

# Fundamentos de Base de Dados

## Modelo Relacional

Pedro Nogueira Ramos

(Pedro.Ramos@iscte.pt)

ISTA/ DCTI

# Modelo Relacional

(Codd, 1970)

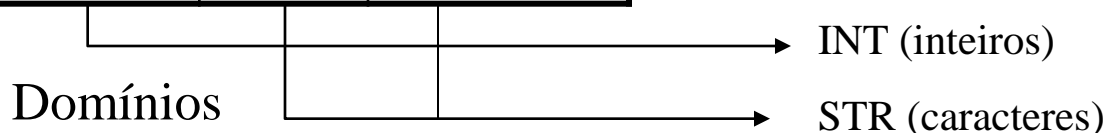
No Modelo relacional a informação é representada através de **relações** (ou tabelas). As relações correspondem a conjuntos, ou seja, são manipuladas através dos habituais operadores que operam sobre conjuntos: **Intersecção**, **Produto Cartesiano**, **União**, etc..

## Relação

Uma Base de Dados relacional é um **conjunto de relações** com um número específico de **atributos (colunas)** e um número variável de **tuplos (linhas ou instâncias)**. A cada atributo é atribuído um **domínio (conjunto de valores válidos)** e um **nome único** na relação

Relação: Cliente

Número	Nome	Morada
001	João	NULL
013	Ana	NULL
056	Luís	NULL

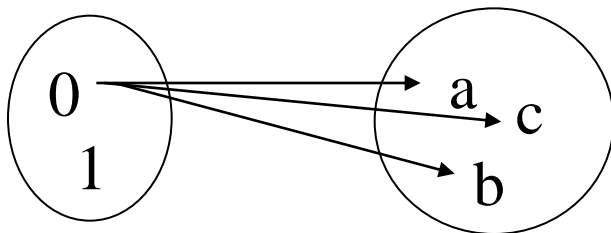


Um **valor** é armazenado na intersecção entre uma linha e um atributo e diz respeito ao domínio do atributo ou tem o valor NULL (ausência de valor).

## Produto Cartesiano

O Produto Cartesiano é obtido através de todas as combinações entre os elementos dos conjuntos.

Exemplo



Domínio D1 = {0,1}

Domínio D2 = {A,B,C}

Representação Relacional

DI	D2
0	a
0	b
0	c
1	a
1	b
1	c

Produto Cartesiano ( $D1 \times D2$ ) =  
 $\{(0,A), (0,B), (0,C), (1,A), (1,B), (1,C)\}$

## Definição de Relação

**Uma relação** é o subconjunto do Produto Cartesiano de uma lista de domínios. A tabela de clientes é um subconjunto do produto cartesiano entre os domínios INT e STR, nomeadamente  $\text{INT} \times \text{STR} \times \text{STR}$ .

Número	Nome	Morada
001	João	Lisboa
013	Ana	NULL

Qualquer subconjunto de  $\text{INT} \times \text{STR} \times \text{STR}$  é uma tabela (inclusive  $\text{INT} \times \text{STR} \times \text{STR}$  ou  $\text{INT} \times \text{STR}$ ).


**Um conjunto não é ordenado.** Isto é, caso seleccione elementos de um conjunto (e.g., linhas de uma tabela) sem indicar uma forma de ordenação, os elementos são seleccionados através de **uma ordem aleatória.**

## Chave Primária (I)

Todas as tabelas têm de possuir uma chave primária.

Chave Primária (ou chave): conjunto minimal de atributos que permitem **identificar univocamente** uma tuplo de uma relação.

O conjunto {Número} é chave porque não podem existir dois clientes com o mesmo número.



Número	Nome	Morada
001	João	NULL
013	Ana	NULL

O conjunto {Nome} **não é chave** porque podem existir dois clientes com o mesmo nome.

O conjunto {Número, Nome} **não é chave** porque **não é minimal** (contém uma chave).

## Chave Primária (II)

Apenas o conjunto {Fila, Lugar} ←  
**garante** que identificamos  
**apenas uma linha.**

Sala de Cinema

Fila	Lugar	Ocupado?
A	1	sim
A	2	não
B	1	não

Cliente

Número	Nome	Morada	BI
001	João	NULL	1234567
013	Ana	NULL	7654321

Chaves Candidatas (ou Alternativas)

É obrigatório  
optar por **uma**  
**única chave** !

## Critérios para adopção de uma Chave Primária (I)

- Atributos *familiares* ao utilizador
- Domínio Numérico (por razões de eficiência)
- Apenas um atributo (por razões de eficiência)
- Preenchimento Obrigatório



## Critérios para adopção de uma Chave Primária (II)

### Livro

Título	Editora	Edição	ID
Database Systems	Addison Wesley	5	OO1
Database Systems	Addison Wesley	6	002
UML, User Guide	Addison Wesley	NULL	003

Alternativa a { Título, Editora, Edição }

Apesar de *familiar*, é pouco eficiente e obriga ao preenchimento de Edição

## Chave Estrangeira (I)

As chaves estrangeiras ocorrem quando existem dependências entre domínios.

Cliente

Número	Nome	Morada
001	João	NULL
013	Ana	NULL
056	Luís	NULL

Factura

Número	Data	Cliente
001	12-12-1999	001
002	01-03-2000	013
0003	02-03-2000	001



O domínio de **Factura.Cliente** não é INT, mas sim: o conjunto de valores da coluna **Cliente.Número**. Ou seja, uma factura não pode estar associada a um cliente (Número) que não conste na tabela Cliente.

## Chave Estrangeira (II)

Factura.Cliente é Chave Estrangeira na tabela Factura.

Cliente

Número	Nome	Morada
001	João	NULL
013	Ana	NULL
056	Luís	NULL

Factura

Número	Data	Cliente
001	12-12-1999	001
002	01-03-2000	013
0003	02-03-2000	001

Dependência

1

0..\*

Cliente.Número não é Chave Estrangeira porque podem existir clientes sem facturas. A *existência* de um cliente não é condicionada à *existência* de facturas.

