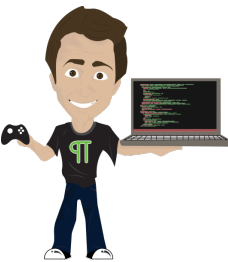




# **Treinamento SQL**

## **Projeto Lógico Transformando Modelo Entidade- Relacionamento para Modelo Relacional**

Prof. Dr. Francisco Isidro Massetto  
[francisco.isidro@techschool.com.br](mailto:francisco.isidro@techschool.com.br)

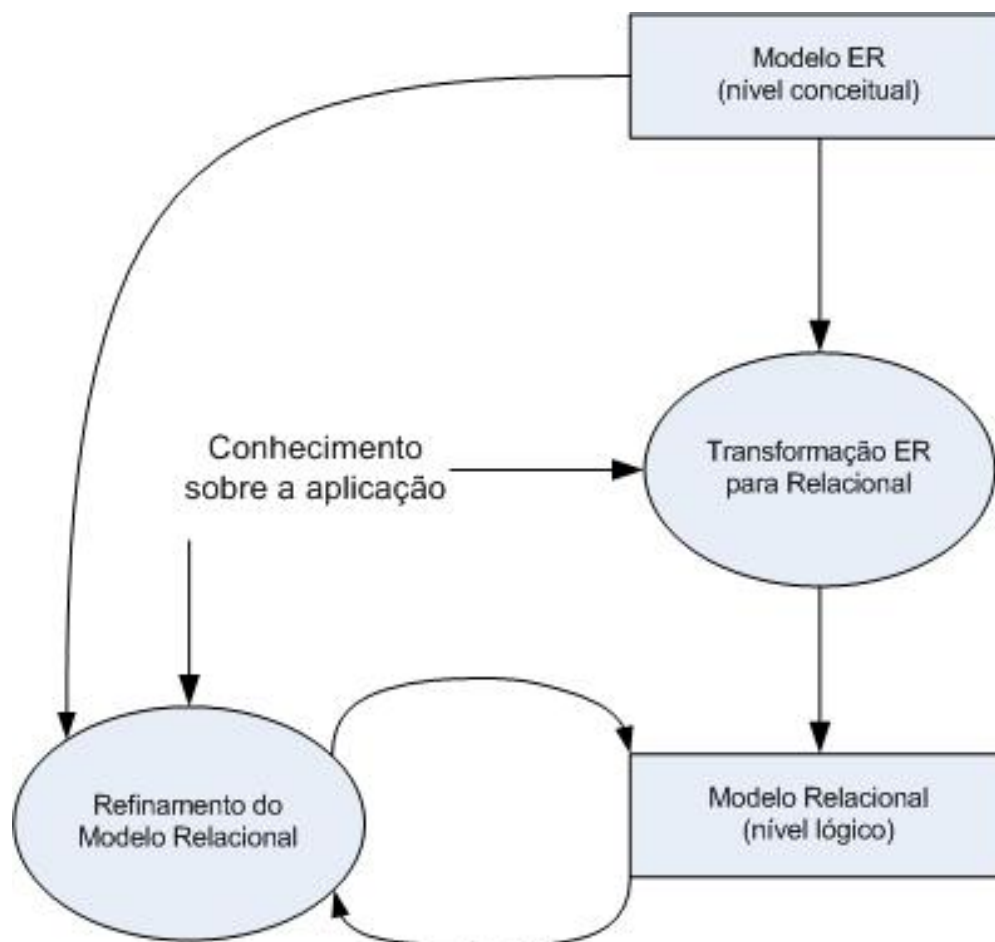


# Agenda

- Visão geral de projeto de Banco de Dados
- Refinamentos, abstrações
- Considerações sobre o projeto lógico
- Regras de transformação
  - Entidades
  - Relacionamentos identificadores
  - Relacionamentos
  - Atributos Multivalorados
  - Especializações, Generalizações



# Visão geral do Projeto de um BD





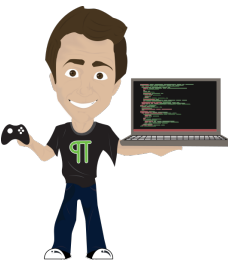
# Considerações sobre o Mapeamento

- Algumas regras
  - Banco de dados com bom desempenho de instruções de consultas e alterações
    - Diminuir acessos a disco – já que é uma operação custosa
  - Banco de dados que simplifique o desenvolvimento e manutenção de aplicações
    - Regras bem claras e que consigam expressar o máximo de informações no modelo



# Princípios para projeto de um Banco de Dados

- Evitar junções excessivas
  - Obviamente operações de associação de tabelas requerem vários acessos a disco
  - Ideal é manter todas as informações relevantes em uma única linha da tabela, pois essas informações são armazenadas de forma contígua no disco
    - Entretanto projetos eficientes de bancos de dados requerem um cuidadoso planejamento sobre quando criar novas tabelas e quando associar valores a campos



