou mais **CLIENTES**.



Atributos em um Relacionamento

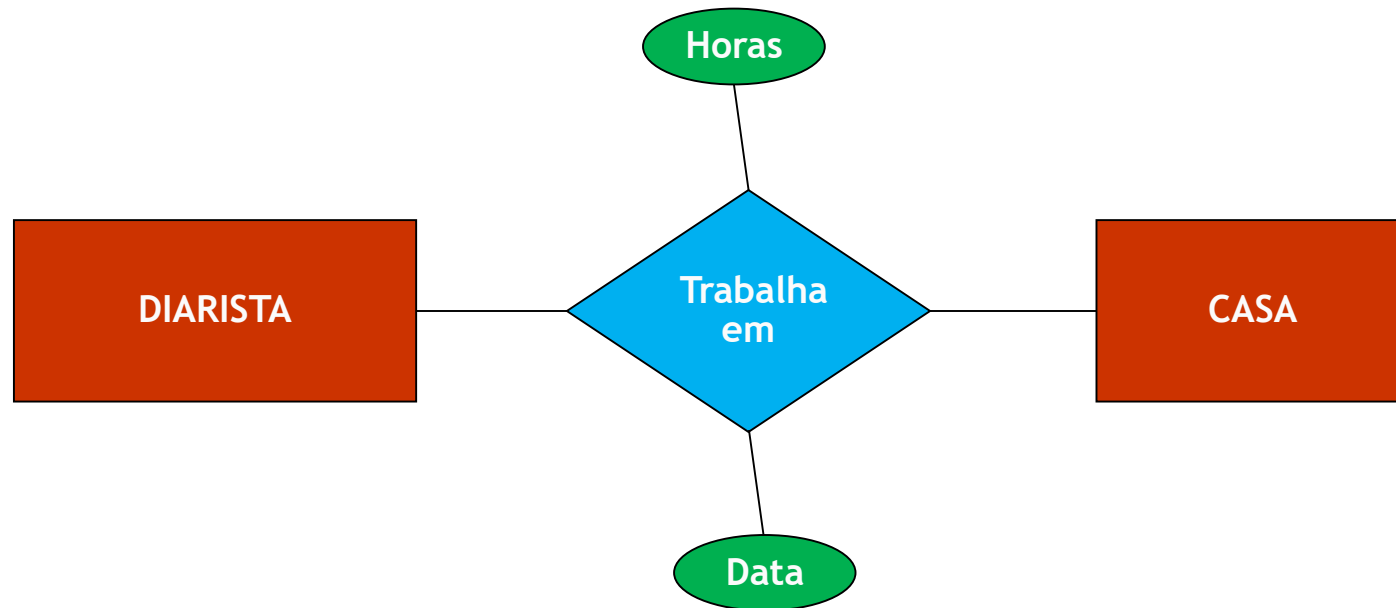
- Na maior parte das vezes, os **atributos** estão associados a **Entidades**.
- Todavia, nem sempre é assim.
- Em alguns casos, *podemos encontrar atributos em determinados relacionamentos*.

Atributos em um Relacionamento

▣ Exemplo: Minimundo de uma Central de Diaristas

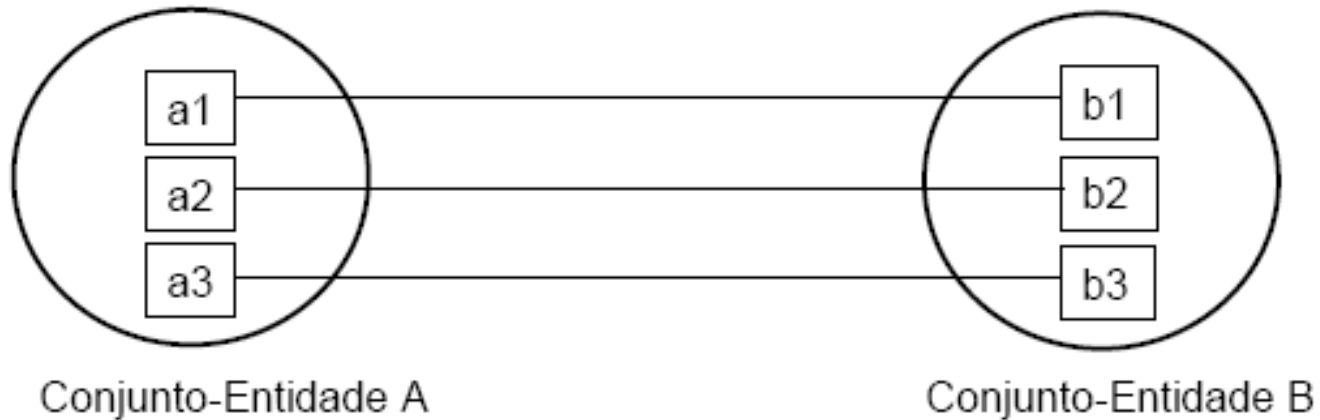
- A central de diaristas mantém o cadastro de suas diaristas por matrícula, nome, endereço, telefone e data de nascimento.
- As diaristas trabalham em várias casas, e em cada casa podem trabalhar uma ou mais diaristas.
- Tais casas são caracterizadas pelo código (que deve ser *único*) e endereço .
- Deseja-se guardar a *data do serviço* e o *número de horas* que uma diarista *trabalha* em uma casa.

Atributos em um Relacionamento



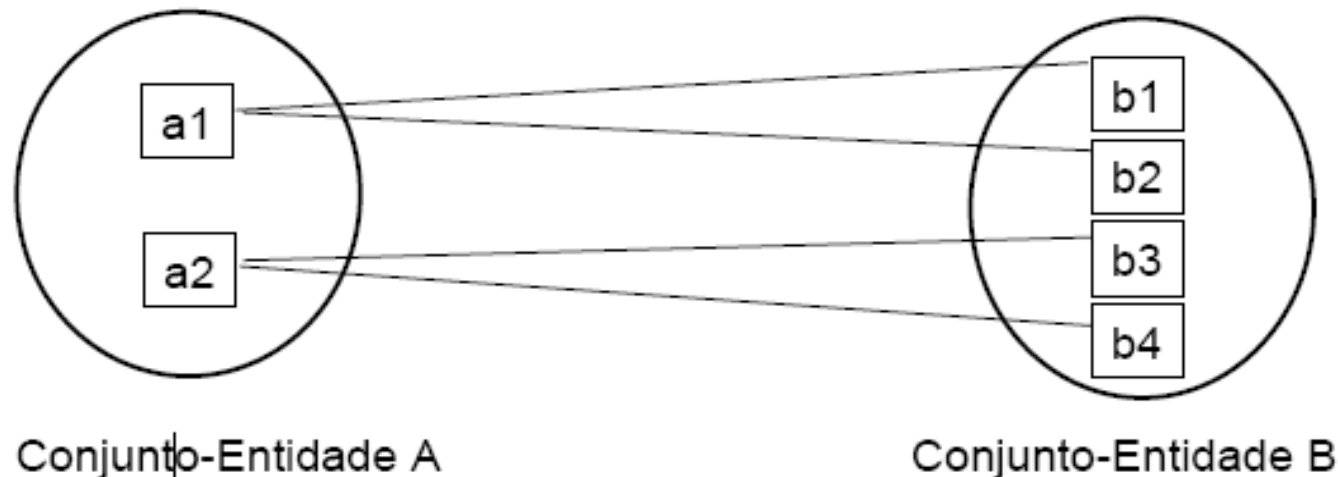
Cardinalidade

- **Um-para-um.** Uma entidade em A está associada no máximo a uma entidade em B e uma entidade em B está associada no máximo a uma entidade em A



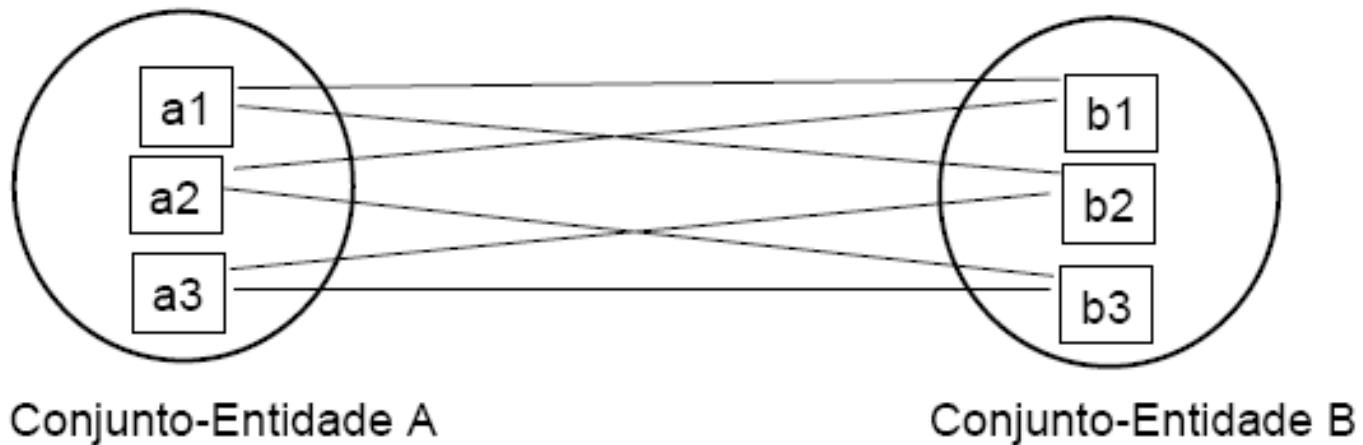
Cardinalidade

- **Um-para-muitos.** Uma entidade em A está associada a qualquer número de entidades em B, enquanto uma entidade em B está associada no máximo a uma entidade em A



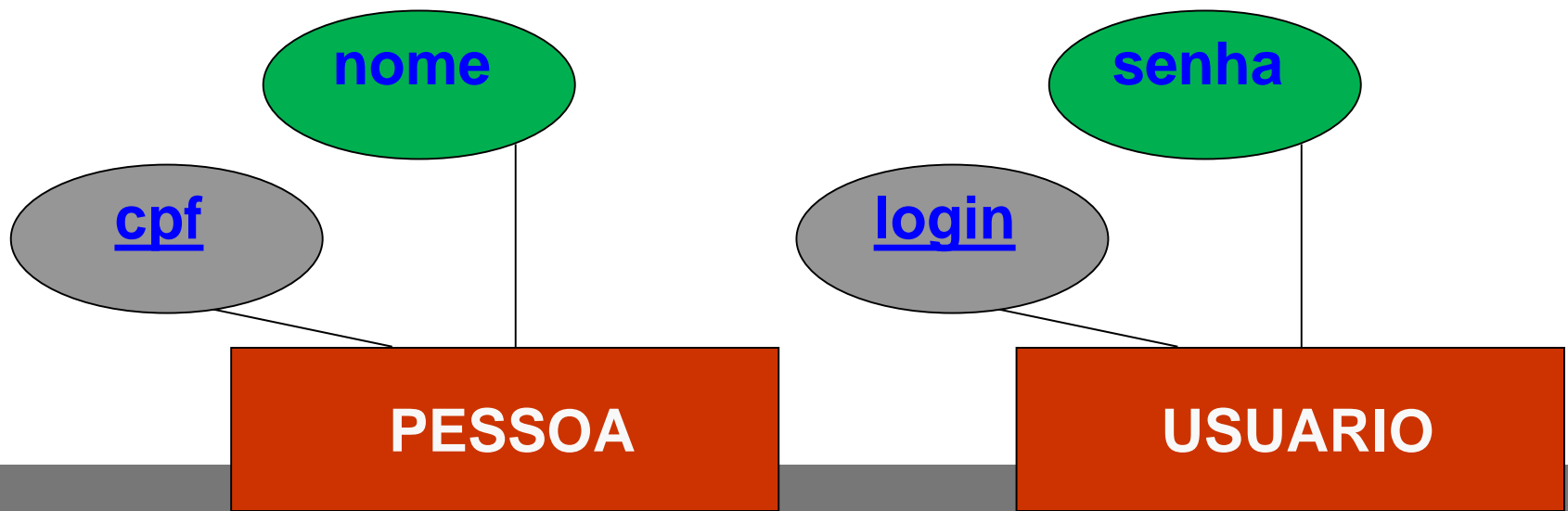
Cardinalidade

- **Muitos-para-muitos.** Uma entidade em A está associada a qualquer número de entidades em B, e uma entidade em B está associada a qualquer número de entidades em A.



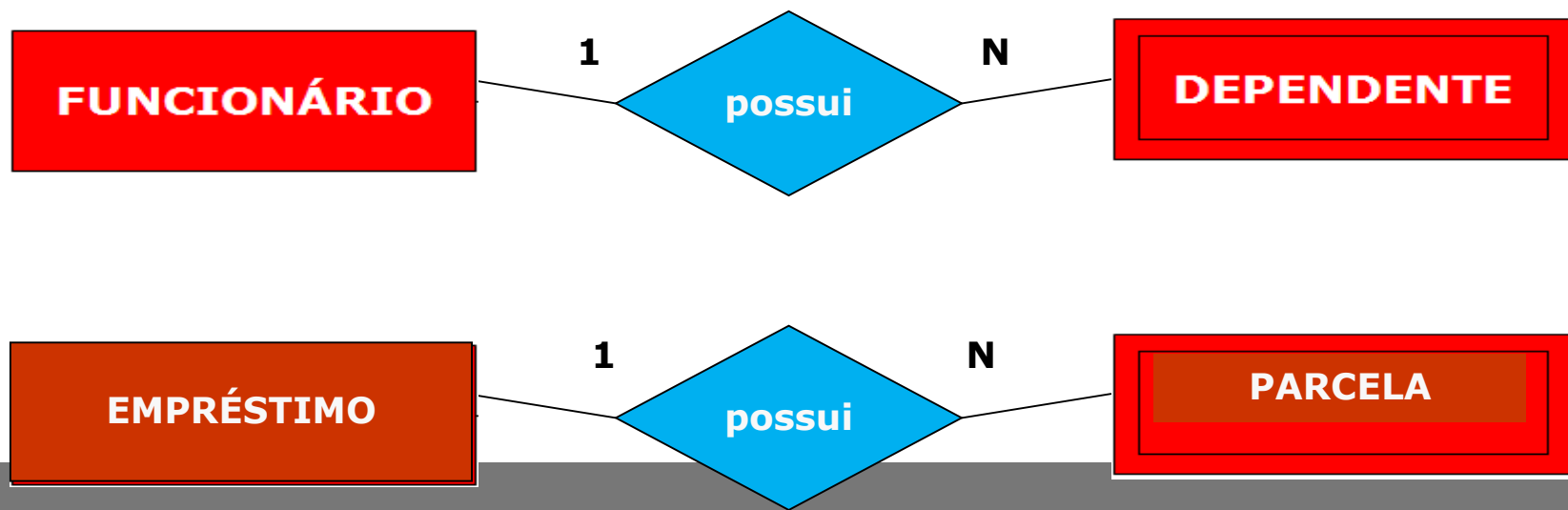
Entidade Forte (ou Regular)

- Ocorre quando a Entidade possui identidade própria, sendo sua identificação composta pelo seu atributo chave.
- Exemplos: pessoa, usuario