## Chave Estrangeira (III)

Cliente

Número	Nome	CodPostal
001	João	1500
013	Ana	2100
056	Luís	NULL

Localidade

CodPostal	Localidade
1500	Lisboa
2100	Porto
3999	Évora

0...\*

0 ... 1

**Cliente.CodPostal é Chave**

**Estrangeira** porque aos clientes não podem ser atribuídos códigos postais que não constem na tabela de localidades.

Mas **não é obrigatório** atribuir um Código Postal a um cliente.

$$\text{Cliente.CodPostal} \supseteq \{\text{Localidade.CodPostal}\} \cup \text{NULL}$$

## Integridade das Chaves Estrangeiras (Operação *Delete*)

Cliente

Localidade

Número	Nome	CodPostal
001	João	1500
013	Ana	2100
056	Luís	NULL

CodPostal	Localidade
1500	Lisboa
2100	<del>Porto</del>
3999	Évora

Hipótese: um utilizador  
pretende apagar a linha cujo  
CodPostal é 2100

Dado que  $\text{Cliente.CodPostal} \supseteq \{\text{Localidade.CodPostal}\} \cup \text{NULL}$ ,  
três alternativas se colocam ao gestor da base de dados:

1. Não permite apagar; (Restricted)
2. Permite apagar, mas apaga o cliente 013; (Cascade)
3. Permite apagar, mas substitui o CodPostal do cliente 013 por NULL.  
(Set Null)

## Integridade das Chaves Estrangeiras (Operação *Update*)

Cliente

Número	Nome	CodPostal
001	João	1500
013	Ana	2100
056	Luís	NULL

Localidade

CodPostal	Localidade
1500	Lisboa
2100	Porto
3999	Évora

Hipótese: um utilizador  
pretende alterar o CodPostal  
2100 para o valor 2200

Dado que  $\text{Cliente.CodPostal} \supseteq \{\text{Localidade.CodPostal}\} \cup \text{NULL}$ ,  
três alternativas se colocam ao gestor da base de dados:

1. Não permite alterar; (Restricted)
2. Permite alterar, mas altera igualmente o código postal do cliente 013 (de 2100 passa também para 2200); (Cascade)
3. Permite alterar, mas substitui o CodPostal do cliente 013 por NULL. (Set Null)

## Cruzamento de Informação

No Modelo Relacional a informação é obtida através do **cruzamento entre tabelas (produtos cartesianos)** através das **chaves estrangeiras**. Trata-se de uma forma fácil e intuitiva de obter informação, mas **pouco eficiente**.

Exemplo: Listar informação de clientes (incluindo localidades)

Cliente

Número	Nome	CodPostal
001	João	1500
013	Ana	2100
056	Luís	NULL

X

Número	Nome	CodPostal	Localidade
001	João	1500	Lisboa
013	Ana	2100	Porto
056	Luís	NULL	NULL

Localidade

CodPostal	Localidade
1500	Lisboa
2100	Porto
3999	Évora

Para cada Cliente.CodPostal procura-se um Localidade.CodPostal idêntico e selecciona-se a localidade respectiva.

## Joins

A informação obtida **unicamente através do produtos cartesiano** entre tabelas é inconsistente, daí a necessidade de **efectuar Joins**.

Cliente

Número	Nome	CodPostal
001	João	1500
013	Ana	2100
056	Luís	NULL

Localidade

CodPostal	Localidade
1500	Lisboa
2100	Porto
3999	Évora

$\times =$

Número	Nome	Cliente. CodPostal	Localidade. CodPostal	Localidade
<b>001</b>	<b>João</b>	<b>1500</b>	<b>1500</b>	<b>Lisboa</b>
001	João	1500	2100	Porto
001	João	1500	3999	Évora
013	Ana	2100	1500	Lisboa
<b>013</b>	<b>Ana</b>	<b>2100</b>	<b>2100</b>	<b>Porto</b>
013	Ana	2100	3999	Évora
056	Luís	NULL	1500	Lisboa
056	Luís	NULL	2100	Porto
056	Luís	NULL	3999	Évora

Únicas linhas coerentes: **Chave Primária = Chave Estrangeira** (*Key Join*)



## Transposição Modelo de Classes / Relacional

A transposição do modelo de classes para o modelo relacional tem como objectivo final a **criação de uma base de dados coerente** com a modelação da fase de análise.

As regras asseguram que

- 1) **não ocorre perda de informação**, i.e., é possível aceder a toda a informação;
- 2) **não existe informação redundante.**

As regras apresentadas não devem ser interpretadas como “leis” rígidas de transposição, mas uma indicação susceptível de adaptação em função da análise do problema em questão. As regras usualmente geram modelos relacionais ineficientes.

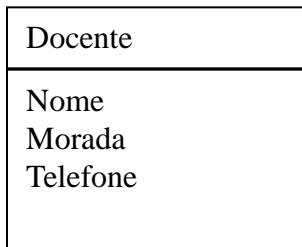
Na transposição existe perda de informação semântica relativa às relações entre as classes: a partir de um modelo relacional **pode não ser possível** obter o Diagrama de Classes a partir do qual ele foi gerado.

## Regras de Transposição

Regra 1 Todas as tabelas deverão ter uma chave primária. No caso de não existirem atributos que satisfaçam esta condição dever-se-á criar um identificador único usualmente designado por *id*.

Regra 2 As tabelas resultam exclusivamente das classes do modelo, e das associações de “muitos para muitos”.

Regra 3 Todos os atributos de uma classe são atributos da tabela que implementa a classe, inclusive os atributos das classes associativas.

