

## 6 . Modelo Lógico

[illegible]

o modelo de dados relacional usa o conceito de uma relação matemática, como se fosse uma tabela com valores. Os modelos que antecederam o modelo relacional são os modelos hierárquicos e modelos de rede.

Quando uma relação é usada como uma tabela de valores, cada linha desta relação representa uma coleção de valores de dados relacionados.

### Exemplo:

## PESSOA

| Codigo | Nome  | Telefone | Endereço |
|--------|-------|----------|----------|
| 1      | Maria | 232323   | Av. JK   |
| 2      | José  | 242424   | Av. JB   |
| 3      | Carla | 252525   | Av.JX    |

No exemplo acima, a linha com o código 1 representa os dados relacionados de uma pessoa que mora na Av. JK, com telefone de número 232323 cujo nome é Maria.

Associando com a unidade 5 (modelo conceitual), onde cada entidade e relacionamento modelava um objeto do mundo real. Na abordagem relacional, cada linha representa dados de uma ocorrência de entidade. No caso acima a entidade PESSOA do modelo conceitual representa a relação no modelo relacional.

O modelo relacional formal chama cada linha de uma tabela de **tupla**, o cabeçalho da coluna é chamado de **atributo** e o conjunto ordenado das linhas é denominado de **tabela** ou **relação**. E ainda podemos definir o tipo de dado que descreve os valores que podem aparecer nas tabelas como **domínios**.

**Nome da Tabela (Relação)** → **PESSOA**

**Atributos** →

| Nome  | Telefone | Endereço |
|-------|----------|----------|
| Maria | 232323   | Av. JK   |
| José  | 242424   | Av. JB   |

← **Linhas (Tuplas)**

Uma tabela é um conjunto de tuplas,  
onde cada tupla é uma lista ordenada de valores.

Para identificar cada tupla e estabelecer relações entre estas o conceito básico é o da **chave**.

Vamos considerar o que o autor Heuser descreve sobre chave no seu livro Projeto de Banco de Dados . Neste ele considera três tipos de chaves: chave primária, chave alternativa e chave estrangeira.

### CHAVE PRIMÁRIA:

Como exposto anteriormente, uma tabela é constituída por uma lista ordenada de valores, ou seja, por tuplas. Como forma de garantir que não aconteça a duplicidade dessas tuplas, utiliza-se o procedimento de inserir nessa tabela uma chave.

A chave funciona como identificadora de uma tupla, garantindo, assim, a regra da unicidade.

A chave que identifica uma tabela é denominada de chave primária, podendo ser composta por um ou mais campos da tabela em questão. A especificação de uma chave define uma restrição de integridade, ou seja, uma regra que garante que a unicidade será obedecida em todos os estados do Banco de Dados.

#### Exemplo:

#### ALUNO

| Codigo | Nome  | Endereco |
|--------|-------|----------|
| 1      | Maria | Av. JK   |
| 2      | José  | Av. JB   |
| 3      | Pedro | Av. JC   |

#### ALUNO

| Nome  | Endereco |
|-------|----------|
| Maria | Av. JK   |
| José  | Av. JB   |
| José  | Av. JC   |

Não será inserido, pois a chave não permite a duplicidade, no caso da chave primária ser o nome.

Na tabela acima, o valor de um atributo chave é usado para identificar, unicamente, cada tupla, pois, o atributo “nome de aluno” pode receber um valor igual, ou seja, ter mais de um aluno com o mesmo nome. Se o campo “nome do aluno” for indicado como chave, será garantida a regra da unicidade, contudo, quando há a ocorrência de nomes iguais, apenas um é inserido, no caso o primeiro. Nesse caso, procede-se a criação de um campo, preferencialmente numérico, que fará a identificação da tupla, podendo assim, serem gravados nomes iguais que, entretanto, constituem tuplas diferentes.

#### Exemplo:

#### ALUNO

| Codigo | Nome  | Endereco |
|--------|-------|----------|
| 1      | Maria | Av. JK   |
| 2      | José  | Av. JB   |
| 3      | José  | Av. JC   |

O campo código permite a inclusão de homônimos, sem desrespeitar a regra da unicidade

#### CHAVE ESTRANGEIRA:

Uma chave estrangeira é um atributo ou uma combinação de atributos que aparecem necessariamente como chave primária em outra tabela.