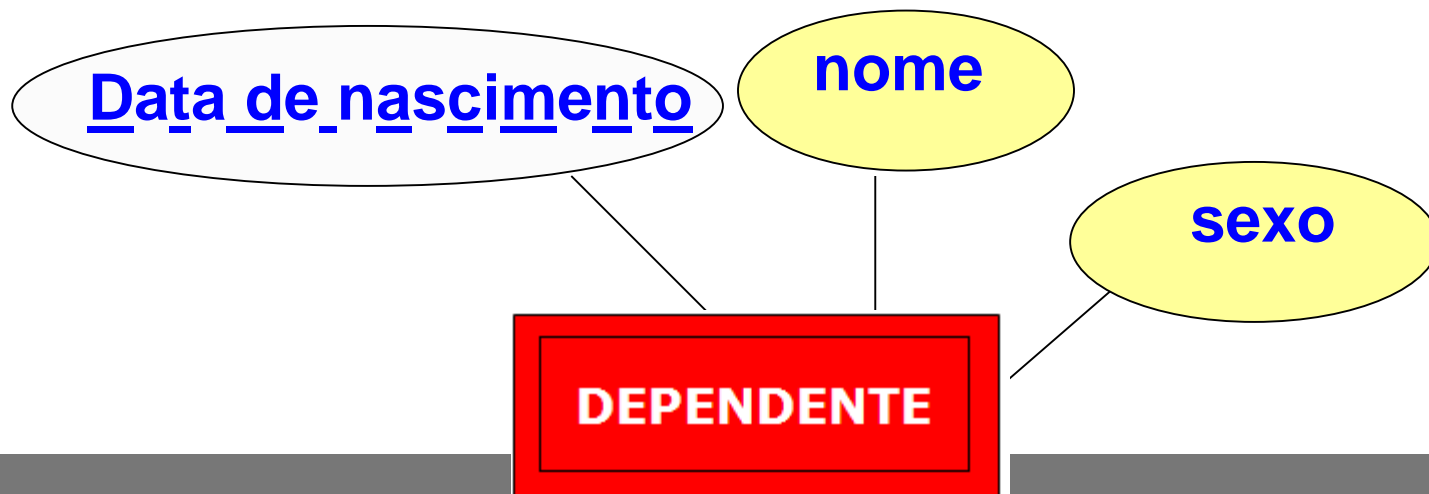
Entidade Fraca

- Ocorre quando a Entidade não possui sequer identidade própria, sendo sua identificação composta pela chave proveniente da entidade dona concatenada.
- Exemplos: dependente, parcela



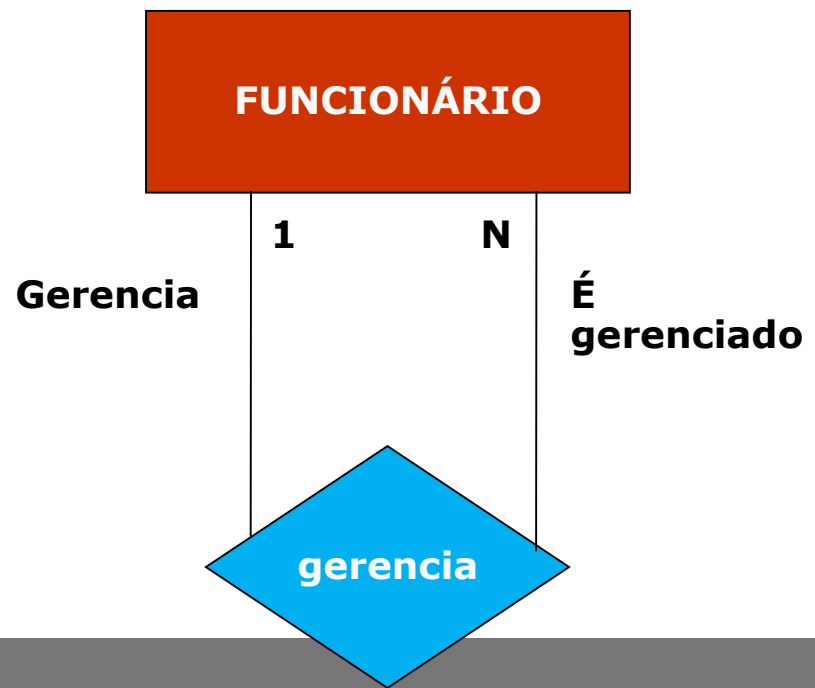
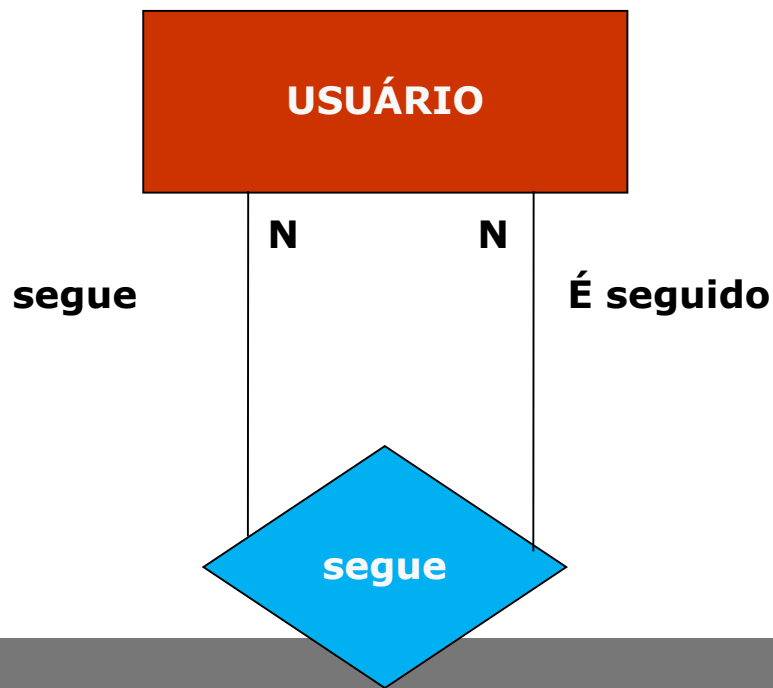
Tipos de Atributos

- ▣ Atributo **Chave parcial**: identificador parcial de uma Entidade Fraca.
- ▣ Pode assumir valores repetidos, não conseguindo garantir totalmente a distinção entre uma Entidade e outra.
- ▣ Exemplo: data de nascimento



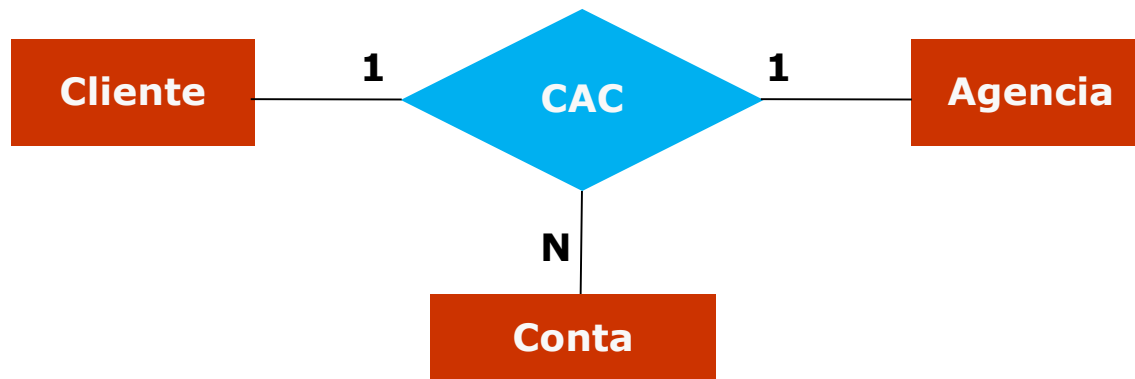
Auto-Relacionamento

- Ocorre quando uma entidade se relaciona com ela mesma
- Exemplos: segue, gerencia



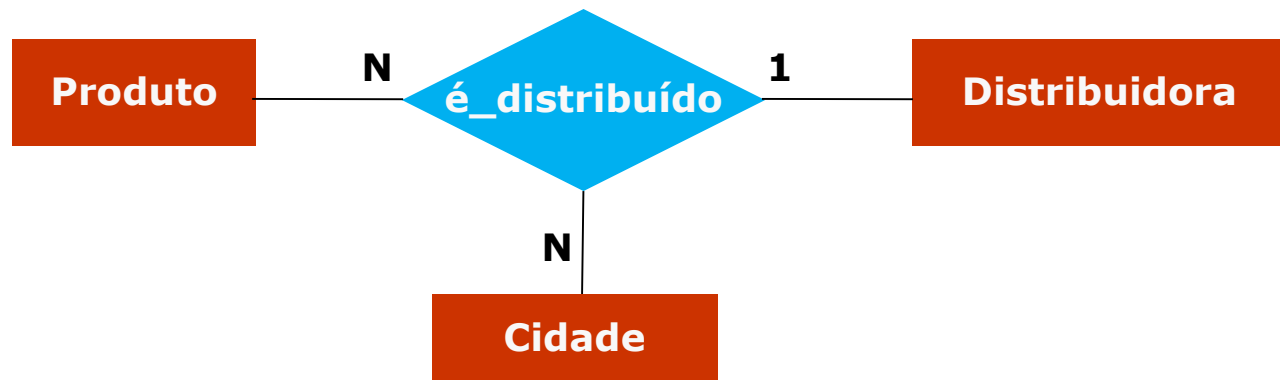
Relacionamento Ternário

- Ocorre quando um relacionamento envolve três entidades.
- Leitura unidimensional: *um cliente possui uma ou mais contas em uma determinada agência.*
- Exemplo: CAC



Relacionamento Ternário

- Exemplo: *é_distribuído*
- Leitura unidimensional: *um ou mais produtos são distribuídos por uma distribuidora em uma ou mais cidades.*



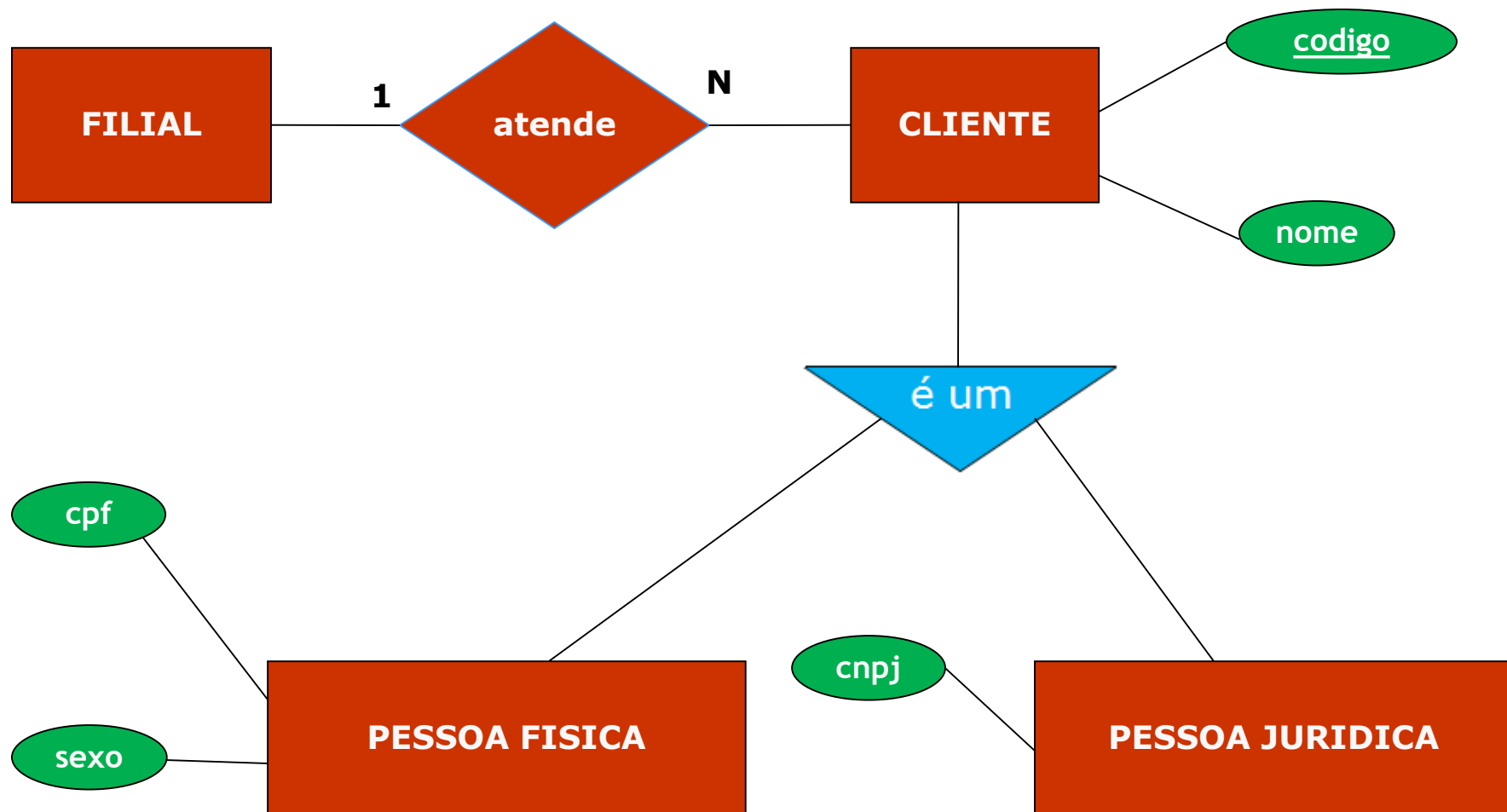
Herança

- ▣ Em Programação Orientada a Objetos, herança é uma forma de reutilização de software na qual uma nova classe é criada, absorvendo membros de um classe existente e aprimorada com capacidades novas ou modificadas.

Herança

- ▣ Podemos Generalizar / Especializar uma Entidade dentro de um modelo E-R
- ▣ Podemos dividir em categorias cada entidade

Herança (representação)



Exemplos de Herança

▣ Exemplo 1:

- Um **Funcionário** (matrícula, nome, salário e endereço) *pode ser* uma **secretaria**, um **caixa** ou um **gerente**.
- Uma **secretária** é um **Funcionário** que possui **velocidade de digitação** como atributo adicional.
- Um **caixa** é um **Funcionário** que possui **número** como atributo adicional.
- Um **gerente**, por sua vez, é um **Funcionário** que possui **número** como atributo adicional.

Exemplos de Herança

▣ Exemplo 2:

- Um **Veículo** (cod, preço, cor e marca) pode ser um **carro** ou um **caminhão**.
- Um **carro** é um **Veículo** que possui **velocidade máxima** e **número de passageiros** como atributos adicionais.
- Um **caminhão**, por sua vez, também é um **Veículo**, mas que possui **número de eixos** e **capacidade de peso** como atributos adicionais.

Exemplos de Herança

▣ Exemplo 3:

- Uma **Pessoa** (CPF, nome, sexo, data de nascimento e endereço) pode ser uma **funcionário** ou um **aluno**.
- Um **funcionário** é *uma* **Pessoa** que possui **salário** como atributo adicional.
- Um **aluno**, por sua vez, é *uma* **Pessoa** que possui **dependente principal** como atributo adicional.