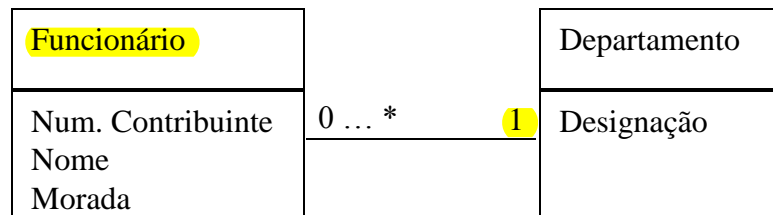


Docente (Nome, Morada, Telefone, ID)

## Regras de Transposição (“um/n” I)

### Regra 5: Transposição de Relações de Um para Muitos

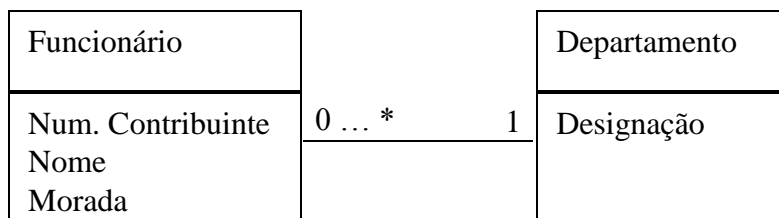
Numa relação de “um para muitos” a tabela cujos registos são susceptíveis de serem endereçados diversas vezes (lado *muitos*) herda a chave da tabela cuja correspondência é unitária (lado *um*).



**Funcionário** (NCont, Nome, Morada, Departamento)

**Departamento** (Designação)

## Regras de Transposição (“um/n” II)



Funcionário (NCont, Nome, Morada, Departamento)

Departamento (Designação)

Funcionário

<u>NCont</u>	Nome	Morada	Departamento
001	Ana	NULL	Produção
013	João	NULL	Produção

Departamento

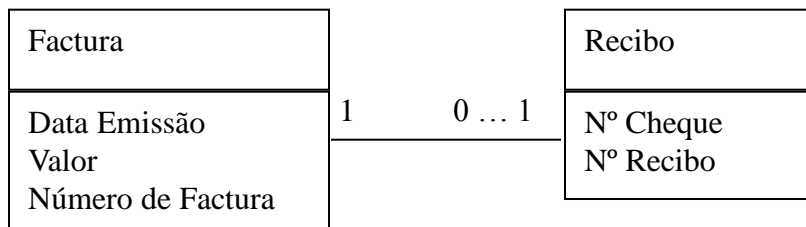
<u>Designação</u>
Produção
Comercial

Chave Estrangeira

## Regras de Transposição (“um/um” I)

### Regra 4 Transposição de Relações de Um para Um

Neste tipo de relação uma das tabelas da relação deverá herdar a chave principal da outra tabela como um atributo não chave (chave estrangeira). A determinação da tabela que herdará a chave estrangeira fica ao critério do analista e da interpretação que faz da realidade, devendo optar pelo que fizer mais sentido.



Factura (Data, Valor, NFact)

Recibo (NCheque, NRecibo, Nfact)

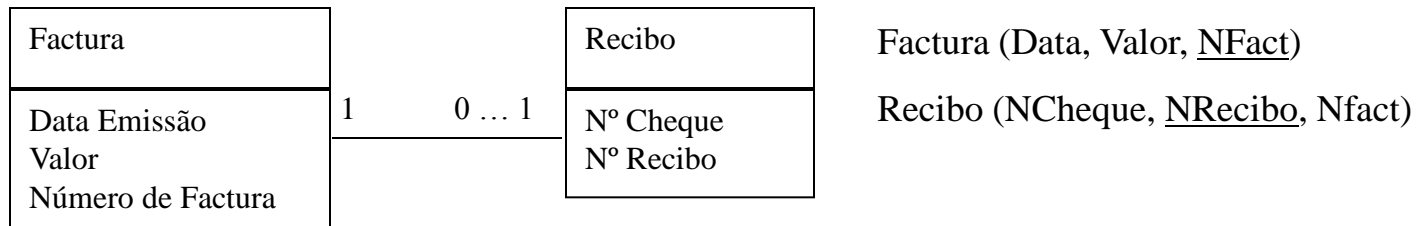
Factura (Data, Valor, Nfact, NRecibo)

Recibo (NCheque, NRecibo)

← Alternativa

(desvantagem: o registo de um novo recibo obriga a uma alteração na tabela de facturas)

## Regras de Transposição (“um/um” II)



Factura

<u>NFact</u>	Data	Valor
001	12-12-99	12000
013	02-01-00	13000

Recibo

<u>NRecibo</u>	NCheque	NFact
05	Null	001
07	Null	003

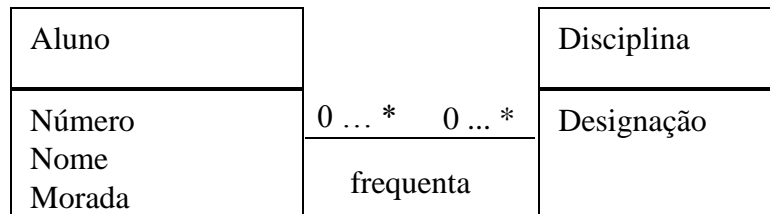


Chave Estrangeira

## Regras de Transposição (“n/n” I)

### Regra 6: Transposição de Relações de Muitos para Muitos

A relação dá origem a uma tabela representativa da associação onde a chave primária é composta pelos atributos chave das tabelas que implementam as classes associadas.