# Minimizar o número de chaves

- Obviamente sem comprometer a estrutura de busca de informações da base de dados
  - Isso porque cada chave gera uma estrutura auxiliar de pesquisa
  - Muitas chaves → muitos índices → maior ocupação e maior número de acessos ao disco
- Exemplo:
  - Cliente(codigo, nome, apelido, endereco, telefone)
  - Cliente(codigo, nome, apelido)
  - ClienteEndereco(codigo, endereco, telefone)



# Evitar Campos opcionais

- Um campo opcional é aquele que pode assumir o estado **null** (VAZIO) em seu conteúdo
- Problema maior não é a falta da informação armazenada no valor do campo
  - SGBDs usam técnicas de campos de tamanho variável para resolver esse problema
- Maior problema é manter a consistência na inclusão dos valores por parte da aplicação
  - Quando a obrigatoriedade do preenchimento de um determinado campo depende do valor de outros campos
  - Modelo de dados não consegue atingir esse nível de restrições



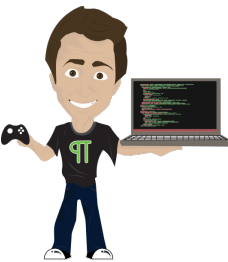
# Implementação Inicial - Entidades

- Idéia básica
  - Cada entidade torna-se uma tabela, cujos atributos são mapeados para campos (colunas)



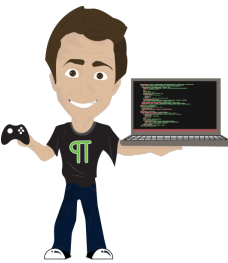
Pessoa(codigo, nome, endereço, nascimento, admissao)



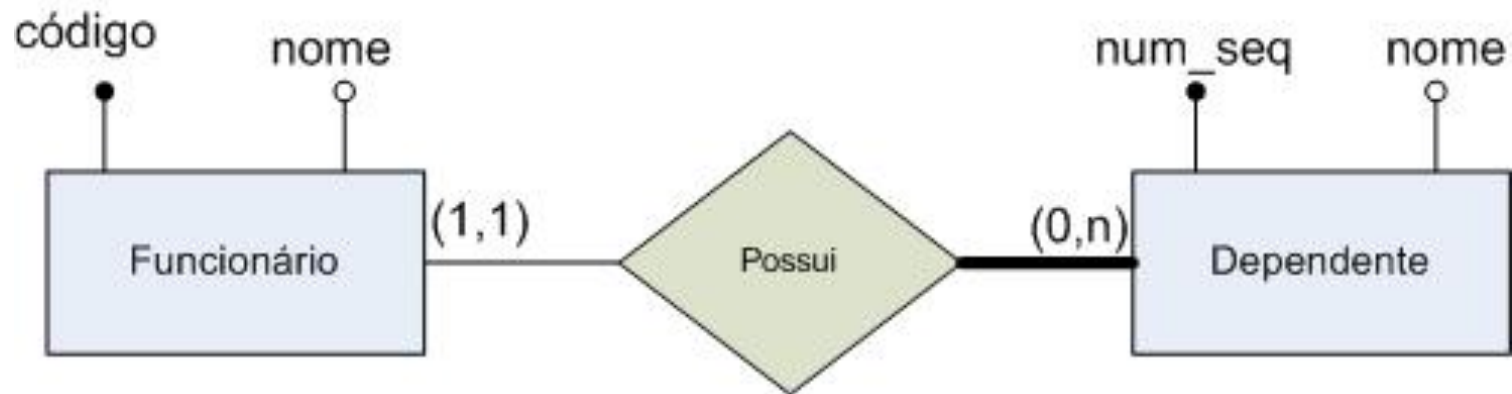


# Algumas convenções

- Nas linguagens de manipulação de bancos de dados, o próprio nome da tabela é usado como identificador da coluna
  - Coluna Nome na tabela Pessoa é referenciada como Pessoa.Nome
- Facilita a compreensão e evita casos tais como
  - NomePessoa
  - NomePess
  - NomeP
- Entretanto existem alguns casos em que usar notações como as relacionadas acima auxiliam a compreensão
  - Uso de chaves estrangeiras (veremos a seguir)

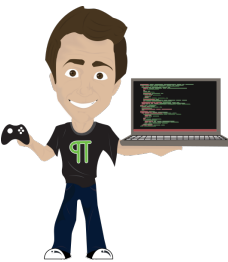


# Relacionamento Identificador



Empregado (codigoEmpregado, nome)

Dependente(codigoEmpregado, num\_seq, nome)