Considerações Finais

- Para qualquer sistema, é essencial que façamos a modelagem de dados
- A Modelagem Entidade-Relacionamento é a modelagem conceitual mais utilizada hoje em dia

Considerações Finais

- Revisando o Modelo E-R
 - Principais componentes
 - Tipos de atributos
 - Conceitos chave
 - Atributo em um relacionamento
 - Cardinalidade
 - Entidade Fraca x Entidade Forte (ou Regular)
 - Auto-relacionamento
 - Relacionamento Ternário
 - Herança

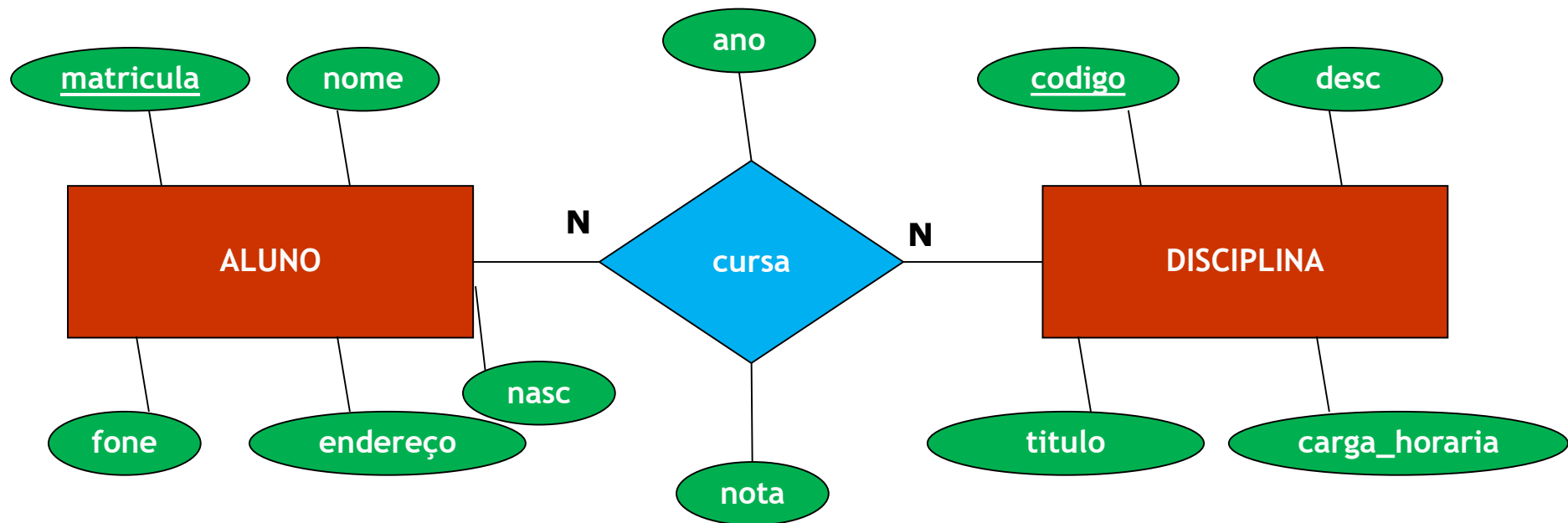
Dúvidas



Exercício

- ▣ Faça a Modelagem E-R da seguinte situação:
 - Os alunos de uma Universidade são caracterizados por matrícula, nome, cidade, estado, bairro, telefones e data de nascimento.
 - Um aluno pode cursar uma ou mais disciplinas, e cada disciplina está associada a um ou mais alunos.
 - Tais disciplinas são caracterizadas pelo código, título, descrição e carga horária.
 - Deseja-se guardar a nota obtida e o ano em que um aluno cursa uma disciplina.

Exercício



Exercícios

Modelo Entidade- Relacionamento



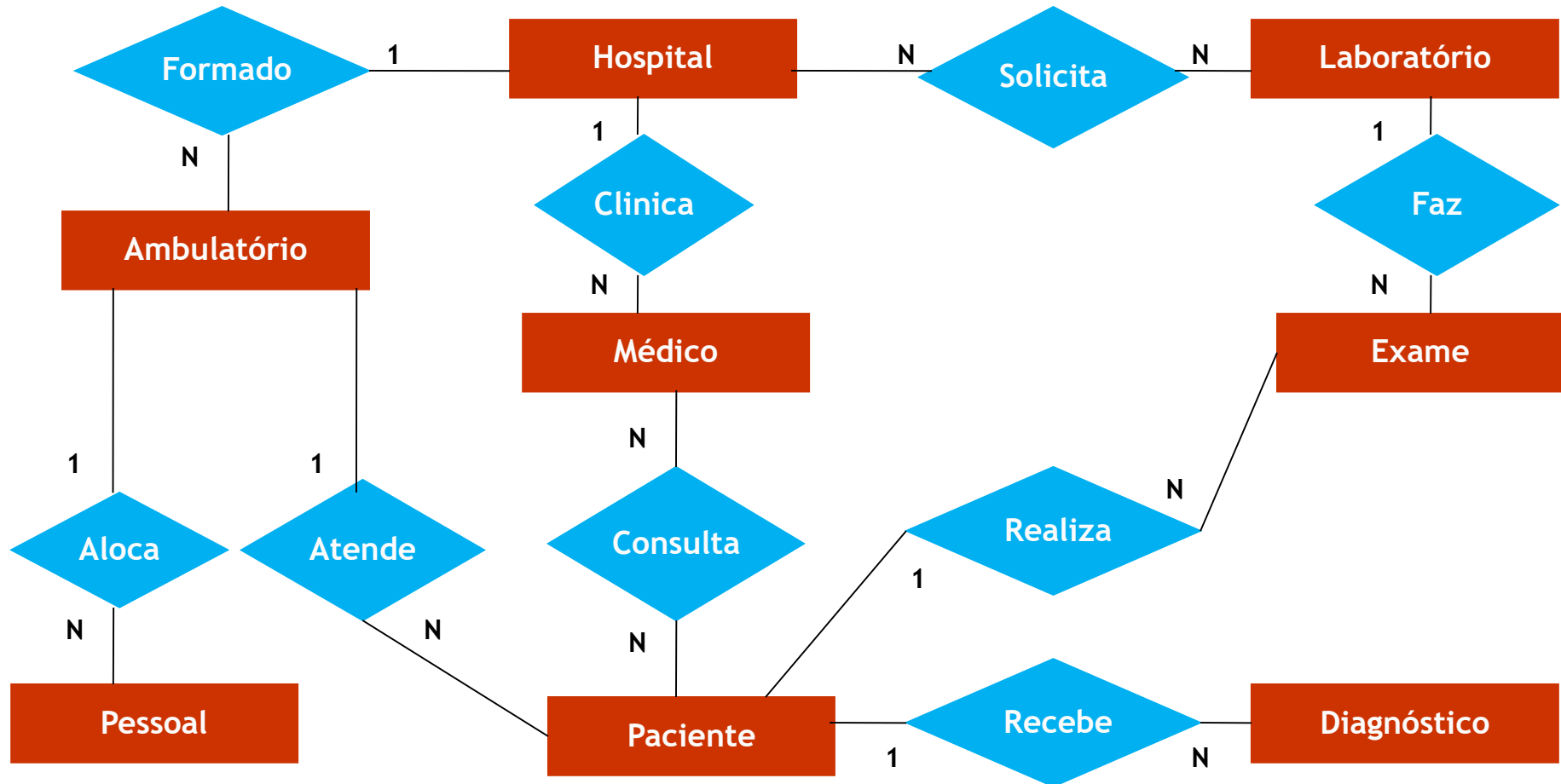
Exemplo: um Sistema de Saúde Ideal

- ▣ **Hospitais** são formados por um ou mais **Ambulatórios** e cada um destes está em um único **Hospital**
- ▣ **Médicos** clinicam em um único **Hospital**, cada um deles agregando vários **Médicos**
- ▣ **Hospitais** fazem solicitações em vários **Laboratórios**, cada um destes pode ter solicitações de vários **Hospitais**
- ▣ **Médicos** consultam vários **Pacientes**, e estes são consultados por vários **Médicos**
- ▣ **Ambulatórios** atendem vários **Pacientes**, enquanto estes só podem ser atendidos em um único **Ambulatório**
- ▣ **Pessoal** de apoio está alocado a cada **Ambulatório**, e cada um destes conta com vários integrantes do **Pessoal** de apoio

Exemplo: um Sistema de Saúde Ideal

- **Pacientes** realizam vários **Exames**, e cada **Exame** é realizado por um único **Paciente**
- **Laboratórios** fazem vários **Exames**, e cada um dos **Exames** é feito em um único **Laboratório**
- Cada **Paciente** pode receber vários **Diagnósticos**, e cada **Diagnóstico** é de um único **Paciente**

Exemplo: um Sistema de Saúde Ideal



Exercício (1/4)

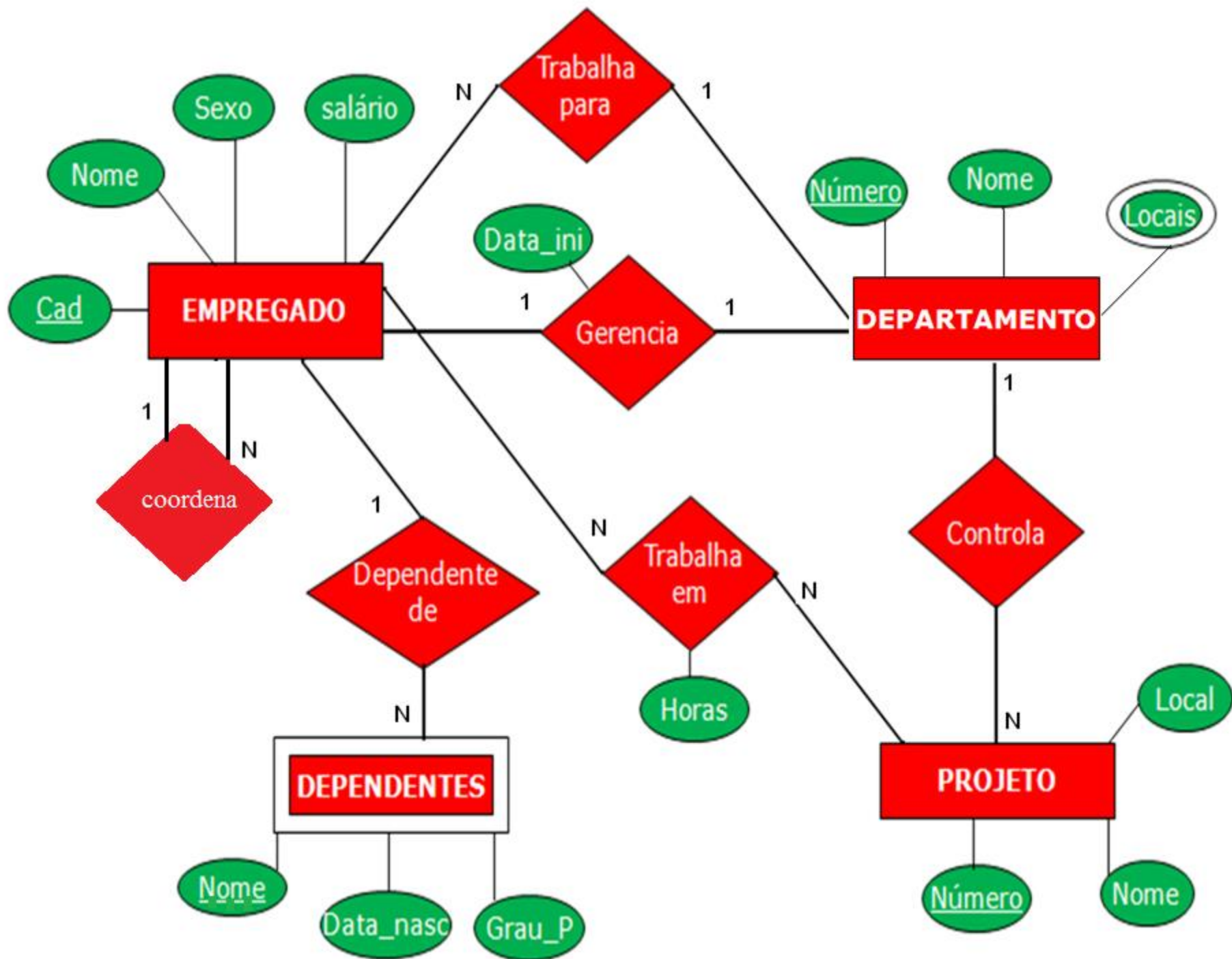
- Deseja-se guardar dados sobre empregados, departamentos e projetos de uma **companhia**
- Suponha que, depois da fase de análise de requisitos, os projetistas chegaram à seguinte descrição do “mini-mundo”, ou seja, a parte da companhia a ser representada no banco de dados.

Exercício (1/4)

- Um departamento é caracterizado por nome, número e locais.
- Um empregado gerencia um departamento, e cada um deste pode ser gerenciado por no máximo um empregado.
- Deseja-se guardar ainda a data na qual o empregado começou a gerenciar o departamento.
- Um departamento controla um ou mais projetos, e cada um destes é controlado por no máximo um departamento.
- Um projeto é caracterizado por número, nome e local.
- Um empregado é caracterizado por cadastro, nome, sexo e salário.
- Em um departamento podem trabalhar um ou mais empregados, estes por sua vez estão associados a no máximo um departamento.

Exercício (1/4)

- Um empregado pode trabalhar em um ou mais projetos, e cada um destes pode ter um ou mais empregados.
- Deseja-se guardar o número de horas que um empregado trabalha em um projeto.
- Um empregado (coordenador) pode coordenar um ou mais empregados, e cada empregado, por sua vez, pode ser coordenado por no máximo um outro empregado.
- Cada empregado possui dependentes caracterizados por nome, data de nascimento e grau de parentesco.



Exercício (2/4)

- Uma **Administradora de Condomínios** deseja guardar dados sobre edifícios, apartamentos e de seus moradores.
- Suponha que, depois da fase de análise de requisitos, os projetistas chegaram à seguinte descrição de mini-mundo, ou seja, a parte da Administradora a ser representada no Banco de Dados:

Exercício (2/4)

- Cada edifício tem um código, um endereço, uma data de construção e uma data de vistoria.
- Cada edifício possui vários apartamentos, e cada um destes está associado a um único edifício.
- Um apartamento é caracterizado por um número (que deve ser único) e uma área.
- Em cada apartamento podem morar várias pessoas, e cada uma destas pode estar associada a um ou mais apartamentos.
- Uma pessoa é caracterizada por CPF, nome, sexo e data de nascimento.
- Deseja-se guardar ainda o tipo de moradia (para indicar se o morador é inquilino ou proprietário do apartamento) e a data em que uma pessoa começou a morar em um apartamento.

Exercício (3/4)

- Uma **Locadora** deseja guardar dados sobre filmes, clientes, funcionários e fornecedores.
- Suponha que, depois da fase de análise de requisitos, os projetistas chegaram à seguinte descrição de mini-mundo, ou seja, a parte da Locadora a ser representada no Banco de Dados:

Exercício (3/4)

- ▣ A Locadora deseja cadastrar cada cliente com cpf, nome, endereço e seus telefones.
- ▣ Cada cliente possui dependentes caracterizados por nome, sexo e data de nascimento.
- ▣ Cada cliente pode alugar um ou mais filmes, e cada um destes pode ser locado por vários clientes.
- ▣ Deseja-se guarda a data, o valor e a quantidade de filmes locados por um cliente.
- ▣ Um filme é caracterizado por código, duração, título, gênero e sinopse. A locadora disponibiliza dois tipos particulares de filmes: dvd e vhs.
- ▣ Um dvd *é um filme* que possui características adicionais como versão e idiomas.
- ▣ Um vhs *é um filme* que possui a cor (preto e branco, ou colorida) como atributo adicional.