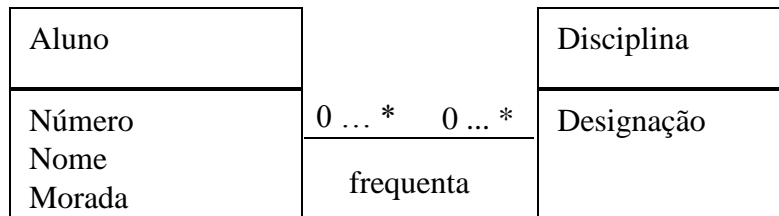


Aluno (NAluno, Nome, Morada)

Disciplina (Designação)

Frequenta (Naluno, Disciplina)

## Regras de Transposição (“n/n” II)



Aluno (NAluno, Nome, Morada)

Disciplina (Designação)

Frequenta (Naluno, Disciplina)

Aluno

<u>NAluno</u>	Nome	Morada
001	Ana	NULL
013	João	NULL

Frequenta

<u>NAluno</u>	<u>Disciplina</u>
001	Marketing
001	Comunicação

Disciplina

<u>Designação</u>
Marketing
Comunicação

Chave Estrangeira

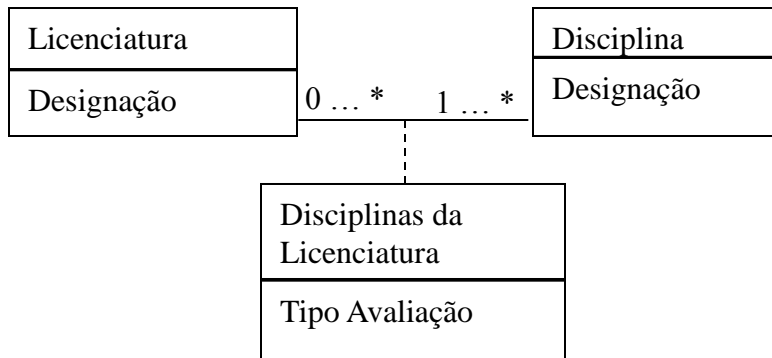
Chave Estrangeira



## Regras de Transposição (Classes Associativas)

### Regra 7: Transposição de Classes Associativas

Para as Classes Associativas aplicam-se as regras correspondentes à associação. Os atributos da classe associativa são herdados pela(s) tabela(s) que herda(m) a(s) chave(s) estrangeira(s).



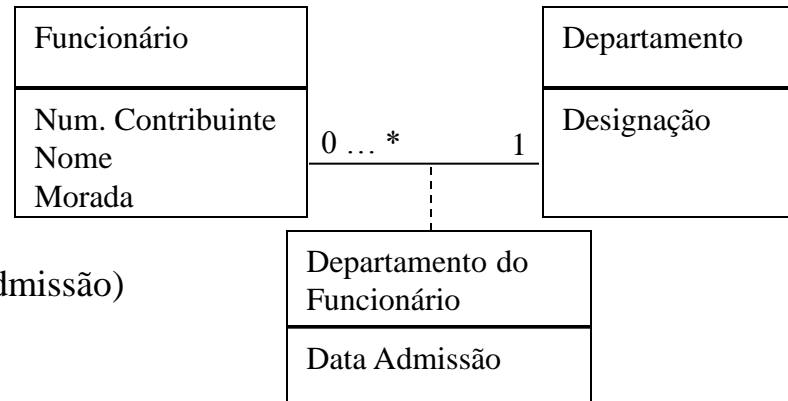
Licenciatura (Designação)

Disciplina (Designação)

DisCLic (Licenciatura, Disciplina, TipoAval)

Departamento (Designação)

Funcionário (Ncont, Nome, Morada, Departamento, DtAdmissão)



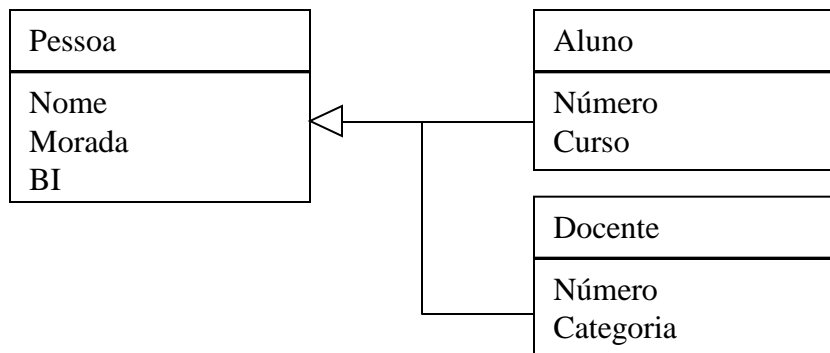
## Regras de Transposição (Generalizações I)

### Regra 8: Transposição de Generalizações

Duas situações distintas podem ocorrer

a) As classes *filhas* têm **entidade própria independentemente** da classe *pai*

A chave das tabelas que implementam as classes *filhas* é obtida através dos atributos da própria tabela. Terá que ser criado um atributo chave para a tabela que implementa a classe pai, sendo que essa tabela deverá ter uma propriedade discriminante (um atributo) que indica a qual das *filhas* o registo diz respeito. Todos os atributos chave da tabela *pai* terão que constar nas tabelas *filhas* como atributos não chave.

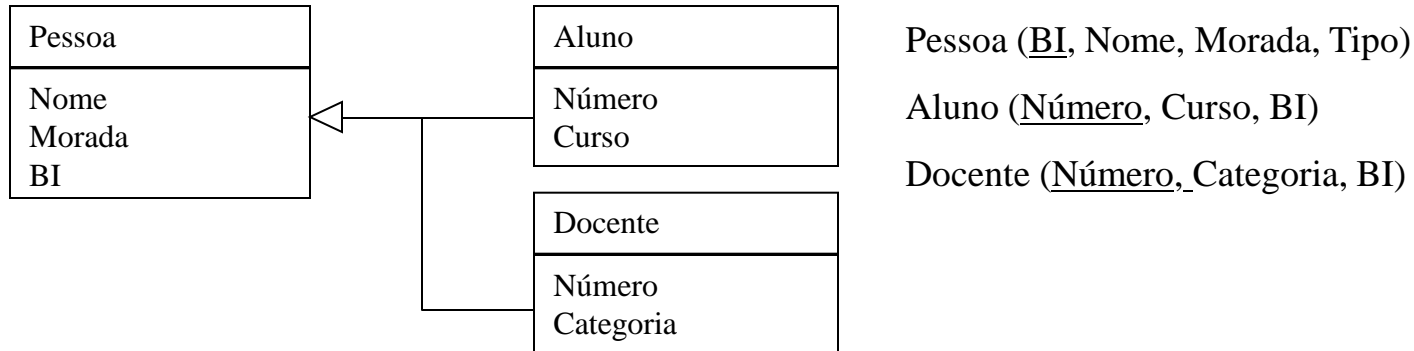


Pessoa (BI, Nome, Morada, Tipo)

