



## Banco de Dados I

Prof. Tiago Eugenio de Melo  
[tmelo@uea.edu.br](mailto:tmelo@uea.edu.br)

# Sumário

- Discussão sobre o plano de ensino da disciplina
- Conceitos Básicos de Banco de Dados
- Modelo de Entidade-Relacionamento
- Modelo Entidade-Relacionamento Estendido
- Modelo Relacional
- Ferramenta de Modelagem
- *Structured Query Language* (SQL)

# Conceitos Básicos: Banco de Dados

- Banco de Dados (BD) se transformou em um componente essencial do dia-a-dia na sociedade moderna.

## Exemplos:

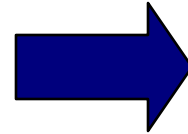
- ir a um banco para fazer depósito ou retirada de dinheiro.
  - fazer a reserva em um hotel ou em uma companhia aérea.
  - fazer pesquisa de itens em uma biblioteca computadorizada.
  - pesquisar preços de itens em um supermercado.
- As atividades acima são exemplos de aplicações tradicionais de BD. Onde a maioria das informações são armazenadas através de textos ou números.
  - Há poucos anos atrás, a tecnologia permitiu novas aplicações para BD

## Exemplos:

- Banco de dados multimídia:** armazena figuras, som e vídeo.
- SIGS** - Sistemas de informações geográficas: armazenam e analisam mapas, tempo e imagem de satélite.
- Sistemas em tempo real:** controle de chão de fábrica e processos de manufatura.

# Sistemas de Informação

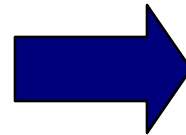
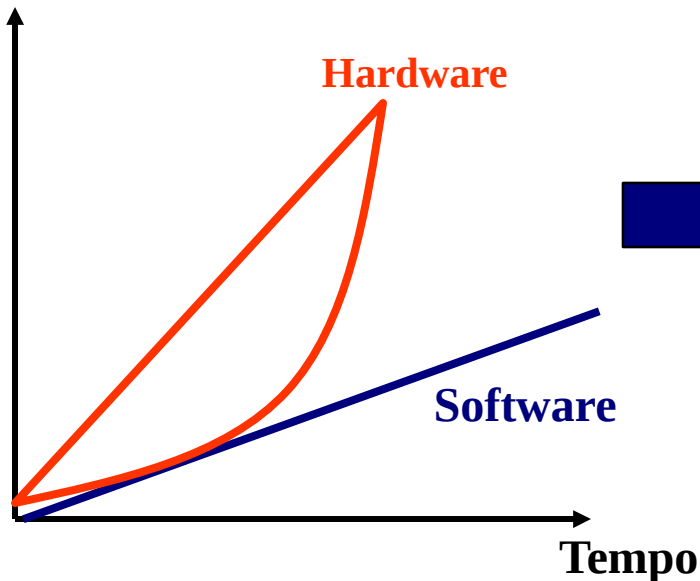
- AUMENTO DE COMPETITIVIDADE
- VANTAGEM ESTRATÉGICA



## Sistemas

- Passagem Aérea
- Supermercado

## Avanço tecnológico



## Processo de Desenvolvimento

- falta de métodos padrões
- falta de ferramentas produtivas

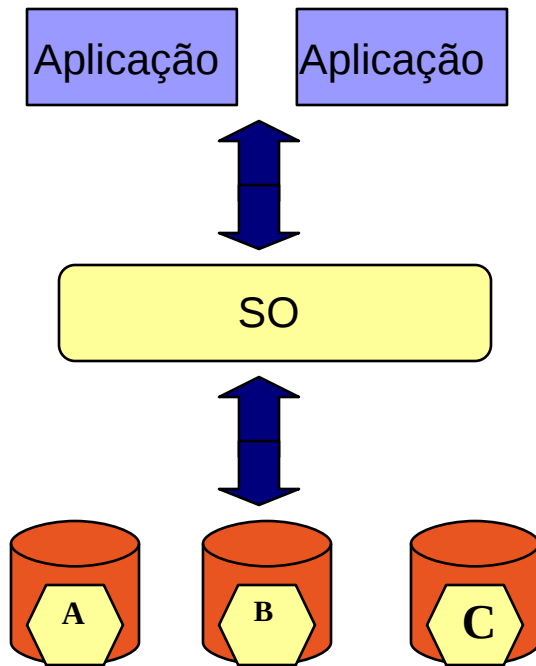
# Conceitos Básicos: Banco de Dados

- É uma coleção de **dados inter-relacionados**, representando informações sobre um domínio específico (conceito geral)
  - um BD representa **aspectos do mundo real**. Mudanças no mundo real são refletidas no BD.
  - um BD é uma coleção lógica e coerente de dados com relacionamentos intrínsecos.
    - ➔ um conjunto de dados sem nenhum relacionamento, não pode ser considerado um BD.
  - um BD é projetado, construído e mantido para uma **proposta específica**. É direcionado a um grupo de usuários de uma determinada aplicação.
  - um BD pode possuir qualquer tamanho/complexidade.
- Em outras palavras, um BD tem alguma fonte onde os dados são derivados, algum grau de interação com eventos no mundo real, e uma audiência interessada no conteúdo desse BD

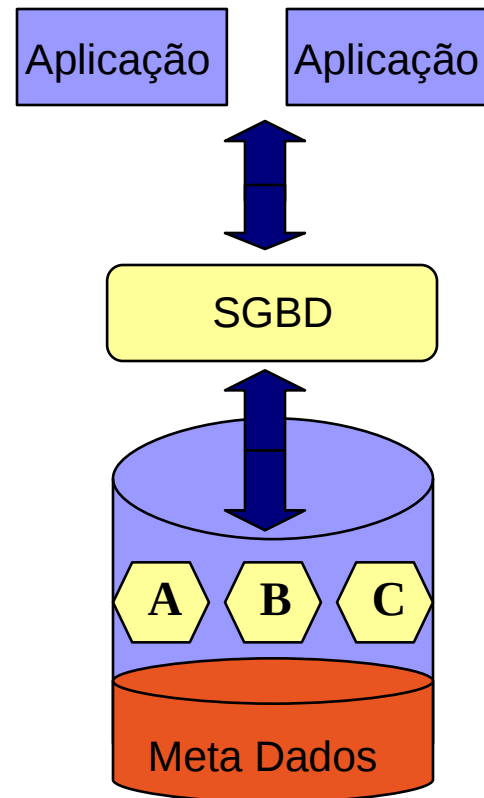
# Conceitos Básicos: Banco de Dados

- Criação/Controle de um BD:
  - manual
  - por um grupo de aplicações específicas
  - Sistema de Gerenciamento de Banco de Dados (**SGBD**): é uma coleção de programas que permite aos usuários criar e manter um BD

# Sistemas de Informação: Arquivos x BD



os aplicativos acessam e manipulam os arquivos diretamente nos discos



os aplicativos acessam e manipulam as informações através dos SGBDs

# Sistemas de Informação baseados em Arquivos

- tipo de arquivo/formato do registro escolhido de acordo com as **necessidades** de uma aplicação
- problemas aparecem na manutenção, evolução ou integração de sistemas
- falta de gestão centralizada de dados
- falta de autonomia dos dados em relação aos programas
- falta de facilidade de alto nível para tratamento de problemas comuns a qualquer manipulação de dados



# Sistemas de Informação baseados em arquivos

- Complexidade / Volume de registros
  - número máximo de arquivos
  - tamanho de memória
  - limitações do tipo de arquivo, tipo de acesso
  - preocupações técnicas junto com problemas de domínio

## **Ex: fazer empréstimo de um livro**

- sem reservas?
- sem multas pendentes?
- registra empréstimo
- abre arquivos (fechando outros...)
- carrega registros na memória (abre índice, usa o ponteiro, estourou memória?, ...)

# Sistemas de Informação baseados em arquivos

- Falta de integração e centralização
  - redundância
  - inconsistência
  - entrada repetida de informações
  - usuário tem a responsabilidade de garantir a sincronia entre as diferentes cópias da mesma informação
  
- Dificuldade de acesso à nova informação
  - nova informação = nova aplicação
  
- Isolamento
  - a organização sabe que os dados existem?
  - formato de arquivo é apropriado para uma nova aplicação?
  - é possível integrar dados de diferentes sistemas?

# Sistemas de Informação baseados em arquivos

## ■ Problemas de integridade

- Os valores dos dados atribuídos e armazenados em um banco de dados devem satisfazer certas restrições para manutenção da consistência.

## ■ Problemas de atomicidade

- Em muitas aplicações é crucial assegurar que, uma vez detectada uma falha, os dados sejam salvos em seu último estado consistente.

# Sistemas de Informação baseados em arquivos

- Concorrência
  - difícil implementação
  
- Tolerância a falhas
  - falta de luz, interrupção de funcionamento, etc
  - cópias? restauração do estado anterior? consistência da base?
  
- Segurança
  - acesso diferenciado por tipo de usuário
  - ex:
    - funcionário visualiza reserva de livros, identifica usuários e cancela reservas
    - usuário visualiza reserva de livros

# Características da Abordagem de Banco de Dados

## ■ natureza auto-contida

- um BD armazena **dados + os dados que descrevem esses dados** (catálogo)
  - O catálogo traz informações da estrutura do BD, formato dos dados, restrições de valores que os dados podem assumir, quem pode acessar os dados, etc
- Um BD pode guardar informações no catálogo, referentes às restrições. Se um registro for removido de um arquivo, automaticamente os registros serão também removidos em arquivos interligados (preservando a integridade)

# Características da Abordagem de Banco de Dados

## ■ Acesso aos dados

- Os SGBDs são responsáveis pelo **acesso concorrente** a um mesmo dado, deixando os programas de ter este tipo de responsabilidade.
- Como um BD é um repositório centralizado de dados, os SGBDs provêm **mecanismo de controle de acesso** aos dados, ou seja, só permite acesso para os usuários autorizados.

# Características da Abordagem de Banco de Dados

## ■ Abstração e Visões de dados

- O SGBD provê uma **representação conceitual** dos dados, excluindo detalhes de como os dados são armazenados.
- ***O maior propósito de um BD é o de oferecer aos usuários uma visão abstrata dos dados.*** Isto é, o sistema esconde certos detalhes de como o dado é armazenado e mantido. A complexidade está escondida através de diversos níveis de abstração que simplificam a interação do usuário do Sistema.
- Um BD permite que os usuários tenham visões abstratas dos dados, isso é possível porque os BDs são construídos através de um **modelo de dados**.
- Um BD tem muitos usuários com visões diferentes dos dados. Um SGBD deve prover mecanismos para definir múltiplas visões dos dados.

# Características da Abordagem de Banco de Dados

- Independência entre dados e programas
  - Na abordagem de arquivos, cada programa possui em seu código uma descrição da estrutura dos arquivos
  - Na abordagem de banco de dados existe um catálogo que permite que qualquer programa possa recuperar as informações dinamicamente



# Um SGBD

- É um conjunto de aplicações usado para gerenciar um Banco de Dados:
  - armazenar, recuperar e modificar informações.
  - proporcionar um ambiente conveniente e eficiente para recuperar e armazenar informações de um banco de dados.
  - manipular grande volume de informações.
  - prover segurança às informações armazenadas.
  - controlar concorrência, evitando resultados anômalos na atualização de informações no BD.
  - prover mecanismos para criação e manipulação de estruturas de armazenamento de informação.
  - restringir acesso a dados de usuários não autorizados.

# Um SGBD

- Um SGBD é um software de propósito geral que facilita os seguintes processos:
  - Definição
  - Construção
  - Manipulação
- É um meio conveniente e eficiente para recuperação e armazenamento.
- **IMPORTANTE: Não é necessário usar um SGBD para implementar um banco de dados.**

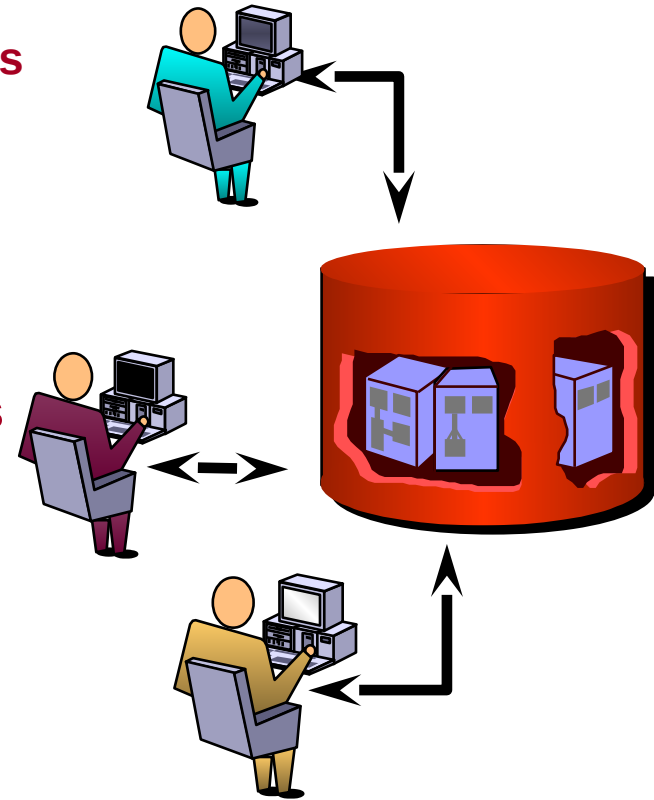
# SGBDs - Atores

## ■ Programador de Aplicações

- **Definição e implementação de programas que USAM a base de dados**
  - **Programas enviam solicitações de serviços ao SGBD**
- **Trabalham sobre a definição lógica ou sobre uma visão externa específica**

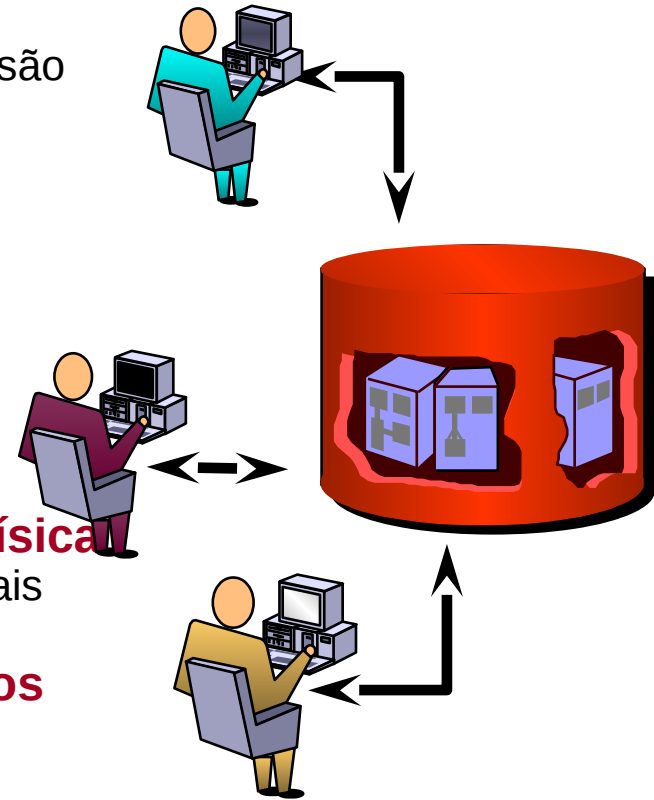
## ■ Usuário Final

- **Usam a base de dados para suas funções do dia-a-dia**
- **Interagem com o sistema a partir de uma estação de trabalho**
  - aplicação
  - linguagem de consulta interativa



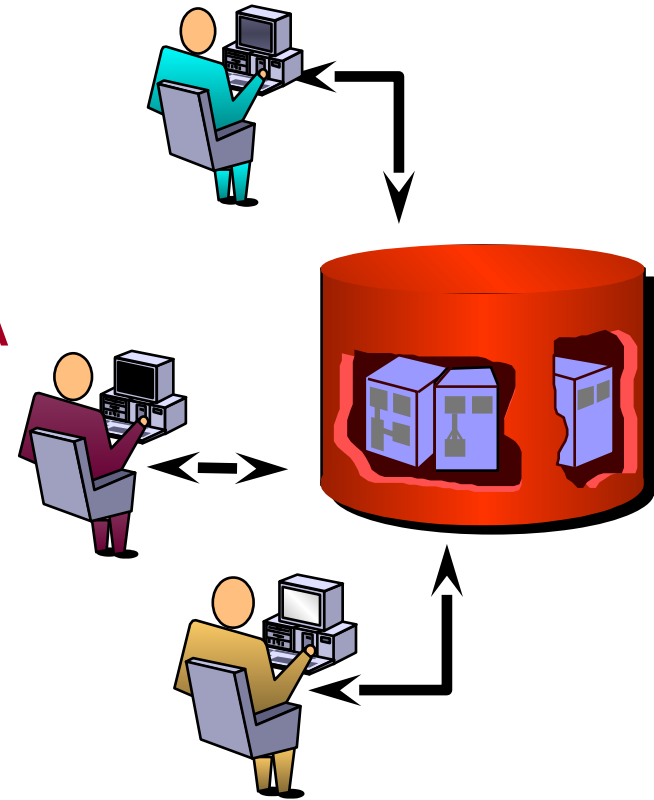
# SGBDs - Atores

- Administrador do Banco de Dados (DBA)
  - **Pessoa(s) que detêm a responsabilidade (técnica) central sobre os dados**
  - **Definição do esquema**
    - conjunto de descrições escritas com a DDL que são armazenadas no Dicionário de Dados
  - **Definição da estrutura de armazenamento**
    - Requisitos de espaço de armazenamento, desempenho, concorrência, criação ou não de índices, etc.
  - **Monitorar desempenho**
  - **Modificação do esquema e Reorganização física**
    - desempenho e alteração nos requisitos funcionais
  - **Concessão de autorização de acesso a dados**
  - **Especificação de restrição de integridade**
  - **Definição de estratégias de recuperação de dados**



# SGBDs - Atores

- **Projetista do Banco de Dados**
  - **Identificar requisitos informacionais da corporação**
  - **Escolher estruturas apropriadas para representação da informação**
  - **Interagir com o usuário**
  - **Pode ser confundido com o papel de DBA**



# Quando usar um SGBD ?

- Controlar redundância
  - Aumento da consistência através de uma maior integração e centralização dos dados
- Restringir acesso não autorizado
- Persistência dos dados (além da execução)
- Representação de relações complexas entre os dados
- Estabelecimento de regras e padrões
- Fornecer back-up e recuperação
- Controle de acesso concorrente
- Esforço reduzido de desenvolvimento para aplicações orientadas a dados

# Razões para **NÃO** se usar um SGBD

- Custo maior que o benefício !!!
  - custo de HW, SW e treinamento.
  - soluções genéricas para definir e processar dados.
  - custo pela segurança, controle de concorrência, recuperação, manutenção de integridade, etc.
- Base de dados e aplicações simples, bem definidas, e **sem previsão de alteração** a médio prazo.
- Aplicações com **requisitos de desempenho** (ex: tempo real).
- Aplicações mono-usuário.

# Modelos de Dados (Data Models)



Modelo é a representação abstrata e simplificada de uma determinada realidade, com a qual se pode explicar ou testar o seu comportamento, em sua totalidade ou em partes antes de sua existência real.



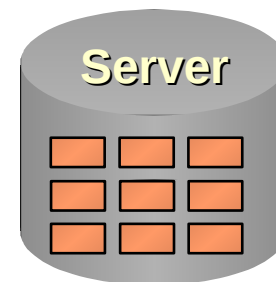
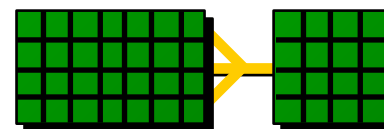
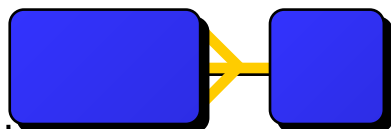
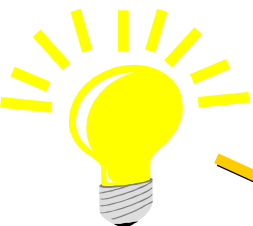
# Modelo de Dados

Modelo do Sistema  
(mente do Cliente)

Modelo de Dados de Alto Nível  
(conceitual)

Modelo de Dados de Nível Intermediário  
(representação ou implementação)

Modelo de Baixo Nível (físico)

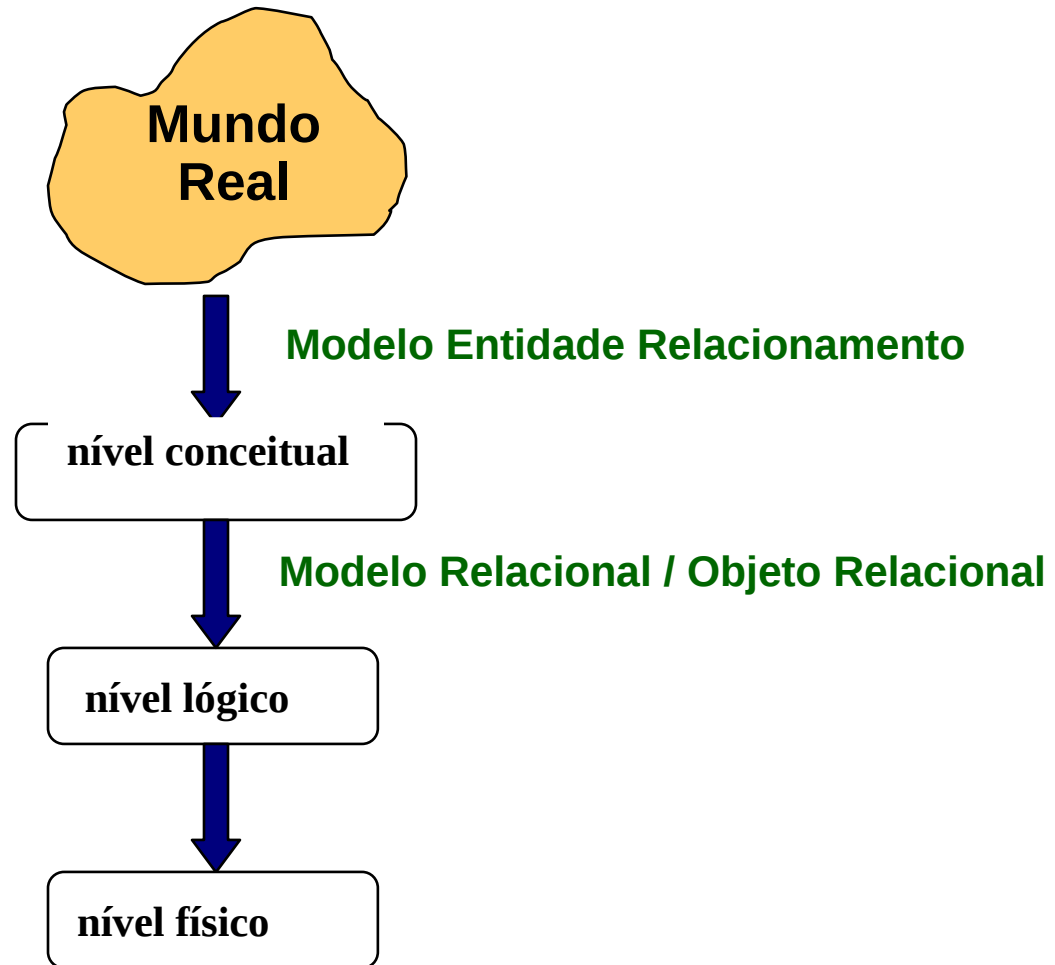


Modelo de Entidade do Cliente

Representação Tabular do Modelo de Entidade

Tabelas no disco

# Categorías de Modelos de datos



# Categorias de Modelos de dados

- nível conceitual

---

- é o nível mais alto de abstração, fala de objetos do mundo real e suas respectivas operações. Ex: aluno, livro, empregado. O importante nesta fase é escolher abstrações do mundo real que capturem o universo de discurso, segundo o ponto de vista do sistema a ser construído. **A ênfase está na informação** e não como ela será implementada.

- nível lógico

---

- neste nível de abstração são **escolhidas as estruturas lógicas** que representarão os objetos do modelo conceitual no computador.

- nível físico

---

- é o mais baixo nível de abstração e descreve como os dados serão armazenados, como deve ser a representação física das estruturas lógicas definidas no modelo lógico. Especifica-se também as operações do modelo lógico usando-se a representação física.

# Modelos de dados, Esquemas e Instâncias

## ■ Modelo de dados

---

- um modelo de dados consiste de um conjunto de conceitos que é usado para **descrever** o banco de dados da mesma forma que uma linguagem de programação é usada para descrever um programa
- Descreve a **estrutura** do Banco de Dados. A estrutura de um BD significa os **seus objetos**, **tipos de dados**, **procedimentos**, e as **restrições** que devem ser obedecidas, é geralmente feita segundo um **Modelo de Dados**

## ■ Esquema

---

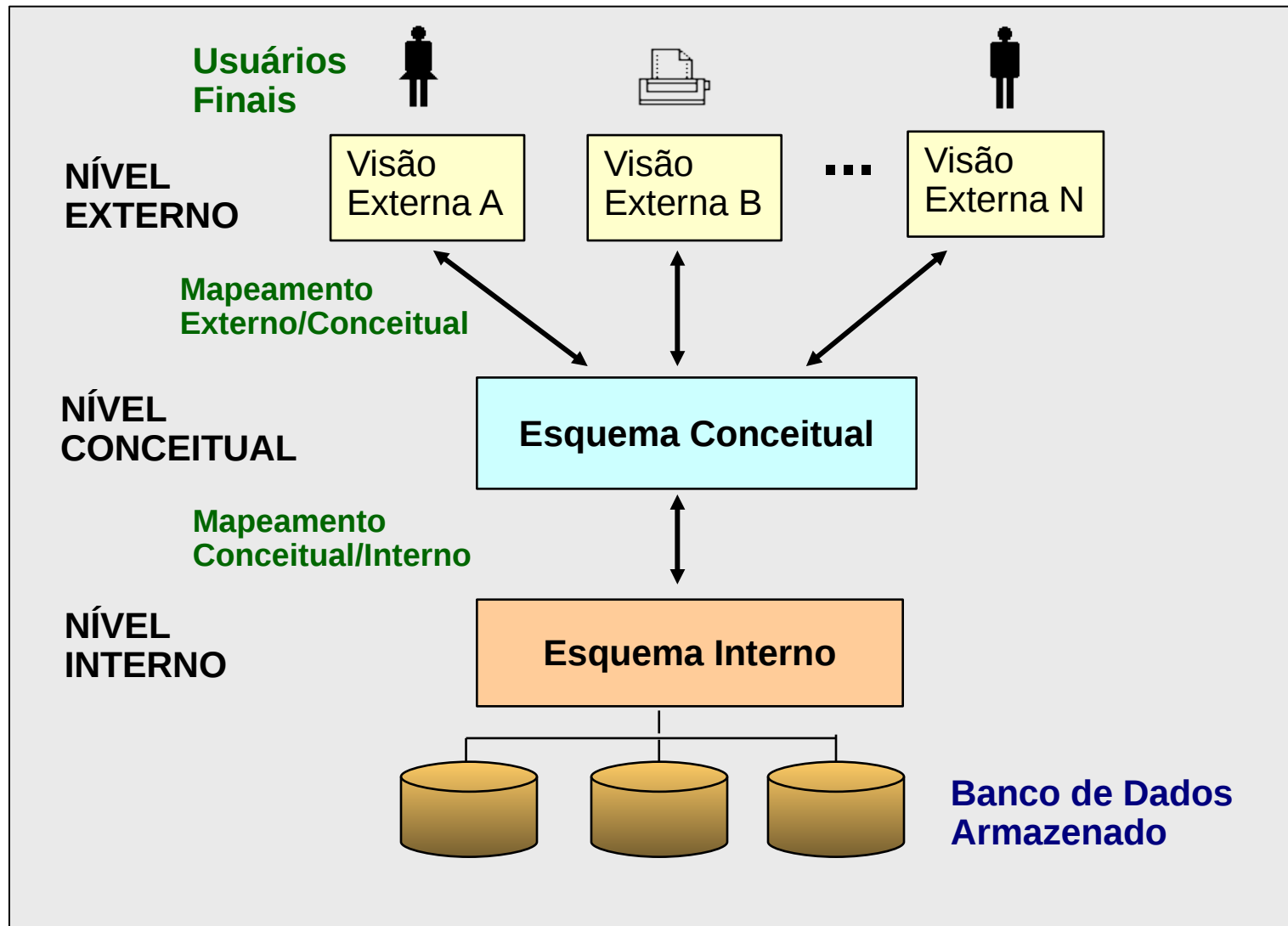
- A descrição de um BD é chamada de Esquema de Banco de Dados
- O Projeto do BD é denominado de esquema do BD

## ■ Instância (estado de um BD)

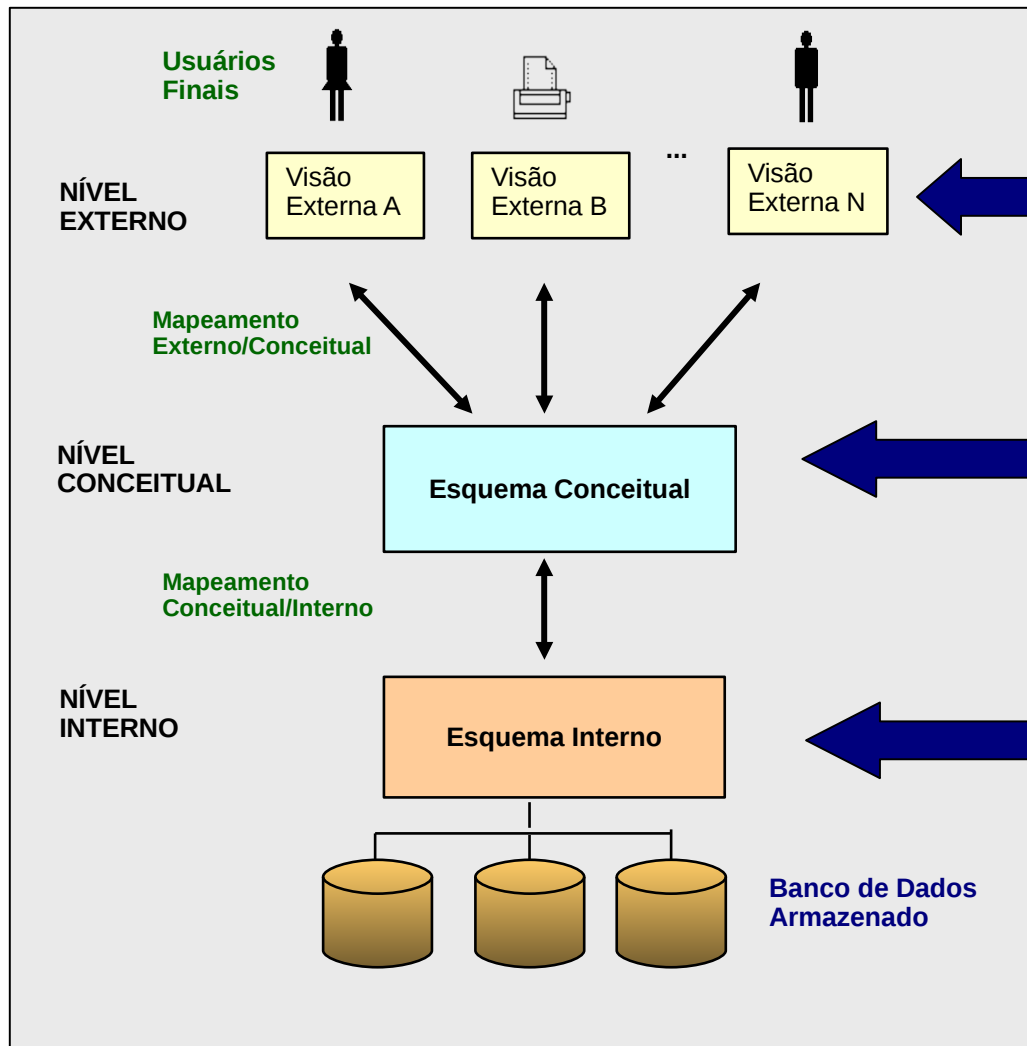
---

- A coleção de informações armazenadas em um BD em um dado momento no tempo é denominado instâncias do BD

# Arquitetura dos 3 Esquemas



# Esquemas x Modelos de Dados



Este Nível descreve a parte de um BD para um grupo de usuário particular. Um **MODELO DE DADOS DE ALTO NÍVEL** ou de **IMPLEMENTAÇÃO** pode ser usado.

O Esquema Conceitual esconde os detalhes das estruturas físicas. Um **MODELO DE DADOS DE ALTO NÍVEL** ou de **IMPLEMENTAÇÃO** pode ser usado.

O Esquema interno usa um **MODELO DE DADOS FÍSICO** e descreve detalhes sobre o armazenamento dos dados

# Independência de Dados

- Habilidade de modificar a definição do esquema em um nível sem afetar a definição do esquema no próximo nível acima
  - **Independência de Dados Física** : modificações no esquema físico não causam modificações nos programas
    - Geralmente ocorrem para melhorar desempenho (reorganização física)
    - Refere-se ao ISOLAMENTO de uma aplicação das estruturas físicas de armazenamento
  - **Independência de Dados Lógica**: modificações no esquema conceitual não causam modificações nos programas

# Modelagem da Realidade

## Processo

identificação de um conjunto de procedimentos que nela se realizam, interação entre eles e identificação dos dados necessários para a execução desses procedimentos

**Enfoques**

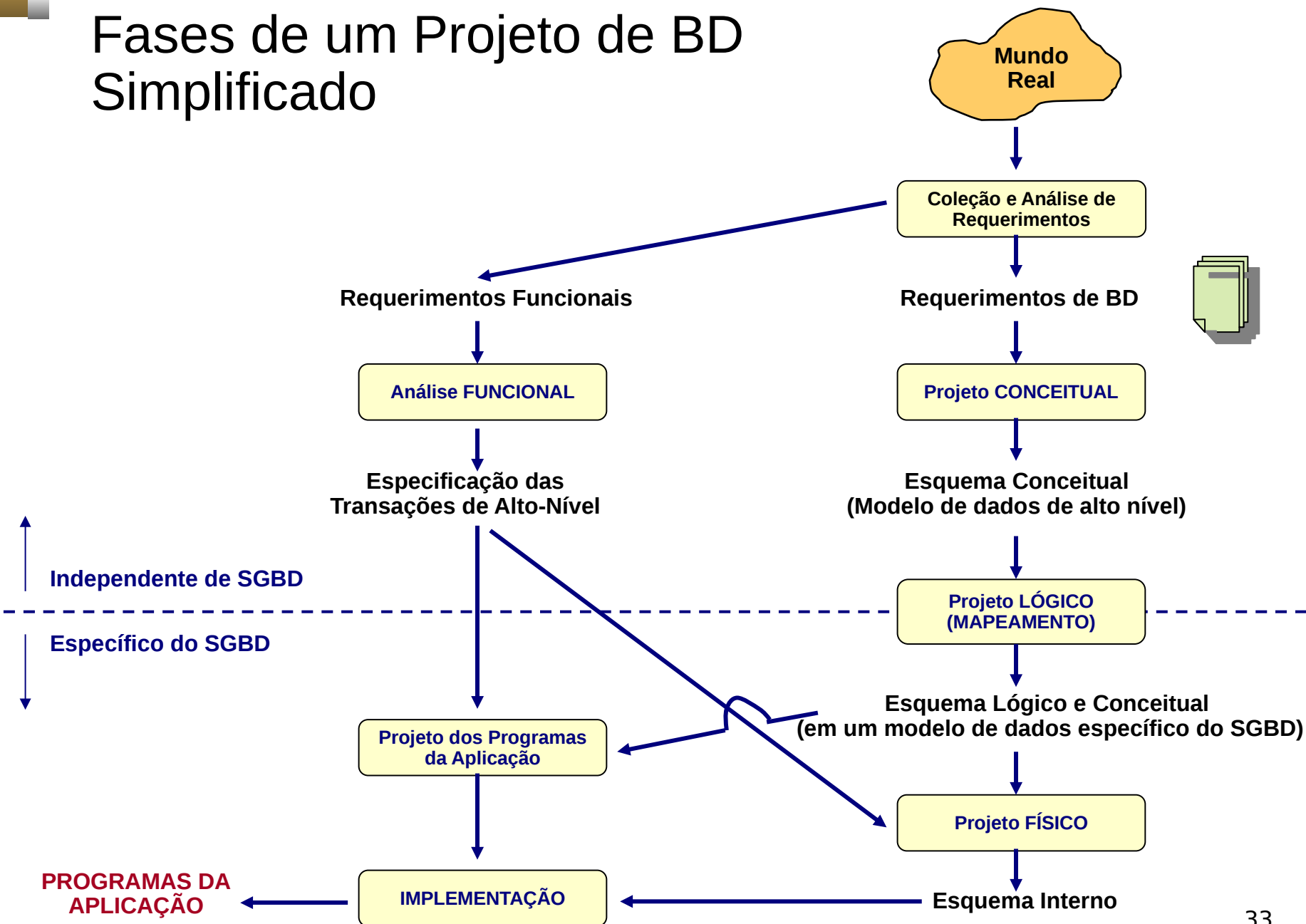
```
graph LR; Enfoques --> Processo; Enfoques --> Dados;
```

## Dados

identificação dos objetos que compõe a realidade, seguida pela identificação das operações que incidem sobre estes

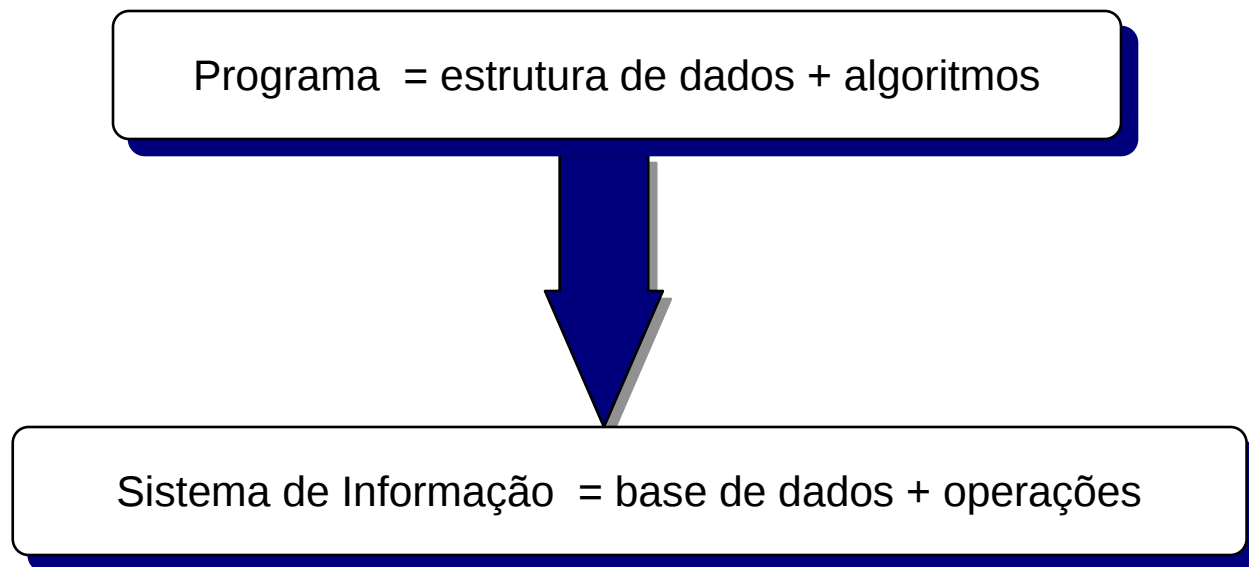


# Fases de um Projeto de BD Simplificado



# Modelagem de Dados

**Em Resumo:** É o processo de especificação das estruturas de dados e regras de negócio para a definição de um *sistema de informação*



# Questões para revisão

- Qual é a diferença entre banco de dados e sistemas de gerenciamento de banco de dados?
- Explique as vantagens e desvantagens no uso de arquivos em relação aos sistemas de gerenciamento de bancos de dados.
- Explique duas funções de um DBA.
- Quais os níveis de abstração de dados?
- O que descreve o modelo conceitual?
- Qual é a função do catálogo em um BD?
- O que é um SGBD e qual é o seu objetivo?
- Quais as vantagens de um SGBD?
- Qual a importância dos modelos de dados para a estrutura de um BD?