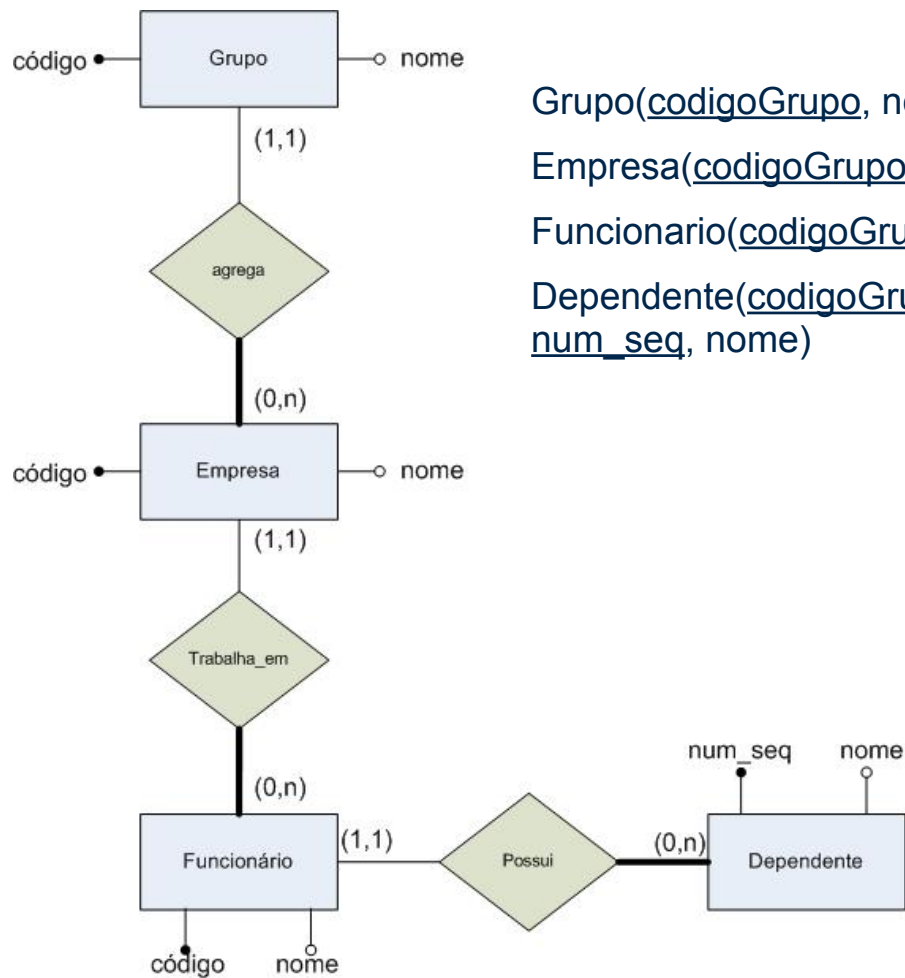


# Relacionamento identificador

- A regra básica é que, para cada identificador externo, seja criada uma coluna (ou várias se o identificador externo for um atributo composto)
- Em alguns casos, atributos de várias entidades devem ser combinados para poder identificar um único registro na tabela com identificadores externos



# Relacionamento Identificador



Grupo(codigoGrupo, nome)

Empresa(codigoGrupo, codigoEmpresa, nome)

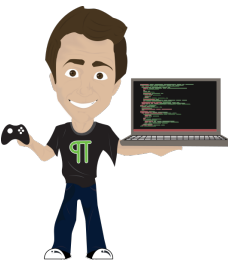
Funcionario(codigoGrupo, codigoEmpresa, codigoFuncionario, nome)

Dependente(codigoGrupo, codigoEmpresa, codigoFuncionario, num\_seq, nome)



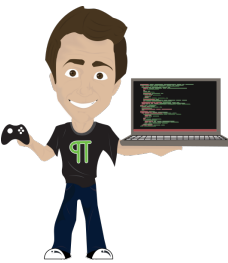
# Relacionamentos

- Tabelas Próprias
- Colunas adicionais em uma entidade
- Fusão de entidades em uma única tabela
- Participação opcional x Participação Obrigatória



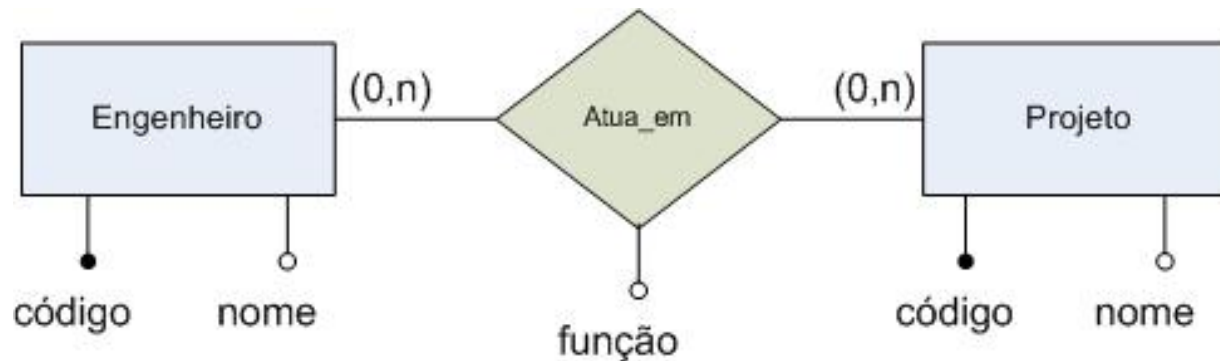
# Relacionamentos

- Tabela própria
  - A tabela resultante do relacionamento é composta de:
    - Colunas correspondentes aos identificadores de cada tabela
    - Colunas correspondentes aos atributos do relacionamento
  - A chave primária é composta dos atributos das tabelas relacionadas e também pelos atributos identificadores do relacionamento (caso existam)