Aluno (Número, Curso, BI)

Docente (Número, Categoria, BI)

## Regras de Transposição (Generalizações II)



Aluno

<u>Número</u>	Curso	BI
001	História	123456

Docente

<u>Número</u>	Categoria	BI
10	Assistente	321456

Pessoa

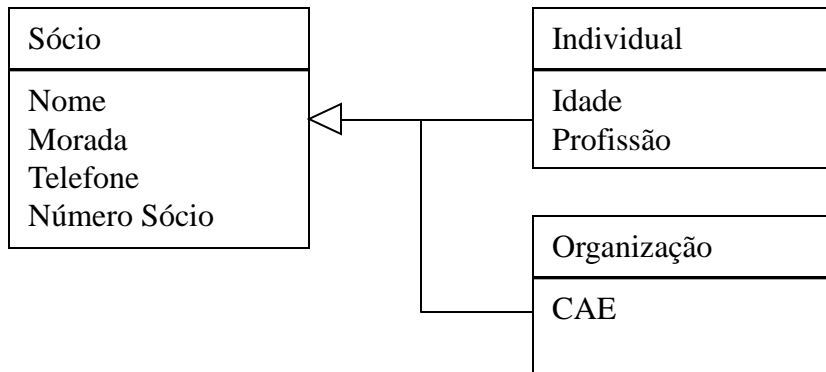
<u>BI</u>	Nome	Morada	Tipo
123456	João	NULL	A
321456	Ana	NULL	D

Chave Estrangeira

## Regras de Transposição (Generalizações III)

b) As classes *filhas* só têm identidade enquanto associadas à classe *pai*

Neste caso, a chave da tabela que implementa a classe *pai* é obtida através dos atributos da própria tabela. As tabelas correspondentes às classes *filhas* herdarão a mesma chave da tabela *pai*.

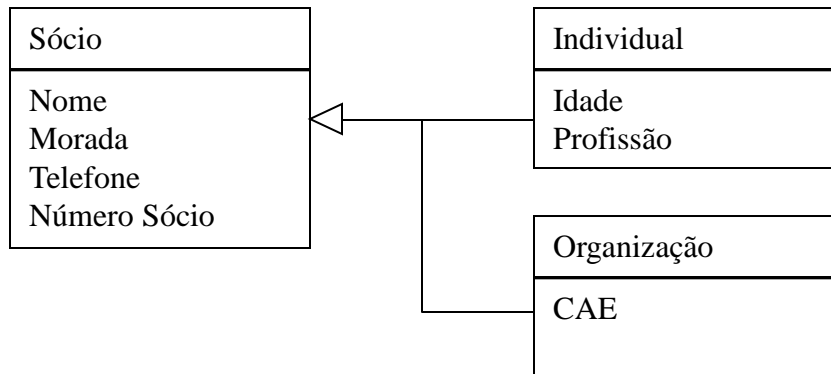


Sócio (NSócio, Nome, Morada, Telefone)

Individual (NSócio, Idade, Profissão)

Organização (NSócio, CAE)

## Regras de Transposição (Generalizações IV)



Sócio (NSócio, Nome, Morada, Telefone)

Individual (NSócio, Idade, Profissão)

Organização (NSócio, CAE)

Individual

<u>NSócio</u>	Idade	Profissão
11	32	Docente

Organização

<u>NSócio</u>	CAE
10	A.99.8872

Sócio

<u>NSócio</u>	Nome	Morada	Telefone
10	João	NULL	21345676
11	Ana	NULL	22456543

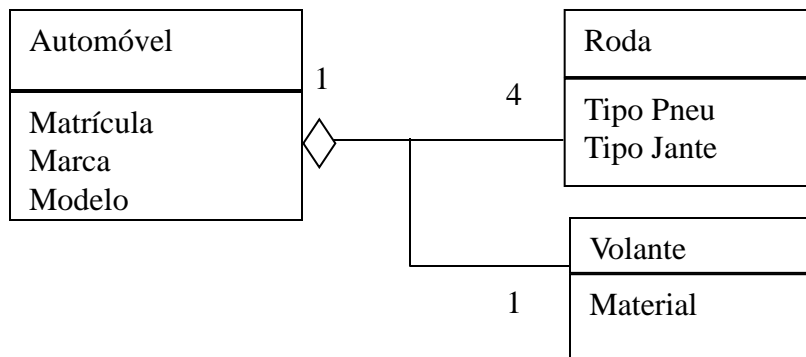
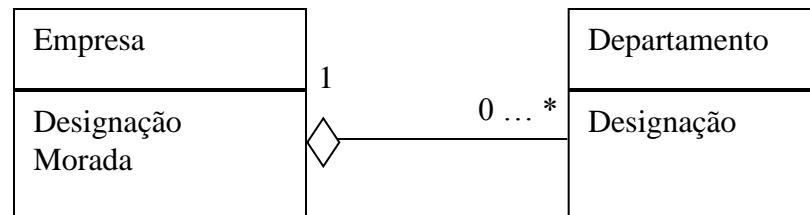
Chave Estrangeira

## Regras de Transposição (Agregação)

A transposição para o relacional obedece as regras de transposição das associações com a mesma multiplicidade.