# Questões para revisão

- Comente dois problemas no uso de arquivos para armazenamento de dados.
- O que é inconsistência? Dê um exemplo.
- O que representa a natureza auto-contida dos bancos de dados?
- O que é armazenado no catálogo de um banco de dados?
- Comente duas atividades de um administrador de banco de dados (DBA).
- Em que situações o uso de um sistema gerenciador de banco de dados não é recomendável?
- Explique a diferença entre modelo e esquema de banco de dados.
- Dê um exemplo de instância de banco de dados.

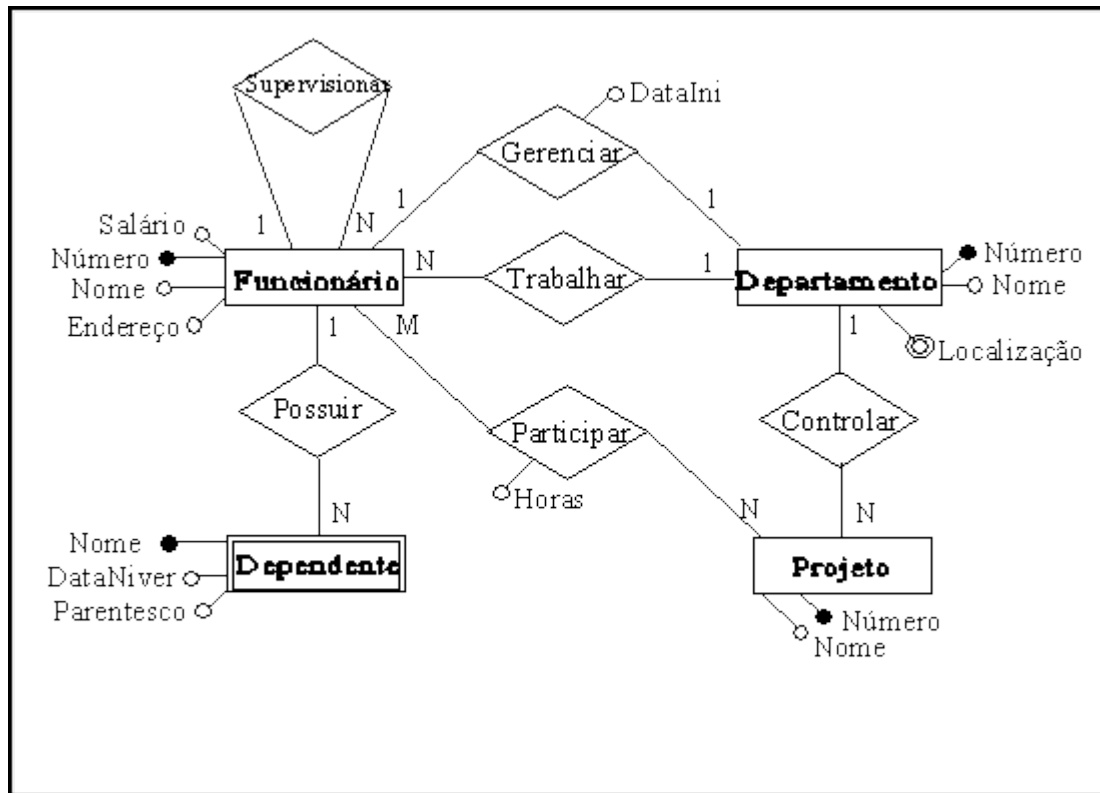


# Modelo de Entidade e Relacionamento

Banco de Dados

# Modelo Entidade Relacionamento - MER

- proporciona uma visão lógica de alto nível dos dados
- é uma descrição abstrata de uma porção do mundo real
- todos os dados são visualizados como fatos específicos sobre entidades, relacionamentos e atributos
- através do MER, podemos ter uma fotografia do sistema
- as entidades, relacionamentos e atributos descrevem as regras de negócio da empresa

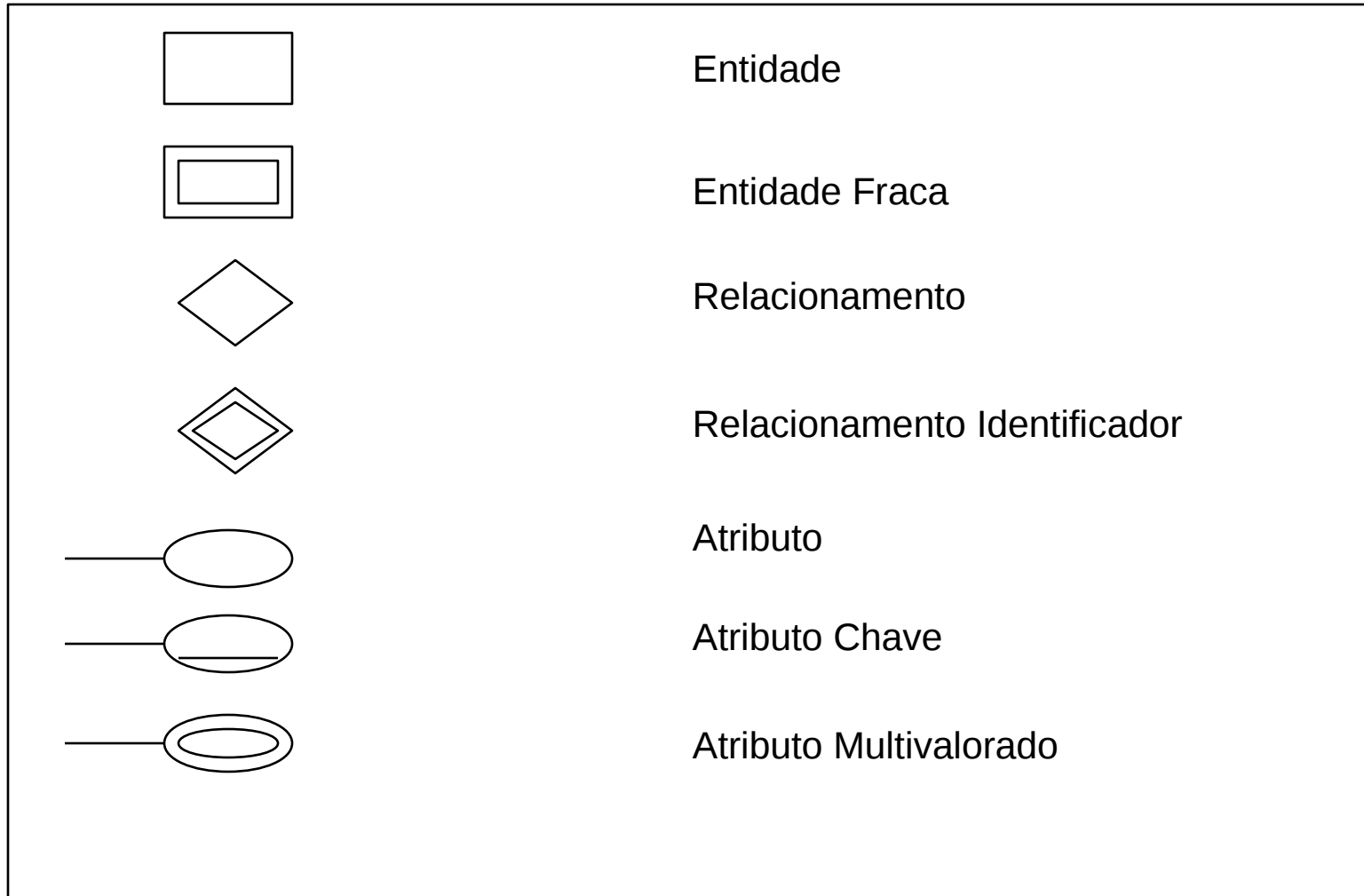


relacionamento

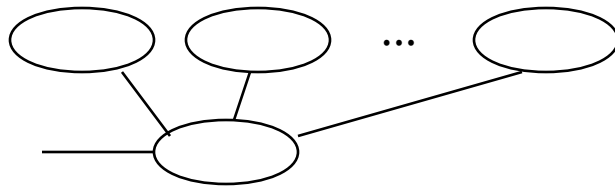
atributo

entidade

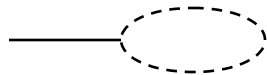
# Modelo Entidade Relacionamento - Convenções



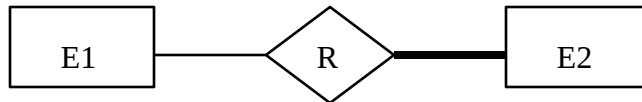
# Modelo Entidade Relacionamento - Convenções



Atributo Composto

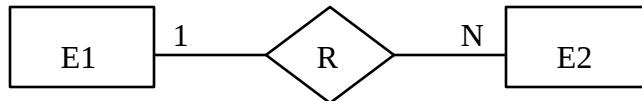


Atributo Derivado

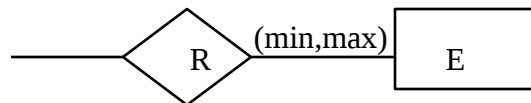


Participação total de E2 em R

Ex: Agência e Conta Corrente



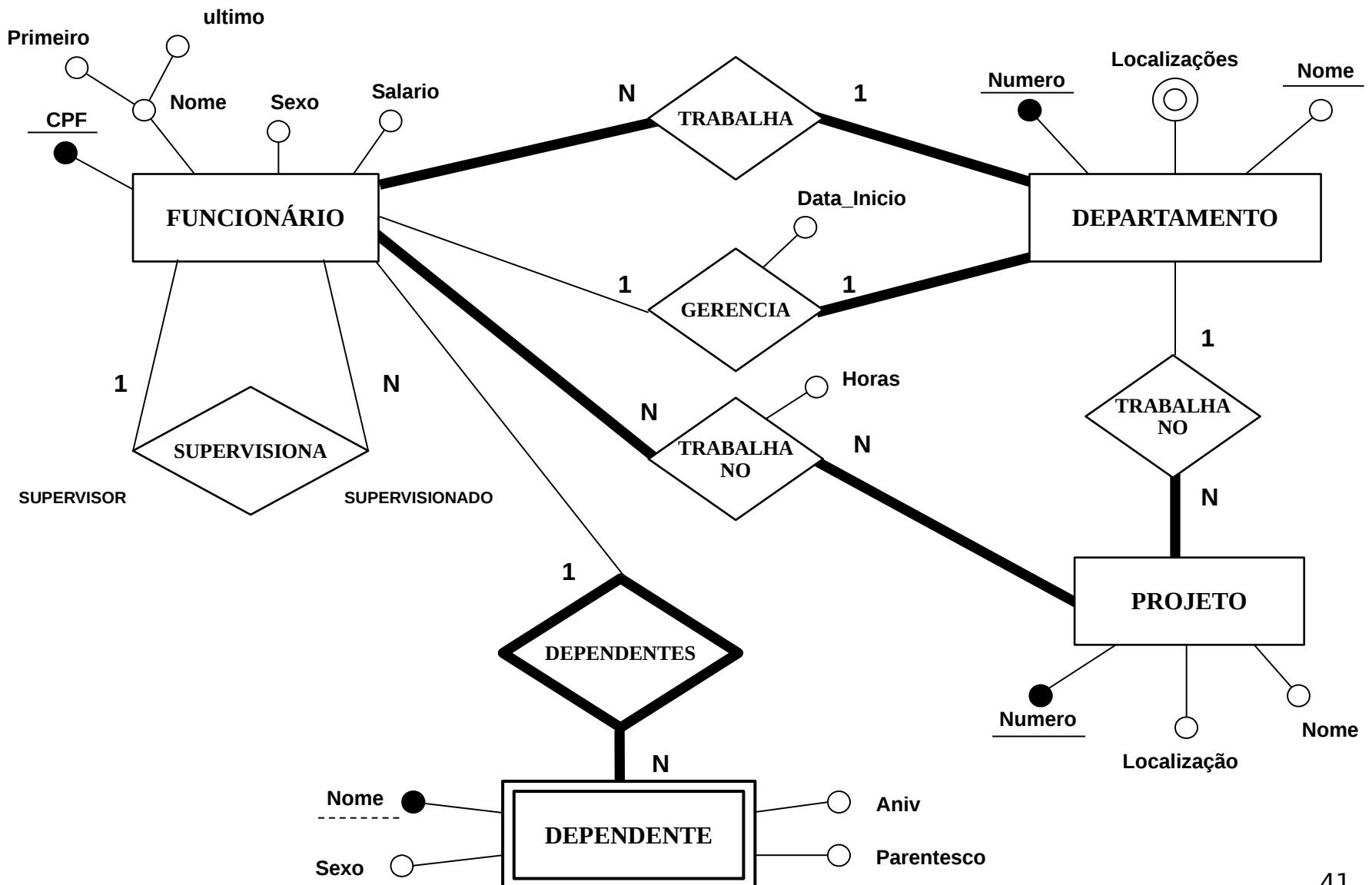
Cardinalidade 1:N para E1:E2 em R



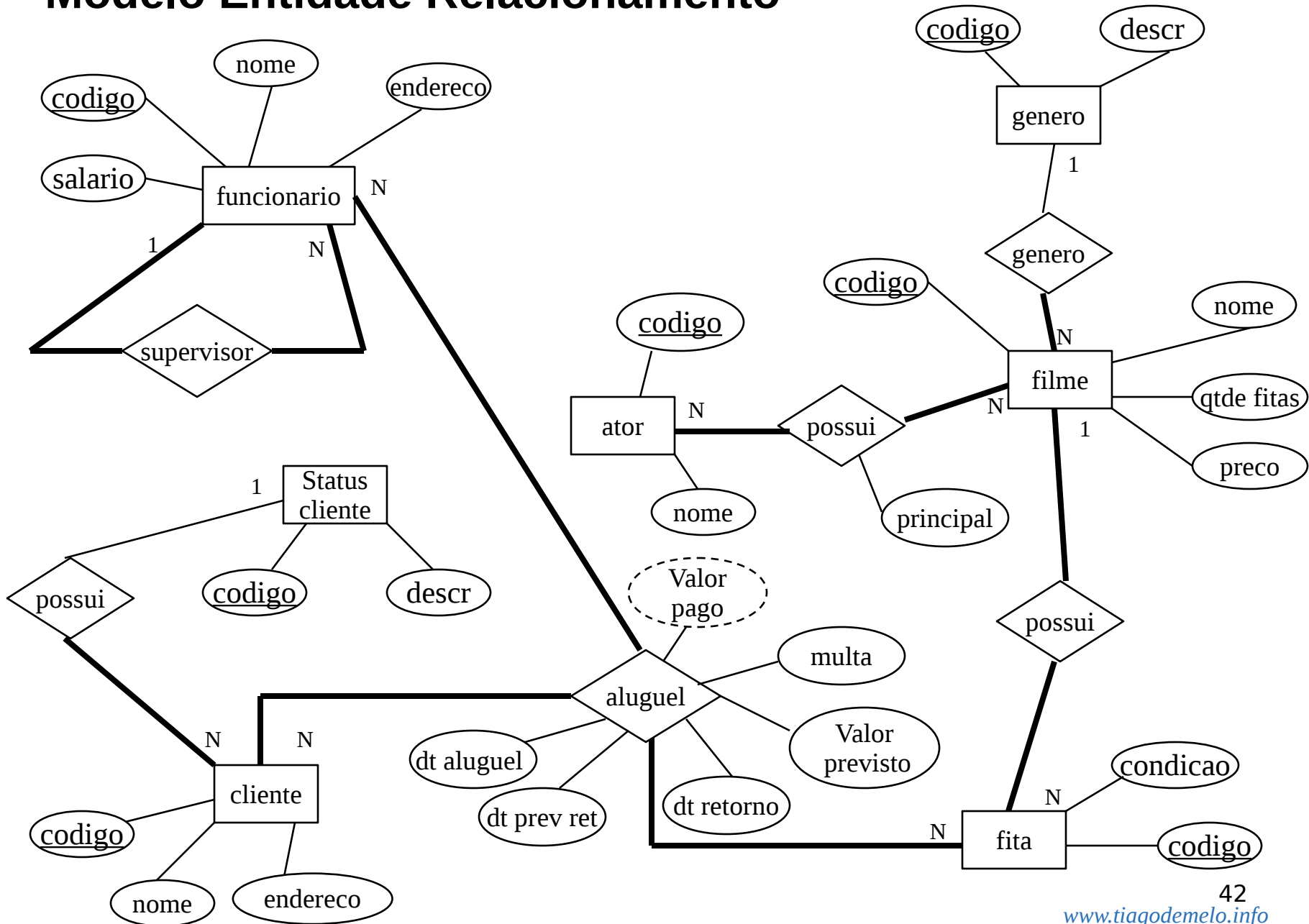
Constraint (min,max) de E em R



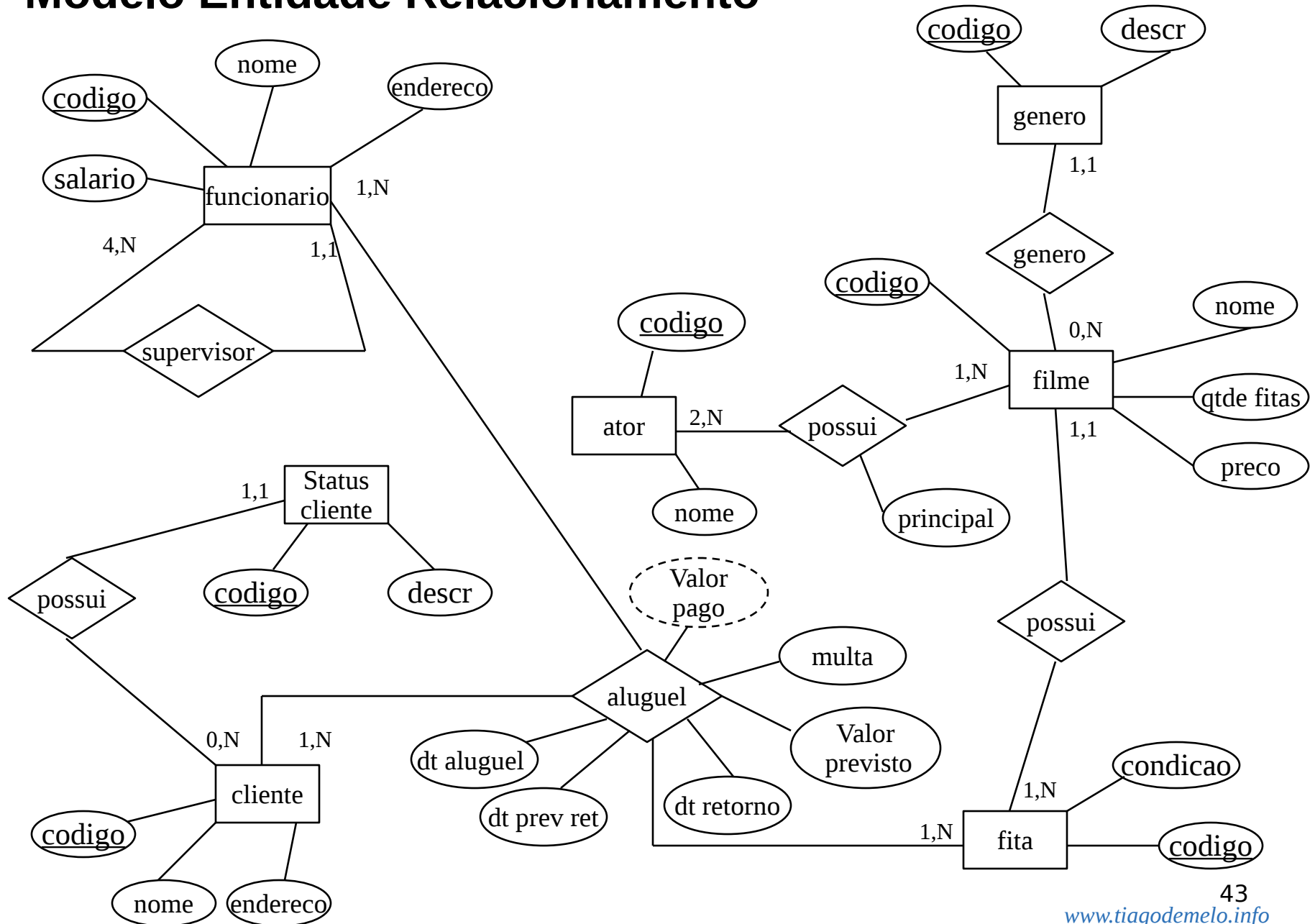
# Diagrama ER do Banco de uma Empresa



# Modelo Entidade Relacionamento



# Modelo Entidade Relacionamento



# Componentes do MER

## ■ Entidade

- qualquer coisa pela qual desejamos guardar informação
- conjunto de objetos individuais chamados *instâncias*
- uma *instância* é uma simples ocorrência de uma entidade
- cada instância representa um conjunto de fatos sobre a entidade
- uma instância deve ter uma identidade distinta de todas as outras

# Entidades Dependentes e Independentes

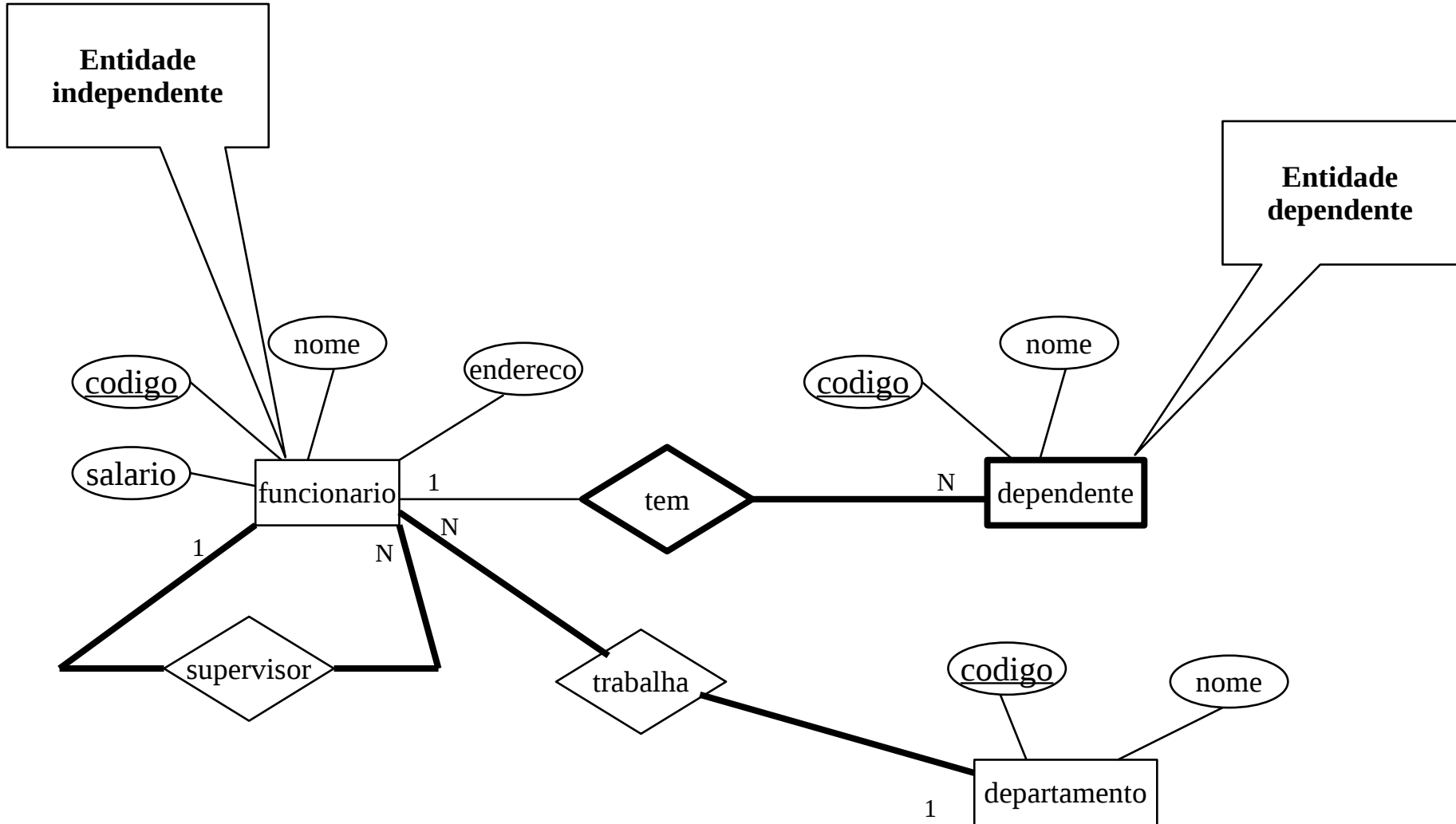
## ■ *Dependentes*

- entidades que dependem de outras para sua existência (*dependência por existência*)
- entidades que dependem de outras para sua identificação (*dependência por identificação*)

## ■ *Independentes*

- entidades que não dependem de outras para sua existência e identificação

# Entidades Dependentes e Independentes



# Componentes do MER

## ■ *Atributo*

- características particulares do conjunto de entidades
- os fatos ou propriedades de uma entidade são chamados de atributos
- cada atributo de uma entidade representa uma informação sobre essa entidade

## ■ *Relacionamento*

- relacionamento representa um link ou associação entre entidades

# Relacionamentos: Identificadores e não Identificadores

## ■ **Identificadores**

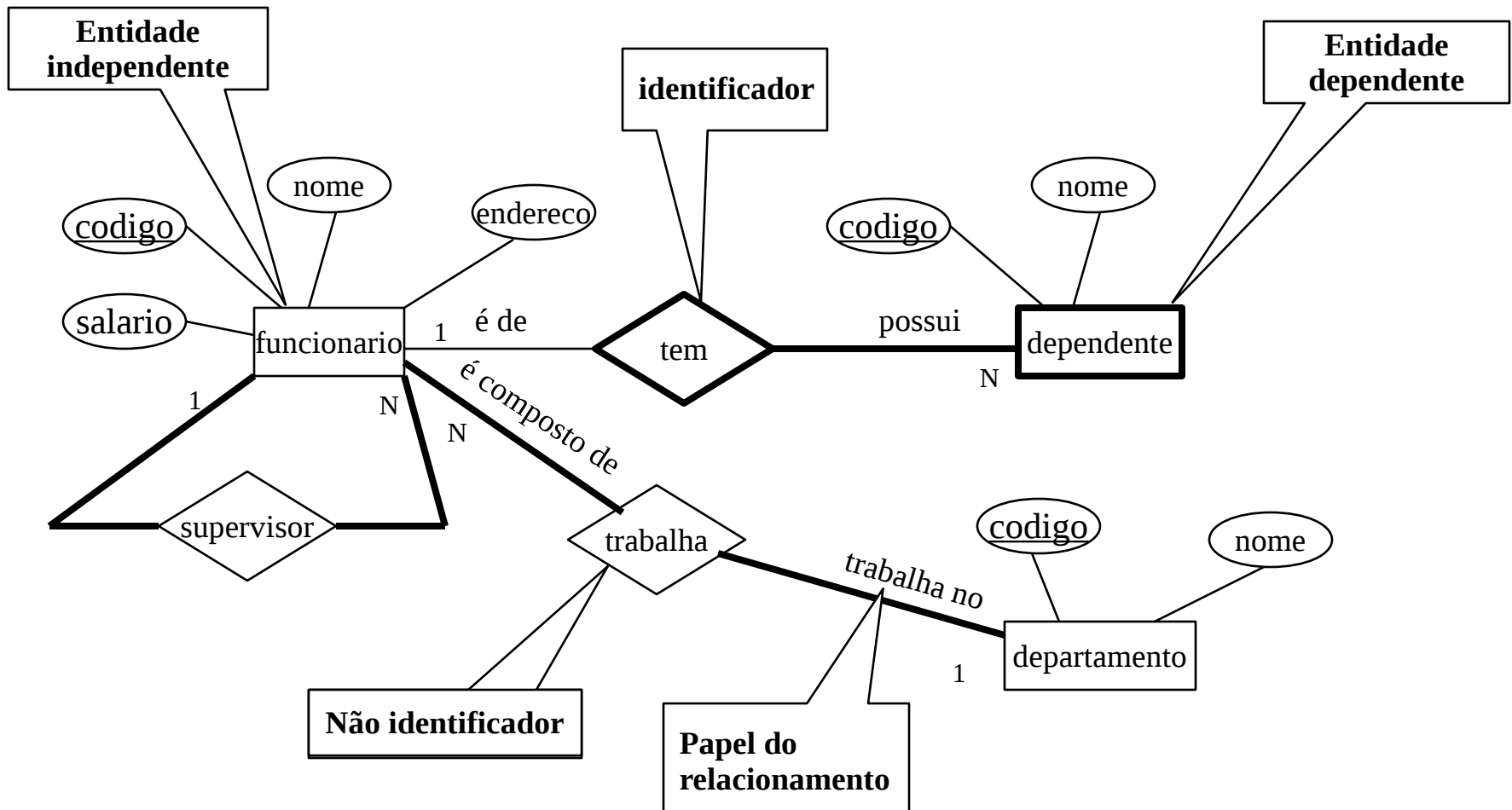
- o conceito de entidade dependente e independente é reforçado pelo tipo de relacionamento
- quando se quer que uma entidade se torne dependente, cria-se um *relacionamento identificador*

## ■ **Não Identificadores**

- também conecta entidade mãe e filha, porém não é capaz de identificar de forma única, instâncias na entidade filha

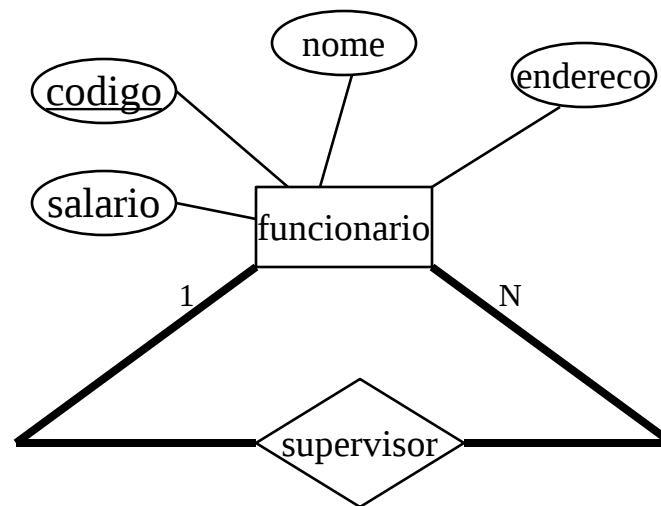


# Relacionamentos: Identificadores e não Identificadores



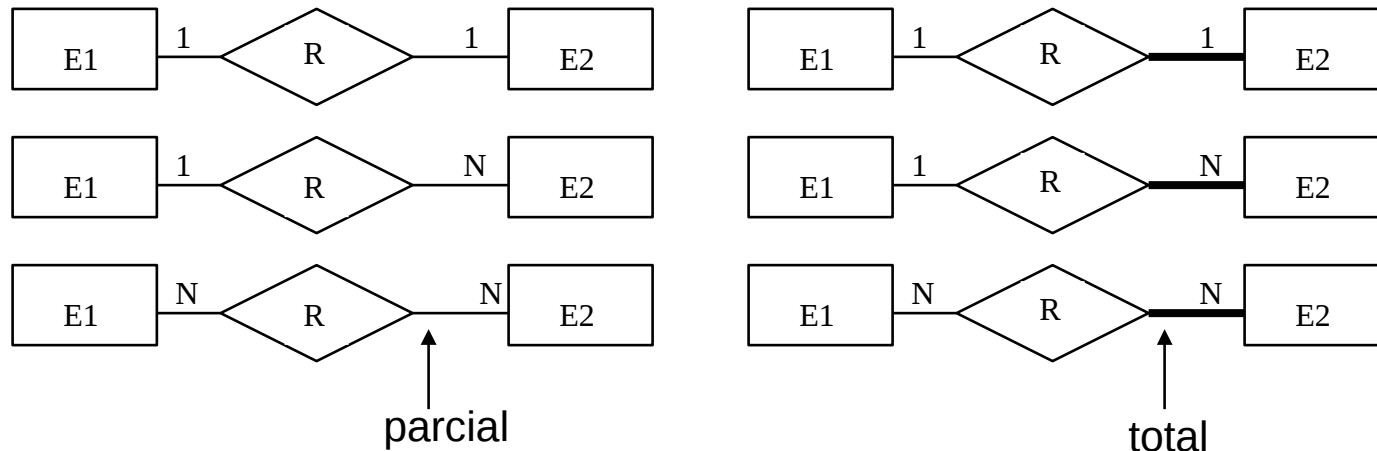
# Relacionamentos Recursivos

- situação em que uma entidade é mãe e filha ao mesmo tempo



# Relacionamentos: Cardinalidade e Participação

- É a propriedade do relacionamento que define exatamente quantas instâncias aparecem na entidade filha para cada instância correspondente na entidade mãe (cardinalidade) e como é sua participação (obrigatoriedade)
- **cardinalidade**: um ou muitos
- **obrigatoriedade**: total (obrigatória) ou parcial (opcional)



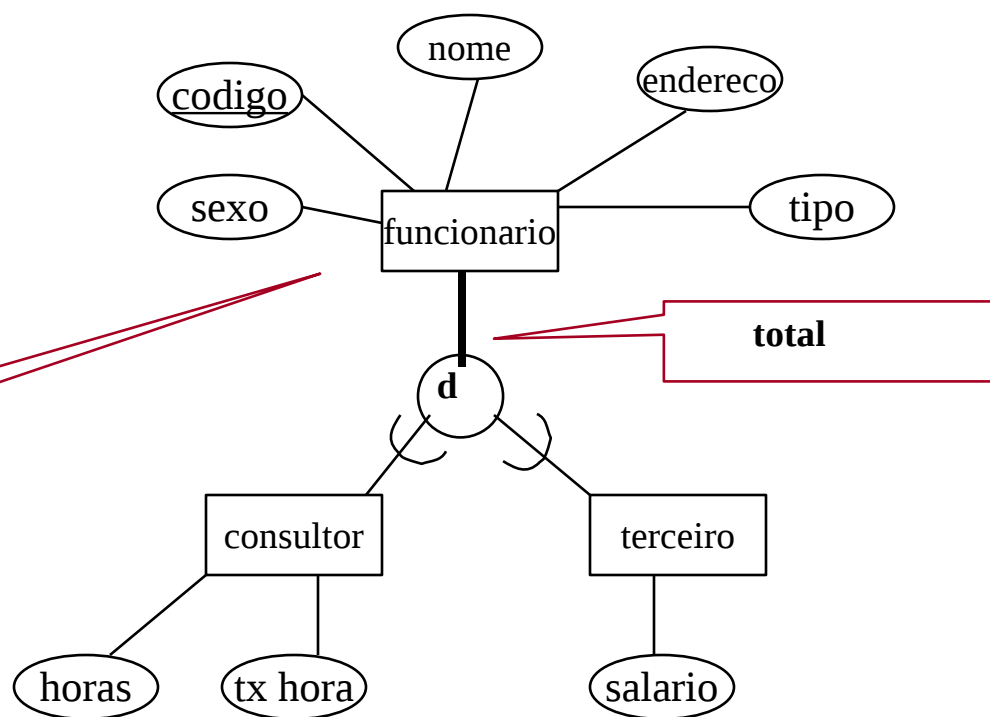
# Herança: **Generalização e Especialização**

- É a forma de agrupar um conjunto de *entidades* que compartilham *características* comuns.