Exercício (3/4)

- Cada cliente pode ser atendido por um ou mais funcionários, e cada um destes pode atender um ou mais clientes.
- Um funcionário é caracterizado por cpf, nome e endereço.
- Cada funcionário pode comprar a um ou mais fornecedores, estes por sua vez, podem efetuar vendas a vários funcionários.
- Um fornecedor é caracterizado por código, nome, endereço e telefone.
- Deseja-se guardar ainda a data em que a compra foi realizada e o seu valor.

Exercício (4/4)

- Deseja-se guardar dados sobre alunos, professores, disciplinas, séries e salas de aula de uma **Escola**.
- Suponha que, depois da fase de análise de requisitos, os projetistas chegaram à seguinte descrição de mini-mundo, ou seja, a parte da Escola a ser representada no Banco de Dados:

Exercício (4/4)

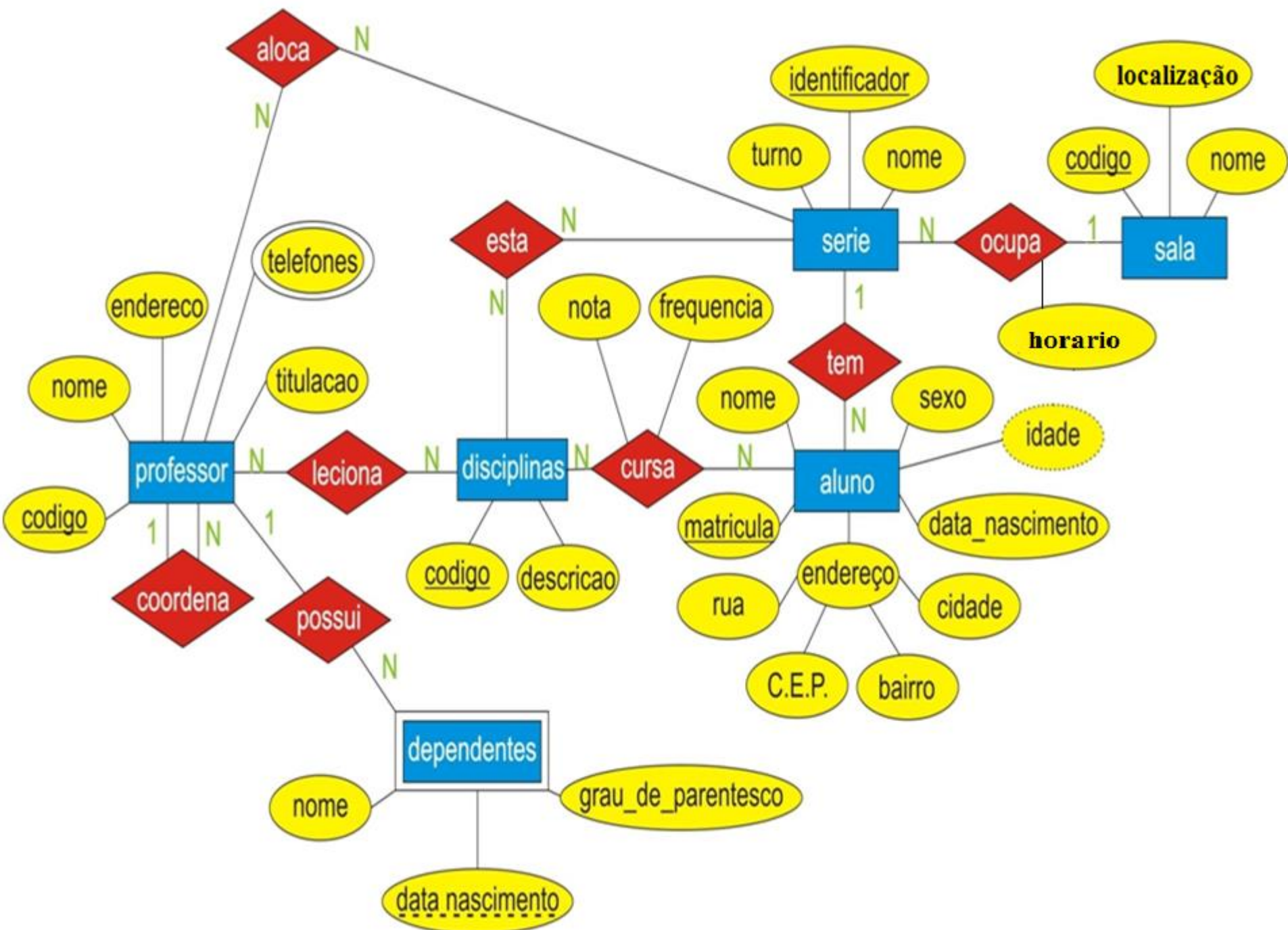
- A Escola é organizada em séries, e cada série é caracterizada por um nome, um turno e um identificador. Além disso, ela pode ter um ou mais alunos matriculados.
- Deseja-se guardar a data em que o aluno fez sua matrícula na série. Cada aluno pode estar matriculado em uma única série.
- Um aluno é caracterizado por nome, número de matrícula, data de nascimento, sexo, rua, CEP, bairro e cidade.
- Deseja-se também guardar a idade de cada aluno.
- Cada aluno cursa um determinado número de disciplinas.
- Uma disciplina é caracterizada por um código e uma descrição.
- Cada disciplina é cursada por qualquer número de alunos.
- Deseja-se guardar a frequência e a nota que o aluno obteve em cada disciplina cursada.

Exercício (4/4)

- Uma mesma disciplina pode estar associada a várias séries. Cada uma destas pode agregar várias disciplinas.
- Cada série ocupa uma única sala num determinado horário.
- Uma sala é caracterizada por um código, um nome e uma localização.
- Em horários obrigatoriamente distintos, uma sala pode ser ocupada por várias séries.

Exercício (4/4)

- A Escola é formada ainda por um corpo de professores, onde cada professor pode lecionar em uma ou mais disciplinas e cada uma destas pode ser lecionada por vários professores.
- Deseja-se guardar o coordenador direto de cada professor (que também é um professor).
- Cada professor está alocado em uma ou mais séries, e cada uma destas pode agregar vários professores.
- Um professor é caracterizado por um código, um nome, uma titulação, um endereço e pode ter vários telefones.
- Cada professor pode possuir dependentes, que por sua vez, são caracterizados por nome, data de nascimento e grau de parentesco.



Regras de Normalização



Roteiro

- ▣ Introdução
- ▣ Primeira Forma Normal
- ▣ Anomalias
- ▣ Dependência Funcional
- ▣ Segunda Forma Normal
- ▣ Dependência Transitiva
- ▣ Terceira Forma Normal
- ▣ Considerações Finais
- ▣ Exercícios

Introdução

- ▣ O conceito de normalização foi introduzido por E. F. Codd em 1970
- ▣ Através do processo de normalização pode-se, gradativamente, substituir um conjunto de entidades e relacionamentos por um outro, o qual se apresenta "purificado" em relação às Anomalias de Atualização (inclusão, alteração e exclusão).

Introdução

- ▣ As **Anomalias de Atualização** podem causar certos problemas, tais como:
 - redundância de dados desnecessária;
 - perdas (acidentais) de informação;
 - inconsistência;
 - dificuldade na representação de fatos da realidade observada;
 - grupos repetitivos (atributos multivalorados) de dados;
 - dependências funcionais totais ou parciais em relação a uma chave concatenada;
 - dependências transitivas entre atributos.

Introdução

- Na aula de hoje, veremos como estes problemas podem ser minimizados, sensivelmente, através da normalização, tornando o modelo de dados elaborado mais estável e sujeito a poucas manutenções.

Introdução

■ Anomalias de Atualização

- Que anomalias podem ser encontradas na tabela PEDIDO, mostrada abaixo?

Num Ped	Cliente	Endereco	Cpf	Data	Cod Prod	Unid	Qtd	Desc	Valor Unid.	Total Prod.	Total Pedido
55	Ivan	R.Aurora...	835	20-03-2008	45	L	20	Álcool	5,00	100,00	1799,00
55	Ivan	R.Aurora...	835	20-03-2008	130	M	2	Tecido	20,00	40,00	1799,00
55	Ivan	R.Aurora...	835	20-03-2008	35	Kg	30	Farinh a	1,00	30,00	1799,00
55	Ivan	R.Aurora...	835	20-03-2008	78	Kg	50	Cimen to	30,00	1500,0 0	1799,00

■ Anomalia de Inclusão:

- Ao ser incluído um novo cliente, o mesmo tem que estar (obrigatoriamente) relacionado a uma venda.
- Desta forma, toda vez que um cliente efetuar uma compra, o sistema armazenará seus dados novamente (nome-do-cliente, endereco-cliente e cpf-cliente) => **REDUNDÂNCIA DE DADOS DESNECESSÁRIA.**

Introdução

■ Anomalias de Atualização

- Que anomalias podem ser encontradas na tabela PEDIDO, mostrada abaixo?

Num Ped	Cliente	Endereco	Cpf	Data	Cod Prod	Unid	Qtd	Desc	Valor Unid.	Total Prod.	Total Pedido
55	Ivan	R.Aurora...	835	20-03-2008	45	L	20	Álcool	5,00	100,00	1799,00
55	Ivan	R.Aurora...	835	20-03-2008	130	M	2	Tecido	20,00	40,00	1799,00
55	Ivan	R.Aurora...	835	20-03-2008	35	Kg	30	Farinh a	1,00	30,00	1799,00
55	Ivan	R.Aurora...	835	20-03-2008	78	Kg	50	Cimen to	30,00	1500,0 0	1799,00

■ Anomalia de Exclusão:

- Ao ser excluído um cliente, os dados referentes as suas compras serão perdidos, ocasionando **PERDA DE INFORMAÇÃO**.
- Desta forma, consultas importantes não poderão ser efetuadas.
 - Exemplo:
 - No total, quantos litros de álcool foram vendidos no mês?
 - Quantos quilogramas de cimento foram vendidos em 20 de março de 2008?

Introdução

■ Anomalias de Atualização

- Que anomalias podem ser encontradas na tabela PEDIDO, mostrada abaixo?

Num Ped	Cliente	Endereco	Cpf	Data	Cod Prod	Unid	Qtd	Desc	Valor Unid.	Total Prod.	Total Pedido
91	Joyce	R.127...	121	01-03-2008	23	M	50	Nylon	5,00	250,00	395,00
91	Joyce	R.Ipê...	121	01-03-2008	130	M	5	Tecido	20,00	100,00	395,00
99	Max	R.Ceni...	747	16-03-2008	23	M	30	Nylon	5,00	150,0	150,00
95	Nadia	Trv. Borges...	998	15-03-2008	23	M	40	Nylon	5,00	200,00	200,00
91	Joyce	R.127...	121	01-03-2008	90	L	15	Cola	3,00	45,00	395,00

■ Anomalia de Alteração:

- Caso o endereço de um cliente seja alterado, **essa alteração tem de ser feita em várias linhas da tabela**, podendo provocar **INCONSISTÊNCIA** na base de dados, isto é, numa determinada linha, o cliente poderá aparecer em um endereço e, em outra linha, em outro.

Introdução

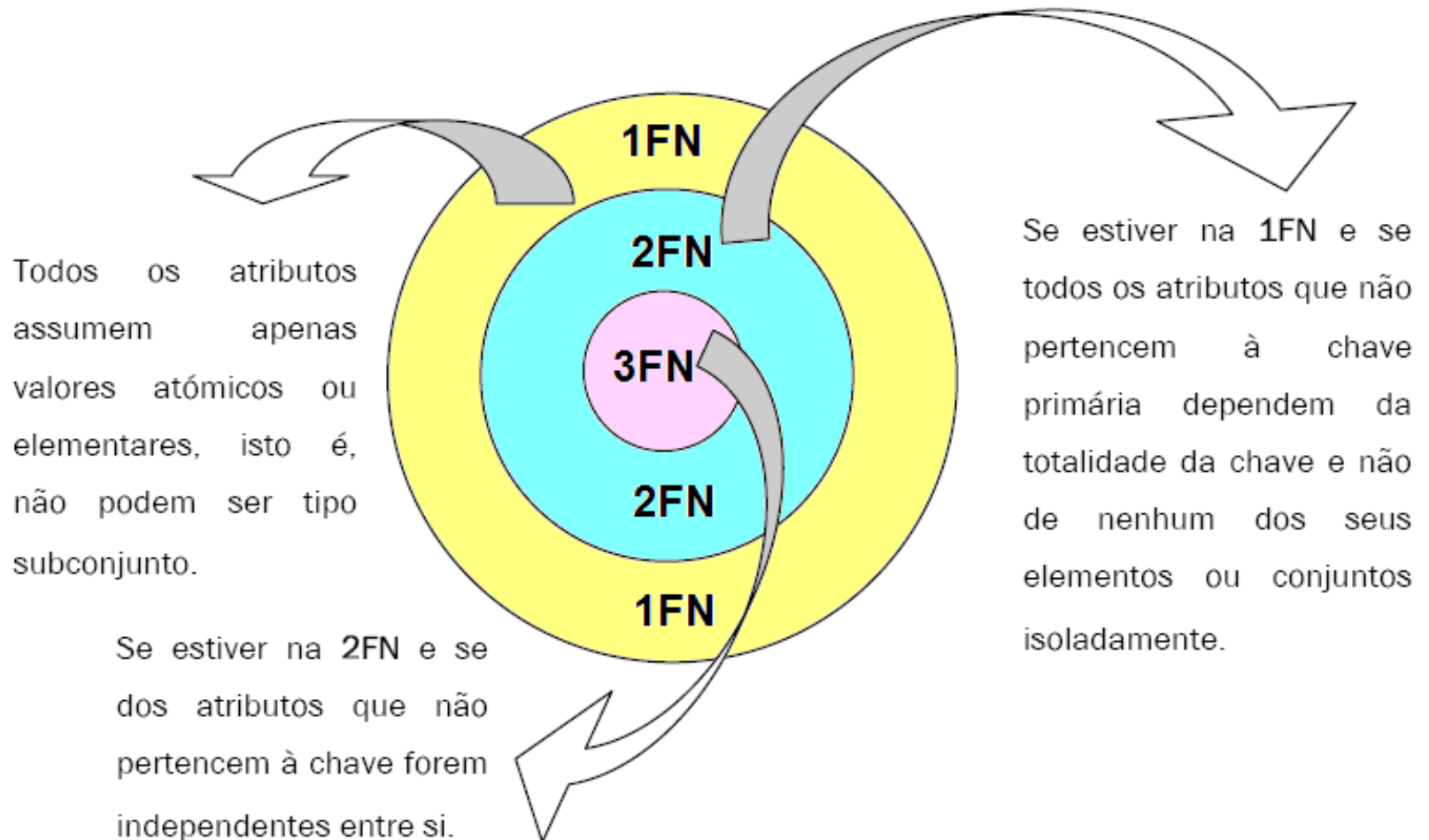
- ▣ **Normalização** de relações é portanto uma técnica que **permite depurar um projeto de banco de dados, através da identificação de inconsistências** (informações em duplicidade, dependências funcionais mal resolvidas, etc)
- ▣ À medida que um conjunto de relações passa para uma forma normal, vamos construindo um banco de dados **mais confiável**
- ▣ O objetivo da normalização não é eliminar todas as **inconsistências**, e sim **controlá-las**

Introdução

- ▣ Esse processo é composto pelas chamadas formas normais
 - 1ª Forma Normal (1ª FN);
 - 2ª Forma Normal (2ª FN);
 - 3ª Forma Normal (3ª FN);

 - Forma Normal de Boyce-Codd (FNBC);
 - 4ª Forma Normal (4ª FN);
 - 5ª Forma Normal (5ª FN);
- ▣ Um modelo de base de dados que respeite os princípios estipulados até à 3ª FN é considerado adequadamente elaborado para funcionar num SGBD relacional.