#### ALUNO

| Cd_Aluno | Nome_Aluno | End_Aluno | Cd_Curso |
|----------|------------|-----------|----------|
| 1        | Maria      | Av. JK    | 3        |
| 2        | José       | Av. JB    | 2        |
| 3        | Pedro      | Av. JC    | 3        |

## CURSO

| Cd_Curso | Nome_Curso  | Sigla_Curso |
|----------|-------------|-------------|
| 1        | Informatica | INF         |
| 2        | Quimica     | QUI         |
| 3        | Biologia    | BIO         |

Nas tabelas acima podemos identificar que o **Cd\_Curso** da relação **CURSO** é uma chave estrangeira em relação a chave primária da tabela **ALUNO**. Com esta relação podemos identificar que todo aluno está relacionado a um curso através de um atributo Cd\_Curso.

Exemplo:

- O aluno José está vinculado ao curso de Química
- O aluno Pedro está vinculado ao curso de Biologia
- A aluna Maria está vinculada ao curso de Biologia

### CHAVE ALTERNATIVA:

Quando podemos identificar mais de uma chave em uma tabela, escolhemos uma como chave primária e as demais são consideradas chaves alternativas.

## PESSOA

| Codigo | Nome  | Telefone | CPF          |
|--------|-------|----------|--------------|
| 1      | Maria | 232323   | 90909090990  |
| 2      | José  | 242424   | 808080808080 |

No exemplo acima, podemos usar tanto o código como o CPF para identificar a unicidade da linha.

## 6.2 Restrições do modelo relacional

Até aqui vimos características de uma única tabela em um banco de dados, normalmente haverá muitas relações e as tuplas se relacionamento de diversas maneiras. Há muitas restrições em um Banco de dados para seus valores reais.

### Vamos ver algumas restrições no Banco de dados Relacional:

#### Restrição de Domínio:

Como já foi apresentado no item anterior, cada coluna de uma tabela é representada por um atributo. Quando uma tabela é definida, em cada coluna serão atribuídos valores para os campos. Ao conjunto de valores atribuídos chamamos de domínio do campo.

#### Para especificação de domínio, podemos:

- Definir o tipo de dado aceito pelos valores incluídos.
- Podemos usar como referência os tipos de dados oferecidos pelo padrão SQL-99

| Tipo de Dado                 | Descrição  |
|------------------------------|--|
| INTEGER ou INT               | Número positivo ou negativo inteiro. O número de bytes que pode ser utilizado varia em função do banco de dados utilizado.   |
| SMALLINT                     | Mesma função do INTEGER, mas ocupa cerca da metade do espaço.  |
| NUMERIC                      | Número positivo ou negativo de ponto flutuante. Normalmente, deve-se informar o tamanho total do campo e definir quantas casas decimais devem ser armazenadas após a vírgula.  |
| DECIMAL                      | Semelhante ao NUMERIC, mas, em alguns bancos de dados, poderá ter uma maior precisão após a vírgula.   |
| REAL                         | Número de ponto flutuante de simples precisão. A diferença básica é que os valores serão armazenados em representação exponencial, portanto serão arredondados para o nível mais próximo de precisão.  |
| DOUBLE PRECISION             | Número de ponto flutuante de dupla precisão. Comporta-se como o REAL, mas permite maior aproximação de resultados.   |
| FLOAT                        | Número de ponto flutuante em que você define o nível de precisão (número de dígitos significativos).   |
| BIT                          | Armazenamento de um número fixo de bits. O número de bits deve ser indicado, do contrário o padrão será 1.   |
| BIT VARYING                  | Igual ao BIT, permitindo armazenar valores maiores. Normalmente, utiliza-se para armazenamento de imagens.   |
| DATE                         | Permite armazenar datas.   |
| TIME                         | Permite armazenar horários.  |
| TIMESTAMP                    | Permite armazenar uma combinação de data e hora.   |
| CHARACTER ou CHAR            | Permite armazenar cadeias de caracteres (letras, símbolos e números). O tamanho deve ser informado e será fixo, ou seja, mesmo que não utilizado totalmente, será ocupado o espaço fisicamente. O valor definido será o tamanho máximo da cadeia armazenada. |
| CHARACTER VARYING ou VARCHAR | Permite armazenar cadeias de caracteres, mas com tamanho variável. Nesse caso, especifica-se o tamanho máximo da coluna. Se for utilizado menos espaço que o máximo definido, o espaço restante não será ocupado.  |
| INTERVAL                     | Intervalo de data ou hora.   |

Tabela extraída do livro: OLIVEIRA, Celso Henrique Poderoso. SQL Curso Prático. São Paulo: Novatec, 2002

## Restrição de Chave e Valores Null

### Restrição de Chave

No item 5, apresentamos as chaves primárias, estrangeira e candidata. Estas chaves especificam uma restrição de unicidade, na qual duas linhas diferentes serão distinguidas pela chave atribuída.