Empresa (IDEmp, Designação, Morada)

Departamento (IDDep, Designação, IDEmp)



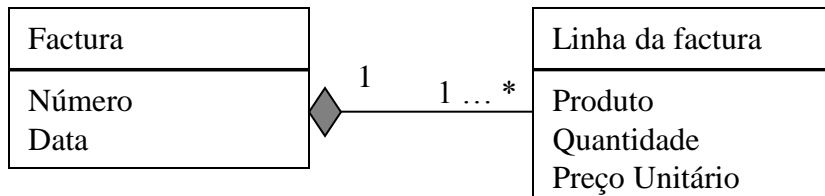
Automovel (Matrícula, Marca, Modelo)

Roda (IDRoda, TipoPneu, TipoJante, Matrícula)

Volante (IDVol, Material, Matrícula)

## Regras de Transposição (Composição I)

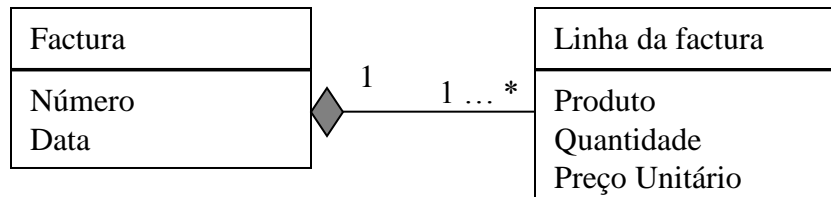
A tabela que implementa a classe composição tem como atributos chave a chave da tabela correspondente à classe que representa a composição e um atributo (ou mais) pertencente à classe componente (caso não exista é necessário criá-lo).



Factura (NFact, Data)

Linha (Nfact, NLinha, Produto, Quantidade, PreçoUnit)

## Regras de Transposição (Composição II)



Factura (NFact, Data)

Linha (Nfact, NLinha, Produto, Quantidade, PreçoUnit)

Factura

<u>NFact</u>	Data
11	03-03-00
12	03-03-00

Linha

<u>NFac</u> <u>t</u>	<u>Nlinha</u>	Produto	Quantidade	PreçoUnit
11	1	ProdA	3	2000
11	2	ProdB	4	4000
12	1	ProdB	3	2000

Chave Estrangeira



## Especificação de Atributos

Na especificação dos atributos, para além da sua designação e tipo de dados, é possível indicar outras propriedades:

- Chave primária;
- Chave Estrangeira
- **Allow NULLS** – admite o valor null
- **Unique** – não admite valores duplicados (**para criar chaves alternativas; pode agrupar vários atributos**)
- Validações (**CHECK**) – regras simples com operadores lógicos (<, >, <>, or, and) e.g., IN (lista de valores);
- Valor por omissão;
- Comentários.

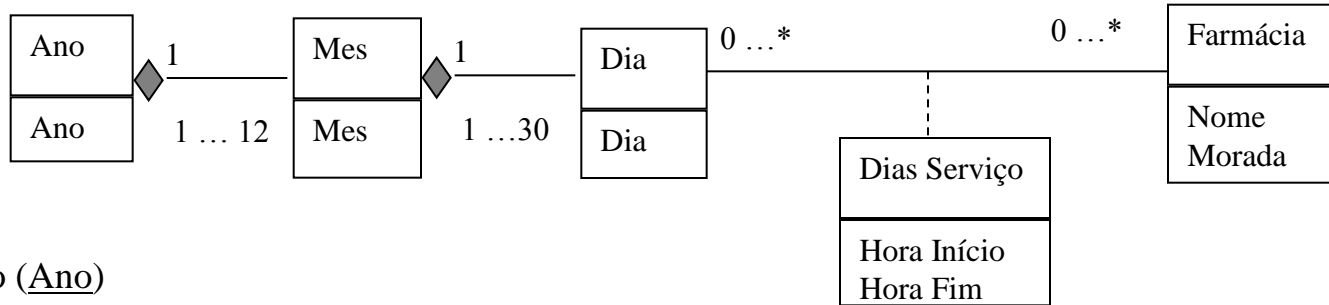
## Optimizações do Modelo Relacional

As regras de transposição, apesar de assegurarem um modelo completo (sem perda de informação) e coerente, geram usualmente modelos ineficientes. Sempre que possível (desde que não haja perda de informação relevante) dever-se-à otimizar o modelo relacional obtido, nomeadamente, no que diz respeito a:

- a) **Número de tabelas** (um elevado número de tabelas – **joins** – pode comprometer a eficiência do modelo;
- b) **Número de atributos que compõem a chave das tabelas** (deve ser reduzido).

Ao contrário de um diagrama de classes, o modelo relacional não pretende ser descritivo, mas **sim eficaz e eficiente.**

## Exemplos de Optimizações (I)



Ano (Ano)

Mes (Ano, Mes)

Dia (Ano, Mes, Dia)

Farmácia (IDFarmacia, Nome, Morada)

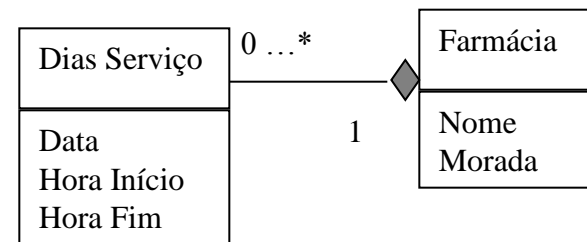
Dias-Servico (IDFarmacia, Ano, Mes, Dia, Hora\_Inicio, Hora\_Fim)

As três classes para modelar a data podem-se justificar para realçar o facto de não haver periodicidade (semanal, mensal) nos dias de serviço.

## Utilidade das tabelas Ano, Mês e Dia ?

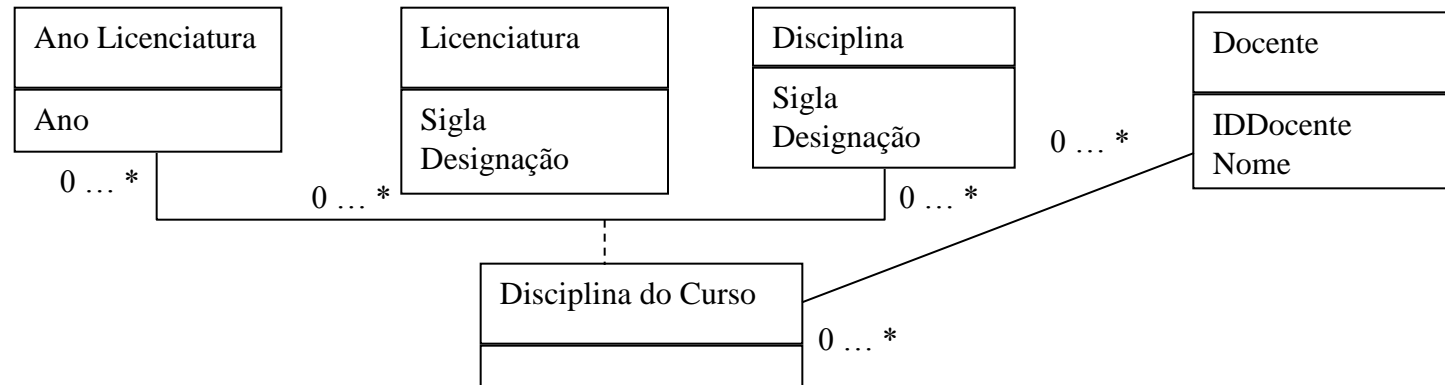
Farmácia (IDFarmacia, Nome, Morada)