Introdução



1ª Forma Normal (1 FN)

- Uma tabela encontra-se na 1FN se todos os seus atributos estiverem definidos em domínios que contenham apenas valores atômicos e monovalorados
- Os domínios devem ser formados por valores elementares e não por conjuntos de valores.

1ª Forma Normal (1 FN)

- Imaginemos uma tabela destinada a registrar a informação sobre os alunos e as disciplinas em que estes estão matriculados

ALUNOS (CodAluno, Nome, Morada, Disciplinas)

1ª Forma Normal (1 FN)

<u>CodAluno</u>	Nome	Morada	Disciplinas
1214	Rui Costa	Rua A	Português, Matemática, Física
1250	Ana Maria	Rua B	Latim, Português, Inglês
1356	Carla Silva	Av. ABC	Economia, Matemática, Direito
1456	Hugo Leal	Bairro DEF	Português, Matemática

- ❑ Esta tabela não obedece à primeira forma normal (1FN), uma vez que o atributo Disciplinas admite conjuntos de valores.

1ª Forma Normal (1 FN)

□ Soluções:

- Quando a **quantidade de valores** é **pequena** e **conhecida a priori**:
 - Substitui o atributo multivalorado por um conjunto de atributos de mesmo domínio, cada um representando a ocorrência de um valor
 - Exemplo: Cada aluno só pode pagar duas disciplinas

<u>CodAluno</u>	Nome	Morada	Disciplina 1	Disciplina 2
1214	Rui Costa	Rua A	Português	Física
1250	Ana Maria	Rua B	Latim	Português
1356	Carla Silva	Av. ABC	Economia	Matemática
1456	Hugo Leal	Bairro DEF	Português	Inglês

1ª Forma Normal (1 FN)

□ Soluções

- Quando a **quantidade de valores** é **muito grande**, **variável** ou **desconhecida**.
 - Retira-se da relação o atributo multivalorado, e cria-se uma nova relação que tem o mesmo conjunto de atributos chave, mais o atributo multivalorado como chave, porém tomado como monovalorado

<u>CodAluno</u>	Nome	Morada
1214	Rui Costa	Rua A
1250	Ana Maria	Rua B
1356	Carla Silva	Av. ABC
1456	Hugo Leal	Bairro DEF

<u>CodAluno</u>	<u>Disciplina</u>
1214	Português
1214	Física
1356	Economia
1456	Português

1ª Forma Normal (1 FN)

▣ Resumindo...

- A primeira forma normal impõe uma diretiva básica de organização e projeto de tabelas
 - Eliminar colunas duplicadas dentro da mesma tabela

Anomalias

□ Está na 1FN?

<u>CodAluno</u>	Nome	Morada	Disciplina
1214	Rui Costa	Rua A	Português
1214	Rui Costa	Rua A	Matemática
1214	Rui Costa	Rua A	Física
1250	Ana Maria	Rua B	Latim
1250	Ana Maria	Rua B	Português
1250	Ana Maria	Rua B	Inglês
1356	Carla Silva	Av. ABC	Economia
1356	Carla Silva	Av. ABC	Matemática
1356	Carla Silva	Av. ABC	Direito
1456	Hugo Leal	Bairro DEF	Português
1456	Hugo Leal	Bairro DEF	Matemática

Anomalias

- A tabela ALUNOS agora está na 1FN, pois todos os atributos contêm apenas valores elementares
- Apresenta, **no entanto**, **grande redundância de informação**, que se reflete na repetição dos identificadores dos nomes e moradas dos alunos.
- Além desse inconveniente, podem apontar-se outros.

Anomalias

- ▣ **Problemas de atualização** - se a morada de um aluno for alterada, essa alteração tem de ser feita em várias linhas da tabela, sob o risco de gerar incoerências na Base de Dados, isto é, numa determinada linha o aluno poderá aparecer uma morada e noutra linha outra;

Anomalias

- ▣ Problemas de eliminação - porque para anular a matrícula de um aluno implica ter de eliminar várias linhas da tabela, e mesmo perder a informação do aluno, tal como NÚMERO, NOME e MORADA.

Dependência Funcional

- Um atributo ou conjunto de atributos é **determinante** de outros atributos **quando os identifica de modo único**

- Ex:

- codAluno
- codDisciplina

ALUNOS (CodAluno, Nome, Morada, CodDisciplina, Disciplina)

- Os atributos identificados de modo único por um outro atributo, ou conjunto de atributos, **são funcionalmente dependentes deste último**

- Ex:

- NomeAluno e Morada_aluno
- Disciplina

Dependência Funcional

- Exemplo:

ALUNOS (CodAluno, Nome, Morada, CodDisciplina, Disciplina)

- Temos:

- **Nome e Morada** são dependentes de CodAluno
- **Disciplina** é dependente de CodDisciplina

- Podemos ainda ler da seguinte forma:

- Com um código do aluno, podemos encontrar seu nome e sua morada
- Com um código da disciplina, podemos encontrar sua disciplina.

2ª Forma Normal (2 FN)

- ▣ Uma relação está na 2FN quando duas condições são satisfeitas
 - a relação está na primeira forma normal;
 - todos os atributos não-chave (ou atributos primos) dependem funcionalmente de toda a chave primária

2ª Forma Normal (2 FN)

- Observe a relação abaixo:

BOLETIM = {matricula-aluno, codigo-materia, numero-prova, nota,
data-da-prova, nome-aluno, endereço-aluno, nome-materia}

Qual a chave-primária?

- Fazendo a análise da dependência funcional de cada atributo primo, chegamos às seguintes dependências funcionais:
 - matricula-aluno, codigo-materia, numero-prova -> nota
 - Nota depende de matricula-aluno, codigo-materia, numero-prova
 - codigo-materia, numero-prova -> data-da-prova
 - Data da prova depende de...
 - matricula-aluno -> nome-aluno, endereço-aluno
 - ?
 - codigo-materia -> nome-materia
 - ?

2ª Forma Normal (2 FN)

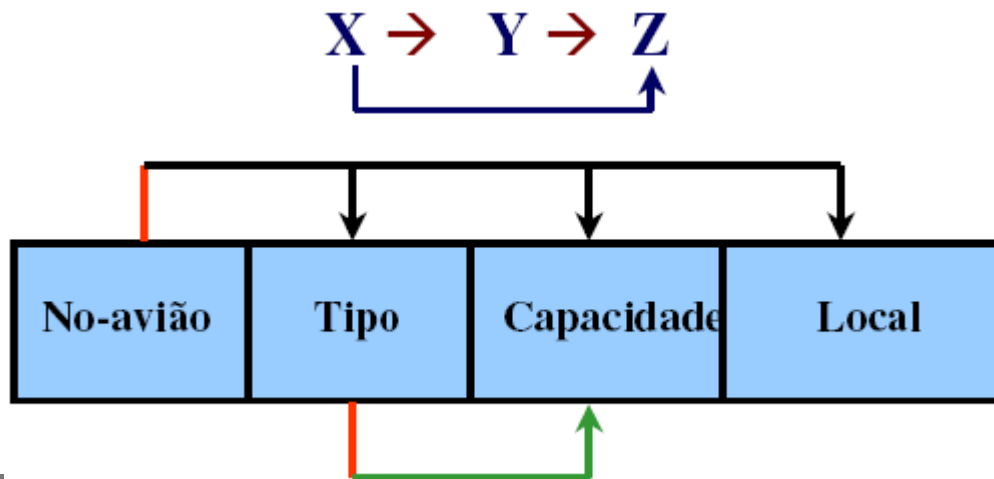
- Concluímos então que na tabela Boletim (mostrada anteriormente) apenas o atributo primo (ou não chave) **nota** depende totalmente de **toda chave primária**.
- Desta forma, os atributos **data-da-prova**, **nome-aluno**, **endereço-aluno**, **nome-materia** não respeitam a 2FN.
- Sendo assim, **para que toda a relação seja passada para a 2FN**, deve-se criar novas relações, agrupando os atributos de acordo com suas dependências funcionais:

```
BOLETIM = {matricula-aluno, codigo-materia, numero-prova, nota}  
PROVA   = {codigo-materia, numero-prova, data-da-prova}  
ALUNO   = {matricula-aluno, nome-aluno, endereço-aluno}  
MATERIA = {codigo-materia, nome-materia}
```

Dependências Transitivas

□ Dependência Transitiva

- Significa que um atributo não depende diretamente do atributo determinante e sim de algum outro atributo, que por sua vez depende do determinante.
- Dessa forma, todo atributo não-chave deve ser analisado em relação a outros atributos não-chave.



- Y (**tipo**) depende de X (**no-avião**)
- Z (**capacidade**) depende de Y (**tipo**)
- Logo, Z (**capacidade**) também depende de X (**no-avião**)

Dependências Transitivas

- CPF -> código-cidade
 - Código-cidade depende de CPF
- código-cidade -> nome-cidade
 - Nome-cidade depende de Código-cidade

Logo CPF -> nome-cidade

- Logo, Nome-cidade depende (transitivamente) de CPF

Você pode ler o exemplo acima também da seguinte maneira:

- Se com um número de CPF eu encontro o código da cidade de uma pessoa, e com o código da cidade eu encontro o nome da cidade, então com o número do CPF eu posso encontrar o nome da cidade

3ª Forma Normal (3 FN)

- Uma relação está na 3FN quando duas condições forem satisfeitas:
 - a relação está na segunda forma normal
 - todos os atributos primos dependem não transitivamente de toda a chave primária

3ª Forma Normal (3 FN)

- ▣ Observe a relação abaixo:

PEDIDO = {numero-pedido, codigo-cliente, data-pedido, nome-cliente, codigo-cidade-cliente, nome-cidade-cliente}

- ▣ Fazendo a análise da dependência funcional de cada atributo primo, chegamos às seguintes dependências funcionais:

numero-pedido -> codigo-cliente

numero-pedido -> data-pedido

codigo-cliente -> nome-cliente

codigo-cliente -> codigo-cidade-cliente

codigo-cidade-cliente -> nome-cidade-cliente

3ª Forma Normal (3 FN)

- Isto é dependência transitiva, devemos resolver inicialmente as dependências mais simples, criando uma nova relação onde código-cliente é a chave, o código-cliente continuará na relação PEDIDO como atributo primo, porém, os atributos que dependem dele devem ser transferidos para a nova relação:

PEDIDO = {numero-pedido, código-cliente, data-pedido}

CLIENTE = {código-cliente, nome-cliente, código-cidade-cliente, nome-cidade-cliente}

- Assim, as dependências transitivas da relação PEDIDO são eliminadas
- Porém, ...

3ª Forma Normal (3 FN)

- ▣ Agora devemos analisar a nova relação CLIENTE:

codigo-cliente -> codigo-cidade-cliente -> nome-cidade-cliente

- ▣ Observem que o nome-cidade-cliente continua com uma dependência transitiva, vamos resolvê-la da mesma maneira :

PEDIDO = {numero-pedido, codigo-cliente, data-pedido}

CLIENTE = {codigo-cliente, nome-cliente, codigo-cidade-cliente}

CIDADE = {codigo-cidade-cliente, nome-cidade-cliente}

Tabelas normalizadas na 3FN

3ª Forma Normal (3 FN)

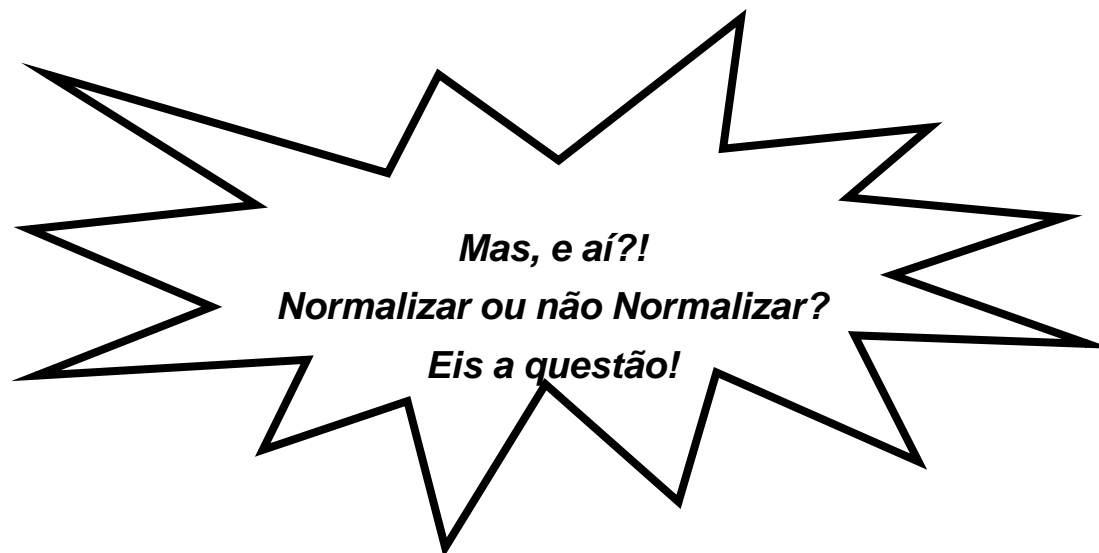
▣ Resumindo.... Convertendo da **2FN** para a **3FN**

- 1) Verificar se existem atributos que sejam **dependentes transitivos** (isto é, se **a** depende de **b**, e **b** depende de **c**, por transitividade, **a** também depende de **c**) de outros que não pertencem à chave primária, sendo ela concatenada ou não, bem como atributos que sejam dependentes de cálculo realizado a partir de outros atributos (ex: idade, tempo_de_casa)
- 2) Destacar os atributos com **dependência transitiva**, gerando uma nova tabela com este atributo e cuja chave primária é o atributo que originou a dependência.
- 3) **Eliminar** os atributos obtidos através de cálculos realizados a partir de outros atributos.

Considerações Finais

Normalizar evita introduzir inconsistências quando se alteram relações; porém obriga a execução de custosas operações de junção para a consulta de informações.

Considerações Finais



A decisão deve ser tomada considerando-se o compromisso entre se garantir a eliminação de inconsistências na base, e eficiência de acesso.

Exercício 1

- Quais anomalias são encontradas nesta tabela PILOTO?
- O que poderíamos fazer para retirar tais anomalias?
- Quais as formas normais que encontraríamos?

