GRADUAÇÃO

UniCesumar

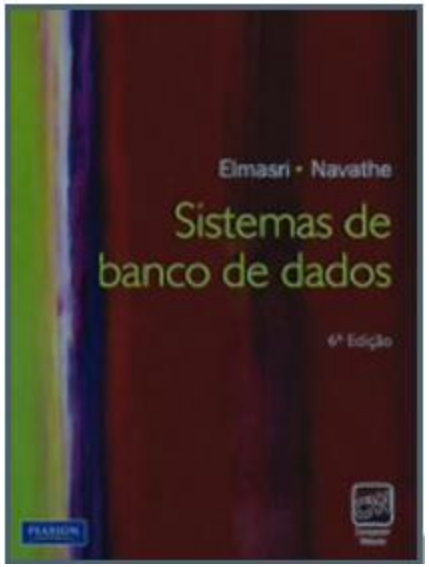
Banco de Dados I

Professor Msc. Aparecido Vilela Junior

aparecido.vilela@unicesumar.edu.br

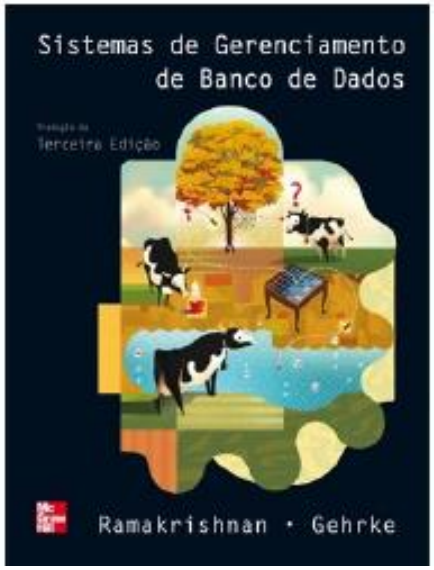
Bibliografia

Elmasri & Navathe – Elmasri, R., Navathe, S. **Sistemas de Banco de Dados**. 6ª Edição, Pearson Brasil, 2011.



Bibliografia

- Silberschatz, A., Korth, H., Sudarshan, S. **Sistema de Banco de Dados**. 6ª Edição, Elsevier-Campus, 2012.
- Ramakrishnan, R. **Sistemas de Gerenciamento de Banco de Dados**, 3ª Edição, McGraw-Hill, 2008.
- Heuser, Carlos Alberto. **Projeto de Banco de Dados – Vol.4**. Série Livros Didáticos Informática UFRGS. 6ª Edição, Editora Bookman, 2009.



Conceitos básicos

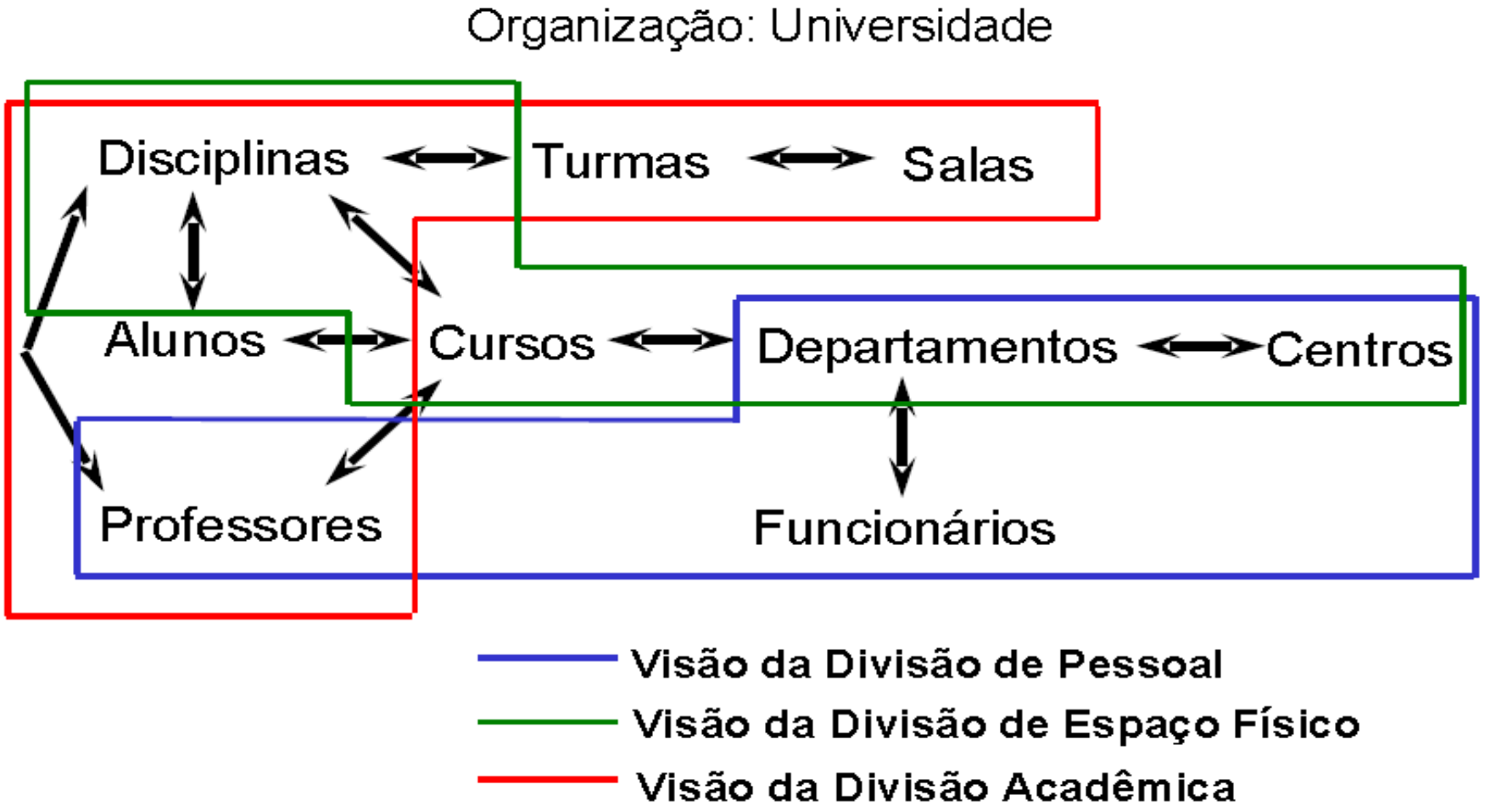
Dado: São fatos conhecidos que podem ser registrados e possuem significado implícito.

- exemplos: endereço, data

- ***Informação***: fato útil que pode ser extraído direta ou indiretamente a partir dos dados
  - exemplos: endereço de entrega, idade

- **Banco de Dados (BD)**: coleção de dados inter-relacionados e persistentes que representa um sub-conjunto dos fatos presentes em um domínio de aplicação(universo de discurso)

Exemplo de um BD



Banco de dados

- Banco de dados = instância de dado + meta-dados

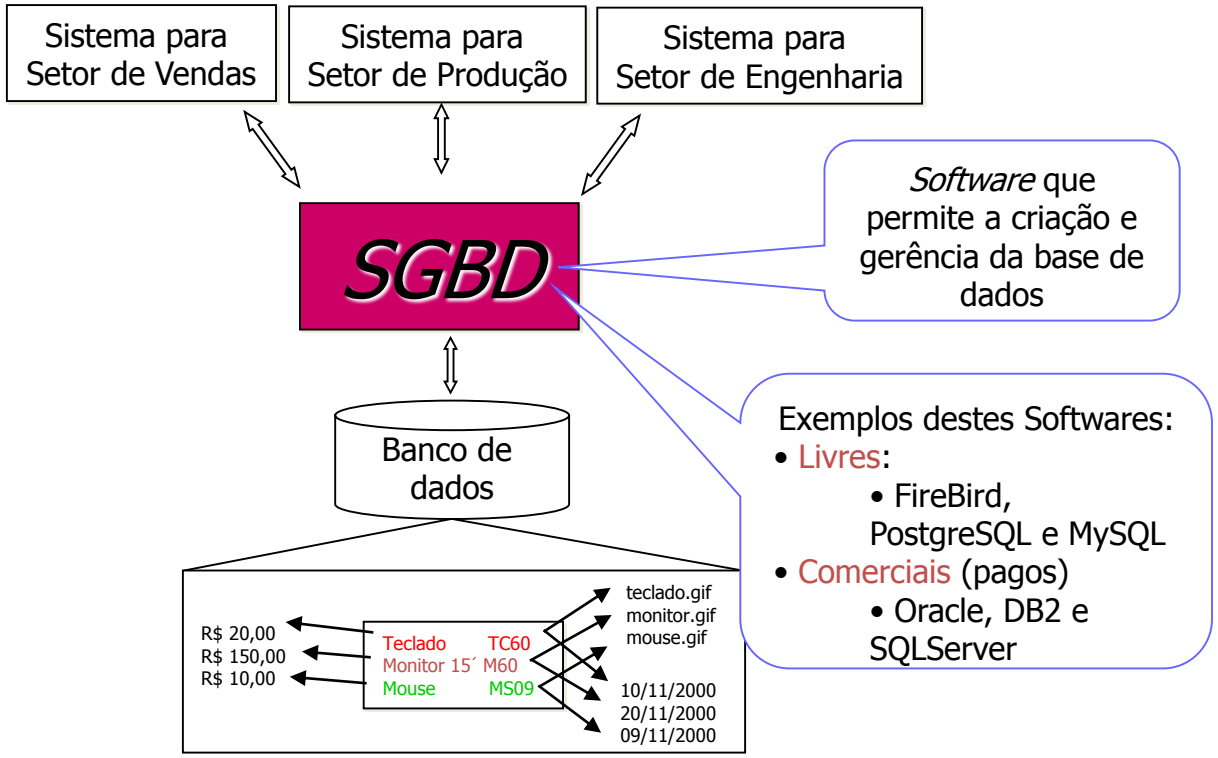
✓ Instância de dado
- Dado propriamente
-

✓ Meta-dados
- *Dicionário de dados*
  - Esquema da base de dados
  - Acessado através de linguagens de definição de dados

▫ A definição ou informação descritiva do banco de dados é armazenada pelo SGBD na forma de um **catálogo** ou **dicionário de dados**, chamado de **Meta-dados**.

Gerenciamento do banco de dados

- BD de uma fábrica:



- Um SGBD (Sistema Gerenciador de Banco de Dados) consiste em uma coleção de dados inter-relacionados **e em um conjunto de programas para acessá-los**

- SGBDs são projetados para gerenciar grandes grupos de informações

SGBD

Armazenamento **multi-usuário** e acesso a **grandes volumes de dados persistentes** de forma:

- **Eficiente** – suporte a milhares de consultas / atualizações por segundo.

- **Confiável** - garantia de que as aplicações podem acessá-lo a qualquer momento.

- **Conveniente** - tornar fácil o acesso e manipulação de grandes quantidades de dados.
  . Independência física dos dados
  . Linguagens de consultas de alto-nível
- **Segura** - garantir que não haja dados inconsistentes mesmo em caso de falhas.

Natureza autodescritiva

O sistema de banco de dados contém a definição completa da estrutura e restrições.

- **Meta-dados** – dados sobre os dados.
  - Descrevem a estrutura do banco de dados.
- **Catálogo** – armazena a descrição (meta-dados) do banco de dados.
  - Usado pelo software SGBD e pelos usuários que precisam de informações sobre a estrutura do banco de dados.

Natureza autodescritiva

- Catálogo

RELACOES

Nome_relacao	Numero_de_colunas
ALUNO	4
DISCIPLINA	4
TURMA	5
HISTORICO_ESCOLAR	3
PRE_REQUISITO	2

COLUNAS

Nome_coluna	Tipo_de_dado	Pertence_a_relacao
Nome	Caractere (30)	ALUNO
Numero_aluno	Caractere (4)	ALUNO
Tipo_aluno	Inteiro (1)	ALUNO
Curso	Tipo_curso	ALUNO
Nome_disciplina	Caractere (10)	DISCIPLINA
Numero_disciplina	XXXXNNNN	DISCIPLINA
....	....	....
....	....	....
....	....	....
Numero_pre_requisito	XXXXNNNN	PRE-REQUISITO

Natureza autodescritiva

A estrutura dos arquivos de dados é armazenada em um catálogo do SGBD separadamente dos programas de acesso.

- Independência da operação do programa
  - Operações especificadas em duas partes:
    . Interface
      . Nome da operação e tipos de dados dos parâmetros.
    . Implementação
      . Pode ser alterada sem afetar a interface.

Compart. de dados e processam. de transações multiusuário

Permite que múltiplos usuários acessem o banco de dados ao mesmo tempo.

- Software de controle de concorrência
  - Garante que vários usuários tentando atualizar o mesmo dado, façam isso de forma controlada.
- Exemplo:
    - Vários agentes de viagem tentando reservar um mesmo assento.

  . O SGBD precisa garantir que cada assento só possa ser acessado por um agente de cada vez.
- Esse tipo de aplicação é chamada de

Processamento de Transação On-line (OLTP).

Compart. de dados e processam. de transações multiusuário

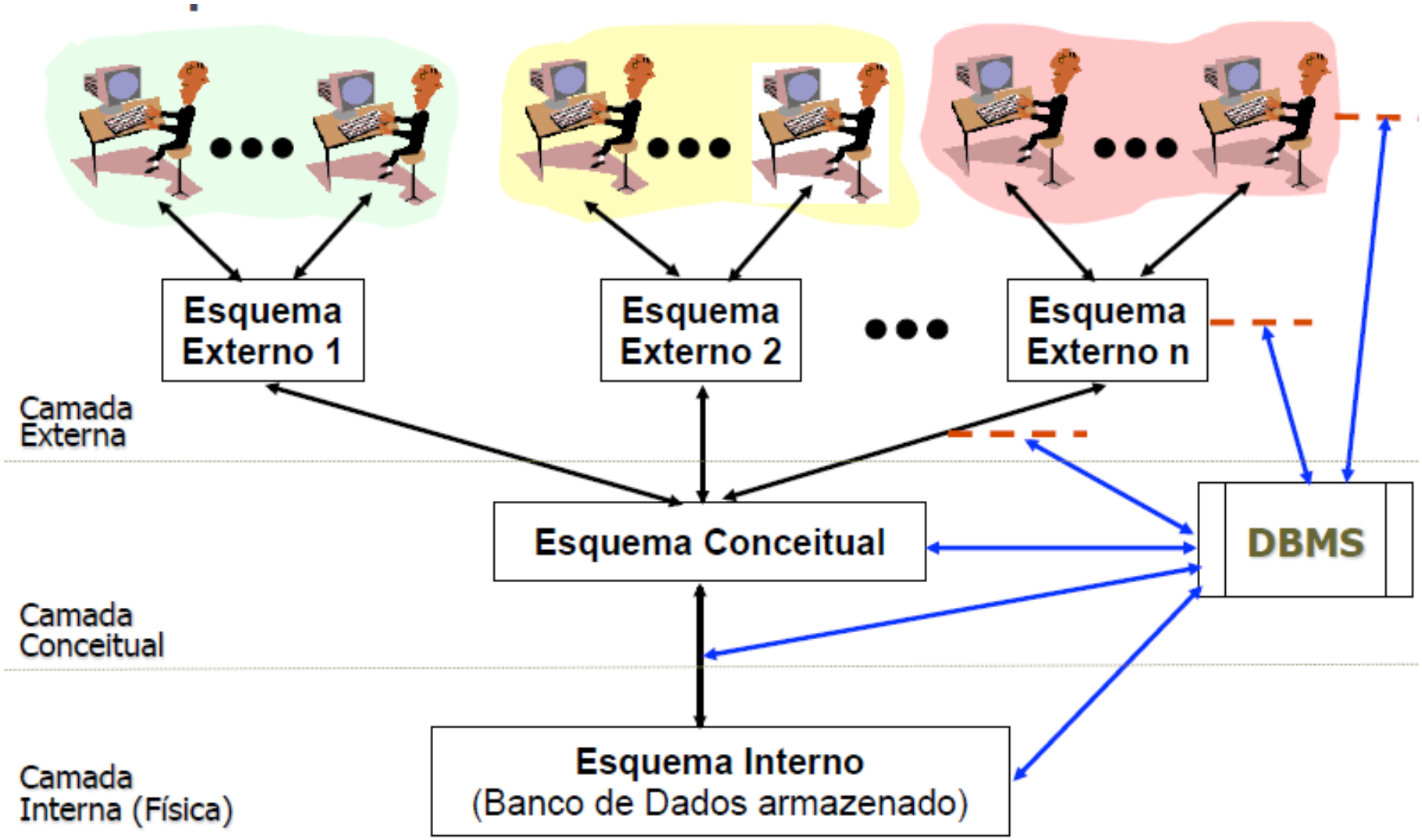
- Transação
  - Unidade de execução de programa que acessa e possivelmente atualiza vários itens de dados.
  - Conceito fundamental para muitas aplicações de banco de dados.
  - Inclui um ou mais acessos ao BD.
  - Executa um acesso logicamente correto a um BD quando ela é executada de forma completa e sem interferência de outras transações.
  - Propriedades das transações:

    . ACID – Atomicidade, Consistência, Isolamento e Durabilidade.

Compart. de dados e processam. de transações multiusuário

- Propriedades ACID:
  - . **Atomicidade**
    - . Ou todos os efeitos de uma transação são refletidos no banco de dados, ou nenhum deles ocorre.
  - . **Consistência**
    - . A execução da transação leva o banco de dados a um estado consistente.
  - . **Isolamento**
    - . A execução de transações concorrentes são isoladas umas das outras.
  - . **Durabilidade**
    - . Atualizações de transações confirmadas não são perdidas, mesmo que ocorra falha do sistema.

Arquitetura em 3 camadas



Fases do projeto de um banco de dados

Especificação e análise de requisitos.

- Projeto conceitual.
- Projeto lógico.
- Projeto físico.

Modelos de Dados

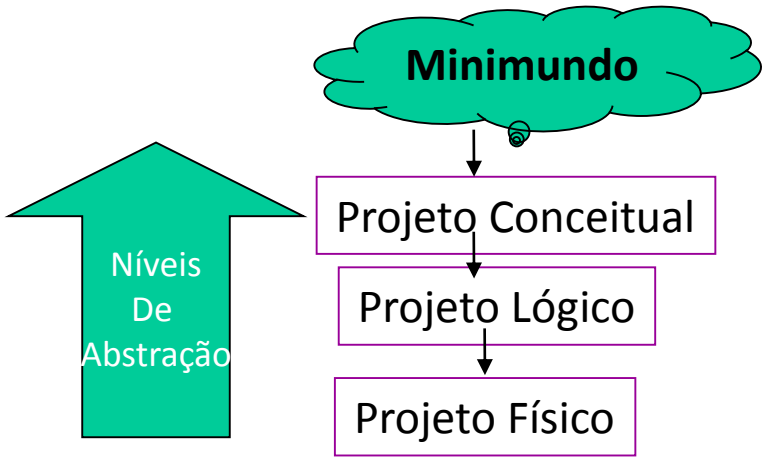
GRADUAÇÃO
UniCesumar

Modelo de dados:

Descrição formal da estrutura de um banco de dados

Modelos propostos:

- Modelo conceitual
- Modelo Lógico
- Modelo Físico



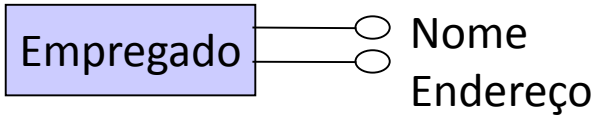
Modelo Entidade-Relacionamento (ER)

Modelos de Dad

Abordado na Aula de Hoje

Modelo conceitual (projeto conceitual)

Modelo de dados abstrato que descreve a estrutura de um banco de dados independente de um SGBD



Modelo lógico (projeto lógico)

Modelo de dados que representa a estrutura dos dados de um banco de dados

Dependente do modelo do SGBD

Empregado (Nome, Endereço)

Modelo físico (projeto físico)

Nível de Implementação

Depende do SGBD

ênfase na eficiência de acesso

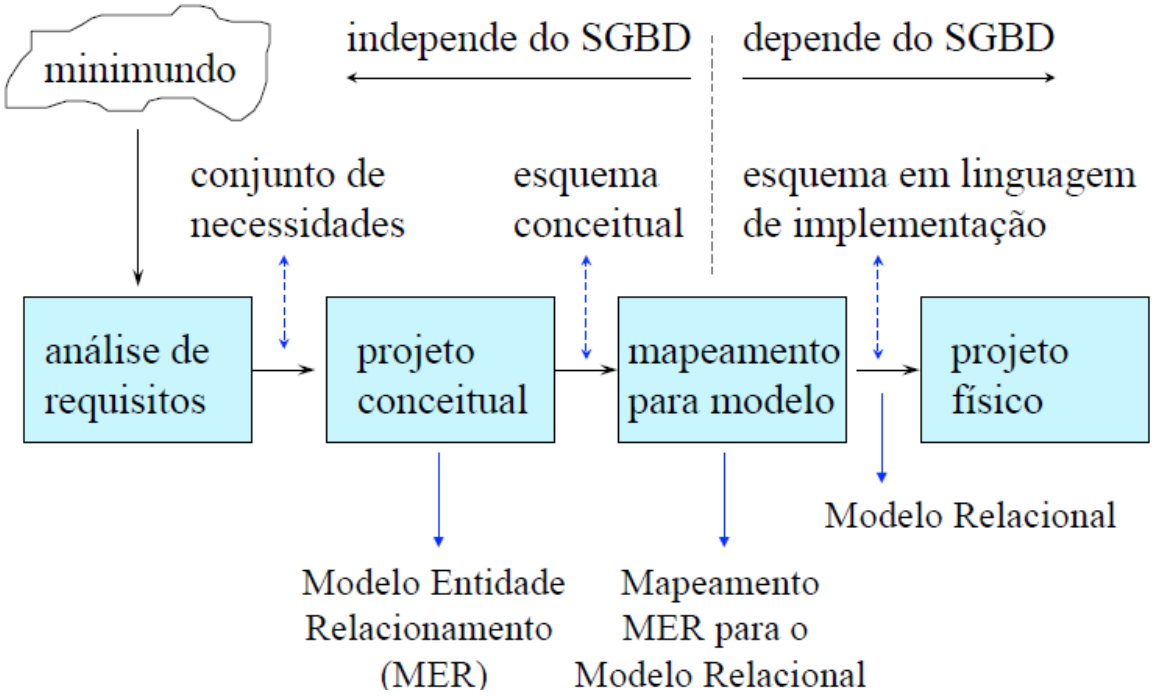
Modelo Entidade-Relacionamento (ER)

Modelo de dados

Código, preço, descrição...

- Exemplo: Livro
- Modelo de dados informa:
  - são armazenadas informações sobre Livros;
  - para cada livro, são armazenados código, preço e título.
- Modelo de dados não informa:
  - quais livros estão armazenados.
- Pode ser representado em forma de texto ou figura (diagrama).
- Cada apresentação do modelo é chamada de esquema do banco de dados.

Modelo de Dados e o Projeto de BD



Modelos de Dados

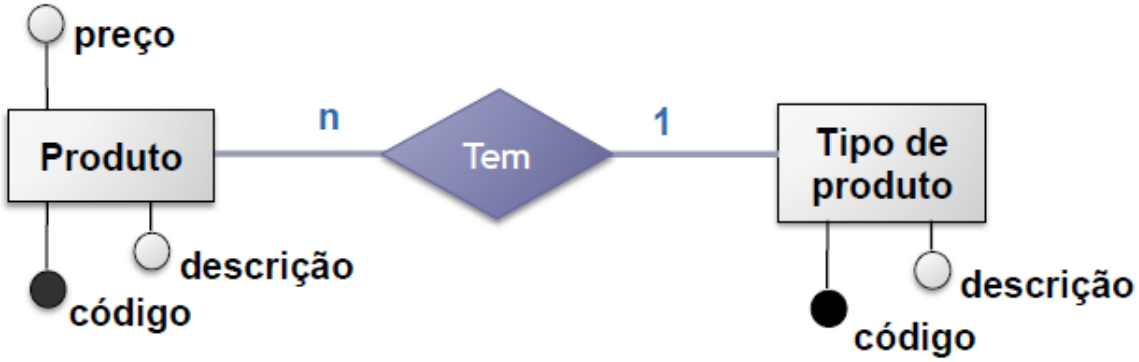
O Modelo Entidade-Relacionamento

Entidades

Atributos

Relacionamentos

- Diagrama Entidade-Relacionamento



Modelo Entidade-Relacionamento (ER)

Modelo Conceitual

Independente de tipo de SGBD;

- Registra a estrutura dos dados;
- Não registra como estes dados estão armazenados no nível de SGBD.