Dias-Servico (IDFarmacia, Data, Hora\_Inicio, Hora\_Fim)



Apenas um atributo

## Exemplos de Optimizações (II)



Ano Licenciatura (Ano) Licenciatura (Sigla, Designação) Disciplina (Sigla, Designação)

Disciplina do Curso (Sigla Disciplina, Sigla Curso, Ano) Docente (IDDocente, Nome)

Lecciona (IDDocente, Sigla Disciplina, Sigla Curso, Ano)

Chave ineficiente

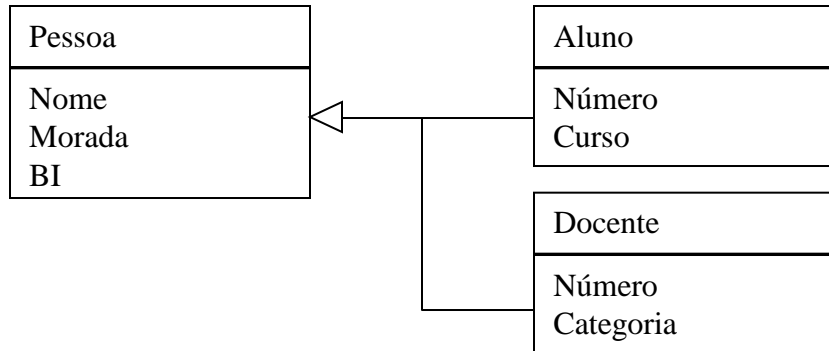
~~Ano Licenciatura~~ (Ano) Licenciatura (Sigla, Designação) Disciplina (Sigla, Designação)

Disciplina do Curso (Sigla Disciplina, Sigla Curso, Ano, IDDis) Docente (IDDocente, Nome)

Lecciona (IDDocente, IDDis)

A validação da chave original terá que ser feita por programação

## Exemplos de Optimizações (III)



Pessoa (BI, Nome, Morada, Tipo)

Aluno (Número, Curso, BI)

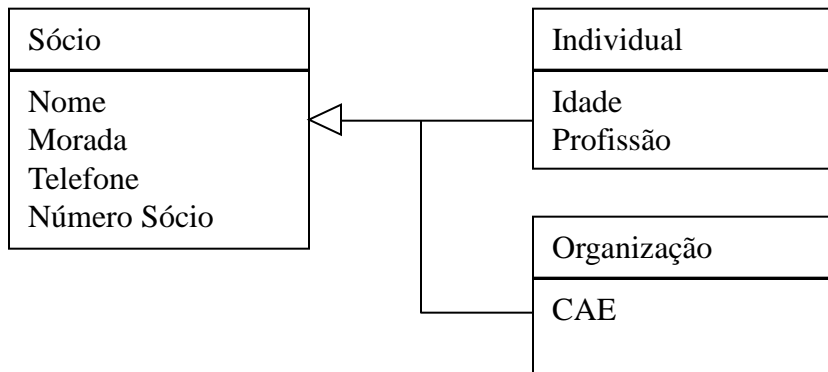
Docente (Número, Categoria, BI)



Aluno (Número, Curso, BI, Nome, Morada)

Docente (Número, Categoria, BI, Nome, Morada)

Facilita listagens de alunos (ou docentes) mas penaliza *mailings*.



Sócio (NSócio, Nome, Morada, Telefone)

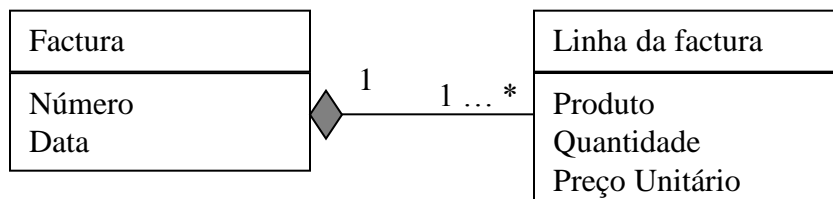
Individual (NSócio, Idade, Profissão)

Organização (NSócio, CAE)



Sócio (NSócio, Nome, Morada, Telefone, Idade, Profissão, CAE, Tipo {Individual, CAE})

## Exemplos de Optimizações (IV)



Factura (Número, Data)

Linha (Número, Item, Produto, Quantidade, PU)

Pode obrigar a re-cálculos de subtotais e totais (e.g., para estatísticas)



Factura (Número, Data, Total)

Linha (Número, Item, Produto, Quantidade, PU, SubTotal)

Para além dos perigos da redundância, penaliza o cálculo de cada factura (*update* na tabela de factura).

## Índices (I)

Uma técnica habitual de optimização de **interrogações (*queries*)** a bases de dados consiste na utilização de índices. O objectivo é acelerar o acesso a uma tabela através de um (ou vários) campo(s).