Num-cad	Nome	CPF	Salário	Diploma	Descrição
0010	José	123456	5.000,00	D1	Helicópteros
0010	José	123456	5.000,00	D2	Aviões a jato
0015	João	234567	3.000,00	D3	Bi-motor
0020	Manuel	345678	8.000,00	D1	Helicópteros
0020	Manuel	345678	8.000,00	D2	Aviões a jato
0020	Manuel	345678	8.000,00	D4	Concorde
0018	José	987654	4.000,00	D2	Aviões a jato

Exercício 1 (Resposta)

Piloto

Num-cad	Nome	CPF	Salário
0010	José	123456	5.000,00
0015	João	234567	3.000,00
0020	Manuel	345678	8.000,00
0018	José	987654	4.000,00

Diploma

Diploma	Descrição
D1	Helicópteros
D2	Aviões a jato
D3	Bi-motor
D4	Concorde

Formação

Num-cad	Diploma
0010	D1
0010	D2
0015	D3
0020	D1
0020	D2
0020	D4
0018	D2

Exercício 2

- Quais anomalias são encontradas nesta tabela AVIÃO?
- O que poderíamos fazer para retirar tais anomalias?
- De acordo com qual Forma Normal retiraríamos tal anomalia?

Avião

No-av	Tipo	Capacidade	Local
101	A320	320	Rio
104	B727	250	S.Paulo
105	DC10	350	Rio
103	B727	250	Recife
110	B727	250	Rio

Exercício 2 (Resposta)

Avião1

No-av	Tipo	Local
101	A320	Rio
104	B727	S.Paulo
105	DC10	Rio
103	B727	Recife
110	B727	Rio

Tipo-av

Tipo	Capacidade
A320	320
B727	250
DC10	350

SQL



SQL

- ▣ Structured Query Language
 - Linguagem de Consulta Estruturada
- ▣ Fundamentada no Modelo Relacional, inclui comando para:
 - Definição de dados
 - Consulta
 - Atualização

Usos de SQL

- ▣ DDL - Linguagem de Definição de Dados
 - Criar (create)
 - Destruir (drop)
 - Modificar (alter)

- ▣ DML - Linguagem de Manipulação de Dados
 - Consultar (select)
 - Inserir (insert)
 - Remover (delete)
 - Atualizar (update)

Comandos Sql



SQL - Criação de Tabelas

- Comando *Create table* (sintaxe)

```
CREATE TABLE <tabela>  
  
(  
  
<descrição das colunas>,  
  
<descrição das chaves>  
  
);
```

SQL - Criação de Tabelas

□ Comando *Create table*

❖ Ao se criar uma estrutura de uma tabela é necessário que o usuário forneça, para cada coluna, as seguintes informações:

❖ TIPO DE DADO

❖ Varia de acordo com o SGBD

❖ Exemplo:

❖ Oracle: **char, varchar2, number, date**, etc.

❖ Mysql: **char, varchar, int, date**, etc.

❖ TAMANHO

❖ RESTRIÇÕES

❖ EX: **primary key, foreign key, unique, not null, check**.

SQL - Criação de Tabelas

- Comando *Create table* (mais exemplos)

CREATE TABLE cliente

(

cod_cliente number(5),

nome varchar2(70) NOT NULL,

sexo char(1),

cidade varchar2(70),

estado char(2),

telefone number(7),

PRIMARY KEY (cod_cliente)

Tipo de dado 'number'

Tamanho de dado '5'

9 9 9 9 9

(valor máximo a ser representado)

Restrições de dado 'PRIMARY KEY'

SQL - Criação de Tabelas

- Comando *Create table* (mais exemplos)

```
CREATE TABLE vendedor  
(  
cod_vendedor number(5),  
nome varchar2(70) NOT NULL,  
salario_fixo number(8,2),  
faixa_comissao char(1),  
PRIMARY KEY (cod_vendedor)  
);
```

Como assim number(8,2) ?

SQL - Criação de Tabelas

- Comando *Create table* (mais exemplos)

```
CREATE TABLE telefones_vendedor  
(  
  cod_vendedor number(5),  
  telefone number(7),  
  PRIMARY KEY (cod_vendedor, telefone),  
  FOREIGN KEY (cod_vendedor)  
  REFERENCES VENDEDOR(cod_vendedor)  
);
```

Espeficando
UMA PK composta, e
UMA FK

SQL - Criação de Tabelas

- Comando *Create table* (mais exemplos)

```
CREATE TABLE produto
```

```
(
```

```
cod_produto number(5),
```

```
descricao varchar2(100),
```

```
unidade varchar2(4),
```

```
valor_unitario number(8),
```

```
PRIMARY KEY (cod_produto)
```

```
);
```

valor_unitario foi especificado adequadamente?

SQL - Criação de Tabelas

- Comando *Create table* (mais exemplos)

```
CREATE TABLE pedido (  
    num_pedido number(5),  
    prazo_entrega number(5),  
    cod_cliente number(5),  
    cod_vendedor number(5),  
    PRIMARY KEY (num_pedido),  
    FOREIGN KEY (cod_cliente)  
    REFERENCES CLIENTE(cod_cliente),  
    FOREIGN KEY (cod_vendedor)  
    REFERENCES VENDEDOR(cod_vendedor)  
);
```

Espefizando
UMA PK simples, e
DUAS FKs

SQL - Criação de Tabelas

- Comando *Create table* (mais exemplos)

```
CREATE TABLE item_do_pedido (  
  num_pedido number(5),  
  cod_produto number(5),  
  quantidade  number(6),  
  PRIMARY KEY (num_pedido, cod_produto),  
  FOREIGN KEY (num_pedido)  
  REFERENCES PEDIDO(num_pedido),  
  FOREIGN KEY (cod_produto)  
  REFERENCES PRODUTO(cod_produto) );
```

Espeficando
UMA PK composta, e
DUAS FKs

Exercício

- ❑ Faça o comando SQL para criar as seguintes tabelas:
- ❑ *aluno* (*cod*, nome, data_de_nascimento, idade, email, sexo)
- ❑ *disciplina* (*cod*, nome, carga_horaria)
- ❑ *aluno_cursa_disciplina* (*codAluno*, *codDisciplina*, media)

PK *itálico*
FK _____

SQL - Exclusão de Tabelas

- Comando *Drop table* (sintaxe)

DROP TABLE <tabela>;

SQL - Exclusão de Tabelas

- Comando *Drop table* (exemplos)

```
DROP TABLE item_do_pedido;
```

SQL - Alteração de Tabelas

- Comando *Alter table* (sintaxe)

ALTER TABLE <tabela> <ADD | MODIFY | DROP>
(<descrição das colunas>);

SQL - Alteração de Tabelas

□ Comando *Alter table* (exemplos)

- Para adicionar a coluna *data_nascimento* na tabela cliente

ALTER TABLE cliente **ADD** (data_nascimento date);

- Para excluir a coluna *data_nascimento* na tabela cliente

ALTER TABLE cliente **DROP** (data_nascimento);

- Para modificar o tamanho da coluna *telefone* na tabela telefones_vendedor

ALTER TABLE telefones_vendedor

MODIFY (telefone number(9));

SQL - Alteração de Tabelas

□ Comando *Alter table* (exemplos)

- Para renomear o nome de uma coluna de uma tabela

ALTER TABLE cliente **RENAME COLUMN** telefone TO fone;

- Para renomear o nome de uma tabela

ALTER TABLE cliente **RENAME TO** cliente_virtual;

SQL - Inserção de Dados

- Comando *Insert* (*sintaxe*)

INSERT INTO <nome da tabela>

(<nomes das colunas>) **VALUES** (<valores>);

SQL - Inserção de Dados

□ Comando *Insert* (exemplos)

INSERT INTO cliente (cod_cliente, nome, sexo, cidade, estado, telefone)

VALUES (17,'Esdras','m','Recife','PE',34558787);

INSERT INTO vendedor (cod_vendedor, nome, salario_fixo, faixa_comissao) **VALUES**
(29,'Almir',350.0,'C');

INSERT INTO pedido (num_pedido, prazo_entrega, cod_cliente, cod_vendedor)
VALUES (11,20,17,29);

INSERT INTO cliente **VALUES** (19,'Ivan','m','Caruaru','PE',3333333);

Campos podem ser omitidos.... Mas será que devem?

Menor clareza, e intuitividade valem a pena?

SQL - Inserção de Dados

□ Comando *Insert* (exemplos)

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(1,'sal','kg',0.85);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(13,'ouro','g',6.18);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(87,'cano','m',1.97);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(77,'papel','m',1.05);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(30,'açúcar','sac',0.99);
```

```
INSERT INTO produto (cod_produto, descricao, unidade, valor_unitario) VALUES  
(45,'madeira','m',0.25);
```

SQL - Inserção de Dados

□ Comando *Insert* (exemplos)

```
INSERT INTO item_do_pedido (num_pedido, cod_produto, quantidade) VALUES  
(11,87,10);
```

```
INSERT INTO item_do_pedido (num_pedido, cod_produto, quantidade) VALUES  
(11,45,30);
```

SQL - Atualização de Dados

- Comando *Update* (*sintaxe*)

UPDATE <nome da tabela>

SET <nome da coluna> = (<valor>)

WHERE <condicao>;

ONDE condição pode ser mais bem detalhada em:

Campo	Operador	Valor
-------	----------	-------

Ex1: sexo	=	'm'
-----------	---	-----

Ex2: cidade	=	'Rodelas'
-------------	---	-----------

Ex3: idade	>	20 ...
------------	---	--------

SQL - Atualização de Dados

□ Comando *Update* (exemplos)

```
UPDATE cliente  
SET     nome = 'Esdras Ricardo'  
WHERE  cod_cliente = 17;
```

```
UPDATE vendedor  
SET     salario_fixo = 400  
WHERE  cod_vendedor = 29;
```

```
UPDATE produto  
SET     unidade = 'kg', valor_unitario = 6.38  
WHERE  cod_produto = 13;
```

Se eu quiser atualizar **mais** de um campo ao mesmo tempo?

SQL - Exclusão de Dados

- Comando *Delete* (*sintaxe*)

DELETE FROM <nome da tabela>

WHERE <condicao>;

SQL - Atualização de Dados

□ Comando *Delete* (exemplos)

- *Excluindo o comprador “Josias” da tabela Cliente*

```
INSERT INTO cliente VALUES (35, 'Josias', 'm', 'Caruaru', 'PE', 6666666666);
```

```
DELETE FROM cliente WHERE cod_cliente = 35;
```

- *Excluindo o item “fósforo” da tabela Produto*

```
INSERT INTO produto VALUES (99, 'fosforo', 'caixa', 1);
```

```
DELETE FROM produto WHERE cod_produto = 99;
```


Comando Select



Recuperar dados de uma tabela

□ Comando *Select* (conceitos)

- *Os dados armazenados em uma tabela podem ser recuperados (visualizados) de várias maneiras.*
- *O comando Select permite a seleção e a manipulação, para visualização das informações armazenadas no Banco de Dados.*

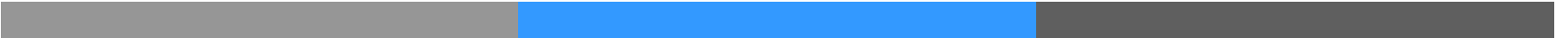
Recuperar dados de uma tabela

□ Comando *Select* (conceitos)

• *O Comando Select permite selecionar:*

- *Colunas específicas em uma tabela*
- *Todas as colunas*
- *Colunas com apelidos (alias)*
- *Expressões aritméticas*
- *Apenas algumas linhas das tabelas*
- *Linhas de forma ordenada*

SQL - Extração de Dados (1/6)



Selecionando Colunas Específicas
De uma Tabela

SQL - Extração de Dados (1/6)

□ Comando *Select* (sintaxe)

- *Seleção de colunas específicas em uma tabela*

SELECT <nome(s) da(s) colunas>

FROM <tabela>;

SQL - Extração de Dados (1/6)

□ Comando *Select* (exemplos)

- *Seleção de colunas específicas em uma tabela*

- Listar todos os produtos com as respectivas descrições, unidades e valores unitários.

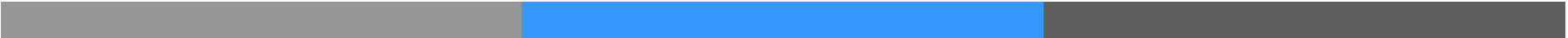
```
SELECT descricao, unidade, valor_unitario  
FROM produto;
```

- Listar da tabela CLIENTE, o nome do cliente, o sexo e a cidade onde mora.

```
SELECT nome, sexo, cidade  
FROM cliente;
```

SQL - Extração de Dados

(2/6)



Selecionando Todas as colunas
De uma Tabela

SQL - Extração de Dados (2/6)

□ Comando *Select* (sintaxe)

- *Seleção de todas as colunas de uma tabela*

```
SELECT * FROM <tabela>;
```


SQL - Extração de Dados (2/6)

□ Comando *Select* (Exemplos)

- *Seleção de todas as colunas de uma tabela*

- Listar todo o conteúdo de vendedor.

```
SELECT * FROM vendedor;
```

- Listar da tabela CLIENTE, o código do cliente, o nome, o sexo, a cidade, o estado e o telefone.

```
SELECT * FROM cliente;
```

SQL - Extração de Dados (23/6)



Colunas com apelidos (alias)



SQL - Extração de Dados (3/6)

- Comando *Select* (contextualização)
 - *Seleção de colunas com apelidos*
- *Por default, o heading (nome da coluna criado no bd) apresentado na saída do SELECT é o nome da coluna na tabela.*
- *SQL permite que se apresente a saída de um SELECT com cabeçalhos de colunas ao nosso gosto.*

SQL - Extração de Dados (3/6)

- Comando *Select* (sintaxe)
 - *Seleção de colunas com apelidos*

SELECT <cabeçalho da coluna>

AS <novo nome do cabeçalho da coluna> **FROM**
 <tabela>;

SQL - Extração de Dados (3/6)

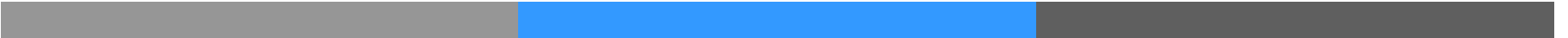
□ Comando *Select* (Exemplos)

- *Seleção de colunas com apelidos*

- Listar da tabela CLIENTE, a cidade e o sexo do cliente como cidade_cliente e sexo_cliente.

```
SELECT cidade as cidade_cliente, sexo as sexo_cliente FROM cliente;
```

SQL - Extração de Dados (4/6)



Manipulando dados numéricos:
Operadores Aritméticos

SQL - Extração de Dados (4/6)

- Comando *Select* (contextualização)
 - Manipulando dados numéricos: Operadores Aritméticos
- *Operadores aritméticos podem ser usados sobre qualquer coluna numérica.*
- *Operadores aritméticos são:*

Símbolo	Operação
+	Adição
-	Subtração
/	Divisão
*	Multiplicação

SQL - Extração de Dados (4/6)

□ Comando *Select* (Exemplos)

- Manipulando dados numéricos: Operadores Aritméticos

- Listar da tabela VENDEDOR, o nome do vendedor, o salário fixo e o triplo do mesmo.