- Técnica mais difundida:
  - Abordagem entidade-relacionamento (ER)

  . Representada através de um Diagrama entidade relacionamento(DER)

Modelagem Conceitual

Modelo Entidade Relacionamento (ER)

- É a técnica mais conhecida
- Tem como objetivo auxiliar na especificação geral do sistema
- O modelo de dados é representado graficamente através de um *Diagrama de Entidade-Relacionamento (DER)*.

Principais conceitos do Modelo ER são:

- Entidades
- Atributos e
- Relacionamentos

Notação: Criada por Peter Chen em 1976

Notação usada: Heuser

Modelo Físico

Contém detalhes de armazenamento interno de informações.

- Detalhes que:
  - não têm influência sobre a programação de aplicações no SGBD, mas, influenciam a performance da aplicações.
  - são usados por profissionais que fazem sintonia (ajuste de desempenho – “tuning”) de banco de dados.

Conceitos Principais

- **Esquema x Dados**
  - **Esquema** - feito inicialmente e geralmente não muda muito.
  - **Dados** - estão em constante mudança.
- **Linguagem de Definição de Dados (DDL)**
  - Usada para definir o Esquema.
- **Linguagem de Manipulação de Dados (DML)**
  - Usada para consultar e modificar os dados.

Projeto de BD

- Fases
  - Modelagem conceitual
  - Projeto lógico
- Processo adequado para a construção de um novo banco de dados.

Atores em cena

- Administradores de banco de dados (DBA)
  - Autorizam acesso ao BD.
  - Coordenam e monitoram o uso do BD.
  - Adquirem recursos de software e hardware.
- Projetistas de banco de dados
  - Identificam os dados a serem armazenados.
  - Escolhem estruturas adequadas para representar e armazenar esses dados.
- Usuários finais
  - Pessoas cujas funções requerem acesso ao BD.
  - Tipos: casuais, iniciantes ou paramétricos, sofisticados, isolados (standalone).
- Analistas de sistemas
  - Determinam os requisitos dos usuários finais.
- Programadores de aplicação
  - Implementam essas especificações como programas.

Quando não usar um BD

- Aplicações de BD simples e bem definidas para as quais não se espera muitas mudanças.

- Requisitos rigorosos, de tempo real, que podem não ser atendidos devido a operações extras executadas pelo SGBD.
- Sistemas embarcados com capacidade de armazenamento limitada, onde um SGBD de uso geral não seria apropriado.
- Nenhum acesso de múltiplos usuários aos dados.

Visão geral a nível prático

- Suponhamos que você precise armazenar os dados de filmes e de seus respectivos diretores. Cada filme tem um código e um título.

Cada diretor tem um código e um nome. Além disso, um filme tem sempre um único diretor, mas um diretor pode dirigir vários filmes.

Passos

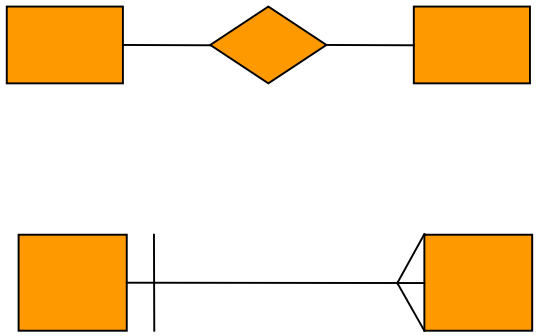
1. Modele o Diagrama Entidade-Relacionamento (DER) para o problema proposto.
2. Crie um Modelo Relacional correspondente ao Diagrama ER criado.
3. Escreva comandos SQL DDL para criar o esquema do Banco de Dados.
4. Crie comandos SQL DML para inserir tuplas nas relações do esquema criado.
5. Crie comandos SQL DML para obter informações relevantes do Banco de Dados criado.
    - Ex: Exibir os nomes dos diretores e quantos filmes cada um deles realizou.

Referências

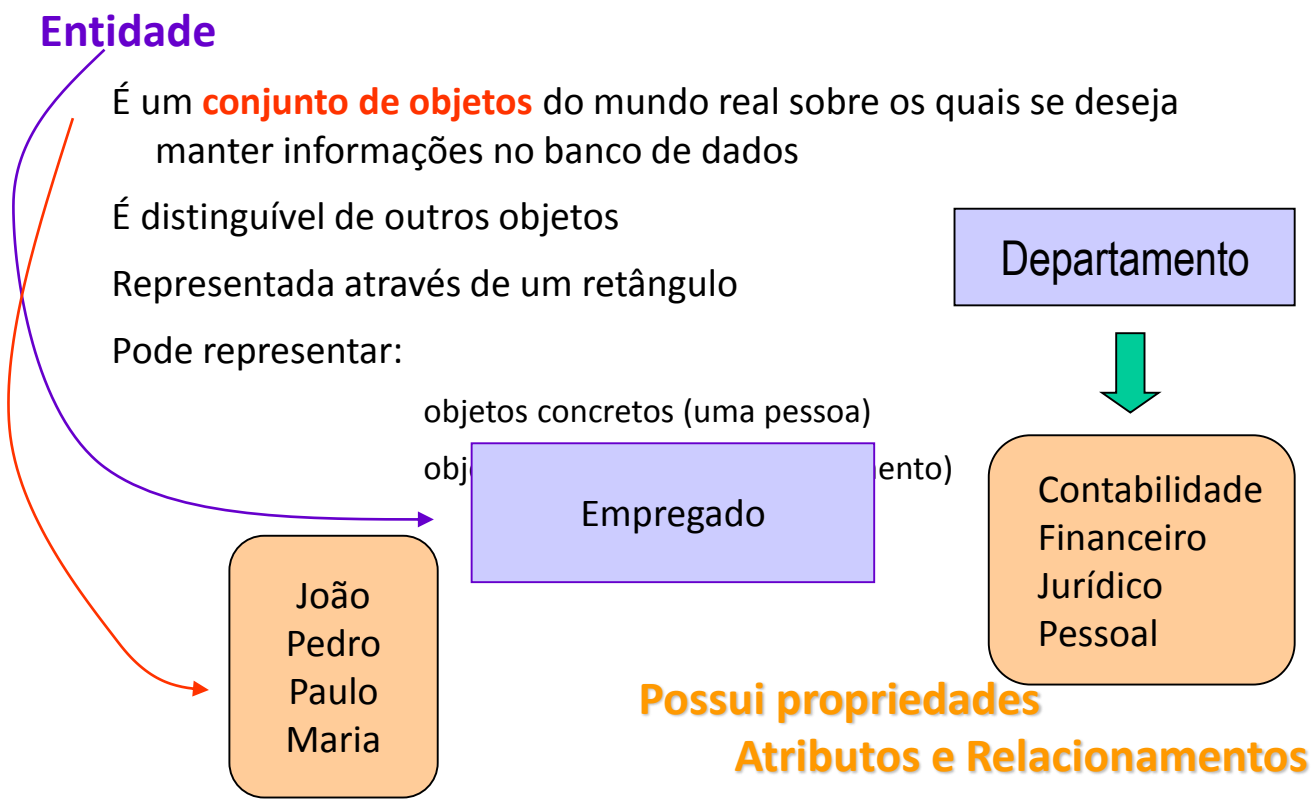
- Elsmari, R., Navathe, Shamkant B. “Sistemas de Banco de Dados”. 6ª Edição, Pearson Brasil, 2011.
- Silberschatz, A., Korth, H., Sudarshan, S. “Sistema de Banco de Dados”. 5ª Edição, Editora Campus, 2006.
- Heuser, Carlos Alberto. “Projeto de Banco de Dados”. 6ª Edição, Editora Bookman, 2009.

Modelo de Entidades e Relacionamentos (MER)

- Representação semântica das estruturas de dados mantidas num banco de dados
- Foi proposto por Peter Chen em 1976
- Possui várias notações:
  - Relacionamentos como objetos do Modelo (Chen)
  - Relacionamentos apenas como simples ligações (Codd, Martin)



Entidade


Entidade
É um conjunto de objetos do mundo real sobre os quais se deseja manter informações no banco de dados
É distinguível de outros objetos
Representada através de um retângulo
Pode representar:
objetos concretos (uma pessoa)
obj
ento)
Empregado
João
Pedro
Paulo
Maria
Departamento
Contabilidade
Financeiro
Jurídico
Pessoal
Possui propriedades
Atributos e Relacionamentos

Atributos

São as propriedades que caracterizam ou descrevem uma entidade ou um relacionamento.

Ex.: A entidade CARRO poderia ter os seguintes atributos:

Placa, fabricante, modelo, ano de fabricação, cor, preço

Atributos

Simples: é atômico.

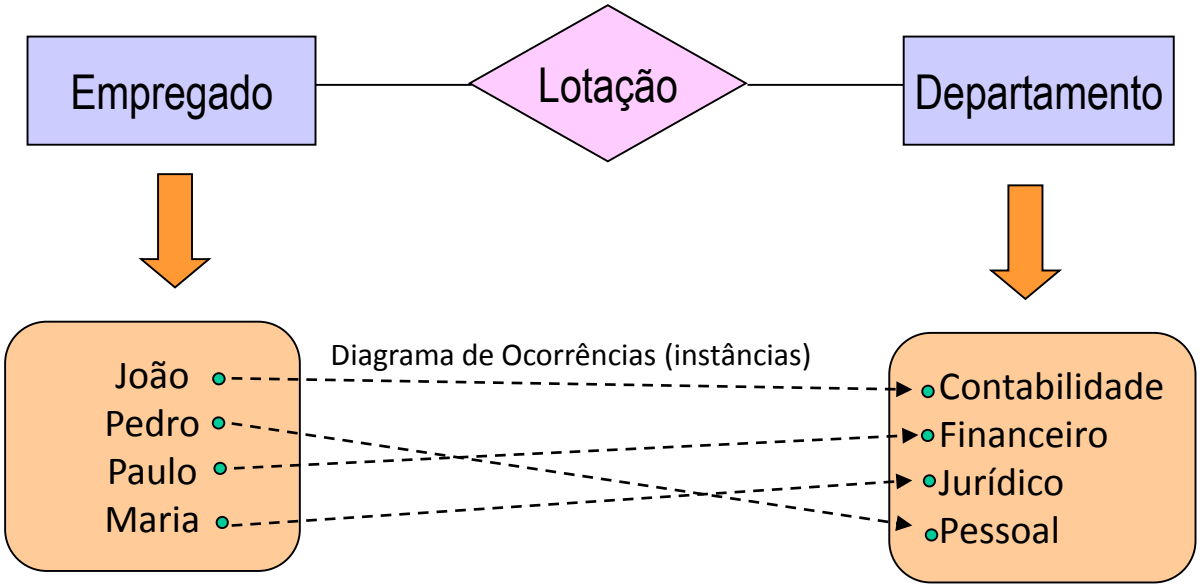
Ex. Idade: numérico Nome: cadeia de caracteres

Composto: contém sub-atributos que compõem o atributo.

Ex.: Endereço(rua, número, bairro, CEP, cidade,)

Relacionamentos

Como expressamos que João trabalha no Departamento de Contabilidade?

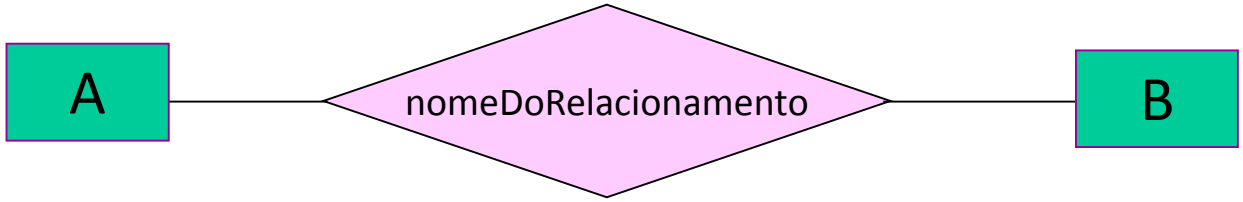


Relacionamentos

Relacionamento:

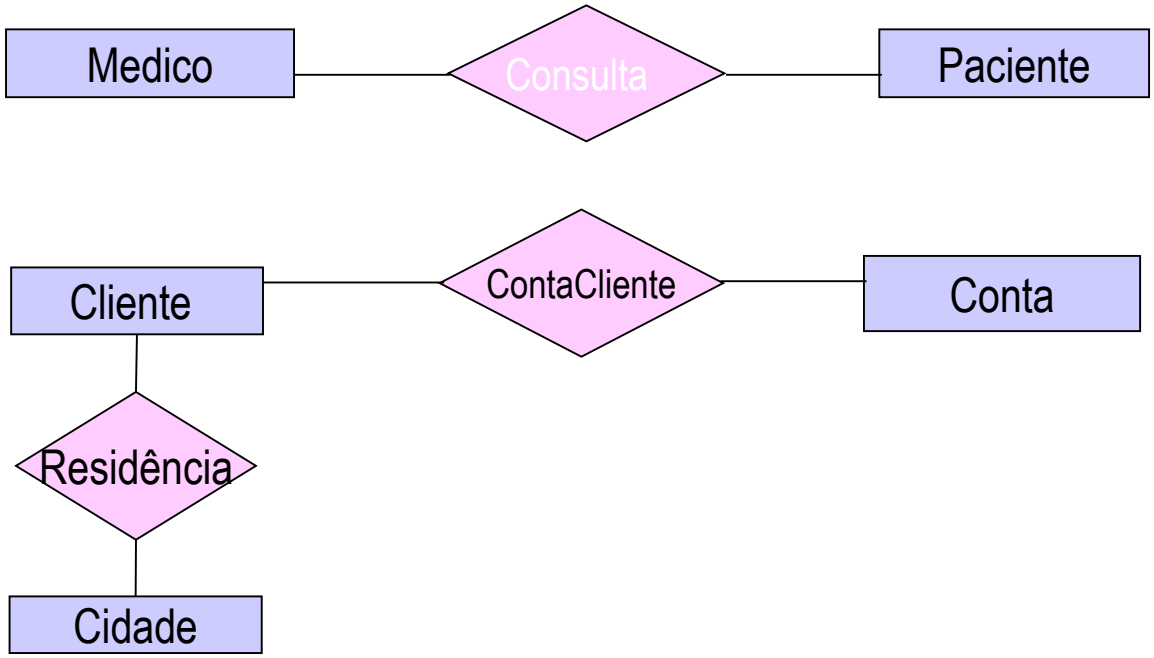
É uma associação entre entidades

Representado através de um losângulo e linhas que ligam as entidades relacionadas



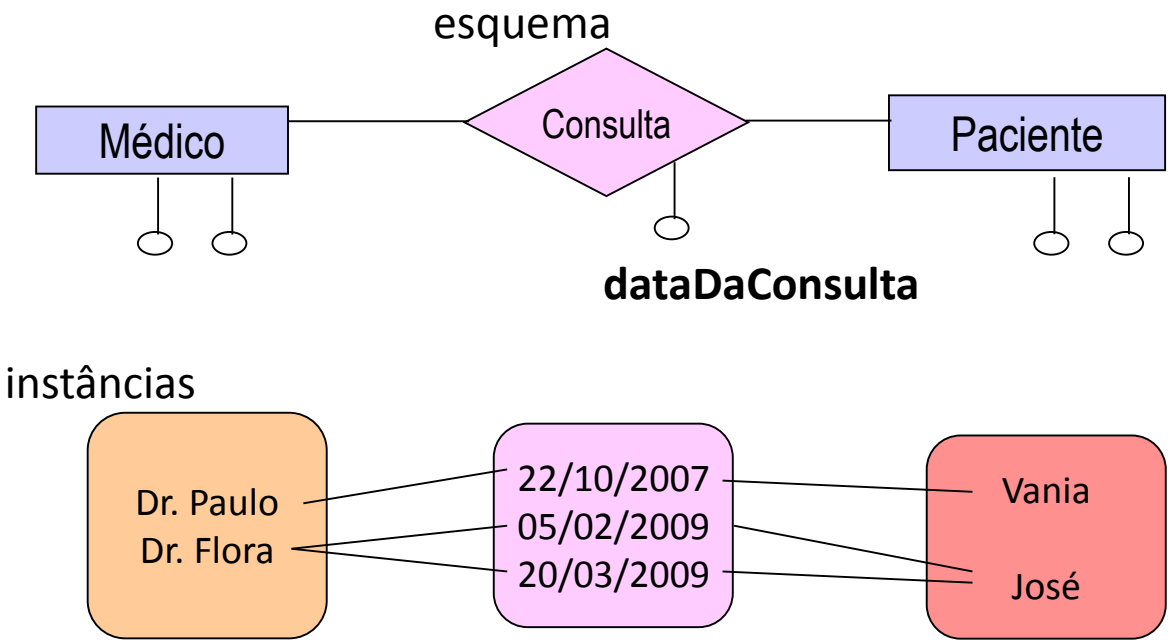
Relacionamentos

Exemplos de Relacionamentos



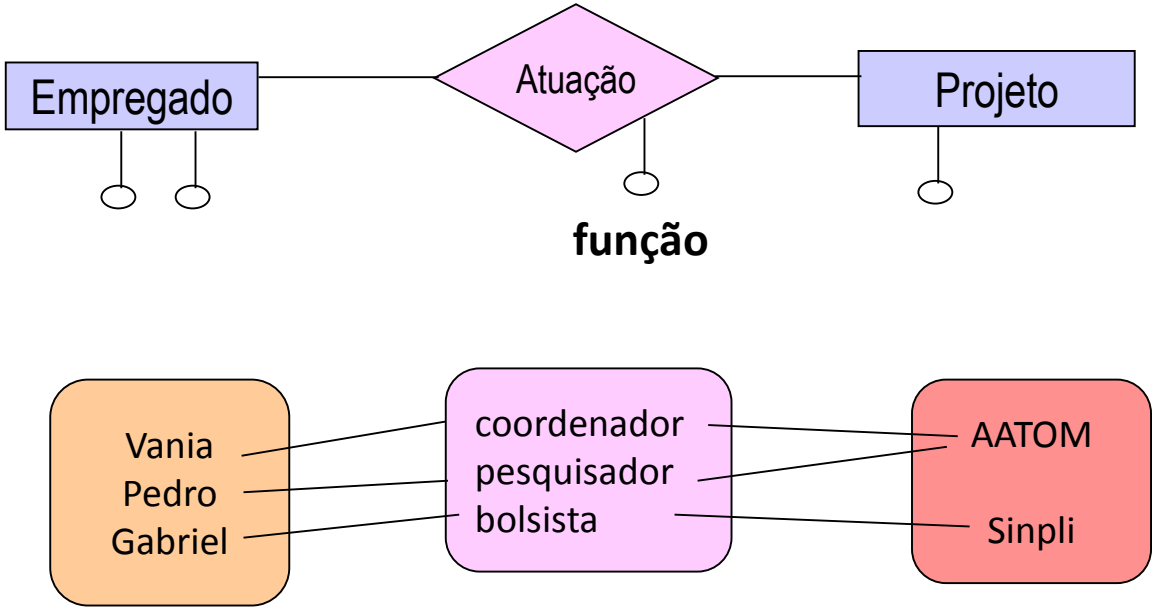
Relacionamentos com Atributos

Exemplo I



Relacionamentos com Atributos

Exemplo II

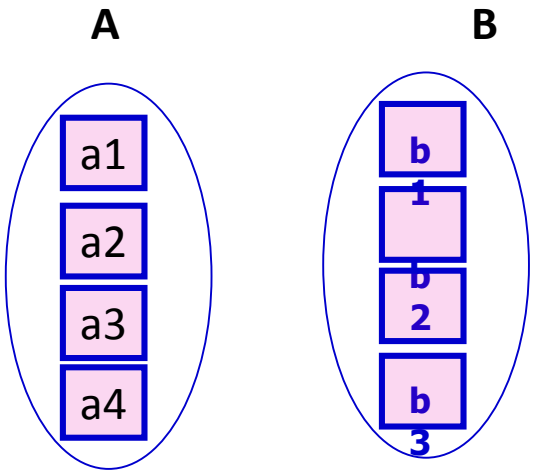


Modelo

Cardinalidade de Relacionamentos

- Uma propriedade importante dos relacionamentos é a especificação de quantas ocorrências de uma entidade podem estar associadas a uma determinada ocorrência de outra entidade

  - Existem 2 cardinalidades:
    - Máxima
    - Mínima

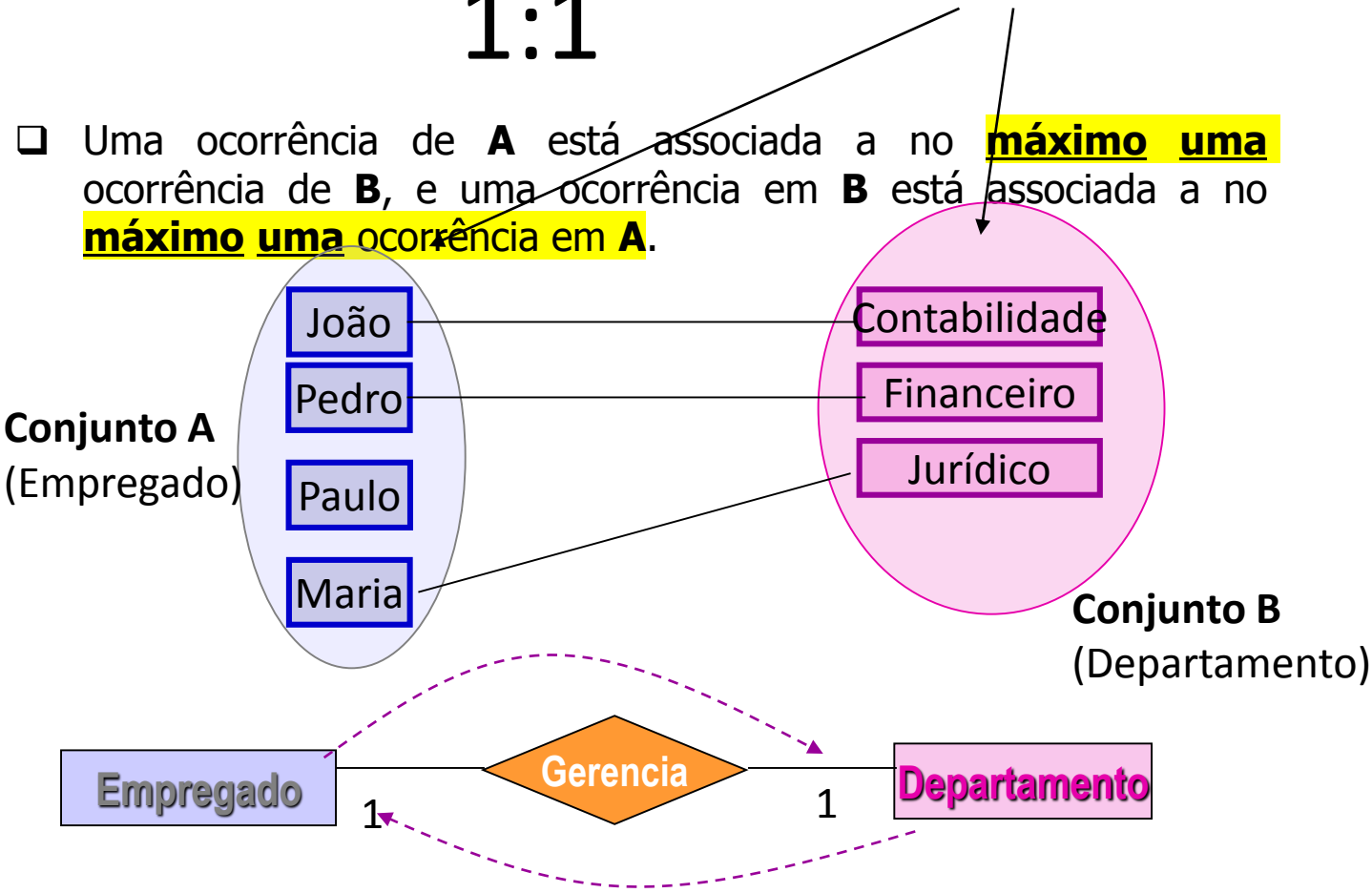


A ocorrência a1 da entidade A está relacionado a quantas Ocorrências em B?