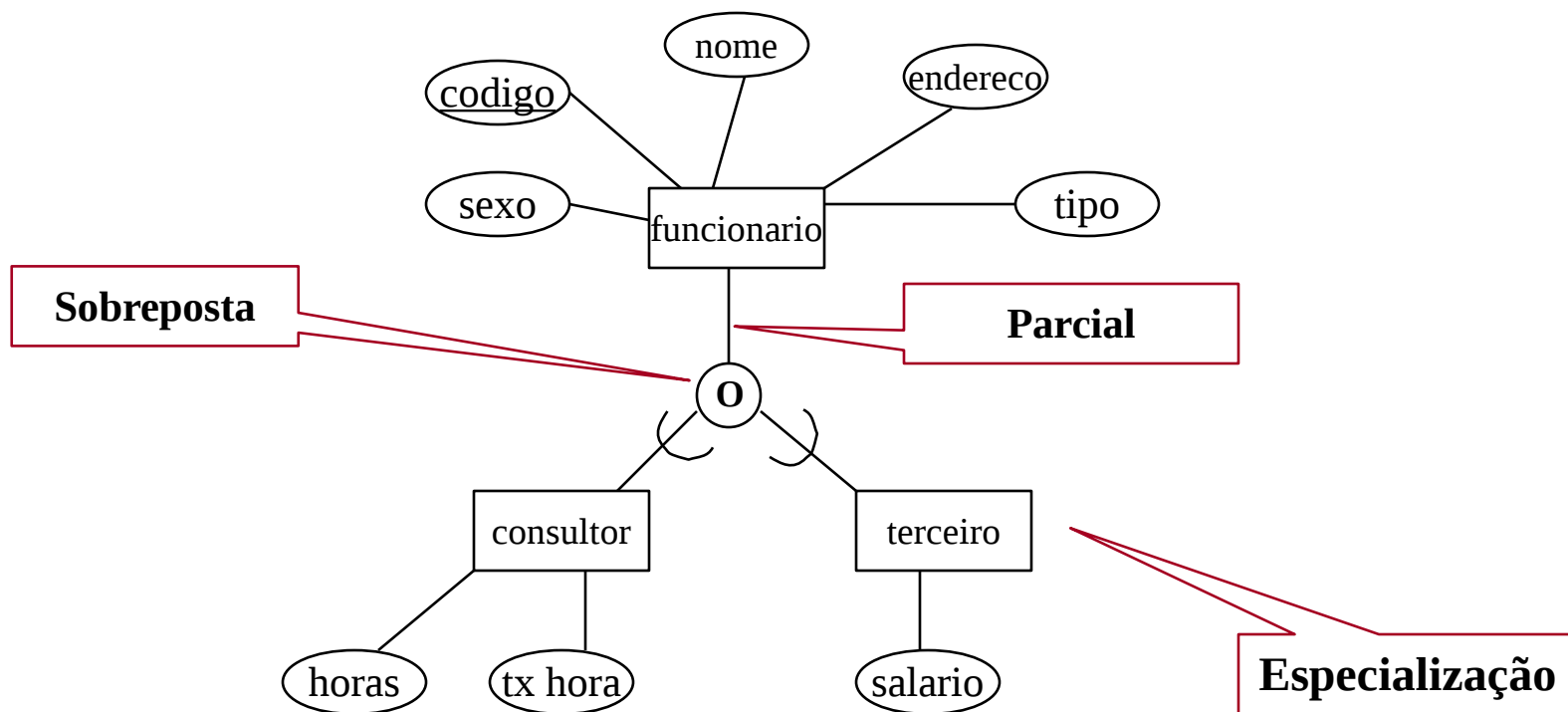
## Restrições:

- **d: mutuamente exclusivo** - quando uma instância da entidade generalização só pode estar em uma entidade de especialização
- **o: sobrepostos** - quando uma instância da entidade generalização pode estar em duas ou mais entidade de especialização
- **total:** cada entidade da generalização deve pertencer a pelo menos uma entidade de especialização
- **parcial:** cada entidade da generalização pode ou não pertencer a uma entidade de especialização

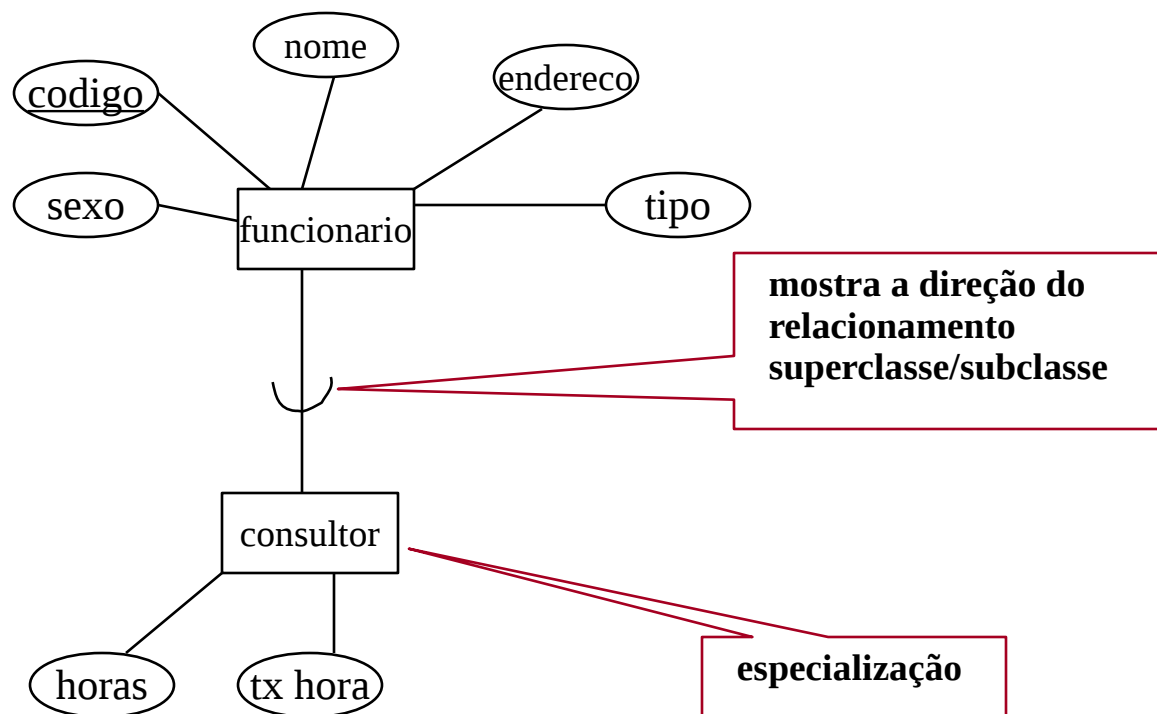
# Herança: Generalização e Especialização



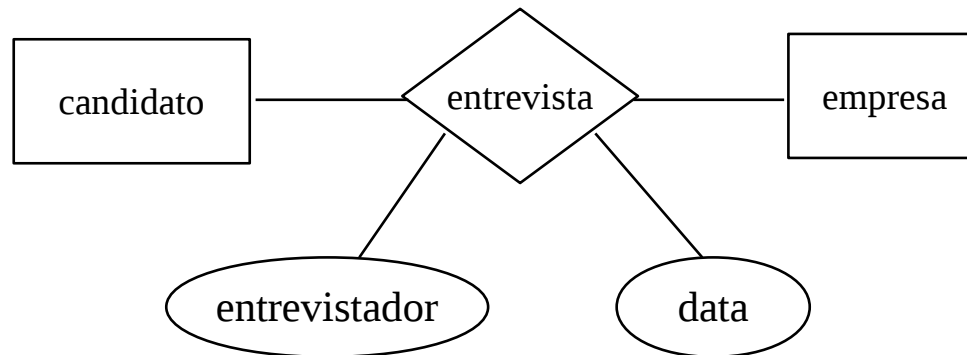
# Herança: Generalização e Especialização



# Herança: Generalização e Especialização



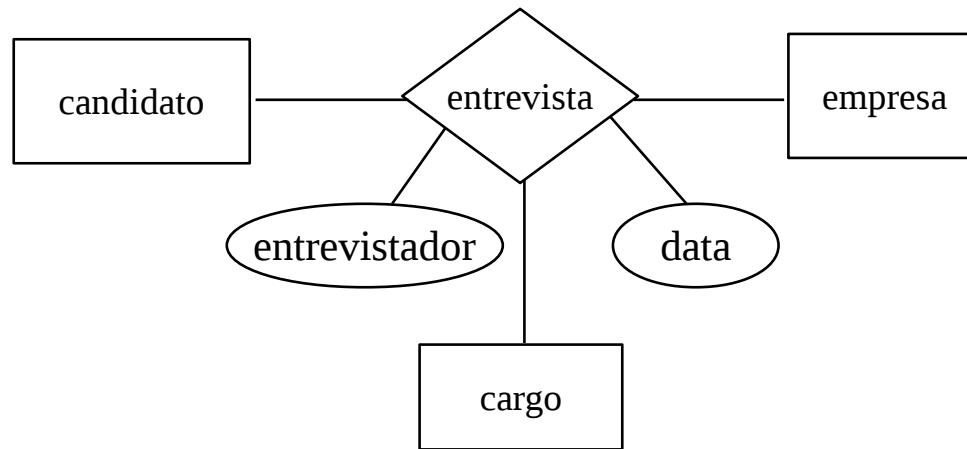
# Aggregação



- **O modelo** mostra um relacionamento binário
  - **entrevista**, onde guarda informações sobre **candidatos** a **cargos** para empresas. Guarda também informações sobre o entrevistador e a data.

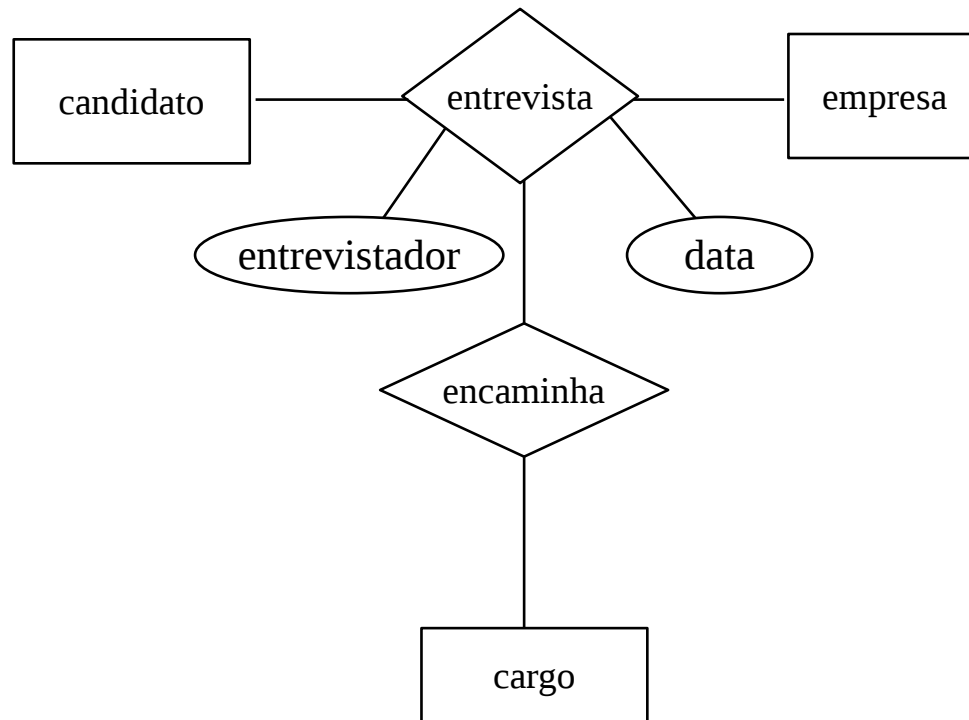


# Agregação



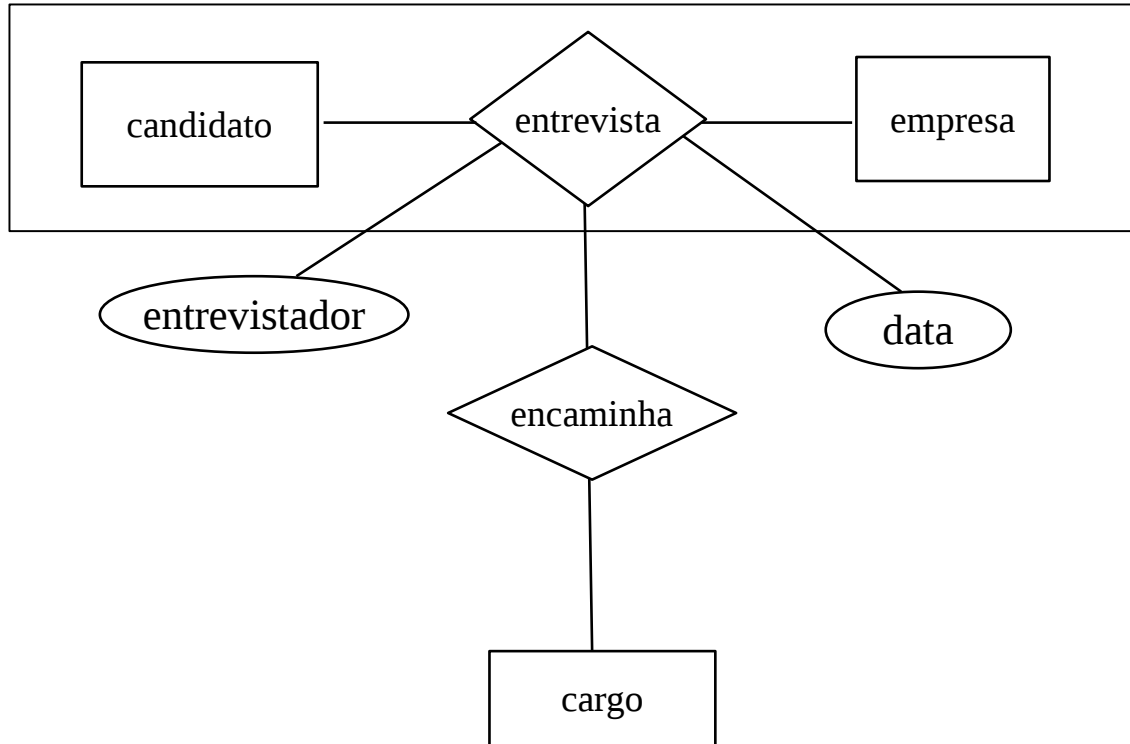
- Suponha que algumas entrevistas resultem em cargos oferecidos e outras não.
- → Este modelo está **incorreto**, pois exige que cada entrevista tenha um cargo oferecido

# Agregação




- O **MER** não permite relacionamento entre relacionamento

# Agregação



- O melhor modo para representar a situação anterior é usando a **agregação**.
- No modelo acima **não existe obrigatoriedade** na entrevista para encaminhar um candidato a um cargo



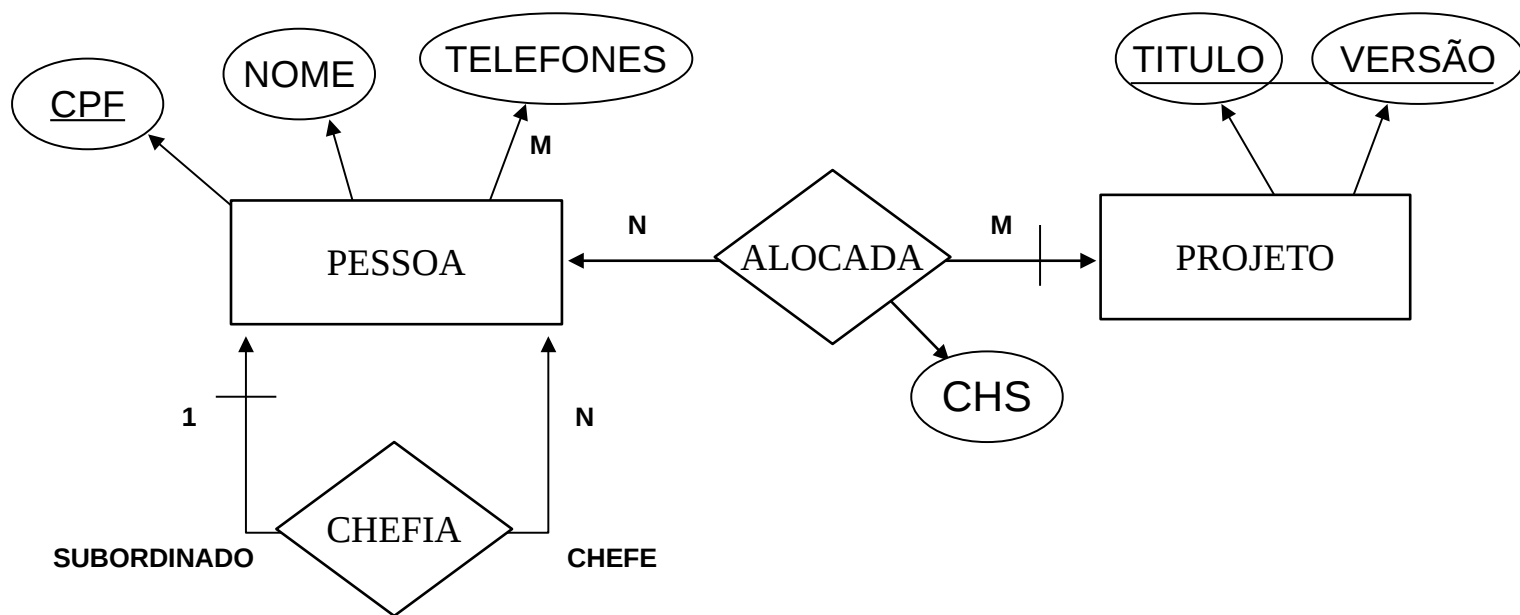
# Outras Notações

## Exemplos de Diagramas ER e ER Estendidos

Banco de Dados

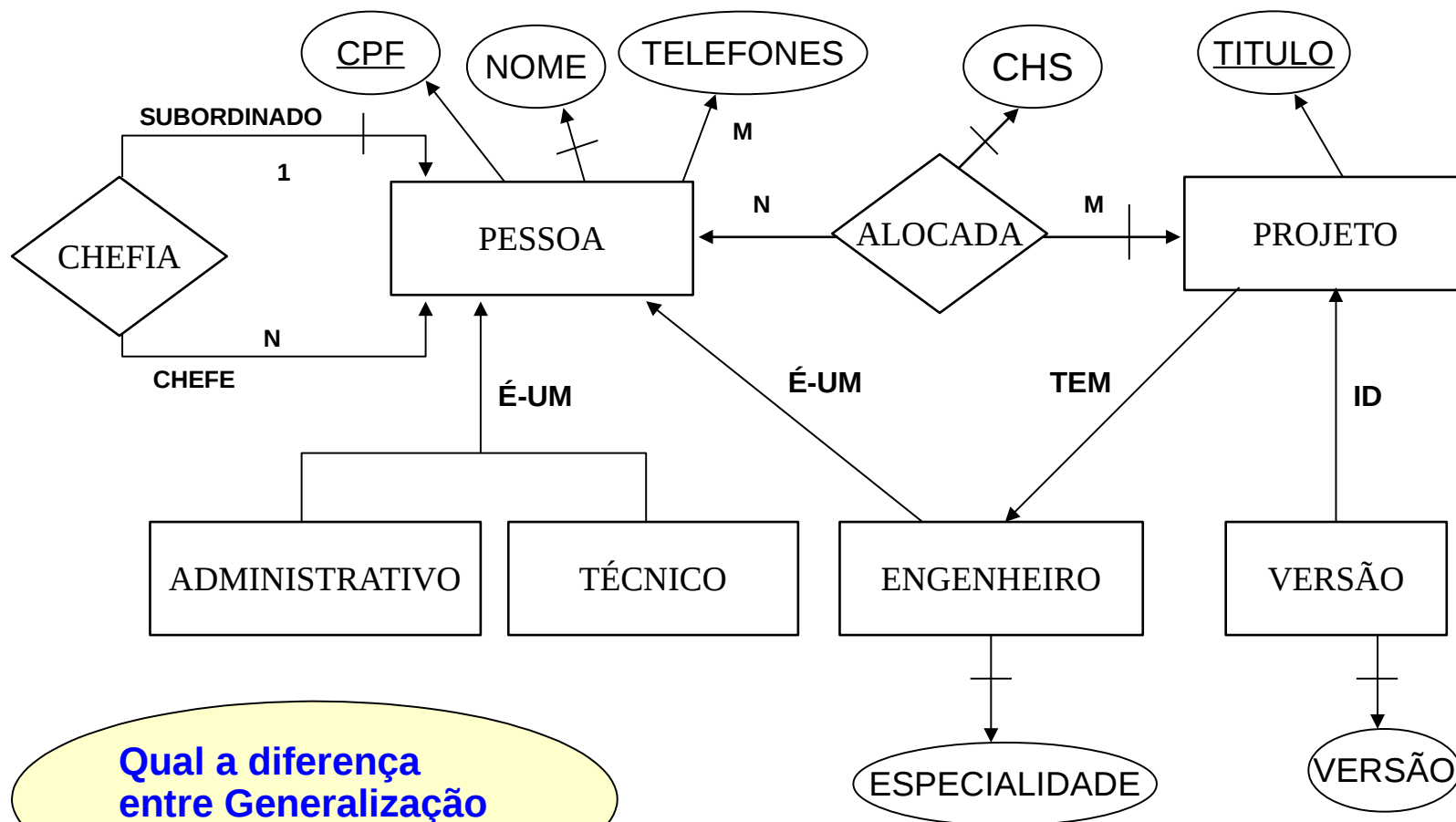
# Diagramas EER

- Diagrama com Notação de Markowitz / Shoshani (1994)



# Diagramas EER

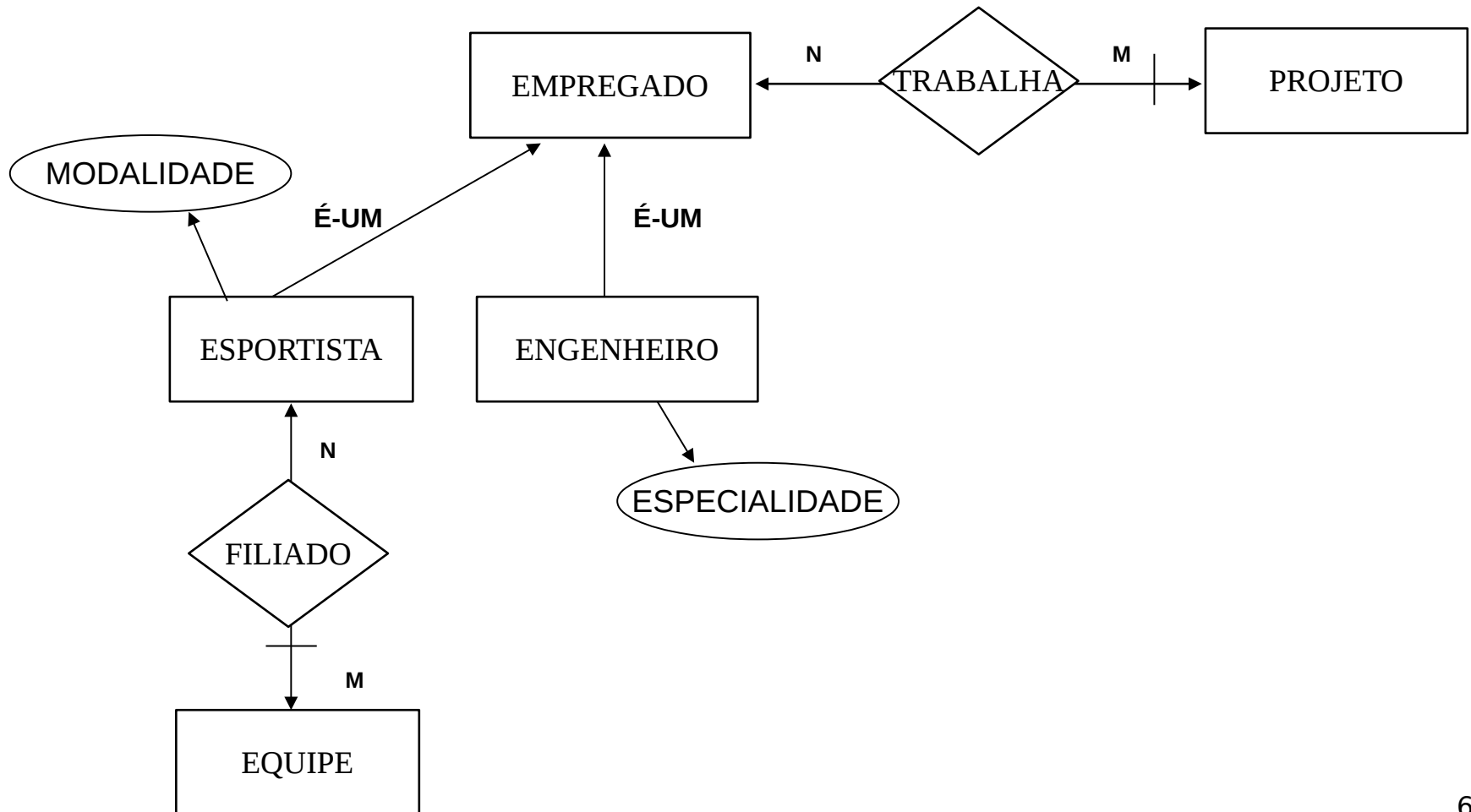
- Vejam estas notações para: **Generalização**, **Especialização** e as Associações **É-UM**, Identificação ou **ID**, e **TEM**



Qual a diferença entre Generalização e Especialização ?

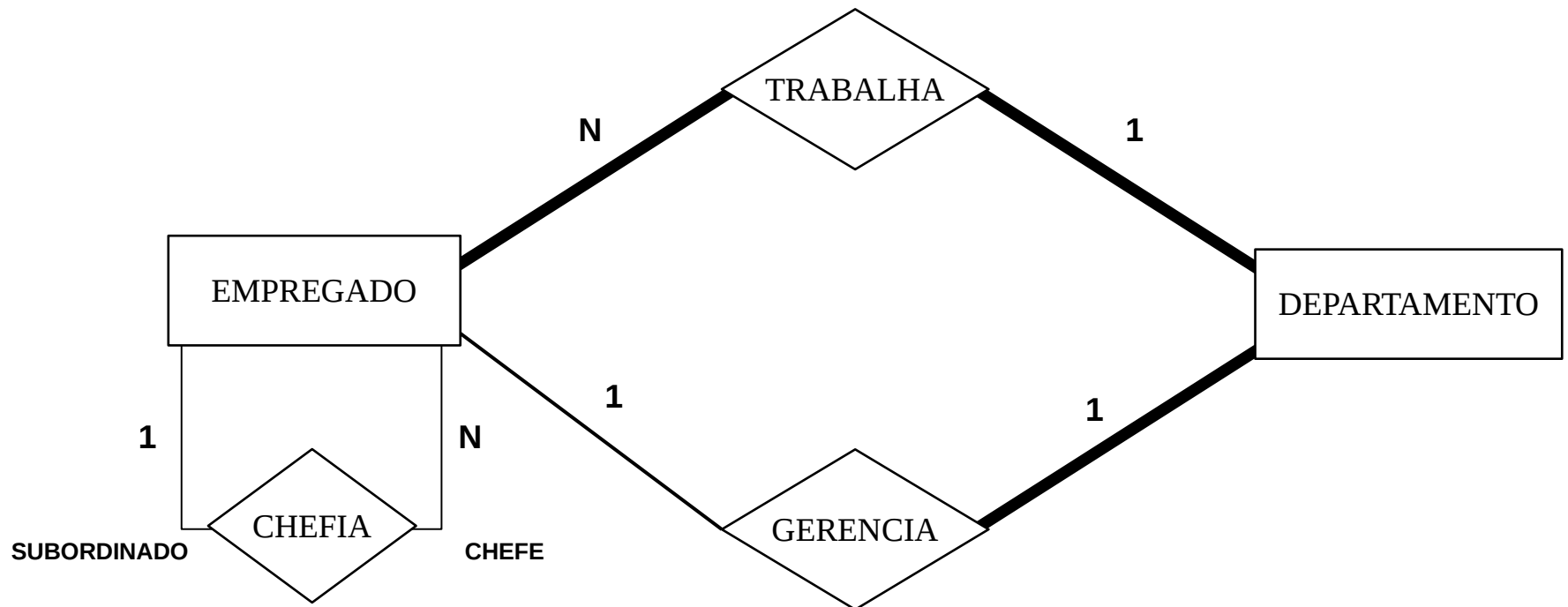
# Diagramas EER

## ■ Especialização



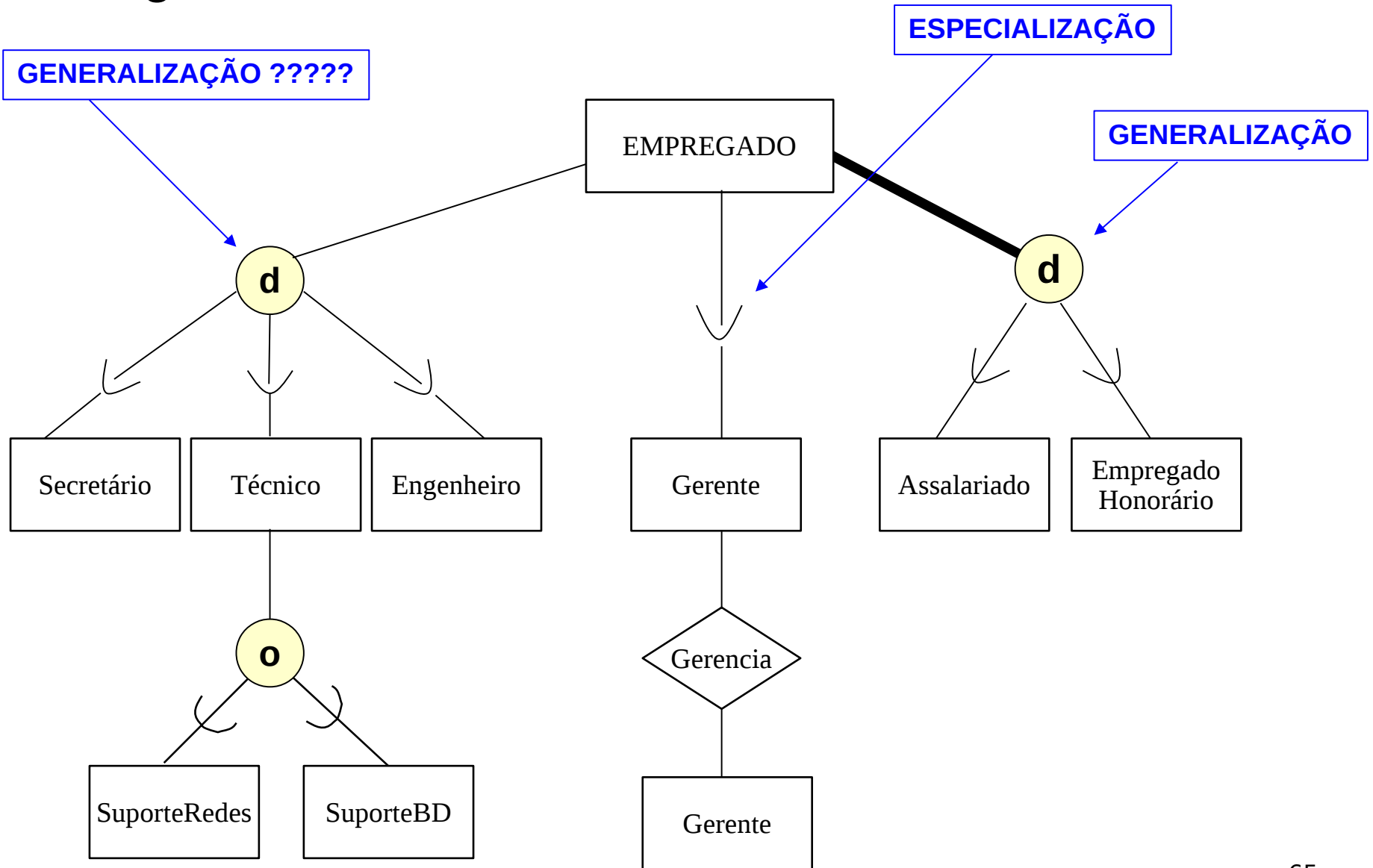
# Diagramas EER

- Diagrama com a Notação do Navathe



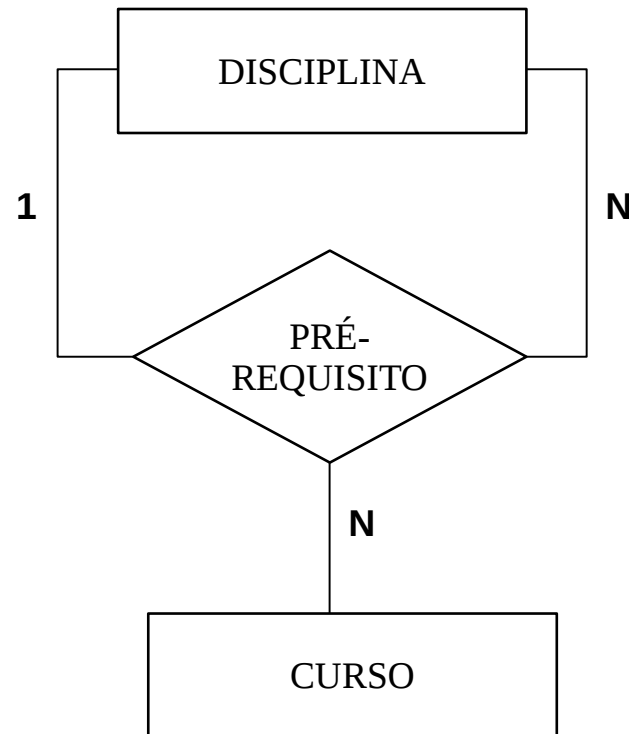


# Diagramas EER



## Vamos analisar este diagrama !!

- O que este DER descreve ?
- Qual o problema deste DER ?



# Metodologia para projeto de Banco de Dados

- requerimentos e análise
- projeto conceitual do banco de dados
  - escolha de um SGBD
    - mapeamento do modelo de dados
    - projeto físico do banco de dados
    - implementação e configuração do BD

# Metodologia para projeto de Banco de Dados

## ■ *Requerimentos e Análise*

- descobrir os atores que irão interagir com o sistema
- ator pode ser um usuário, um sistema, etc.
- são os atores que irão listar as regras de negócio do sistema
- atores podem ser internos ou externos
- peça a cada ator para ter uma explosão de idéias (requerimentos) e fazer uma lista de todos os documentos, caso exista
- neste momento você também descobre alguns requerimentos que o cliente não listou, liste-os

# Metodologia para projeto de Banco de Dados

## ■ *Projeto Conceitual do Banco de Dados – MER*

- O MER do Sistema será montado a partir de cada requerimento de um ator
- Para cada requerimento:
  - Incluir no MER objetos para atender somente a este requerimento
  - Quando o MER estiver sendo montado, é possível que você descubra requerimentos que o ator não listou, liste-os. Isto também deve gerar novos objetos no MER
- Se necessário, defina entidades de apoio: para acumulados, para ganho de performance em consultas, etc

# Metodologia para projeto de Banco de Dados

## ■ *Escolha de um SGBD*

- Custo de aquisição do software: web, interface, linguagem de programação, tipos de dados suportados, etc
- Custo de manutenção
- Custo de aquisição de hardware
- Criação do BD e custo de conversão
- Custo personalizado
- Custo de treinamento
- Custo de operação

# Metodologia para projeto de Banco de Dados

## ■ *Mapeamento do Modelo de Dados*

- **criação do modelo lógico**: mapeamento do projeto conceitual para o projeto lógico
- mapeamento independente do sistema: neste caso o mapeamento não considera nenhuma características específicas do SGBD
- mapeamento dependente do sistema: neste caso o mapeamento leva em consideração as características do SGBD

# Metodologia para projeto de Banco de Dados

## ■ *Projeto Físico do Banco de Dados*

- tempo resposta
- espaço de utilização
- transações



# Metodologia para projeto de Banco de Dados

## ■ *Implementação e Configuração do Banco de Dados*

- Criação dos objetos do banco de dados através de scripts ou carga: tabelas, triggers, views, etc
- Criação de usuários, definição de permissões
- Configurações de parâmetros do banco
- etc

# Estudo de Caso: Locadora de Vídeo

ATORES: cliente, funcionário e o dono

## REQUERIMENTOS:

---

- controlar acervo de filmes
- controlar acervo de fitas
- fazer aluguel de fitas
- fazer devolução de fitas
- consultar fitas em atraso
- consultar previsão de caixa
- cadastro de cliente
- cliente quer consultar seus filmes alugados, seus melhores filmes e os melhores do rank
- cálculos de multas

# Estudo de Caso: Biblioteca

ATORES: aluno, professor, funcionário

## REQUERIMENTOS:

---

- cadastro de alunos, professores e funcionários
- controlar acervo da biblioteca
- fazer empréstimo de itens do acervo
- fazer reserva do acervo
- consultar itens do acervo: título, palavras-chave, etc
- consultar itens em atraso



# Listas de Exercícios Modelo Conceitual - MER

## Banco de Dados

# Modelo Conceitual: **Lista de Exercício – L1**

- definir SGBDs e quais são os objetivos principais?
- identificar as principais vantagens dos SGBDs exemplificando tais vantagens quando comparados aos sistemas tradicionais
- descrever os níveis de abstração de dados
- qual a importância dos modelos de dados para a estrutura de um BD?
- o que você entende por instância e esquema de dados
- qual a função do catálogo em um BD
- o que você entende por: abstração de dados, independência entre dados e programas, natureza auto-contida, acesso concorrente, controle de Acesso, manutenção de restrições
- descrever e exemplificar os objetos do MER
- o que você entende por taxa de Cardinalidade e Participação
- definir e exemplificar entidades fracas e fortes
- definir e exemplificar relacionamentos identificadores e não identificadores

# Modelo Conceitual: **Lista de Exercício – L2**

## Exercícios básicos de MER

### 1. Venda de Produtos

Uma firma vende produtos de limpeza e deseja controlar melhor os produtos que vende, seus clientes e os pedidos. Cada produto é caracterizado por um código único, nome do produto, categoria (ex. detergente, sabão em pó, sabonete, etc), e seu preço. A categoria é uma classificação criada pela própria firma. A firma possui informações sobre todos os seus clientes. Cada cliente é identificado por um código único (interno à firma), o nome do cliente, endereço (rua, nro, sala, cidade, CEP, UF), telefone, status do cliente (bom, médio, ruim), e o seu limite de crédito. Guarda-se igualmente a informação dos pedidos feitos pelos clientes. Cada pedido possui um número (único), e guarda-se a data de elaboração do pedido. Cada pedido pode envolver de 1 a vários produtos, e para cada produto, indica-se a quantidade pedida. Atualmente, a firma usa o formulário a seguir para controle de pedidos, preenchido a título de exemplo. As demais informações são hoje mantidas pelos vendedores em listas de papel.