Hipótese: é frequente as publicações serem consultadas pelo assunto

Ficheiro de Índice

Assunto	Índice
Direito	
História	
Informática	
Informática	
Sociologia	

Tabela de Publicação

ISBN	Título	Data	Assunto
1	A	1998	História
2	D	1997	Informática
3	C	1994	Sociologia
4	Z	2000	Informática
5	F	1986	Direito



**Ficheiro ordenado pela primeira coluna**

## Índices (II)

Ficheiro de Índice

Assunto	Índice
Direito	
História	
Informática	
Informática	
Sociologia	

Tabela de Publicação

ISBM	Título	Data	Assunto
1	A	1998	História
2	D	1997	Informática
3	C	1994	Sociologia
4	Z	2000	Informática
5	F	1986	Direito

### Exemplos de interrogações que beneficiam do índice

1. Quais os títulos das publicações de História ? (mesmo que a maioria das publicações sejam de história, é **mais rápido** percorrer **sequencialmente** um ficheiro mais pequeno);
2. Quantas publicações de informática existem (apenas necessita de **abrir o ficheiro de índices**);
3. (se campo data indexado) Quais os títulos entre 1990 e 1998 ? (a **ordenação do índice** acelera **muito a procura por intervalos**).

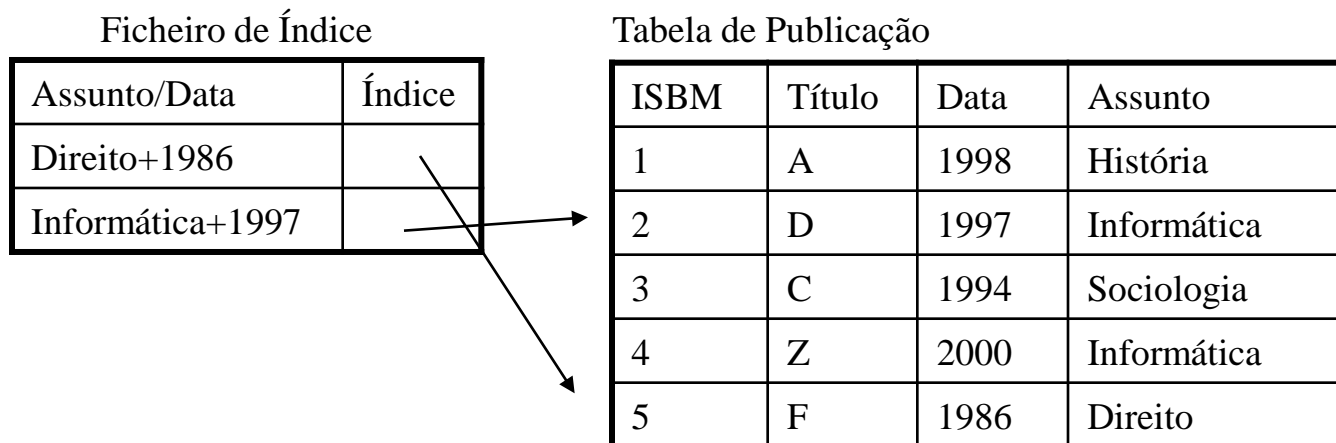


## Índices (III)

É usual criar-se um índice para a chave primária dado que a forma privilegiada de acesso às tabelas é através da chave, e.g., *joins*).

Pela mesma razão também se opta por vezes por criar índices para as chaves estrangeiras.

Pode ser criado um índice para dois ou mais atributos.



## Índices (IV)

Apesar de acelerarem a consulta de informação, os índices penalizam a introdução e alteração de informação.

A inserção de um registo obriga à introdução de um registo (ordenado) na tabela de índices. A alteração de um valor num atributo indexado obriga ao reordenamento do ficheiro de índices.

A **gestão de índices** é fundamental mas *perigosa* (uma má gestão – excesso – pode degradar muito o desempenho da base de dados) e **nunca é definitiva**. Depende do número de registos existentes e no tipo de acessos (consultas) mais frequentes.

## Índices (V)

### Exemplos de acessos

Tabela: Publicacao (ISBN (Int), Assunto (Str), Ano (Int), Titulo (Str))  
100.000 registos

Inserção: ISBN sequencial, restantes 3 aleatórios

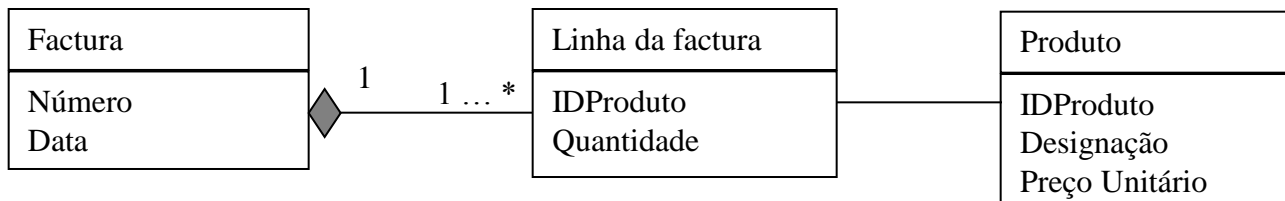
Consulta: Número de publicações com um determinado assunto

Valores em segundos

Índice	Inserção	Consulta
Nenhum	398	2
ISBN	402	2
ISBN, Assunto	<b>547 (+37%)</b>	0

## Arquivo

O arquivo (histórico / *backup*) de informação por vezes origina problemas de consistência.



Hipótese: o preço de um produto foi alterado e é necessário imprimir segunda via de uma factura antiga. **Três alternativas de *backup*:**

- a) Cópia integral da base de dados – **é sempre coerente;**
- b) Cópia de algumas tabelas – **pode gerar incoerência.**
- c) Tabelas próprias para *backup* (**alternativa a b)**)



Linha (Num\_Fact, Linha, IDProduto, Designação, PU, Quantidade)

## Parameterização

Todas as constantes de uma aplicação devem estar em uma(s) tabela, e nunca no código. Normalmente essa tabela apenas contém uma linha e uma coluna para cada parâmetro da aplicação.

Exemplos de atributos:

Taxa IVA;

Designação e Morada da Empresa (para impressões);

Dia do mês em que são automaticamente processados os salários;

Prazo de devolução de uma publicação (biblioteca);

etc.