```
SELECT nome, salario_fixo, (salario_fixo * 3) FROM vendedor;
```

```
SELECT nome, salario_fixo, (salario_fixo * 3) as salario_triplo  
FROM vendedor;
```

- Listar da tabela VENDEDOR, o nome do vendedor, o salário fixo e o dobro de seu salário acrescido de 350.

```
SELECT nome, salario_fixo, ((salario_fixo * 2) + (350)) as dobro_salario_mais  
FROM vendedor;
```


SQL - Extração de Dados

(5/6)

Selecionando apenas algumas Linhas da Tabela

SQL - Extração de Dados (5/6)

- Comando *Select* (contextualização)
 - Selecionando apenas algumas Linhas da Tabela
- A cláusula *WHERE* em um comando *SELECT* especifica quais linhas queremos obter, baseada em *<condições de seleção>*

SQL - Extração de Dados (5/6)

□ Comando *Select* (sintaxe)

- Selecionando apenas algumas Linhas da Tabela

```
SELECT <nome(s) da(s) colunas>  
FROM   <tabela>  
WHERE  <condicao>;
```

ONDE condição pode ser mais bem detalhada em:

Campo Operador Valor

```
Ex1: sexo      =      'm'  
Ex2: cidade    =      'Rodelas'  
Ex3: idade     >      20   ...
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (sintaxe)

- Selecionando apenas algumas Linhas da Tabela

```
SELECT <nome(s) da(s) colunas>  
FROM <tabela>  
WHERE <campo> <operador> <valor>
```

□ Operadores

- a) De Comparação
- b) Lógicos
- c) BETWEEN e NOT BETWEEN
- d) LIKE e NOT LIKE
- e) IN e NOT IN
- f) IS NULL e IS NOT NULL

SQL - Extração de Dados (5/6)

- Comando *Select* (contextualização)
 - Selecionando apenas algumas Linhas da Tabela
 - Operadores de Comparação

Operador	Significado
=	Igual
!= ou <>	Diferente
<	Menor que
>	Maior que
>=	Maior ou igual que
<=	Menor ou igual que

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação

- Listar o número do pedido, o código do produto e a quantidade de itens do pedido com a quantidade igual a 30.

```
SELECT num_pedido, cod_produto, quantidade  
FROM item_do_pedido  
WHERE quantidade = 30;
```

- Quais os clientes que moram em Recife?

```
SELECT nome  
FROM cliente  
WHERE cidade = 'Recife';
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação

- Quais os clientes que não moram em Recife?

```
SELECT nome  
FROM cliente  
WHERE cidade <> 'Recife';
```

- Quais os produtos com valor unitário menor ou igual a R\$ 2.00?

```
SELECT descricao  
FROM produto  
WHERE valor_unitario <= 2;
```

SQL - Extração de Dados (5/6)

- Comando *Select* (contextualização)
 - Selecionando apenas algumas Linhas da Tabela
 - Operadores de Comparação
 - Operadores Lógicos

Operador	Significado
AND	“E” lógico
OR	“Ou” lógico
NOT	Negação

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- Listar os produtos que tenham unidade igual a 'm' e valor unitário maior que R\$ 1,00 da tabela PRODUTO.

```
SELECT descricao          FROM  produto
WHERE unidade = 'm' AND valor_unitario > 1;
```

- Liste os salários, as faixas de comissão e os nomes dos vendedores que pertencem a faixa de comissão 'A' ou que ganham um salário fixo acima de R\$ 300,00.

```
SELECT nome, salario_fixo, faixa_comissao FROM  vendedor
WHERE faixa_comissao = 'A' OR      salario_fixo > 300;
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- Mostrar todos os pedidos que não tenham prazo de entrega igual a 15 dias.

```
SELECT num_pedido  
FROM pedido  
WHERE NOT (prazo_entrega = 15);
```

- Mostrar todos os pedidos que não tenham prazo de entrega igual a 15 dias (**sem** utilizar o NOT).

```
SELECT num_pedido  
FROM pedido  
WHERE (prazo_entrega <> 15);
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (contextualização)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- Este operador propicia a pesquisa por uma determinada coluna e selecionando as linhas cujo valor da coluna esteja dentro de uma faixa determinada de valores, sem a necessidade dos operadores >=, <= e AND.
- Tanto o <valor1> quanto o <valor2> têm de ser do mesmo tipo de dado da coluna.

SQL - Extração de Dados (5/6)

□ Comando *Select* (*sintaxe*)

- Selecionando apenas algumas Linhas da Tabela
 - Operadores de Comparação
 - Operadores Lógicos
 - BETWEEN e NOT BETWEEN

WHERE <nome da coluna>

BETWEEN <valor1> AND <valor2>;

WHERE <nome da coluna>

NOT BETWEEN <valor1> AND <valor2>;

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- Listar o código e a descrição dos produtos que tenham o valor unitário na faixa de R\$ 0,97 e R\$ 2,00.

```
SELECT cod_produto, descricao FROM produto  
WHERE valor_unitario BETWEEN 0.97 AND 2;
```

- Listar o código e a descrição dos produtos que não tenham o valor unitário na faixa de R\$ 1,00 e R\$ 2,00.

```
SELECT cod_produto, descricao FROM produto  
WHERE valor_unitario NOT BETWEEN 1 AND 2;
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (contextualização)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE
- Os operadores LIKE e NOT LIKE só trabalham sobre colunas que sejam do tipo CHAR.
- Eles têm praticamente o mesmo funcionamento que os operadores = e <>, porém o poder desses operadores está na utilização do símbolo (%) que pode fazer o papel de “curinga”.
 - % - substitui um ou mais caracteres
 - _ - substitui um caractere qualquer

SQL - Extração de Dados (5/6)

□ Comando *Select* (*sintaxe*)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE

WHERE <nome da coluna> LIKE <valor>

WHERE <nome da coluna> NOT LIKE <valor>

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE

- <nome da coluna> LIKE 'LÁPIS %' pode enxergar os seguintes registros:

- 'LÁPIS PRETO'
- 'LÁPIS BORRACHA'
- 'LÁPIS CERA'

- Ou seja, todos os registros que contenham 'LÁPIS' seguido de qualquer palavra ou conjunto de caracteres.

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE
- <nome da coluna> LIKE '%ÃO' pode enxergar os seguintes registros:
 - 'JOÃO'
 - 'SÃO JOÃO'
 - 'ALÇAPÃO'
 - 'CONDIÇÃO'
- Ou seja, pode enxergar qualquer nome que termine com "ÃO".

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE

- <nome da coluna> LIKE '%INHO%' pode enxergar os seguintes registros:

- 'LEITE NINHO'
- 'DANONINHO'
- 'MINHOTO'

- Ou seja, pode enxergar qualquer nome que possua o termo "INHO" em qualquer parte da palavra.

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE

- Listar todos os produtos que tenham o seu nome começando por “p”.

```
SELECT descricao FROM produto WHERE descricao LIKE 'p%';
```

- Listar todos os produtos que tenham o seu nome terminando com “o”.

```
SELECT descricao FROM produto WHERE descricao LIKE '%o';
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela
 - Operadores de Comparação
 - Operadores Lógicos
 - BETWEEN e NOT BETWEEN
 - LIKE e NOT LIKE
- Listar os vendedores, cujos nomes não começam por “Jo”.

```
SELECT nome_vendedor FROM vendedor  
WHERE nome_vendedor NOT LIKE 'Jo%';
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE

- EXERCÍCIO:

- 1) Cadastre quatro novos clientes na sua base de dados: Ivan, Ivanildo, Ivanilda e Ivana.
- 2) Pergunta-se: as consultas abaixo retornam o mesmo resultado?

```
SELECT * FROM cliente WHERE nome LIKE 'lv_n';
```

```
SELECT * FROM cliente WHERE nome LIKE 'lv_n%';
```

SQL - Extração de Dados (5/6)

□ Comando *Select* (contextualização)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE
- IN e NOT IN

- Esses operadores pesquisam registros que estão ou não contidos no conjunto de valores fornecido.
- Estes operadores minimizam o uso dos operadores =, <>, AND e OR.

SQL - Extração de Dados (5/6)

□ Comando *Select* (sintaxe)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE
- IN e NOT IN

WHERE <nome da coluna> IN <valor>

WHERE <nome da coluna> NOT IN <valor>

SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE
- IN e NOT IN

- Listar os vendedores que são da faixa de comissão A e B.

```
SELECT nome FROM vendedor  
WHERE faixa_comissao IN ('A', 'B');
```

- Listar os vendedores que não são da faixa de comissão B e C.

```
SELECT nome FROM vendedor  
WHERE faixa_comissao NOT IN ('B', 'C');
```


SQL - Extração de Dados (5/6)

□ Comando *Select* (exemplos)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE
- IN e NOT IN

- Listar os clientes que são de estados do Nordeste.

```
SELECT nome FROM cliente  
WHERE estado IN ('AL', 'BA', 'CE', 'MA', 'PB', 'PE', 'PI', 'RN', 'SE');
```

SQL - Extração de Dados (5/6)

- Comando *Select* (*contextualização*)
 - Selecionando apenas algumas Linhas da Tabela
 - Operadores de Comparação
 - Operadores Lógicos
 - BETWEEN e NOT BETWEEN
 - LIKE e NOT LIKE
 - IN e NOT IN
 - IS NULL e IS NOT NULL
 - Esses operadores pesquisam colunas que são nulas, ou não.

SQL - Extração de Dados (5/6)

□ Comando *Select* (sintaxe)

- Selecionando apenas algumas Linhas da Tabela

- Operadores de Comparação
- Operadores Lógicos
- BETWEEN e NOT BETWEEN
- LIKE e NOT LIKE
- IN e NOT IN
- IS NULL e IS NOT NULL

WHERE <nome da coluna> IS NULL <valor>

WHERE <nome da coluna> IS NOT NULL <valor>

SQL - Extração de Dados (5/6)

□ Comando *Select* (sintaxe)

- Selecionando apenas algumas Linhas da Tabela
 - Operadores de Comparação
 - Operadores Lógicos
 - BETWEEN e NOT BETWEEN
 - LIKE e NOT LIKE
 - IN e NOT IN
 - IS NULL e IS NOT NULL
- Mostrar os nomes dos clientes que não tenham estado.

```
SELECT nome  
FROM cliente  
WHERE estado IS NULL;
```

SQL - Extração de Dados (6/6)



Ordenando os Dados Seleccionados



SQL - Extração de Dados (6/6)

- Comando *Select* (*sintaxe*)
 - Ordenando os Dados Selecionados

SELECT <nome da(s) coluna(s)>

FROM <tabela>

WHERE <condição>

ORDER BY <nome da coluna> <ASC | DESC>;

SQL - Extração de Dados (6/6)

- Comando *Select* (contextualização)
 - Ordenando os Dados Seleccionados
 - As palavras *ASC* e *DESC* significam, respectivamente, *ascendente* e *descendente*.
 - A forma *ascendente* de ordenação é assumida como padrão.

SQL - Extração de Dados (6/6)

□ Comando *Select* (exemplos)

- Ordenando os Dados Seleccionados

- Mostrar em ordem alfabética a lista de vendedores e seus respectivos salários fixos.

```
SELECT    nome, salario_fixo
FROM      vendedor
ORDER BY  nome ASC;
```

- Listar os nomes, cidades e estados de todos os clientes, ordenados por cidade de forma descendente.

```
SELECT    nome, cidade, estado
FROM      cliente
ORDER BY  cidade DESC;
```


SQL - Extração de Dados (6/6)

- Comando *Select* (exemplos)
 - Ordenando os Dados Seleccionados

- Mostrar a descrição e o valor unitário de todos os produtos que tenham a unidade 'm', em ordem de valor unitário ascendente.

```
SELECT    descricao, valor_unitario
FROM      produto
WHERE     unidade = 'm'
ORDER BY  valor_unitario ASC;
```

SQL - Extração de Dados (6/6)

□ Comando *Select* (exemplos)

• Ordenando os Dados Seleccionados

- Listar os nomes, cidades e estados de todos os clientes, ordenados por estado e cidade de forma descendente.

```
SELECT nome as nome_cliente, cidade, estado
FROM cliente
ORDER BY estado DESC, cidade DESC;
```

Neste caso, ele ordena primeiro por estado, depois por cidade 😊

nome_cliente	cidade	estado
Karlos	Vitoria de Santo A	PE
Ivan	Recife	PE
Antonio	Petrolina	PE
Josias	Caruaru	PE
Manuel	Vitoria da Conquista	BA
Marcos	Salvador	BA

EXERCÍCIO:

1) Pergunta-se: o resultado seria o mesmo se a consulta fosse esta?

```
SELECT nome as nome_cliente, cidade, estado
FROM cliente
ORDER BY cidade DESC, estado DESC;
```

O que veremos agora...

□ Comando *Select* (mais opções)

- O comando *Select* permite efetuar:

- Consultas em grupos

- Group by
- Having
- Funções de agrupamento
 - count, max, min, avg, sum

SQL - Scripts

```
Create table clube(  
id_clube number(5) not null,  
nome varchar2(60),  
cidade varchar2(30),  
estado char(2),  
primary key (id_clube)  
);
```

SQL - Scripts

```
insert into clube values
```

```
(1,'São Paulo','São Paulo', 'SP');
```

```
insert into clube values
```

```
(2,'Flamengo','Rio de Janeiro', 'RJ');
```

```
insert into clube values
```

```
(3,'Palmeiras','São Paulo', 'SP');
```

```
insert into clube values
```

```
(4,'Corinthians','São Paulo', 'SP');
```

```
insert into clube values
```

```
(5,'Sport','Recife', 'PE');
```

SQL - Scripts

```
Create table jogador (  
id_jogador number(5) not null,  
id_clube number(5),  
nome varchar2(60),  
cidade varchar2(30),  
estado char(2),  
sexo char(2),  
salario number(6),
```

SQL - Scripts

```
insert into jogador values
```

```
(1,1,'Kaka',Gama', 'DF','m','125000');
```

```
insert into jogador values
```

```
(2,1,'Luis Fabiano','São Paulo', 'SP','m','100000');
```

```
insert into jogador values
```

```
(3,2,'Ronaldo','Rio de Janeiro', 'RJ','m','30000');
```

```
insert into jogador values
```

```
(4,2,'Adriano','Rio de Janeiro', 'RJ','m','70000');
```

```
insert into jogador values
```

```
(5,5,'Carlinhos Bala','Recife', 'PE','m','15000');
```

SQL - Scripts

```
insert into jogador values
```

```
(6,4,'Dentinho','São Paulo', 'SP','m','25000');
```

```
insert into jogador values
```

```
(7,4,'Cristiane','São Paulo', 'SP','f','8000');
```

```
insert into jogador values
```

```
(8,3,'Marta','Dois Riachos', 'Al','f','20000');
```

```
insert into jogador values
```

```
(9,5,'Andrea','Recife', 'PE','f','4000');
```

```
insert into jogador values
```

```
(10,1,'Pretinha','São Paulo', 'SP','f','6000');
```


SQL - Extração de Dados (1/2)



Consultas em grupos



SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Agrupamentos (GROUP BY)*
- *Os dados resultantes de uma seleção podem ser AGRUPADOS de acordo com um critério específico.*
- *Exemplos: por sexo, por cidade, por estado, etc.*
- *Apenas uma linha por grupo é apresentada.*

	JOGADORES_POR_CIDADE	CIDADE
	1	Rio de Janeiro
	2	Recife
	1	Maceió
	1	Gama
58	DF	

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Agrupamentos (HAVING)*
- O *HAVING* é equivalente à cláusula *WHERE* em sua função, no entanto, ela aceita apenas *FUNÇÕES DE AGRUPAMENTO* não aceitas na cláusula *WHERE*.
- Quando aparecer a cláusula *HAVING* no *SELECT*, ela deve vir *após a cláusula GROUP BY*.
- *Where* deverá aparecer antes da cláusula *GROUP BY*

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Agrupamentos (HAVING)*
- O *HAVING* especifica os resultados do agrupamento.
- *Eu não quero saber apenas o total de jogadores por cidade. Eu quero que a consulta me retorne os grupos cujo total de jogadores por cidade seja inferior a dois.*

JOGADORES_POR_CIDADE	CIDADE
1	Rio de Janeiro
2	Recife
1	Maceió
1	Gama

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Agrupamentos (HAVING)*
- O *HAVING* especifica os resultados do agrupamento.
- Eu não quero saber apenas o *maior* salário *por estado*.
- Eu quero que a consulta me retorne os grupos cujo *maior* salário *por estado* seja *superior a vinte*.

