

Instituto Federal Minas Gerais - Campus Bambuí
Departamento de Transformação e Computação - DEC
Curso de Transformação de Computação - ENGCAMP

Banco de Dados I

05 - Transformação entre os Modelos Conceitual e Relacional


Marcos Roberto Ribeiro

2024

- A modelagem de dados na abordagem entidade-relacionamento (ER) tem como objetivo a construção do **modelo conceitual** (ou **esquema conceitual**) de forma independente do SGBD
- A abordagem relacional modela os dados a nível de SGBD obtendo o **modelo lógico** (ou **esquema lógico**)
- Apesar de ser possível projetar um banco de dados diretamente no modelo lógico, o ideal é criar o modelo conceitual e, em seguida, convertê-lo para o modelo lógico
- Outra transformação possível é a **transformação reversa** do modelo lógico para o modelo conceitual (em processos de engenharia reversa)
- Existem algumas regras a serem seguidas para a obtenção de um modelo de banco de dados otimizado

Transformação de DER para esquema lógico

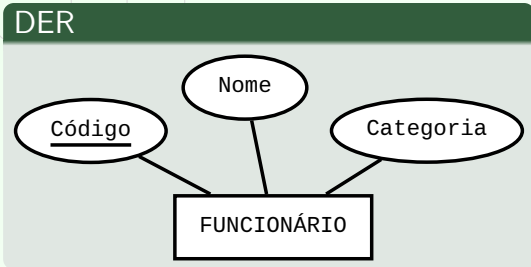


- 
- 1 Tradução inicial de entidades e respectivos atributos
 - 2 Tradução de relacionamentos e respectivos atributos
 - 3 Tradução de generalizações e especializações

Tradução inicial de entidades

- 1 Cada entidade é traduzida para uma tabela
 - 2 Cada atributo da entidade define uma coluna da tabela
 - 3 Os atributos identificadores da entidade correspondem às colunas que compõem a chave primária da tabela
- Essa é apenas uma transformação inicial nos próximos passos as tabelas poderão sofrer modificações

Exemplo de tradução de entidade



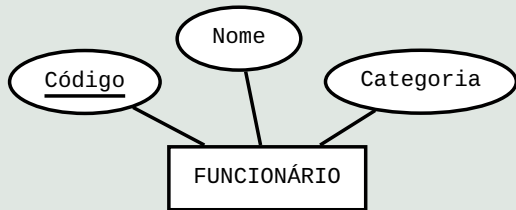
Esquema lógico

```
funcionario(  
  codigo_funcionario integer,  
  nome_funcionario varchar(60),  
  categoria varchar(30))
```

- Os nomes de tabelas e atributos devem obedecer às mesmas regras de nomes de variáveis da maioria das linguagens de programação
- Uma recomendação importante é colocar o nome da tabela no nome da chave primária e na coluna de descrição da tabela, pois, muitas vezes, essas colunas serão utilizadas em conjunto com colunas de outras tabelas

Exemplo de tradução de entidade

DER



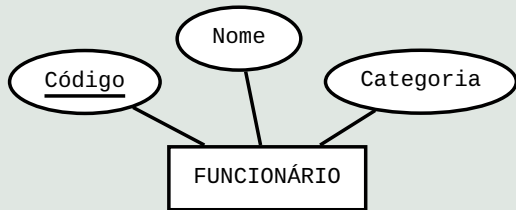
Esquema lógico

```
funcionario(  
  codigo_funcionario integer,  
  nome_funcionario varchar(60),  
  categoria varchar(30))
```

- Os nomes de tabelas e atributos devem obedecer às mesmas regras de nomes de variáveis da maioria das linguagens de programação
- Uma recomendação importante é colocar o nome da tabela no nome da chave primária e na coluna de descrição da tabela, pois, muitas vezes, essas colunas serão utilizadas em conjunto com colunas de outras tabelas

Exemplo de tradução de entidade

DER



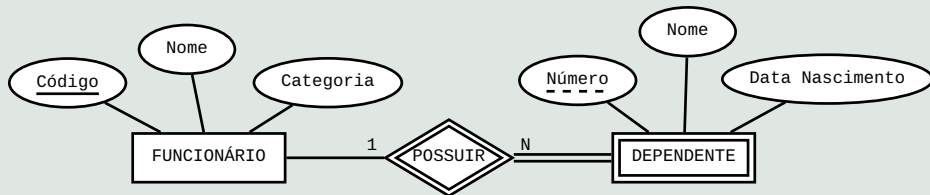
Esquema lógico

```
funcionario(  
  codigo_funcionario integer,  
  nome_funcionario varchar(60),  
  categoria varchar(30))
```

- Os nomes de tabelas e atributos devem obedecer às mesmas regras de nomes de variáveis da maioria das linguagens de programação
- Uma recomendação importante é colocar o nome da tabela no nome da chave primária e na coluna de descrição da tabela, pois, muitas vezes, essas colunas serão utilizadas em conjunto com colunas de outras tabelas

Entidades fracas

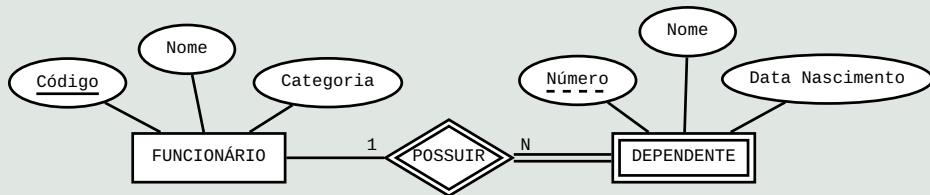
- A chave primária da entidade fraca será composta por suas chaves fracas mais os atributos identificadores de sua entidade forte



```
funcionario(  
    codigo_funcionario integer, nome_funcionario varchar(60),  
    categoria varchar(30))  
dependente(  
    codigo_funcionario integer, numero_dependente integer,  
    nome_dependente varchar(60), data_nascimento date)
```


Entidades fracas

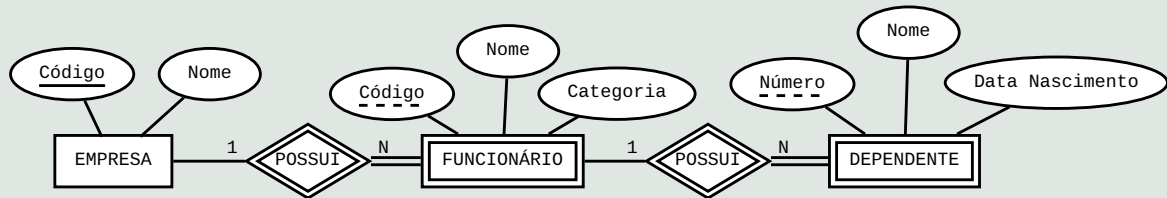
- A chave primária da entidade fraca será composta por suas chaves fracas mais os atributos identificadores de sua entidade forte



```
funcionario(  
  codigo_funcionario integer, nome_funcionario varchar(60),  
  categoria varchar(30))  
dependente(  
  codigo_funcionario integer, numero_dependente integer,  
  nome_dependente varchar(60), data_nascimento date)
```