# Relacionamentos

- Tabela Própria



Engenheiro(codigoEngenheiro, Nome)

Projeto(codigoProjeto, Titulo)

Atua\_em(CodigoEngenheiro, CodigoProjeto, Funcao)

CodigoEngenheiro Referencia Engenheiro

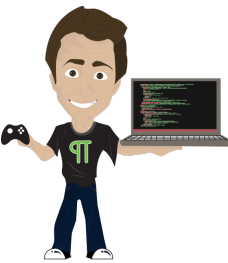
CodigoProjeto Referencia Projeto



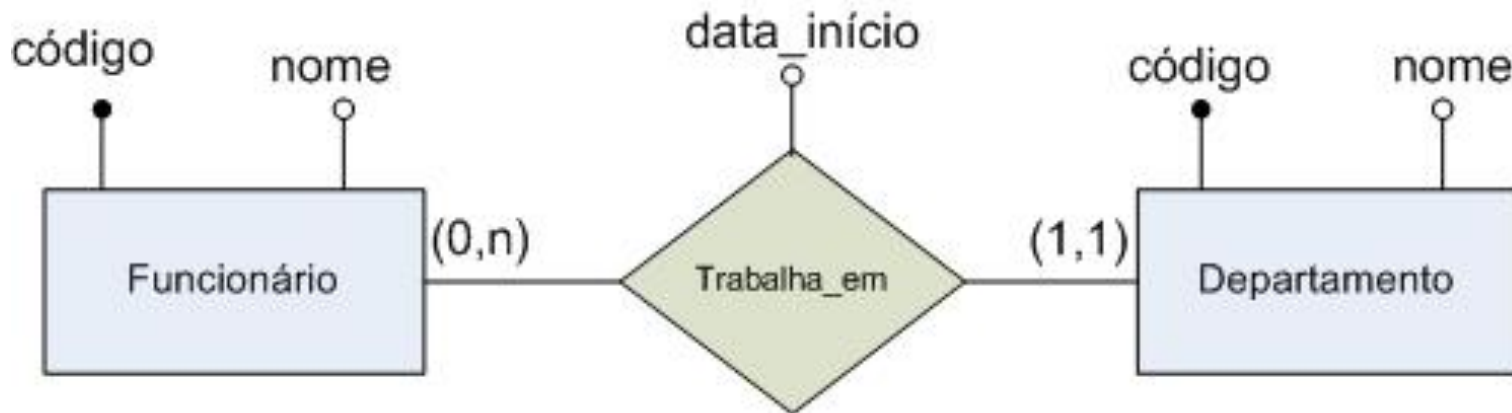
# Relacionamentos – Colunas Adicionais

- Uma das entidades deve ter cardinalidade máxima 1
- Dessa forma, a entidade de cardinalidade máxima 1 tem sua chave incorporada na entidade de cardinalidade máxima N
- Além disso, os atributos do relacionamento também são incorporados na entidade de cardinalidade máxima N



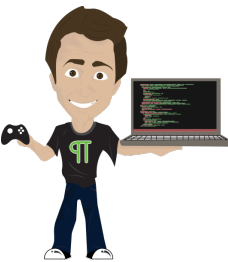


# Relacionamentos – Colunas Adicionais



Departamento(codigoDepartamento, nome)

Funcionário(codigoFuncionario, Nome, codigoDepartamento, data\_inicio)  
codigoDepartamento Referencia Departamento



# Relacionamentos – Fusão de Entidades

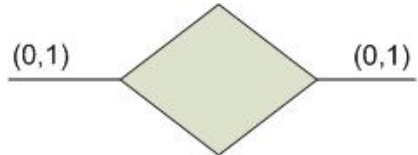
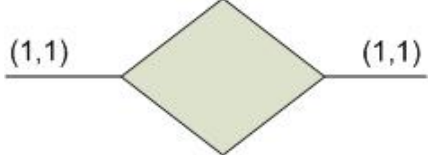
- Apenas para relacionamentos 1:1
- Cria-se apenas uma tabela contendo todos os campos de ambas as entidades



Conferência(codigoConferencia, nomeConferencia, dataNomeacao, nomeComissão)



# Regras para Implementação de Relacionamentos

Tipo Relacionamento	Regra de Implementação		
	Tabela Própria	Adição de Coluna	Fusão de Tabelas
<b>Relacionamento 1:1</b>	Tabela Própria	Adição de Coluna	Fusão de Tabelas
	Talvez	Sim	Não
	Não	Talvez	Sim
	Não	Não	Sim


