



Matemática e Informática
Módulo IV

Bases de Dados Relacionais
(linguagem SQL)

Departamento de Matemática
Instituto Superior de Agronomia

2009/2010

Utilidade das Bases de Dados

- 📄 Recolha e processamento de dados
 - que possuem um volume significativo,
 - que são interrelacionados,
 - de forma a que não existam perdas de informação,
 - cuja organização permita a sua fácil compreensão e
 - que o seu processamento seja eficiente.

Eficácia!!!

Base de Dados

É um conjunto de dados relacionados que é utilizado recorrendo a um **Sistema Gestor de Bases de Dados** (SGBD).

- Uma **Base de Dados** representa uma parte da realidade que é relevante para a resolução de um determinado conjunto de problemas.

Existem no mercado diversos SGBD, os mais utilizados são os SGBD **relacionais** que permitem criar bases de dados relacionais.

SGBD relacional

Os conceitos em que se baseia foram apresentados em 1970.

Desde então vários fornecedores de software informático têm vindo a desenvolver SGBD relacionais.

- O OpenOffice.org Base é um SGBD relacional
 - pode ser utilizado em computadores com sistema operativo Windows ou Linux
- O Microsoft Access é também um SGBD relacional
 - destina-se a ser utilizado em computadores com sistema operativo Windows
- O ORACLE é outro exemplo de SGBD relacional
 - muito utilizado em computadores com sistema operativo Windows ou Linux

Métodos de desenvolvimento de bases de dados

Os mais divulgados incluem:

- uma fase de criação de um modelo conceptual de dados
- uma fase de conversão deste modelo para o SGBD escolhido para a sua implementação

Um modelo conceptual de dados pode ser criado utilizando conceitos e regras de tipo **Entidade-Associação (E-A)**.

Vantagens

- necessita apenas de um conjunto diminuto de conceitos e regras simples,
- representação por diagramas com uma notação muito elementar.

Conceitos para modelos conceptuais de tipo E-A

- 📄 Os modelos de tipo E-A são compostos por **entidades** e **associações**.
- 📄 As entidades representam objectos reais que possuem uma descrição que é determinada pelos problemas que se pretendem resolver por meio da Base de Dados.
- 📄 As associações representam relacionamentos relevantes entre entidades.

Tipo de entidades e atributos

- As entidades que possuem uma descrição comum são representadas por um **tipo de entidade**, o qual é caracterizado por um conjunto de **atributos**.
- Os **valores** que tomam cada um dos atributos de um tipo de entidade permitem descrever e distinguir entre si as entidades que pertencem a um mesmo tipo de entidade.

Exemplo

- ☰ Numa base de dados que se destina a servir os alunos de uma Faculdade:
 - os alunos, os professores e as disciplinas são entidades
 - aluno, professor e disciplina são tipos de entidade
 - identificados, respectivamente, por **Aluno**, **Professor**, **Disciplina**
 - nº de aluno (**nAluno**), nome de aluno (**nomeA**) e data de nascimento (**datanasc**) são atributos do tipo de entidade **Aluno**; código da disciplina (**código**), designação (**designação**) e créditos (**créditos**) são atributos do tipo de entidade **Disciplina**
 - 12345, José Silva, 1980-11-30 são valores dos atributos₈
...

Chave primária de um tipo de entidade

☰ A um atributo de um tipo de entidade cujos valores permitam identificar de forma inequívoca cada uma das entidades desse tipo dá-se o nome de **chave primária**.