Entidades fracas encadeadas

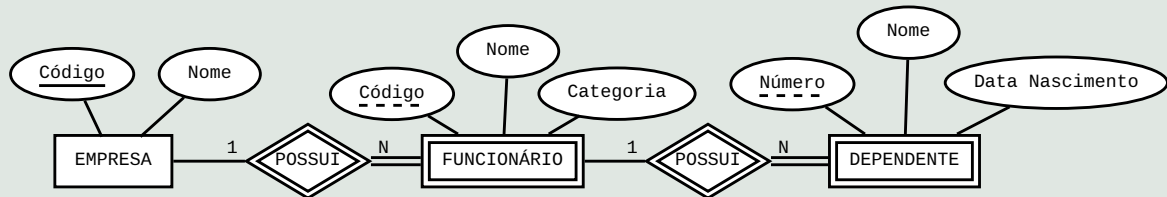
- No caso de entidades fracas encadeadas, o atributo da entidade forte se propaga para todas suas as entidades fracas



```
empresa(codigo_empresa integer, nome_empresa varchar(60))
funcionario(codigo_empresa integer, codigo_funcionario integer,
            nome_funcionario varchar(60), categoria varchar(30))
dependente(codigo_empresa integer, codigo_funcionario integer,
            numero_dependente integer, nome_dependente varchar(60),
            data_nascimento date)
```

Entidades fracas encadeadas

- No caso de entidades fracas encadeadas, o atributo da entidade forte se propaga para todas suas as entidades fracas



```
empresa(codigo_empresa integer, nome_empresa varchar(60))  
funcionario(codigo_empresa integer, codigo_funcionario integer,  
           nome_funcionario varchar(60), categoria varchar(30))  
dependente(codigo_empresa integer, codigo_funcionario integer,  
           numero_dependente integer, nome_dependente varchar(60),  
           data_nascimento date)
```

Tradução de relacionamentos

- A tradução dos relacionamentos leva em consideração a cardinalidade máxima e as restrições de participação

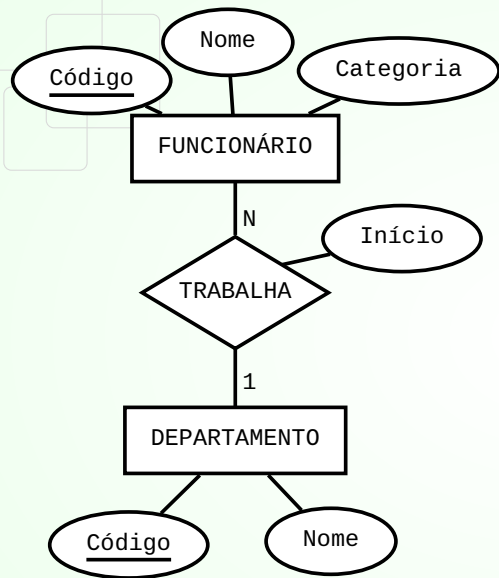
Possibilidades de tradução

- Adição de colunas
- Criação de tabela própria



- A tradução de relacionamentos com cardinalidade 1:N é feita com a adição de colunas
- A chave primária da entidade com cardinalidade 1 aparece como chave estrangeira na tabela gerada pela entidade com cardinalidade N
- Os atributos do relacionamento acompanham a chave estrangeira

Exemplo com relacionamento 1:N

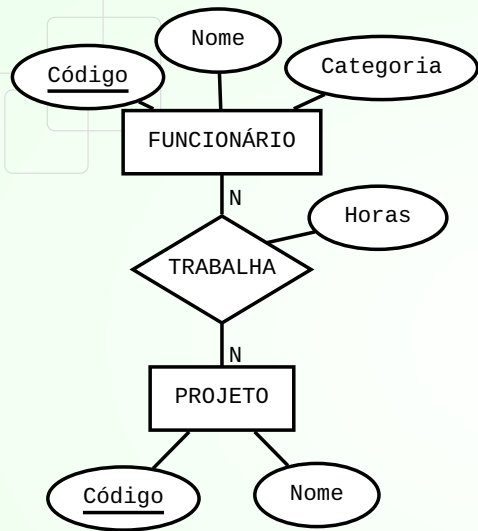


```
departamento(  
  codigo_departamento integer,  
  nome_departamento varchar(60))  
funcionario(  
  codigo_funcionario integer,  
  nome_funcionario varchar(60),  
  categoria varchar(30), inicio date  
  *codigo_departamento integer)  
*funcionario.codigo_departamento:  
  departamento:codigo_departamento
```



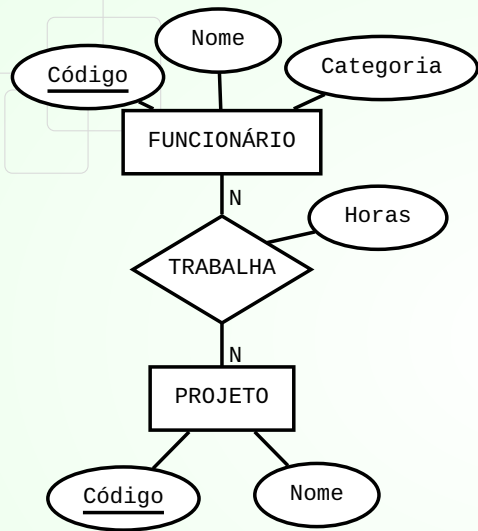
- No caso dos relacionamentos com cardinalidade N:N, é necessária a criação de uma tabela própria na tradução
- A tabela conterá as chaves primárias das entidades participantes transformadas em chaves estrangeiras e os atributos do relacionamento
- A chave primária da tabela será composta por todas as chaves estrangeiras
- O nome da tabela criada pode ser o próprio nome do relacionamento ou outro nome mais coerente com o modelo

Exemplo de tradução de relacionamento N:N



```
projeto(  
    codigo_projeto integer,  
    nome_projeto varchar(60))  
funcionario(  
    codigo_funcionario integer,  
    nome_funcionario varchar(60),  
    categoria varchar(30))  
funcionario_projeto(  
    *codigo_projeto integer,  
    *codigo_funcionario integer,  
    horas integer)  
*funcionario_projeto.codigo_projeto:  
    projeto:codigo_projeto  
*funcionario_projeto.codigo_funcionario:  
    funcionario.codigo_funcionario
```