### Valores Nulos

Outro exemplo de restrição é a especificação de valores nulos. Podemos especificar, com isso que um campo não poderá ser nulo.

### Integridade da Entidade

Uma entidade íntegra estabelece os valores de chave primária não podem ser nulos, caso isso ocorra não poderemos distinguir as linhas através da chave primária.

## 6.3 Transformar um Modelo Conceitual em Modelo Relacional (modelo lógico)

O esquema de um Banco de dados relacional (modelo lógico) deve ter no mínimo as relações que formam o Banco, os atributos que as relações possuem e as restrições do modelo.

### 1ª REGRA - TRANSFORMAR ENTIDADE REGULAR EM UM RELAÇÃO



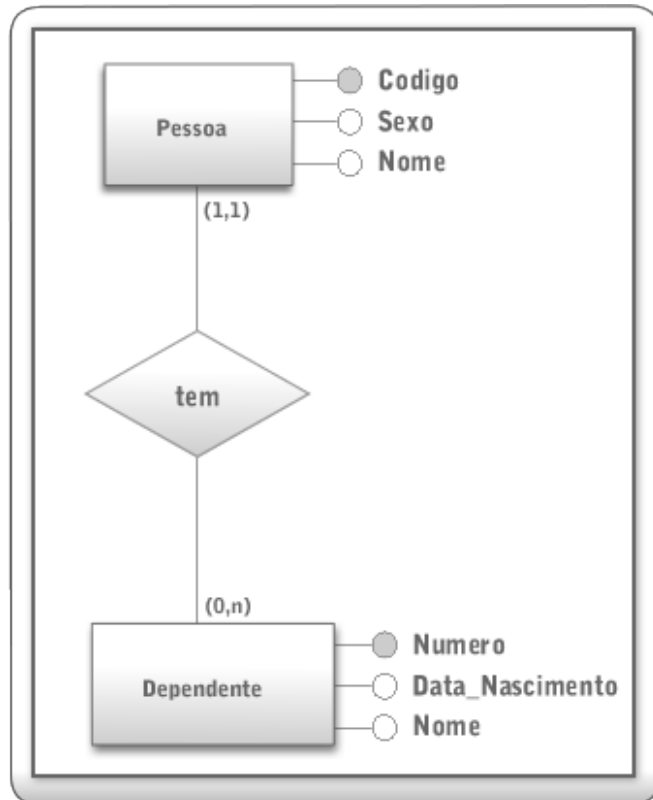
Para cada tipo de entidade regular transformar em uma relação que inclua todos os atributos simples

Em relação ao exemplo acima vamos fazer o primeiro regra para transformação, que é transformar toda entidade regular em uma relação que inclua todos os atributos simples, trata-se de uma tradução inicial que na seqüência poderá ser fundida com outra entidade.

Para representar o modelo lógico vamos utilizar as mesmas notações utilizadas na bibliografia do HEUSER que será explicado neste primeiro exemplo:

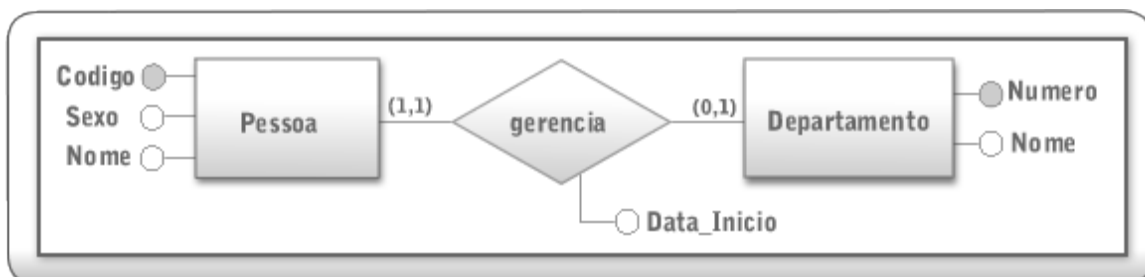
Pessoa(Codigo, Sexo, Nome)

Com isso a entidade PESSOA é transformada em uma tabela PESSOA e cada atributo da entidade é transformado em uma coluna da tabela sendo que o atributo chave primária será identificado com um sublinhado.

**2ª REGRA – TRANSFORMAR UMA ENTIDADE FRACA**

*Dependente(Numero, Codigo, Data\_Nascimento, Nome)*

Para cada tipo de entidade fraca transformar em uma relação que inclua todos os atributos simples e mais a chave primária da entidade dependente. Ambas passam a ser a chave primária composta da relação dependente.