Cardinalidade Máxima

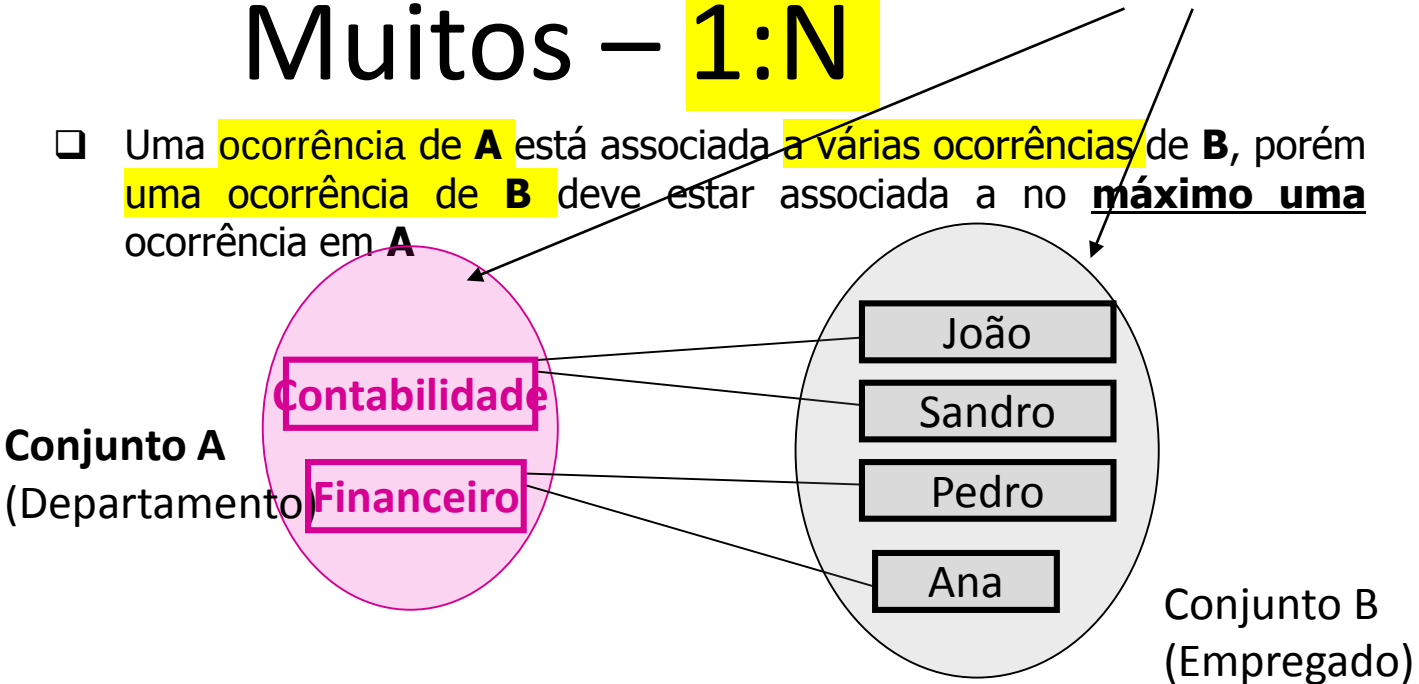
Relacionamento Um para Um – 1:1

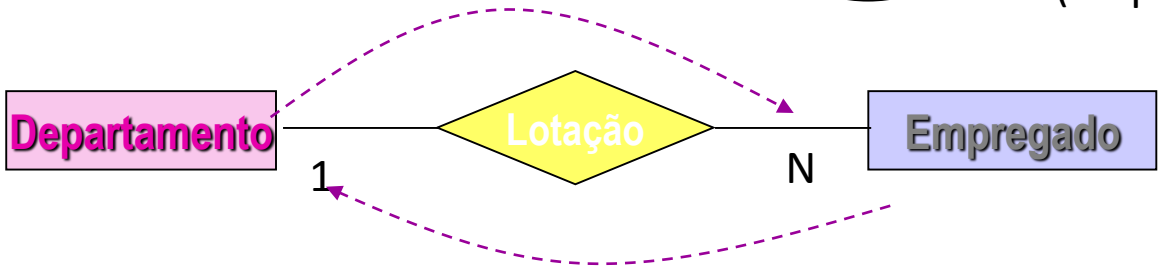
- Uma ocorrência de **A** está associada a no **máximo uma** ocorrência de **B**, e uma ocorrência em **B** está associada a no **máximo uma** ocorrência em **A**.



Relacionamento Um para Muitos – 1:N

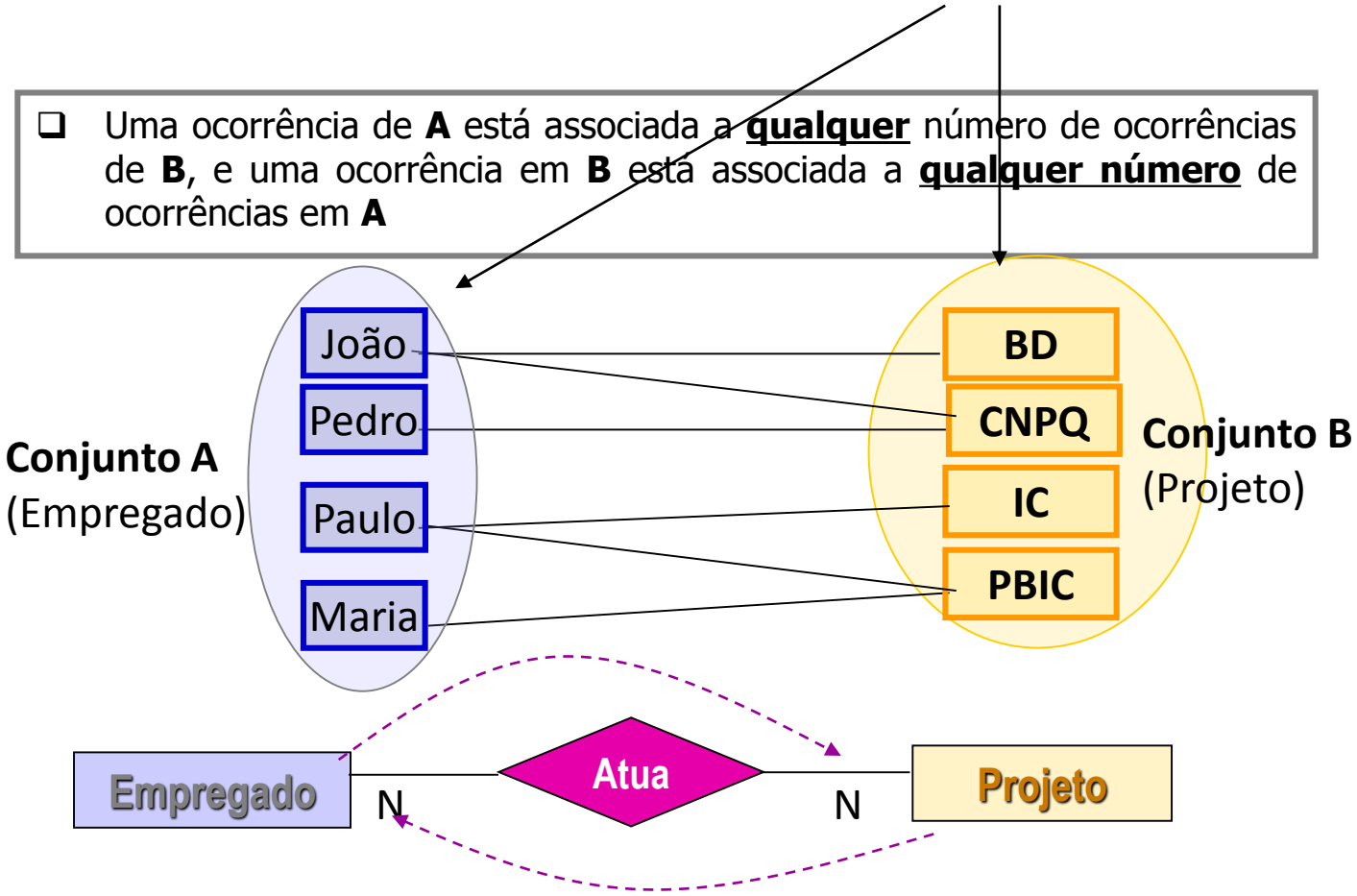
- Uma ocorrência de **A** está associada a várias ocorrências de **B**, porém uma ocorrência de **B** deve estar associada a no **máximo uma** ocorrência em **A**





Relacionamento Muitos para Muitos – M:N ou N:N

- Uma ocorrência de **A** está associada a **qualquer** número de ocorrências de **B**, e uma ocorrência em **B** está associada a **qualquer número** de ocorrências em **A**



Cardinalidade Mínima

O modelo ER permite expressar cardinalidades mínimas e máximas em cada relacionamento

Cardinalidade Mínima:

número mínimo de ocorrências de uma entidade A com relação a uma outra entidade B

Representação:

(cardinalidade mínima, cardinalidade máxima)

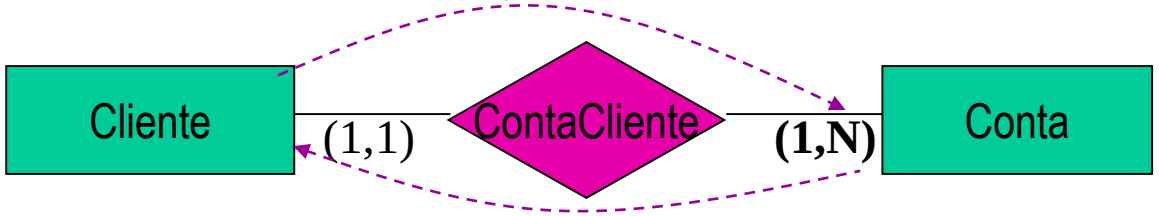
Cardinalidades Possíveis: (1,1); (1,N); (0,1);(0,N)

Cardinalidade **mínima** = 1 (relacionamento obrigatório)

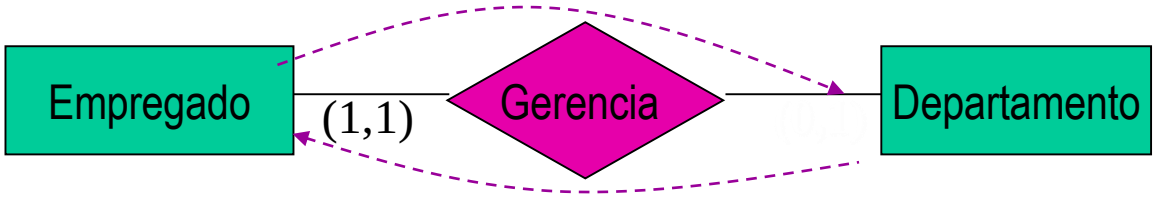
Cardinalidade **mínima** = 0 (relacionamento opcional)

Cardinalidade Mínima e Máxima

- Exemplo de Relacionamento **Obrigatório**:
  - cada ocorrência de cliente está relacionado a no mínimo quantas contas e no máximo quantas contas?
  - Cada ocorrência de conta está relacionada a no mínimo quantos clientes e no máximo quantos clientes?



- Exemplo de Relacionamento **Opcional**:



Banco de Dados
Mapeamento de ER

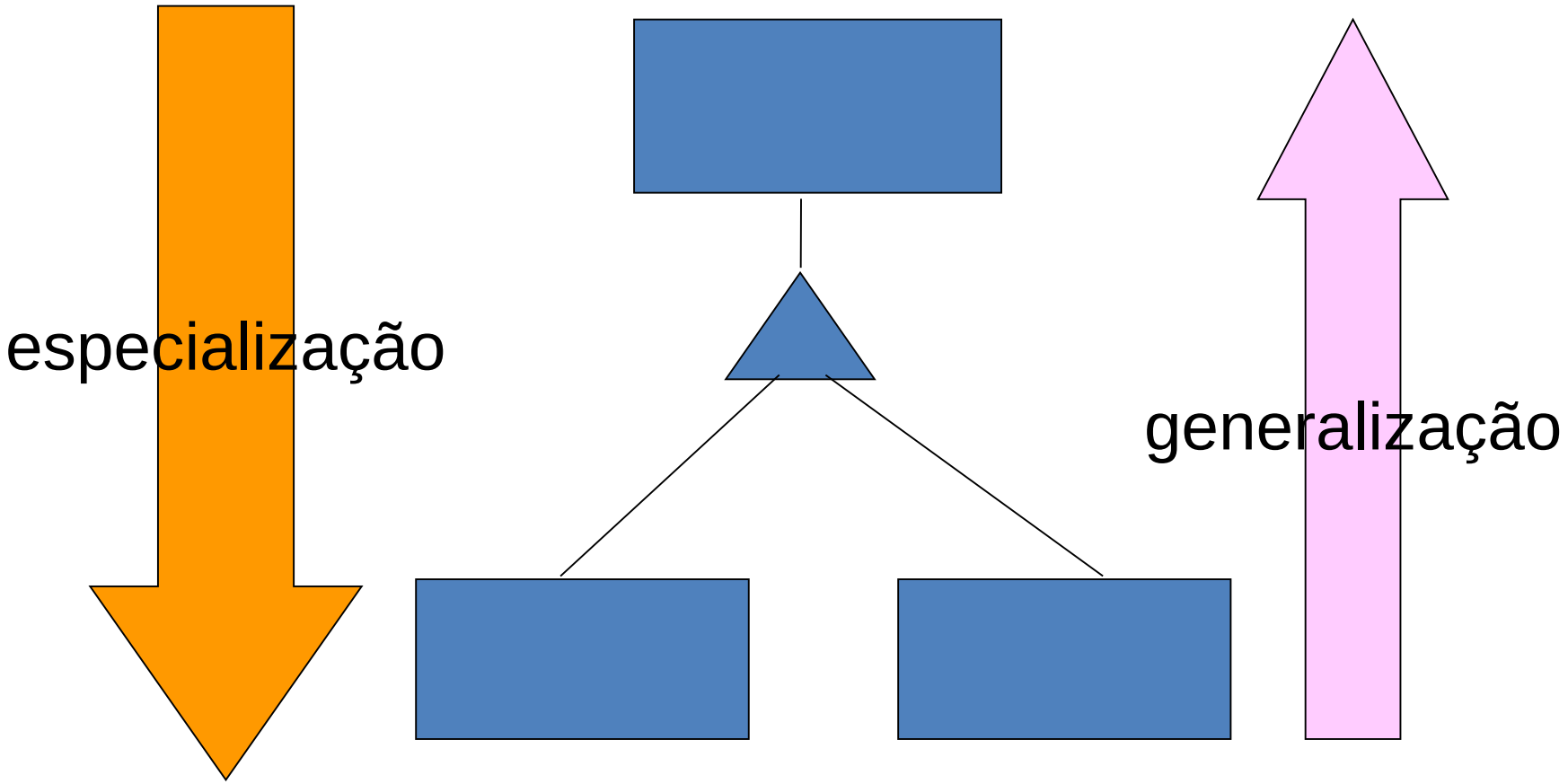
Relacionamento de Generalização

GRADUAÇÃO UniCesumar

- É um relacionamento de classificação entre um elemento mais geral e outro mais específico
- O elemento mais geral tem todas as características (atributos) que são comuns aos elementos específicos → define herança
- O elemento mais geral é denominado entidade de nível superior (superclasse) e o mais específico de entidade de nível inferior (subclasse)
- As características do nível superior são herdadas no nível inferior
  - Por isso o processo é conhecido como herança
- Representado por um triangulo isósceles

Relacionamento de Generalização

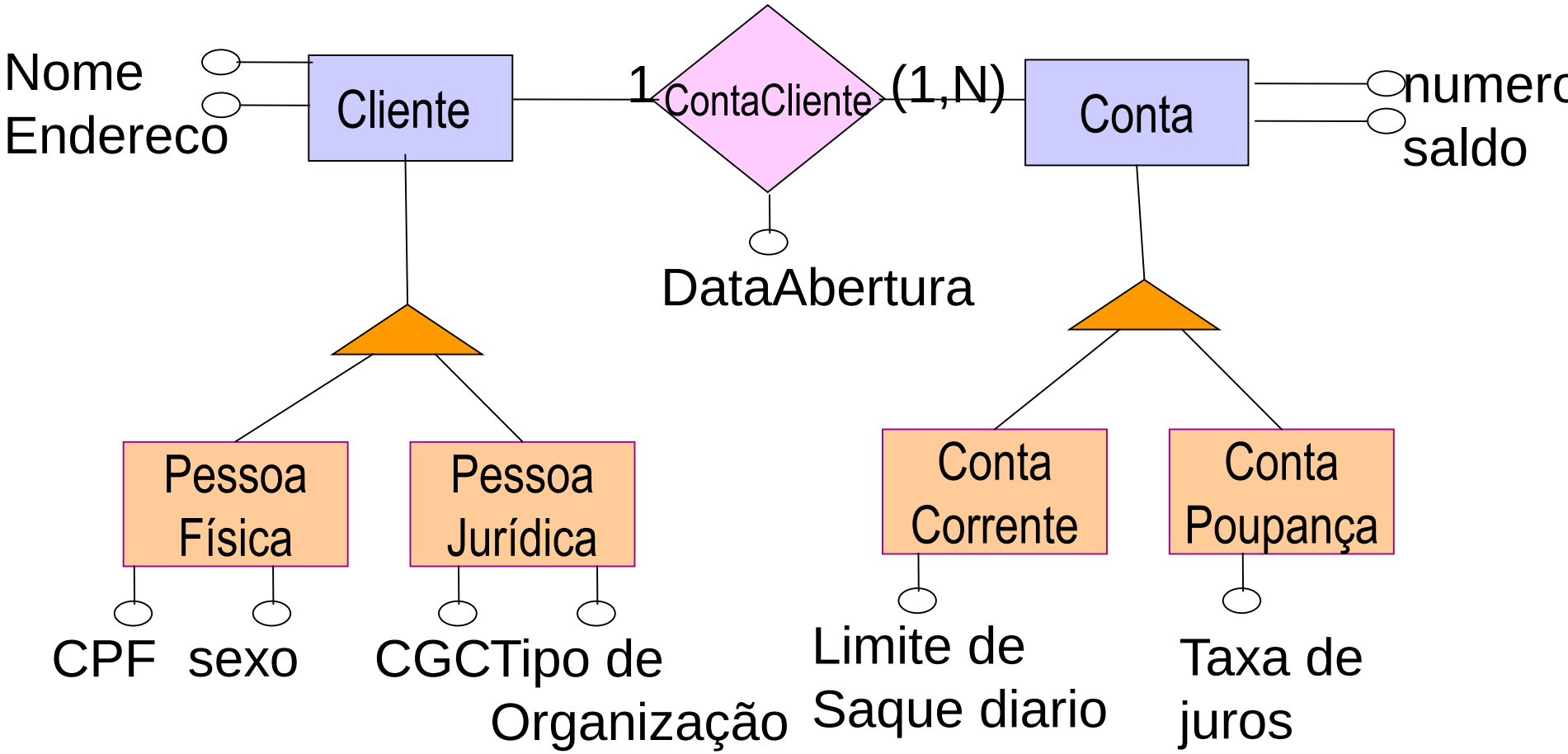
GRADUAÇÃO
UniCesumar



Herança de Propriedades

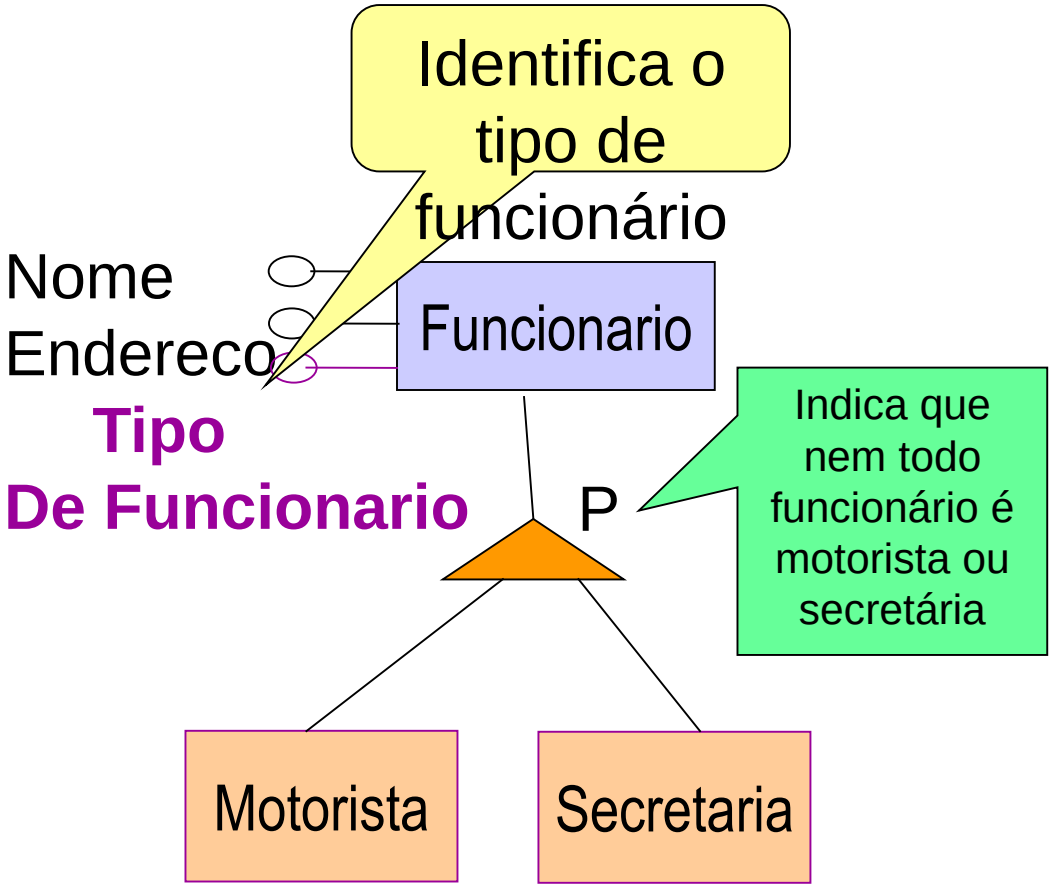
- Significa
  - cada ocorrência da entidade especializada (subclasse) possui
    - além de suas próprias propriedades
    - as propriedades da entidade genérica (superclasse)
- Não há limites no níveis da hierarquia

Exemplo


Nome
Endereco
Cliente
1
ContaCliente
(1,N)
Conta
numero
saldo
DataAbertura
Pessoa Física
Pessoa Jurídica
Conta Corrente
Conta Poupança
CPF
sexo
CGC
Tipo de Organização
Limite de Saque diario
Taxa de juros