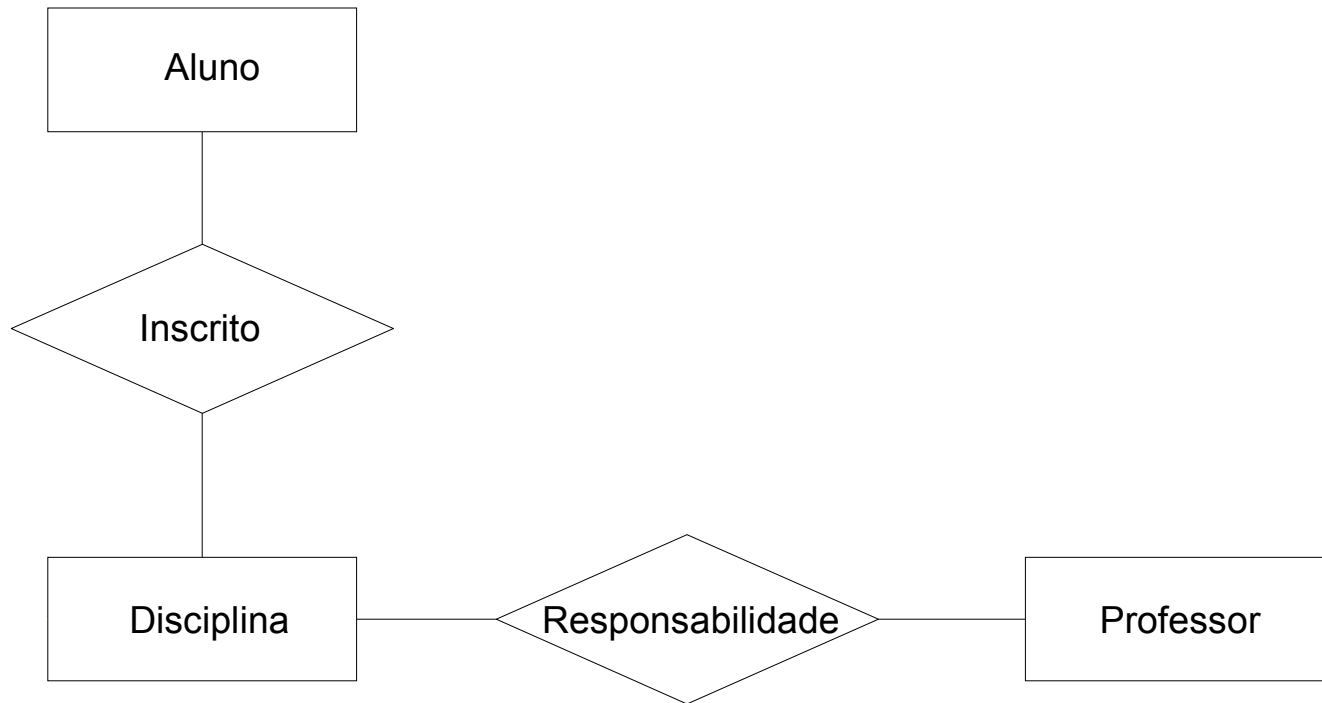
☰ Regra: cada tipo de entidade deve possuir uma chave primária.

- no exemplo, cada entidade do tipo **Aluno** pode ser identificada pelo **nAluno** e cada entidade do tipo **Disciplina** pode ser identificada pelo **código**
- alunos distintos terão números diferentes; duas disciplinas diferentes terão códigos diferentes.

Associação e tipo de associação

- Entre alunos e disciplinas existem associações, entre professores e disciplinas também existem associações ...
- Um **tipo de associação** representa as associações com características comuns que se estabelecem entre as várias entidades.
 - no exemplo, pode ser criado um tipo de associação, com o nome **Inscrito**, que representa as associações que se estabelecem entre as entidades do tipo **Aluno** e as entidades do tipo **Disciplina**; pode ainda ser criado um tipo de associação **Responsabilidade** para representar as associações existentes entre as entidades do tipo **Professor** e as entidades do tipo **Disciplina**