**3ª REGRA – TRANSFORMAR RELACIONAMENTOS BINÁRIOS 1:1**

Quando temos este tipo de relação, há algumas opções para a transformação:

**1ª Opção:** Escolher uma das tabelas e inserir nela como chave estrangeira a chave primária do outra tabela. No nosso exemplo acima, temos a relação entre uma entidade pessoa e departamento.

Desta forma estamos apenas transformando as entidade regulares em tabelas:  
 Pessoa(Codigo, Nome, Sexo)  
 Departamento(Numero, Nome)

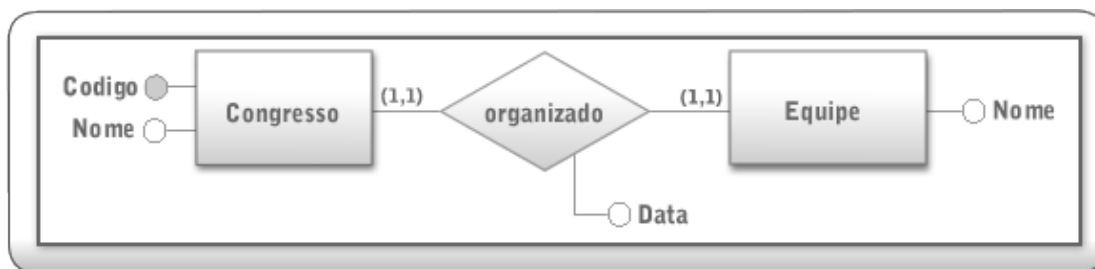
O próximo passo é analisar a relação existente. Como o relacionamento é binário 1:1 vamos usar a 1ª opção e inserir em uma das entidades a chave primária da outra. É melhor escolher, entre as duas entidades, aquela com participação total, ou seja, a entidade de departamento que preve a obrigatoriedade de ter um gerente. Ainda na entidade escolhida para receber a chave estrangeira também inserimos os atributos do relacionamento.

Pessoa(Codigo, Nome, Sexo)  
 Departamento(Numero, Codigo, Nome, Data\_Inicio)  
 Codigo referencia Pessoa

### Observação da Notação

Para representar a chave estrangeira, após a definição da relação que contém chave estrangeira descrevemos: <nome da coluna ch. Estrangeira> referencia <nome da relação>

**2º Opção:** Nesta opção, pode-se fazer uma fusão entre as entidades participantes. Normalmente ocorre quando as duas entidades têm participação total, ou seja, a cardinalidade mínima nas duas entidades é 1.



CongressoEquipe(Codigo, Nome, NomeEquipe, Data)

Neste exemplo, fizemos a fusão das duas entidades em uma única relação incluindo os atributos das duas entidades e do relacionamento. No exemplo acima tivemos que alterar o nome da equipe para NomeEquipe já que o atributo de congresso também estava identificado por Nome.

### DICAS:

Nos exemplos estamos colocando como nome de coluna os nomes dos atributos. Para os exercícios é importante que criemos uma nomenclatura para esta conversão. Por exemplo:

Codigo = Cod

Numero = No

A recomendação é que se use sempre a mesma abreviatura em todas as relações do Banco de Dados.

| Tipo de Relacionamento     | Regra de Implementação |               |               |
|----------------------------|------------------------|---------------|---------------|
|                            | Tabela Própria         | Adição Coluna | Fusão Tabelas |
| <b>Relacionamentos 1:1</b> |                        |               |               |
| (0,1)  (0,1)               | <u>+</u>               | ✓             | ✗             |
| (0,1)  (1,1)               | ✗                      | <u>+</u>      | ✓             |
| (1,1)  (1,1)               | ✗                      | <u>+</u>      | ✓             |

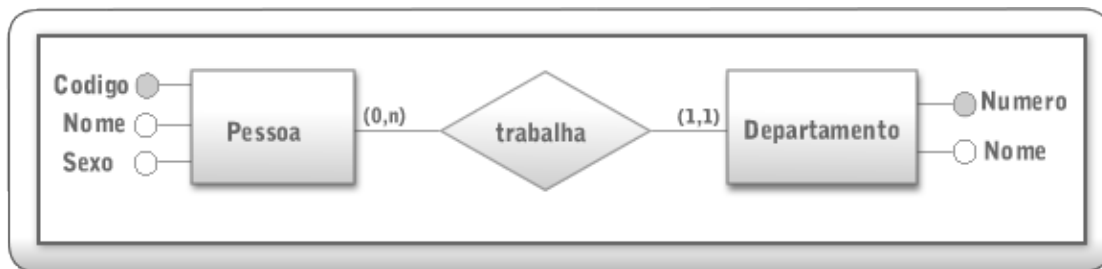
|          |                       |
|----------|-----------------------|
| <u>+</u> | Não pode ser          |
| ✓        | Alternativa Preferida |
| ✗        | Não usar              |

Tabela Proposta por: HEUSER, Carlos Alberto. Projeto de Banco de Dados. Porto Alegre: Sagra Luzato, 1999.