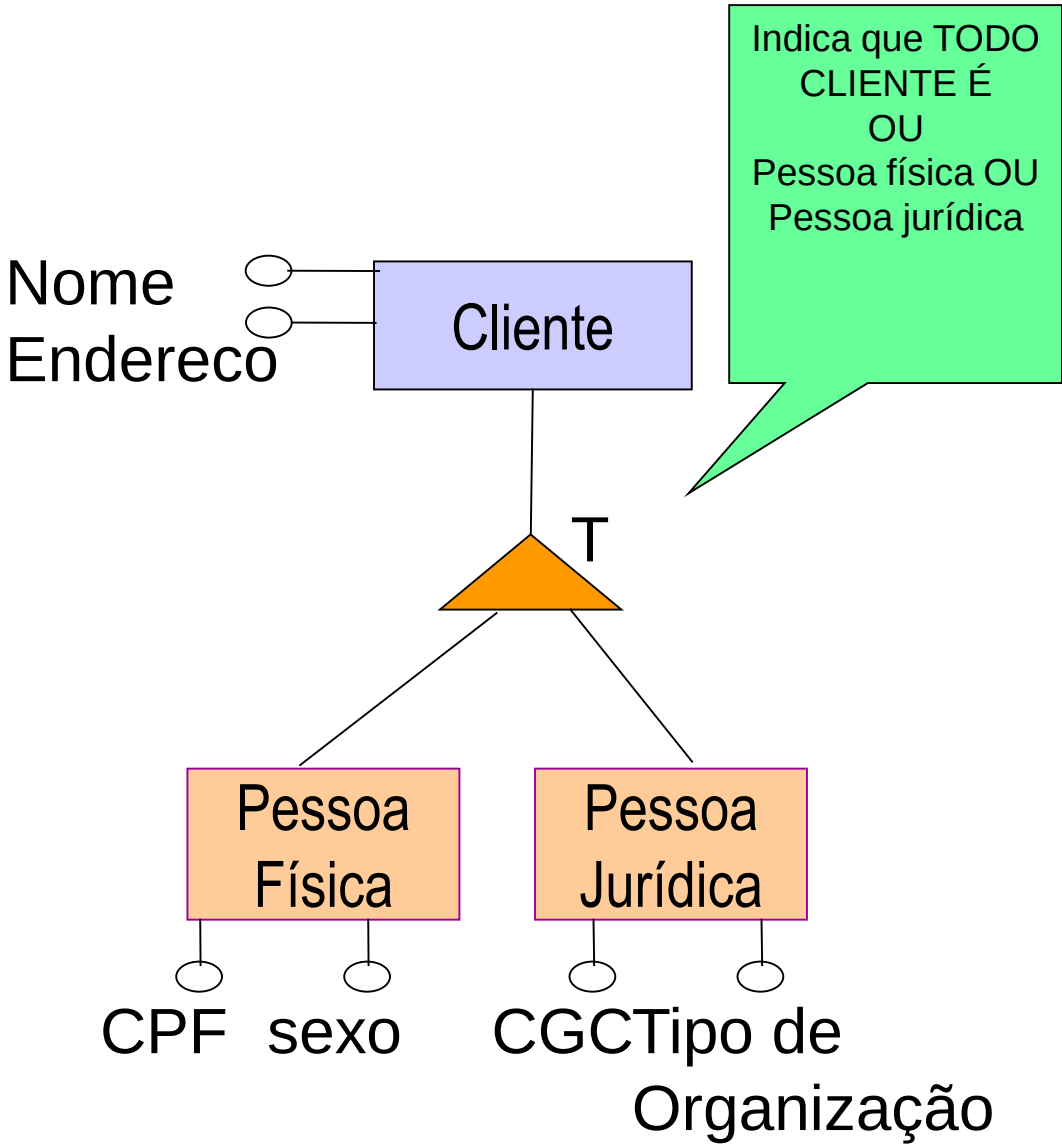
Generalização Parcial

- **Nem toda** ocorrência da entidade genérica possui uma ocorrência correspondente em uma entidade especializada
- A ocorrência pode estar na classe genérica



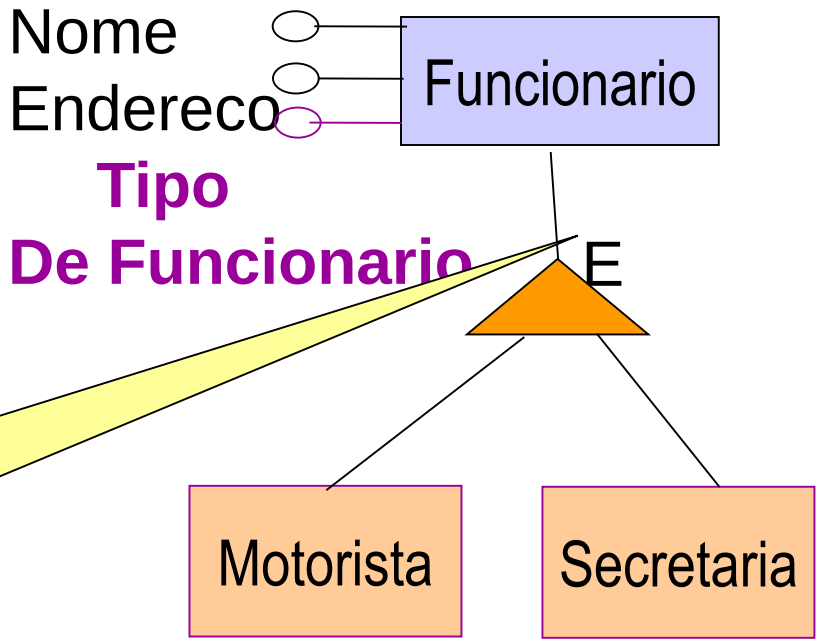
Generalização Total

- Para cada ocorrência da entidade genérica existe **sempre** uma entidade especializada
- A ocorrência está sempre na entidade especializada



Generalização Compartilhada/Exclusiva

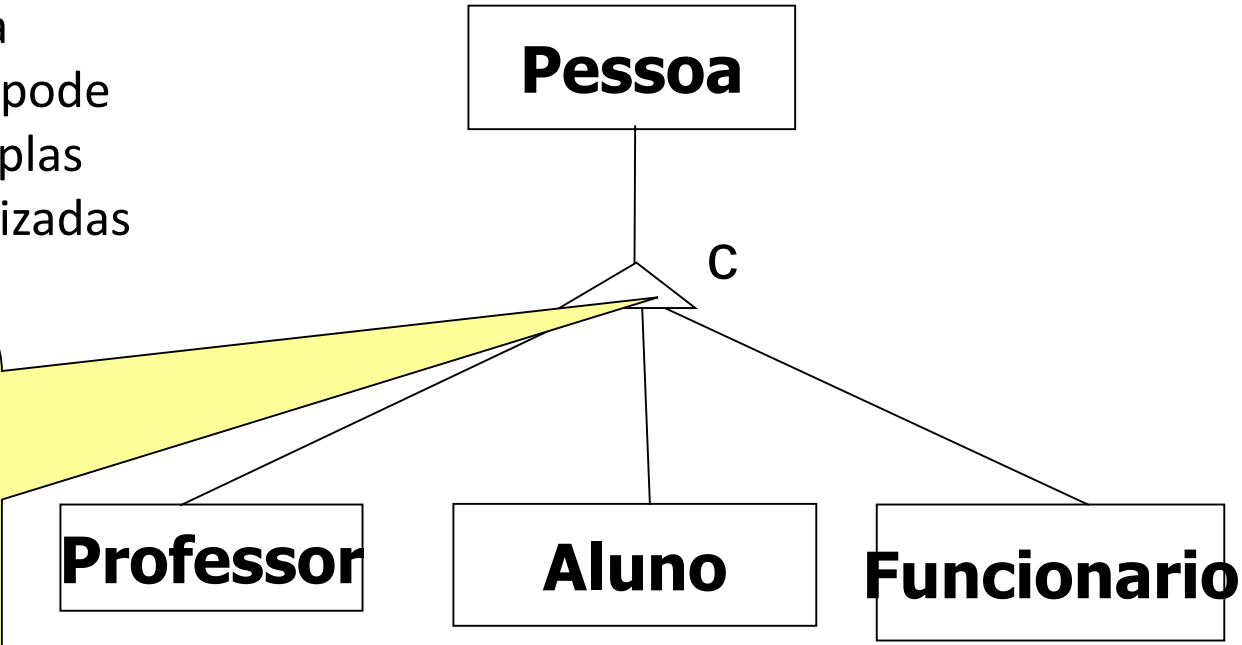
- **Exclusiva**
  - A ocorrência da entidade especializada é exclusiva, aparecendo em apenas uma das entidades especializadas



O Funcionário somente pode ser OU Motorista OU ~~Secretaria, jamais~~ ambos

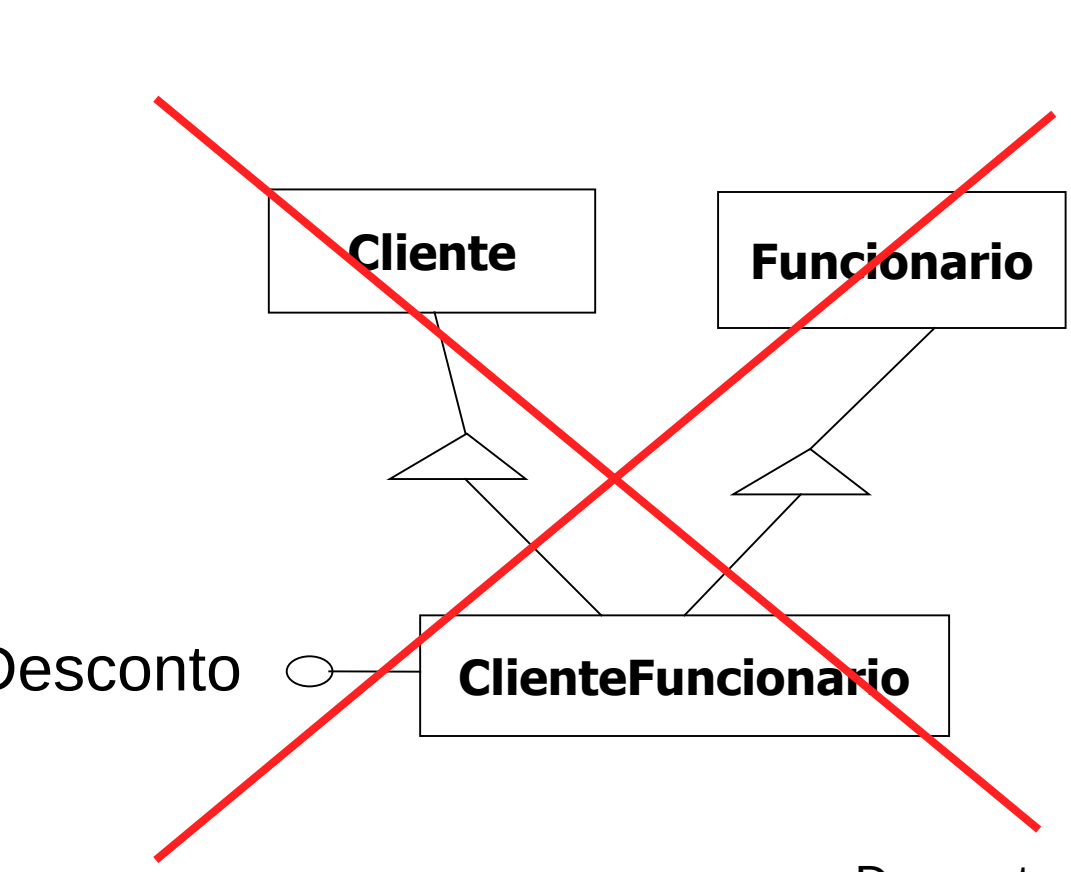
Generalização Compartilhada/Exclusiva

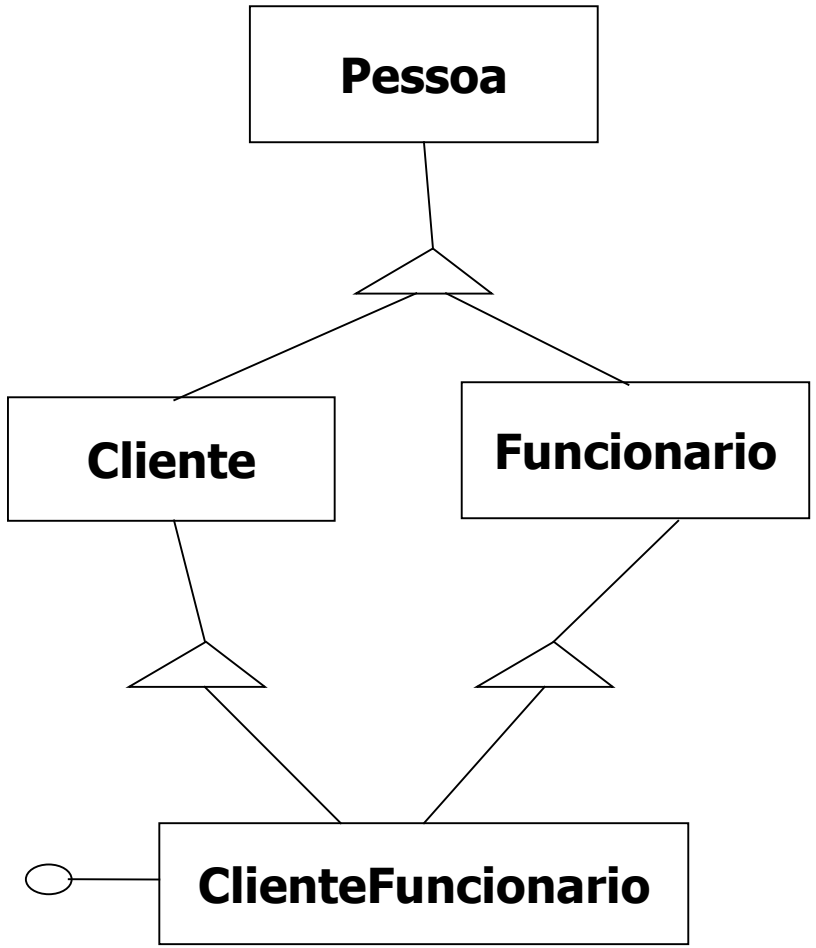
- **Compartilhada**
  - Uma ocorrência da entidade genérica pode aparecer em múltiplas entidades especializadas



A pessoa em uma universidade pode ser um professor (na graduação), ser um funcionário e ser um aluno (de doutorado)

Herança Múltipla

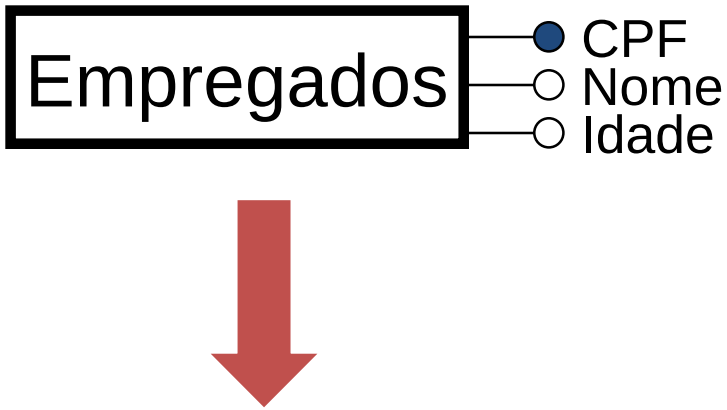




Modelagem Lógica de BD

- Foco
  - mapeamento ER->relacional
- Para 1 esquema ER – N esquemas relacionais
  - existem várias maneiras de “se implementar” uma modelagem conceitual abstrata

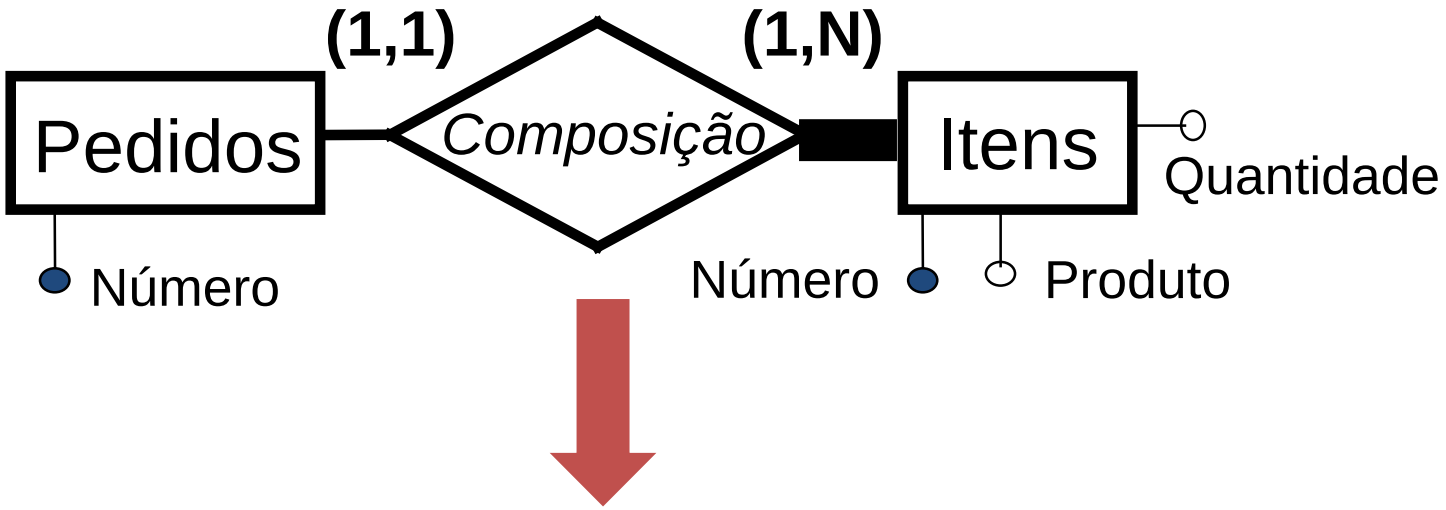
Mapeamento de Entidades



Empregados (<u>CPF</u>, Nome, Idade)

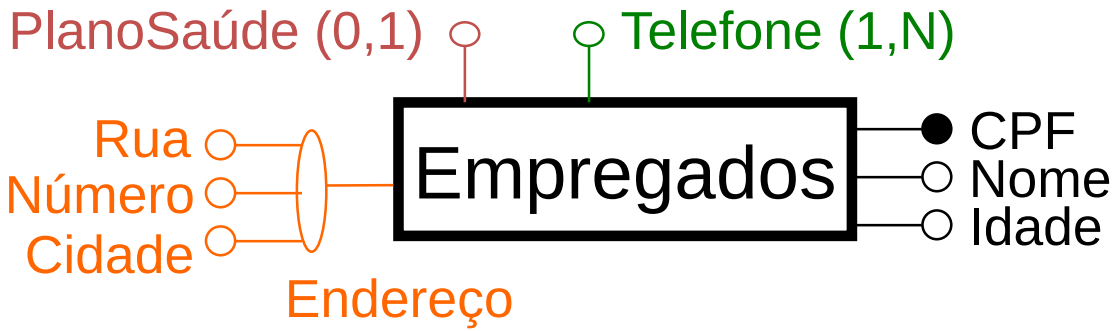
Mapeamento de Entidades Fracas

- Identificador da entidade forte torna-se
  - parte da chave primária na tabela correspondente à entidade fraca (*tabelaFraca*)
  - chave estrangeira na *tabelaFraca*



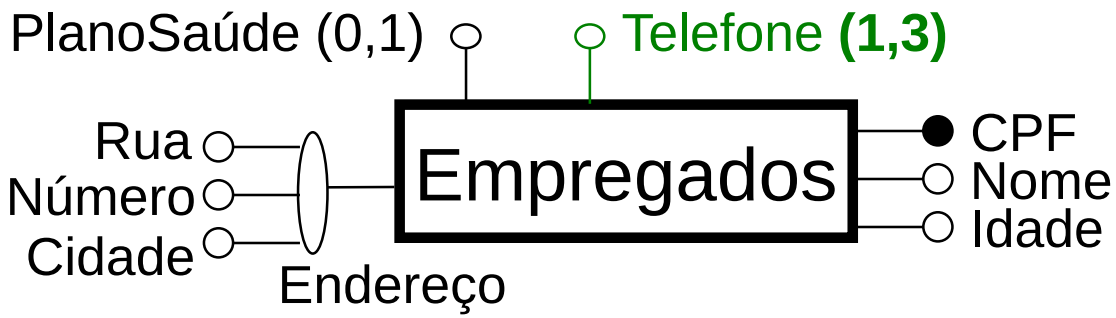
Itens (NroPedido, NroItem, Produto, Quantidade)

Mapeamento de Atributos



Empregados (<u>CPF</u>, Nome, Idade, PlanoSaúde,
Rua, Número, Cidade)
Telefone(<u>CPF</u>, <u>Número</u>)
ou
Telefone (<u>CPF</u>, <u>Número</u>)

Mapeamento de Atributos



Empregados (<u>CPF</u>, Nome, Idade, PlanoSaúde,
Rua, Número, Cidade,
FoneRes, FoneCom, Celular)

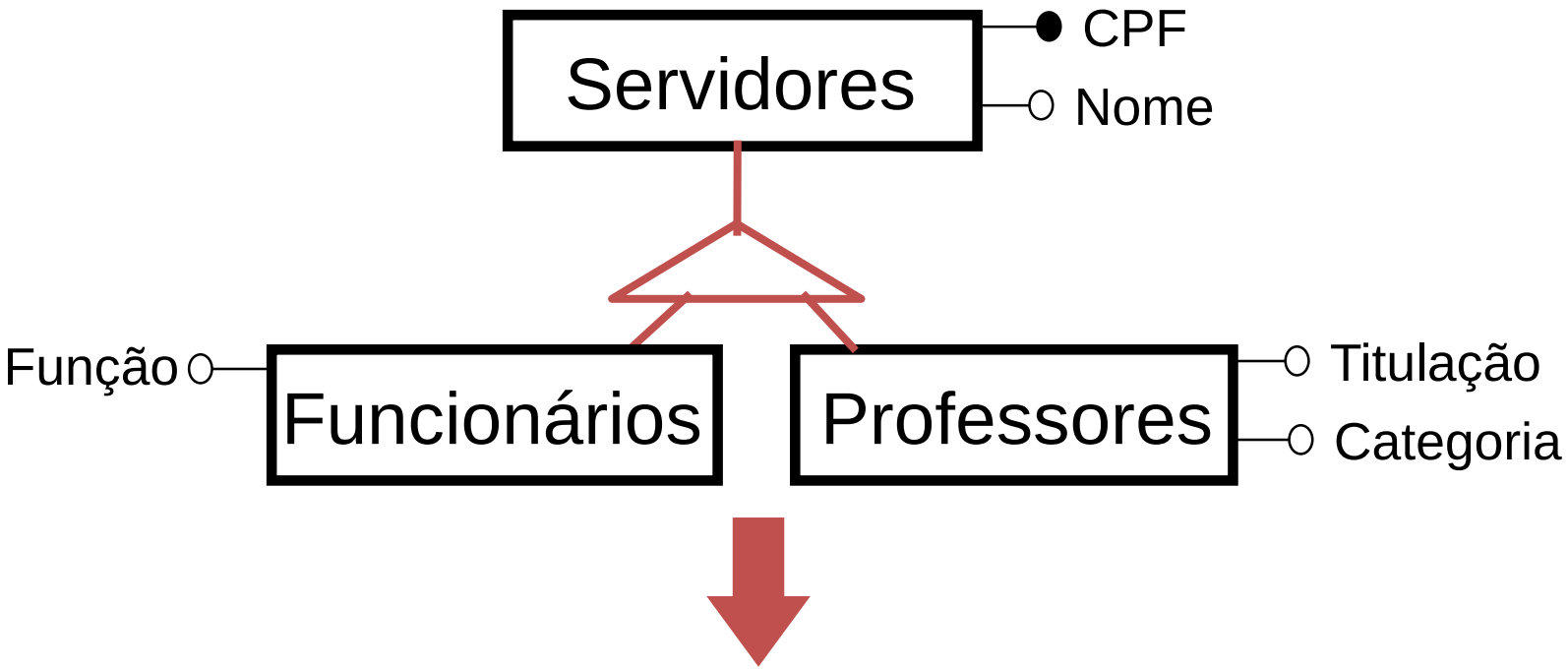
Processo de Mapeamento

1. Mapeamento preliminar de entidades e seus atributos
2. Mapeamento de especializações
3. Mapeamento de relacionamentos e seus atributos

Mapeamento de Especializações

- Três alternativas são geralmente adotadas
  1. tabela única para entidade genérica e suas especializações
  2. tabelas para a entidade genérica e as entidades especializadas
  3. tabelas apenas para as entidades especializadas