Exemplo de tradução de relacionamento N:N

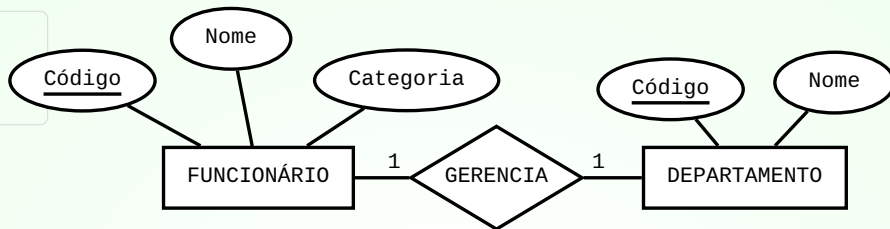


```
projeto(  
  codigo_projeto integer,  
  nome_projeto varchar(60))  
funcionario(  
  codigo_funcionario integer,  
  nome_funcionario varchar(60),  
  categoria varchar(30))  
funcionario_projeto(  
  *codigo_projeto integer,  
  *codigo_funcionario integer,  
  horas integer)  
*funcionario_projeto.codigo_projeto:  
  projeto:codigo_projeto  
*funcionario_projeto.codigo_funcionario:  
  funcionario.codigo_funcionario
```



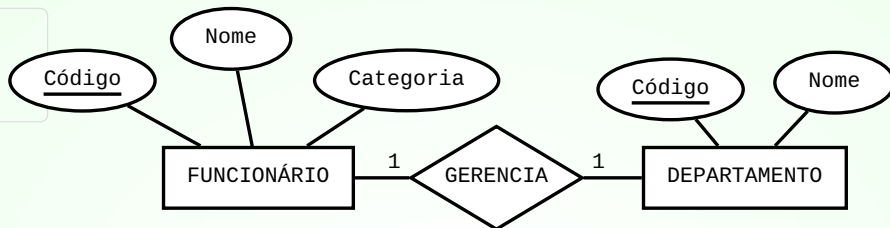
- A tradução dos relacionamentos 1:1 utiliza as mesmas regras da tradução de relacionamentos 1:N
- A principal dificuldade é determinar qual entidade pode ser considerada com cardinalidade 1 e qual pode ser considerada com cardinalidade N
- Em alguns casos, é interessante considerar as restrições de participação
- Se todas ou nenhuma das entidades do relacionamento possuem participação total, é preciso usar o bom senso
- Caso contrário, podemos considerar as entidades com participação total como se tivessem cardinalidade N

Exemplo de tradução de relacionamento 1:1 I



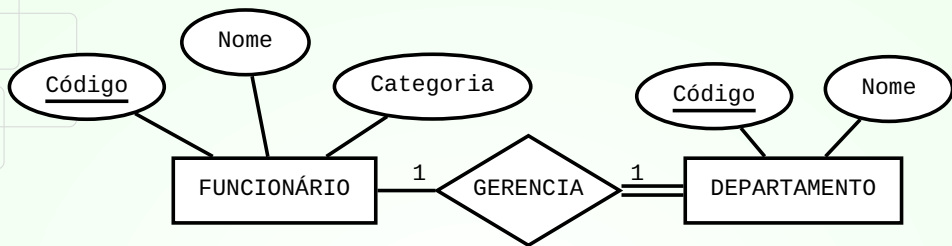
```
funcionario(  
  codigo_funcionario integer, nome_funcionario varchar(60),  
  categoria varchar(30))  
departamento(  
  codigo_departamento integer, nome_departamento varchar(30),  
  *codigo_funcionario integer)  
*departamento.codigo_funcionario: funcionario.codigo_funcionario
```

Exemplo de tradução de relacionamento 1:1 II



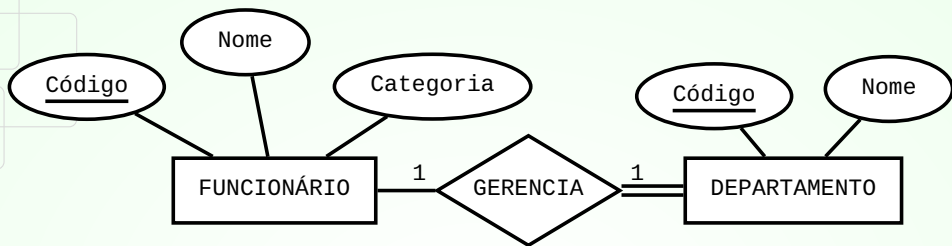
```
departamento(  
  codigo_departamento integer, nome_departamento varchar(30))  
funcionario(  
  codigo_funcionario integer, nome_funcionario varchar(60),  
  categoria varchar(30), *codigo_departamento integer)  
*funcionario.codigo_departamento: departamento.codigo_departamento
```

Tradução de relacionamento 1:1 com participação total



```
funcionario(  
  codigo_funcionario integer, nome_funcionario varchar(60),  
  categoria varchar(30))  
departamento(  
  codigo_departamento integer, nome_departamento varchar(30),  
  *codigo_funcionario integer)  
*departamento.codigo_funcionario: funcionario.codigo_funcionario
```

Tradução de relacionamento 1:1 com participação total

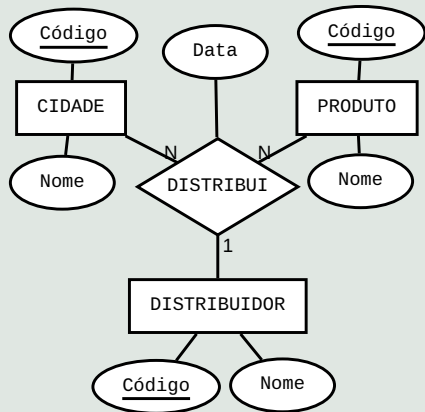


```
funcionario(  
  codigo_funcionario integer, nome_funcionario varchar(60),  
  categoria varchar(30))  
departamento(  
  codigo_departamento integer, nome_departamento varchar(30),  
  *codigo_funcionario integer)  
*departamento.codigo_funcionario: funcionario.codigo_funcionario
```