**4ª REGRA – TRANSFORMAR RELACIONAMENTOS BINÁRIOS 1:N**

Para cada tipo de relacionamento Binário 1:N identificar a entidade participante do lado N e inserir como chave estrangeira a chave primária do outra entidade. Isso é feito porque cada instância de entidade do lado N está relacionada a no máximo uma instância do lado 1.



Pessoa (Codigo, Nome, Sexo, Numero)

Numero referencia Departamento

Departamento(Numero, Nome)

No exemplo acima, inserimos a chave primária de departamento como chave estrangeira na relação Pessoa.

| Tipo de Relacionamento     | Regra de Implementação |               |               |
|----------------------------|------------------------|---------------|---------------|
|                            | Tabela Própria         | Adição Coluna | Fusão Tabelas |
| <b>Relacionamentos 1:n</b> |                        |               |               |
| (0,1)  (0,n)               | <u>+</u>               | ✓             | ✗             |
| (0,1)  (1,n)               | <u>+</u>               | ✓             | ✗             |
| (1,1)  (0,n)               | ✗                      | ✓             | ✗             |
| (1,1)  (1,n)               | ✗                      | ✓             | ✗             |

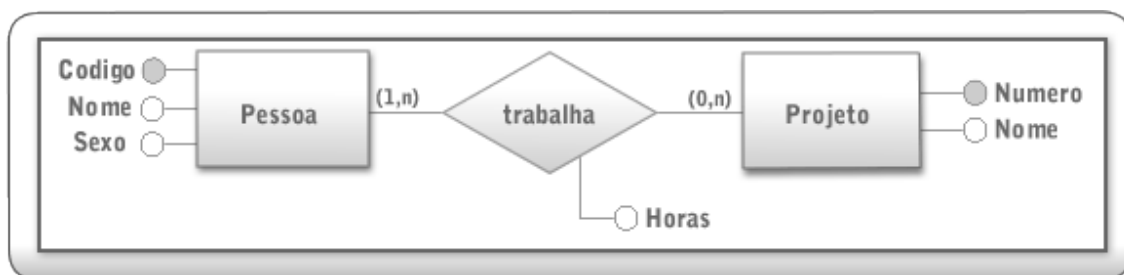
|          |                       |
|----------|-----------------------|
| <u>+</u> | Não pode ser          |
| ✓        | Alternativa Preferida |
| ✗        | Não usar              |

Tabela Proposta por: HEUSER, Carlos Alberto. Projeto de Banco de Dados. Porto Alegre: Sagra Luzato, 1999.

**5ª REGRA – TRANSFORMAR RELACIONAMENTOS BINÁRIOS N:M**

Neste caso, para cada tipo de relacionamento cria-se uma tabela nova. Inserir como chave estrangeira as chaves primárias das duas tabelas que fazem parte e as duas chaves irão representar a chave primária da nova tabela.

Também devemos incluir todo tipo de atributo que tiver na relação entre as entidades.



Pessoa(Codigo, Nome, Sexo)

Projeto(Numero, Nome)

Trabalha(Codigo, Numero, horas)

Código referencia Pessoa

Numero referencia Projeto

O Diagrama acima gerou 3 relações, uma para entidade pessoa outra para entidade projeto e outra para a relação trabalha.