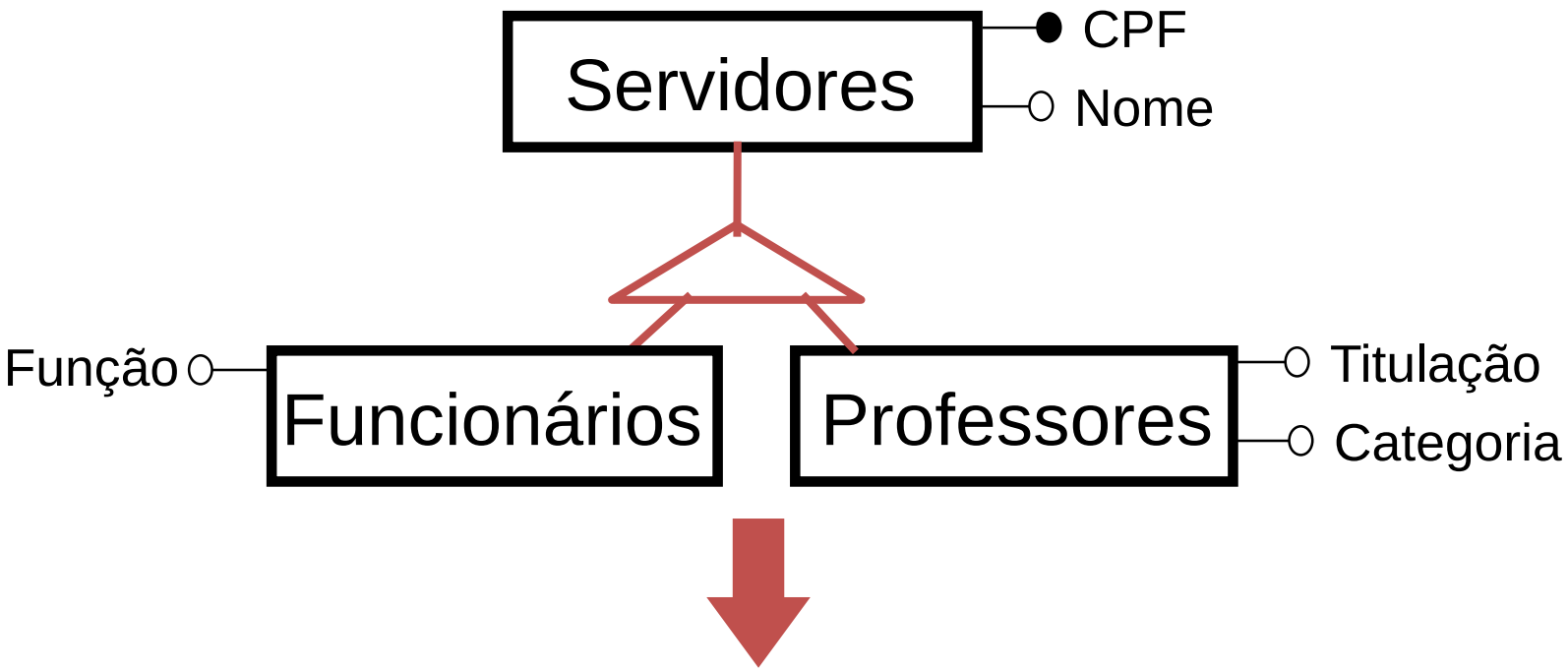
Alternativa 1



Servidores (CPF, Nome, Tipo, Função, Titulação, Categoria)

- Tipo pode assumir mais de um valor se a especialização é não-exclusiva

Alternativa 2

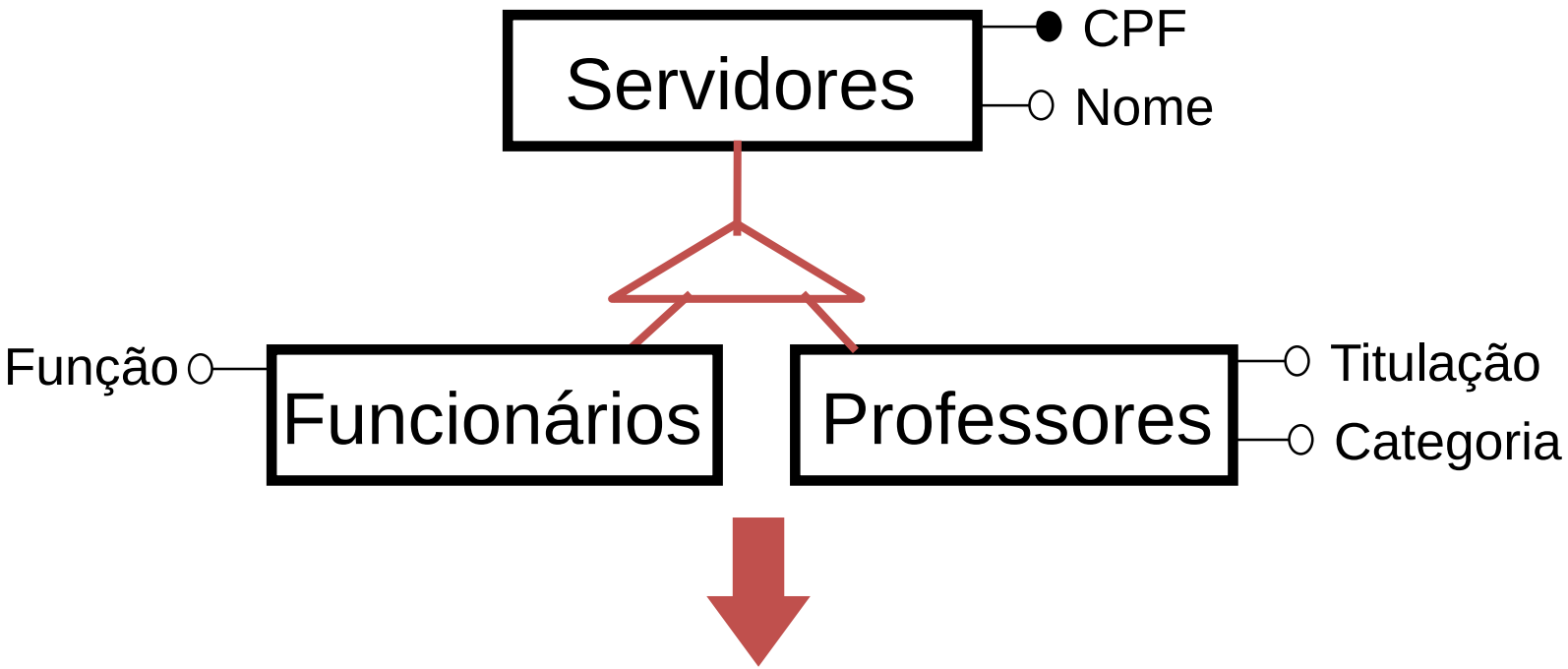


Servidores (<u>CPF</u>, Nome)

Funcionários (<u>CPF</u>, Função)

Professores (<u>CPF</u>, Titulação, Categoria)

Alternativa 3



Funcionários (CPF, Nome, Função)

Professores (CPF, Nome, Titulação, Categoria)

- Não se aplica a especializações parciais

Processo de Mapeamento

1. Mapeamento preliminar de entidades e seus atributos
2. Mapeamento de especializações
3. Mapeamento de relacionamentos e seus atributos

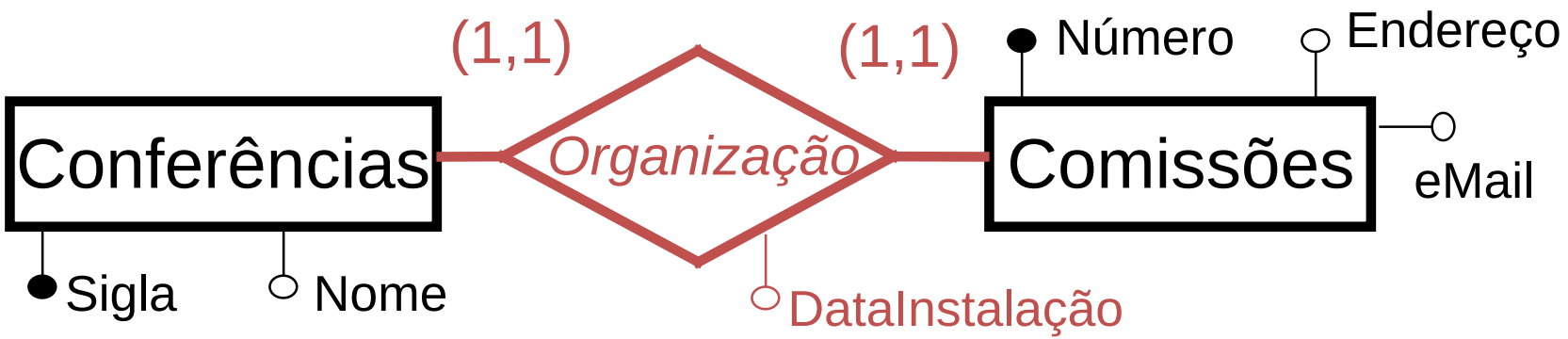
Mapeamento de Relacionamentos

GRADUAÇÃO
UniCesumar

- Recomendações de mapeamento baseiam-se na análise da cardinalidade dos relacionamentos
  - com base nesta análise, algumas alternativas de mapeamento podem ser adotadas
    1. entidades relacionadas podem ser fundidas em uma única tabela
    2. tabelas podem ser criadas para o relacionamento
    3. chaves estrangeiras podem ser criadas em tabelas a fim de representar adequadamente o relacionamento

Relacionamento 1-1

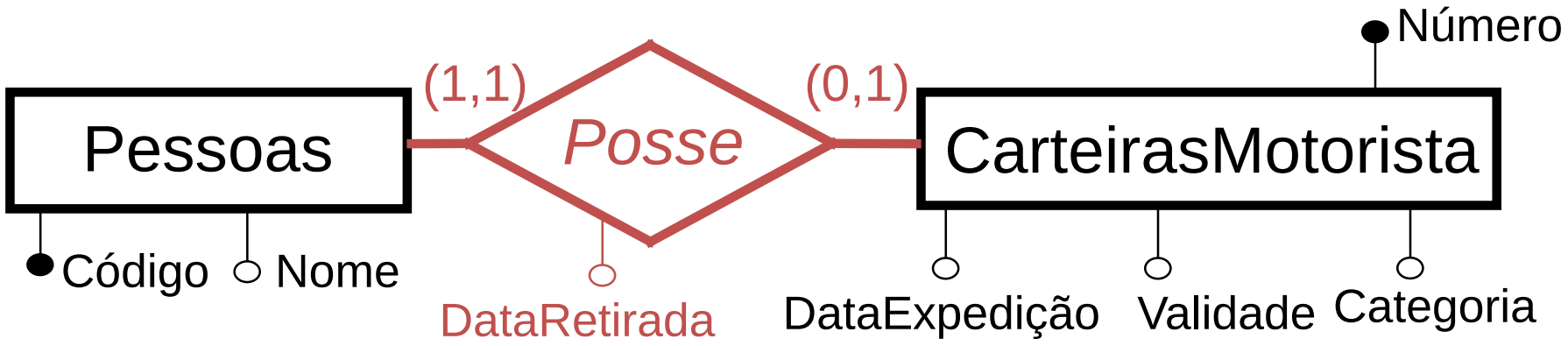
- Obrigatório em ambos os sentidos



Conferências (<u>Sigla</u>, Nome, DataInstCom, NroCom, EndereçoCom, eMailCom)

Relacionamento 1-1

- Opcional em um dos sentidos

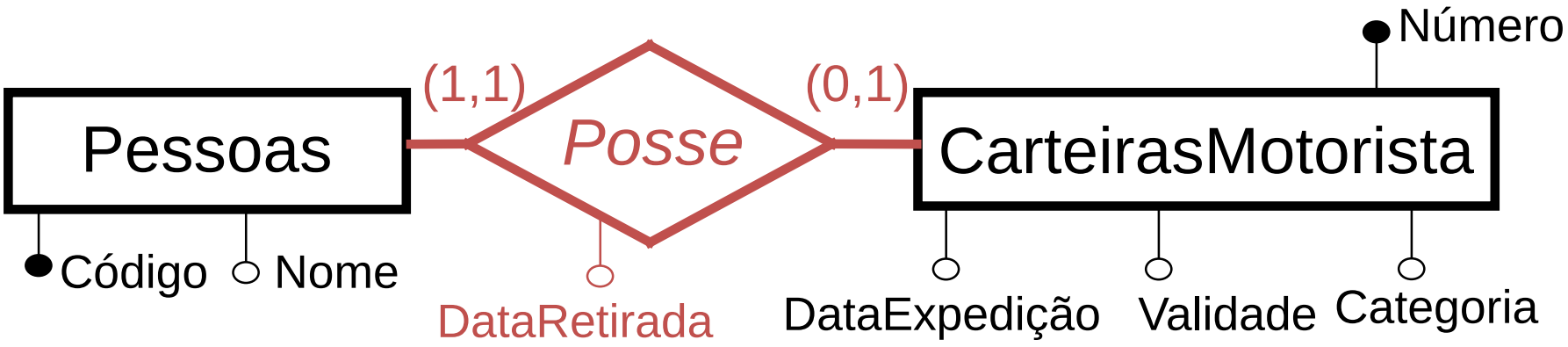


alternativa 1

Pessoas (Código, Nome, NúmeroCarteiraMotorista, DataExpedição, Validade, Categoria, DataRetirada)

Relacionamento 1-1

- Opcional em um dos sentidos



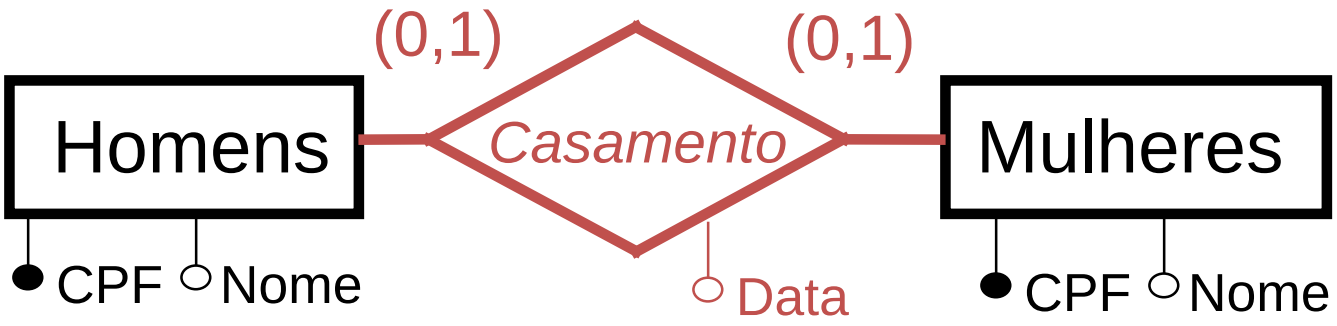
alternativa 2

Pessoas (Código, Nome)

CarteirasMotorista (Número, DataExpedição, Validade, Categoria, Código, DataRetirada)

Relacionamento 1-1

- Opcional em ambos os sentidos



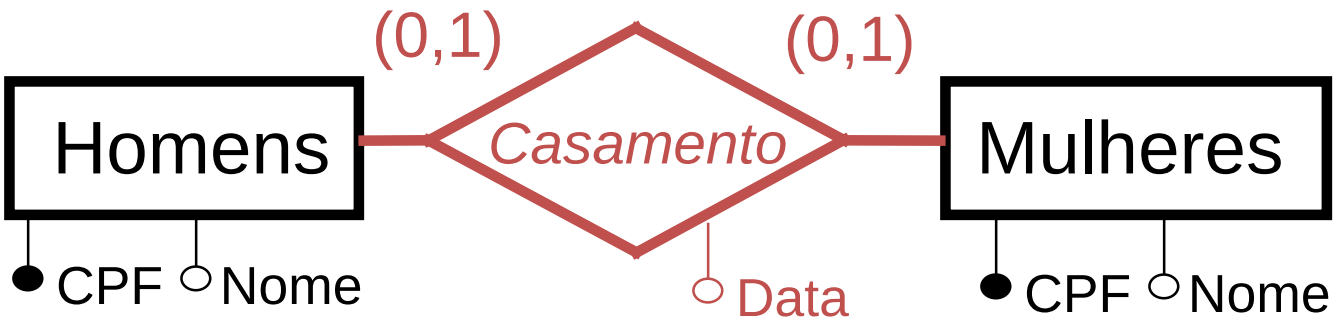
alternativa 1

Homens (<u>CPF</u>, Nome) Mulheres (<u>CPF</u>, Nome)

Casamento (<u>CPFh</u>, <u>CPFm</u>, Data)

Relacionamento 1-1

- Opcional em ambos os sentidos



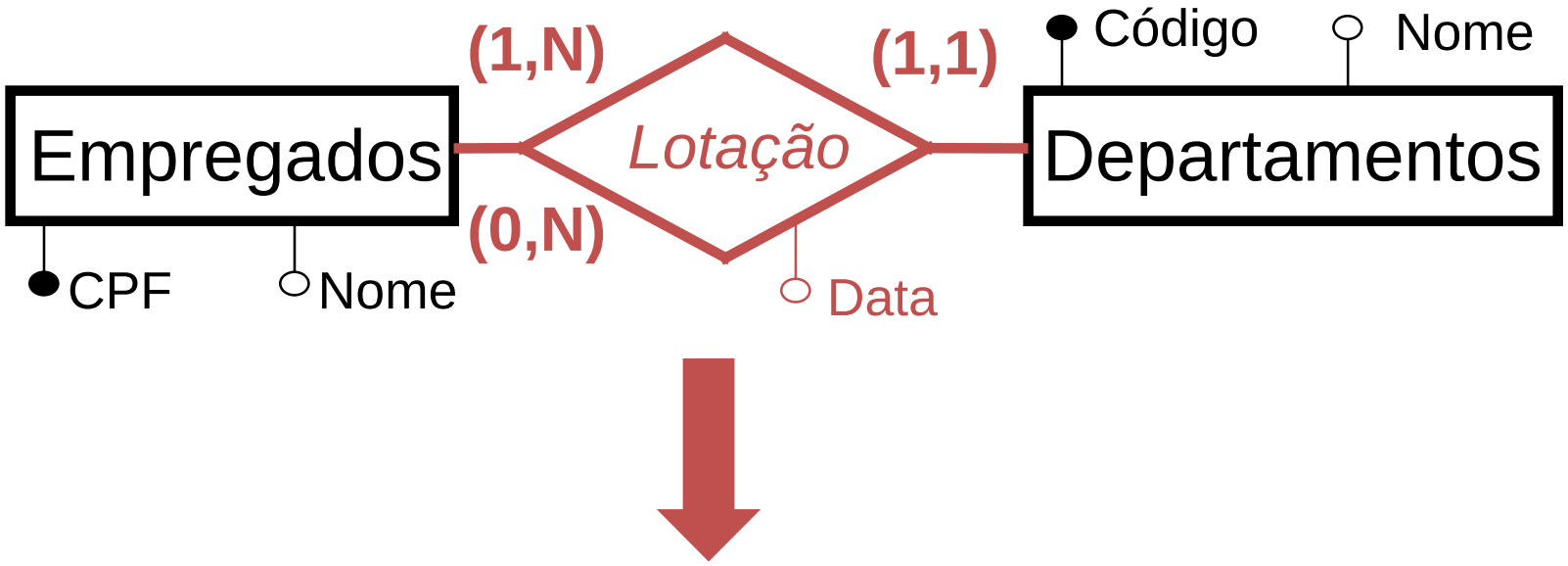
alternativa 2

Homens (<u>CPF</u>, Nome)

Mulheres (<u>CPF</u>, Nome, CPFmarido, DataCasamento)

Relacionamento 1-N

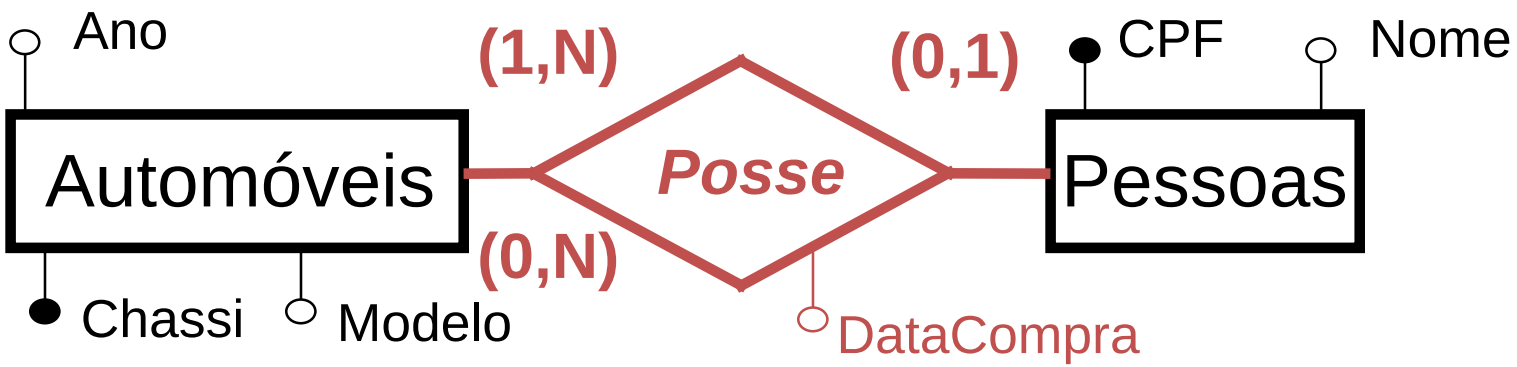
- Obrigatório/opcional no “lado N”



Departamentos (Código, Nome)
Empregados (CPF, Nome, CodDepto, DataLotação)

Relacionamento 1-N

- Opcional no “lado 1”

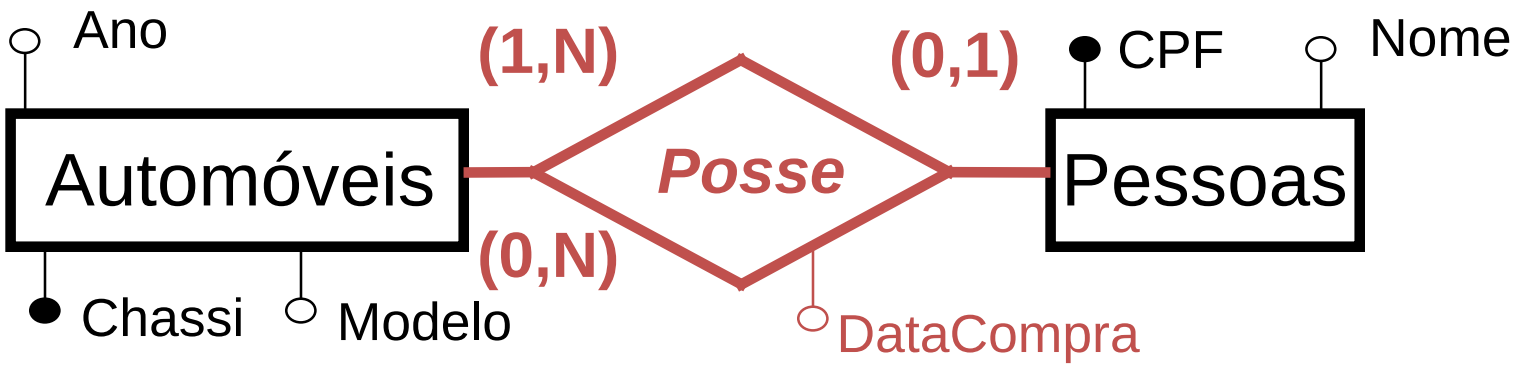


alternativa 1

Pessoas (CPF, Nome)
Automóveis (Chassi, Modelo, Ano)
Posse (CPF, Chassi, DataCompra)

Relacionamento 1-N

- Opcional no “lado 1”