# Modelo Conceitual: **Lista de Exercício – L2**

## Exercícios básicos de MER

### 1. Venda de Produtos (cont.)

Limpex S.A.		Controle Interno	
Comendador Oliveira, 27		Pedido: 98765	
CGC: 7654321/09		Data: 27/03/2002	
Código: C-1234			
Nome: João da Silva			
End: Anita Garibaldi, 8765, Porto Alegre, RS, 90345-678			
Telefone: (051) 234-5678			
Cod	Desc	Qtde	Total
123	Limpa Tudo	1	10,00
345	Detergentex	2	4,00
678	Escovex	3	3
TOTAL			17,00

# Modelo Conceitual: **Lista de Exercício – L2**

## Exercícios básicos de MER

### Hollywood

Hollywood possui diversos estúdios cinematográficos, cada um caracterizado por um nome único, um dono, data de fundação e o faturamento do ano anterior. Estes estúdios produzem filmes que possuem um nome único, o número de meses que levou sendo feito, o ano de lançamento, o número do "copyright" e o custo total do filme. Em cada filme atuam atores, que possuem um nome artístico único, um número de seguro social (também único), uma nacionalidade, idade, sexo, e um conjunto de tipos de papéis para o qual seu tipo físico é aconselhável (ex: avó, mocinha jovem, galã com idade avançada, adolescente). Estes tipos de papéis não são pré-definidos, constituindo uma lista preenchida a critério de cada ator. Em cada filme onde atua, um ator ganha um cachê, e desempenha um personagem que possui um nome. Estúdios podem existir mesmo que ainda não tiverem produzido um filme, mas só são considerados atores os que já atuaram em pelo menos um filme.



# Modelo Conceitual: **Lista de Exercício – L2**

## Exercícios básicos de MER

### Biblioteca

O acervo de uma biblioteca é composto por exemplares de livros. Cada livro é caracterizado por um ou mais autores, um título, uma editora, local de edição, um código ISBN e um conjunto de palavras-chave. A biblioteca possui pelo menos um exemplar de cada livro, numerados seqüencialmente (exemplares 1, 2, 3, etc).

Os associados da biblioteca podem retirar exemplares dos livros. Cada associado pode ter emprestado no máximo três exemplares. Para cada empréstimo, é registrada a data em que este foi realizado. Cada associado possui um código, nome e endereço.

**Varição 1:** A biblioteca deseja manter registro somente dos empréstimos correntes (ou seja, ainda não devolvidos).

**Varição 2:** A biblioteca deseja manter todo o histórico de empréstimos.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Aeroclube

Num aeroclube, estão inscritos pilotos, instrutores e alunos de pilotagem. Todos sócios (inscritos) são identificados pelo número de matrícula, e caracterizados por nome, endereço e idade. Os pilotos possuem um número de brevê (único). Os instrutores são pilotos com formação adicional de instrutor, e deve ser registrado o nome do curso, a data de obtenção do diploma, bem como a instituição.

Para os alunos de pilotagem, guarda-se o registros de todas suas saídas para contabilização de horas para obtenção do brevê. Para cada saída registra-se a data, instrutor, hora de saída e de chegada, bem como o parecer do instrutor sobre o vôo. A escola só ministra cursos básicos, e portanto não há professores que são alunos de cursos avançados. Para emissão do brevê, é necessário que o aluno comprove ter o número de horas mínimo de vôo, bem como apresente os pareceres dos instrutores sobre as habilidades desenvolvidas a cada aula prática.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Receitas

Uma empresa deseja informatizar o acervo de receitas que ela inventa, e comercializa sob a forma de livros. Estão envolvidos na elaboração das receitas e dos livros os cozinheiros, os degustadores (que controlam a qualidade das receitas), e os editores dos livros. Todas estas pessoas são empregados da empresa, e são caracterizados por um RG, nome, data de ingresso na firma, e salário recebido.

Cada receita tem código único, um nome, foi inventada por um cozinheiro numa dada data, e pertence a uma categoria. Podem existir diferentes receitas com o mesmo nome, mas um mesmo cozinheiro não elabora duas receitas com o mesmo nome. Uma categoria é extraída de uma lista de categorias fixas elaborada pela empresa (ex: carne, ave, bolo, torta, sopa, etc). É norma da empresa não permitir a elaboração de receitas que não pertençam a categorias registradas pela firma, sendo possível não existirem receitas para categorias recém criadas.

Diversos ingredientes (ex: açúcar, farinha, leite) são usados para elaborar uma receita, cada um deles usado numa certa quantidade (ex: 2) e numa certa medida (ex: colher de chá, xícara, ml). A medida pode ser opcional para certos ingredientes (ex: ovo). Uma receita possui também uma descrição de seu modo de preparação, e o número de porções que rende.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Receitas (cont.)

Cada ingrediente possui um nome único e uma descrição, particularmente útil para ingredientes exóticos tais como blachan, kiri ou umeboshi.

Os cozinheiros renomados podem, para efeito de publicidade dos livros, fornecer um nome fantasia, bem como uma lista de restaurantes importantes nos quais já trabalhou. Esta lista é fornecida pelo cozinheiro no momento de sua contratação, sendo que nenhum cozinheiro informá-la. Todo cozinheiro deve produzir um certo número de receitas por mês, sendo que os cozinheiros recém-contratados têm um prazo de até 45 dias para entregar suas primeiras receitas.

Receitas podem ser testadas por degustadores. Cada teste envolve um degustador, é executado numa data, e envolve a atribuição de uma nota. Podem existir receitas sem teste, mas todo degustador contratado pela firma já executou pelo menos um teste.

A empresa edita livros de receitas, nos quais, obviamente, constam diversas receitas. Essas receitas podem ou não ser inéditas, ou seja, já terem sido publicadas em outros livros. Além de suas receitas, cada livro é caracterizado por um título único, um código ISBN (também único), e pelo editor do livro.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Programa de Milhagem

A companhia aérea WARIGUI deseja oferecer um programa de milhagens a todo passageiro seu que o solicitar. Com estas milhas, passageiros podem usufruir de várias promoções, como vôos gratuitos, upgrade de classe, descontos em redes de hotéis credenciados, etc. Para boa gestão do programa, ela deseja um sistema de informação que controle o crédito das milhas e a emissão de certificados para usufruto dos benefícios.

Um passageiro é admitido no programa quando este encaminha à companhia uma solicitação com dados cadastrais (nome, endereço, cpf, rg, telefone(s) de contato, profissão, renda mensal e opcionalmente tipos de cartão de crédito que já possui - VISA, MASTERCARD, etc), junto com pelo menos um bilhete de avião acompanhado do respectivo cartão de embarque usado, comprovante de renda, e cópia dos documentos de identidade (RG, CPF). A companhia analisa a solicitação, e se positiva, atribui a este cliente um número de cartão SORRISO (único), cadastrando o passageiro como cliente. Ela também fabrica e envia o cartão ao cliente, que deve apresentar o cartão ou informar seu número para obtenção de créditos. Ao cadastrar o cliente, já são atribuídos seu(s) primeiro(s) crédito(s) no programa de milhagens. Um mesmo passageiro não pode ter dois cartões SORRISO, e esta verificação é feita através dos documentos de identificação fornecidos.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Programa de Milhagem (cont.)

As solicitações indeferidas são descartadas, e não interessam ao sistema, isto é, o sistema somente gerencia clientes do programa SORRISO.

Todo crédito tem um número, único entre os créditos de um mesmo cliente (isto é, dois clientes distintos podem ter créditos com o mesmo número). Além do número do crédito, devem ser registrados a data do crédito, o número de milhas creditadas, e opcionalmente as milhas bônus, atribuídas somente em promoções. Estas informações servem para emissão de correspondência para informe de crédito de milhas e milhas acumuladas, bem como para emissão de certificados de milhagem. Todo o crédito já utilizado para emissão de certificado deve ser marcado com esta informação.

O programa SORRISO de milhagens atualmente credita créditos por três tipos de serviço: vôos da companhia e das companhias associadas, compra de produtos em estabelecimentos credenciados, e hospedagem em hotéis credenciados.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Programa de Milhagem (cont.)

Para crédito de milhas, a WARIGUI necessita saber o código do voo (e.g. RG230, TR450), o trecho voado pelo passageiro daquele voo (origem e destino), a classe utilizada, e a data de partida do passageiro. Todos os trechos (combinação de origem e destino) devem estar cadastrados no sistema junto com a milhagem correspondente, independentemente dos voos que servem o trecho (e.g. o voo de Porto Alegre a São Paulo credita 900 milhas). A WARIGUI não se interessa através deste sistema controlar os voos por ela oferecidos, nem por suas companhias associadas: somente as milhagens correspondentes a trechos, e os voos realizados por clientes que dão origem a créditos. Os créditos podem ser atribuídos na recepção do aeroporto por ocasião do embarque, ou posteriormente, quando o cliente envia os comprovantes necessários (passagem e cartão de embarque).

Para créditos obtidos através de compras ou de hospedagem, o sistema necessita saber dados sobre o estabelecimento conveniado, tais como nome, código do convênio (único), endereço para correspondência (rua, nro, complemento - opcional, cidade, estado, país, código postal), percentagem de milhas sobre valor de consumo no estabelecimento acertado no convênio.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Programa de Milhagem (cont.)

Para o crédito é necessário, além do estabelecimento, a data da compra ou hospedagem, valor comprado ou do total de diárias, e uma descrição do consumo efetuado (e.g. compra de jóia, estadia fim de semana, compra de coleção em promoção). Para obtenção deste crédito, o cliente deve enviar a gestão do programa SORRISO a nota fiscal comprovando o consumo (compra ou estadia), e o número desta é armazenado junto com o crédito.

Todo o mês, a WARIGUI emite avisos de créditos para aqueles clientes com novos créditos no mês. Ela também verifica se a soma dos créditos não usados pode dar origem a um ou mais certificados. Em caso positivo, os créditos são marcados como usados, e os certificados de milhagem são emitidos. Cada certificado tem um número único, é pessoal de um cliente do programa, e possui uma data de emissão e uma de validade. Quando o cliente usa o certificado para obtenção de benefícios, os certificados são marcados como usados.

A WARIGUI deseja guardar todas as informações sobre créditos e certificados já atribuídos/emitidos a seus clientes, pois deseja analisar como o programa de milhagem está funcionando, e fazer modificações conforme necessidade.



# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Evento científico

O CBBD é o maior evento científico anual no Brasil na área de banco de dados. A cada ano, sua organização fica a cargo de alguma universidade brasileira, que fica encarregada de controlar todos os aspectos científicos e operacionais do evento. Sabendo da dificuldade desta complexa tarefa, e que o congresso tem atraído um número cada vez maior de interessados, os organizadores do ano que vem resolveram se antecipar, e projetar um sistema de informação que os auxilie na melhor organização do evento. O sistema deve dar apoio a todo o processo de seleção de trabalhos científicos, submetidos pelos membros da comunidade. Ele também deve numa etapa ulterior auxiliar a gestão da organização, mantendo informações sobre toda a programação e os participantes do evento. O evento científico começa com a designação de um comitê de programa (CP), que tem como função avaliar os artigos científicos submetidos, selecionando os melhores. Para cada membro do CP (revisor), registra-se seu nome (único), a instituição (há no máximo um representante por instituição), suas áreas de especialidade para revisão dos trabalhos (e.g. data warehouse, data mining, bancos de dados orientados a objetos), seu endereço eletrônico (único), e coordenadas para contato (endereço regular, telefone, fax).

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Evento científico (cont.)

Alguns meses depois, os artigos começam a ser recebidos, e cada artigo deve ser cadastrado. Para cada artigo gerar-se-á um código único, que o identificará ao longo de todo o processo de avaliação. Além disso, devem ser cadastrados o título, seus autores, a instituição de cada um de seus autores, o endereço eletrônico do primeiro autor, e as palavras-chaves. Tanto as palavras-chaves quanto as especialidades dos revisores são itens de uma lista de assunto divulgada junto à chamada de trabalhos.

Encerrado o prazo para submissão de artigos, o presidente atribui a cada artigo 3 revisores, e envia-os para avaliação. Os avaliadores têm um prazo para lê-los, e atribuir uma nota ao artigo. Como os revisores sempre se atrasam, é imperativo saber quem está com que artigo para revisar, para poder cobrar os pareceres na época adequada. Os artigos com melhores notas são selecionados, e devem ser enviados e-mails para o primeiro autor dos artigos selecionados e dos não selecionados para comunicação do resultado. Os autores dos artigos aceitos, com base no parecer, fazem modificações e enviam a versão final de seu artigo, usando um formato eletrônico pré-definido. Deseja-se saber quem já mandou a versão final, e qual o nome do respectivo arquivo.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Evento científico (cont.)

Começam então os preparativos para a organização do evento. A universidade realizará o evento em suas dependências, e já reservou uma série de salas e auditórios para este fim. Cada local é identificado por seu nome, e descrito pela sua capacidade. O evento contará com sessões técnicas (onde serão apresentados os trabalhos aceitos), bem como palestras convidadas e minicursos, ministrados por cientistas de renome nacional e internacional. Estes 3 tipos de atividades compõem o programa da conferência. Cada atividade ocorre em um local em uma dada data/hora, sendo que algumas atividades podem ocorrer em paralelo. Obviamente, não há duas atividades iniciando no mesmo local ao mesmo tempo. Para montar o programa, os organizadores atribuem as atividades aos locais, determinando horário de início e fim, quais recursos devem estar disponíveis (ex. retroprojetor, canhão, computador), e quem é o responsável pela coordenação da atividade. Os recursos e o responsável podem ser informados posteriormente à definição da atividade.

Cada sessão técnica tem um nome único (data warehouse I, KDD II), e é descrita pelos artigos que serão nela apresentados, e em que ordem (1, 2, etc). Cada artigo é apresentado uma única vez.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Evento científico (cont.)

As palestras e minicursos possuem um título, um autor, uma instituição, e além destas informações técnicas, devem ser informados, quando sabido, a data/hora de chegada e partida, e as informações dos vôos de ida e volta (companhia, código de cada vôo), já que, por serem pessoas mais importantes, é de bom tom buscá-los e levá-los ao aeroporto. A diferença entre um minicurso e uma palestra convidada é que na primeira os participantes devem se inscrever e pagar uma taxa. A taxa de cada curso deve ser mantida junto à descrição do curso, bem como o número de vagas disponíveis e as pessoas inscritas. As informações sobre a programação são muito importantes, pois é a partir delas que são divulgadas as atividades do evento (e.g. página web, livreto, cartazes, etc).

Por fim, a organização quer poder controlar os inscritos. Cada participante possui um nome, instituição (opcional), endereço, telefone, e-mail, categoria (sócio, não sócio, estudante, estudante não sócio, já que há descontos para sócios da SBC). Pelo menos um autor de cada artigo selecionado deve se inscrever no evento até uma dada ocasião (divulgada aos autores), condição necessária de publicação do artigo nos anais da conferência. O sistema deve permitir o registro da inscrição de um autor associada ao(s) seu(s) respectivo(s) artigo(s).

# Modelo Conceitual: **Lista de Exercício – L3**

Exercícios avançados de MER

Evento científico (cont.)

Também, porque são cobrados e porque são distribuídas apostilas, deseja-se saber em qual(uais) minicursos um participante eventualmente se inscreveu. Com estas informações são gerados os crachás, é definido o número de cópias das apostilas de cada minicurso bem como a quem devem ser distribuídas, além dos certificados de participação no evento e nos minicursos

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Eletrotécnica

Uma eletrotécnica tem enfrentado problemas para atender a demanda de serviço de consertos que tem recebido. Os clientes reclamam de atrasos freqüentes na entrega prevista dos consertos, e estima-se que esta é devida a má previsão dos recursos humanos e materiais para realização de consertos.

Cada vez que um cliente traz um aparelho para consertar é aberta uma ordem de serviço (uma por aparelho). Esta ordem de serviço diz respeito a um cliente, que deixa seu nome, endereço, e um ou mais números de telefone para contato. Um mesmo cliente pode ter outras ordens de serviço suas em andamento, e a empresa mantém um cadastro de todos os seus clientes. Na ordem de serviço constam ainda a data de recebimento do aparelho, a marca e número de série do aparelho, a descrição do defeito, e a data de previsão da entrega do aparelho ao cliente, e a data na qual foi efetivamente retirado.

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Eletrotécnica (cont.)

Para resolver o problema dos atrasos, a firma optou por designar um técnico experiente como gerente. Este gerente, inicia o dia examinando todas as ordens de serviço novas, designando então um conjunto de técnicos, e para cada técnico designado a este conserto, as ferramentas/aparelhos de teste necessários para que este detecte o defeito e/ou teste (a parte do) o conserto realizado (sempre há pelo menos um aparelho). A previsão dos recursos materiais também é importante pois o número de ferramentas/aparelhos de teste é limitado. Cada técnico pode estar associado a várias ordens de serviço. Cada aparelho de teste possui um número de série único, e é caracterizado por uma descrição. Os técnicos são caracterizados por seu nome, endereço, eventualmente número(s) de telefone para contato, e contrato de trabalho (número único).

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Cinema

Um cinema possui várias salas de cinema, as quais exibem filmes em horários diversos. O cinema tem interesse em saber quais filmes estão atualmente em cartaz, em que salas e em que horários.

Cada sala possui um nome (único) e capacidade (número de lugares). Os filmes são caracterizados por seu nome em português, nome na língua original (se estrangeiro), diretor, ano de lançamento, tipo, e sinopse. Não existem dois filmes com o mesmo nome (em português) e ano de lançamento. Eventualmente, podem existir para o filme premiações ou indicações para premiação (e.g. Palma de Ouro em 1987, Oscar de melhor atriz em 89, indicado para melhor filme estrangeiro em 1996), e esta informação é usada para divulgação dos filmes. Uma exibição de filme ocorre em uma dada sala e horário. Um mesmo filme pode ser exibido na mesma sala, em vários horários. Para filmes muito procurados, o cinema pode ter exibição simultâneas em várias salas (em horários simultâneos ou não). Filmes diferentes podem passar na mesma sala, desde que obviamente não no mesmo horário.



# Modelo Conceitual: **Lista de Exercício – L3**

Exercícios avançados de MER

Cinema (cont.)

O cinema só trabalha com horários fixos de filmes, os quais atualmente são: 16:00, 17:00, 18:00, 19:30, 20:00, 22:00, 24:00. A cada um destes horários está vinculado um conjunto de funcionários responsáveis pelo bom andamento das atividades do cinema naquele horário, e que desempenham uma função (ex: caixa, balas, lanterninha, bilheteiro). Cada funcionário é caracterizado pelo número da carteira da trabalho (único), nome, data de admissão e salário. Para maior satisfação dos funcionários, existe um rodízio das funções conforme o horário (ex: um mesmo funcionário pode ser caixa no horário das 16:00, e baleiro no horário das 21:00). Todo horário tem pelo menos três funcionários alocados

# Modelo Conceitual: **Lista de Exercício – L3**

Exercícios avançados de MER

Cinema (variação)

Para aumentar a renda do cinema, em cada sessão são exibidas propagandas. Uma propaganda é identificada por um código, e caracterizada por um nome, agência, e faixa etária apropriada, e pode ser exibida em várias sessões. Cada sessão possui sua própria programação de propagandas (e.g. a sessão do Titanic das 14:00 horas na sala 1 pode ou não ter o mesmo conjunto de propagandas que a sessão do Titanic em outro horário e/ou sala).

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Academia

Uma academia oferece várias opções de prática de esporte a seus clientes. Para este fim, ela conta com uma equipe de professores, e com uma infraestrutura de locais adequados para práticas esportivas diversas.

Os clientes são caracterizados pelo RG, nome, endereço, e um ou mais telefones de contato. Para aqueles clientes que freqüentam (já freqüentaram) aulas ou musculação, é mantido o histórico dos contratos já firmados por estes alunos. Para cada contrato (novo ou renovação), são registrados a data de início do contrato (dia/mês/ano), data de fim (dia/mês/ano), valor total do contrato, e forma de pagamento (e.g. pré-datado, à vista, parcelado em 3 vezes, etc). Um mesmo aluno não possui dois contratos que iniciam na mesma data. As formas de pagamento são estabelecidas em função do momento econômico e do aluno, não sendo possível pré-defini-las. O sistema não é responsável pela manutenção do controle do pagamento das mensalidades do contrato.

Cada professor possui um RG e um número de carteira de trabalho (ambos são únicos), e é caracterizado por um nome, endereço, um ou mais telefones de contato, e salário. Cada professor está apto a orientar pelo menos uma modalidade esportiva (e.g. vôlei, basquete, aeróbica, musculação, etc).

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Academia (cont.)

Eventualmente o professor tem uma qualificação específica para a orientação de uma dada modalidade (e.g. mestrado em voleibol pela ESEF, especialização em capoeira com Mestre Dadá, etc), podendo possuir mais de uma (e.g. mestrado em voleibol pela ESEF e especialização em voleibol para terceira idade). Ao ser contratado, o professor informa todas as modalidades que pode orientar, e sua qualificação (ou qualificações) para cada uma delas, caso exista(m). Uma mesma modalidade pode ser orientada por um ou mais vários professores, ou mesmo por nenhum (por exemplo, as aulas de aeróbica são orientadas por um professor, mas paddle pode ser praticado por clientes da academia que simplesmente alugam as canchas).

A academia oferece um elenco de modalidades esportivas, de acordo com a infraestrutura existente. A qualquer momento, a academia quer poder saber quais modalidades esportivas oferece, quais professores podem atuar em quais modalidade esportivas, e quais aulas estão previstas para uma mesma modalidade (eventualmente, com seus respectivos professores).

# Modelo Conceitual: **Lista de Exercício – L3**

## Exercícios avançados de MER

### Academia (cont.)

Para a prática de cada modalidade, a academia oferece espaços adequados, na forma de salas ou canchas. Cada espaço é identificado por código, e caracterizado pelo andar no qual se encontra (e.g. térreo, primeiro, subsolo) e eventualmente um número de sala. As canchas podem ser locadas pelos clientes ou pelos professores, numa dada data e horário. O sistema é responsável apenas por manter as reservas já feitas (e não as disponibilidades futuras). Cada aula é realizada em uma dada sala, é referente a uma modalidade, e orientada por um professor habilitado para aquela modalidade. A frequência dos alunos nas aulas é livre (i.e. não necessita de inscrição em determinada aula). Cada aula tem um horário de início e duração.

As canchas podem ser locadas pelos clientes ou pelos professores, numa dada data e horário. As aulas são realizadas em uma dada sala, e orientadas por um professor em uma modalidade na qual ele atua. Cada aula tem um horário de início, duração, e eventualmente limite de alunos

# Modelo Conceitual: **Lista de Exercício – L4**

## Reserva de passagens aéreas

O objetivo é projetar um sistema de reservas para uma companhia de aviação. O sistema contará com um banco de dados central, que será acessado por aplicações clientes, rodando tanto dentro da própria companhia, quanto fora dela. A transação central do sistema é a reserva. Uma reserva é identificada por um código gerado pelo sistema em computador. A reserva é feita para um único passageiro, do qual se conhece apenas o nome. A reserva compreende um conjunto de trechos de vôos, que acontecerão em determinada data e hora. Para cada trecho, a reserva é feita em uma classe (econômica, executiva, etc.). Um vôo é identificado por um código e possui uma origem e um destino. Por exemplo, o vôo 595 sai de Porto Alegre, com destino a São Paulo. Um vôo é composto de vários trechos, correspondendo às escalas intermediárias do vôo. Por exemplo, o vôo 595 é composto de dois trechos, um de Porto Alegre a Londrina, o outro de Londrina a São Paulo. Cabe salientar que há cidades que são servidas por vários aeroportos. Por isso, é importante informar ao passageiro que faz a reserva, qual é o aeroporto no qual o vôo passa. Às vezes os clientes, ao fazer a reserva, desejam saber qual é o tipo de aeronave que será utilizada em determinado trecho do vôo. Alguns poucos vôos, principalmente internacionais, têm troca de aeronave em determinadas escalas. Nem todos os vôos operam em todos os dias da semana. Inclusive, certos vôos têm pequenas mudanças de horário em certos dias da semana. Cada reserva possui um prazo de validade. Caso os bilhetes não tenham sido emitidos, até esgotarse o prazo da reserva, a mesma é cancelada. Reservas podem ser prorrogadas. Como o "check-in" de todos os vôos está informatizado, a companhia possibilita a reserva de assento para o passageiro. Reservas de assento podem ser feitas com até 6 meses de antecedência. Além de efetivar reservas, o sistema deve servir para vários tipos de consultas que os clientes podem querer fazer: a) possibilidades de viagem de uma cidade ou de um aeroporto para o outro; b) o mesmo, mas restrito a determinados dias da semana; c) horários de chegada ou de saída em determinados vôos; d) disponibilidade de vagas em um trecho de vôo; e) disponibilidade de determinados assentos em um trecho de vôo.

# Modelo Conceitual: **Lista de Exercício – L4**

## **Vídeo Locadora**

Uma pequena locadora de vídeo possui ao redor de 2.000 fitas de vídeo, cujo empréstimo deve ser controlado. Cada fita possui um número de identificação. Para cada filme, é necessário saber seu título e sua categoria (comédia, drama, aventura, ...). Cada filme recebe um identificador próprio. Para cada fita é controlado que filme ela contém. Para cada filme há pelo menos uma fita, e cada fita contém somente um filme. Alguns poucos filmes necessitam duas fitas. Os clientes podem desejar encontrar os filmes estrelados por seu ator predileto. Por isso, é necessário manter a informação dos atores que estrelam em cada filme. Nem todo filme possui estrelas. Para cada ator os clientes às vezes desejam saber o seu nome real, bem como a data de nascimento. A locadora possui muitos clientes cadastrados. Somente clientes cadastrados podem alugar fitas. Para cada cliente é necessário saber o seu prenome e o seu sobrenome, o seu telefone e o seu endereço. Além disso, cada cliente recebe um número de associado. Finalmente, desejamos saber que fitas cada cliente retém num dado instante.

# Modelo Conceitual: **Lista de Exercício – L4**

## Clínica

Em uma clínica trabalham médicos e existem pacientes internados. Cada médico é identificado pelo seu CRM, possui um nome e recebe um salário na clínica. Um médico tem formação em diversas especialidades (ortopedia, traumatologia, etc), mas só exerce uma delas na clínica. Para todo paciente internado na clínica são cadastrados alguns dados pessoais: nome, RG, CPF, endereço, telefone(s) para contato e data do nascimento. Um paciente tem sempre um determinado médico como responsável (com um horário de visita diário predeterminado), porém vários outros médicos podem participar do seu tratamento. Pacientes estão sempre internados em quartos individuais, que são identificados por um número e ficam em um andar da clínica.



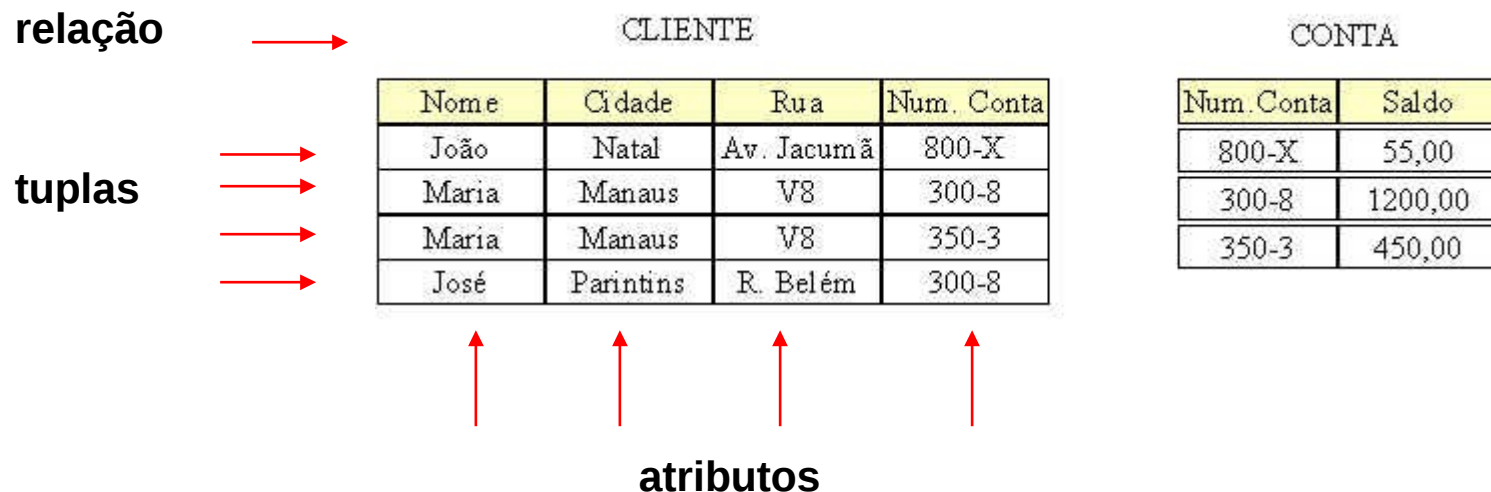


# Modelo RELACIONAL

Banco de Dados

# Modelo Relacional

- representa **DADOS** e **RELACIONAMENTOS** por um **conjunto de Tabelas**
- o modelo relacional utiliza o conceito de **relações ou tabelas** no lugar de arquivos.
- As colunas de uma tabela são chamadas de **atributos** e suas linhas de **tuplas**. A importância do modelo relacional em nosso curso deve-se ao fato de ser o modelo sobre o qual é baseada a maioria dos SGBDs comerciais disponíveis hoje em dia (MySQL, FireBird e PostgreSQL, por exemplo)



# Projeto de Banco de Dados

- “O objetivo básico de um projeto de banco de dados é possibilitar ao usuário obter a informação exata em um limite aceitável de tempo, de maneira a executar sua tarefa dentro da organização.” (Teorey e Fry)
- “O objetivo de um projeto de banco de dados relacional é gerar um conjunto de esquemas relacionais, que nos permita guardar informações sem redundância desnecessária, apesar de nos permitir recuperar a informação facilmente.” (Korth e Silberschatz)

# Perigos potenciais de projetos

- Repetição de informação
  - Informações repetidas consomem espaço de armazenamento e dificultam a atualização.
- Incapacidade de representar parte da informação. Por vezes tem-se que incluir valores nulos.
- Perda de informação.
- Projetos mal elaborados sugerem a decomposição de esquemas relacionais com muitos atributos.

# Modelo Relacional

- O Modelo Relacional é **Simples** e sua estrutura **Uniforme** é baseada em conceitos da **Teoria dos Conjuntos**.
- A **simplicidade** do modelo relacional faz com que a representação do mundo real através de seus conceitos seja de certa forma ineficiente o que ocasiona perdas semânticas consideráveis
- O **MER**, ao contrário, utiliza conceitos que permitem a representação mais fiel dos objetos do mundo real e dos relacionamentos entre eles
- O Modelo Relacional tem sido implementado nos vários SGBDs tendo como **LDD/LMD** a linguagem **SQL**.
- O MER é hoje a ferramenta mais usada em projetos de banco de dados. Dizemos que o **MER** é um **modelo do nível conceitual**, pois possui um forte **poder semântico**, capaz de capturar conceitos do mundo real com um mínimo de perdas semânticas, facilitando o seu entendimento.
- O **modelo relacional** é, por outro lado, um modelo do **nível lógico** porque é utilizado para representação em computador de conceitos do mundo real.

# Modelo Relacional

## ■ Codd, E. F.

- O Dr. Codd propôs o modelo relacional de sistemas de bancos de dados em 1970.
- Ele é a base para o RDBMS (relational database management system).
- O modelo relacional consiste nos seguintes itens:
  - Conjunto de objetos ou relações
  - Conjunto de operadores para agir sobre as relações
  - Integridade de dados para precisão e consistência
- “A relational model of data for large shared data banks”. Communications of the ACM, 13(6):377-87, June 1970

# Modelo Relacional

## ■ Objetivos

- Independência de dados
  - ordem
  - indexação
  - caminhos de acesso
- reduzir inconsistências
  - regras de projetos (normalização)

# Modelo Relacional

## Modelo Relacional: Informal

- um banco de dados relacional é um conjunto de relações ou tabelas bidimensionais, gerenciados por operações relacionais e regidos por restrições de integridade de dados

- Pode ser acessado e modificado executando instruções SQL (Structured Query Language)



- Usa um conjunto de operadores

Nome da Tabela: **EMP**

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

Nome da Tabela: **DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



### Relação: Representação Tabular

- Propriedades:
    - cada linha representa uma tupla
    - não existe ordenamento entre as tuplas
    - não existem linhas duplicadas
    - o relacionamento das colunas com os domínios estabelece-se a princípio dando os nomes dos domínios às colunas. Caso um mesmo domínio seja usado mais de uma vez, o nome do domínio é adicionalmente qualificado pelo **papel** desempenhado
- PEÇA-COMPOSTA = PEÇA x PEÇA x QTD
- composto.PEÇA | componente.PEÇA | QTD
- não existe ordem entre as colunas, respeitada a propriedade acima

### Banco de Dados Relacional

- Esquema
  - definição das tabelas
- Instâncias
  - conjunto de tuplas que compõem as relações
- Para cada relação define-se, usando a DDL (linguagem de definição de dados)
  - nome único
  - atributos
  - restrições de integridade (chave primária, chave estrangeira, valores nulos, etc)

### Restrições de Integridade do Modelo relacional

- Restrições de Integridade (RI) que se aplicam a todo modelo que se diz conforme ao modelo relacional:
  - integridade de identidade
  - integridade referencial

### Integridade de Identidade: Conceitos

- chave candidata
  - grupo mínimo de atributos tal que a combinação de valores assumida por este grupo corresponde a no máximo uma tupla da relação
  - propriedade mínima
  - propriedade de imparidade
  - uma relação possui no **mínimo** uma chave candidata
- chave primária
  - uma entre as chaves candidatas, selecionada pelo projetista como a principal identificação das tuplas de uma relação
  - uma relação possui **uma e somente** uma chave primária

### Integridade de Identidade: Conceitos

- chave alternativa
  - toda chave candidata que não for chave primária
- valor nulo
  - a inserção de tuplas incompletas pode introduzir valores nulos na base de dados
  - evitar, sempre que possível
- integridade de identidade
  - nenhum atributo que participe de uma chave candidata de uma relação pode assumir valor nulo

A DDL utilizada deve prover algum tipo de mecanismo para definir chaves primárias e candidatas (valores únicos), e para especificar quando um atributo pode ou não aceitar valores nulos

```
CREATE TABLE <table_name>
```

```
(<atr_name> <domain> [NOT NULL]
```

```
[, <atr_name> <domain> [NOT NULL]] *
```

```
[, PRIMARY KEY (atr_name(s))]
```

```
[, UNIQUE (atr_name(s))] * ;
```

```
CREATE [UNIQUE] INDEX <index_name>
```

```
ON <table_name>(atr_name(s));
```

### Integridade Referencial: Conceitos

- chave estrangeira (externa)
  - um ou mais atributos de uma relação R2 cujos valores são necessários para equivaler à chave primária de uma relação R1 (R1 e R2 não necessariamente distintos)
  - não necessariamente a chave estrangeira participa da chave primária da relação que a contém (R2)
- integridade referencial
  - se uma relação R2 inclui uma chave estrangeira FK equivalendo à chave primária PK de uma relação R1, então todo valor FK em R2 deve ser:
    - ✓ igual ao valor de PK em alguma tupla de R1 ou
    - ✓ totalmente nulo

- Definição de Chave Estrangeira
  - política de rejeição (DEFAULT)
    - ✓ a operação só não é rejeitada se não houver tuplas (chave estrangeira) fazendo referência a uma dada chave primária
  - políticas compensatórias (EXPLICITAMENTE DECLARADAS)
    - ✓ CASCADE: propaga a alteração/remoção de tuplas
    - ✓ SET NULL: o valor da chave estrangeira é ajustada para valor NULO