A relação trabalha ficou composta de duas chaves primárias, código da entidade Pessoa e Numero da entidade Projeto. E ainda faz parte desta relação o atributo horas do relacionamento trabalha.

| Tipo de Relacionamento     | Regra de Implementação |               |               |
|----------------------------|------------------------|---------------|---------------|
|                            | Tabela Própria         | Adição Coluna | Fusão Tabelas |
| <b>Relacionamentos n:n</b> |                        |               |               |
| (0,n)  (0,n)               | ✓                      | ✗             | ✗             |
| (0,n)  (1,n)               | ✓                      | ✗             | ✗             |
| (1,n)  (1,n)               | ✓                      | ✗             | ✗             |

|   |                       |
|---|-----------------------|
| + | Não pode ser          |
| ✓ | Alternativa Preferida |
| ✗ | Não usar              |

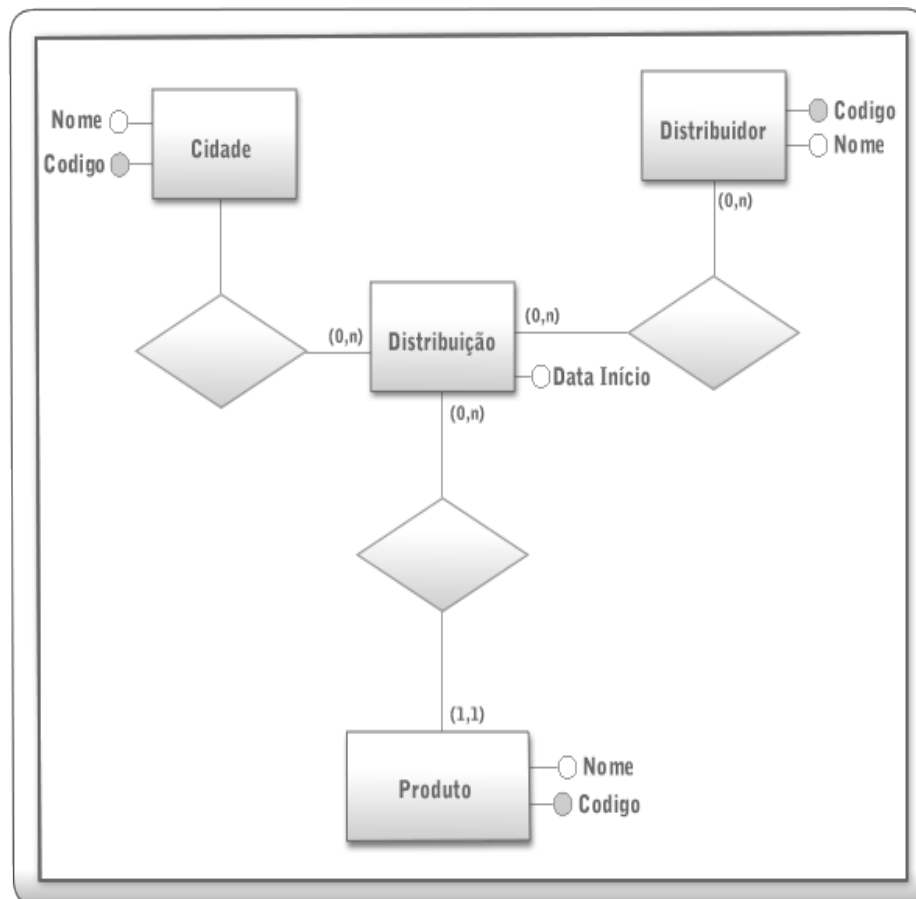
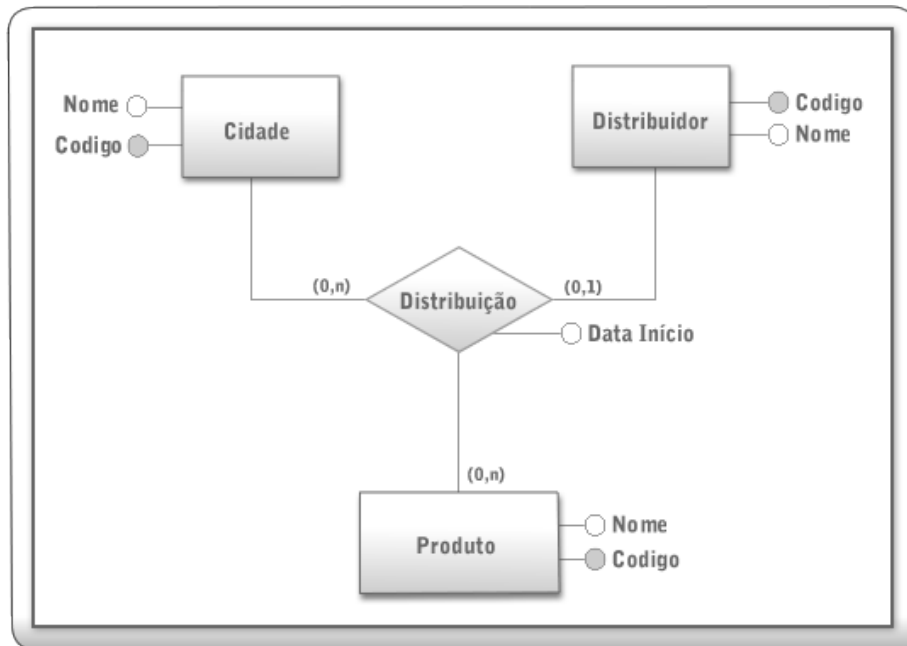
Tabela Proposta por: HEUSER, Carlos Alberto. Projeto de Banco de Dados. Porto Alegre: Sagra Luzato, 1999.