Diagrama do modelo conceitual

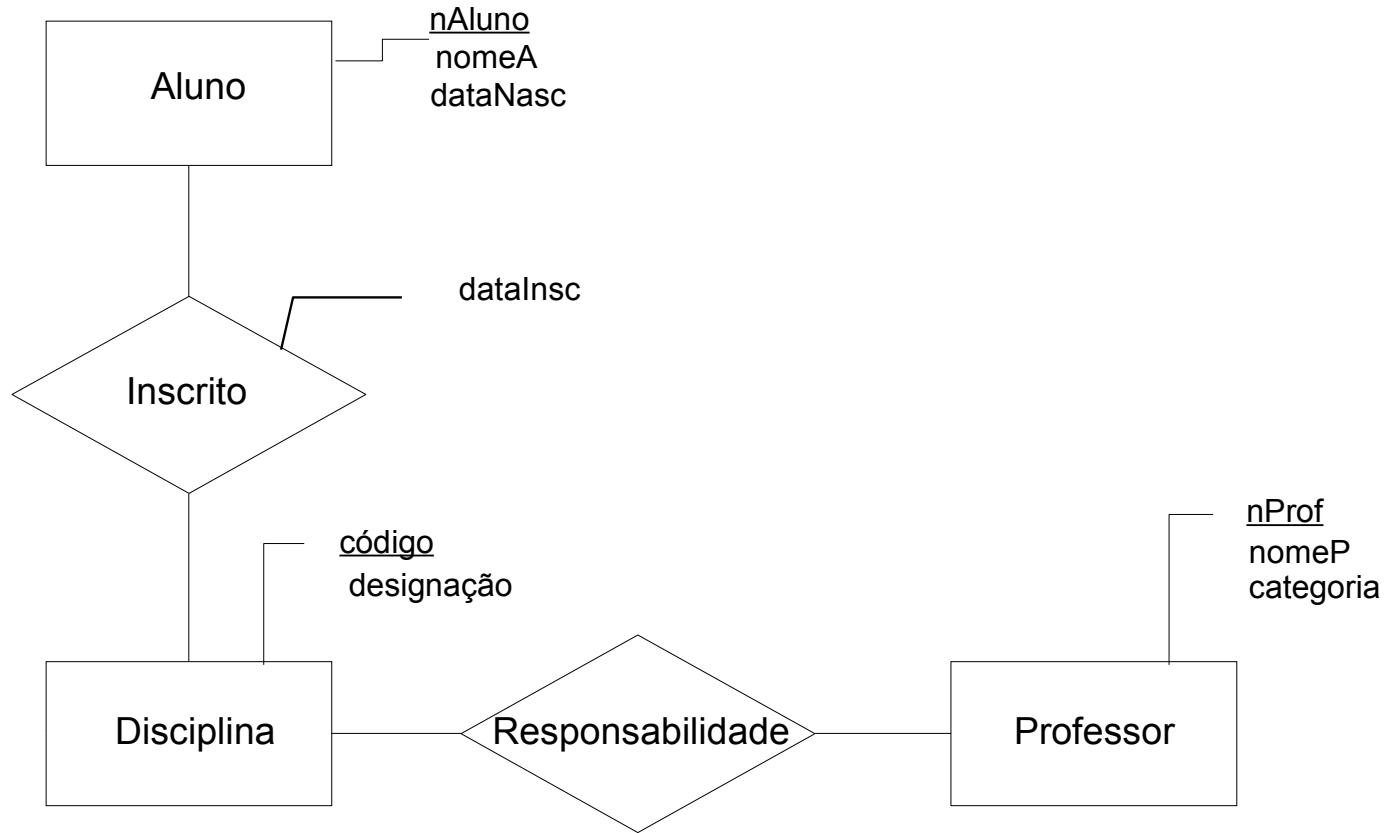


Atributos de um tipo de Associação


Um tipo de associação pode possuir **atributos** que permitam descrever as características próprias de cada associação.

- como atributo do tipo de associação **Inscrito** pode incluir-se a data de inscrição dos alunos na disciplina (`dataInsc`).

Diagrama do modelo conceptual



Cardinalidade de um tipo de associação

 Um tipo de associação caracteriza-se pelo número de tipos de entidade que envolve, podendo ser unária, ou binária (o caso mais frequente), ou ternária, ou ...

Multiplicidade de um tipo de associação

- ☰ A multiplicidade do tipo de associação indica o modo como se podem estabelecer associações desse tipo.
- ☰ Um tipo de associação binária pode ser:
 - de **um para um** (1:1) quando cada entidade só pode ocorrer numa única associação desse tipo;
 - de **um para muitos** (1:n) quando cada entidade de um tipo só pode ocorrer numa associação, mas as entidades do outro tipo podem ocorrer em mais do que uma associação desse tipo;
 - de **muitos para muitos** (n:n) quando não existe nenhuma restrição ao número de ocorrências de cada entidade em associações desse tipo.

Exemplos

- 📄 O tipo de associação *Responsabilidade* entre os tipos de entidade *Professor* e *Disciplina* é de 1:n
 - cada disciplina é da responsabilidade de um único professor mas um professor pode ser responsável por mais do que uma disciplina
- 📄 O tipo de associação *Inscrito* entre *Aluno* e *Disciplina* é de n:n
 - um aluno pode estar inscrito em mais do que uma disciplina e uma disciplina pode ter vários alunos inscritos.

Multiplicidade (mais detalhe)

- ☰ Geralmente, é vantajoso definir a multiplicidade de um tipo de associação de uma forma mais detalhada, indicando dois números: o mínimo e o máximo de vezes que uma entidade pode participar em associações de um mesmo tipo.
- ☰ A multiplicidade é frequentemente 0,1 ou 1,1 ou 0,n ou 1,n, mas pode ser diferente (2,2 ou 2,n, por exemplo).

Exemplos

- ☰ O tipo de associação **Inscrito** tem multiplicidade 1,n relativamente ao tipo de entidade **Aluno**
 - nesta base de dados, um aluno tem que estar inscrito em pelo menos uma disciplina
- ☰ O tipo de associação **Inscrito** tem multiplicidade 0,n relativamente ao tipo de entidade **Disciplina**
 - nesta base de dados, podem existir entidades de tipo **Disciplina** que não estejam associadas a nenhuma entidade do tipo **Aluno** e outras que estejam associadas a uma ou mais entidades do tipo **Aluno**
- ☰ A multiplicidade do tipo de associação **Responsabilidade** é 1,1 relativamente ao tipo de entidade **Professor**
 - para cada disciplina existe sempre um e só um professor responsável

Restrições de integridade

- ☰ São condições que têm que ser verificadas para que a base de dados seja fiel à realidade.
- ☰ A multiplicidade dos tipos de associação representam restrições de integridade do modelo de dados.
- ☰ Nem todas as restrições de integridade dos dados podem ser representadas no modelo pela multiplicidade dos tipos de associação
 - por exemplo: a data de inscrição numa disciplina tem que ser posterior à data de nascimento do aluno.

1ª etapa da criação de um modelo conceptual de dados

 Identificação dos tipos de entidade e respectivos atributos.

Regras gerais:

- As entidades representam objectos que podem possuir uma descrição; os atributos não possuem informação descritiva.
- Quando a uma entidade pode corresponder mais do que um valor de um dado atributo, este deve ser considerado como uma entidade mesmo que sobre ele não exista mais nenhuma informação descritiva.
- Os atributos devem ser usados nas entidades a que mais directamente dizem respeito.
- Como chave primária deve-se evitar a utilização de conjuntos de atributos.