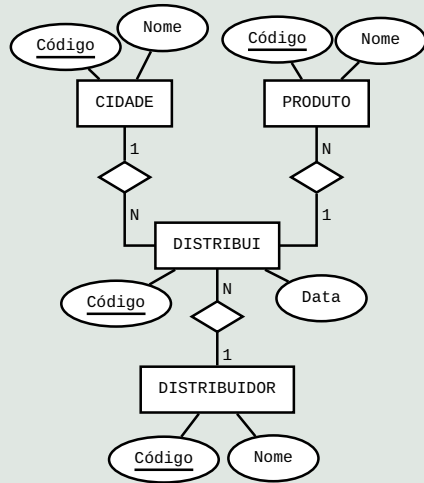
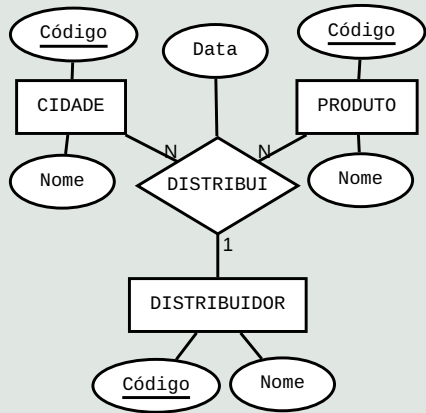


- As regras apresentadas até este ponto, aplicam-se somente à relacionamentos binários
- Passos para tradução de relacionamentos com grau maior que 2:
 - 1 O relacionamento é transformado em uma entidade ligada através de relacionamentos binários a cada uma das entidades que participavam do relacionamentos original
 - 2 As regras de implementação de entidades e relacionamentos binários apresentadas anteriormente são aplicadas às entidades e aos relacionamentos binários assim criados

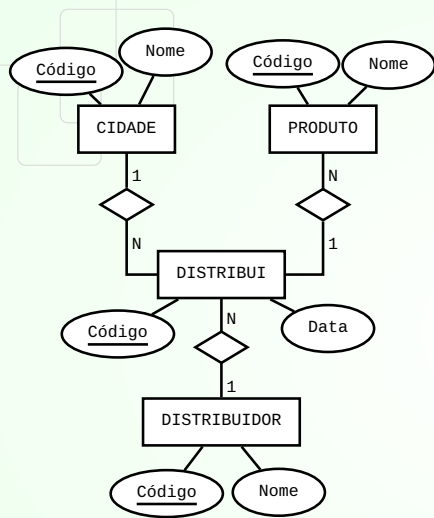
Exemplo com relacionamento de grau maior que 2 I



Exemplo com relacionamento de grau maior que 2 |

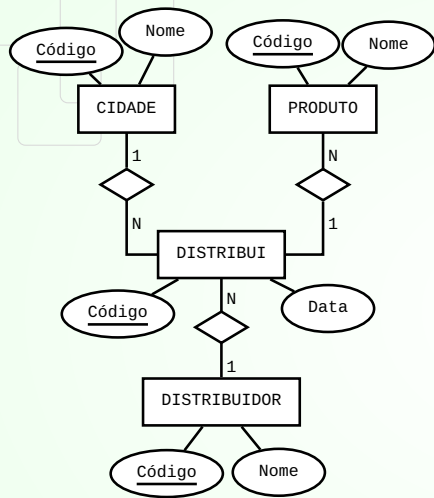


Exemplo com relacionamento de grau maior que 2 II



```
cidade(codigo_cidade integer,  
      nome_cidade varchar(50))  
produto(codigo_produto integer,  
        nome_produto varchar(50))  
distribuidor(codigo_distribuidor integer,  
            nome_distribuidor varchar(50))  
distribui(codigo integer,  
          data date, *codigo_cidade integer,  
          *codigo_produto integer,  
          *codigo_distribuidor integer)  
*distribui.codigo_cidade: cidade.codigo_cidade  
*distribui.codigo_produto:  
    produto.codigo_produto  
*distribui.codigo_distribuidor:  
    distribuidor.codigo_distribuidor
```

Exemplo com relacionamento de grau maior que 2 II



```
cidade(codigo_cidade integer,  
      nome_cidade varchar(50))  
produto(codigo_produto integer,  
        nome_produto varchar(50))  
distribuidor(codigo_distribuidor integer,  
            nome_distribuidor varchar(50))  
distribui(codigo integer,  
          data date, *codigo_cidade integer,  
          *codigo_produto integer,  
          *codigo_distribuidor integer)  
*distribui.codigo_cidade: cidade.codigo_cidade  
*distribui.codigo_produto:  
    produto.codigo_produto  
*distribui.codigo_distribuidor:  
    distribuidor.codigo_distribuidor
```

Participação total



- Até o momento não nos preocupamos com a obrigatoriedade das colunas, mas isso deve constar no modelo relacional
- Assim, deve ser observado se a entidade possui participação total no relacionamento
- Nesse caso, a chave estrangeira criada na tabela deverá ser obrigatória

Exemplo de tradução de participação total



```
funcionario(+codigo_funcionario integer, +nome_funcionario varchar(60),  
            *codigo_departamento integer)  
*funcionario.codigo_departamento: departamento.codigo_departamento  
departamento(+codigo_departamento integer, +nome_departamento varchar(30),  
              +*codigo_funcionario integer)  
*departamento.codigo_funcionario: funcionario.codigo_funcionario
```

Exemplo de tradução de participação total



```
funcionario(+codigo_funcionario integer, +nome_funcionario varchar(60),  
            *codigo_departamento integer)  
*funcionario.codigo_departamento: departamento.codigo_departamento  
departamento(+codigo_departamento integer, +nome_departamento varchar(30),  
              +*codigo_funcionario integer)  
*departamento.codigo_funcionario: funcionario.codigo_funcionario
```

Tradução de generalização e especialização

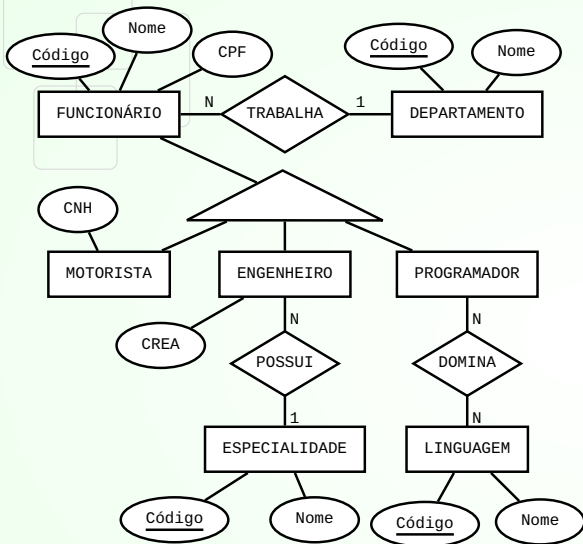


- Para a implementação de hierarquias de generalização e especialização na abordagem relacional, há duas alternativas:
 - Uso de uma tabela para cada entidade
 - Uso de uma única tabela para toda hierarquia