## 6ª REGRA – TRANSFORMAR RELACIONAMENTOS N-ário:

Para transformar relacionamentos n-ários deve seguir os seguintes passos:

1. Transformar o relacionamento em uma entidade e esta entidade fica ligado através de um relacionamento binário com as outras entidades.
2. Aplicar as regras para os relacionamentos binários.

Exemplo:



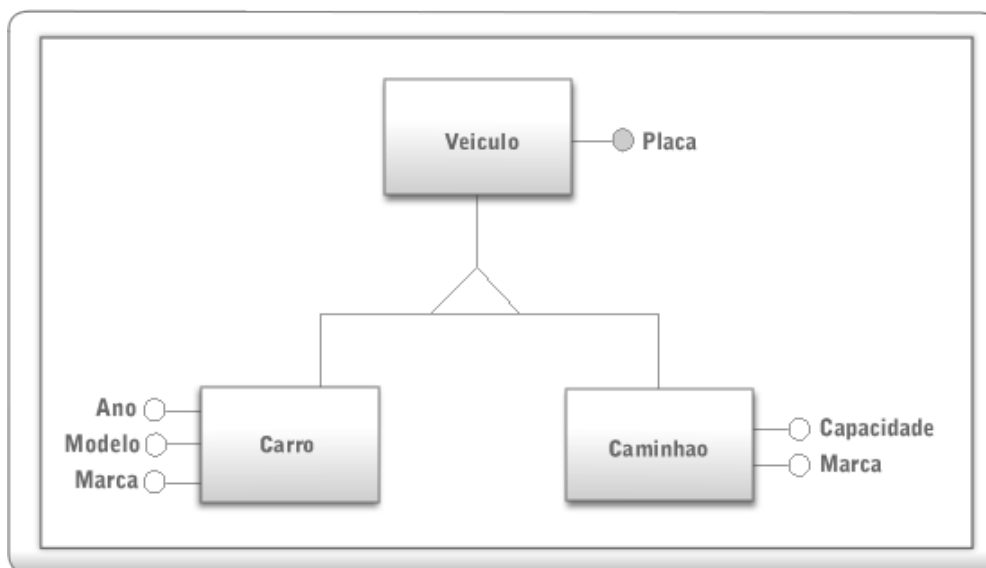
Seguindo os passos para a transformação acima, temos o seguinte modelo relacional:

- Produto(CodProd, Nome)
- Cidade(CodCid, Nome)
- Distribuidor(CodDistr, Nome)
- Distribuição(CodProd, CodCid, CodDistr, DataInicio)
- CodProd referencia Produto
- CodCid referencia Cidade
- CodDistr referencia Distribuidor

**7ª REGRA – TRANSFORMAR GENERALIZAÇÕES / ESPECIALIZAÇÕES**

Existem duas opções no caso de Generalização / Especialização:

- 1ª) Utilizar uma tabela para cada entidade
- 2ª) Utilizar uma tabela para toda Generalização / Especialização

**1º caso – Uma relação para cada entidade**

- Acrescentar à chave primária da tabela correspondente a entidade genérica, em cada tabela correspondente a uma entidade especializada.

Carro (Placa, Ano, Modelo, Marca)

*Placa referencia Veiculo*

Caminhao(Placa, Capacidade, Marca)

*Placa referencia Veiculo*

**2º caso – Fusão em um única relação**

Veiculo (Placa, Ano, Modelo, MarcaCarro, Capacidade, MarcaCaminhao)

No exemplo de modelo relacional acima temos a chave primária Placa que foi derivada da entidade Generalizada e todos os atributos das entidades especializadas se tornaram colunas na tabela.

**6.4 Normalização de dados**

Uma categoria importante de restrições são as dependências de dados, que incluem as dependências funcionais e as multivaloradas. Estas restrições são usadas para certificar o projeto de banco de dados relacional e são utilizadas em um processo chamado Normalização.

Este processo faz com que uma relação passe por diversos testes para que satisfaça uma **forma normal**.