2ª etapa da criação de um modelo conceptual de dados

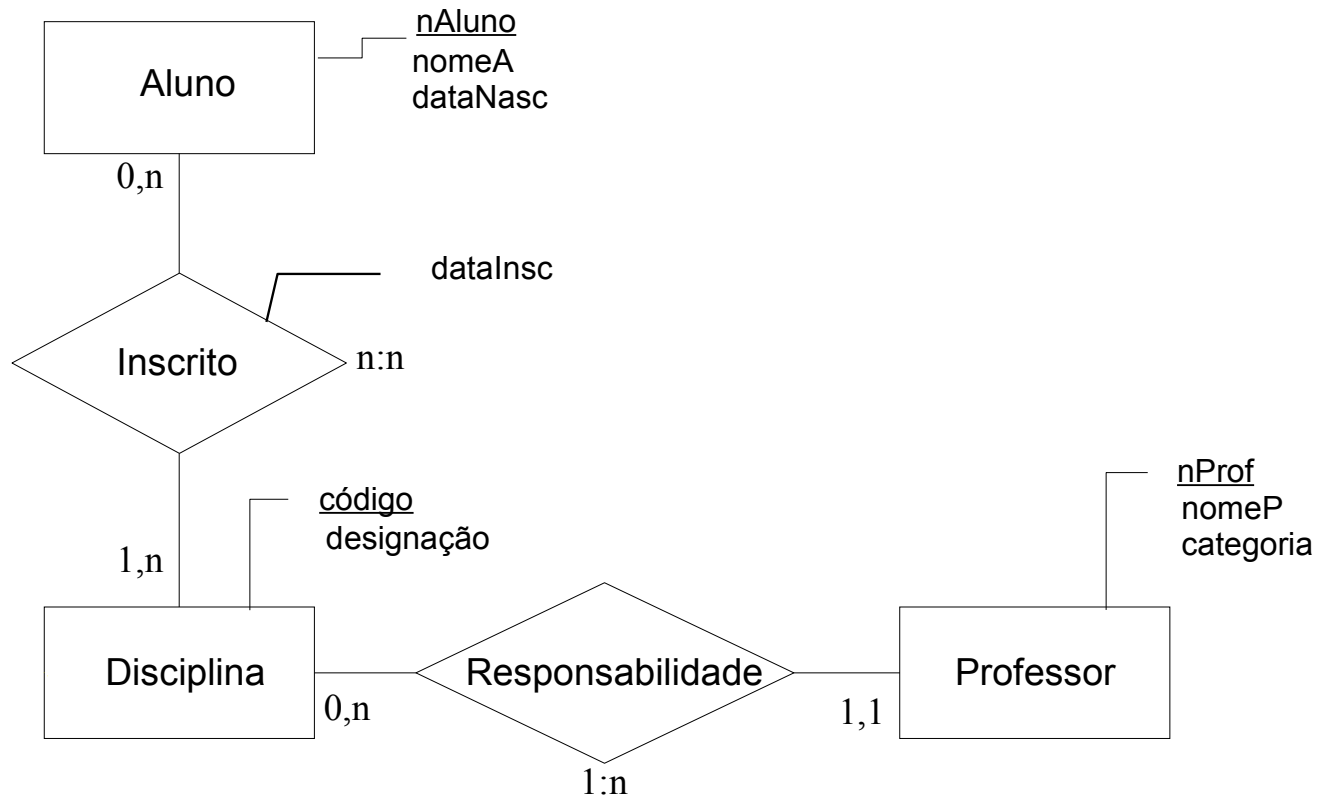
☰ Identificação dos tipos de associação e respectivos atributos.

- Por exclusão de partes, deverão representar os objectos que não foram considerados nem entidades nem atributos.

Regras gerais:

- As associações redundantes devem ser eliminadas. Consideram-se redundantes as associações que representam um mesmo aspecto da realidade; as redundâncias mais frequentes resultam da existência de associações transitivas
- Só devem ser usadas associações ternárias quando for impossível representá-las por meio de várias associações binárias.

Diagrama do modelo conceptual



Exercício 1:

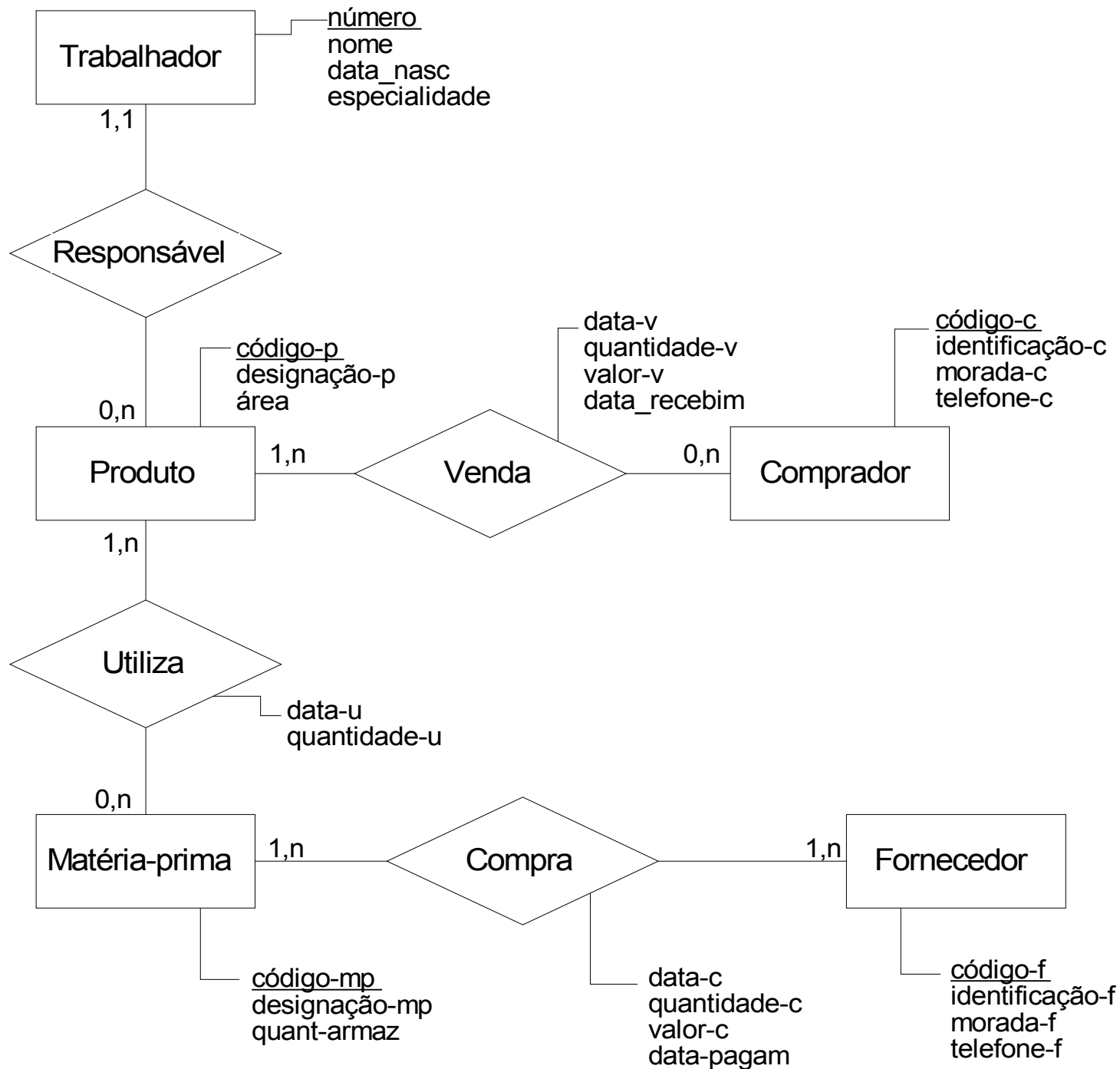
- a) Uma loja pretende construir uma base de dados com informação relevante sobre os discos que vende. Relativamente a cada disco, pretende registar na base de dados um código identificativo, o título, nº de cópias vendidas, data de gravação e o grupo/intérprete. Pretende também registar as músicas que cada disco inclui. Sobre cada música, pretende registar o título que identifica a música e o primeiro autor. Uma mesma música pode figurar em discos distintos.

Exercício 1 (cont.):

b) Uma empresa agrícola pretende construir uma base de dados com informação relevante sobre os animais que possui. Relativamente a cada variedade de animal, pretende registar na base de dados um nome identificativo e o número de animais dessa variedade que possui. Pretende também registar os produtos derivados de cada variedade de animal, em particular a sua designação, a quantidade produzida e o seu preço de venda. Cada um destes produtos é produzido apenas por uma variedade de animal. Cada variedade de animal consome vários alimentos e um dado alimento pode ser incluído na alimentação de diversas variedades de animal. É necessário registar os alimentos utilizados, sendo cada alimento caracterizado por um nome identificativo e pelo seu preço de aquisição. A base de dados deve ainda conter informação relativa à quantidade de cada alimento consumida por cada variedade de animal.

Exercício 1 (cont.):

c) Pretende-se criar uma base de dados, para efectuar a gestão de uma empresa agrícola, onde seja registada informação relativa aos seus trabalhadores e à produção e venda. Sobre cada trabalhador pretende-se registar o seu nome, data de nascimento e especialidade. Sobre cada produto da exploração agrícola pretende-se conhecer a sua designação, área cultivada e qual o trabalhador responsável. Em relação aos compradores e aos fornecedores, pretende-se conhecer a sua identificação, morada e telefone. Para cada matéria-prima pretende-se registar a sua designação e quantidade armazenada. Para cada produto vendido deverá ser registado o comprador, a data da venda, a quantidade vendida, o valor da venda e a data do recebimento. Para a matéria-prima consumida pretende-se registar os produtos a que esta se destina, a data da utilização e a quantidade utilizada. Para cada aquisição de matéria-prima deve-se conhecer o fornecedor, a data da compra, a quantidade comprada, o valor da compra e a data de pagamento.