## Modelo Relacional

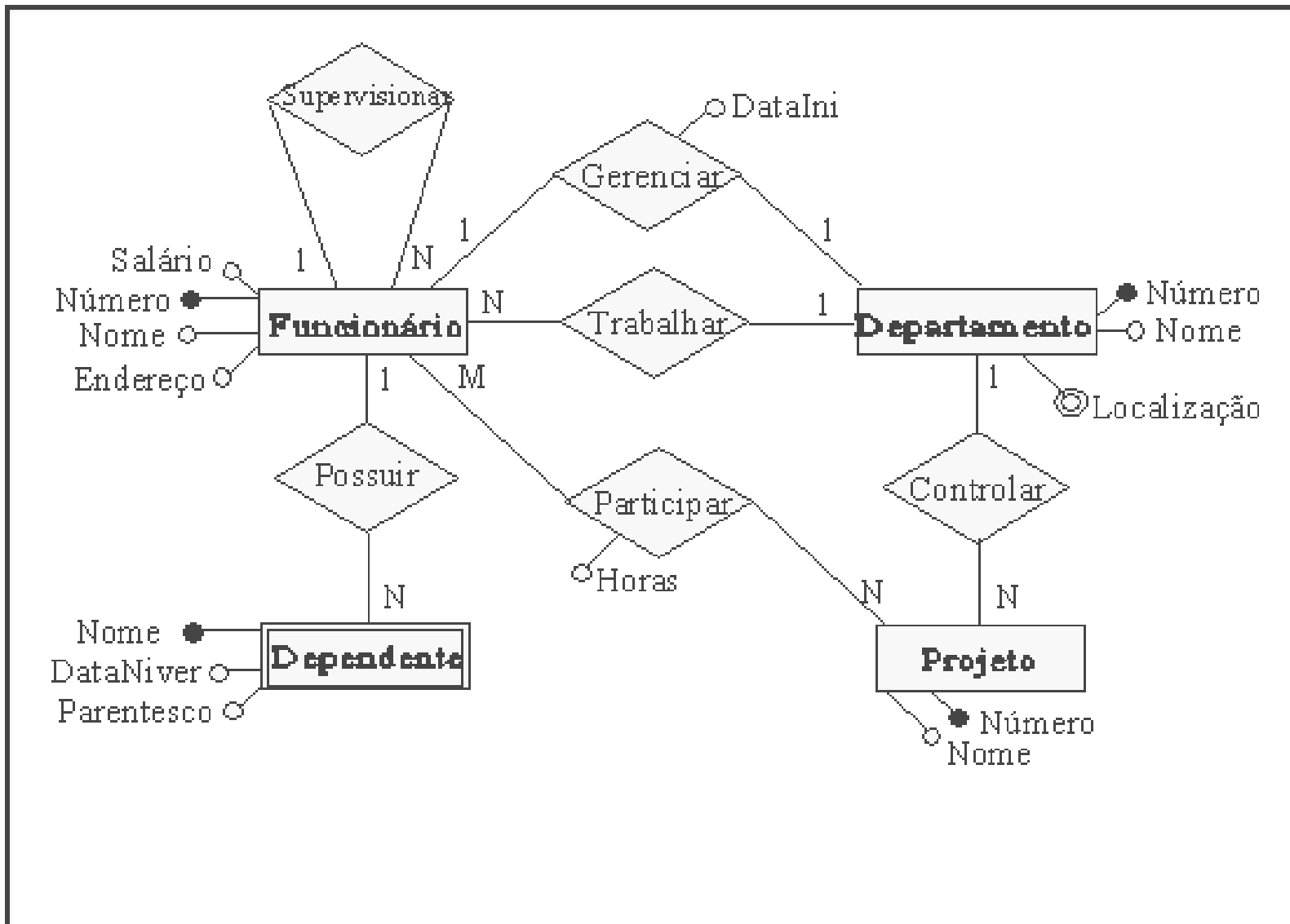
```
CREATE TABLE departamento  
(cod INT NOT NULL,  
nome char(15) NOT NULL,  
ramal CHAR(4),  
PRIMARY KEY (cod));
```

```
CREATE TABLE empregado  
(cpf CHAR(8) NOT NULL,  
nome CHAR(60) NOT NULL,  
salario MONEY,  
dep INT,  
gerente CHAR(8),  
PRIMARY KEY (cpf),  
FOREIGN KEY dep REFERENCES departamento(cod),  
FOREIGN KEY gerente REFERENCES empregado(cpf));
```

**QUAL A POLÍTICA?**

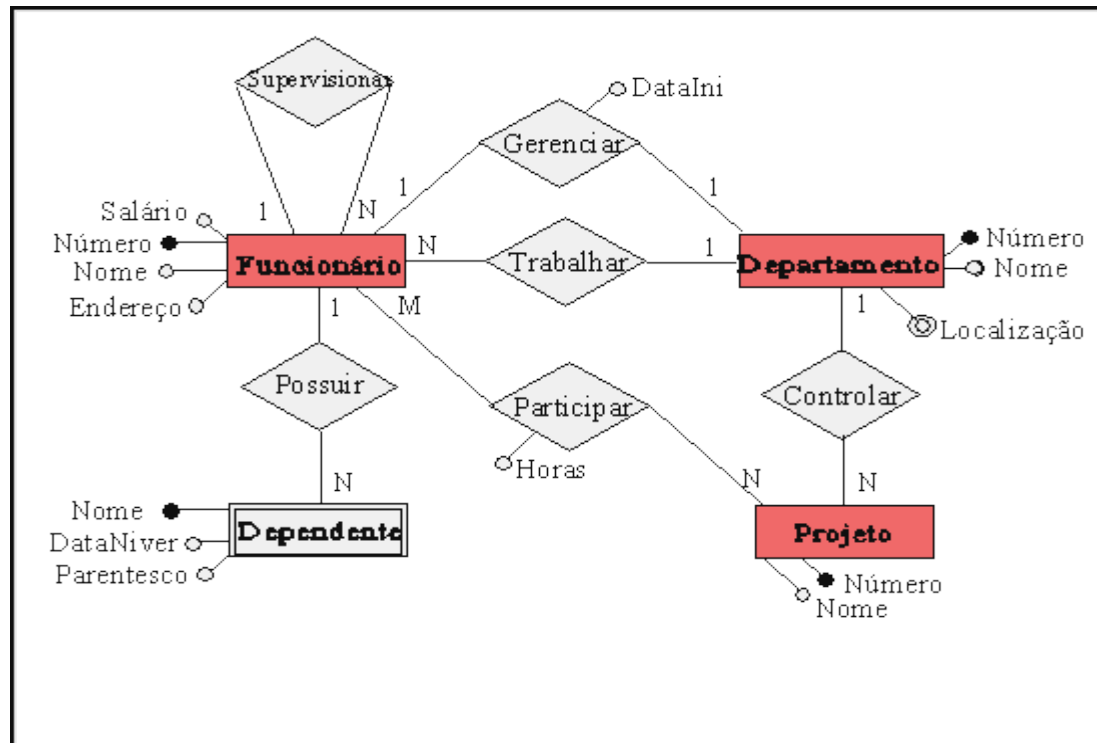
```
CREATE TABLE empregado
(cpf CHAR(8) NOT NULL,
nome CHAR(60) NOT NULL,
salario MONEY,
dep INT,
gerente CHAR(8),
PRIMARY KEY (cpf),
FOREIGN KEY dep REFERENCES departamento(cod)
ON DELETE SET NULL
ON UPDATE CASCADE
FOREIGN KEY gerente REFERENCES empregado(cpf)
ON DELETE SET NULL);
```

## Mapeamento do MER para o Relacional



## Passo 1 - Mapear Conjuntos de Entidades Regulares

- Mapear todos os conjuntos de entidades regulares (que não são fracas)
- Para cada conjunto de entidade E no esquema ER cria-se uma relação R que inclui todos os atributos de E
- Caso exista atributo composto, inclua todos os atributos elementares que compõem o atributo composto
- Escolha um dos atributos chave de E como chave primária para a relação R



## Passo 1 - Mapear Conjuntos de Entidades Regulares

- **Primeiro Passo:**

Funcionário = {FNúmero, FNome, Endereço, Salário}

Departamento = {DNúmero, DNome}

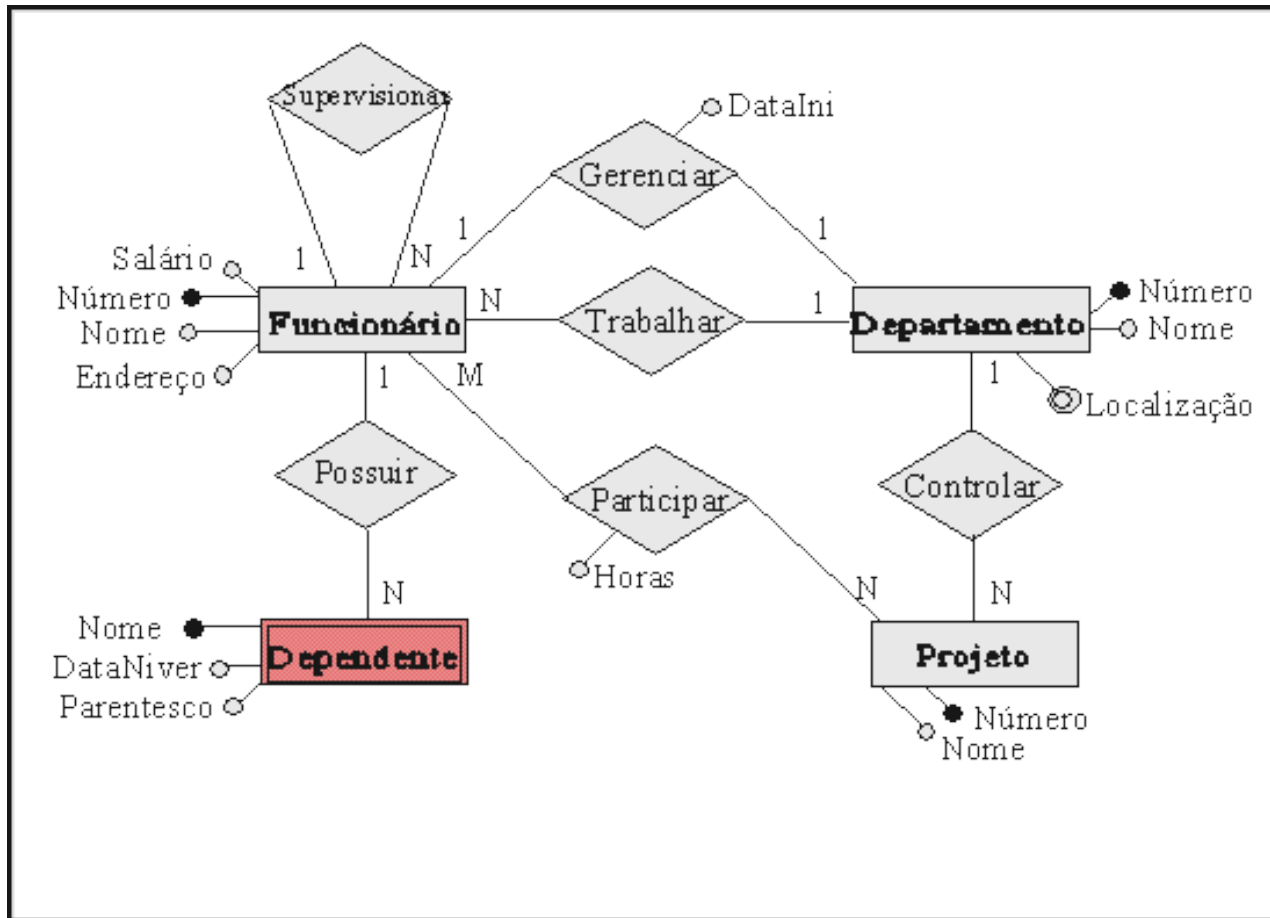
Projeto = {PNúmero, PNome}

### **OBSERVAÇÕES:**

- O atributo "Localização" não foi representado na relação "Departamento" pois é um atributo multi-valorado e será tratado no passo 7.
- O conjunto de entidade "Dependente", é um conjunto de entidade fraco e será tratado no seguinte, o passo 2

- Para cada entidade fraca  $F$  no esquema ER cria-se uma relação  $R$  formada por todos os atributos de  $F$  mais os atributos que são chave das entidades envolvidas nos relacionamentos com  $F$
- A chave de  $R$  é a chave de  $F$  concatenada com as chaves das entidades envolvidas com  $F$

## Passo 2 - Mapear Conjuntos de Entidades Fracas



- **Segundo Passo:**

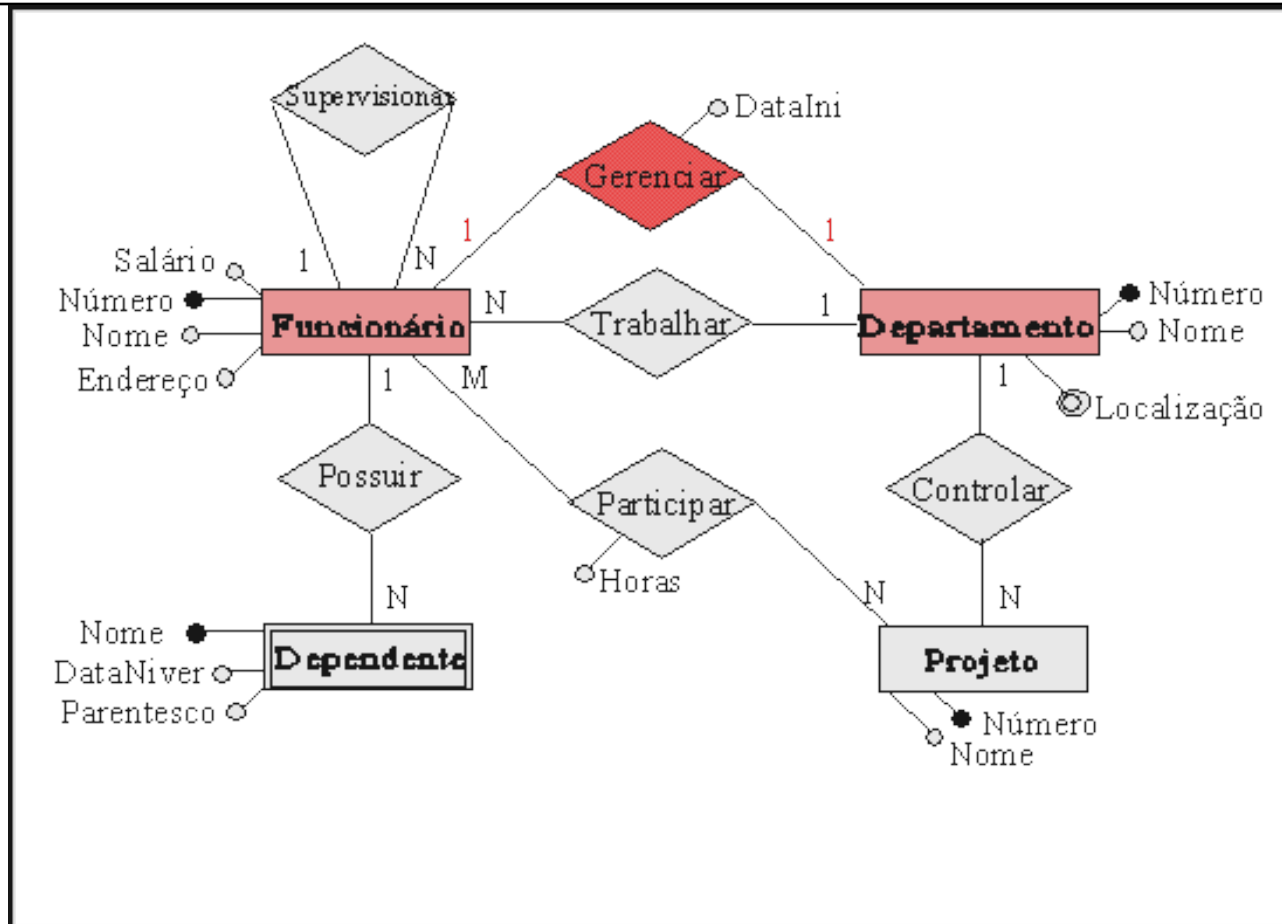
Dependente = {DNome, FNúmero, DataNiver, Parentesco}

### Passo 3 - Mapear Conjuntos de Relacionamentos Binário 1:1

- O mapeamento de conjuntos de relacionamentos (CR) nem sempre resultam em uma nova relação, como é o caso dos conjuntos de relacionamento binário 1:1. Deve-se identificar os conjuntos de entidades S e T que participam do relacionamento.
- Uma das entidades poderá ser escolhida. Se uma entidade participar de forma total no relacionamento, esta deverá ser escolhida
- Acrescenta-se a entidade escolhida os atributos do relacionamento e ainda os atributos chaves da outra entidade
- Deve-se ressaltar que os atributos chave que foram acrescentados à entidade escolhida, são incluídos como atributos não chave nesta relação



### Passo 3 - Mapear Conjuntos de Relacionamentos Binário 1:1



- **Terceiro Passo:**

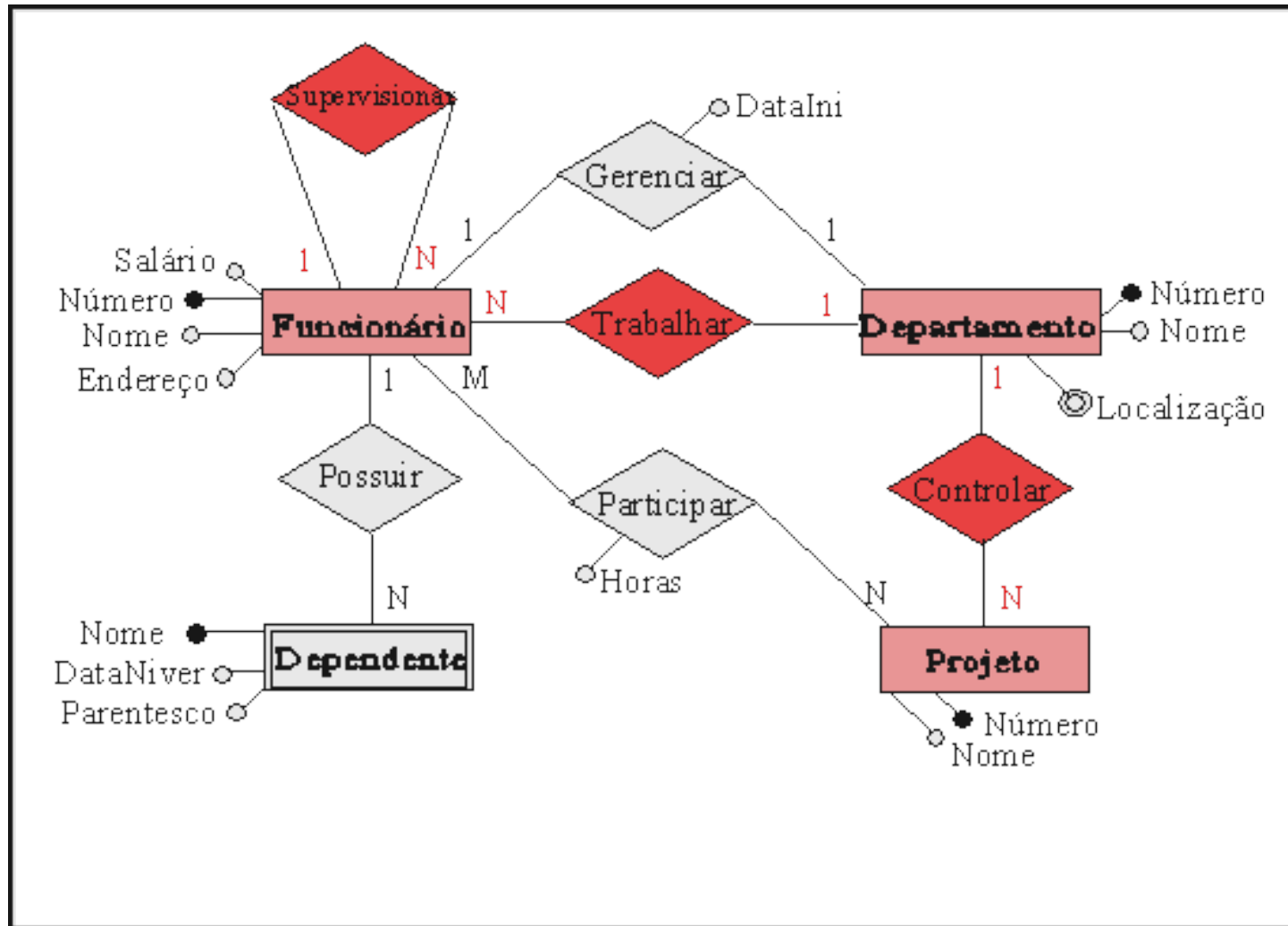
Departamento = {DNúmero, Dnome, DataIni, FNúmero}

Os atributos **FNúmero** e **DataIni** são adicionados à relação "Departamento", definida no primeiro passo. Note que o atributo **FNúmero** é adicionado como um atributo não chave na relação "Departamento"

## Passo 4 - Mapear Conjuntos de Relacionamento Binário Regular 1:N

- Os conjuntos de relacionamento binário regular (não fraco) **1:N** também **não** são representados como novas relações.
- Primeiro identifica-se o conjunto de entidade que participa da relação com cardinalidade N que será chamada de S e o outro conjunto de entidade chamada de T.
- Os atributos do conjunto de relacionamento são acrescentados no conjunto de entidade S, ou seja, o conjunto de entidade com cardinalidade N.
- Os atributos chave da relação que mapeia o conjunto de entidade que participa com cardinalidade 1, representado por T, são também acrescentados no conjunto de entidade S como atributos não chave.

## Passo 4 - Mapear Conjuntos de Relacionamento Binário Regular 1:N



Neste passo são mapeados os relacionamentos: supervisionar, trabalhar e controlar.

- **Quarto Passo:**

Funcionário = {FNúmero, FNome, Endereço, Salário, SuperNúmero, DNúmero}

Projeto = {PNúmero, Pnome, DNúmero}

Avaliando o relacionamento supervisionar, nota-se que este relacionamento não possui atributos, logo, apenas o atributo **SuperNúmero** foi adicionado a relação "Funcionário", definida no primeiro passo.

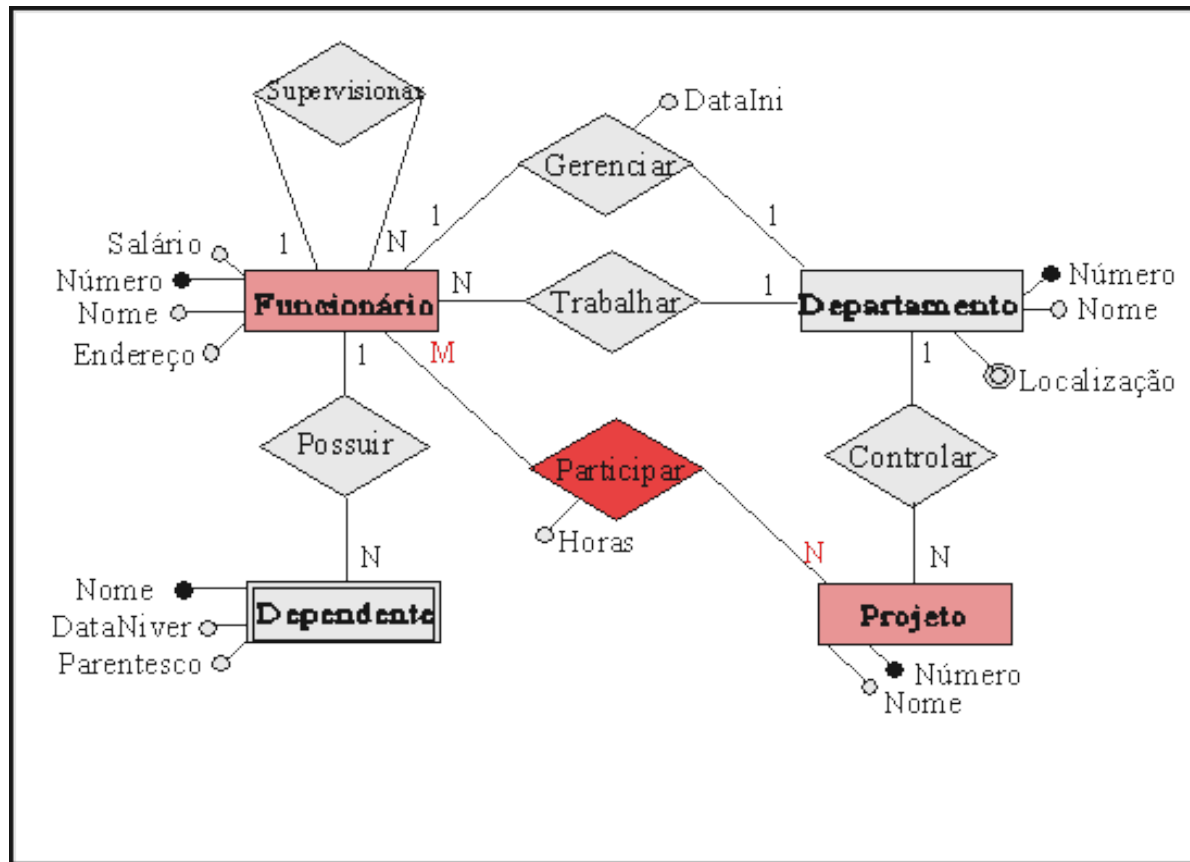
O mesmo acontece com o relacionamento trabalhar; apenas o atributo **DNumero** foi adicionado a relação "Funcionário".

No relacionamento controlar, que também não possui atributos, foi adicionado o atributo **DNumero** na relação Projeto que já havia sido definida no primeiro passo.

## Passo 5 - Mapear Relacionamento Binário N:N

- Para cada relacionamento binário N:N cria-se uma nova relação.
- Os atributos da relação são os atributos do conjunto de relacionamento juntamente com os atributos chave das relações que mapeiam os conjuntos de entidades envolvidas.
- A chave da relação é a concatenação dos atributos chave das relações que mapeiam os conjuntos de entidades envolvidos.

## Passo 5 - Mapear Relacionamento Binário M:N



### • Quinto Passo:

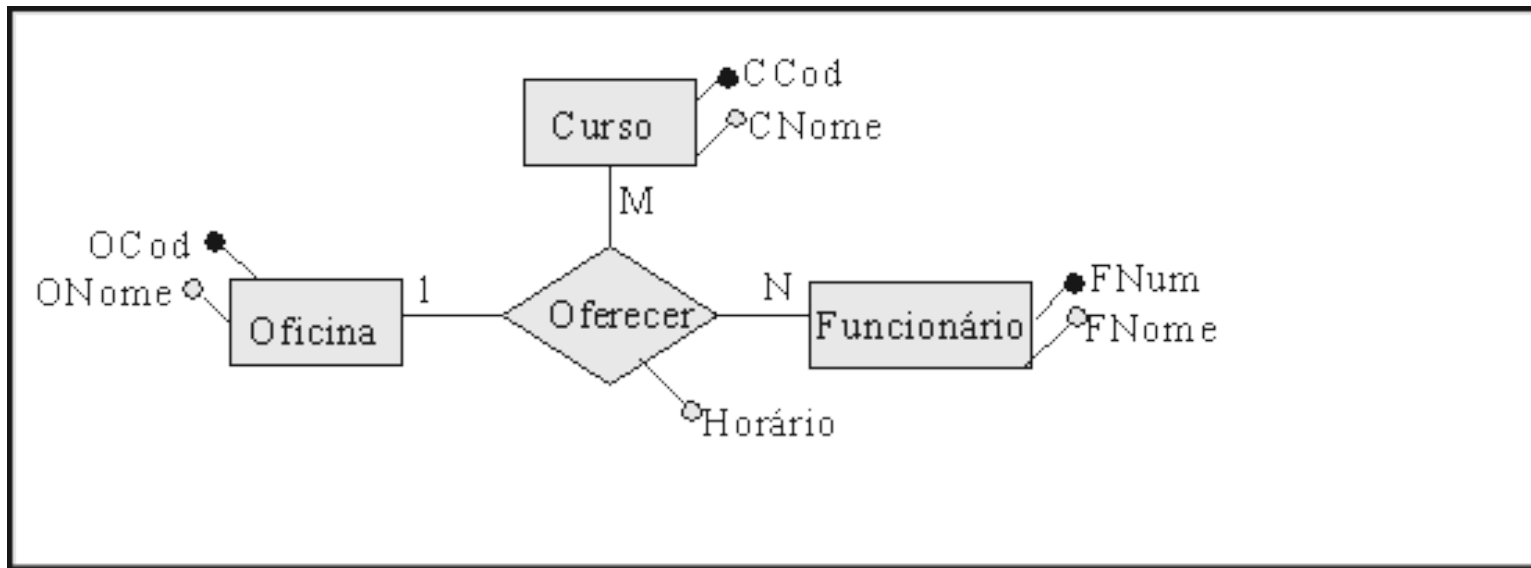
O relacionamento **participar** que envolve os conjuntos de entidades "Funcionário" e "Projeto", é avaliado neste passo.

Cria-se uma nova relação: Participar = {FNúmero, PNúmero, horas}

## Passo 6 - Mapear Conjuntos de Relacionamentos n-ário, $n > 2$

- Para conjuntos de relacionamentos n-ário,  $n > 2$  sempre considera-se que possuam cardinalidade vários:vários:vários. Para cada conjunto de relacionamento (CR) será criada uma nova relação cujos atributos próprios são os do CR (se existirem) e cuja chave é formada pelos atributos chave das relações que mapeiam os conjuntos de entidades (CE's) envolvidos.
- Os CR's de ordem maior que três são tratados da mesma maneira que os ternários. Seu mapeamento cria uma relação para cada CR e esta relação possui: os atributos do CR como atributos próprios e, como chave, os atributos concatenados de cada relação que mapeia os CE's envolvidos.
- Como a modelagem utilizada para ilustrar os passos anteriores não tem relacionamento n-ário, com  $n > 2$ , escolheu-se um exemplo particular que será apresentado a seguir:

## Passo 6 - Mapear Conjuntos de Relacionamentos n-ário, n>2



- **Mapeamento do conjunto de relacionamento ternário:**

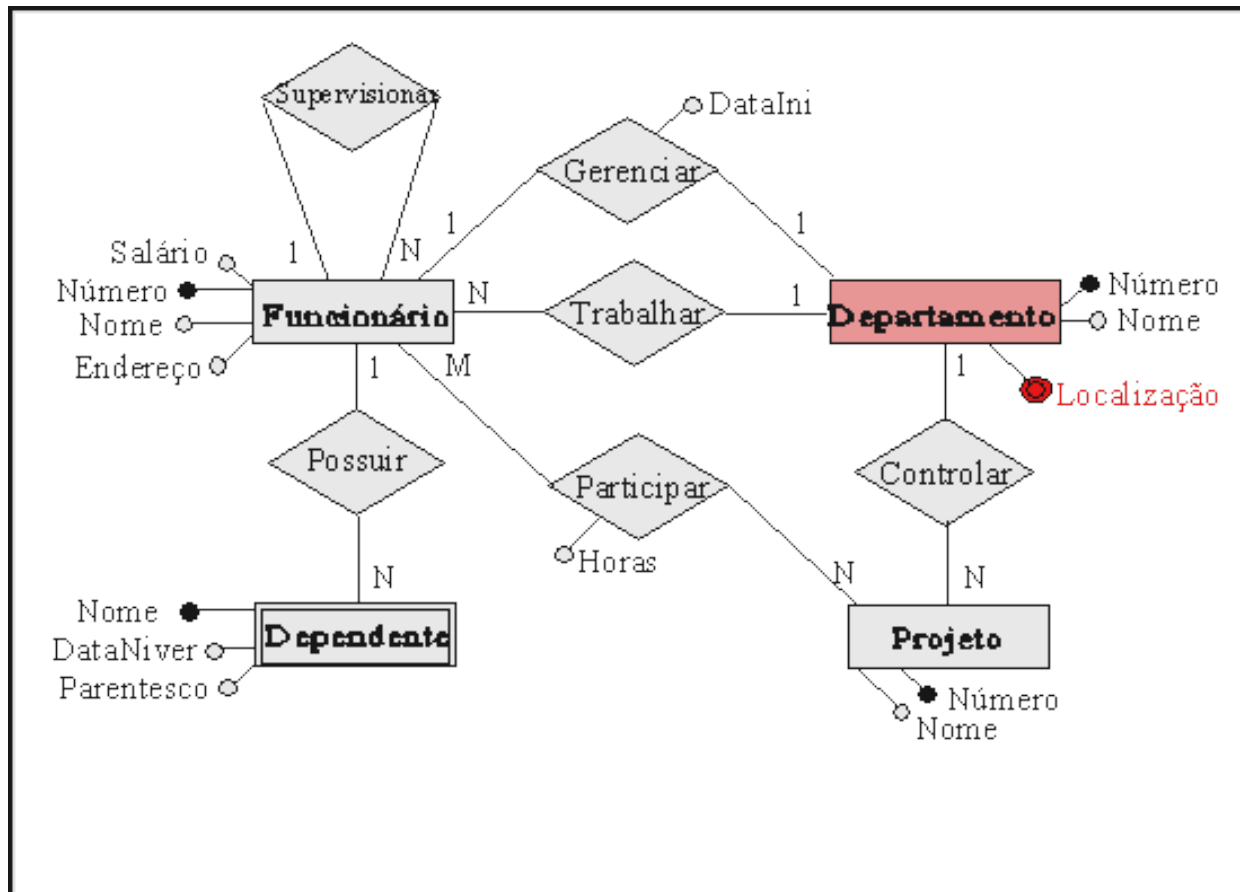
Oferecer = {OCod, CCod, FNum, Horário}



## Passo 7 - Mapear Atributos Multi-valorados

- Existem duas maneiras de mapear atributos multi-valorados.
- A primeira maneira não leva-se em conta conhecimento adicional sobre o atributo que está sendo mapeado. Para cada atributo multi-valorado cria-se uma nova relação que tem como chave os atributos chave da relação a qual pertencia juntamente com o atributo multi-valorado tomado como um atributo mono-valorado.
- A segunda forma de mapear atributos multi-valorados leva-se em conta o conhecimento adicional sobre o atributo que está sendo mapeado. Em alguns casos é possível determinar a quantidade de ocorrências de valores nos atributos.
- Quando isso acontece e essa quantidade é pequena, pode-se instanciar essa quantidade de atributos como mono-valorados na mesma relação que mapeia o conjunto de entidade ou conjunto de relacionamento ao qual o atributo multi-valorado está associado.

## Passo 7 - Mapear Atributos Multi-valorados



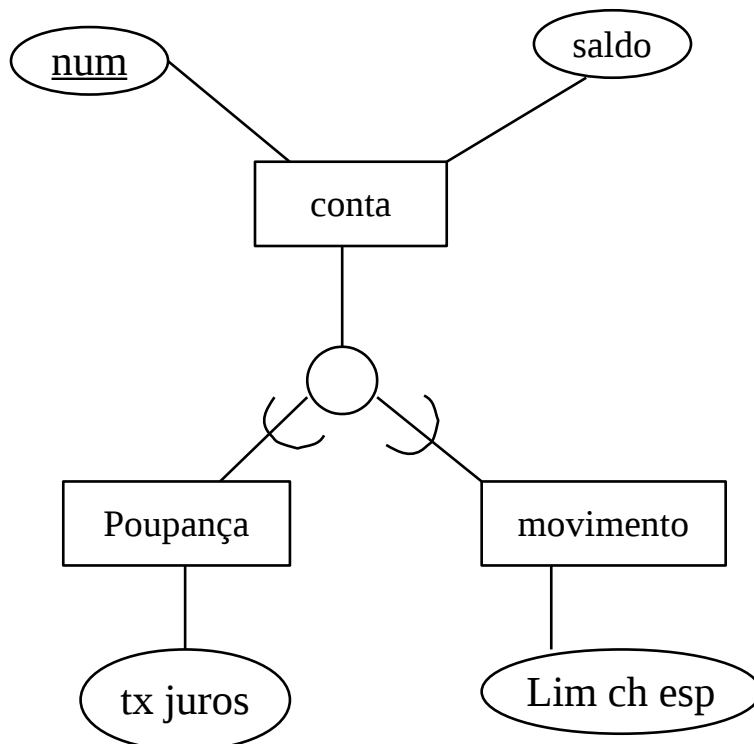
### • Sétimo Passo:

LocalDep = {DNúmero, Localização}

## Passo 8 - Mapear Generalização/Especialização

Existem duas maneiras de transformar uma generalização em tabelas:

1. Criar a tabela para o conjunto de entidades de nível superior. Para cada conjunto de entidades de nível inferior, criar uma tabela que inclua uma coluna para cada um dos atributos daquele conjunto de entidades mais uma coluna para cada atributo da chave primária do conjunto de entidades de nível superior

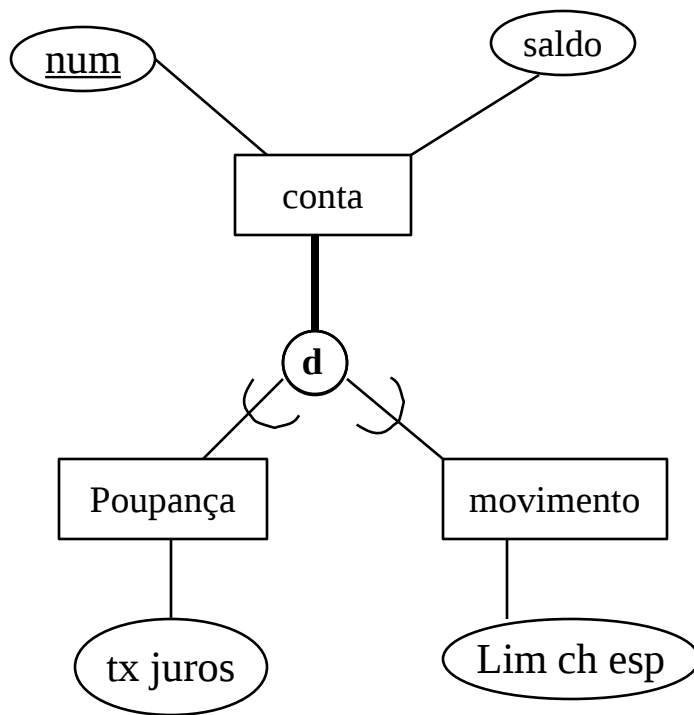


conta = {num, saldo}  
poupança = {num, tx juros}  
movimento = {num, lim ch esp}

## Passo 8 - Mapear Generalização/Especialização

2. Se a generalização é mutuamente exclusiva e total, isto é, se nenhuma entidade é membro de mais de um conjunto de entidades de nível imediatamente inferior ao conjunto de entidades de nível superior e se todas as entidades do conjunto de entidades de nível superior são membros também de um dos conjuntos de entidades de nível inferior, então, uma outra representação alternativa é possível.