A normalização de dados é uma seqüência de processos executados nas tabelas em função das suas dependências funcionais e chaves primárias para alcançar algumas propriedades como:

- Minimização de redundâncias;
- Minimização de anomalias de inserção, exclusão e atualização.

**Há cinco regras que se aplicam a Banco de Dados:**

- Primeira Forma Normal (1FN)
- Segunda Forma Normal (2FN)
- Terceira Forma Normal (3FN)

- Quarta Forma Normal (4FN)
- Quinta Forma Normal (5FN)

**OBS:**

*Pode-se ter um modelo estável atingindo a terceira forma normal sendo a quarta e a quinta utilizada em casos específicos apenas quando for necessário.*

Exemplo de uma tabela Não Normalizada:

| cd_Aluno | nome_aluno | dt_nascimento | nome_disciplina       |
|----------|------------|---------------|-----------------------|
| 1        | Vera       | 2/6/1982      | Português, matemática |
| 2        | Tânia      | 9/12/1981     | História, matemática  |
| 3        | Matias     | 12/11/1980    | Português, história   |
| 4        | Carlos     | 7/4/1979      | História, matemática  |
| 5        | Sérgio     | 25/2/1982     | Português, matemática |

#### 6.4.1 Primeira Forma Normal (1FN)

Para uma tabela estar na 1FN não poderá conter atributos multivalorados, compostos e combinação entre eles, com isso os atributos devem possuir valores atômicos (indivisíveis).

**Vamos utilizar o exemplo da tabela não normalizada acima.**

**Problema:** A coluna disciplina possui atributos multivalorados, que não pode haver na 1FN.

**Solução:** Para cada disciplina fazer um cadastro novo de Aluno

A primeira forma normal não permite também atributos multivalorados que sejam compostos. São chamados de relações aninhadas, pois contém uma relação em cada tupla.

| cd_aluno | cd_disciplina | nome_aluno | dt_nascimento | nome_disciplina |
|----------|---------------|------------|---------------|-----------------|
| 1        | 1             | Vera       | 2/6/1982      | Português       |
| 2        | 2             | Vera       | 2/6/1982      | Matemática      |
| 3        | 3             | Tânia      | 9/12/1981     | História        |
| 4        | 2             | Tânia      | 9/12/1981     | Matemática      |
| 5        | 1             | Matias     | 12/11/1980    | Português       |
| 6        | 3             | Matias     | 12/11/1980    | História        |
| 7        | 3             | Carlos     | 7/4/1979      | História        |
| 8        | 2             | Carlos     | 7/4/1979      | Matemática      |
| 9        | 1             | Sérgio     | 25/2/1982     | Português       |
| 10       | 2             | Sérgio     | 25/2/1982     | Matemática      |

**Problema:** Com isso, estamos ocasionando o problema de redundância de dados, pois os atributos de alunos estão sendo repetidos para a inclusão de novas disciplinas.

### 6.4.2 Segunda Forma Normal (2FN)

Para resolver o problema da redundância de dados, passamos esta tabela para segunda forma normal. Para isso é preciso: A tabela estar na primeira forma normal e todos os atributos da tabela deve ser dependentes funcionais da chave completa e não de parte da chave. Uma tabela com uma chave primária formada por apenas um atributo esta automaticamente na 2FN.

Exemplo transformando a tabela que esta na 1FN acima

| cd_aluno | nome_aluno | dt_nascimento |
|----------|------------|---------------|
| 1        | Vera       | 2/6/1982      |
| 2        | Tânia      | 9/12/1981     |
| 3        | Matias     | 12/11/1980    |
| 4        | Carlos     | 7/4/1979      |
| 5        | Sérgio     | 25/2/1982     |

| cd_disciplina | nome_disciplina |
|---------------|-----------------|
| 1             | Português       |
| 2             | Matemática      |
| 3             | História        |

### 6.4.3 Terceira Forma Normal (3FN)

Para a transformação na 3FN, eliminamos um outro tipo de redundância de dados. Para transformação precisamos analisar se os atributos não-chave dependem diretamente da chave primária ou se dependem de outros atributos não chave.

Chamamos de dependência transitiva quando uma coluna, além de depender da chave primária da tabela, depende também de outra coluna ou conjunto de colunas da tabela.

#### Referências:

ELMASRI, R. NAVATHE, *Sistemas de Banco de Dados. 4ª Edição: Addison-Wesley, 2008.*  
 HEUSER, Carlos Alberto. *Projeto de Banco de Dados. Porto Alegre: Sagra Luzzatto, 1998*