SGBD relacional

- ☰ Baseiam-se num conjunto de conceitos teóricos apresentados em 1970 por **E. F. Codd**.
 - os conceitos que Codd introduziu foram desenvolvidos por outros autores
 - actualmente são os mais utilizados
- ☰ Vantagens dos SGBD relacionais:
 - **simplicidade** dos conceitos que utilizam
 - existência de **definições formais** para os conceitos
 - permitem uma rápida divulgação
 - permitem a adesão de diversos fabricantes de software;
 - **adequação** à representação de muitos dos aspectos que constituem a realidade

Relação

- ☰ Nas bases de dados relacionais a estrutura fundamental é a **relação** ou **tabela**.
- ☰ Uma relação é definida por um **esquema**.
- ☰ Um esquema é composto:
 - pelo **nome da relação**
 - pelos nomes de cada um dos **atributos**
 - pelo **domínio** de cada atributo
 - a definição de cada atributo depende do tipo de dados (inteiro, real, alfanumérico, data, lógico ...) que o atributo irá armazenar.

Exemplos de esquemas


Aluno(nAluno, nomeA, datanasc)

Professor(nProf, nomeP, categoria)

Atributos

- ☞ Um atributo A_i toma valores num conjunto D_i chamado **domínio** do atributo.
 - O domínio determina o tipo de valores que o atributo pode tomar.
- ☞ Dado $U = \{A_1, A_2, \dots, A_n\}$, uma relação R sobre U é um **subconjunto** de $D_1 \times D_2 \times \dots \times D_n$.
- ☞ A cada tuplo deste produto cartesiano dá-se o nome de **instância** da relação R .
 - as linhas da relação R são os **registos**
 - as colunas são os atributos (ou **campos**)

Tabela

 O conjunto das instâncias da relação R constitui uma **tabela** em que

- as linhas são as instâncias (ou **registos**)
- as colunas são os atributos (ou **campos**)

Exemplo de uma tabela

Atributos ou campos

↓

←				→
	naluno	nomeA	datanasc	
	11111	Ana Maria Santos	1978-12-03	←
	12345	José Silva	1980-11-30	←
	22222	Natália Costa	1985-12-25	←

← Instâncias
ou
registos

- Domínio de naluno : integer (N)
- Domínio de nome : string 50 (A)
- Domínio de datanasc : date (D)
- Cada registo (instância) é um tuplo

$(12345, \text{"José Silva"}, 1980-11-03) \in N \times A \times D$


Observações

- ☞ Numa relação todas as instâncias têm o mesmo número de atributos;
 - por exemplo,
Aluno (nAluno, nomeA, disciplina1, ..., disciplinaj, ..., dataNasc)
não é uma relação;
- ☞ Numa instância o valor de um atributo é sempre **atômico**;
 - isto é, numa tabela, no cruzamento de uma linha com uma coluna só pode existir um valor de atributo;
- ☞ numa relação **não** podem existir instâncias iguais;
- ☞ a **ordem** porque se encontram as instâncias de uma relação e os seus atributos é **irrelevante**;
- ☞ os valores de cada atributo pertencem a um mesmo **domínio**;
- ☞ podem existir instâncias sem valores em alguns dos seus atributos; neste caso o atributo diz-se opcional e o seu valor é **null**;
- ☞ os nomes (ou **identificadores**) dos atributos que constituem o esquema de uma relação são únicos nessa relação.


Chave(s) de uma relação

- Um atributo que toma valores diferentes para cada instância da relação é uma **chave candidata** da relação.
 - cada instância pode ser identificada pelo valor da chave: o valor da chave nunca se repete.
- Por vezes, uma chave pode ser composta por mais do que um atributo. E, no caso extremo, uma chave pode até ser composta por todos os atributos que definem a relação.

Chave primária

 **Chave primária** de uma relação é um subconjunto mínimo de atributos cujos valores permitam identificar de modo único cada uma das instâncias dessa relação.

 *Atributo primário*: pertence à chave primária.

 *Atributo não primário*: não pertence à chave primária.

Chave estrangeira

☰ Nos SGBD relacionais, para representar as associações existentes entre as várias entidades utilizam-se esquemas de relações em que figuram **atributos comuns**.

☰ Uma **chave estrangeira** de uma relação é um conjunto de atributos que é chave primária de outra relação.

Restrições de integridade

☰ **Condições** que devem ser verificadas para que um elemento do produto cartesiano dos domínios dos atributos possa pertencer à relação.

☰ Consequências imediatas das definições dadas:

- **integridade de domínio**
 - impostas pelo domínio de cada atributo;
- **integridade de entidade**
 - não existem duas instâncias de uma relação cujas chaves tenham o mesmo valor
 - valor de uma chave primária (ou mesmo de uma parte dela) não pode ser *null*;
- **integridade de referência**
 - o valor de qualquer chave estrangeira de uma relação é um valor da chave primária da relação que refere

Outras restrições de integridade

Outra categoria de restrições:

- condições que devem ser verificadas pelos valores de dois ou mais atributos de uma mesma instância da relação.
 - Algumas restrições requerem a utilização de fórmulas matemáticas ou lógicas.