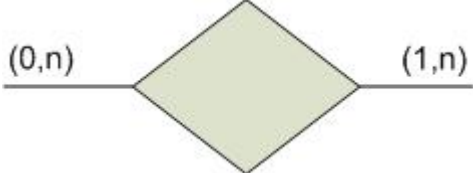
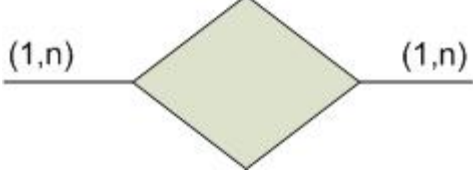


# Regras para Implementação de Relacionamentos

Tipo Relacionamento	Regra de Implementação		
	Tabela Própria	Adição de Coluna	Fusão de Tabelas
<b>Relacionamento 1:N</b>			
	Talvez	Sim	Não
	Talvez	Sim	Não
	Não	Sim	Não
	Não	Sim	Não



# Regras para Implementação de Relacionamentos

Tipo Relacionamento	Regra de Implementação		
	Tabela Própria	Adição de Coluna	Fusão de Tabelas
Relacionamento N:N			
	Sim	Não	Não
	Sim	Não	Não
	Sim	Não	Não



# Relacionamentos 1:1 (participação opcional)

- Exemplo: Homem casa-se com Mulher
- Opção 1 – Adição de colunas
  - Homem(identH, nome)
  - Mulher (identM, nome, identH, data, regime)
- Opção 2 – Criação de Tabela Própria
  - Homem (identH, nome)
  - Mulher (identM, nome
  - Casamento(identM, identH, data, regime)
    - identM referencia Mulher
    - identH referencia Homem
- Opção 1 minimiza a realização de Junções – menos acesso ao disco
  - Deve ser feita uma verificação de haver valores vazios para representar as mulheres não casadas → aplicação
  - Dependendo do SGBD, pode haver problemas em verificar colunas opcionais. Neste caso, a 2a alternativa pode ser mais viável



# Relacionamentos 1:1 (participação obrigatória e opcional)

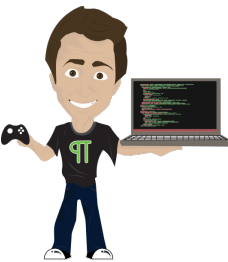
- Exemplo de aplicação bancária
  - O correntista pode ter ou não 1 e apenas 1 cartão magnético
- Opção 1 – fusão de tabelas
  - Correntista(codigoCorrentista, Nome, CodigoCartao, data)
- Opção 2 – adição de colunas na tabela de cardinalidade mínima 0
  - Correntista(codigoCorrentista, Nome)
  - Cartao(codigoCartao, data, codigoCorrentista)
    - codigoCorrentista referencia Correntista



# Relacionamentos 1:1 (participação obrigatória de ambas as entidades)

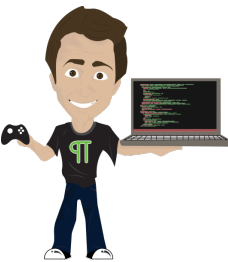
- Caso da Conferência organizada por comissão
- Opção única
  - Conferência(CodConf, Nome, DataNomeacao, NomeComissao)
- Nenhuma das outras alternativas atende plenamente o modelo, já que a obrigatoriedade da presença de ambas as entidades não permite registros com campos vazios ou ausência de registros





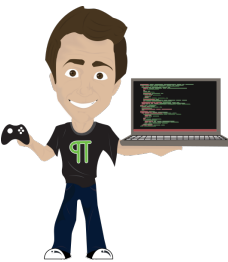
# Relacionamentos 1:N

- A alternativa sempre preferida é a adição de colunas à tabela de cardinalidade N
- Exemplo: Modelar um edifício que possui vários apartamentos
  - Edificio(codigoEdificio, Endereço)
  - Apartamento(codigoEdificio, numeroAp, areaUtil)
- Este é um caso típico de relacionamento identificador (visto anteriormente)
  - Código do Edifício também faz parte da chave primária de Apartamento (apartamento é entidade “fraca” de edifício)



# Relacionamentos 1:N

- Casos com cardinalidade mínima 0
- Exemplo: Empresa Financeira financia uma venda (de produto, apartamento, etc)
- Opção 1 – adição de colunas na tabela Venda
  - Financeira(codFinanceira, nome)
  - Venda(idVenda, data, codFinanceira, numParc, txJuro)
    - codFinanceira referencia Financeira
- Opção 2 – implementação de tabela própria (Alternativa)
  - Financeira(codFinanceira, nome)
  - Venda(idVenda, data)
  - Financia(idVenda, codFinanceira, numParc, txJuros)
    - idVenda referencia Venda
    - codFinanceira referencia Financeira