Para cada conjunto de entidades de nível inferior, cria-se uma tabela que inclua uma coluna para cada um dos atributos do conjunto de entidades mais uma coluna para cada atributo de conjunto de entidades de nível superior

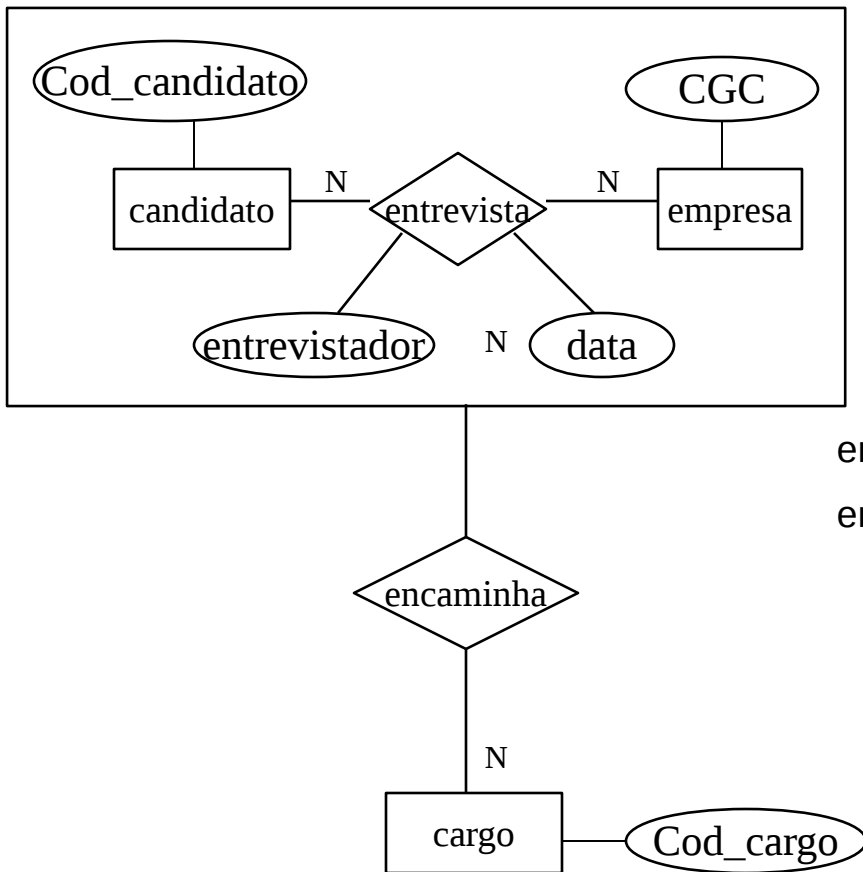


poupança = {num, saldo, tx juros}

movimento = {num, saldo, lim ch esp}

## Passo 9 - Mapear Agregação

- A transformação de agregação em tabela é bastante direta. Considere o exemplo abaixo. A tabela para o relacionamento **entrevista** inclui uma coluna para cada atributo do relacionamento, uma para a chave primária de **candidato** e uma para **empresa**.

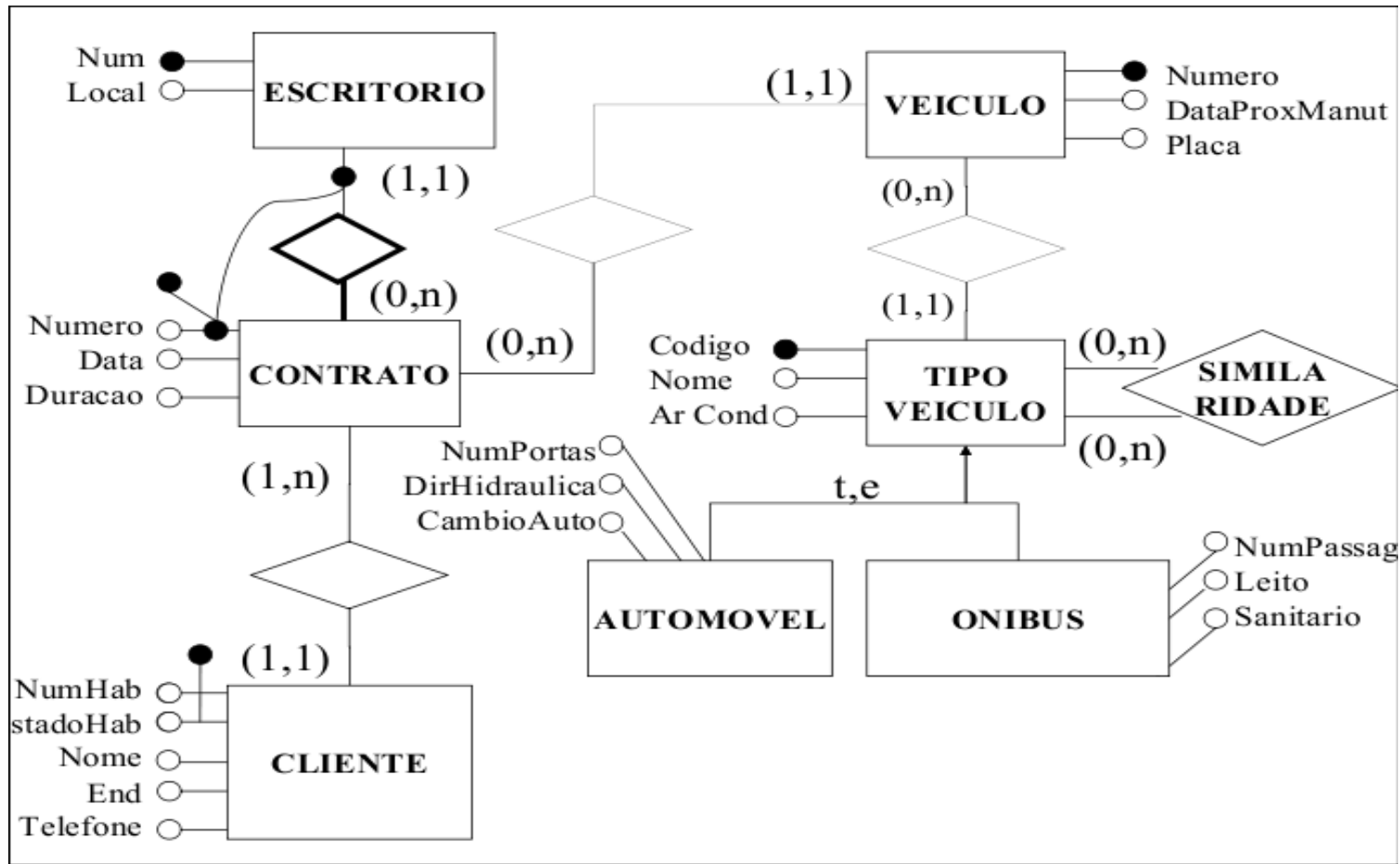


entrevista = {cod\_candidato, CGC, entrevistador, data}

encaminha = {cod\_candidato, CGC, cod\_cargo}

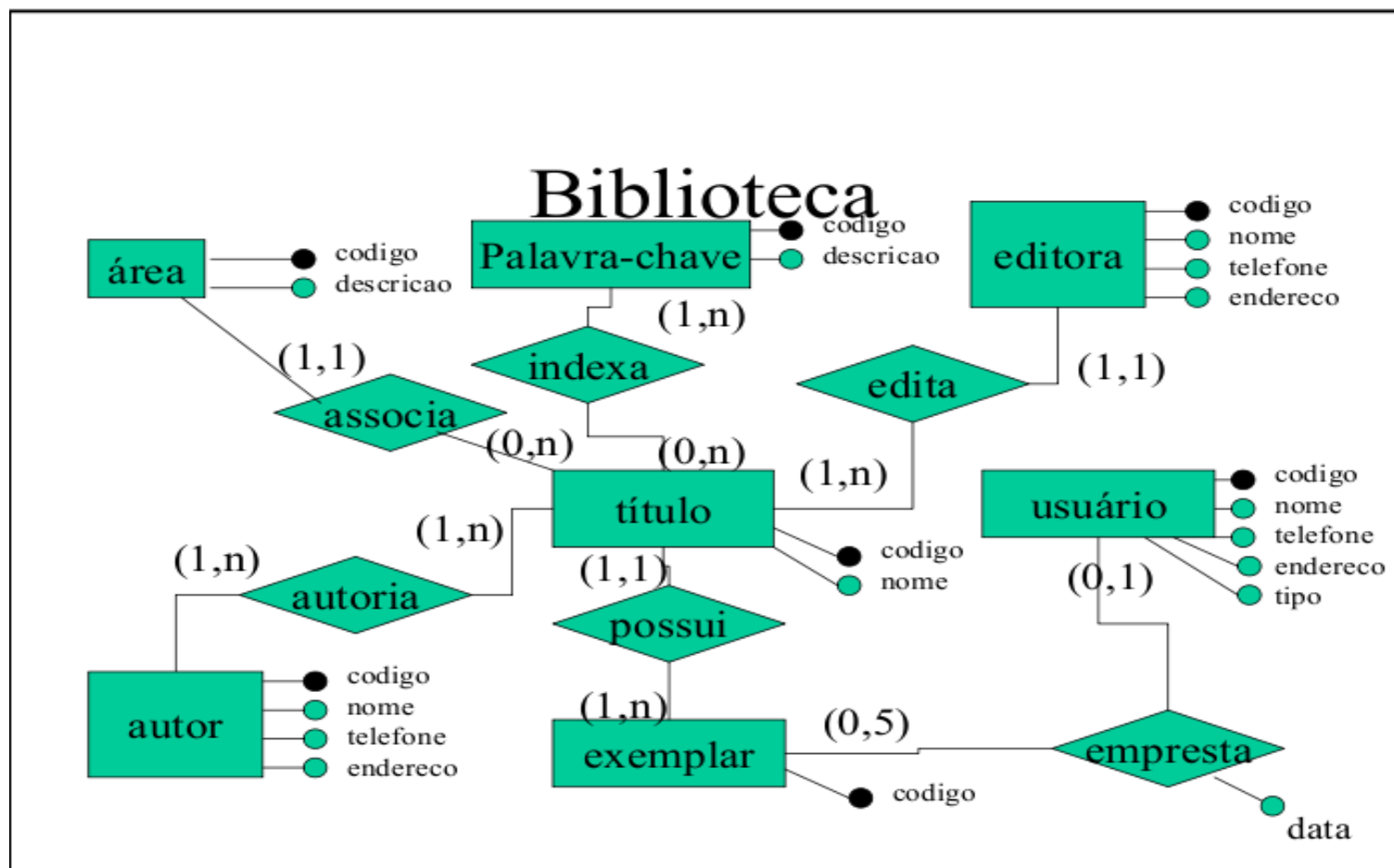
# Exercícios

- Faça o mapeamento para o modelo relacional



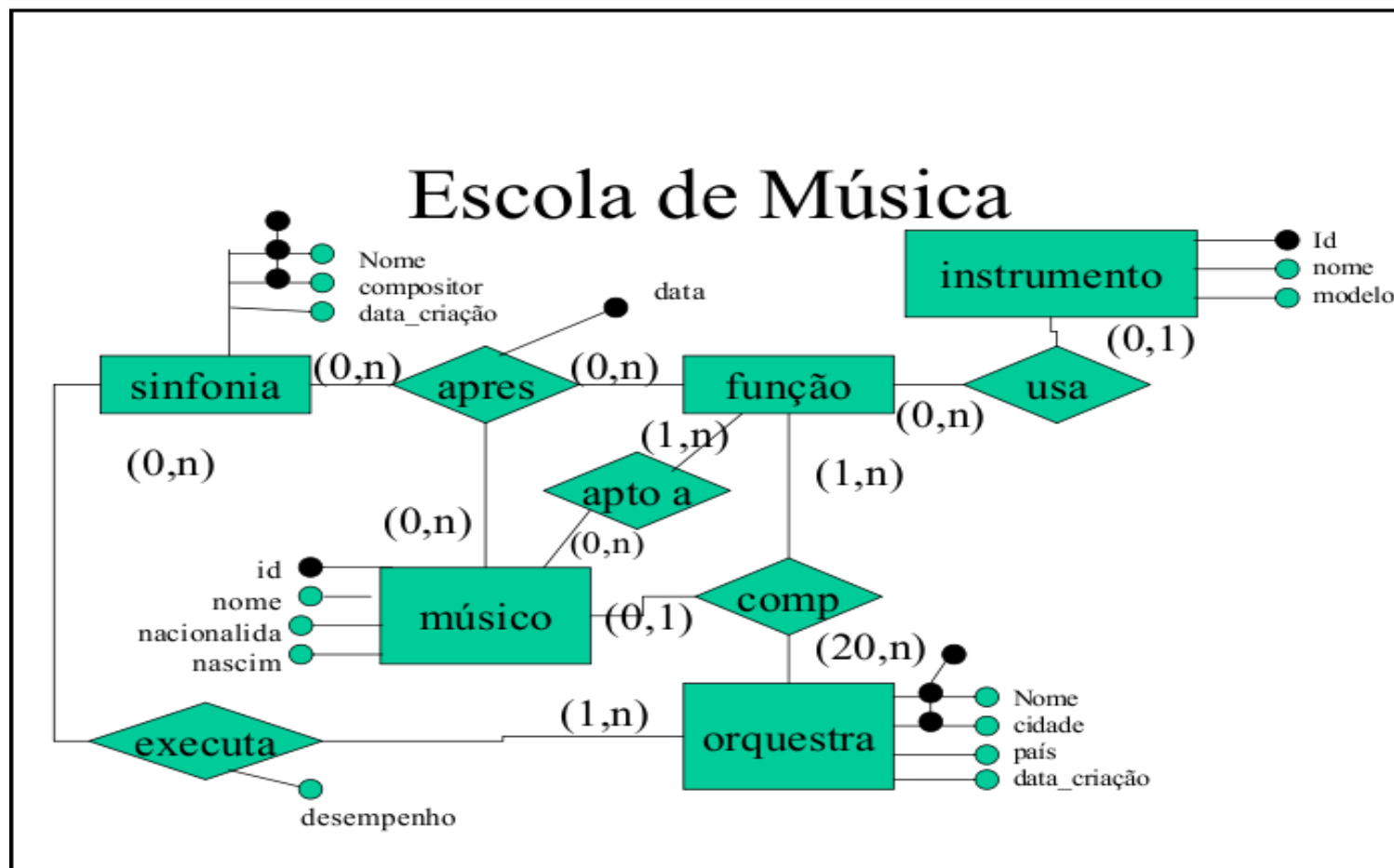
# Exercícios

- Faça o mapeamento para o modelo relacional



# Exercícios

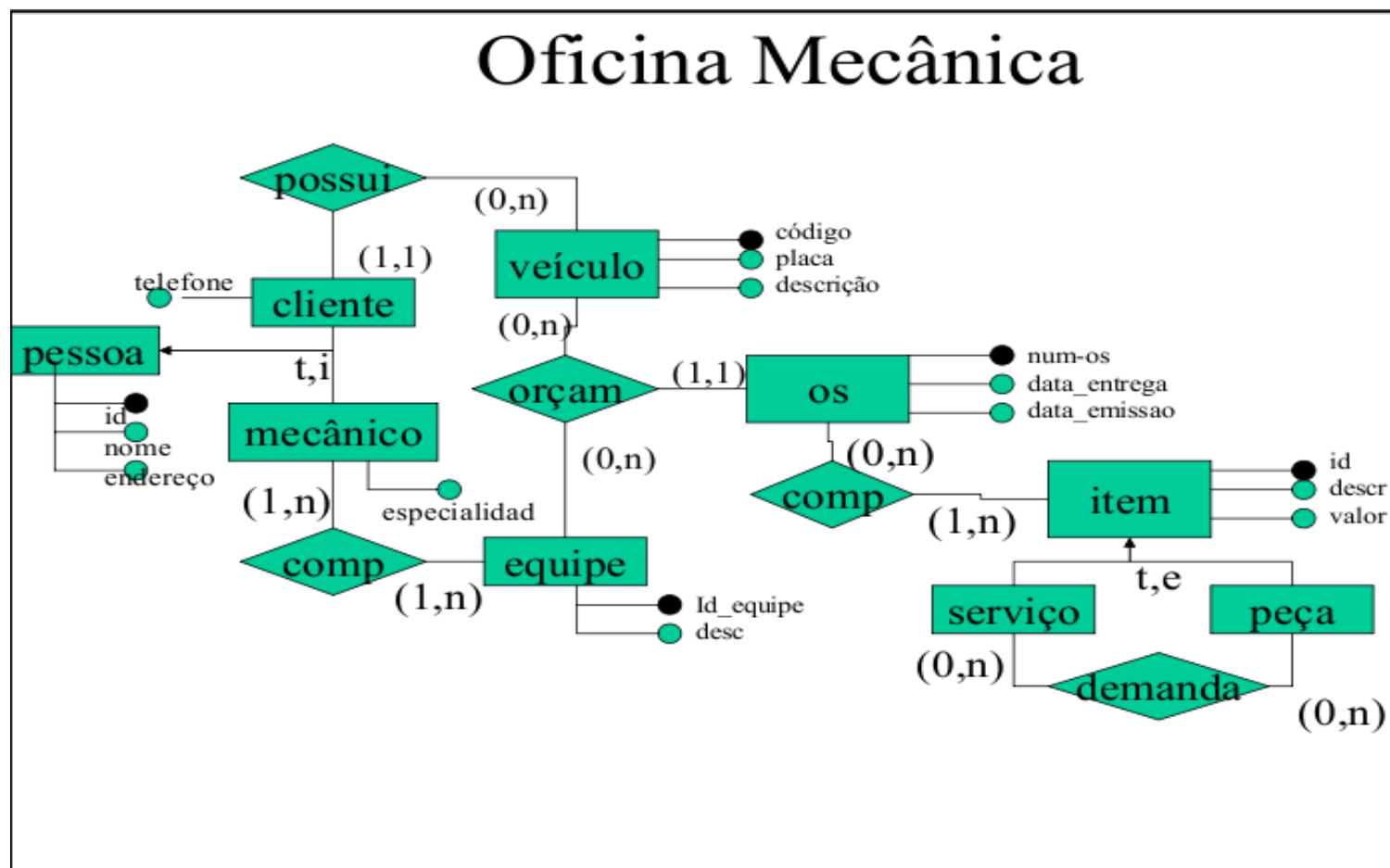
- Faça o mapeamento para o modelo relacional





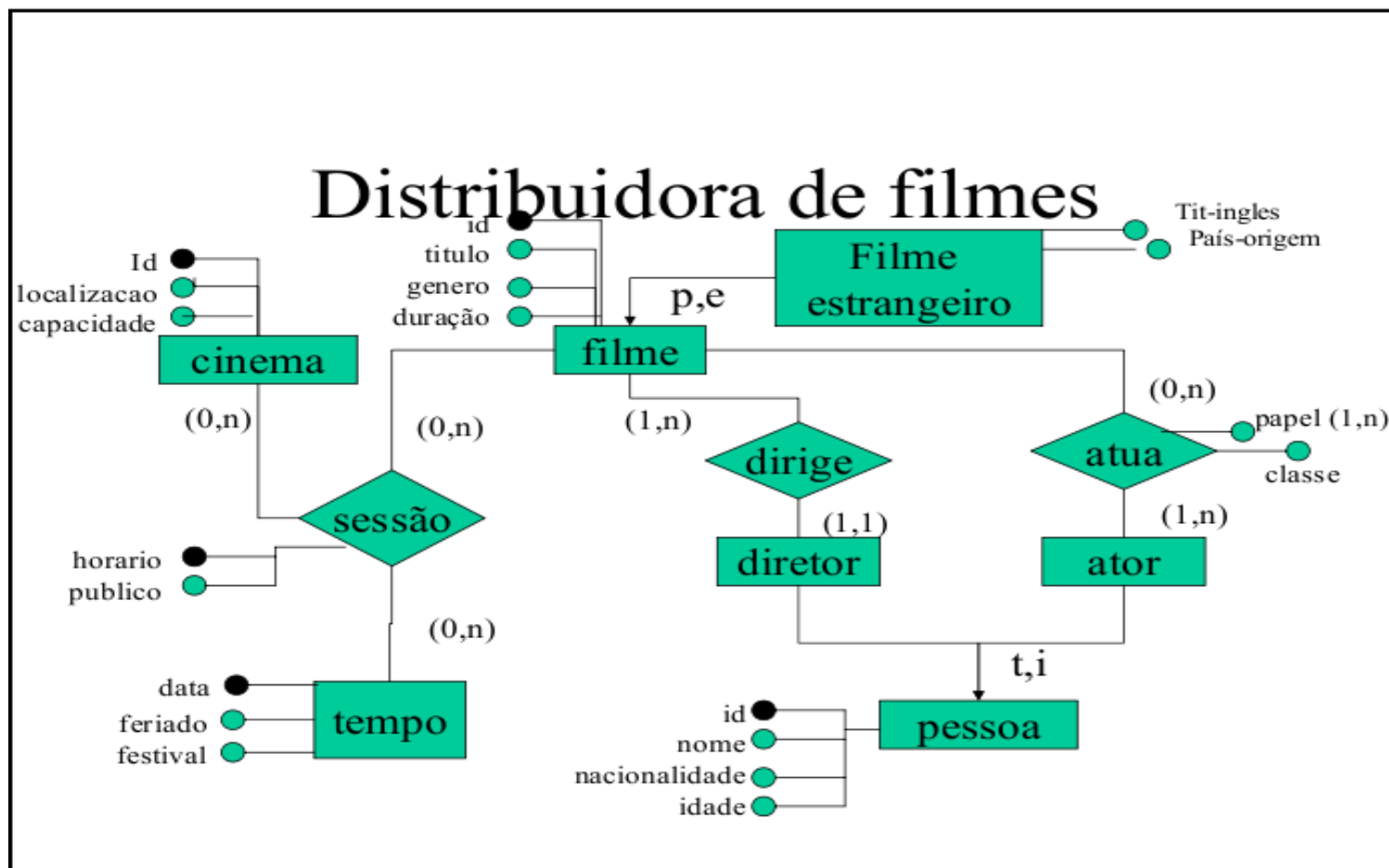
# Exercícios

- Faça o mapeamento para o modelo relacional



# Exercícios

- Faça o mapeamento para o modelo relacional





# DEPENDÊNCIA FUNCIONAL E NORMALIZAÇÃO

# Dependência Funcional

- Uma dependência funcional é um relacionamento muitos para um entre dois conjuntos de atributos de uma determinada relação  $R$ .
- Ela é uma espécie particularmente comum e importante de restrição de integridade.

# Dependência Funcional

- Sejam os seguintes subconjuntos de atributos de um esquema T:

$$A = (A_1, A_2, \dots, A_n) \text{ e } B = (B_1, B_2, \dots, B_n)$$

- Dizemos que B é **dependente funcionalmente** de um outro atributo A contido em T se a cada valor de A existir nas linhas da relação T, em que aparece, um único valor de B.
  - Notação:  **$A \rightarrow B$**
  - Lê-se: **A determina funcionalmente B.**
- DF é uma propriedade do projeto do BD, isto é, do seu esquema. O conhecimento advém do conhecimento da semântica dos dados armazenados na relação.

# Dependência Funcional

- Dependências triviais e não triviais
  - A redução do conjunto de dependências funcionais é feito através da eliminação das dependências triviais.
  - Uma dependência é trivial se não puder deixar de ser satisfeita.
  - Dependências não triviais são as mais interessantes para o projeto de banco de dados, pois elas são as únicas que correspondem a restrições de integridade genuínas.

# Dependência Funcional

## ■ Exemplo de tabela de vendas

- F# : chave primária do fornecedor
- Cidade : cidade do fornecedor
- P# : chave primária do produto
- QDE : quantidade

F#	CIDADE	P#	QDE
F1	Londres	P1	100
F1	Londres	P2	100
F2	Paris	P1	200
F2	Paris	P2	200
F3	Paris	P2	300
F4	Londres	P2	400
F4	Londres	P4	400
F4	Londres	P5	400

# Dependência Funcional

- Exemplos de dependências funcionais válidas:
  - $\{ F\# \} \rightarrow \{ CIDADE \}$
  - $\{ F\#, P\# \} \rightarrow \{ QDE \}$
  - $\{ F\#, P\# \} \rightarrow \{ CIDADE \}$
  - $\{ F\#, P\# \} \rightarrow \{ CIDADE, QDE \}$
  - $\{ F\#, P\# \} \rightarrow \{ F\# \}$
  - $\{ F\#, P\# \} \rightarrow \{ F\#, P\#, CIDADE, QDE \}$
  - $\{ F\# \} \rightarrow \{ QDE \}$
  - $\{ QDE \} \rightarrow \{ F\# \}$



# Dependência Funcional

- Propriedades funcionais
- Sejam  $A$ ,  $B$  e  $C$  subconjuntos arbitrários do conjunto de atributos de uma relação  $R$ , e considerando que  $AB$  é usada para indicar a união de  $A$  e  $B$ , teremos as seguintes propriedades (regras de inferência de Armstrong):
  - **Reflexão**: se  $B$  é um subconjunto de  $A$ , então  $A \rightarrow B$ .
  - **Aumento**: se  $A \rightarrow B$ , então  $AC \rightarrow BC$ .
  - **Transitividade**: se  $A \rightarrow B$  e  $B \rightarrow C$ , então  $A \rightarrow C$ .

# Dependência Funcional

- **União:** se  $A \rightarrow B$  e  $A \rightarrow C$ , então  $A \rightarrow BC$ .

Prova:

$A \rightarrow AB$  (regra aumento),

e, como  $A \rightarrow C$ ,  $AB \rightarrow BC$  (aumento),

e, como  $A \rightarrow AB$  e  $AB \rightarrow BC$ ,

logo  $A \rightarrow BC$  (regra transitividade).

# Dependência Funcional

- **Decomposição:** se  $A \rightarrow BC$ , então  $A \rightarrow B$  e  $A \rightarrow C$ .

Prova:

$A \rightarrow AB$  e  $AB \rightarrow BC$  (regra aumento),

logo  $A \rightarrow BC$  (regra transitividade).

# Dependência Funcional

- **Autoderminação:**  $A \rightarrow A$ .  
Prova: regra automática.
- **Composição:** se  $A \rightarrow B$  e  $C \rightarrow D$ , então  $AC \rightarrow BD$ .  
Prova: exercício.

# Dependência Funcional

- Exemplo de uso das propriedades
  - Considere a relação R com os atributos A, B, C, D, E, F e as DFs:
    - $A \rightarrow BC$
    - $B \rightarrow E$
    - $CD \rightarrow EF$
    - Mostre que a DF  $AD \rightarrow F$  é válida para R e, portanto, é um membro do fecho do conjunto dado.

# Dependência Funcional

## ■ Solução:

1.  $A \rightarrow BC$  (dada)
2.  $A \rightarrow C$  (1, decomposição)
3.  $AD \rightarrow CD$  (2, aumento)
4.  $CD \rightarrow EF$  (dada)
5.  $AD \rightarrow EF$  (3 e 4, transitividade)
6.  $AD \rightarrow F$  (5, decomposição)

# Normalização

- O processo de Normalização, proposto primeiramente por Codd, faz uma série de testes para certificar se um Esquema Relacional satisfaz a uma Forma Normal.
- Cada Relação é avaliada e decomposta em novas Relações, se necessário. Projeto Relacional por Análise.
- Inicialmente, Codd propôs três formas normais.

# Normalização

- Conseqüências:
  - Problemas de anomalias e inconsistências diminuem.
  - Relações simplificadas e estrutura regular.
  - Aumento da integridade dos dados.
  - Necessidade de realização de junções.
  - Eventual queda na performance.



# Normalização - 1ª Forma Normal (1FN)

- Uma relação R está na 1FN se e somente se, em todo valor válido dessa relação, cada tupla contém exatamente um valor para cada atributo.
- Todo e qualquer atributo deve ter valor **ATÔMICO** e **INDIVISÍVEL**, ou seja, no modelo relacional não pode haver atributos multivalorados ou conjuntos de atributos.
- Esta FN é considerada parte da definição do Modelo Relacional.

# Normalização - 1ª Forma Normal (1FN)

## 1º FN

- as linhas da tabela são unívocas
- as linhas não contêm itens repetitivos
- os atributos são atômicos

# Normalização - 1FN

**Tabela: PEDIDO (não normalizada)**

NumPed	DataEmis	Fornecedor	CGC	End	CodProd	NomeProd	Qtde	Preço
3	Jan 20	Casa Software	1111111-11	Lapa Rua A	111	Prod 1	10	R\$100.00
					222	Prod 2	44	R\$150.00
					333	Prod 3	50	R\$120.00
4	Feb 10	Computer	2222222-22	Itu 49	222	Prod 4	73	R\$150.00
					333	Prod 5	80	R\$120.00

**Tabela: PEDIDO – sem itens repetidos**

NumPed	DataEmis	Fornecedor	CGC	End	Cod Prod	NomeProd	QtdeProd	PreçoProd
3	Jan 20	Casa Software	1111111-11	Lapa Rua A	111	Prod 1	10	R\$100.00
3	Jan 20	Casa Software	1111111-11	Lapa Rua A	222	Prod 2	44	R\$150.00
3	Jan 20	Casa Software	1111111-11	Lapa Rua A	333	Prod 3	50	R\$120.00
4	Feb 10	Computer	2222222-22	Itu 49	222	Prod 4	73	R\$150.00
4	Feb 10	Computer	2222222-22	Itu 49	333	Prod 5	80	R\$120.00

# Normalização – 1FN

## Processo para obtenção da 1FN

- em cada tabela eliminar grupos repetitivos gerando novas linhas, uma para cada ocorrência de item repetitivo, mantendo os valores dos demais itens
- transformar os atributos compostos em atômicos

# Normalização – 1FN

**Tabela: PEDIDO – sem itens repetidos**

NumPed	DataEmis	Fornecedor	CGC	End	Cod Prod	NomeProd	QtdeProd	PreçoProd
3	Jan 20	Casa Software	1111111-11	Lapa Rua A	111	Prod 1	10	R\$100.00
3	Jan 20	Casa Software	1111111-11	Lapa Rua A	222	Prod 2	44	R\$150.00
3	Jan 20	Casa Software	1111111-11	Lapa Rua A	333	Prod 3	50	R\$120.00
4	Feb 10	Computer	2222222-22	Itu Rua 49	222	Prod 2	73	R\$150.00
4	Feb 10	Computer	2222222-22	Itu Rua 49	333	Prod 3	80	R\$120.00

**Tabela: PEDIDO – com Atributos Atômicos**

NumPed	DataEmis	Fornecedor	CGC	Bairro	Rua	Cod Prod	NomeProd	QtdeProd
3	Jan 20	Casa Software	1111111-11	Lapa	Rua A	111	Prod 1	10
3	Jan 20	Casa Software	1111111-11	Lapa	Rua A	222	Prod 2	44
3	Jan 20	Casa Software	1111111-11	Lapa	Rua A	333	Prod 3	50
4	Feb 10	Computer	2222222-22	Itu	Rua 49	222	Prod 2	73
4	Feb 10	Computer	2222222-22	Itu	Rua 49	333	Prod 3	80

# Normalização – 1FN

## Processo para obtenção da 1FN

- em cada tabela eliminar grupos repetitivos gerando novas linhas, uma para cada ocorrência de item repetitivo, mantendo os valores dos demais itens
- transformar os atributos compostos em atômicos
- definir as chaves candidatas e escolher a chave primária da tabela (unicidade nas linhas)

# Normalização – 1FN

Tabela: PEDIDO – com Chave Primária: NumPed+CodProd

NumPed	Cod Prod	DataEmis	Fornecedor	CGC	Bairro	Rua	NomeProd	QtdeProd
3	111	Jan 20	Casa Software	11111111-11	Lapa	Rua A	Prod 1	10
3	222	Jan 20	Casa Software	11111111-11	Lapa	Rua A	Prod 2	44
3	333	Jan 20	Casa Software	11111111-11	Lapa	Rua A	Prod 3	50
4	222	Feb 10	Computer	22222222-22	Itu	Rua 49	Prod 2	73
4	333	Feb 10	Computer	22222222-22	Itu	Rua 49	Prod 3	80

## Normalização – 1FN

- Deve-se observar que se uma relação R estiver apenas na 1FN (ou seja, não esteja na 2FN, e portanto também não está na 3FN) tem uma estrutura indesejável por uma série de razões.



## Normalização – 2FN

- Uma relação R está na 2FN se e somente se ela está em 1FN e todo atributo não chave é irreduzivelmente dependente da chave primária.

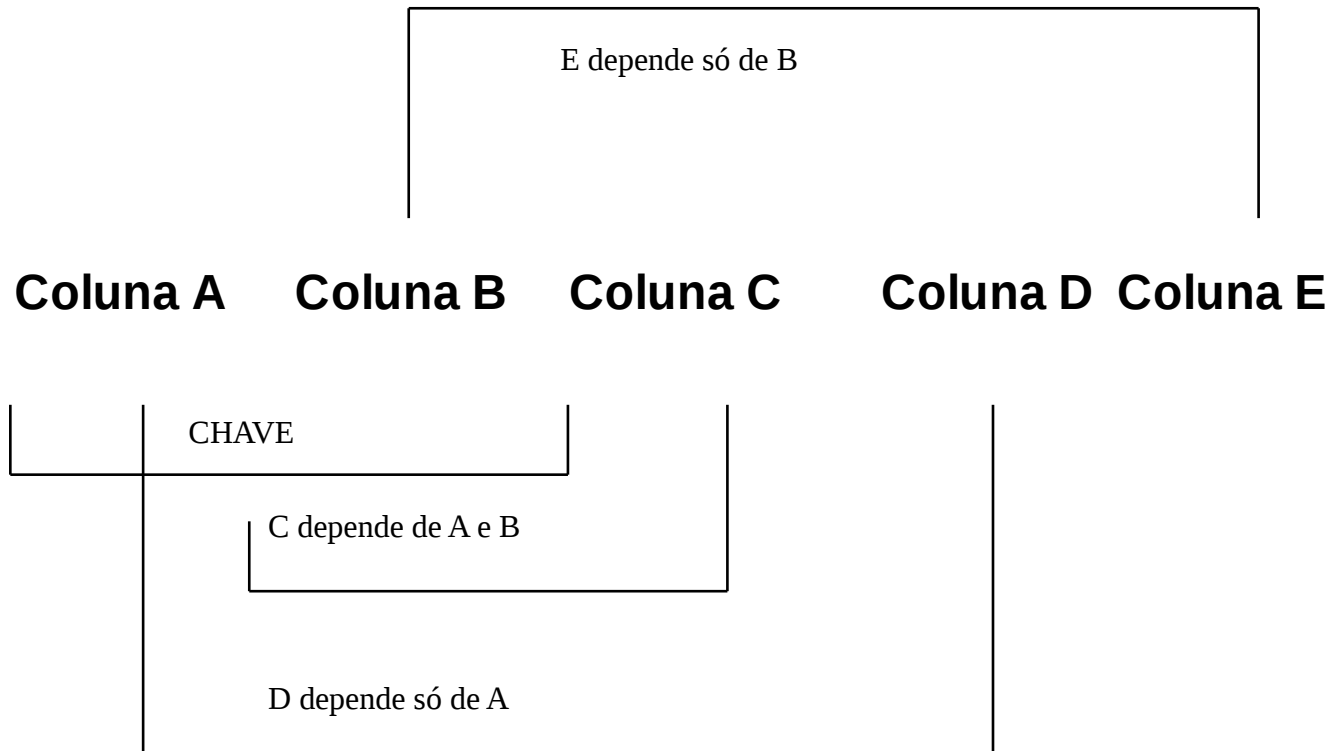
# Normalização – 2FN

## Segunda Forma Normal (2FN)

- está na 1FN
- cada uma das colunas não pertencentes à chave primária não é dependente parcial dessa chave
- cada atributo não-chave é dependente de toda a chave primária.
- a dependência parcial de uma chave só será possível se esta chave for definida com mais de uma coluna
- dizemos que uma coluna é parcialmente dependente da chave se, para que seu valor seja determinado não necessitamos conhecer a chave como um todo

# Normalização – 2FN

Tabela: não normalizada



# Normalização – 2FN

Tabela: normalizada

Coluna A    Coluna B    Coluna C

Coluna A    Coluna D

Coluna B    Coluna E

# Normalização – 2FN

## Processo para obtenção da 2FN

- identificar as colunas que não participam da chave primária da tabela
- para cada uma das colunas identificadas, analisar se seu valor é determinado por parte, ou pela totalidade da chave
- para as colunas dependentes parcialmente:
  - criar novas tabelas onde a chave primária será(ão) a(s) coluna(s) da chave primária original que determinou o valor da coluna analisada
  - excluir da tabela original as colunas dependentes parcialmente da chave

# Normalização – 2FN

**Tabela na 1FN: Pedido**

NumPed	Cod Prod	DataEmis	Fornecedor	CGC	Bairro	Rua	NomeProd	QtdeProd	PreçoProd
3	111	Jan 20	Casa Software	1111111-11	Lapa	Rua A	Prod 1	10	R\$100.00
3	222	Jan 20	Casa Software	1111111-11	Lapa	Rua A	Prod 2	44	R\$150.00
3	333	Jan 20	Casa Software	1111111-11	Lapa	Rua A	Prod 3	50	R\$120.00
4	222	Feb 10	Computer	2222222-22	Itu	Rua 49	Prod 2	73	R\$150.00
4	333	Feb 10	Computer	2222222-22	Itu	Rua 49	Prod 3	80	R\$120.00

**Tabela na 2FN: Pedido**

NumPed	Cod Prod	DataEmis	Fornecedor	CGC	Bairro	Rua	QtdeProd
3	111	Jan 20	Casa Software	1111111-11	Lapa	Rua A	10
3	222	Jan 20	Casa Software	1111111-11	Lapa	Rua A	44
3	333	Jan 20	Casa Software	1111111-11	Lapa	Rua A	50
4	222	Feb 10	Computer	2222222-22	Itu	Rua 49	73
4	333	Feb 10	Computer	2222222-22	Itu	Rua 49	80

**Tabela na 2FN: Produto**

Cod Prod	NomeProd	PreçoProd
111	Prod 1	R\$100.00
222	Prod 2	R\$150.00
333	Prod 3	R\$120.00

## Normalização – 3FN

- Uma relação R está na 3FN se e somente se ela está na 2FN e todo atributo chave é dependente de forma não transitiva da chave primária.