Exemplo:

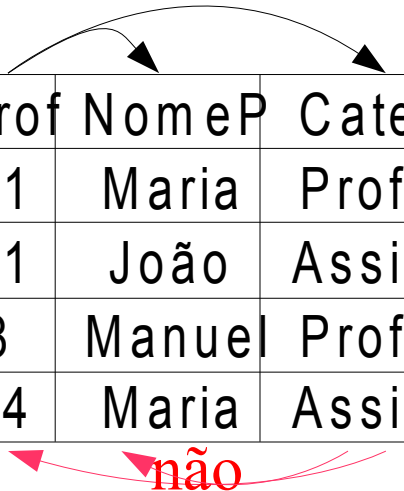
- a data de nascimento de um aluno tem de ser posterior à data de inscrição nas várias disciplinas.

Dependências funcionais

☰ Dada um relação R definida sobre um conjunto de atributos $U = \{A_1, A_2, \dots, A_n\}$, diz-se que o atributo A_j **depende funcionalmente do atributo A_i** ($A_i \rightarrow A_j$) quando é impossível terem-se duas instâncias de R com o mesmo valor para A_i e diferentes valores para A_j .

Exemplo:

Nprof	NomeP	Categoria
101	Maria	Prof. Aux.
321	João	Assistente
98	Manuel	Prof. Aux.
294	Maria	Assistente



Esquema de uma base de dados

- ☞ O conjunto das relações que a compõem, definidas por meio dos seus atributos, e das restrições de integridade.
- ☞ Para cada base de dados pode existir um número quase ilimitado de esquemas.
- ☞ Um esquema é tanto melhor quanto mais fácil for o seu manuseamento, nomeadamente no que se refere à manutenção da sua coerência após as actualizações que venha a sofrer.
- ☞ Um esquema é tanto melhor quanto menor for o número de dependências funcionais nele existente.

Normalização

- 📄 O objectivo das regras de normalização é definir regras para o agrupamento de atributos de forma a minimizar o número de dependências funcionais existentes numa base de dados.
- 📄 Resumidamente, pode dizer-se que numa relação que verifica as primeiras três formas normais, qualquer atributo que não pertence à chave depende completamente e exclusivamente da totalidade da chave.

Primeira forma normal

Uma relação R encontra-se na primeira forma normal se e só se todos os seus atributos contêm valores atômicos.

Por exemplo, o esquema
Aluno (nAluno, nomeA, disciplina1, ..., disciplinaj, ..., dataNasc)
não respeita a 1ª forma normal (nem é sequer uma relação) porque possui um número variável de atributos.

Segunda forma normal

Uma relação R encontra-se na segunda forma normal se e só se:

- verifica a primeira forma normal e
- todos os atributos não primários são funcionalmente dependentes da **totalidade** da chave primária de R.
 - Esta normalização visa diminuir a duplicação de dados.

Exemplo:

Inscrito (nAluno, código, nomeA, dataNasc)

não se encontra na 2ª forma normal porque ...

Aplicação da 2ª forma normal a esta relação:

Inscrito (nAluno, código)

Aluno (nAluno, nomeA, dataNasc)

Terceira forma normal

Uma relação R encontra-se na terceira forma normal se e só se:

- verifica a segunda forma normal e
- não contém dependências funcionais entre atributos não primários.
 - Um atributo que não pertence à chave primária da relação não pode ser funcionalmente dependente de outro atributo que também não pertence à chave da relação.

Exemplo:

Disciplina (código, designação, nProf, nomeP, categoria)

não se encontra na 3ª forma normal, mas

Disciplina (código, designação, nProf)

Professor (nProf, nomeP, categoria)

já estão na terceira forma normal.

Passagem do modelo E-A para relacional

- Um modelo de tipo E-A descreve bem e de uma forma natural e simples a realidade.
- Os modelos deste tipo são suficientemente flexíveis para poderem ser utilizados eficazmente numa fase em que a estruturação dos dados é ainda confusa.
- A estas vantagens pode ainda adicionar-se a facilidade com que é possível efectuar a sua passagem para um SGBD relacional, bem formalizado e comercializado por diversos fornecedores de *software*.