# Relacionamentos 1:N - Observações

- Operações que envolvem acesso a dados de uma venda e do respectivo financiamento exigem junções
  - Na Opção 1 isso não ocorre, uma vez que as informações da venda estão na própria tabela Vendas.
    - Na 2a opção, para buscar a data da venda, deve-se consultar a tabela “Vendas” a partir da tabela “Financiam”
    - Ou ainda, buscar qual foi a financeira que viabilizou a operação, deve-se buscar a respectiva venda e, em seguida a respectiva financeira.
  - Na Opção 1 as vendas à vista não teriam os dados da financeira cadastrados → campos vazios



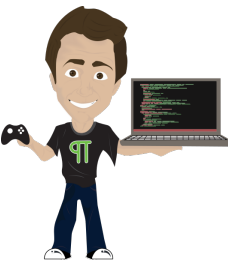
# Relacionamentos N:N

- Independente de cardinalidade mínima, os relacionamentos N:N sempre são implementados através de tabela própria.
- Mesmo que a cardinalidade seja opcional, não existe possibilidade de se agregar como campos de tabela, valores que não tem uma quantidade fixa definida
- Operações de junções são imprescindíveis
  - Médico(crmMedico, nome)
  - Paciente(idPaciente, Nome)
  - Consulta(idPaciente, crmMedico, data)
    - idPaciente referencia Paciente
    - crmMedico referencia Médico

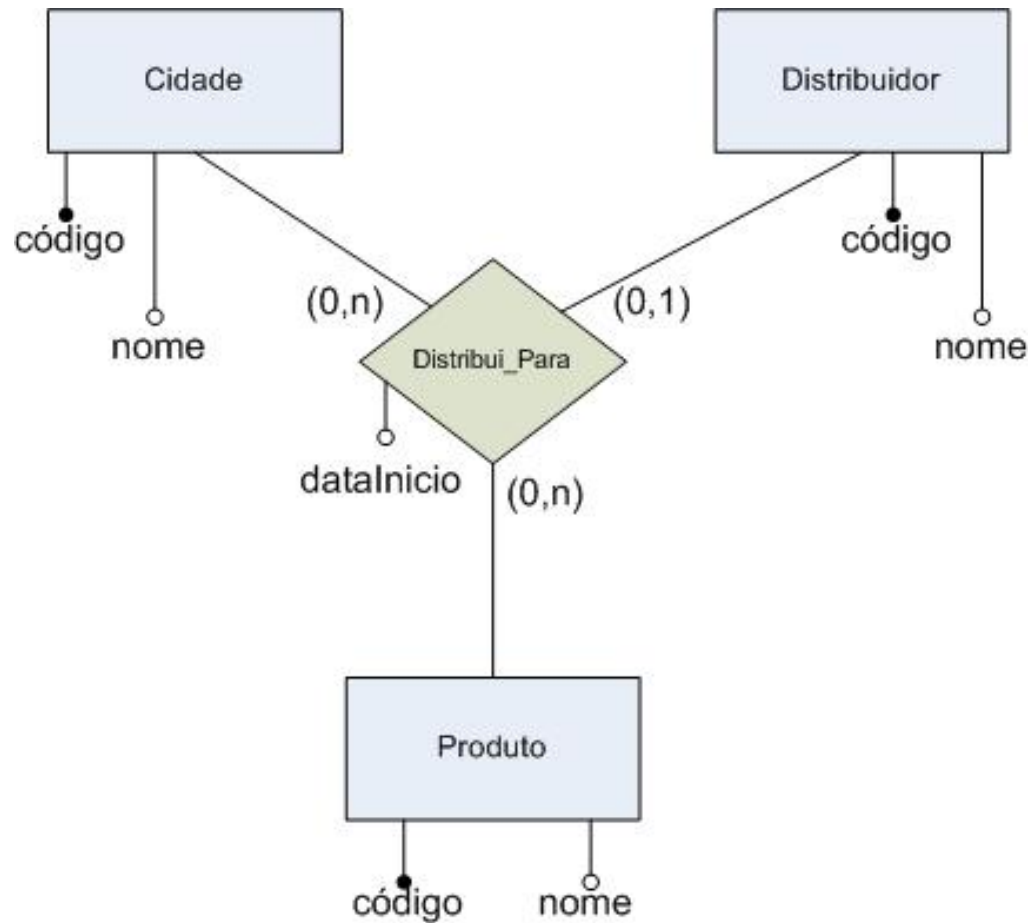


# Relacionamentos com grau maior que 2

- Não existem regras formais definidas para relacionamentos maior que 2
- Como implementar então?
  - O relacionamento é transformado em uma entidade que é ligada de forma binária a cada um dos participantes do relacionamento original
  - Aplicam-se as regras já conhecidas para poder gerar um mapeamento correto