# Normalização – 3FN

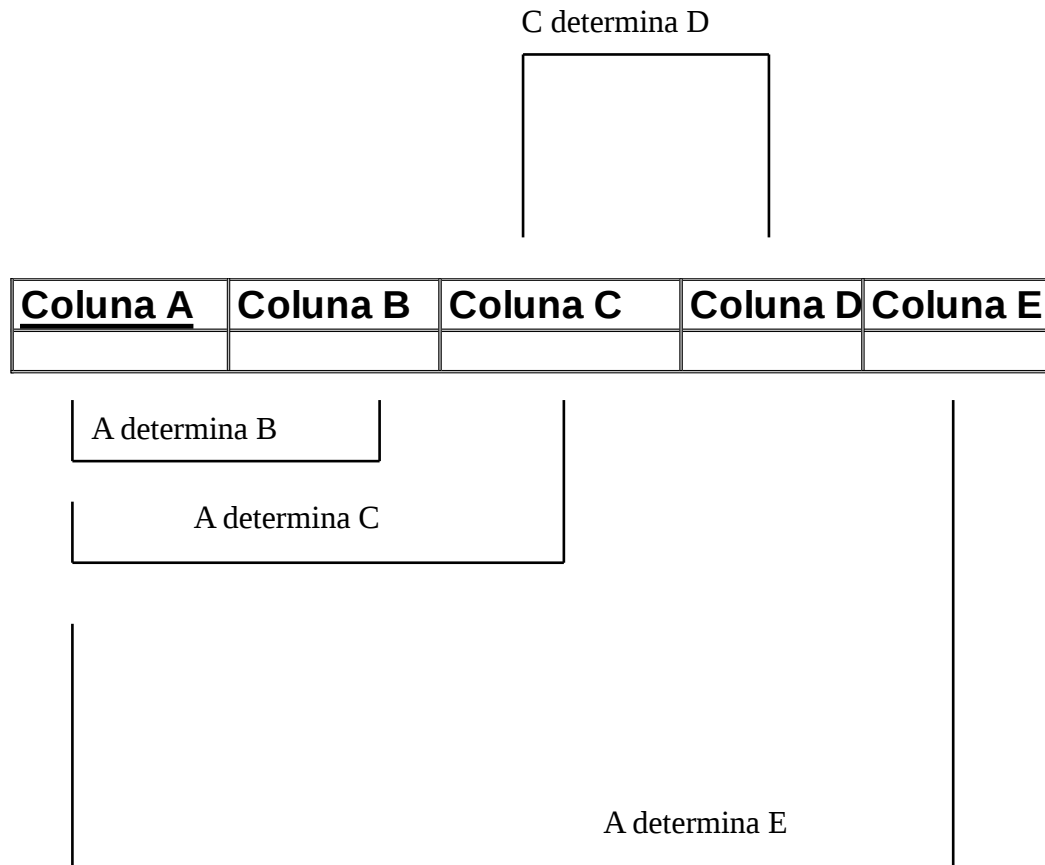
## Terceira Forma Normal (3FN)

- está na 2FN
- nenhuma coluna não pertencente à chave fica determinada transitivamente por esta.
- cada atributo não chave é dependente não transitivo da chave primária.
- a dependência transitiva de uma chave só será possível se a tabela tiver pelo menos duas colunas não pertencentes à chave
- uma coluna depende transitivamente da chave se seu valor é determinado pelo conteúdo de uma coluna não chave



# Normalização – 3FN

Tabela: não normalizada



# Normalização – 3FN

Tabela: normalizada

Coluna A   Coluna B   Coluna C   Coluna E

Coluna C   Coluna D

# Normalização – 3FN

Tabela na 2FN: Pedido

NumPed	Cod Prod	DataEmis	Fornecedor	CGC	Bairro	Rua	QtdeProd
3	111	20/Jan	Casa Software	1111111-11	Lapa	Rua A	10
3	222	20/Jan	Casa Software	1111111-11	Lapa	Rua A	44
3	333	20/Jan	Casa Software	1111111-11	Lapa	Rua A	50
4	222	10/Fev	Computer	2222222-22	Itu	Rua 49	73
4	333	10/Fev	Computer	2222222-22	Itu	Rua 49	80

Tabela na 3FN: Pedido

NumPed	Cod Prod	DataEmis	CGC	QtdeProd
3	111	20/Jan	1111111-11	10
3	222	20/Jan	1111111-11	44
3	333	20/Jan	1111111-11	50
4	222	10/Fev	2222222-22	73
4	333	10/Fev	2222222-22	80

Tabela na 3FN: Produto

Cod Prod	NomeProd	PreçoProd
111	Prod 1	R\$100,00
222	Prod 2	R\$150,00
333	Prod 3	R\$120,00

Tabela na 3FN: Fornecedor

CGC	Fornecedor	Bairro	Rua
1111111-11	Casa Software	Lapa	Rua A
2222222-22	Computer	Itu	Rua 49

## Normalização – FN de Boyce Codd (FNBC)

- Existe uma restrição da 3FN que foi denominada de FNBC.
- Não é a 4FN, pois foi descoberta posteriormente.
- Definição:
  - Uma relação R está em FNBC se, para toda DF não trivial,  $X \rightarrow Y$ , válida para R, então X é uma superchave de R.
- Portanto, as únicas dependências funcionais não triviais são derivadas de chaves da relação.

## Normalização – FN de Boyce Codd (FNBC)

- Excepcionalmente, não se pode ter todas as relações de um BD na FNBC. As exceções, porém, são muito raras. Exemplo:
- Considere a seguinte relação:  
Endereços (Cidade, Rua, CEP), com DFs,  
CidadeRua  $\rightarrow$  CEP e CEP  $\rightarrow$  Cidade.
  - O par Cidade Rua determina o CEP, mas nenhum deles isoladamente determina o CEP.
  - Dado um CEP, a Cidade fica determinada (mas não vice-versa, nem CEP determina Rua).
  - (Cidade Rua) é chave da relação Endereços, mas ela não está na FNBC, pois CEP não é chave ou superchave.

# Normalização – 1FN (anomalias)

- Considere a tabela Empregados, sendo chave primária os campos Matrícula e CodProj.

Matrícula	Nome	CodCargo	NomeCargo	CodProj	DataFim	Horas
120	João	1	Programador	01	17/07/95	37
120	João	1	Programador	08	12/01/96	12
121	Hélio	1	Programador	01	17/07/95	45
121	Hélio	1	Programador	12	21/03/96	107
270	Gabriel	2	Analista	08	12/01/96	10
270	Gabriel	2	Analista	12	21/03/96	38
273	Silva	3	Projetista	01	17/07/95	22
274	Abraão	2	Analista	12	21/03/96	31

# Normalização – 1FN (anomalias)

## ■ Anomalias:

- Inserir: não é possível inserir um empregado sem que este esteja alocado num projeto, nem inserir um projeto sem que haja um empregado trabalhando nele.
- Remover: se for necessário remover um projeto, as informações de empregado que estiverem alocados naquele projeto serão perdidas.
- Atualizar: se um empregado for promovido de cargo, teremos que atualizar os atributos CodCargo e NomeCargo em todas as tuplas nas quais aquele empregado está presente.

# Normalização – 2FN (anomalias)

## EMPREGADO

<u>Matrícula</u>	Nome	CodCargo	NomeCargo
120	João	1	Programador
121	Hélio	1	Programador
270	Gabriel	2	Analista
273	Silva	3	Projetista
274	Abraão	2	Analista

## ALOCAÇÃO

<u>Matrícula</u>	CodProj	Horas
120	01	37
120	08	12
121	01	45
121	08	21
121	12	107
270	08	10
270	12	78
273	01	22
274	12	31

## PROJETO

<u>CodProj</u>	DataFim
01	17/07/95
08	12/01/96
12	21/03/96



# Normalização – 2FN (anomalias)

## ■ Anomalias:

- Inserir: só é possível criar cargos se houver empregados designados para eles.
- Remover: se apagarmos um empregado que ocupa unicamente um cargo na empresa, perderemos a informação do cargo.
- Atualizar: se um cargo mudar de nome, será necessário mudar todas as tabelas em que este cargo aparece.

# Normalização – 3FN (anomalias)

## EMPREGADO

<u>Matrícula</u>	Nome	CodCargo
120	João	1
121	Hélio	1
270	Gabriel	2
273	Silva	3
274	Abraão	2

## CARGO

CodCargo	Nome
1	Programador
2	Analista
3	Projetista

## Normalização – 4FN

- Uma relação está na 4FN se, e somente se, estiver na 3FN e não existirem dependências multivaloradas.
- Método para corrigir:
  - Para cada grupo de repetição separado, gera-se uma nova relação correspondente contendo este grupo de repetição e a chave primária da relação original.
  - Determinar a chave primária da nova relação, a qual será a concatenação da chave primária da relação original com a chave para o grupo de repetição.

# Normalização – 4FN

## ■ Exemplo:

- Vendedor (codVendedor, nomeVendedor, {cliente (codCliente, nomeCliente), parente (nomeParente, parentesco)})
- Dependências funcionais  
codVendedor → nomeVendedor
- Dependências multivaloradas  
codVendedor → cliente (codCliente, nomeCliente)  
codVendedor → parente (nomeParente, parentesco)
- Solução  
vendedor (codVendedor, nomeVendedor)  
vendCliente (codVendedor, codCliente, nomeCliente)  
vendParente (codVendedor, codParente, nomeParente)

# Passos para o projeto de banco de dados relacionais<sup>1</sup>

- Guiado pelo bom-senso, construa um diagrama ER, agrupando os atributos nas tabelas que vão representar as entidades e os relacionamentos do seu banco de dados.
- Construa o diagrama de dependências funcionais para as tabelas propostas no MER (ou um único diagrama de dependências funcionais considerando todos os atributos do seu banco de dados).
- Elimine os atributos repetitivos (se houver), de modo a obter um modelo de dados na 1FN.

# Passos para o projeto de banco de dados relacionais

- Elimine as dependências parciais da chave primária em suas tabelas (se houver), obtendo um projeto na 2FN.
- Elimine as dependências transitivas nas tabelas (se houver), obtendo um esquema na 3FN.

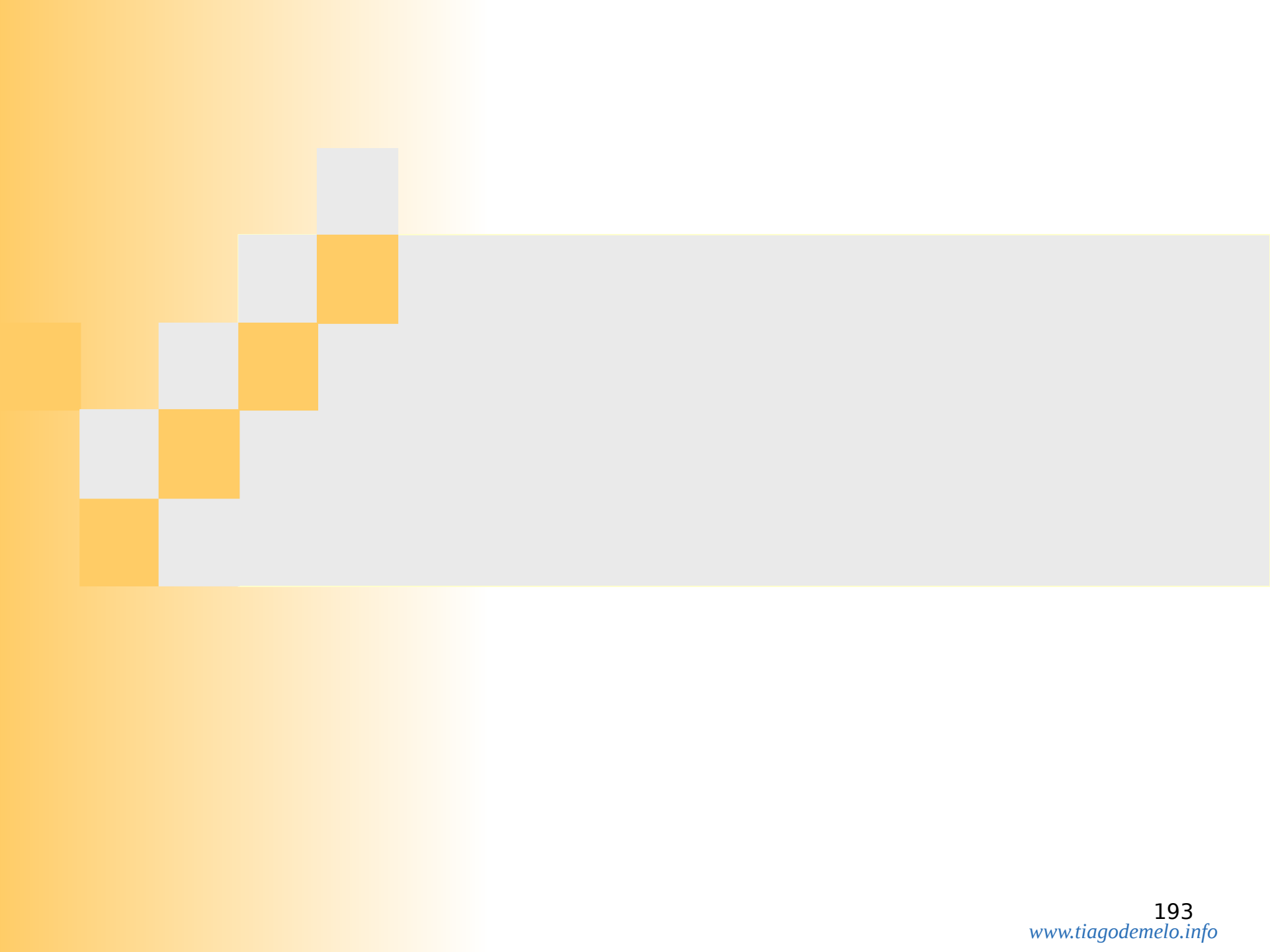
### **Objetivo:**

Apresentar ferramenta livre de banco de dados



# DBDesigner 4





# DBDesigner 4 - Principal

The screenshot displays the DBDesigner 4 interface for a database named 'fabFORCE'. The main workspace shows a complex Entity-Relationship (ER) diagram for an 'OnlineStore' database. The diagram includes several tables and their relationships:

- onlinecustomer**: Attributes include `idonlinecustomer` (INTEGER), `idcreditcard` (INTEGER FK), `name` (VARCHAR(30)), `address1` (VARCHAR(80)), `address2` (VARCHAR(80)), `region` (VARCHAR(45)), `city` (VARCHAR(45)), `zip` (VARCHAR(6)), `phone` (VARCHAR(20)), `creditcardnr` (VARCHAR(20)), and `creditcarddate` (DATE).
- creditcard**: Attributes include `idcreditcard` (INTEGER) and `compang` (VARCHAR(45)).
- onlineorder**: Attributes include `idonlineorder` (INTEGER), `idonlinecustomer` (INTEGER FK), `date` (DATETIME), and `shippingaddress` (TEXT).
- product**: Attributes include `idproduct` (INTEGER), `idproductgroup` (INTEGER FK), `name` (VARCHAR(45)), `ean` (VARCHAR(20)), `price` (FLOAT(10,2)), `info` (TEXT), and `pic` (LONGBLOB).
- productgroup**: Attributes include `idproductgroup` (INTEGER) and `groupname` (VARCHAR(45)).
- forumtopic**: Attributes include `idforumtopic` (INTEGER) and `title` (VARCHAR(80)).
- forumpost**: Attributes include `idforumpost` (INTEGER), `idforumtopic` (INTEGER FK), `idforumpost_parent` (INTEGER FK), `idonlinecustomer` (INTEGER FK), `title` (VARCHAR(45)), `paragraphs` (TEXT), and `createdate` (DATETIME).

Relationships are shown with lines and crow's foot notation symbols. Key relationships include:

- CartRie**: A one-to-many relationship between `onlinecustomer` and `onlineorder`.
- ProductInCartRie**: A one-to-many relationship between `onlineorder` and `product`.
- ProductRie**: A one-to-many relationship between `product` and `productgroup`.
- postHasTop**: A one-to-many relationship between `forumtopic` and `forumpost`.
- Parent**: A one-to-many relationship between `forumpost` and `forumpost` (self-referencing).

On the right side, the 'System Tables' pane shows tables like `weblog`, `webpageclick`, and `webservice`. The 'Datentypen' (Data Types) pane lists common types such as `INTEGER`, `FLOAT`, `VARCHAR`, `DATETIME`, `TEXT`, `LONGBLOB`, and `GUID`. The 'DB Modell' pane on the bottom right lists all tables in the model, including `carthasproduct`, `creditcard`, `Employee`, `forumpost`, `forumtopic`, `News`, `onlinecustomer`, `onlineorder`, `onlineorderhasproduct`, `product`, `productgroup`, `productgroupR`, `weblog`, `webpageclick`, and `webservice`.

The interface also features a zoom menu at the bottom left with options from 23.73% to 400%, and a status bar at the bottom indicating 'Not connected to a Database'.

# DBDesigner 4 – Edição de Tabelas

**Table Editor**

Table Name: onlinecustomer | Table Prefix: Default (no prefix) | Table Type: MYISAM (Standard) | Weak entity:  is n:m Table

Column Name	Data Type	NN	AI	Flags	Default Value	Comments
idonlinecustomer	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL		
idcreditcard	INTEGER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL		
name	VARCHAR(30)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
address1	VARCHAR(80)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
address2	VARCHAR(80)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
region	Varchar(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
city	Varchar(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
zip	VARCHAR(6)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
phone	Varchar(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
creditcardnr	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
creditcarddate	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		

**Indices**

PRIMARY Indexname: PRIMARY | Index Type: PRIMARY | Columns: onlinecustomer

# DBDesigner 4 – Edição de Relacionamentos

The screenshot displays the DB Designer 4 interface with a database model for 'OnlineStore'. The model includes tables like 'onlinecustomer', 'onlineorder', 'carthasproduct', 'productgroup', and 'product'. A 'Relation Editor' dialog box is open, showing the configuration for a 'CartRel' relation. The dialog includes a 'Foreign Keys' table and a 'Reference Definition' section with options for 'Matching', 'On Delete', and 'On Update'. The 'On Update' dropdown menu is open, showing options like 'RESTRICT', 'CASCADE', 'SET NULL', 'NO ACTION', and 'SET DEFAULT'. The 'CASCADE' option is selected.

**Relation Editor**

Relation Name: CartRel  
Relation Kind: 1:n  
Invisible:

Source Field	Dest. Name	Comment
idonlinecustomer	idonlinecustomer	

Reference Definition:  Create Reference Definition

Matching: MATCH FULL  
On Delete: RESTRICT  
On Update: RESTRICT

On Update options: RESTRICT, CASCADE, SET NULL, NO ACTION, SET DEFAULT

# DBDesigner 4 – Edição de Tipos de Dados

The screenshot displays the DBDesigner 4 interface with several key components:

- Table Editor:** A table named 'onlinecustomer' is being edited. The columns and their data types are as follows:

Column Name	Data Type	NN	AI	Flags	Default Value	Comments
idonlinecustomer	INTEGER			<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL		
idcreditcard	INTEGER			<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL		
name	VARCHAR(30)			<input type="checkbox"/> BINARY		
address1	VARCHAR(80)			<input type="checkbox"/> BINARY		
address2	VARCHAR(80)			<input type="checkbox"/> BINARY		
region	Varchar(45)			<input type="checkbox"/> BINARY		
city	Varchar(45)			<input type="checkbox"/> BINARY		
zip	VARCHAR(6)			<input type="checkbox"/> BINARY		
phone	Varchar(20)			<input type="checkbox"/> BINARY		
creditcardnr	VARCHAR(20)			<input type="checkbox"/> BINARY		
creditcarddate	DATE					

- Editor Datatype:** A dialog box for editing the 'FLOAT' data type. It shows a description: "A small (single-precision) floating-point number. Cannot be unsigned. Allowable values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38." It also includes options for 'length decimals', 'Parameters are required', 'Edit values as strings', and 'Enable Physical Datatype Mapping'.
- Navigator & Info:** Shows the current database structure, including tables like 'webpageclick' and 'weblog'.
- Datentypen:** A list of available data types such as INTEGER, FLOAT, VARCHAR, DATETIME, etc.
- DB Modell:** A list of all tables in the model, including 'carthasproduct', 'creditcard', 'Employee', etc.

# DBDesigner 4 – Conexão com Banco de Dados

The screenshot displays the DBDesigner 4 interface with the following components:

- Table Editor:** Shows the 'onlinecustomer' table with columns: idonlinecustomer (INTEGER), idcreditcard (INTEGER), name (VARCHAR(30)), address1 (VARCHAR(80)), address2 (VARCHAR(80)), region (Varchar(45)), city (Varchar(45)), zip (VARCHAR(6)), phone (Varchar(20)), creditcardnr (VARCHAR(20)), and creditcarddate (DATE).
- Indices:** A PRIMARY index is defined on the 'idonlinecustomer' column.
- Editor Datatype:** A dialog box for the 'FLOAT' data type, showing a description: 'A small (single-precision) floating-point number. Cannot be unsigned. Allowable values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38.' It also includes options for 'ZEROFILL', 'Parameters are required', and 'Edit values as strings'.
- Navigator & Info:** Shows a tree view of the database structure.
- Datentypen:** A list of data types including INTEGER, FLOAT, VARCHAR, DATETIME, BOOL, TEXT, LONGBLOB, Varchar(20), Varchar(45), Varchar(255), and GUID.
- DB Modell:** Shows a list of tables in the model, including carthasproduct, creditcard, Employee, forumpost, News, onlinecustomer, onlineorder, onlineorderhasproduct, product, productgroup, weblog, webpageclick, and webserver.
- Diagram:** A partial ER diagram is visible at the bottom, showing a relationship between 'productgroup' and 'ProductgroupRe'.

# DBDesigner 4 – Módulo de Consultas

DB Designer 4 - [DB Model | weboffice.xml]

File Edit Display Database Plugins Options Windows Help

**Lager**

- lager\_artikel\_einheit:
  - idlager\_artikel\_einheit: INTEGER
  - einheit: INTEGER
  - bez: INTEGER
  - stk: INTEGER
- lager\_artikel\_medium:
  - idlager\_artikel\_medium: INTEGER
  - medium: VARCHAR(45)
  - medium\_bez: VARCHAR(2)
- lager\_artikel\_baugruppe:
  - idlager\_artikel\_baugruppe: INTEGER
  - baugruppe: VARCHAR(45)
  - baugruppe\_bez: VARCHAR(4)
- lager:
  - idlager: INTEGER
  - lager: VARCHAR(45)
- lager\_position:
  - idlager\_position: INTEGER
  - idlager: INTEGER (FK)
  - position: VARCHAR(20)
- lager\_hat\_artikel:
  - idlager: INTEGER (FK)
  - idlager\_position: INTEGER (FK)
  - idlager\_artikel: INTEGER (FK)
  - barcode: VARCHAR(45)
  - stk: INTEGER
  - eingangam: DATETIME
  - ausgangam: DATETIME

**Adressen**

- adresse:
  - idadresse: INTEGER
  - idwebuser: INTEGER (FK)
  - idland: INTEGER (FK)
  - idadresse\_kat: INTEGER (FK)
  - idadresse\_subkat: INTEGER (FK)
  - idadresse\_anrede: INTEGER (FK)
  - idadresse\_titel: INTEGER (FK)
  - name: VARCHAR(45)
  - vornname: VARCHAR(45)
  - firmazusatz: VARCHAR(255)
  - ort: VARCHAR(45)
  - strasse: VARCHAR(45)
  - istlieferant: INTEGER
  - tel: VARCHAR(45)
  - telvorwahl: VARCHAR(6)
  - fax: VARCHAR(45)
  - mobil: VARCHAR(45)
  - email: VARCHAR(80)
  - www: VARCHAR(160)
  - ftp: VARCHAR(160)
  - briefansatz: VARCHAR(160)
  - biz: VARCHAR(6)
  - bank: VARCHAR(45)
  - kontonr: VARCHAR(20)
  - aktiv: INTEGER
  - notizen: TEXT
  - adresse\_name
  - name
- plz:
  - plz: VARCHAR(5)
  - idland: INTEGER (FK)
  - ort: VARCHAR(45)
- land:
  - idland: INTEGER
  - land: VARCHAR(45)
  - bez: VARCHAR(2)
  - telvorwahl: VARCHAR(6)

**SELECT**

notizen

bestmengenanschlag: INTEGER

import\_lagerart: VARCHAR(20)

import\_lagerort: VARCHAR(20)

import\_lagerort\_finish: VARCHAR(20)

import\_lagerstand: INTEGER

Importieren

UPDATE

INSERT

DELETE

JOIN Table(s)

LEFT OUTER JOIN

Rel\_01, Rel\_02, Rel\_03, Rel\_04, Rel\_05, Rel\_07, Rel\_10, Rel\_15, Rel\_20, Rel\_51, Rel\_52, Rel\_53, Rel\_54, Rel\_55, Rel\_56, Rel\_57

**Navigator & Info**

Navigator Info

DB Model

**Tables**

- webuser\_druckzeit
- webuser\_emailkonto
- lager\_artikel\_hat\_lieferant
- lager\_artikel\_einheit
- lager\_artikel\_baugruppe
- lager\_artikel\_medium
- lager\_position
- lager\_hat\_artikel
- lager
- lager\_artikel
- doc\_hat\_recht
- recht\_gruppe
- webprinter
- webuser\_hat\_recht
- recht
- webuser\_azplan\_taetigkeit
- webform
- webformsettings
- webuser\_ableitung
- webuser\_azart
- webuser\_azplan
- webuser\_gruppe
- webuser\_timelog
- doc\_order

**SELECT**

```

SELECT idlager, lp.idlager_position,
art.idlager_artikel, l.lager, lp.position, art.artikel,
art.verwendung, art.mindestbestand,
art.import_lagerstand, art.notizen
FROM lager l, lager_position lp, lager_artikel art
WHERE l.idlager=lp.idlager AND
lp.position=art.import_lagerort AND
lp.position<>' ' AND l.idlager=1
ORDER BY l.idlager, lp.position
    
```

idlager	idlager_position	idlager_artikel	lager	position	artikel	verwendung	mindestbestand	im
1	4	1598	Zentrallager	A2/120	E-PNEUMAT. STELLUN DAMPFLANZEN DAMPF		0	
1	5	128	Zentrallager	A2/123	KABEL	x		
1	6	132	Zentrallager	A2/124	DC-SUPPLY	IMPRINTER	1	
1	6	133	Zentrallager	A2/124	AC-SERVO	x	1	
1	7	121	Zentrallager	A2/42	PARALLELE PNP-OPEN (IMPRINTER		0	
1	7	124	Zentrallager	A2/42	SERIELLE EIN/AUSG/x		0	
1	7	131	Zentrallager	A2/42	WÄGEZELLE Z6-2/100 STRANGSTEUERUNG		0	
1	7	134	Zentrallager	A2/42	IMPULSGEBER POG 9E DRUCK-MASCH.		1	
1	7	135	Zentrallager	A2/42	IMPULSGEBER POG 9E DRUCK-MASCH.		2	

75 %



Pointer [Q] (Click on a Table, Note, ... to select the object. Hold Ctrl to select more than one object.)

Connected to Database root@weboffice

# DBDesigner 4 – Módulo de Impressão

The screenshot displays the DBDesigner 4 interface with a database model and a print setup dialog box. The database model shows tables such as 'lager', 'adresse', 'lager\_artikel', 'lager\_artikel\_medium', 'lager\_artikel\_baugruppe', 'lager\_position', and 'lager\_hat\_artikel', along with their attributes and relationships. The print setup dialog is open, showing a preview of the model on a grid of 16 pages (4x4). The dialog includes options for page size (A4), orientation (Portrait/Landscape), and a 'Print Dialog' button.

**Page Setup Dialog Details:**

- Model:  Select Printer ...
- Pagesize: A4 (210x297 mm, 8.26x11.7 inches)
- Orientation:  Portrait  Landscape
- Pages: Horizontal: 4, Vertical: 4
- Print Dialog: 

**Database Model Tables:**

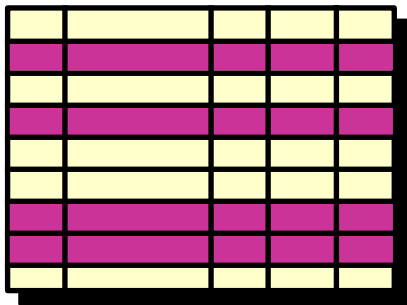
- lager\_artikel\_einheit: idlager\_artikel\_einheit: INTEGER, einheit: INTEGER, bez: INTEGER, stk: INTEGER
- lager\_artikel: idlager\_artikel: INTEGER, idlager\_artikel\_einheit: INTEGER (FK), idadresse\_lieferant: INTEGER (FK), idadresse\_hersteller: INTEGER (FK), idlager\_artikel\_medium: INTEGER (FK), idlager\_artikel\_baugruppe: INTEGER (FK), artikelnr: VARCHAR(45), lieferantname: VARCHAR(30)
- adresse: idadresse: INTEGER, idwebuser: INTEGER (FK), idland: INTEGER (FK), idadresse\_kat: INTEGER (FK), idadresse\_subkat: INTEGER (FK), idadresse\_anrede: INTEGER (FK), idadresse\_tel: INTEGER (FK), name: VARCHAR(45), vorname: VARCHAR(45), firmazusatz: VARCHAR(255)
- lager\_artikel\_medium: idlager\_artikel\_medium: INTEGER, medium: VARCHAR(45), medium\_bez: VARCHAR(45)
- lager\_artikel\_baugruppe: idlager\_artikel\_baugruppe: INTEGER, baugruppe: VARCHAR(45), baugruppe\_bez: VARCHAR(45)
- lager: idlager: INTEGER, lager: VARCHAR(45)
- lager\_position: idlager\_position: INTEGER, idlager: INTEGER (FK), position: VARCHAR(20)
- lager\_hat\_artikel: idlager\_hat\_artikel: INTEGER, idlager\_position: INTEGER (FK), idlager\_artikel: INTEGER (FK), barcode: VARCHAR(45), stk: INTEGER, eingangam: DATETIME, ausgangam: DATETIME



### Objetivos:

- Listar os recursos das instruções SELECT SQL
- Executar uma instrução SELECT básica

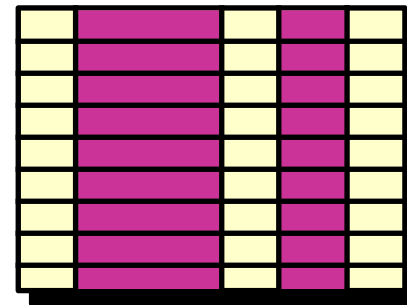
Seleção



A 6x5 grid representing a table. The second, third, and fifth rows are highlighted in pink, indicating they are the result of a selection operation.

Tabela 1

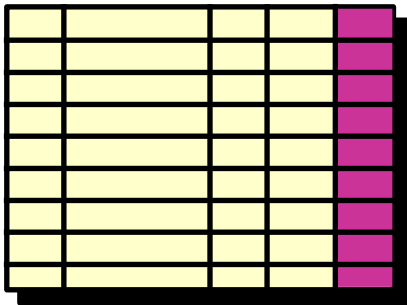
Projeção



A 6x5 grid representing a table. The second, third, and fifth columns are highlighted in pink, indicating they are the result of a projection operation.

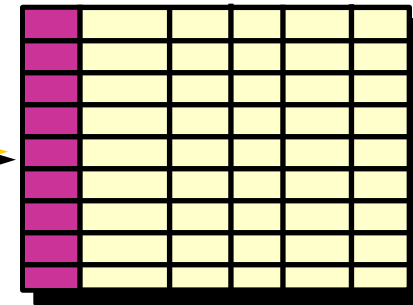
Tabela 1

Junção



A 6x5 grid representing a table. The fifth column is highlighted in pink, indicating it is the result of a join operation.

Tabela 1



A 6x5 grid representing a table. The first column is highlighted in pink, indicating it is the result of a join operation.

Tabela 2

```
SELECT    [DISTINCT] {*, coluna [apelido], ...}  
FROM      tabela;
```

- SELECT identifica **que** colunas.
- FROM identifica **qual** tabela.

- Instruções SQL não fazem distinção entre maiúsculas e minúsculas.
- Instruções SQL podem estar em uma ou mais linhas.
- Palavras-chave não podem ser abreviadas ou divididas entre as linhas.
- Normalmente, as cláusulas são colocadas em linhas separadas.
- Guias e endentações são usadas para aperfeiçoar a legibilidade.

## Seleccionando Todas as Colunas

```
SQL> SELECT *  
2 FROM dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

## Seleccionando Colunas Específicas

```
SQL> SELECT deptno, loc  
2 FROM dept;
```

DEPTNO	LOC
10	NEW YORK
20	DALLAS
30	CHICAGO
40	BOSTON

–Justificada default

- Esquerda: Dados de caractere e data
- Direita: Dados numéricos

–Exibição default: Letra maiúscula

- Criar expressões com dados NUMBER e DATE usando operadores aritméticos

Operador	Descrição
+	Adicionar
-	Subtrair
*	Multiplicar
/	Dividir



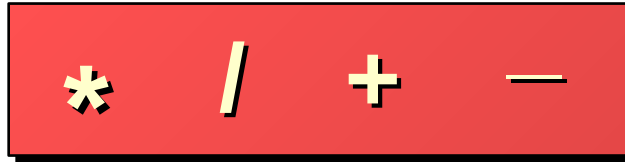
## Usando Operadores Aritméticos

```
SQL> SELECT ename, sal, sal+300  
2 FROM emp;
```

ENAME	SAL	SAL+300
-----	-----	-----
KING	5000	5300
BLAKE	2850	3150
CLARK	2450	2750
JONES	2975	3275
MARTIN	1250	1550
ALLEN	1600	1900
...		

14 rows selected.

## Precedência do Operador



- A multiplicação e a divisão têm prioridade sobre a adição e a subtração.
- Os operadores com a mesma prioridade são avaliados da esquerda para a direita.
- Os parênteses são usados para forçar a avaliação e para esclarecer as instruções.

## Precedência do Operador

```
SQL> SELECT ename, sal, 12*sal+100  
2 FROM emp;
```

ENAME	SAL	12*SAL+100
-----	-----	-----
KING	5000	60100
BLAKE	2850	34300
CLARK	2450	29500
JONES	2975	35800
MARTIN	1250	15100
ALLEN	1600	19300
...		

14 rows selected.

## Usando Parênteses

```
SQL> SELECT ename, sal, 12*(sal+100)
       2 FROM emp;
```

ENAME	SAL	12*(SAL+100)
-----	-----	-----
KING	5000	61200
BLAKE	2850	35400
CLARK	2450	30600
JONES	2975	36900
MARTIN	1250	16200
...		

14 rows selected.

## Definindo um Valor Nulo

–Um valor nulo não está disponível, não é atribuído, é desconhecido ou não é aplicável.

–Um valor nulo não é o mesmo que um zero ou um espaço em branco.

```
SQL> SELECT ename, job, sal, comm  
2 FROM emp;
```

ENAME	JOB	SAL	COMM
-----	-----	-----	-----
KING	PRESIDENT	5000	
BLAKE	MANAGER	2850	
...			
TURNER	SALESMAN	1500	0
...			

14 rows selected.

## Valores Nulos nas Expressões Aritméticas

Expressões aritméticas contendo um valor nulo são avaliadas como nulo.

```
SQL> select  ename, 12*sal+comm  
2  from    emp  
3  WHERE   ename='KING';
```

ENAME	12*SAL+COMM
-----	-----
KING	

## Definindo um Apelido de Coluna

- Renomeia um cabeçalho de coluna
- É útil para cálculos
- Segue imediatamente o nome da coluna
- Palavra-chave **AS** opcional entre o nome da coluna e o apelido
- Necessita de aspas duplas caso contenha espaços ou caracteres especiais ou faça distinção entre maiúsculas e minúsculas

## Usando Apelidos de Coluna

```
SQL> SELECT ename AS name, sal salary
2 FROM emp;
```

```
NAME                SALARY
-----
...
```

```
SQL> SELECT ename "Name",
2          sal*12 "Annual Salary"
3 FROM emp;
```

```
Name                Annual Salary
-----
...
```



- Concatena colunas ou strings de caractere a outras colunas
- É representado por duas barras Verticais - ||
- Cria uma coluna resultante que é uma expressão de caracteres

## Usando um Operador de Concatenação

```
SQL> SELECT  ename||job AS "Employees"  
2 FROM      emp;
```

```
Employees  
-----  
KINGPRESIDENT  
BLAKEMANAGER  
CLARKMANAGER  
JONESMANAGER  
MARTINSALESMAN  
ALLENSALESMAN  
...  
14 rows selected.
```

- Uma literal é um caractere, um número ou uma data incluída na lista SELECT.
- Os valores literais de caractere e data devem estar entre aspas simples.
- Cada string de caractere é gerada uma vez para cada linha retornada.

## Usando Strings Literais de Caracteres

```
SQL> SELECT ename || ' is a ' || job  
2          AS "Employee Details"  
3 FROM    emp;
```

### Employee Details

```
-----  
KING is a PRESIDENT  
BLAKE is a MANAGER  
CLARK is a MANAGER  
JONES is a MANAGER  
MARTIN is a SALESMAN  
...  
14 rows selected.
```

## Linhas Duplicadas

- A exibição default das consultas é de todas as linhas, incluindo linhas duplicadas.

```
SQL> SELECT deptno  
2 FROM emp;
```

```
DEPTNO
```

```
-----
```

```
10
```

```
30
```

```
10
```

```
20
```

```
...
```

```
14 rows selected.
```

## Eliminando Linhas Duplicadas

Elimine linhas duplicadas usando a palavra-chave DISTINCT na cláusula SELECT.