Uma tabela por hierarquia

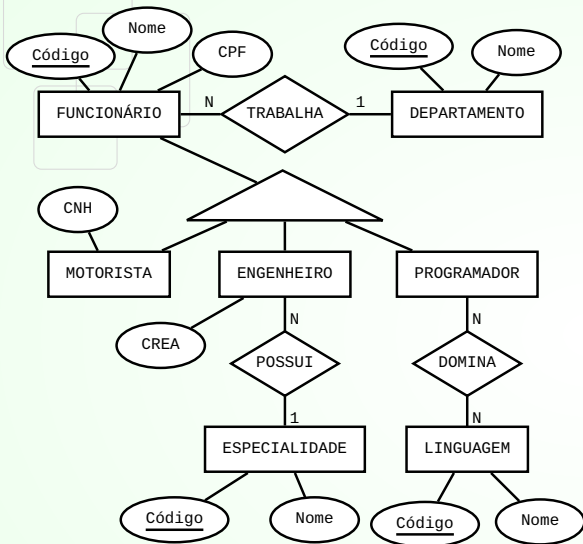
- Todas tabelas referentes às especializações de uma entidade genérica são fundidas em uma única tabela
 - Caso não exista, uma coluna **Tipo**, que identifica o tipo de entidade que está sendo representada por cada linha da tabela
 - Uma coluna para cada atributo da entidade genérica
 - Colunas referentes aos relacionamentos dos quais participa a entidade genérica (se necessário)
 - Uma coluna para cada atributo de cada entidade especializada, *estas colunas devem ser opcionais*
 - Colunas *opcionais* referentes aos relacionamentos dos quais participa cada entidade especializada (se necessário)
- Uma entidade especializada pode não gerar nenhuma coluna, caso não tenha atributos e seus relacionamentos sejam implementados através de tabelas próprias

Exemplo de uma tabela por hierarquia I



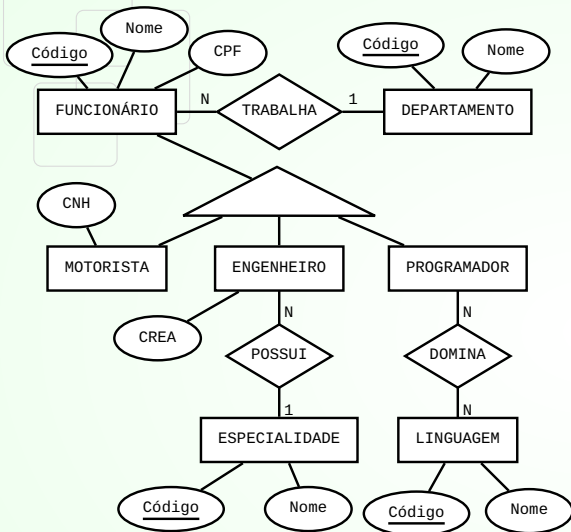
```
departamento(  
    +codigo_departamento integer,  
    +nome_departamento varchar(30))  
especialidade(  
    +codigo_especialidade integer,  
    +nome_especialidade varchar(30))  
linguagem(  
    +codigo_linguagem integer,  
    +nome_linguagem varchar(30))
```

Exemplo de uma tabela por hierarquia I



```
departamento(  
  +codigo_departamento integer,  
  +nome_departamento varchar(30))  
especialidade(  
  +codigo_especialidade integer,  
  +nome_especialidade varchar(30))  
linguagem(  
  +codigo_linguagem integer,  
  +nome_linguagem varchar(30))
```

Exemplo de uma tabela por hierarquia II



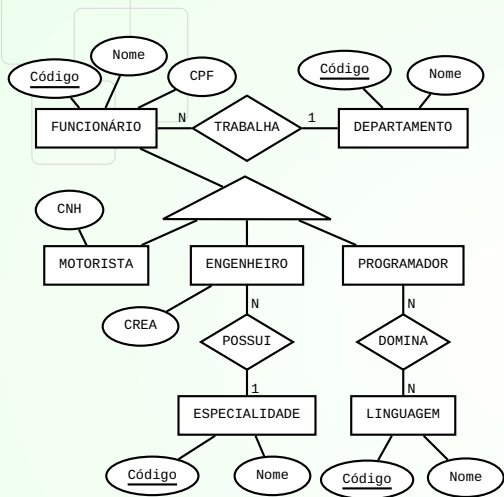
```
funcionario(  
  +codigo_funcionario integer,  
  +nome_funcionario varchar(60),  
  cpf char(11), +tipo char(1),  
  cnh varchar(11), crea varchar(10),  
  *codigo_departamento integer  
  *codigo_especialidade integer)  
*funcionario.codigo_departamento:  
  departamento.codigo_departamento  
*funcionario.codigo_especialidade:  
  especialidade.codigo_especialidade  
domina(+*codigo_linguagem integer,  
  +*codigo_funcionario integer)  
*domina.codigo_linguagem:  
  linguagem.codigo_linguagem  
*domina.codigo_funcionario:  
  funcionario.codigo_funcionario
```

Uma tabela por entidade especializada



- Nessa implementação, devemos criar uma tabela para cada entidade que compõe a hierarquia, aplicando as regras correspondentes à implementação de entidades e relacionamentos já apresentadas
- O único acréscimo é a inclusão da chave primária em todas as tabelas, sendo que, nas entidades especializadas, ela é também uma chave estrangeira para a chave primária da tabela pai

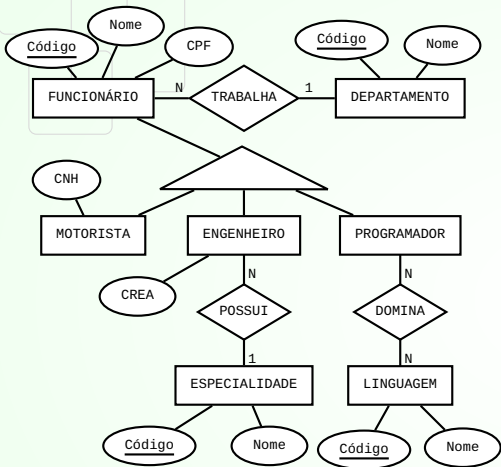
Exemplo de uma tabela para cada entidade I



```
departamento(+codigo_departamento integer,  
             +nome_departamento varchar(30))  
especialidade(+codigo_especialidade integer,  
             +nome_especialidade varchar(30))  
linguagem(+codigo_linguagem integer,  
          +nome_linguagem varchar(30))  
funcionario(+codigo_funcionario integer,  
            +nome_funcionario varchar(60),  
            cpf char(11), +tipo char(1),  
            *codigo_departamento integer)  
*funcionario.codigo_departamento:  
departamento.codigo_departamento
```

E a coluna **tipo**? É realmente necessária?