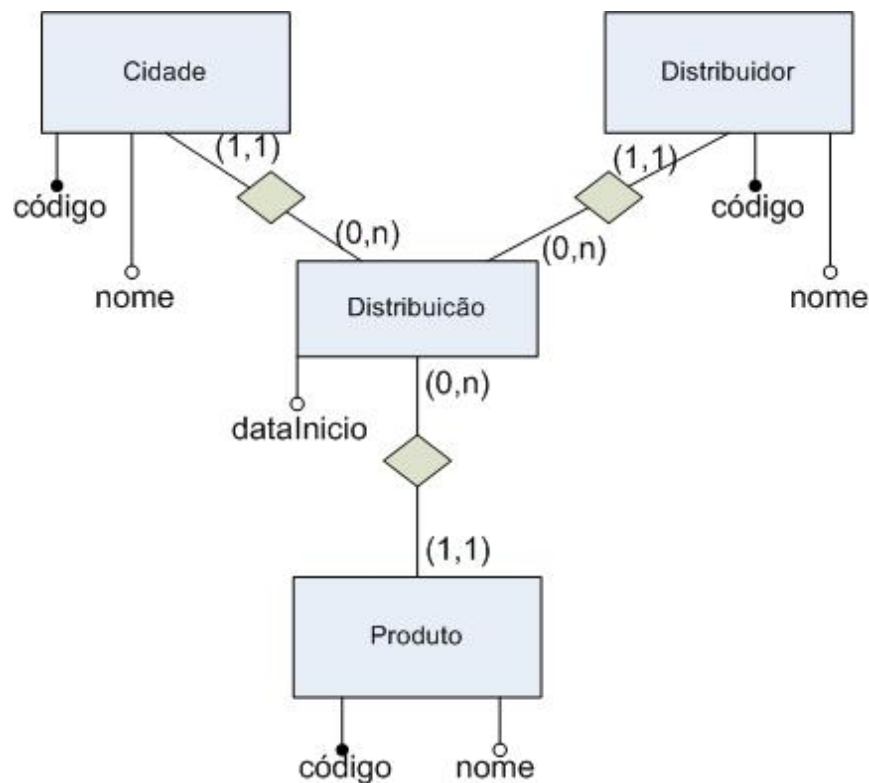


# Relacionamentos com grau maior que 2





# Relacionamentos com grau maior que 2



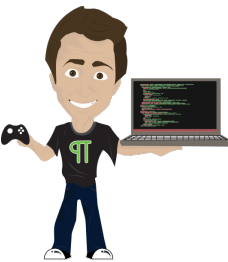
- Produto(codProd, nome)
- Cidade(codCid, nome)
- Distribuidor(codDist, nome)
- Distribuição(codProd, codCid, codDist, dataInício)
  - codProd referencia Produto
  - codCid referencia Cidade
  - codDist referencia Distribuidor



# Generalizações e Especializações

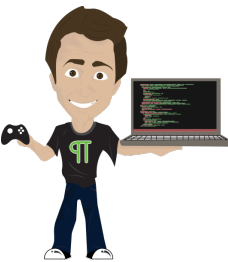
- Duas alternativas
  - Uma tabela por generalização
  - Uma tabela por entidade especializada