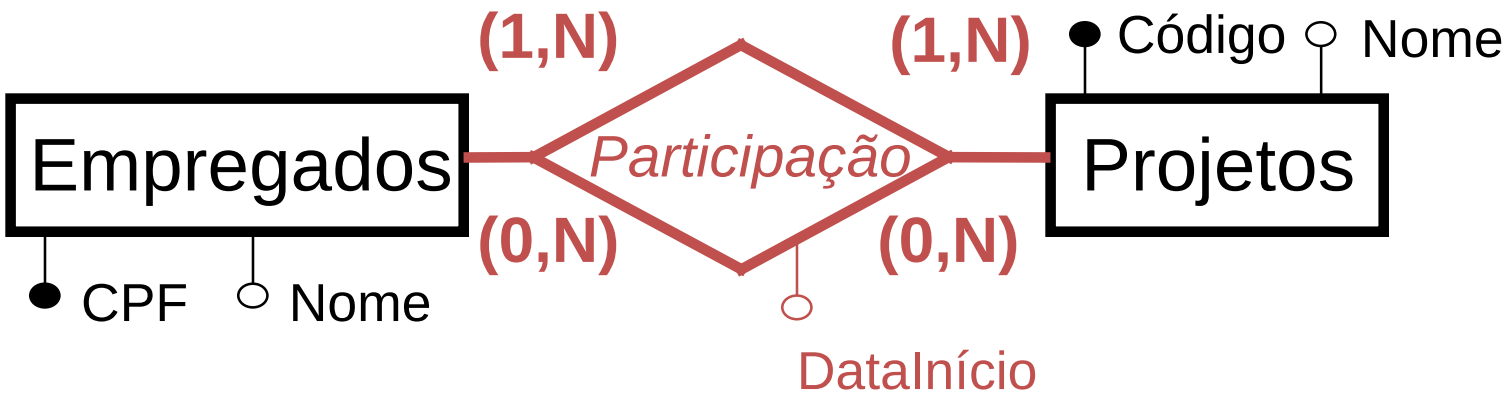


alternativa 2

Pessoas (CPF, Nome)
Automóveis (Chassi, Modelo, Ano, $\overline{\text{CPF}}$, DataCompra)

Relacionamento N-M

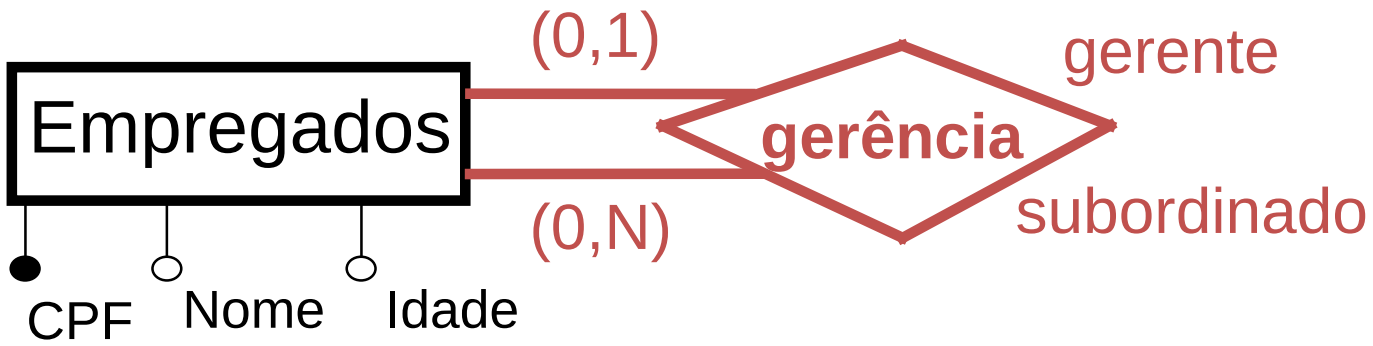
- Obrigatório/opcional em ambos os sentidos



Empregados (<u>CPF</u>, Nome)
Projetos (<u>Código</u>, Nome)
Participação (<u>CPF</u>, <u>Código</u>, DataInício)

Auto-Relacionamento

- Valem as mesmas recomendações anteriores



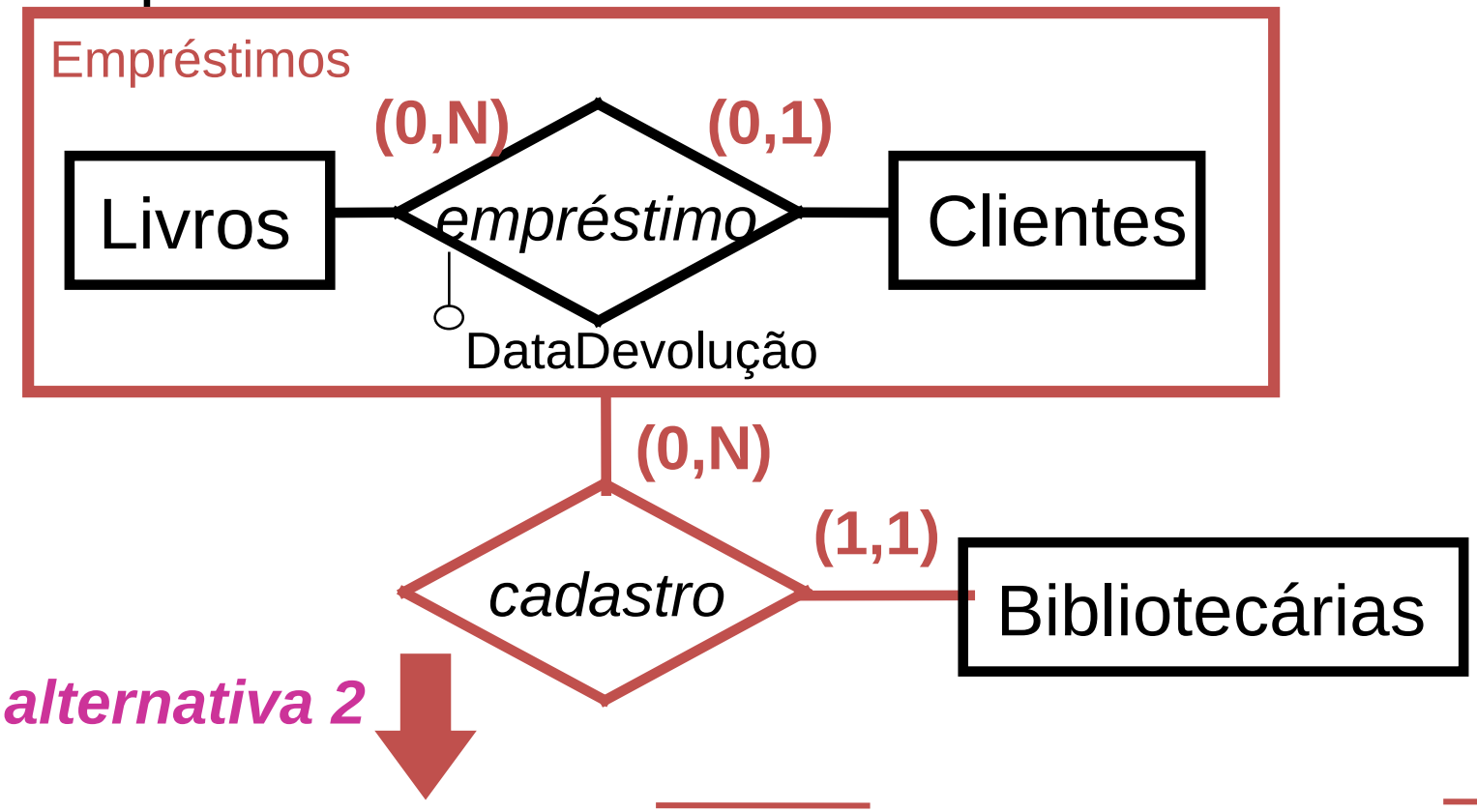
Alternativas:

1) Empregados(CPF, Nome, Idade)
Gerência(CPFe, CPFg)

2) Empregados(CPF, Nome, Idade, CPFg)

Relacionamentos com Entidades Associativas

- Valem as mesmas recomendações anteriores
  – questão: “localizar” a entidade associativa

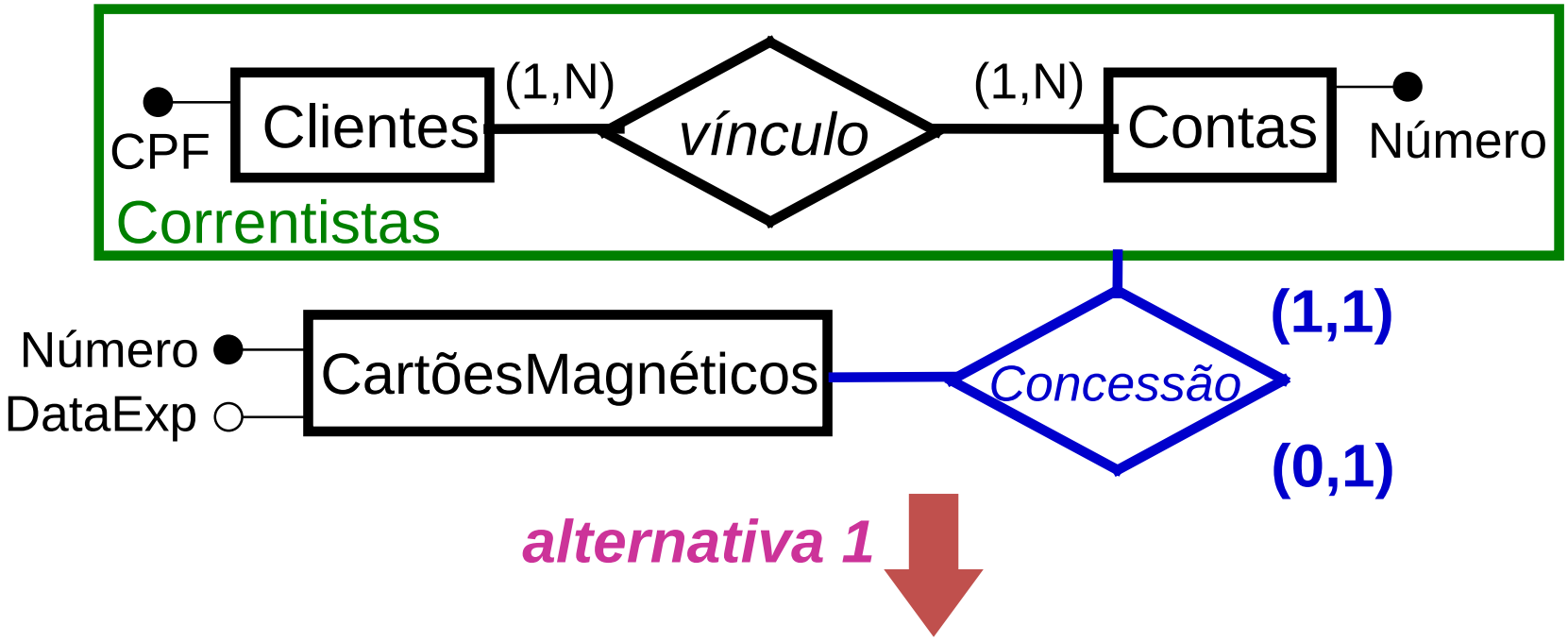


alternativa 2

Livros (Código, ..., CPFcli, DataDevolução, CPFbibl)
Clientes (CPFcli, ...)
Bibliotecárias(CPFbibl, ...)

Relacionamentos com Entidades Associativas

- Outro exemplo



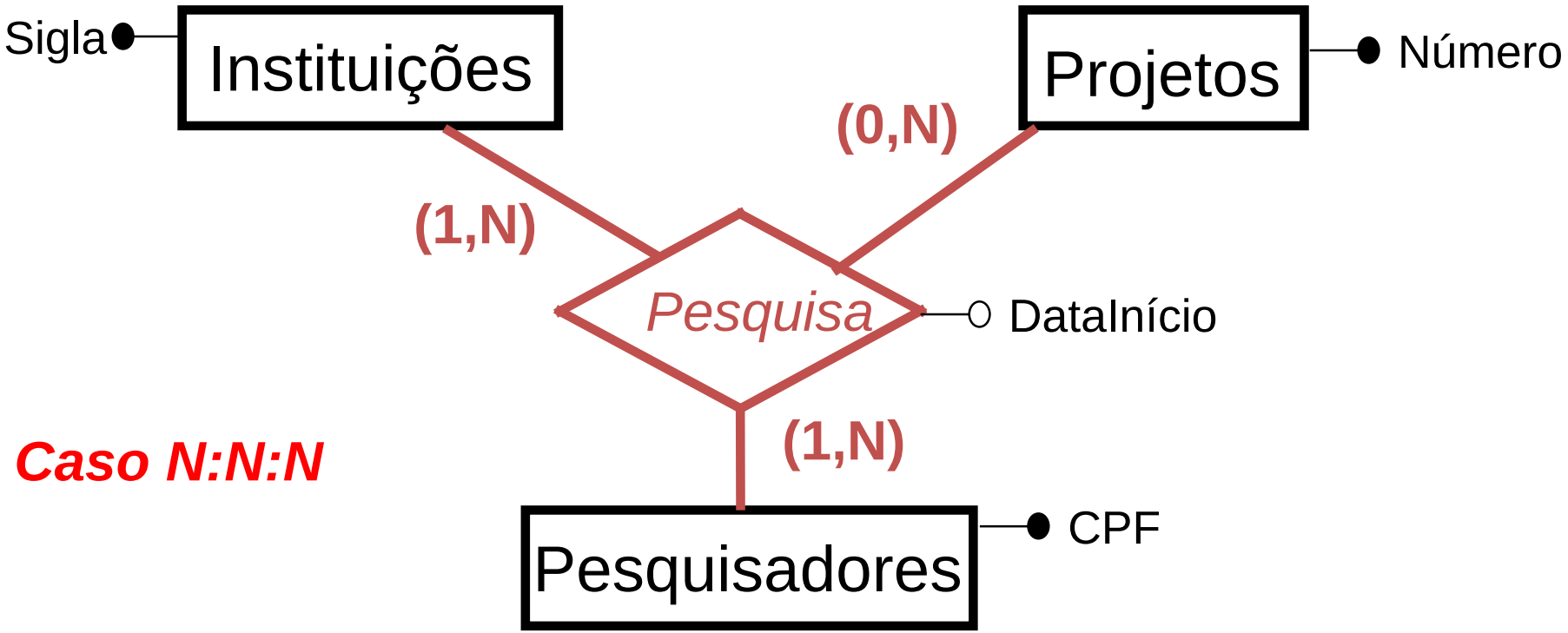
alternativa 1

Correntista(CPF, NroCta, NroCartão, DataExp)

Relacionamentos Ternários

GRADUAÇÃO UniCesumar

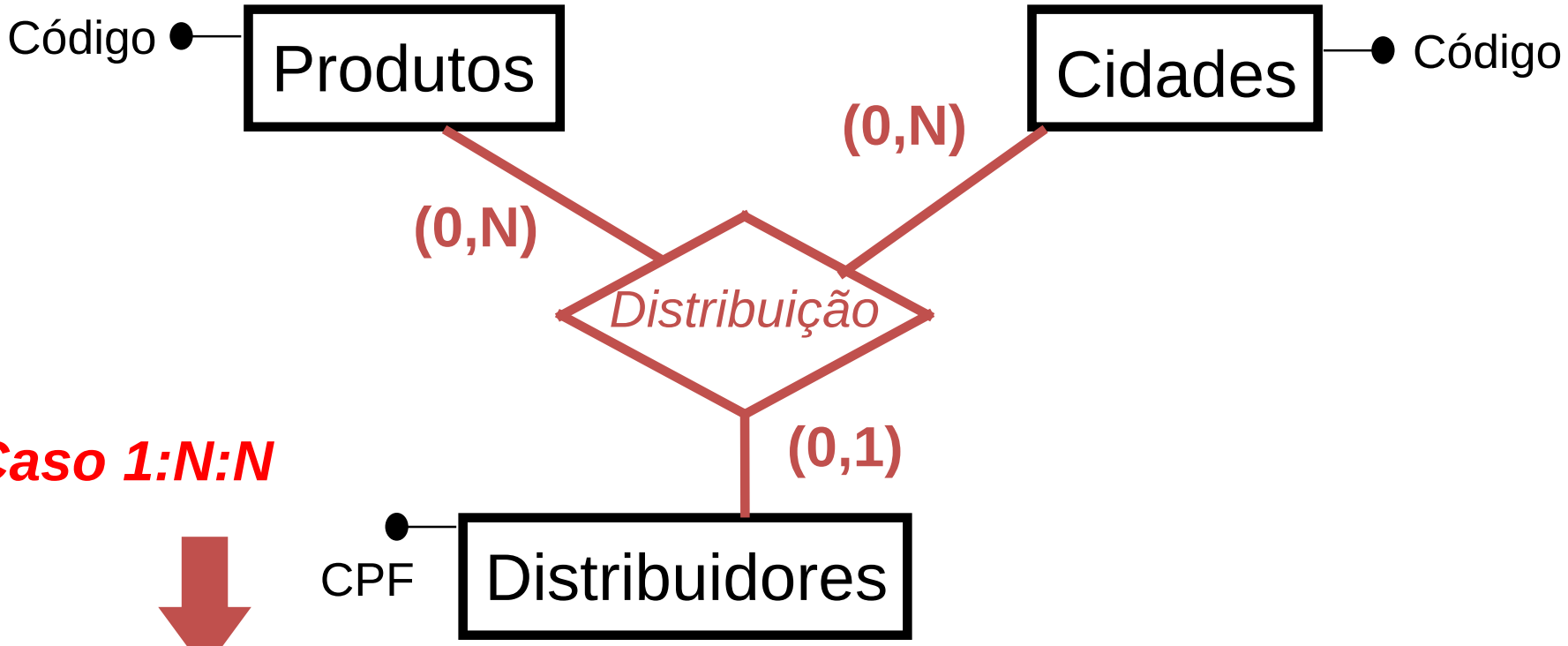
- Gera uma tabela para o relacionamento



Caso N:N:N

Instituições (<u>Sigla</u>, ...) Projetos (<u>Número</u>, ...)
Pesquisadores (<u>CPF</u>, ...)
Pesquisa (<u>Sigla</u>, <u>Número</u>, <u>CPF</u>, DataInício)

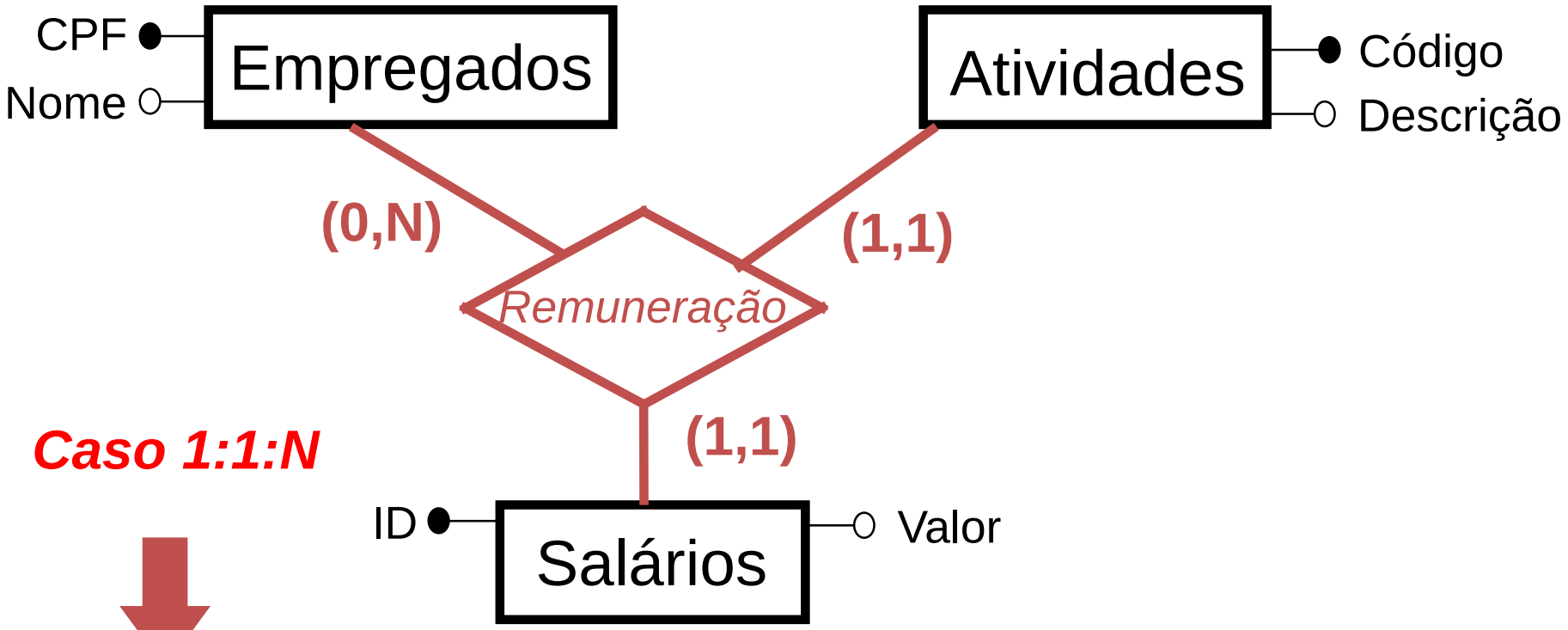
Relacionamentos Ternários



Caso 1:N:N

Produtos (<u>Código</u>, ...)
Cidades (<u>Código</u>, ...)
Distribuidores (<u>CPF</u>, ...)
Distribuição (<u>CodProduto</u>, <u>CodCidade</u>, <u>CPF</u>)

Relacionamentos Ternários

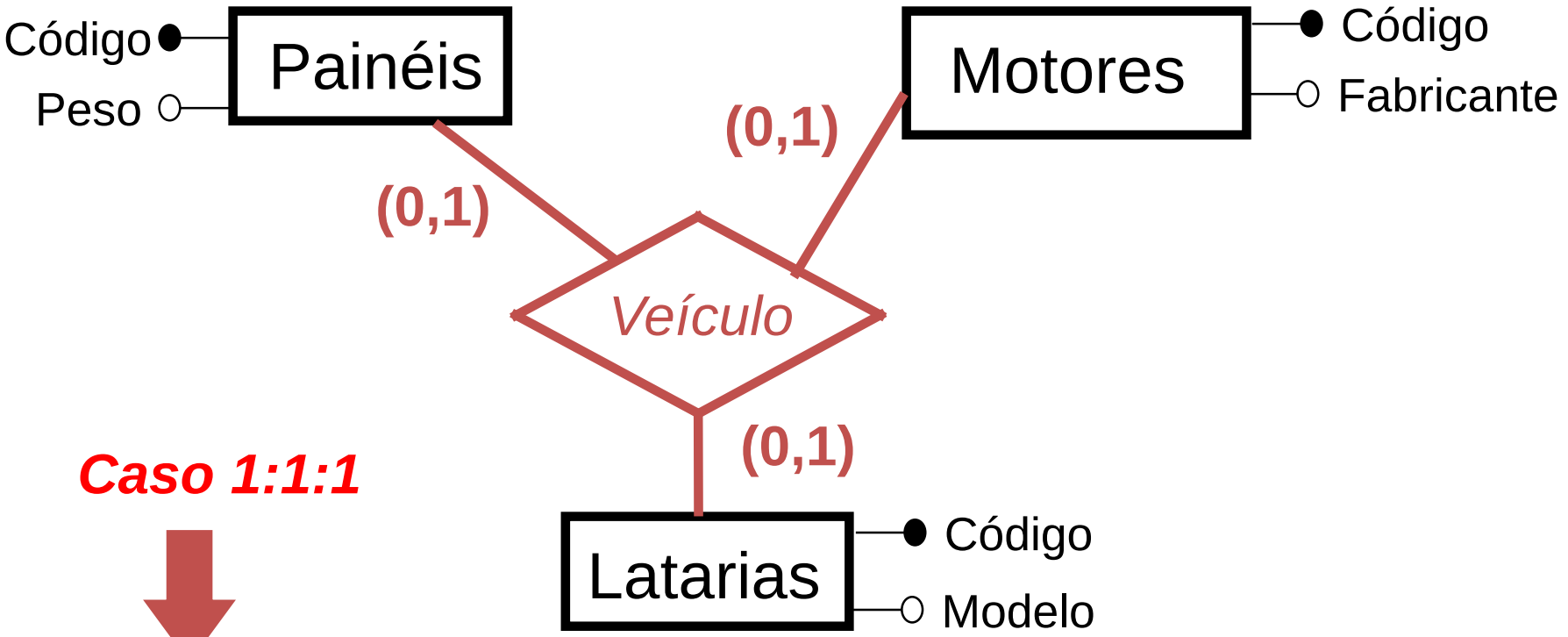


Caso 1:1:N

Empregados (CPF, Nome)
Atividades (Código, Descrição)
Salários (ID, valor)
Remuneração (CodAtiv, CPF, ID-Salario)