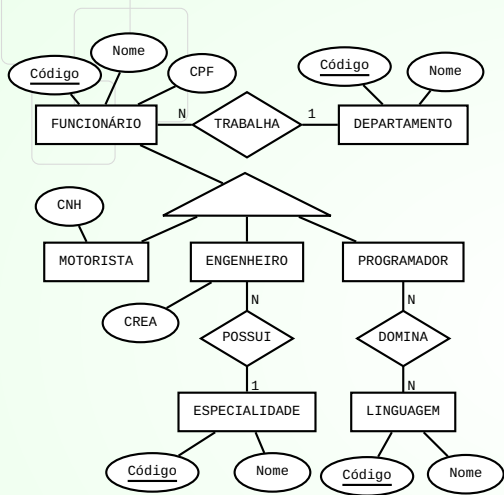
Exemplo de uma tabela para cada entidade I



```
departamento(+codigo_departamento integer,  
             +nome_departamento varchar(30))  
especialidade(+codigo_especialidade integer,  
             +nome_especialidade varchar(30))  
linguagem(+codigo_linguagem integer,  
          +nome_linguagem varchar(30))  
funcionario(+codigo_funcionario integer,  
            +nome_funcionario varchar(60),  
            cpf char(11), +tipo char(1),  
            *codigo_departamento integer)  
*funcionario.codigo_departamento:  
departamento.codigo_departamento
```

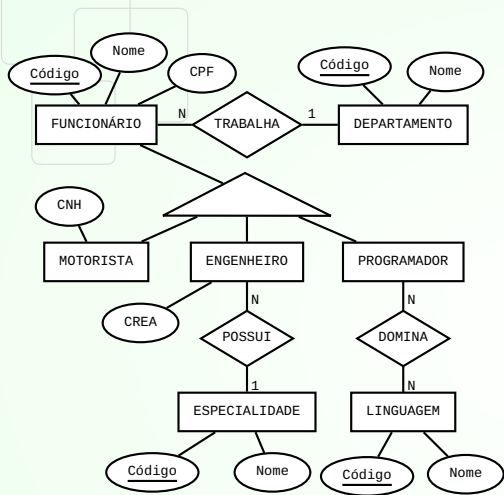
E a coluna **tipo**? É realmente necessária?

Exemplo de uma tabela para cada entidade II



```
motorista(+*codigo_funcionario integer,  
          +cnh varchar(11))  
*motorista.codigo_funcionario:  
  funcionario.codigo_funcionario  
engenheiro(+*codigo_funcionario integer,  
            +crea varchar(10),  
            *codigo_especialidade integer)  
*engenheiro.codigo_funcionario:  
  funcionario.codigo_funcionario  
*engenheiro.codigo_especialidade:  
  especialidade.codigo_especialidade
```

Exemplo de uma tabela para cada entidade III



```
programador(+*codigo_funcionario integer)
programador.codigo_funcionario:
    programador.codigo_funcionario
domina(+*codigo_linguagem integer,
      +*codigo_funcionario integer)
*domina.codigo_linguagem:
    linguagem.codigo_linguagem
*domina.codigo_funcionario:
    programador.codigo_funcionario
```


Comparação entre as duas alternativas

Vantagem da implementação com tabela única

- Todos os dados da entidade genérica e os dados de suas especializações estão em uma única tabela
- Não há necessidade de junções quando desejamos obter dados da entidade genérica juntamente a entidade especializada

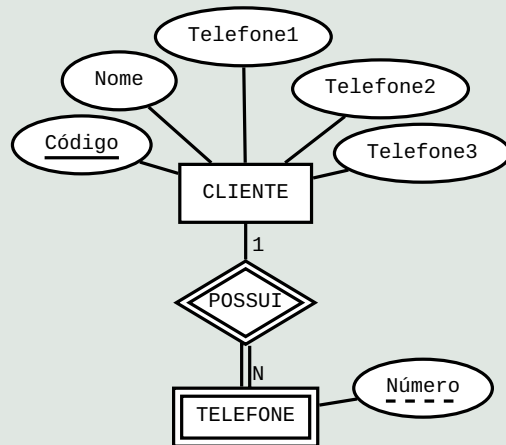
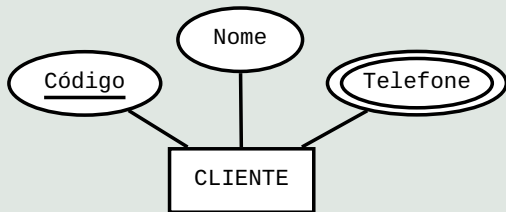
Vantagem da implementação com uma tabela por entidade especializada

- Redução do número de colunas opcionais
- Melhor controle das especializações



- Os atributos multi-valorados não são desejáveis em DER
- Contudo, na permanência desse tipo de atributo, há duas alternativas para a transformação no esquema lógico:
 - Transformar o atributo multi-valorado em vários atributos mono-valorados
 - Criar um entidade para o atributo multi-valorado

Atributo multi-valorado como entidade

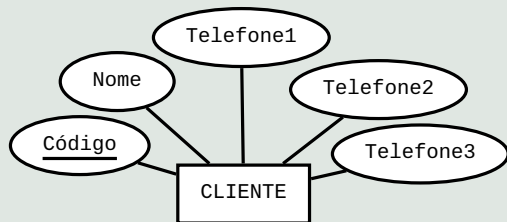
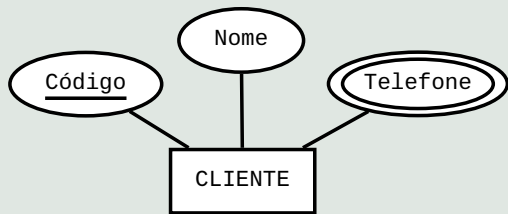


Problemas no atributo multi-valorado como entidade



- São raros os clientes que possuem mais que três telefones
- Quando isso ocorrer, é suficiente armazenarmos apenas três números
- Não há consultas ao banco de dados usando o número de telefone como critério de seleção
- Os números de telefone são apenas exibidos ou impressos juntos às demais informações de cliente

Atributo multi-valorado para mono-valorado




- Essa alternativa permite que os telefones de um cliente sejam obtidos mais rapidamente (todos na mesma tabela)
- Uma eventual consulta usando o número de telefone como critério de busca torna-se mais complicada, já que devem ser referenciados todas as colunas de telefone