```
SQL> SELECT DISTINCT deptno  
2 FROM emp;
```

DEPTNO
-----
10
20
30

## Exibindo a Estrutura de Tabela

Use o comando DESCRIBE do SQL\*Plus para exibir a estrutura de uma tabela.

```
DESC[RIBE] nome da tabela
```

```
SQL> DESCRIBE dept
```

Name	Null?	Type
-----	-----	-----
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

### Objetivos:

- Limitar linhas recuperadas por uma consulta
- Classificar linhas recuperadas por uma consulta

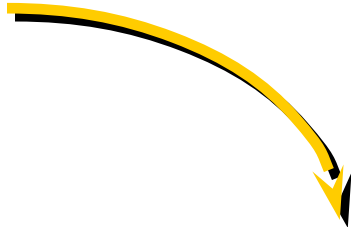


## Limitando Linhas Usando uma Seleção

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...recuperar todos os funcionários do departamento 10"



EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

## Limitando Linhas Selecionadas

–Restringe as linhas retornadas usando a cláusula WHERE.

```
SELECT      [DISTINCT] { * | coluna [apelido], ... }  
FROM        tabela  
[WHERE      condição(ões)];
```

–A cláusula WHERE segue a cláusula FROM.

## Usando a Cláusula WHERE

```
SQL> SELECT ename, job, deptno  
2 FROM emp  
3 WHERE job='CLERK';
```

ENAME	JOB	DEPTNO
JAMES	CLERK	30
SMITH	CLERK	20
ADAMS	CLERK	20
MILLER	CLERK	10

## Strings de Caractere e Datas

- As strings de caractere e valores de data aparecem entre aspas simples.
- Os valores de caractere fazem distinção entre maiúsculas e minúsculas e os valores de data diferenciam formatos.
- O formato de data default é DD-MON-YY.

```
SQL> SELECT  ename, job, deptno  
2 FROM      emp  
3 WHERE     ename = 'JAMES';
```

## Operadores de Comparação

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

## Usando Operadores de Comparação

```
SQL> SELECT ename, sal, comm  
2 FROM emp  
3 WHERE sal<=comm;
```

ENAME	SAL	COMM
-----	-----	-----
MARTIN	1250	1400



## Outros Operadores de Comparação

Operador	Significado
<b>BETWEEN ...AND...</b>	Entre dois valores (inclusive)
<b>IN(list)</b>	Vincula qualquer um de uma lista de valores
<b>LIKE</b>	Vincula um padrão de caractere
<b>IS NULL</b>	É um valor nulo
<b>IS NOT NULL</b>	Não é um valor nulo

## Usando o Operador BETWEEN

Use o operador BETWEEN para exibir linhas baseadas em uma faixa de valores.

```
SQL> SELECT  ename, sal
  2  FROM    emp
  3  WHERE   sal BETWEEN 1000 AND 1500;
```

ENAME	SAL		
-----	-----		
MARTIN	1250	↑	↑
TURNER	1500	Limite inferior	Limite superior
WARD	1250		
ADAMS	1100		
MILLER	1300		



## Usando o Operador IN

Use o operador IN para testar os valores de uma lista.

```
SQL> SELECT empno, ename, sal, mgr
2 FROM emp
3 WHERE mgr IN (7902, 7566, 7788);
```

EMPNO	ENAME	SAL	MGR
7902	FORD	3000	7566
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788

## Usando o Operador LIKE

- Use o operador LIKE para executar pesquisas curinga de valores de string de pesquisa válidos.
- As condições de pesquisa podem conter caracteres literais ou números.
  - % denota zero ou muitos caracteres.
  - \_ denota um caractere.

```
SQL> SELECT  ename
2 FROM      emp
3 WHERE     ename LIKE 'S%';
```

## Usando o Operador LIKE

–Você pode combinar caracteres de vinculação de padrão.

```
SQL> SELECT   ename  
      2 FROM    emp  
      3 WHERE   ename LIKE '_A%';
```

```
ENAME
```

```
-----
```

```
MARTIN
```

```
JAMES
```

```
WARD
```

É possível usar o identificador ESCAPE para procurar por "%" ou "\_".

## Usando o Operador IS NULL

Teste para valores nulos com o operador IS NULL.

```
SQL> SELECT  ename, mgr
  2  FROM    emp
  3  WHERE   mgr IS NULL;
```

ENAME	MGR
-----	-----
KING	

## Operadores Lógicos

Operador	Significado
AND	Retorna TRUE se as condições de componentes forem TRUE
OR	Retorna TRUE se uma condição de componente for TRUE
NOT	Retorna TRUE se a condição seguinte for FALSE

## Usando o Operador AND

AND exige que ambas as condições sejam TRUE.

```
SQL> SELECT empno, ename, job, sal
2 FROM emp
3 WHERE sal >= 1100
4 AND job = 'CLERK';
```

EMPNO	ENAME	JOB	SAL
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

## Usando o Operador OR

OR exige que uma condição seja TRUE.

```
SQL> SELECT empno, ename, job, sal
2 FROM emp
3 WHERE sal >= 1100
4 OR job = 'CLERK';
```

EMPNO	ENAME	JOB	SAL
7839	KING	PRESIDENT	5000
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
...			
7900	JAMES	CLERK	950
...			

14 rows selected.

## Usando o Operador NOT

```
SQL> SELECT ename, job  
2 FROM emp  
3 WHERE job NOT IN ('CLERK', 'MANAGER', 'ANALYST');
```

ENAME	JOB
-----	-----
KING	PRESIDENT
MARTIN	SALESMAN
ALLEN	SALESMAN
TURNER	SALESMAN
WARD	SALESMAN



<b>Ordem de Avaliação</b>	<b>Operador</b>
<b>1</b>	<b>Todos os operadores de comparação</b>
<b>2</b>	<b>NOT</b>
<b>3</b>	<b>AND</b>
<b>4</b>	<b>OR</b>

Sobreponha regras de precedência usando parênteses.

## Regras de Precedência

```
SQL> SELECT ename, job, sal
  2 FROM emp
  3 WHERE job='SALESMAN'
  4 OR job='PRESIDENT'
  5 AND sal>1500;
```

ENAME	JOB	SAL
-----	-----	-----
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
WARD	SALESMAN	1250

## Regras de Precedência

Use parênteses para forçar a prioridade.

```
SQL> SELECT      ename, job, sal
  2  FROM          emp
  3  WHERE         (job='SALESMAN'
  4  OR           job='PRESIDENT' )
  5  AND          sal>1500;
```

ENAME	JOB	SAL
-----	-----	-----
KING	PRESIDENT	5000
ALLEN	SALESMAN	1600

## Cláusula ORDER BY

–Classificar as linhas com a cláusula ORDER BY

- ASC: ordem crescente, default
- DESC: ordem decrescente

–A cláusula ORDER BY vem depois na instrução SELECT.

```
SQL> SELECT      ename, job, deptno, hiredate
  2  FROM          emp
  3  ORDER BY hiredate;
```

ENAME	JOB	DEPTNO	HIREDATE
-----	-----	-----	-----
SMITH	CLERK	20	17-DEC-80
ALLEN	SALESMAN	30	20-FEB-81
...			

14 rows selected.

## Classificando em Ordem Decrescente

```
SQL> SELECT   ename, job, deptno, hiredate
  2  FROM      emp
  3  ORDER BY  hiredate DESC;
```

ENAME	JOB	DEPTNO	HIREDATE
-----	-----	-----	-----
ADAMS	CLERK	20	12-JAN-83
SCOTT	ANALYST	20	09-DEC-82
MILLER	CLERK	10	23-JAN-82
JAMES	CLERK	30	03-DEC-81
FORD	ANALYST	20	03-DEC-81
KING	PRESIDENT	10	17-NOV-81
MARTIN	SALESMAN	30	28-SEP-81
...			

14 rows selected.

## Classificando por Apelido de Coluna

```
SQL> SELECT empno, ename, sal*12 annsal
2 FROM emp
3 ORDER BY annsal;
```

EMPNO	ENAME	ANNSAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7654	MARTIN	15000
7521	WARD	15000
7934	MILLER	15600
7844	TURNER	18000

...

14 rows selected.

## Classificando por Várias Colunas

A ordem da lista ORDER BY é a ordem de classificação.

```
SQL> SELECT      ename, deptno, sal  
 2 FROM          emp  
 3 ORDER BY deptno, sal DESC;
```

ENAME	DEPTNO	SAL
-----	-----	-----
KING	10	5000
CLARK	10	2450
MILLER	10	1300
FORD	20	3000
...		

14 rows selected.

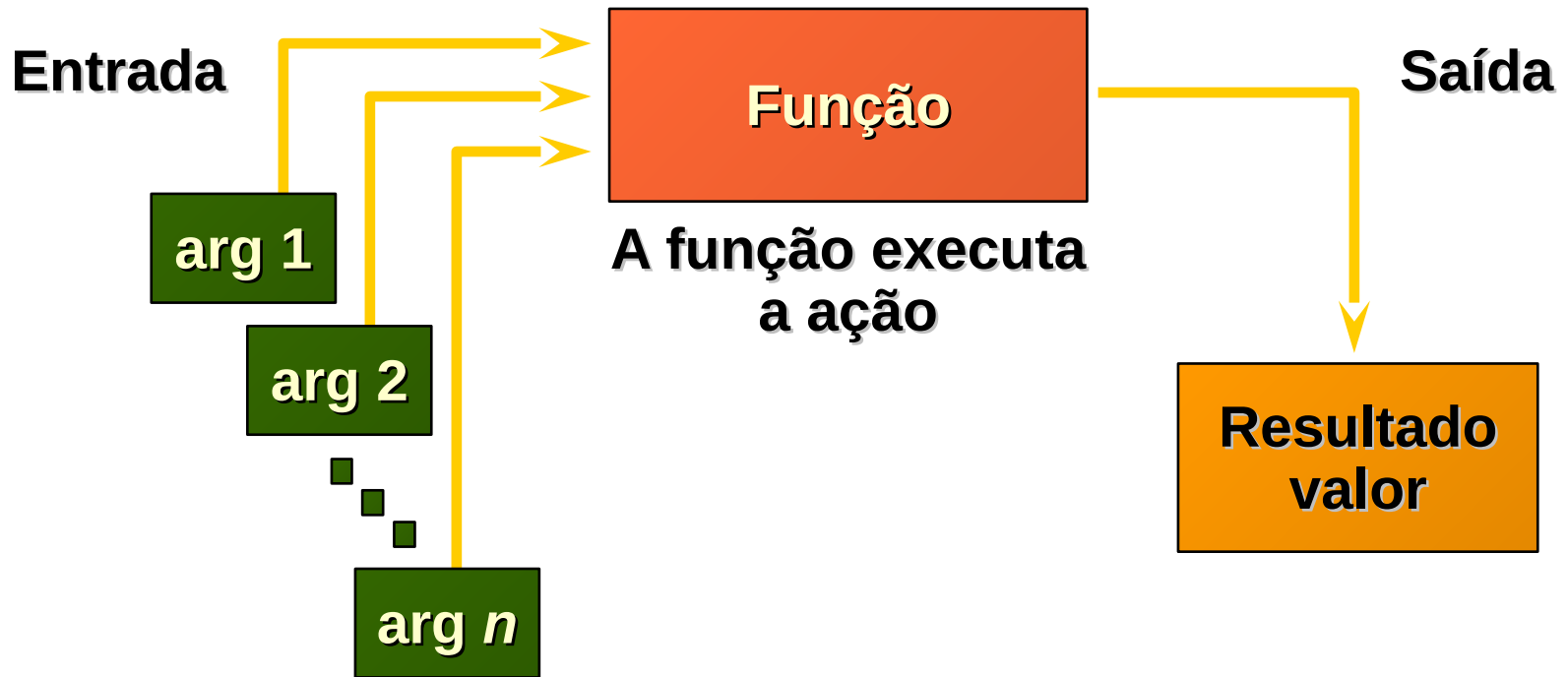
Você pode classificar por uma coluna que não esteja na lista SELECT.

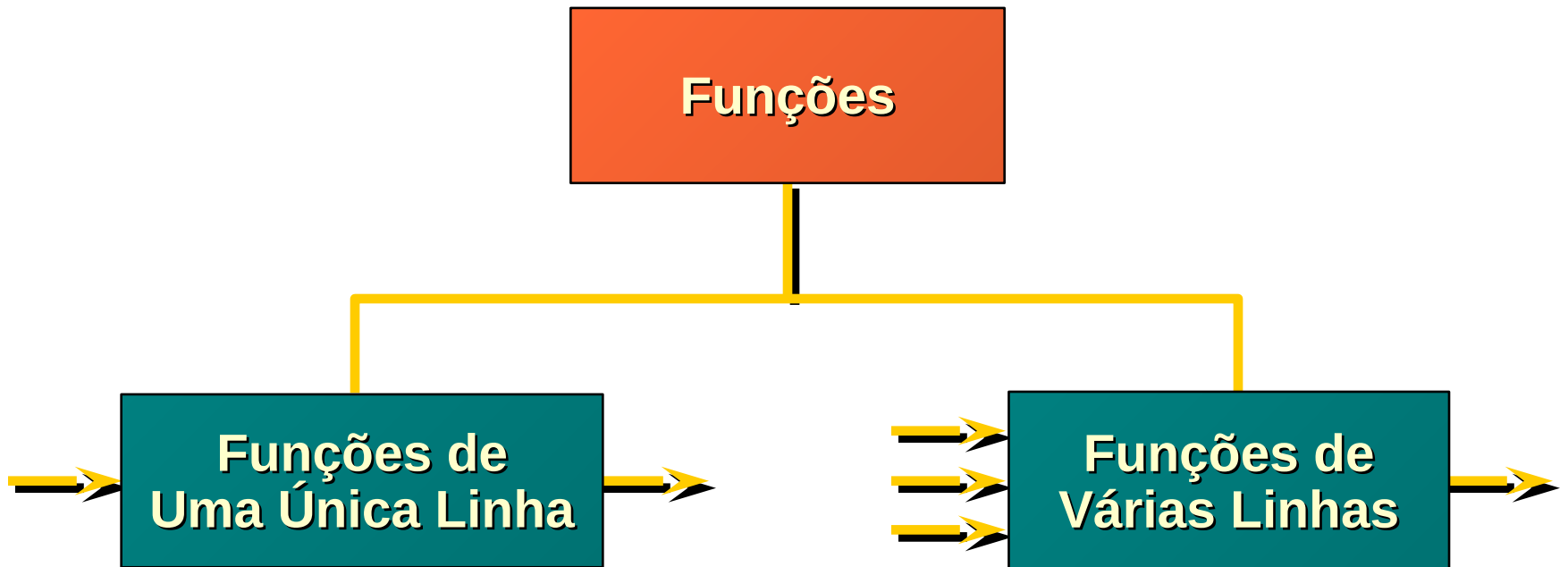
```
SELECT      [DISTINCT] { * | coluna [apelido], ... }  
FROM        tabela  
[WHERE      condição(ões)]  
[ORDER BY   { coluna, expr, apelido } [ASC|DESC]];
```



### Objetivos:

- Descrever vários tipos de funções disponíveis no SQL
- Usar funções de data, número e caractere nas instruções SELECT
- Descrever o uso das funções de conversão

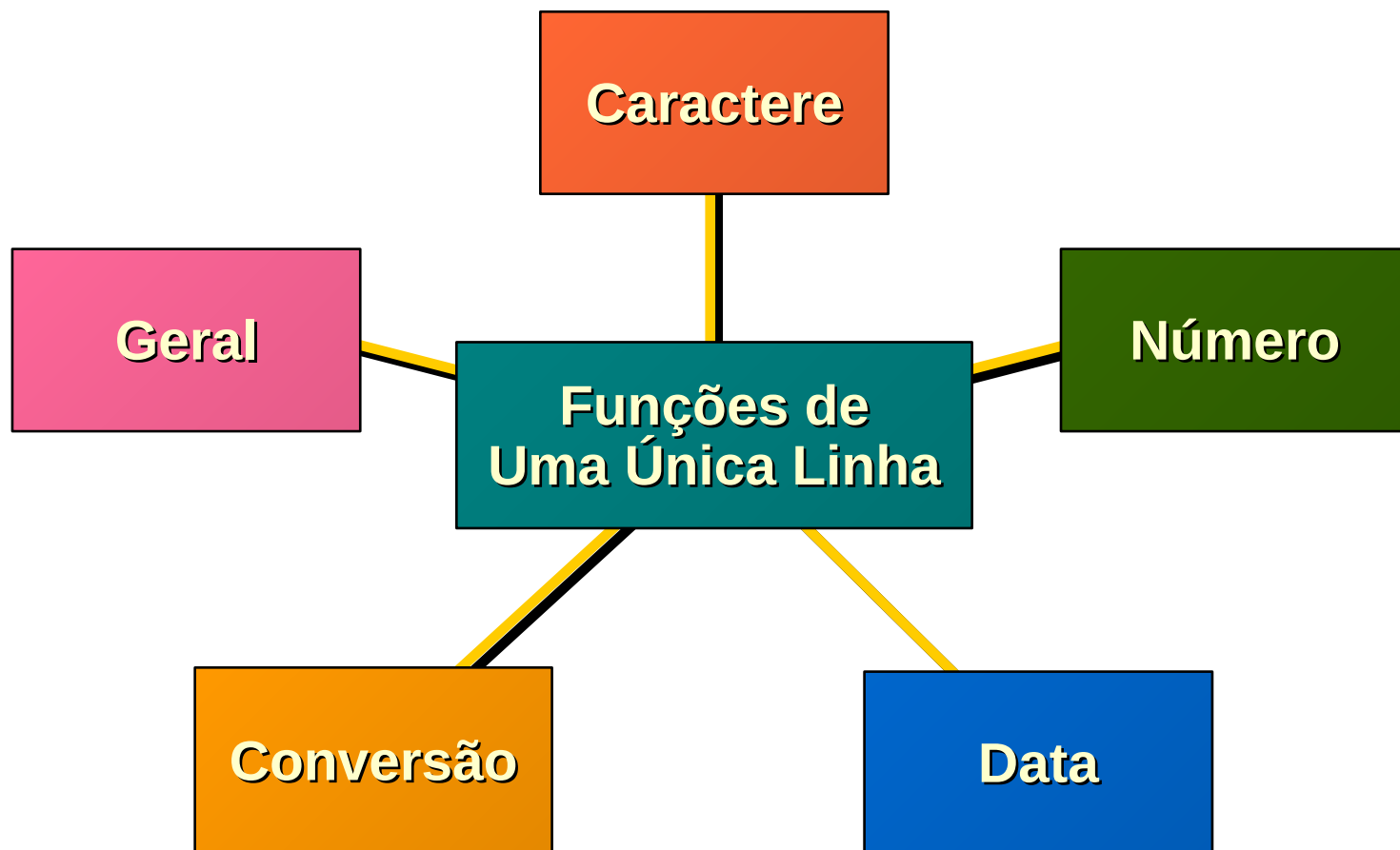




## Funções de Uma Única Linha

- Manipulam itens de dados
- Aceitam argumentos e retornam um valor
- Agem em cada linha retornada
- Retornam um resultado por linha
- Podem modificar o tipo de dados
- Podem ser aninhadas

```
function_name (coluna|expressão, [arg1, arg2, ...])
```



**Funções de caractere**

**Funções de Conversão de Maiúsculas e Minúsculas**

LOWER  
UPPER  
INITCAP

**Funções de manipulação de caractere**

CONCAT  
SUBSTR  
LENGTH  
INSTR  
LPAD  
TRIM

### Objetivos:

- Criar instruções SELECT para obter acesso aos dados a partir de mais de uma tabela usando as junções idênticas e não-idênticas
- Visualizar dados que, em geral, não correspondem a uma condição de junção usando junções externas
- Unindo uma tabela a ela mesma


## Obtendo Dados de Várias Tabelas

EMP

EMPNO	ENAME	...	DEPTNO
7839	KING	...	10
7698	BLAKE	...	30
...			
7934	MILLER	...	10

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



EMPNO	DEPTNO	LOC
7839	10	NEW YORK
7698	30	CHICAGO
7782	10	NEW YORK
7566	20	DALLAS
7654	30	CHICAGO
7499	30	CHICAGO
...		
14 rows selected.		



## O Que É uma Junção?

- Use uma junção para consultar dados a partir de uma ou mais tabelas.

```
SELECT   tabela1.coluna, tabela2.coluna  
FROM     tabela1, tabela2  
WHERE    tabela1.coluna1 = tabela2.coluna2;
```

- Criar uma condição de junção na cláusula WHERE.
- Prefixar o nome da coluna com o nome da tabela quando o mesmo nome da coluna aparecer em mais de uma tabela.

- Um produto cartesiano é formado quando:
  - Uma condição de junção estiver omitida
  - Uma condição de junção estiver inválida
  - Todas as linhas na primeira tabela estão unidas a todas as linhas da segunda tabela
- Para evitar um produto Cartesiano, sempre inclua uma condição de junção válida em uma cláusula WHERE.

# Gerando Produto Cartesiano

EMP (14 linhas)

EMPNO	ENAME	...	DEPTNO
7839	KING	...	10
7698	BLAKE	...	30
...			
7934	MILLER	...	10

DEPT (4 linhas)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



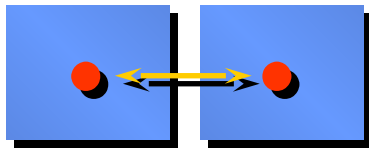
"Produto Cartesiano:  
14\*4=56 linhas"



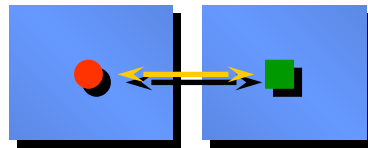
ENAME	DNAME
KING	ACCOUNTING
BLAKE	ACCOUNTING
...	
KING	RESEARCH
BLAKE	RESEARCH
...	
56 rows selected.	

## Tipos de Junções

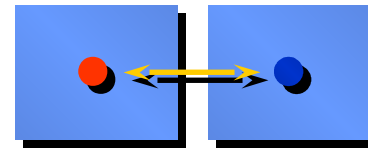
Junção  
idêntica



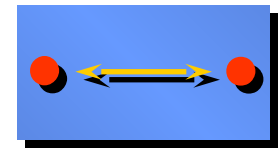
Junção  
não-idêntica



Junção  
externa



Autojunção



# O Que É uma Junção Idêntica?

EMP

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20
...		
14 rows selected.		

Chave estrangeira

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS
...		
14 rows selected.		

Chave primária

## Recuperando Registros com Junções Idênticas

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,  
2         dept.deptno, dept.loc  
3 FROM emp, dept  
4 WHERE emp.deptno=dept.deptno;
```

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7839	KING	10	10	NEW YORK
7698	BLAKE	30	30	CHICAGO
7782	CLARK	10	10	NEW YORK
7566	JONES	20	20	DALLAS
...				

14 rows selected.

## Qualificando Nomes de Coluna Ambíguos

- Use os prefixos de tabela para qualificar nomes de coluna que estão em várias tabelas.
- Melhore o desempenho usando os prefixos de tabela.
- Diferencie colunas que possuem nomes idênticos, mas que residam em tabelas diferentes usando apelidos de coluna.

# Condições de Pesquisa Adicional Usando o Operador AND

EMP

EMPNO	ENAME	DEPTNO
-----	-----	-----
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20
...		
14 rows selected		

DEPT

DEPTNO	DNAME	LOC
-----	-----	-----
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS
...		
14 rows selected		



## Usando Apelidos de Tabela

Simplifique consultas usando apelidos de tabela.

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,  
2      dept.deptno, dept.loc  
3 FROM emp, dept  
4 WHERE emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,  
2      d.deptno, d.loc  
3 FROM emp e, dept d  
4 WHERE e.deptno=d.deptno;
```

# Unindo Mais de Duas Tabelas

CUSTOMER

NAME	CUSTID
-----	-----
JOCKSPORTS	100
TKB SPORT SHOP	101
VOLLYRITE	102
JUST TENNIS	103
K+T SPORTS	105
SHAPE UP	106
WOMENS SPORTS	107
...	...
9 rows selected.	

ORD

CUSTID	ORDID
-----	-----
101	610
102	611
104	612
106	601
102	602
106	
106	
...	...
21 rows selected.	

ITEM

ORDID	ITEMID
-----	-----
610	3
611	1
612	1
601	1
602	1
...	...
61 rows selected.	

## Junções Não-idênticas

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950
...		
14 rows selected.		

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

"o salário na tabela EMP está entre salário inferior e salário superior na tabela SALGRADE"

## Recuperando Registros com Junções Não-idênticas

```
SQL> SELECT e.ename, e.sal, s.grade
2 FROM emp e, salgrade s
3 WHERE e.sal
4 BETWEEN s.losal AND s.hisal;
```

ENAME	SAL	GRADE
-----	-----	-----
JAMES	950	1
SMITH	800	1
ADAMS	1100	1
...		

14 rows selected.

## Junções Externas

EMP		DEPT	
ENAME	DEPTNO	DEPTNO	DNAME
-----	-----	-----	-----
KING	10	10	ACCOUNTING
BLAKE	30	30	SALES
CLARK	10	10	ACCOUNTING
JONES	20	20	RESEARCH
...		...	
		40	OPERATIONS

Nenhum funcionário do departamento OPERATIONS

## Junções Externas

- Use uma junção externa para consultar também todas as linhas que em geral não atendem à condição de junção.
- O operador de junção externo é um sinal de adição (+).

```
SELECT tabela1.coluna, tabela2.coluna  
FROM   tabela1, tabela2  
WHERE  tabela1.coluna(+) = tabela2.coluna;
```

```
SELECT tabela1.coluna , tabela2.coluna  
FROM   tabela1, tabela2  
WHERE  tabela1.coluna = tabela2.coluna(+);
```

## Usando Junções Externas

```
SQL> SELECT    e.ename, d.deptno, d.dname
 2 FROM      emp e,   dept d
 3 WHERE     e.deptno(+) = d.deptno
 4 ORDER BY  e.deptno;
```

```
ENAME          DEPTNO  DNAME
-----
KING            10     ACCOUNTING
CLARK           10     ACCOUNTING
...
                40     OPERATIONS
15 rows selected.
```

# Autojunções

EMP (WORKER)

EMPNO	ENAME	MGR
-----	-----	-----
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

EMP (MANAGER)

EMPNO	ENAME
-----	-----
7839	KING
7839	KING
7839	KING
7698	BLAKE
7698	BLAKE



"MGR na tabela WORKER é igual a EMPNO na tabela MANAGER"



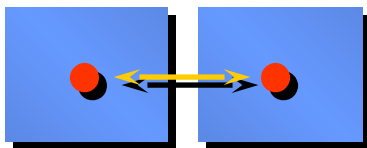
## Unindo uma Tabela a Ela Mesma

```
SQL> SELECT worker.ename||' works for '||manager.ename  
2 FROM emp worker, emp manager  
3 WHERE worker.mgr = manager.empno;
```

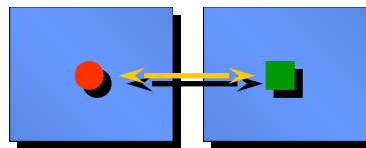
```
WORKER.ENAME||'WORKSFOR'||MANAG  
-----  
BLAKE works for KING  
CLARK works for KING  
JONES works for KING  
MARTIN works for BLAKE  
...  
13 rows selected.
```

```
SELECT  tabela1.coluna, tabela2.coluna
FROM    tabela1, tabela2
WHERE   tabela1.coluna1 = tabela2.coluna2;
```

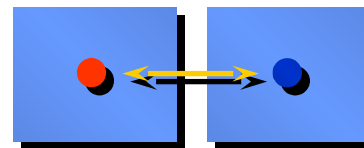
Junção  
idêntica



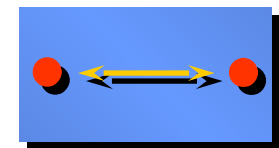
Junção  
não-idêntica



Junção  
externa



Autojunção



## SQL: Lista de Exercício

1. Escreva uma query para mostrar o nome do empregado, número e nome do departamento para todos os empregados
2. Crie uma única lista de todos os cargos que estão no departamento 30.
3. Escreva uma query para mostrar o nome do empregado, nome e localização do departamento de todos os empregados que ganham comissão
4. Mostre o nome do empregado e nome do departamento para todo os empregado que tenha um A em seu nome. Salve em p4q4.sql.
5. Escreva uma query para mostrar o nome, cargo, número e nome do departamento de todos os empregados que trabalham em DALLAS
6. Mostre o nome e número do empregado com o seu respectivo gerente, nome e número. Nomeie as colunas como Employee, emp#, Manager, and Mgr#, respectivamente. Salve em p4q6.sql
7. Modifique p4q6.sql para mostrar todos os empregados, incluindo King, que não tem gerente. Salve em p4q7.sql. Execute.

## SQL: Lista de Exercício

8. Crie uma query que mostre o nome do empregado, número do departamento e todos os empregados que trabalham no mesmo departamento. Nomeie cada coluna apropriadamente.
9. Mostre a estrutura da tabela SALGRADE. Crie uma query que mostre o nome, cargo, nome do departamento, salário e a faixa salarial de todos os empregados.
10. Crie uma query para mostrar o nome e data de contratação de todos empregados contratado após o Blake.
11. Mostre todos os nomes dos empregados com suas datas de contratações, nome dos gerentes e datas de contratações dos empregados que foram contratados antes dos seus gerentes. Nomeie as colunas como Employee, Emp Hiredate, Manager, and Mgr Hiredate, respectivamente.
12. Crie uma query que mostre o nome do empregado e salário como um montante de asteriscos. Cada asterisco significa centenas de dólares. Ordene os dados em ordem descendente de salário. Nomeie a coluna como EMPLOYEE\_AND\_THEIR\_SALARIES.

### Objetivos:

- Identificar as funções de grupo disponíveis
- Descrever o uso de funções de grupo
- Agrupar dados usando a cláusula GROUP BY
- Incluir ou excluir linhas agrupadas usando a cláusula HAVING

## O Que São Funções de Grupo?

As funções de grupo operam em conjuntos de linhas para fornecer um resultado por grupo.

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

"salário  
máximo na  
tabela EMP"

MAX(SAL)
5000

–AVG

–COUNT

–MAX

–MIN

–STDDEV

–SUM

–VARIANCE

```
SELECT      [coluna, ] group_function(coluna)  
FROM        tabela  
[WHERE      condição]  
[GROUP BY   coluna]  
[ORDER BY   coluna];
```



## Usando Funções AVG e SUM

Você pode usar AVG e SUM para dados numéricos.

```
SQL> SELECT  AVG(sal), MAX(sal),  
2           MIN(sal), SUM(sal)  
3 FROM      emp  
4 WHERE     job LIKE 'SALES%';
```

AVG(SAL)	MAX(SAL)	MIN(SAL)	SUM(SAL)
----- 1400	----- 1600	----- 1250	----- 5600

## Usando Funções MIN e MAX

Você pode usar MIN e MAX para qualquer tipo de dados.

```
SQL> SELECT MIN(hiredate), MAX(hiredate)
         2 FROM emp;
```

```
MIN(HIRED   MAX(HIRED
-----
17-DEC-80  12-JAN-83
```

## Usando a Função COUNT

COUNT(\*) retorna o número de linhas em uma tabela.

```
SQL> SELECT COUNT(*)  
2 FROM emp  
3 WHERE deptno = 30;
```

```
COUNT(*)
```

```
-----
```

```
6
```

## Usando a Função COUNT

COUNT(*expr*) retorna o número de linhas não nulas.

```
SQL> SELECT COUNT(comm)
2 FROM emp
3 WHERE deptno = 30;
```

```
COUNT (COMM)
-----
4
```

## Funções de Grupo e Valores Nulos

As funções de grupo ignoram valores nulos na coluna.

```
SQL> SELECT AVG(comm)
2 FROM emp;
```

```
AVG(COMM)
```

```
-----
```

```
550
```

## Usando a Função NVL com Funções de Grupo

A função NVL força as funções de grupo a incluírem valores nulos.

```
SQL> SELECT AVG(NVL(comm, 0))  
2 FROM emp;
```

```
AVG(NVL(COMM, 0))  
-----  
157.14286
```

# Criando Grupos de Dados

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

2916.6667

2175

1566.6667

"salário médio na tabela EMP para cada departamento"

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

## Criando Grupos de Dados: Cláusula GROUP BY

```
SELECT      coluna, group_function(coluna)
FROM        tabela
[WHERE      condição]
[GROUP BY  group_by_expression]
[ORDER BY  coluna];
```

Divida linhas de uma tabela em grupos menores usando a cláusula GROUP BY.



## Usando a Cláusula GROUP BY

- Todas as colunas na lista SELECT que não estejam em funções de grupo devem estar na cláusula GROUP BY

```
SQL> SELECT deptno, AVG(sal)
2 FROM emp
3 GROUP BY deptno;
```

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

## Usando a Cláusula GROUP BY

- A coluna GROUP BY não precisa estar na lista SELECT

```
SQL> SELECT    AVG(sal)
      2 FROM      emp
      3 GROUP BY deptno;
```

```
AVG(SAL)
-----
2916.6667
2175
1566.6667
```

## Agrupando por Mais de Uma Coluna

EMP

DEPTNO	JOB	SAL
10	MANAGER	2450
10	PRESIDENT	5000
10	CLERK	1300
20	CLERK	800
20	CLERK	1100
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
30	SALESMAN	1600
30	MANAGER	2850
30	SALESMAN	1250
30	CLERK	950
30	SALESMAN	1500
30	SALESMAN	1250

"soma de salários na tabela EMP para cada cargo, agrupados por departamento"

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

## Usando a Cláusula GROUP BY em Várias Colunas

```
SQL> SELECT deptno, job, sum(sal)
2 FROM emp
3 GROUP BY deptno, job;
```

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
...		

9 rows selected.

## Consultas Ilegais Usando Funções de Grupo

- Qualquer coluna ou expressão na lista SELECT que não seja uma função agregada deve estar na cláusula GROUP BY.

```
SQL> SELECT deptno, COUNT(ename)
2 FROM emp;
```

**Coluna ausente na cláusula GROUP BY**

```
SELECT deptno, COUNT(ename)
```

```
*
```

```
ERROR at line 1:
```

```
ORA-00937: Nenhuma função de grupo de grupo único
(Not a single-group group function)
```

## Consultas Ilegais Usando Funções de Grupo

- Não é possível usar a cláusula WHERE para restringir grupos.
- Use a cláusula HAVING para restringir grupos.

```
SQL> SELECT deptno, AVG(sal)
2 FROM emp
3 WHERE AVG(sal) > 2000
4 GROUP BY deptno;
```

```
WHERE AVG(sal) > 2000
*
```

```
ERROR at line 3:
```

```
ORA-00934: A função de grupo não é permitida aqui
(Group function is not allowed here)
```

**Não é possível usar a cláusula WHERE para restringir grupos**

# Excluindo Resultados do Grupo

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

5000

3000

2850

"salário máximo por departamento maior do que US\$ 2.900"

DEPTNO	MAX(SAL)
10	5000
20	3000

## Excluindo Resultados do Grupo: Cláusula HAVING

- Use a cláusula HAVING para restringir grupos
  - As linhas são agrupadas.
  - A função de grupo é aplicada.
  - Os grupos que correspondem à cláusula HAVING são exibidos.

```
SELECT      coluna, group_function
FROM        tabela
[WHERE      condição]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   coluna];
```



## Usando a Cláusula HAVING

```
SQL> SELECT deptno, max(sal)
2 FROM emp
3 GROUP BY deptno
4 HAVING max(sal)>2900;
```

DEPTNO	MAX(SAL)
10	5000
20	3000

## Usando a Cláusula HAVING

```
SQL> SELECT      job, SUM(sa1) PAYROLL
  2  FROM          emp
  3  WHERE         job NOT LIKE 'SALES%'
  4  GROUP BY     job
  5  HAVING        SUM(sa1)>5000
  6  ORDER BY     SUM(sa1);
```

JOB	PAYROLL
ANALYST	6000
MANAGER	8275

## Aninhando Funções de Grupo

Exiba o salário médio máximo

```
SQL> SELECT max( avg( sal ) )  
2 FROM emp  
3 GROUP BY deptno;
```

```
MAX( AVG( SAL ) )  
-----  
2916.6667
```

```
SELECT      coluna, group_function(coluna)
FROM        tabela
[WHERE      condição]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   coluna];
```

- Ordem de avaliação das cláusulas:
  - cláusula WHERE
  - cláusula GROUP BY
  - cláusula HAVING

## SQL: Lista de Exercício

Determine se verdadeiro(V) ou falso(F) as seguintes declarações:

1. Funções de grupo trabalham em muitas linhas para produzir um resultado.
2. Funções de grupo usam nulls nos seus cálculos.
3. A cláusula WHERE restringe linhas antes de incluí-las em cálculos de funções de grupos.
4. Mostre o maior, o menor, a soma e a média dos salários de todos os empregados. Nomeie as colunas como Maximum, Minimum, Sum, and Average, respectivamente. Arredonde os resultados para inteiro. Salve em p5q4.sql.
5. Modifique p5q4.sql para mostrar o menor, o maior, a soma e a média dos salários para cada tipo de cargo. Salve em p5q5.sql.
6. Escreva uma query para mostrar o número de empregados com o mesmo cargo.
7. Determine o número de gerentes sem listá-los. Nomeie a coluna como Number of Managers.

## SQL: Lista de Exercício

Determine se verdadeiro(V) ou falso(F) as seguintes declarações:

8. Escreva uma query que mostre a diferença entre o maior e menor salário. Nomeie a coluna como DIFFERENCE.
9. Mostre o número do gerente e o salário mais baixo pago aos funcionários daquele gerente. Exclua o empregado que não possua gerente. Exclua qualquer grupo where o menor salário seja menor que \$1000. Ordene por salário (descendente).
10. Escreva uma query para mostrar o nome do departamento, nome da localização, número de empregados, e média de salário para todos os empregados daquele departamento. Nomeie as colunas como dname, loc, Number of People, and Salary, respectivamente.
11. Crie uma query que mostre o número total de empregados e daquele total, o número que foram contratados em 1980, 1981, 1982, e 1983. Nomeie as colunas de forma apropriada.