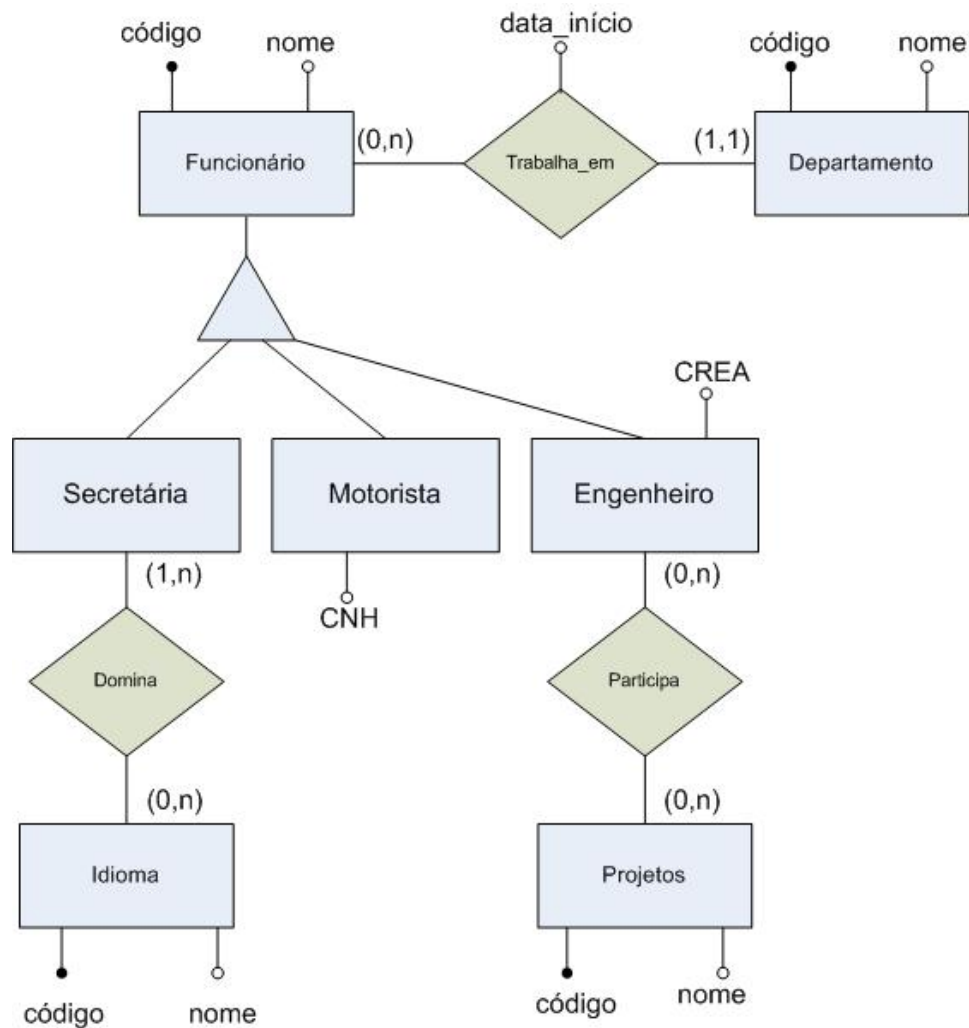


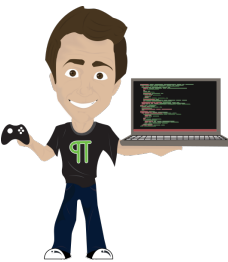
# Uma tabela por hierarquia

- Fundir especializações de uma entidade genérica em uma única tabela
  - Chave primária correspondente ao identificador da entidade mais genérica
  - Caso não exista, uma coluna TIPO, que identifica o tipo de especialização é incluído
  - Uma coluna para cada atributo da entidade genérica
  - Colunas da entidade especializada são tratados como opcionais
  - Relacionamentos
    - Entidade Genérica
    - Entidades Especializadas



# Uma tabela por hierarquia





# Uma tabela por hierarquia

- Funcionário(codFunc, Nome, codDep, **CNH**, **CREA**)
  - codDep referencia Departamento
- Departamento(codDep, Nome)
- Projeto(codProj, Nome)
- Domina(codFunc, codIdioma)
- Participa(codFunc, codProj)
  - codFunc referencia Funcionario
  - codProj referencia Projeto



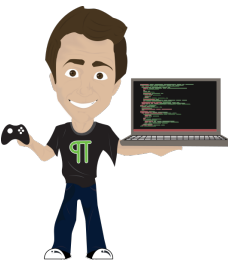
# Uma tabela por entidade especializada

- **Funcionário(codFunc, Nome, codDep)**
  - codDep referencia Departamento
- **Motorista(codFunc, CNH)**
  - codFunc referencia Funcionário
- **Engenheiro(codFunc, CREA)**
  - codFunc referencia Funcionário
- **Departamento(codDep, Nome)**
- **Projeto(codProj, Nome)**
- **Domina(codFunc, codIdioma)**
- **Participa(codFunc, codProj)**
  - codFunc referencia Funcionario
  - codProj referencia Projeto



# Comparando ambas as implementações

- Tabela Única
  - Todos os dados de uma ocorrência genérica e das especializações estão em uma única linha
    - Evita junções, dependendo do tipo de consulta
  - Chave primária é armazenada uma única vez
  - Ocorrência dos campos opcionais
- Uma tabela por entidade especializada
  - Vantagens para entidades especializadas que possuam muitos campos
  - Mais junções, porém menor ocorrência de campos opcionais
    - De qualquer forma, qualquer tratamento de campos opcionais deve ser feita pela aplicação



# Atributos Multivalorados

- Exemplo clássico: telefone
  - Cliente possui vários telefones
- Opção 1
  - Cliente(codCli, Nome)
  - Telefone (codCli, numTelefone)
- Opção 2
  - Cliente(codCli, Nome, Tel1, Tel2, Tel3, Tel4)
- O que se deve discutir?
  - cardinalidade
    - Opção 1: cardinalidade genérica: cliente pode ter 0 ou infinitos telefones
    - Opção 2: cardinalidade máxima: de 0 até 4 telefones
  - números de acessos ao disco
    - Opção 1: obriga junções
    - Opção 2: tudo está na mesma linha



# Quick Exercise

- Voltando ao contexto de escola
  - A escola oferece vários cursos e cada curso tem um rol de disciplinas associadas a ele
  - Cada disciplina pode ter várias turmas (os dependentes e os de curso normal) que pertence.
  - Uma turma sempre tem aulas em um local, que pode ser uma sala de aula ou um laboratório.
    - Lembrando que uma sala pode abrigar várias turmas em horários distintos
  - Entretanto, por ser muito extensa, a escola possui vários prédios e cada prédio abriga várias salas