MAIOR_SALARIO	ESTADO
50	RJ
58	DF
18	PE
58	DF

SQL - Agrupamentos (1/2)

- Comando *Select* (sintaxe)
 - *Agrupamentos (GROUP BY)*

SELECT <nome(s) da(s) colunas>

FROM <tabela>

GROUP BY <coluna-criterio-de-agrupamento>

HAVING <condições>;

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Funções de agrupamento (COUNT)*
- Conta o *total* de tuplas *por grupo* ou *por condição*.

SQL - Agrupamentos (1/2)

- Comando *Select* (exemplos)
 - *Funções de agrupamento (COUNT)*

- Listar o total de jogadores (agrupados) por cidade.

```
SELECT    count(id_jogador) as jogadores_por_cidade, cidade
FROM      jogador
GROUP BY  cidade;
```

Uso de um campo específico (ex: `id_jogador`) conta apenas os valores não nulos do referido campo

- Listar as cidades cujo total de jogadores é superior a um.

```
SELECT    count(*) as jogadores_por_cidade, cidade
FROM      jogador
GROUP BY  cidade
HAVING    count(*) > 1;
```

Uso do `*` conta tudo, inclusive os valores nulos

SQL - Agrupamentos (1/2)

- Comando *Select* (exemplos)
 - *Funções de agrupamento (COUNT)*

- Listar o total de jogadores do sexo masculino.

```
SELECT    count(*) as total_jogadores_sexo_m
FROM      jogador
WHERE     sexo = 'm';
```

- Listar o total de jogadores do estado de pernambuco.

```
SELECT    count(*) as total_jogadores_pe
FROM      jogador
WHERE     estado = 'PE';
```

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Cláusula (DISTINCT)*
- Ao utilizar a função *COUNT(*)*, tem-se a contagem de *todos* os elementos *indistintos* (repetindo-se ou não).
- Com o *DISTINCT*, conta-se apenas os valores, mas não as ocorrências repetidas, ou seja, *serão contados apenas os diferentes valores* que aparecem para determinada situação.

SQL - Agrupamentos (1/2)

□ Comando *Select* (exemplos)

- *Cláusula (DISTINCT)*

- Listando o total de cidades, e o total de cidades distintas inseridas na tabela jogador.

```
SELECT count(cidade) as total_cidades,  
       count(distinct cidade) as total_cidades_distintas  
FROM   jogador;
```

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Funções de agrupamento (MAX)*
- Retorna o *maior* valor de *uma coluna* ou *expressão para o agrupamento*.

SQL - Agrupamentos (1/2)

- Comando *Select* (exemplos)
 - *Funções de agrupamento (MAX)*

- Listar o maior salário dos jogadores (agrupados) por sexo.

```
SELECT    max(salario) as maior_salario_por_sexo, sexo
FROM      jogador
GROUP BY  sexo;
```

- Listar o maior salário dos jogadores (agrupados) por estado.

```
SELECT    max(salario) as maior_salario_por_estado, estado
FROM      jogador
GROUP BY  estado;
```

SQL - Agrupamentos (1/2)

- Comando *Select* (exemplos)
 - *Funções de agrupamento (MAX)*

- Listar o maior salário dos jogadores (agrupados) por estado, com a condição de que o maior salário não pode ser inferior a 70000.

```
SELECT    max(salario) as maior_salario_por_estado, estado
FROM      jogador
GROUP BY  estado
HAVING    max(salario) >= 70000;
```

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Funções de agrupamento (MIN)*
- Retorna o *menor* valor de *uma coluna* ou *expressão para o agrupamento*.

SQL - Agrupamentos (1/2)

- Comando *Select* (exemplos)
 - *Funções de agrupamento (MIN)*

- Listar o menor salário dos jogadores (agrupados) por sexo.

```
SELECT    min(salario) as maior_salario_por_sexo, sexo
FROM      jogador
GROUP BY  sexo;
```

- Listar o menor salário dos jogadores (agrupados) por estado.

```
SELECT    min(salario) as maior_salario_por_estado, estado
FROM      jogador
GROUP BY  estado;
```


SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Funções de agrupamento (AVG)*
- Retorna a *média aritmética* de *uma coluna* ou *expressão para o agrupamento*.

SQL - Agrupamentos (1/2)

- Comando *Select* (exemplos)
 - *Funções de agrupamento (AVG)*

- Listar a média de salários dos jogadores (agrupados) por sexo.

```
SELECT    avg(salario) as media_salario_por_sexo, sexo
FROM      jogador
GROUP BY  sexo;
```

- Listar a média de salários dos jogadores (agrupados) por estado.

```
SELECT    avg(salario) as media_salario_por_estado, estado
FROM      jogador
GROUP BY  estado;
```

SQL - Agrupamentos (1/2)

- Comando *Select* (contextualização)
 - *Funções de agrupamento (SUM)*
- Retorna o *somatório* de *uma coluna* ou *expressão* para o *agrupamento*.
- A função *SUM* ignora os valores NULL.

SQL - Agrupamentos (1/2)

- Comando *Select* (exemplos)
 - *Funções de agrupamento (SUM)*

- Listar o soma dos salários dos jogadores (agrupados) por estado.

```
SELECT    sum(salario) as soma_salario_por_estado, estado
FROM      jogador
GROUP BY  estado;
```

Junções



SQL - Junções

- Comando *Select* (contextualização)
 - *Junções em mais de uma tabela*
- *Muitas vezes, é necessário que a consulta retorne de uma só vez dados de duas ou mais tabelas.*
- *Esta operação de trazer dados de várias tabelas em uma mesma consulta é conhecida como join (junção) e recupera dados com base nos relacionamentos estabelecidos.*
- As junções são implementadas na cláusula **WHERE** do comando **SELECT**.

SQL - Junções

- Comando *Select* (contextualização)
 - *Junções em mais de uma tabela*
- Regras básicas para fazer um junção:
- 1) Operação de join exige que as tabelas envolvidas (direta ou indiretamente) devem estar listadas após a cláusula FROM;
- SELECT ...
- FROM cliente, dvd, cliente_loca_dvd
- WHERE ...

SQL - Junções

- Comando *Select* (contextualização)
 - *Junções em mais de uma tabela*
- Regras básicas para fazer um junção:
- 2) Que cada coluna utilizada no SELECT, nas cláusulas SELECT, WHERE, GROUP By, etc., seja precedida do **nome** ou do apelido **da tabela**:
- SELECT **cliente**.nome, **dvd**.nome
- FROM **cliente**, **dvd**, **cliente_loca_dvd**
- WHERE **dvd**.duracao > 120 ...

SQL - Junções

- Comando *Select* (contextualização)
 - *Junções em mais de uma tabela*
- Regras básicas para fazer um junção:
- 2) Que cada coluna utilizada no SELECT, nas cláusulas SELECT, WHERE, GROUP By, etc., seja precedida do nome ou do apelido da tabela:
- SELECT c.nome, d.nome
- FROM cliente c, dvd d, cliente_loca_dvd cld
- WHERE d.duracao > 120 ...

SQL - Junções

- Comando *Select* (contextualização)
 - *Junções em mais de uma tabela*
- Regras básicas para fazer um junção:
- 3) E por fim, o join exige a criação de uma cláusula WHERE, na qual serão explicitadas as relações entre os pares de tabelas envolvidas.
- SELECT ...
- FROM cliente *c*, dvd *d*, cliente_loca_dvd *cld*
- WHERE *c*.cod = *cld*.cod_cliente AND
- *d*.cod = *cld*.cod_dvd ...

SQL - Junções

- Comando *Select* (contextualização)
 - *Junções em mais de uma tabela*
- Regras básicas para fazer um junção:
- 3) Na prática, o join exige que as tabelas envolvidas na junção estejam ligadas pelo(s) campo(s) em comum na cláusula WHERE. Isto é, $p.k = f.k$
- SELECT a.nome
- FROM aluno a, aluno_cursa_disciplina acd
- WHERE a.cod = acd.cod_aluno
- AND acd.media > 7 ...

SQL - Junções

□ Comando *Select* (exemplo 1)

- *Junções em mais de uma tabela*

- Listar o nome dos jogadores, e dos clubes que defendem, ordenando o resultado por ordem alfabética de nome de jogador.

```
SELECT    j.nome as nome_jogador, c.nome as nome_clube
FROM      jogador j, clube c
WHERE     j.id_clube = c.id_clube
ORDER BY  j.nome
```

```
JOGADOR ( id, id_clube, nome, cidade, estado, sexo, salario )
CLUBE   ( id_clube, nome, cidade, estado )
```

SQL - Junções

□ Comando *Select* (exemplo 2)

- *Junções em mais de uma tabela*

- Listar o nome dos jogadores do São Paulo que são paulistas.

```
SELECT    j.nome as paulistas_do_sao_paulo
FROM      jogador j, clube c
WHERE     j.id_clube = c.id_clube
AND       c.nome = 'São Paulo'
AND       j.estado = 'SP'
```

```
JOGADOR ( id, id_clube, nome, cidade, estado, sexo, salario )
CLUBE   ( id_clube, nome, cidade, estado )
```

SQL - Junções

□ Comando *Select* (exemplo 3)

- *Junções em mais de uma tabela*

- Listar a folha salarial (soma dos salários) do Palmeiras, bem como sua média salarial.

```
SELECT    sum(j.salario) as folha_salarial_palmeiras,  
          avg (j.salario) as media_salarial_palmeiras  
FROM      jogador j, clube c  
WHERE     j.id_clube = c.id_clube  
AND       c.nome = 'Palmeiras'
```

```
JOGADOR ( id, id_clube, nome, cidade, estado, sexo, salario )  
CLUBE   ( id_clube, nome, cidade, estado )
```

SQL - Junções

□ Comando *Select* (exemplo 4)

- *Junções em mais de uma tabela*

- Listar o nome dos jogadores, os seus salários, o nome dos clubes que defendem, além da cidade e o estado de origem do clube, com a condição de que o estado de origem do jogador não seja São Paulo. Apresente os resultados em ordem decrescente de salários.

```
SELECT      j.nome as nome_jogador, j.salario,
            c.nome as nome_clube, c.cidade as cidade_clube,
            c.estado as estado_clube
FROM        jogador j, clube c      WHERE      j.id_clube = c.id_clube
AND         j.estado != 'SP'      ORDER BY    j.salario desc
```

```
JOGADOR ( id, id_clube, nome, cidade, estado, sexo, salario )
CLUBE   ( id_clube, nome, cidade, estado )
```

SQL - Junções

□ Comando *Select* (exemplo 5)

- *Junções em mais de uma tabela*

- Listar o nome do clube que paga o maior salário a um jogador, bem como o nome do jogador (ou dos jogadores) que recebe(m) este salário.

```
SELECT      c.nome as nome_clube,  
            j.nome as nome_jogador_mais_bem_pago  
FROM        jogador j, clube c  
WHERE       j.id_clube = c.id_clube  
AND         j.salario = (SELECT max(salario) FROM jogador)
```

```
JOGADOR ( id, id_clube, nome, cidade, estado, sexo, salario )  
CLUBE   ( id_clube, nome, cidade, estado )
```


SQL - Junções

- Listar os filmes produzidos nos anos 1990 que tiveram a participação de “Wagner Moura”? (*Exemplo 1*)

```
SELECT    f.nome
FROM      filme f, participacao p, profissional_cinema pc
WHERE     pc.nome = 'Wagner Moura'
AND       f.ano between 1990 and 1999
AND       f.cod_filme = p.cod_filme
AND       p.cod_profissional_cinema = pc.cod_profissional_cinema
```

FILME (*cod_filme*, nome, ano, duracao_minutos, preco_sugerido_locacao, censura)

PARTICIPACAO (*cod_papel*, *cod_profissional_cinema*, *cod_filme*)

PROFISSIONAL_CINEMA (*cod_profissional_cinema*, nome, data_nascimento, cidade_nascimento, pais_origem)

SQL - Junções

- Listar os nomes dos profissionais que tiveram alguma participação no filme “Tropa de Elite 2”? (*Exemplo 2*)

```
SELECT      pc.nome
FROM        filme f, participacao p, profissional_cinema pc
WHERE       f.nome = 'Tropa de Elite 2'
AND         f.cod_filme = p.cod_filme
AND         p.cod_profissional_cinema = pc.cod_profissional_cinema
```

FILME (*cod_filme*, nome, ano, duracao_minutos, preco_sugerido_locacao, censura)

PARTICIPACAO (*cod_papel*, *cod_profissional_cinema*, *cod_filme*)

PROFISSIONAL_CINEMA (*cod_profissional_cinema*, nome, data_nascimento, cidade_nascimento, pais_origem)

SQL - Junções

- Listar os nomes dos alunos que são da Bahia, bem como os seus respectivos telefones. (*Exemplo 1*)

```
SELECT  a.nome, fa.fone
FROM    aluno a, fones_aluno fa
WHERE   a.estado = 'BA'
AND     a.mat_aluno = fa.mat_aluno
```

ALUNO (*mat_aluno*, nome, estado, sexo)

ALUNO_CURSA_DISCIPLINA(*mat_aluno*, *cod_disciplina*, media, semestre)

DISCIPLINA(*cod_disciplina*, carga_horaria, titulo, descricao, ementa)

FONES_ALUNO (*mat_aluno*, fone)

SQL - Junções

- Listar os alunos que foram aprovados (isto é, ficaram com média maior ou igual a 7) na disciplina intitulada 'BD2'. Ordene o resultado em forma decrescente de média. (*Exemplo 2*)

```
SELECT  a.nome, fa.media
FROM    aluno a, disciplina d, aluno_cursa_disciplina acd
WHERE   d.titulo      = 'BD2'
AND     acd.media >= 7
AND     a.mat_aluno   = acd.mat_aluno
AND     d.cod_disciplina = acd.cod_disciplina
```

ALUNO (*mat_aluno*, nome, estado, sexo)

ALUNO_CURSA_DISCIPLINA(*mat_aluno*, *cod_disciplina*, *media*, *semestre*)

DISCIPLINA(*cod_disciplina*, carga_horaria, titulo, descricao, ementa)

Referências

- ▣ **Sistemas de Banco de Dados**, Elmasri & Navathe, Pearson, 4^a edição, 2005.