- Uma das RIs pode ser chave primária

Relacionamentos Ternários



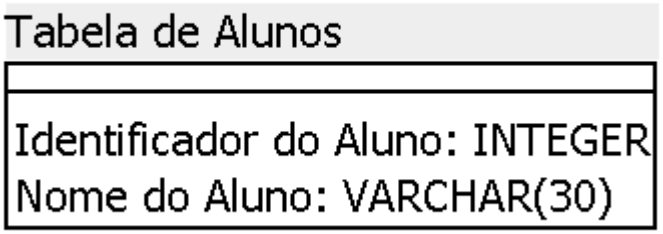
Caso 1:1:1

Painéis (<u>Código</u>, Peso)
Motores (<u>Código</u>, Fabricante)
Latarias (<u>Código</u>, Modelo)
Veículo (<u>CodP, CodM, CodL</u>)

- Uma das RIs pode ser chave primária

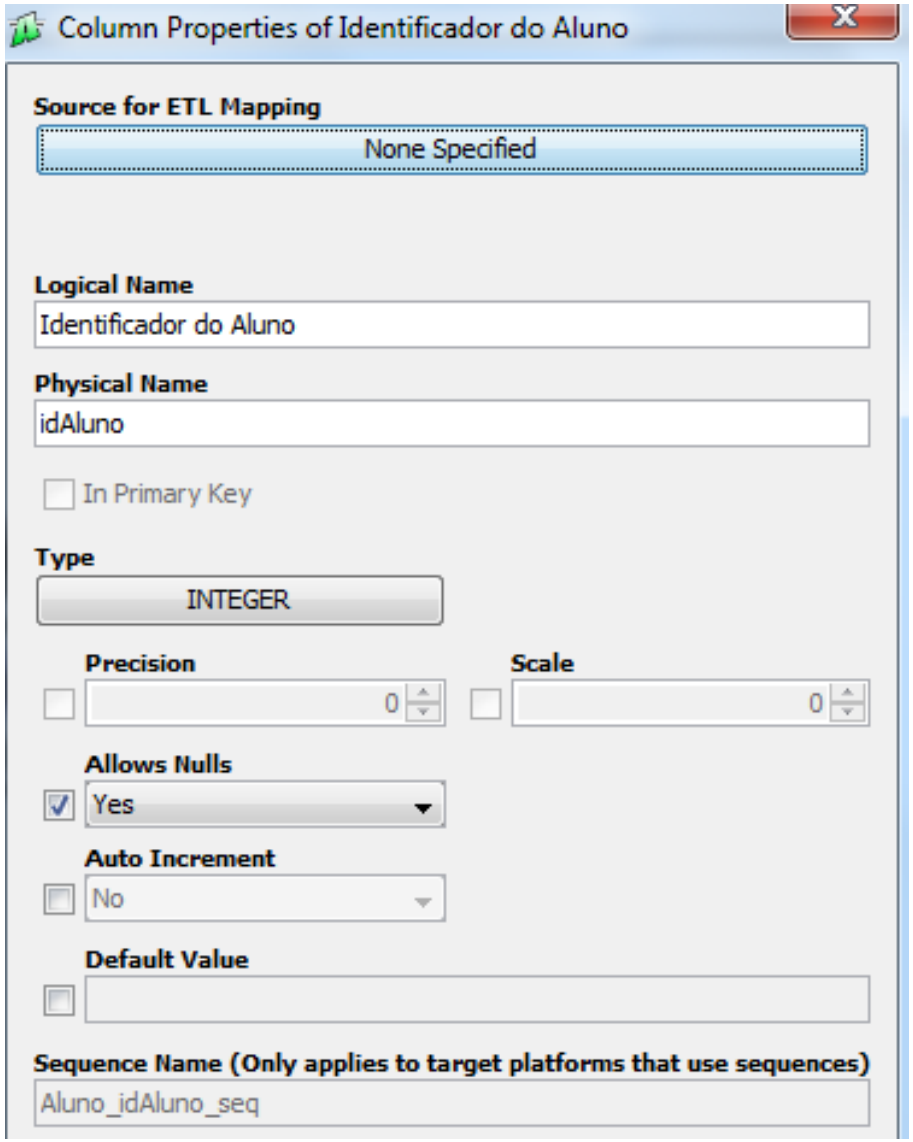
Modelo Lógico

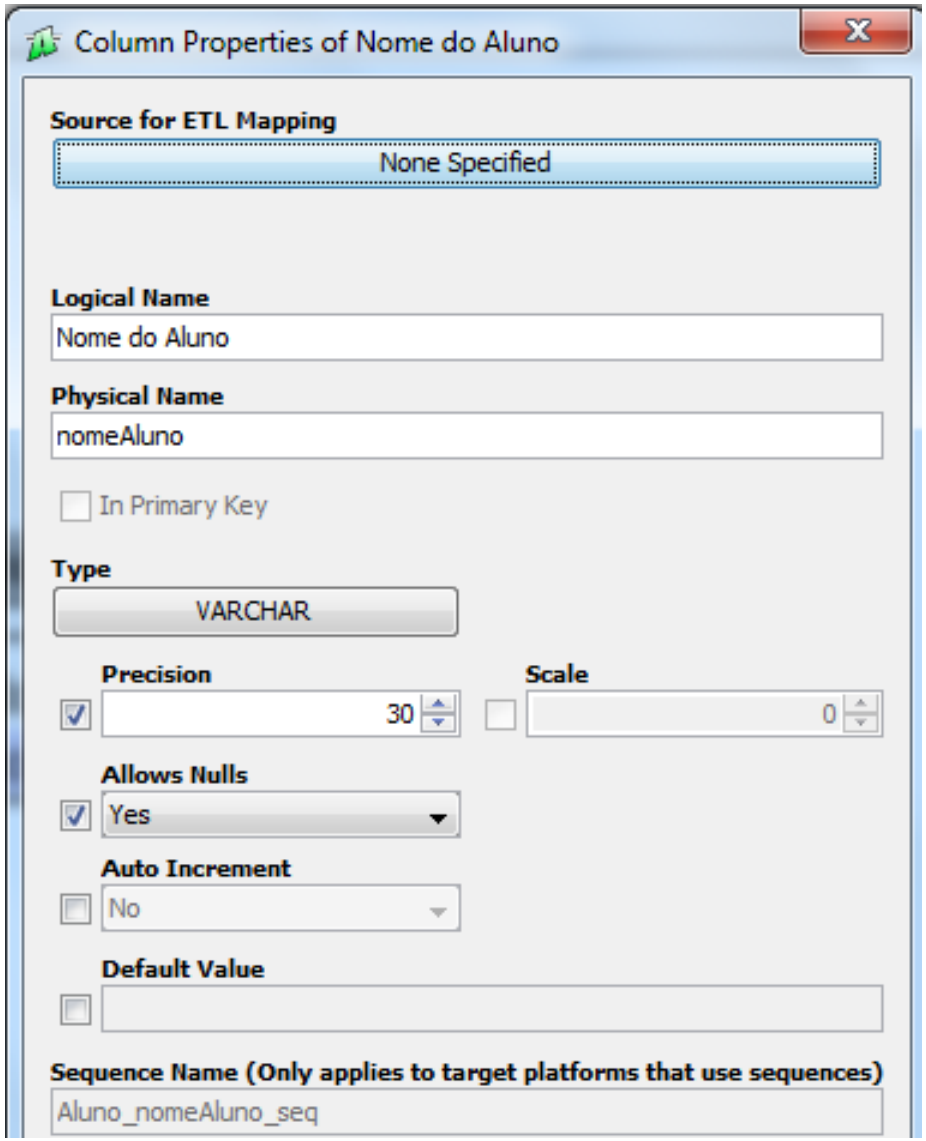
Crie no modelo lógico (Sql Power Architect), a mesma tabela Aluno, ignorando chaves primárias e todos aceitando valores nulos.



Colunas

GRADUAÇÃO UniCesumar

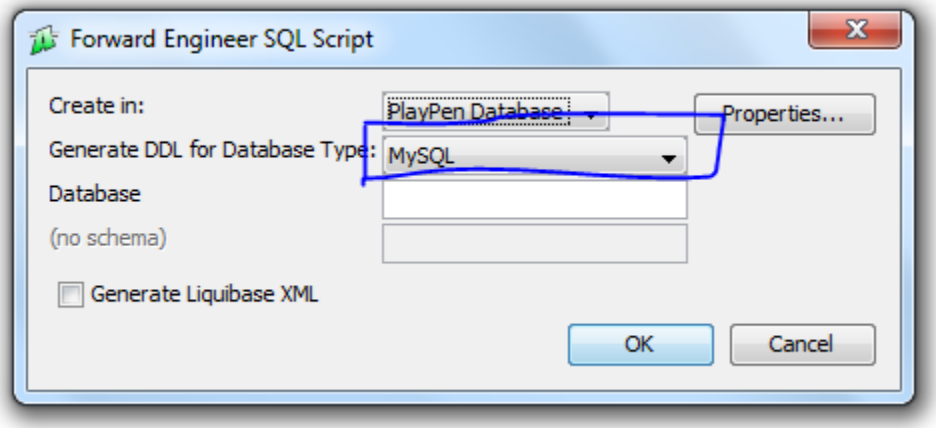




Modelo Físico

Na Ferramenta Sql Power Architect no Menu
Ferramentas -> Engenharia Reversa.

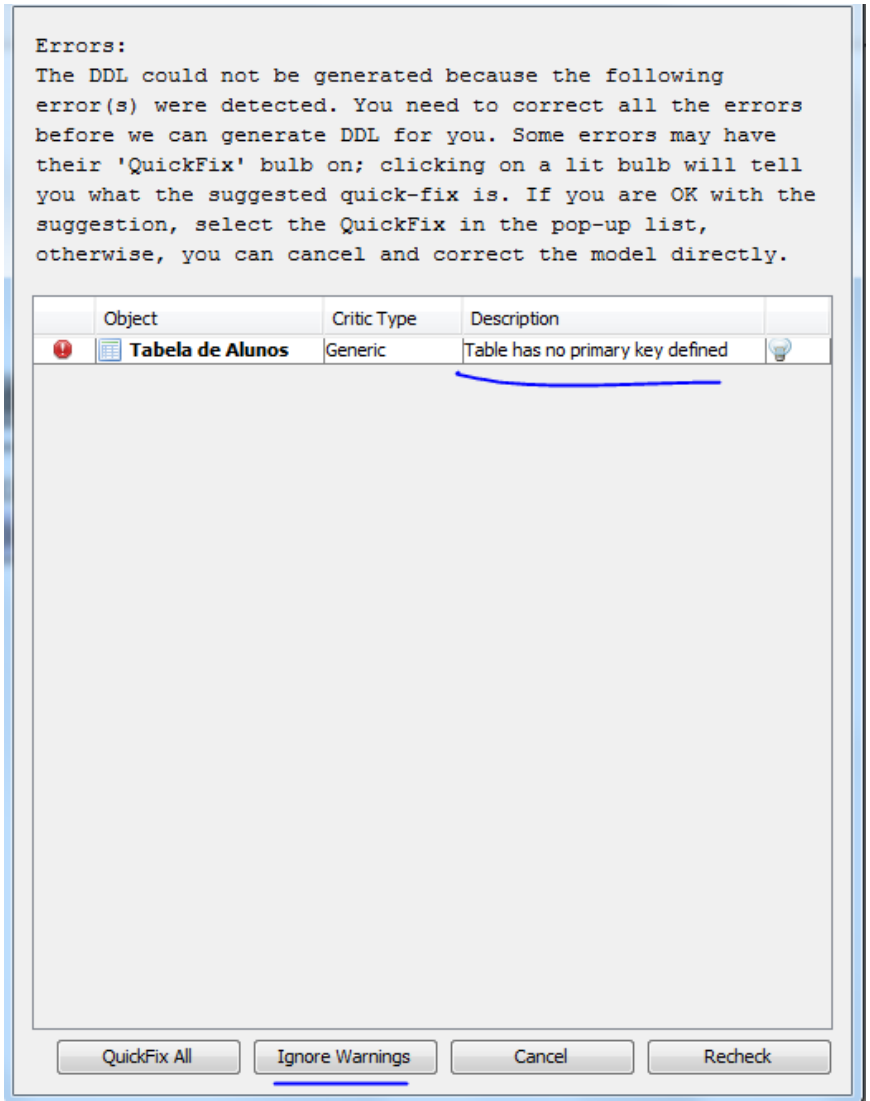
Escolha para qual banco de dados irá gerar o script para o banco físico. Na aula de hoje (Mysql), conforme figura.



Modelo Físico

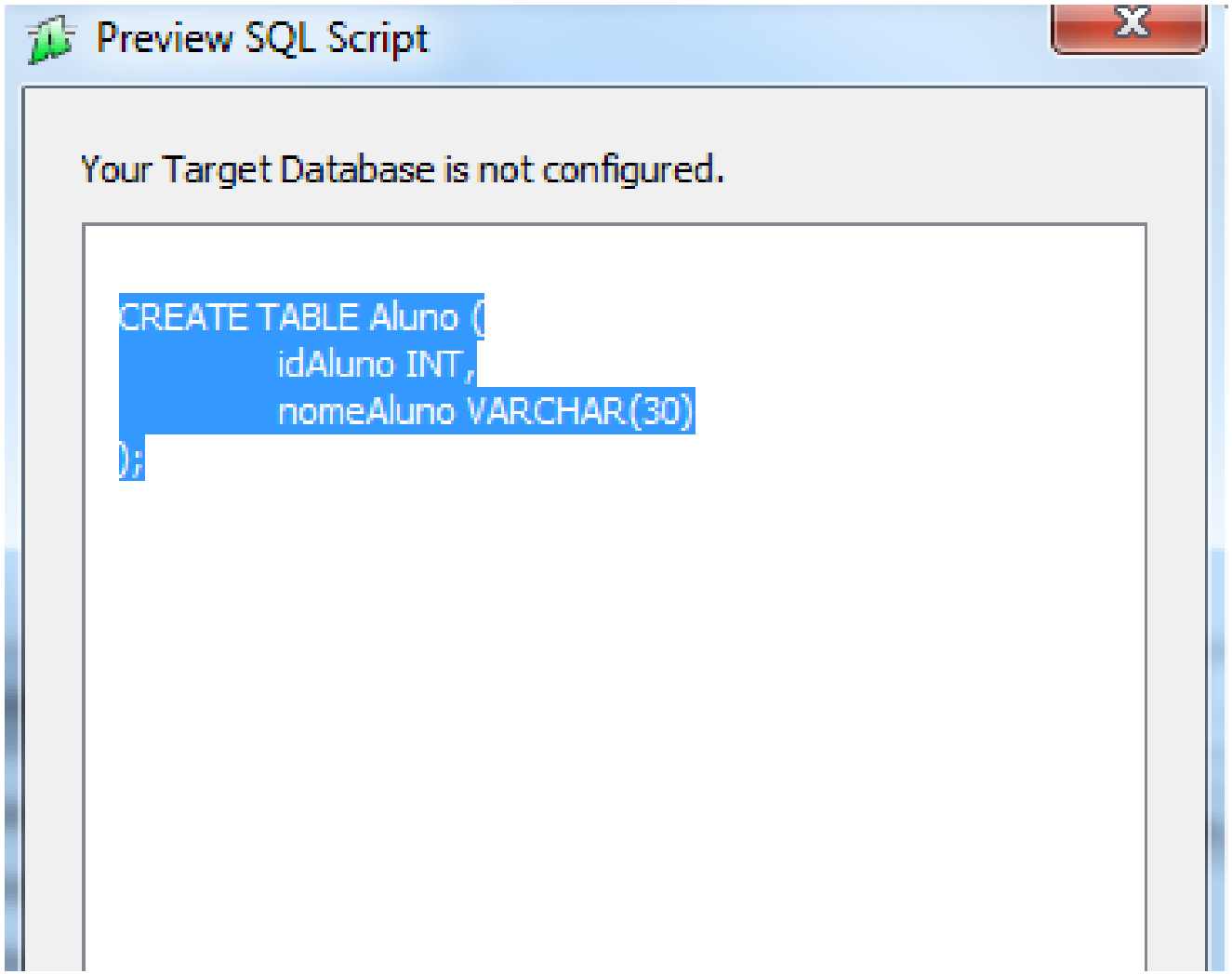
GRADUAÇÃO UniCesumar

Ao gerar o script ele sinalizará uma advertência que não temos chave primária. Pode ignorar, pois nesse exemplo, não utilizaremos.



Modelo Físico

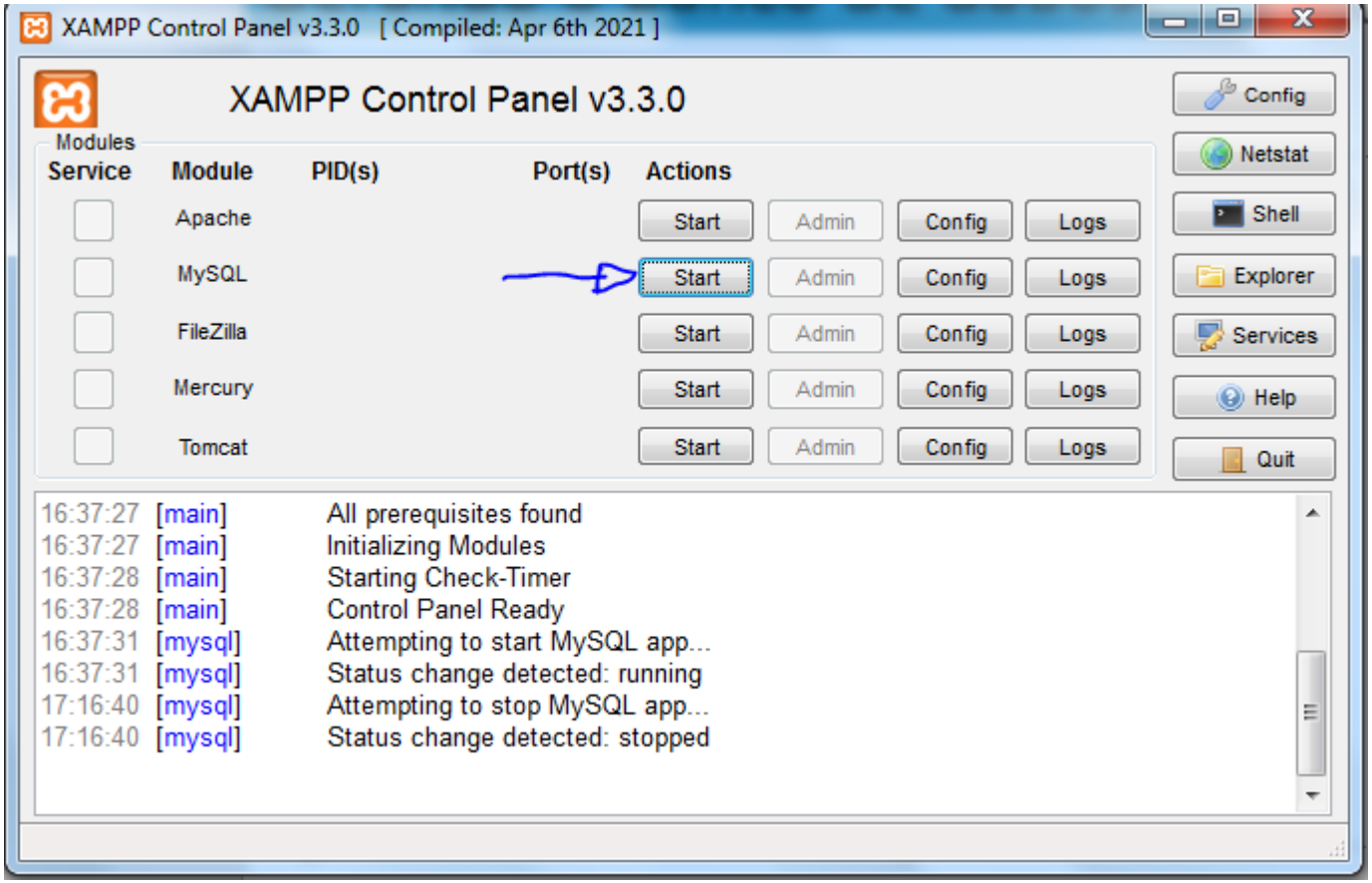
GRADUAÇÃO
UniCesumar



Gerando o Banco de dados.

GRADUAÇÃO UniCesumar

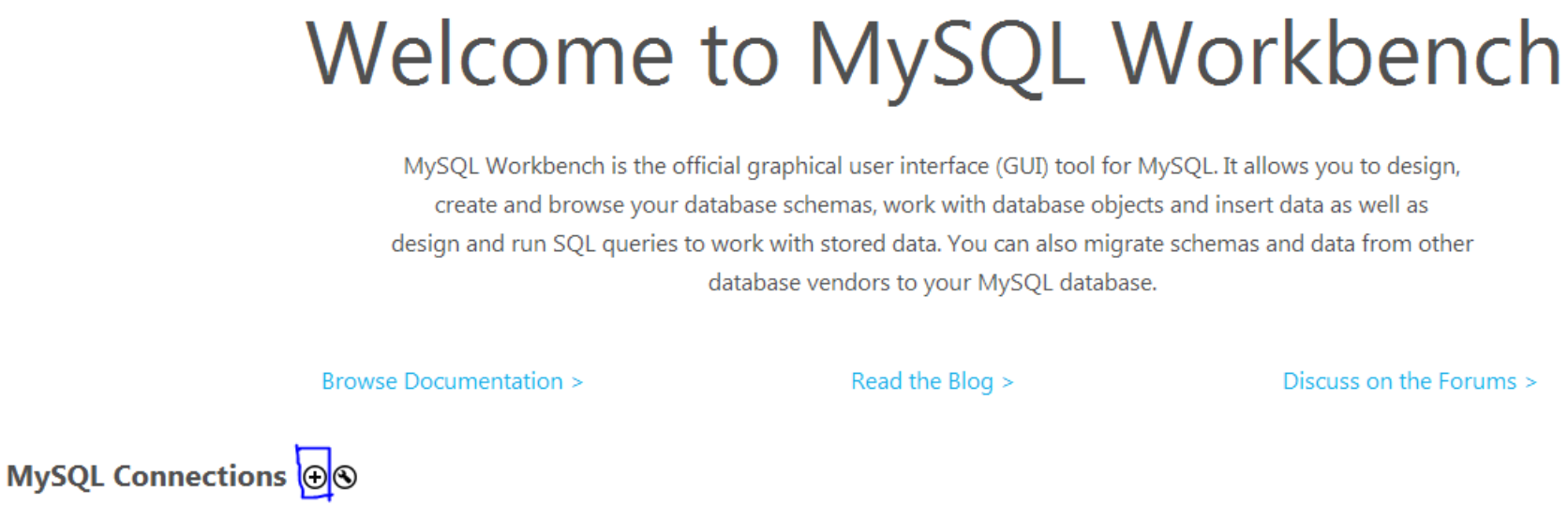
Passo 1. Abra o Xampp e start o MySql.



Gerando o Banco de dados.

GRADUAÇÃO UniCesumar

Depois de inicializar o Mysql (ele executa o arquivo mysqld.exe) pode abrir o SGBD (Sistema Gerenciador de Banco de Dados), que no nesse bimestre utilizaremos, o Mysql Workbench. Caso não tenha uma conexão, precisará criar uma selecionando o ícone abaixo:



Dependência Funcional

Dependências funcionais (DFs) são restrições de integridade mais gerais que as restrições de chave.

Exemplo de dependência funcional:

$$\{Sigla\_convênio\} \rightarrow \{Nome\_convênio, Endereço\_convênio, Telefone\_convênio\}$$

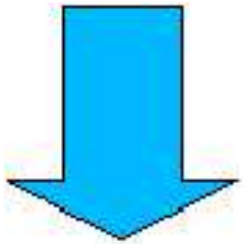
Leia-se: **Sigla_convênio** determina funcionalmente Nome_convênio, Endereço_convênio e Telefone_convênio.