Transformação reversa de modelos relacionais

- A transformação reversa parte de um modelo de implementação e constrói um modelo conceitual que descreve abstratamente a implementação em questão
- Falamos de transformação reversa quando transformamos modelos de banco de dados mais ricos em detalhes de implementação em modelos de dados mais abstratos
- Um caso específico de transformação reversa de banco de dados é o da transformação reversa de modelos relacionais
- Nesse tipo de transformação reversa, temos, como ponto de partida, um modelo lógico de um banco de dados relacional e, como resultado, um modelo conceitual na abordagem ER
- A transformação reversa de modelos relacionais pode ser útil quando não se tem um modelo conceitual para um banco de dados existente
- Isso pode acontecer quando o banco de dados foi desenvolvido de forma empírica, sem o uso de uma metodologia de desenvolvimento, ou quando o esquema do banco de dados sofreu modificações ao longo do tempo, sem que as mesmas tenham sido registradas

Processo de transformação reversa



- 
- 1 Identificação da construção ER correspondente a cada tabela
 - 2 Definição de relacionamentos 1:n e 1:1
 - 3 Definição de atributos
 - 4 Definição de identificadores de entidades e relacionamentos

Exemplo transformação reversa I

```
disciplina(codigo_disciplina integer, nome_disciplina varchar(50))
curso(codigo_curso integer, nome_curso varchar(50))
matriz(*codigo_curso integer, *codigo_disciplina integer,
      obrigatoria char(1))
    *matriz.codigo_curso: curso.codigo_curso
    *matriz.codigo_disciplina: disciplina.codigo_disciplina
sala(*codigo_predio integer, codigo_sala integer,
     capacidade integer)
    *sala.codigo_predio: predio.codigo_predio
predio(codigo_predio integer, local varchar(30))
turma (ano integer, semestre integer, sigla_turma varchar(5),
      *codigo_predio integer, *codigo_sala integer)
    *turma.codigo_disciplina: disciplina.codigo_disciplina
    *turma.(codigo_predio,codigo_sala):
      sala.(codigo_predio,codigo_sala)
```


Exemplo transformação reversa II

```
turma_disciplina(ano integer, semestre integer,  
                 sigla_turma varchar(5), *codigo_disciplina integer)  
*turma_disciplina.(ano, semestre, sigla_turma):  
    turma.(ano, semestre, sigla_turma)  
*turma_disciplina.codigo_disciplina:  
    disciplina.codigo_disciplina  
tecnico(codigo_tecnico integer, nome_tecnico varchar(50))  
laboratorio (*codigo_predio integer, *codigo_sala integer,  
             equipamento varchar(50), *codigo_tecnico integer)  
*laboratorio.codigo_tecnico: tecnico.codigo_tecnico  
*laboratorio.(codigo_predio, codigo_sala):  
    sala.(codigo_predio, codigo_sala)
```

Identificação de elementos do DER

- Na primeira etapa da transformação reversa de um banco de dados relacional, definimos, para cada tabela do modelo relacional, seu elemento correspondente no DER
- Uma tabela pode corresponder a:
 - Uma entidade
 - Um relacionamento N:N
 - Uma entidade especializada
- O fator determinante da construção ER que corresponde a uma tabela é a composição de sua chave primária
- Tabelas podem ser classificadas em três tipos de acordo com sua chave primária:
 - Chave primária composta por mais de uma chave estrangeira
 - Toda chave primária é uma chave estrangeira
 - Outros Casos

Chave primária composta por mais de uma chave estrangeira

- A tabela que possui uma chave primária composta de múltiplas chaves estrangeiras implementa um relacionamento N:N entre as entidades correspondentes às tabelas referenciadas pelas chaves estrangeiras
- Um exemplo de tabela desse tipo é a tabela `matriz` que tem como chave primária `codigo_curso` e `codigo_disciplina`
- Ambas colunas são chave estrangeira em relação às tabelas `curso` e `disciplina`
- Portanto, a tabela `matriz` representa um relacionamento entre as entidades `curso` e `disciplina`

Toda chave primária é uma chave estrangeira

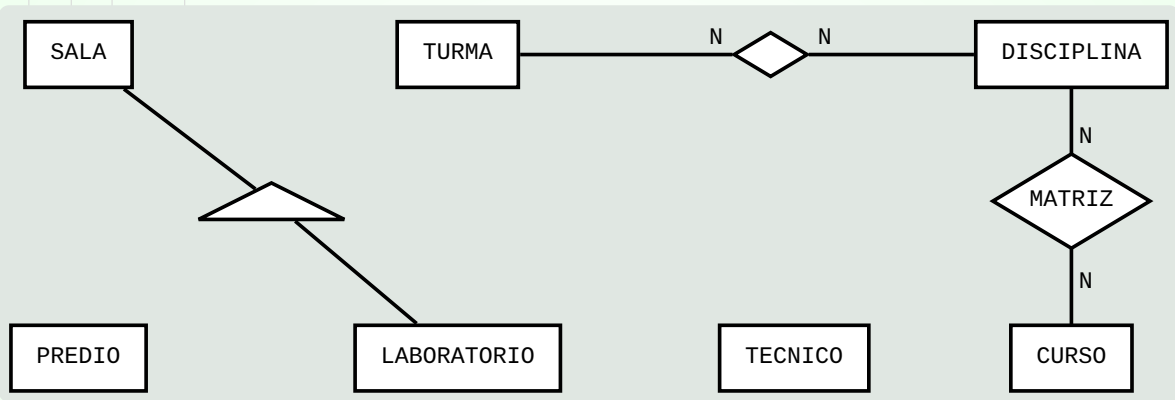
- A tabela cuja chave primária é toda ela uma chave estrangeira representa uma entidade que forma uma especialização da entidade correspondente à tabela referenciada pela chave estrangeira
- Um exemplo de tabela desse tipo é a tabela laboratorio que possui como chave primária as colunas codigo_predio e codigo_sala as quais são chave estrangeira da tabela sala
- A restrição de integridade referencial em questão especifica que uma linha na tabela laboratorio somente existe quando uma linha com a mesma chave existir na tabela sala
- A nível de modelo ER, isso significa que uma ocorrência da entidade laboratório somente pode existir quando a correspondente ocorrência da entidade sala existe, ou seja, significa que a entidade laboratório é uma especialização de sala



- Quando a chave primária da tabela não for composta de múltiplas chaves estrangeiras, nem for toda uma chave estrangeira, a tabela representa uma entidade
- Exemplificando, a tabela curso, cuja chave primária, a coluna `codigo_curso` não contém chaves estrangeiras, representa uma entidade
- Da mesma forma, a tabela sala também representa uma entidade, sua chave primária (colunas `codigo_predio` e `codigo_sala`) contém apenas uma chave estrangeira (coluna `codigo_predio`)
- O mesmo é válido para as tabelas disciplina, predio e turma

DER Inicial

- Até o momento temos o seguinte DER:

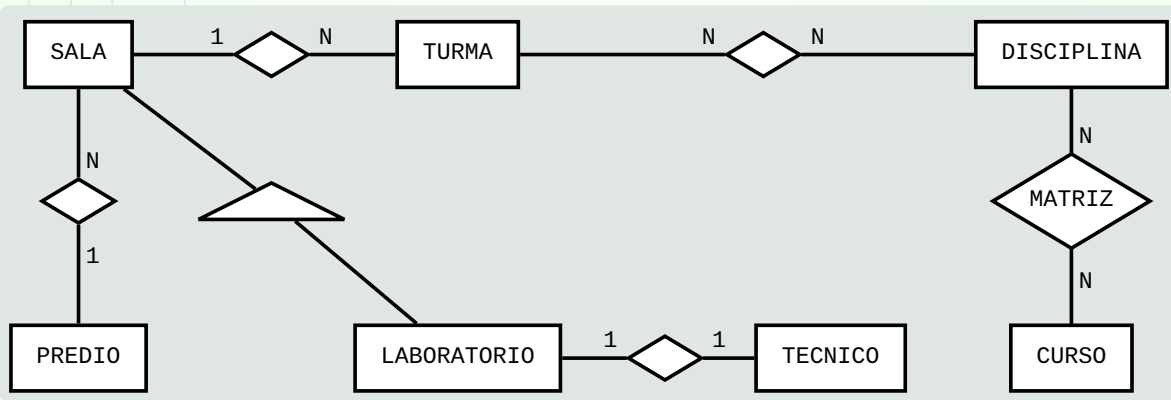


Identificação de relacionamentos 1:N e 1:1

- Toda chave estrangeira que não se enquadra nas regras anteriores, ou seja, toda chave estrangeira que não faz parte de uma chave primária composta por múltiplas chaves estrangeiras, nem é toda ela uma chave primária, representa um relacionamento 1:N ou 1:1
- Em outros termos, toda chave estrangeira que não corresponde a um relacionamento N:N, nem a uma entidade especializada representa um relacionamento 1:N ou 1:1
- A regra não permite definir se a cardinalidade do relacionamento é 1:N ou 1:1
- Para definir qual dos dois tipos de relacionamentos está sendo representado pela chave estrangeira, é necessário verificar se há restrições de unicidade sobre a chave estrangeira, se afirmativo o relacionamento será 1:1

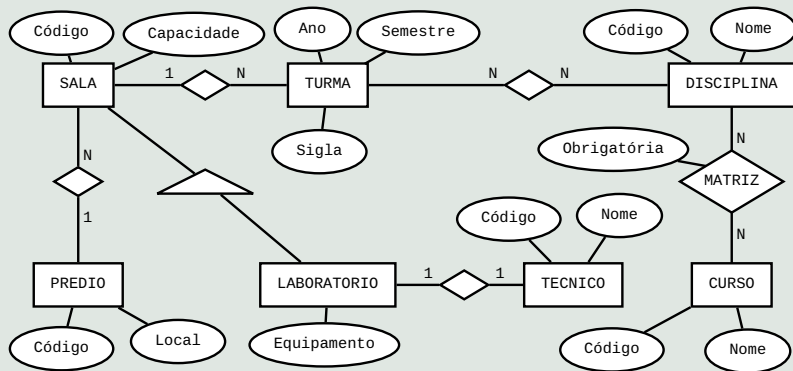
Identificação de relacionamentos 1:N e 1:1

- Em nosso exemplo temos o seguinte:



Definição de atributos

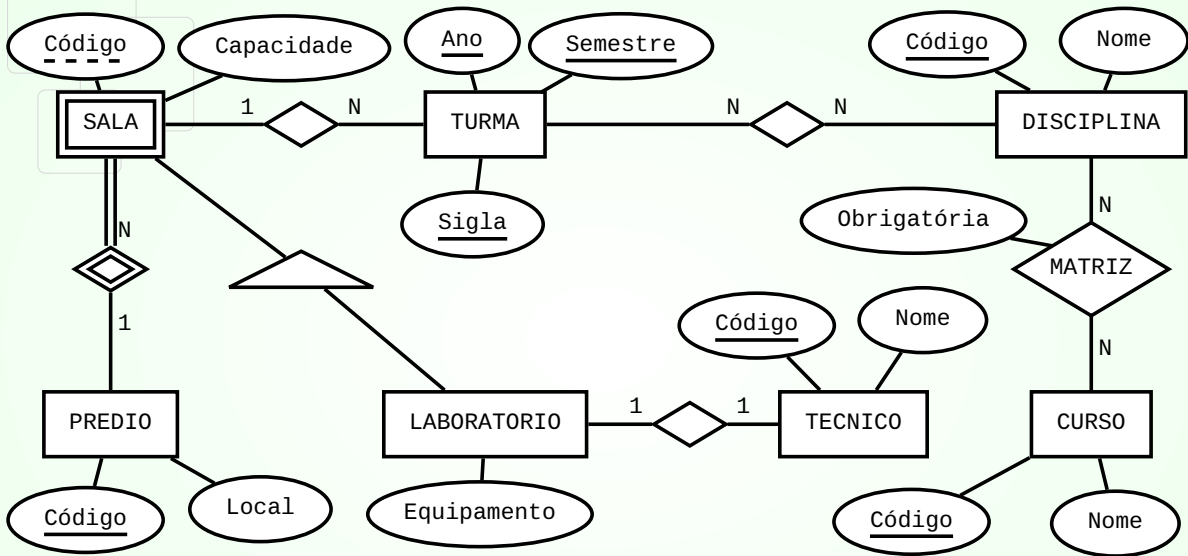
- Para cada coluna de uma tabela, que não seja chave estrangeira, é definido um atributo na entidade ou relacionamento correspondente
- As colunas chave estrangeira não correspondem a atributos no diagrama ER, mas sim a relacionamentos, e por isso já foram tratadas





- Coluna da chave primária que não é chave estrangeira
 - Toda coluna que faz parte da chave primária e que não é chave estrangeira corresponde a um atributo identificador
- Coluna da chave primária que é chave estrangeira
 - Toda coluna que faz parte da chave primária e que é chave estrangeira corresponde a um identificador externo da entidade
 - Com isso, a entidade será uma entidade fraca, por exemplo, a coluna `codigo_predio`, que é parte da chave primária da tabela `sala` é também chave estrangeira em relação a tabela `predio`
 - Portanto, a entidade `sala` é uma entidade fraca identificada pelo relacionamento com `predio`

DER Final





ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

HEUSER, C. A. **Projeto de banco de dados**. 6. ed. Porto Alegre: Bookman, 2009.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados**. 3. ed. São Paulo: McGrawHill, 2008.