Dependência Funcional

Significado: “Se duas linhas da tabela Pacientes tiverem o mesmo valor de **Sigla_convênio**, então elas tem de ter o mesmo valor de Nome_convênio, de Endereço_convênio e de Telefone_convênio.”

Exemplo

Proj (CodProj, Tipo, Descr,
(CodEmp, Nome, Cat, Sal, DataIni, TempAl))



ProjEmp (CodProj, Tipo, Descr, CodEmp, Nome, Cat,
Sal, DataIni, TempAl)

⇨ Dados do projeto aparecem repetidos para cada empregado do projeto

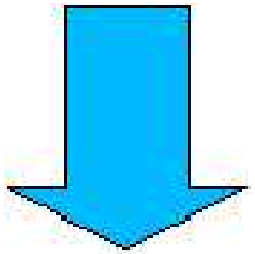
Passagem à 1aFN

uma tabela para cada tabela aninhada

- Cria-se uma tabela referente a própria tabela que está sendo normalizada e uma tabela para cada tabela aninhada

Exemplo

Proj (CodProj, Tipo, Descr,
(CodEmp, Nome, Cat, Sal, DataIni, TempAl))



Proj (CodProj, Tipo, Descr)

ProjEmp (CodProj, CodEmp, Nome, Cat, Sal, DataIni, TempAl)

Alternativas

- Primeira alternativa (tabela única) é mais correta
- Decompor uma tabela em várias tabelas (segunda alternativa)
  - podem ser perdidas relações entre informações

Alternativas

- Para fins práticos
  - preferimos a segunda alternativa
    (decomposição de tabelas)
- Quando houver diversas tabelas aninhadas, eventualmente com diversos níveis de aninhamento, fica difícil visualizar a tabela na 1FN na alternativa de tabela única

Alternativas

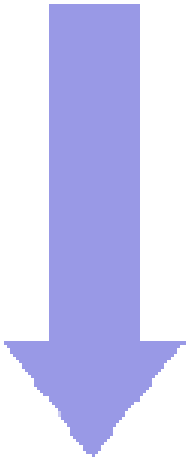
- Criar uma tabela na 1FN referente à tabela não normalizada
- A chave primária da tabela na 1FN é idêntica a chave da tabela ÑN

Passagem à 1FN

ÑN

(CodProj, Tipo, Descr,

(CodEmp, Nome, Cat, Sal, DataIni, TempAl))



1FN

(CodProj, Tipo, Descr)

(CodProj, CodEmp, Nome, Cat, Sal, DataIni, TempAl)

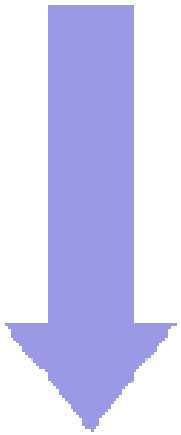
Passagem à 1FN

- Para cada tabela aninhada
  - criar uma tabela na 1FN composta pelas seguintes colunas:
    - a chave primária de cada uma das tabelas na qual a tabela em questão está aninhada
    - as colunas da própria tabela aninhada

Passagem à 1FN

ÑN

(CodProj, Tipo, Descr,
(CodEmp, Nome, Cat, Sal, DataIni, TempAl))



1FN

(CodProj, Tipo, Descr)

(CodProj, CodEmp, Nome, Cat, Sal, DataIni, TempAl)

Passagem à 1FN

ÑN

(CodProj, Tipo, Descr,

(CodEmp, Nome, Cat, Sal, DataIni, TempAI))

Tabela de nível mais externo: basta transcrever a chave

1FN

(CodProj, Tipo, Descr)

(CodProj, CodEmp, Nome, Cat, Sal, DataIni, TempAI)

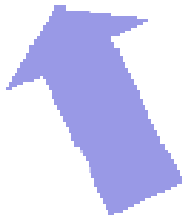
Definição de chave primária

1FN

(CodProj, Tipo, Descr)

(CodProj, CodEmp, Nome, Cat,

Sal, DataIni, TempAl)



Qual é a chave primária desta tabela?
Pergunta a fazer:

"um valor de CodEmp (chave da tabela origem) aparece uma vez só no documento, ou várias?"

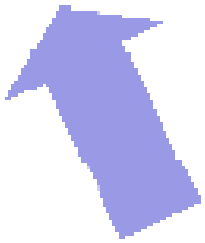
Definição de chave primária

1FN

(CodProj, Tipo, Descr)

(CodProj, CodEmp, Nome, Cat,

Sal, DataIni, TempAl)



Como um valor de CodEmp aparece várias vezes, é necessário CodProj para distinguir as várias aparições

Definição de chave primária

Proj:

CodProj	Tipo	Descr
LSC001	Novo Desenv.	Sistema de Estoque
PAG02	Manutenção	Sistema de RH

ProjEmp:

CodProj	CodEmp	Nome	Cat	Sal	DataIni	TempAl
LSC001	2146	João	A1	4	1/11/91	24
LSC001	3145	Sílvio	A2	4	2/10/91	24
LSC001	6126	José	B1	9	3/10/92	18
LSC001	1214	Carlos	A2	4	4/10/92	18
LSC001	8191	Mário	A1	4	1/11/92	12
PAG02	8191	Mário	A1	4	1/05/93	12
PAG02	4112	João	A2	4	4/01/91	24
PAG02	6126	José	B1	9	1/11/92	12

Dependência Funcional

Identifique as dependências funcionais, crie as tabelas e inclua no Mysql:

Cod_Emp	Nome	Cargo	Salario
001	João da Silva	Analista Júnior	1.000,00
003	Maria João	Programador Pleno	1.200,00
006	Joice Casa	Analista Júnior	1.000,00
009	Caio Carvalho	Programador Pleno	1.200,00
002	Carlos Villa	Analista Sênior	2.200,00

Dependência Funcional

Identifique as dependências funcionais, crie as tabelas e inclua no Mysql:

Num_NF	Cod_Produto	Descricao	Preco	Data	Quantidade
001	PC	Computador	1.500,00	15/04/1998	02
001	IMP	Impressora	500,00	15/04/1998	01
002	PC	Computador	1.500,00	16/04/1998	01
003	PC	Computador	1.500,00	14/04/1998	03

Dependências funcionais

Num_NF → Data

Cod_Produto → Descricao,Preco

Num_NF,Cod_Produto → Quantidade

Dependência Funcional

Uma restrição de chave é um caso especial de DF: a chave determina funcionalmente todos os outros atributos da tabela.

Como Id é chave da tabela Pacientes, temos que:

{Id} → {Nome, Endereço, Telefone, Sexo, Data_nascimento, Sigla_convênio, Nome_convênio, Endereço_convênio, Telefone_convênio}

Dependência Funcional

Certas DFs causam redundância!

Por exemplo → Para cada associado de um convênio, os dados do convênio são repetidos na tabela Pacientes.

A causa desse problema é a DF

{Sigla_convênio} → {Nome_convênio, Endereço_convênio, Telefone_convênio}

Projeto de Banco de Dados

O objetivo do projeto de um BD relacional

- Gerar um conjunto de esquemas de relações que permitam armazenar informações sem redundância desnecessária
- Recuperar informações facilmente

Normalização

Se a normalização é bem sucedida:

- O espaço de armazenamento dos dados diminui
- A tabela pode ser atualizada com maior eficiência
- A descrição do BD será imediata

Normalização

A finalidade das regras de normalização é evitar anomalias de atualização no banco de dados

- Anomalias de inserção
  - Evitar a repetição desnecessária de dados (redundância)
- Anomalias de alteração
  - Evitar inconsistências e reduzir o esforço para a atualização dos dados
- Anomalias de exclusão
  - Evitar a perda de informações associadas a um dado registro

Normalização

Considere uma única tabela Vendas para representar as informações sobre os negócios de uma loja de CDs:

NOME_CLIENTE	COD_CD	MUSICA	CANTOR	PRECO	DATA_COMPRA
Alice Nóbrega	215621	Bem que se quis	Marisa Monte	R$ 20,00	21/03/2003
...	...	...	...	...	...
Juliano Moreira	878650	Corcovado	Tom Jobim	R$ 25,00	10/06/2003

Normalização

Caso fosse preciso registrar a compra de 5 CDs iguais para um mesmo cliente, as seguintes anomalias seriam observadas:

- Anomalia de inserção
  - Redundância em todas as colunas (5 linhas iguais na tabela)
- Anomalia de alteração
  - A mudança no preço do CD deveria ser feita em todas as linhas correspondentes da tabela
- Anomalia de exclusão
  - Só haveria registro dos CDs que fossem comprados; se a única venda de um CD fosse apagada, não haveria mais informações sobre aquele CD

Primeira Forma Normal (1FN)

Conceito: Uma variável de relação (tabela) está em 1FN se, e somente se, em todo valor válido dessa variável de relação, cada tupla contém exatamente um valor para cada atributo

Os atributos devem ser atômicos (indivisíveis)

Atributos compostos ou multivalorados devem ser representados por novas linhas ou novas tabelas

Primeira Forma Normal (1FN)

Exemplo: Tabela Controle de Faltas numa Escola

A tabela abaixo não está na 1 FN

COD_TURMA	**ALUNO**	**PROFESSOR**	**SALA**	**CAPACIDADE**	**QTE_FALTAS**
BD1032	Alice Luna Juliano Camargo Márcio Andrade	Bruno Pereira	101	50	02 00 04
...	...	...	...		...

Os atributos Aluno e Qte_Faltas não são atômicos (há mais de um valor para cada registro)

Primeira Forma Normal (1FN)

Passos para obtenção da 1FN em uma tabela

- Identificar a chave primária da tabela
- Identificar os atributos compostos ou multivalorados
- Incluir uma coluna/linha para cada atributo composto/multivalorado

Primeira Forma Normal (1FN)

A tabela abaixo está na 1FN (atributos atômicos)

COD_TURMA	**ALUNO**	**PROFESSOR**	**SALA**	**CAPACIDADE**	**QTE_FALTAS**
BD1032	Alice Luna	Bruno Pereira	101	50	02
BD1032	Juliano Camargo	Bruno Pereira	101	50	00
BD1032	Márcio Andrade	Bruno Pereira	101	50	04
...	...	...	...	...	...

O próximo passo é observar se ela está também na 2FN

Segunda Forma Normal (2FN)

Conceito 1: uma variável de relação está em 2FN se, e somente se, ela está em 1FN e todo atributo não-chave é irredutivelmente dependente da chave primária

Conceito 2: uma variável de relação está em 2FN se, e somente se, ela está em 1FN e, para tabelas com chave primária composta, cada coluna não-chave depende de toda a chave, e não de apenas uma parte dela

Dica: tabelas em 1FN e com Chave Primária simples estão automaticamente em 2FN

Segunda Forma Normal (2FN)

A tabela abaixo está na 1FN mas não está na 2FN

COD_TURMA	ALUNO	PROFESSOR	SALA	CAPACIDADE	QTE_FALTAS
BD1032	Alice Luna	Bruno Pereira	101	50	02
BD1032	Juliano Camargo	Bruno Pereira	101	50	00
BD1032	Márcio Andrade	Bruno Pereira	101	50	04
...	...	...	...	...	...

Os atributos Professor, Sala e Capacidade dependem apenas de Cod_Turma (repetição para todos os alunos da turma)

Segunda Forma Normal (2FN)

Passos para obtenção da 2FN em uma tabela

- Deixá-la em 1FN
- Identificar os atributos que não fazem parte da chave primária da tabela
- Para cada um desses atributos, analisar se seu valor é determinado por parte ou pela totalidade da chave
- Criar novas tabelas para os atributos parcialmente dependentes, incluindo a parte da chave correspondente, e retirá-los da tabela original

Segunda Forma Normal (2FN)

As tabelas abaixo estão em 2FN

COD_TURMA	ALUNO	QTE_FALTAS
BD1032	Alice Luna	02
BD1032	Juliano Camargo	00
BD1032	Márcio Andrade	04
...	...	...

COD_TURMA	PROFESSOR	SALA	CAPACIDADE
BD1032	Bruno Pereira	101	50
LG1512	Marina Lucena	101	50
JV8796	Ana Barbosa	101	50
...	...	...	...

Terceira Forma Normal (3FN)

Conceito 1: uma variável de relação está em 3FN se, e somente se, ela está em 2FN e todo atributo não-chave é dependente de forma não transitiva da chave primária

Conceito 2: uma variável de relação está em 3FN se, e somente se, ela está em 2FN e todo atributo não-chave depende apenas da chave, e não de outros atributos não-chave

Dica: tabelas em 2FN e com nenhum ou um atributo além da chave estão automaticamente em 3FN

Terceira Forma Normal (3FN)

A tabela abaixo está em 2FN, mas não está em 3FN

COD_TURMA	PROFESSOR	SALA	CAPACIDADE
BD1032	Bruno Pereira	101	50
LG1512	Marina Lucena	101	50
JV8796	Ana Barbosa	101	50
...	...	...	...

O atributo Capacidade depende do atributo Sala, e não da chave Cod_Turma

Terceira Forma Normal (3FN)

Passos para obtenção da 3FN em uma tabela

- Deixá-la em 2FN
- Identificar os atributos que não participam da chave primária da tabela
- Para cada um desses atributos, analisar se seu valor é determinado por algum outro atributo não pertencente à chave primária
- Criar novas tabelas para os atributos que não dependem exclusivamente da chave, incluindo o atributo determinante correspondente, e retirá-los da tabela original

Terceira Forma Normal (3FN)

As tabelas abaixo estão em 3FN

COD_TURMA	ALUNO	QTE_FALTAS
BD1032	Alice Luna	02
BD1032	Juliano Camargo	00
BD1032	Márcio Andrade	04
...	...	...

COD_TURMA	PROFESSOR	SALA
BD1032	Bruno Pereira	101
LG1512	Marina Lucena	101
JV8796	Ana Barbosa	101
...	...	...

SALA	CAPACIDADE
101	50
201	40
301	50
...	...

Regras Gerais – Normalização

1FN: Eliminar atributos multivalorados ou compostos

2FN: Eliminar atributos que dependem apenas de parte da chave primária composta

3FN: Eliminar atributos que dependem de atributos não-chave

CONSTRAINT

- Utiliza-se CONSTRAINTS para impor uma ou mais das seguintes restrições sobre uma coluna ou grupos de colunas:
  - Requerer que uma coluna ou grupo de colunas contenham valores NOT NULL
  - Especificar que o valor de uma coluna seja único na tabela referida
  - Especificar colunas como CHAVE PRIMÁRIA (Primary Key)
  - Especificar uma restrição de CHAVE ESTRANGEIRA (Foreign Key)
  - Requerer que o valor da coluna seja conforme uma valor pré-determinado (CHECK)

CONSTRAINT

- Podem ser definidas para uma tabela e colunas e são especificadas com o parte dos comandos CREATE ou ALTER TABLE

- É um conjunto de restrições de valores para validação.

- Toda declaração INSERT, UPDATE e DELETE causam uma avaliação de constraint.

CONSTRAINT – NOT NULL

- Definida no nível da coluna:

```
CREATE TABLE CO_NULO(
CODIGO NUMERIC(6),
NOME   VARCHAR(25) NOT NULL,
SALARIO NUMERIC(8,2),
DT_ADM DATE NOT NULL);
```

CONSTRAINT – PRIMARY KEY

- Pode ser elaborada na coluna ou na tabela

```
CREATE TABLE CO_PRIMARIA(
ID_DEPTO  NUMERIC(4),
NOME  VARCHAR(30) NOT NULL,
ID_LOC    NUMERIC(4),
CONSTRAINT CO_PRIMARIA_ID_PK PRIMARY KEY(ID_DEPTO));
```

CONSTRAINT – FOREIGN KEY

- Pode ser elaborada na coluna ou na tabela.

```
CREATE TABLE CO_SECUNDARIA(
EMPRESA NUMERIC(06),
NOME    VARCHAR(25) NOT NULL,
DEPARTAMENTO_ID NUMERIC(04),
CONSTRAINT CO_SECUNDARIA_DP_FK
FOREIGN KEY (DEPARTAMENTO_ID)
REFERENCES CO_PRIMARIA (ID_DEPTO))
```

CONSTRAINT – FOREIGN KEY

- Palavras chaves:
  - Foreign Key: Define a coluna ligada a uma outra tabela
  - References: Identifica a tabela e coluna na tabela relacionada
  - On Delete Cascade: Apaga as linhas dependentes na tabela filha quando a linha é apagada na tabela pai
  - On Delete Set Null: Converte os valores da foreign key para NULL

Criando Tabelas

a) Acidente (numero_placa_carro, cpf_motorista, nome_motorista, total_danos_acidente, data_acidente)

Acidente(numero_placa*, cpf_motorista*,total_danos_acidente,data_acidente)

Motorista(cpf_motorista*, nome_motorista)

Criando Tabelas

```
CREATE TABLE MOTORISTA(
  CPF_MOTORISTA NUMERIC(11) NOT NULL PRIMARY KEY,
    NOME_MOTORISTA VARCHAR(30) NOT NULL);


CREATE TABLE ACIDENTE (
   PLACA_VEICULO VARCHAR(7) NOT NULL,
  CPF_MOTORISTA NUMERIC(11) NOT NULL,
     TOTAL_DANOS_ACIDENTE NUMERIC(9,2),
    DATA_ACIDENTE DATE NOT NULL,
     CONSTRAINT ACIDENTE_PK PRIMARY KEY(PLACA_VEICULO, CPF_MOTORISTA),
CONSTRAINT MOTORISTA_FK FOREIGN KEY (CPF_MOTORISTA) REFERENCES
     MOTORISTA (CPF_MOTORISTA));
```

Criando Tabelas

b) Paciente (cod_paciente, nome_paciente, (fone_paciente), (crm_medico, nome_medico, data_consulta), cod_convenio, nome_convenio, (cod_exame, nome exame, diagnostico))

Paciente(cod_paciente*, nome_paciente, cod_convenio**)

Convenio(cod_convenio*,nome_convenio)

Telefone(cod_paciente*,fone_paciente*)

Consulta(cod_paciente*,crm_medico*, data_consulta)

Medico(crm_medico*,nome_medico)

Diagnostico(cod_paciente*,cod_exame*,diagnostico)

Exame(cod_exame*,nome_exame)

Manipulando Dados

- Uma manipulação de dados é executada quando:
  - Adiciona-se informações (linhas) na tabela
  - Modifica-se informações existentes na tabela
  - Remove-se informações existentes na tabela.

INSERT

- O comando INSERT é utilizado para inserir linhas em tabelas.
- A inclusão pode ser feita linha a linha ou os valores podem ser obtidos de outras tabelas.
- Podemos definir quais colunas serão preenchidas, informando valores para somente estas colunas.

INSERT - Sintaxe

- Adicionando novas linhas para uma tabela:

- INSERT INTO table
- [(column [,column...])]
- VALUES
- (value [,value...]);

- Somente uma linha por vez é inserida com esta sintaxe

INSERT

- A inserção de novas linhas (informações) poderá ser feita:
  - Inserindo uma nova linha contendo valores para cada coluna

    INSERT INTO DEPT (DEPTNO, DNAME, LOC)

    VALUES (60,'ESOFT','LONDRINA')
  - Associando valores a cada uma das colunas, não informando as colunas da tabela.

    INSERT INTO DEPT VALUES (65,'ORACLE','MARINGA','PR')

INSERT

- Inserindo linhas com valores nulos.
  - Método Implícito: Omitir a coluna da lista.

```
INSERT INTO DEPT (DEPTNO, DNAME)
VALUES (100,'RH');
```

Método Explícito: Especifica a palavra NULL nos valores.

```
INSERT INTO DEPT
VALUES (110,'CONTABIL',NULL)
```

UPDATE

- O comando UPDATE tem a finalidade de alterar informações já gravadas na base de dados.
- Possui uma cláusula WHERE, que determinará quais linhas serão modificadas.
- O Oracle faz um select implícito no banco de dados, para determinar as linhas que atentem a cláusula WHERE.

UPDATE

- Alterando linhas em uma tabela:

```
UPDATE TABLE
SET column = value [, column = value,...]
[WHERE  condition];
```

UPDATE

- A linha ou linhas que serão alteradas são especificadas na cláusula WHERE.
- UPDATE EMP

  SET DEPTNO = 20

  WHERE EMPNO = 1234
- Todas as linhas serão alteradas se for omitida a cláusula WHERE
- UPDATE EMP

  SET DEPTNO = 20

UPDATE

- Atualizando Linhas: Erro de Constraint de Integridade:

```
UPDATE EMP
SET DEPTNO = 55
WHERE DEPTNO = 110
```

- O departamento 55 não existe

DELETE

- Excluir linhas cadastradas no banco de dados.
- Da mesma forma que o UPDATE, a cláusula WHERE será responsável por determinar que linhas poderão ser removidas.
- Caso essa cláusula não seja informada, o comando tentará remover todas as linhas da tabela informada

DELETE

- Removendo linhas de uma tabela:

```
DELETE [FROM] table
[WHERE  condition]
```

Obs.: Se nenhuma linha for excluída, a mensagem ‘0 rows deleted’ é retornada

DELETE

- Especificando linhas a serem excluídas pela cláusula WHERE:

```
DELETE FROM DEPT
WHERE DNAME = 'Finance'
```

- Todas as linhas da tabela serão excluídas se for omitida a cláusula WHERE:

```
DELETE FROM DEPT
```

DELETE

- Removendo Linhas: Erro de Constraint de Integridade:
  - Não pode ser excluída uma linha que contenha uma primary key, que é utilizada como foreign key em outra tabela.

Linguagem de Manipulação de Dados (DML)

SELECT - Básico

- Selecionando todas as colunas de toda a tabela:

  – SELECT *
    FROM emp

- Selecionando colunas especificadas:

  – SELECT empno,
        ename
    FROM emp

SELECT – Expressões Aritméticas

- Criar expressões com números, datas usando operadores aritméticos

Operadores	Descrição
+	Somar
-	Subtrair
*	Multiplicar
/	Dividir

SELECT – Operadores Aritméticos


```
– SELECT ename,
         sal,
         10 * sal + 500
  FROM   emp
```

- Precedência dos Operadores:
  - Multiplicação e Divisão tem prioridade sobre adição e subtração
  - Operadores da mesma prioridade, são avaliados da esquerda para direita
  - Parênteses são utilizados para forçar a priorização

SELECT – Coluna Alias

- Renomeia o título de uma coluna
- Utilizado com colunas calculadas
  - SELECT name AS nome,
    sal AS salario
    FROM emp
- Ou
- SELECT name “nome”,
  sal “Salario”
  FROM emp

SELECT – Linhas Duplicadas

- Ao menos que seja indicado de outra maneira uma consulta numa base de dados poderá trazer linhas duplicadas, para evitar isso utilizamos o comando DISTINCT

- SELECT DISTINCT deptno
  FROM emp

SELECT – Cláusula WHERE

- As restrições são estabelecidas com o uso da cláusula WHERE.
- Determina que linhas devem ser relacionadas.
- SELECT * | {DISTINCT} coluna | expressão [alias], ... } FROM table;
- WHERE condição;
- SELECT * FROM EMP **WHERE** SAL > 3000

SELECT – Cláusula WHERE

- Condições de Comparação:

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

SELECT – Cláusula WHERE

- Outras Condições de Comparação:

Operador	Significado
BETWEEN ... AND ...	Entre dois valores (inclusive)
IN (set)	Selecionando uma lista de valores
LIKE	Texto dentro de uma determinada coluna
IS NULL	Se o valor é nulo

SELECT – Cláusula WHERE

- Condições Lógicas:

Operador	Significado
AND	Retorna verdadeiro se ambas condições forem verdadeiras
OR	Retorna verdadeiro se pelo menos uma condição for verdadeira
NOT	Retorna verdadeiro se a condição é falsa.

SELECT – Regras de Precedência

1	Operadores Aritméticos
2	Concatenação
3	Condições de Comparação
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT
7	AND
8	OR

SELECT – Cláusula ORDER BY

- Permite que as linhas resultantes da consulta sejam apresentadas em uma determinada ordem:
  - ASC – Ordem Ascendente (default).
  - DESC – Ordem Descendente.

  - SELECT ename,
    sal
    FROM emp
    ORDER BY sal