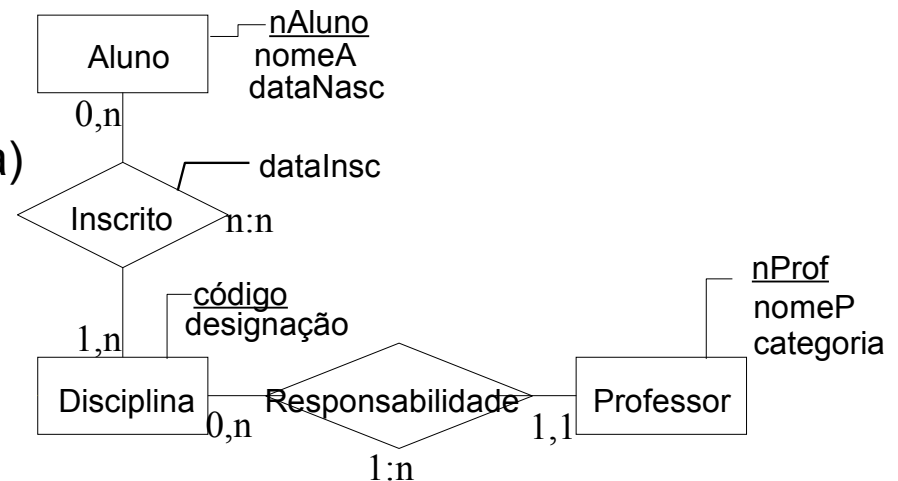
Regras para transformação de um modelo E-A num esquema relacional (1)

☰ Cada tipo de entidades do modelo E-A traduz-se por uma relação em que a chave primária e os atributos provêm do tipo de entidade.

Aluno (nAluno, nomeA, dataNasc)

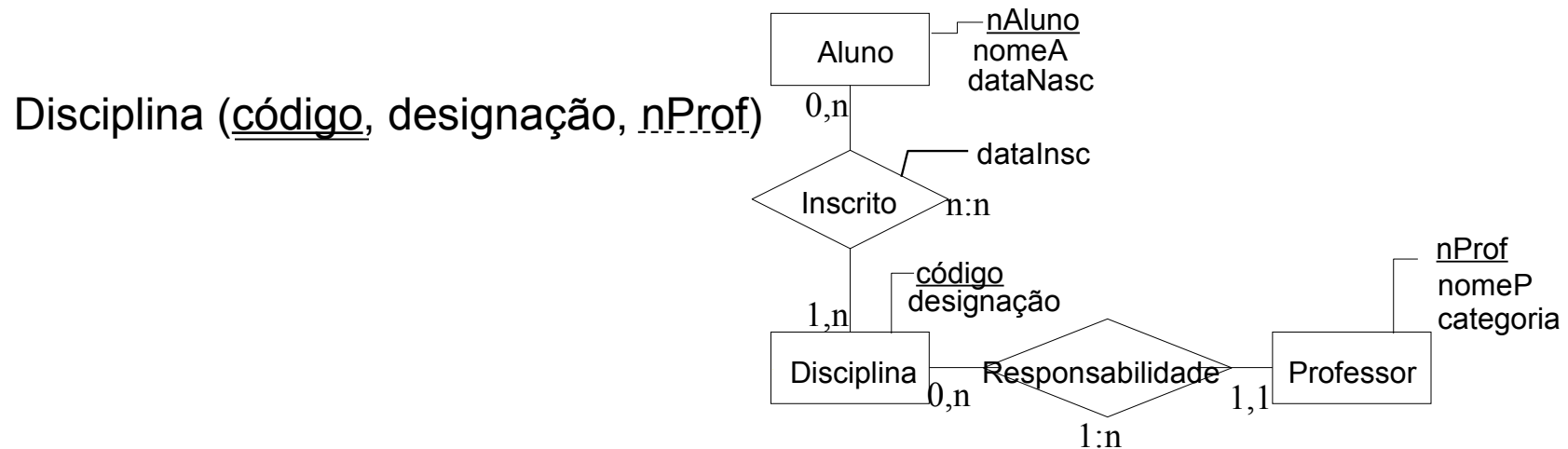
Disciplina (código, designação)

Professor (nProf, nomeP, categoria)



Regras para transformação de um modelo E-A num esquema relacional (2)

Um tipo de associação de 1:n (um para muitos) entre dois tipos de entidades E_i e E_j que tenha uma multiplicidade igual a 0,1 ou 1,1 para um tipo de entidade E_i traduz-se por uma chave estrangeira na relação R que é tradução de E_j .



Regras para transformação de um modelo E-A num esquema relacional (2')

- Um tipo de associação de 1:1 (um para um) entre dois tipos de entidades E_i e E_j é tratado como um caso especial do tipo de associação 1:n
 - traduz-se por uma chave estrangeira na relação que é tradução de E_j ou de E_i ;
 - se apenas para um destes tipos entidades a multiplicidade for 1,1 (sendo para a outra 0,1) dá-se preferência à relação que traduz este tipo de entidade,
 - no caso contrário é indiferente qual a relação que é escolhida.

Regras para transformação de um modelo E-A num esquema relacional (3)

Um tipo de associação de n:n (muitos para muitos) traduz-se por uma relação em que a chave primária inclui as chaves estrangeiras que são chave primária dos tipos de entidade que a constituem; os outros atributos são a tradução dos tipos de atributos da associação (se esta possuir algum).

- No caso das chaves estrangeiras não serem suficientes para formar a chave primária da relação, na constituição desta devem também ser utilizados outros atributos de forma a ser obtida uma chave primária adequada à relação.

Regras para transformação de um modelo E-A num esquema relacional (3) (exemplo)

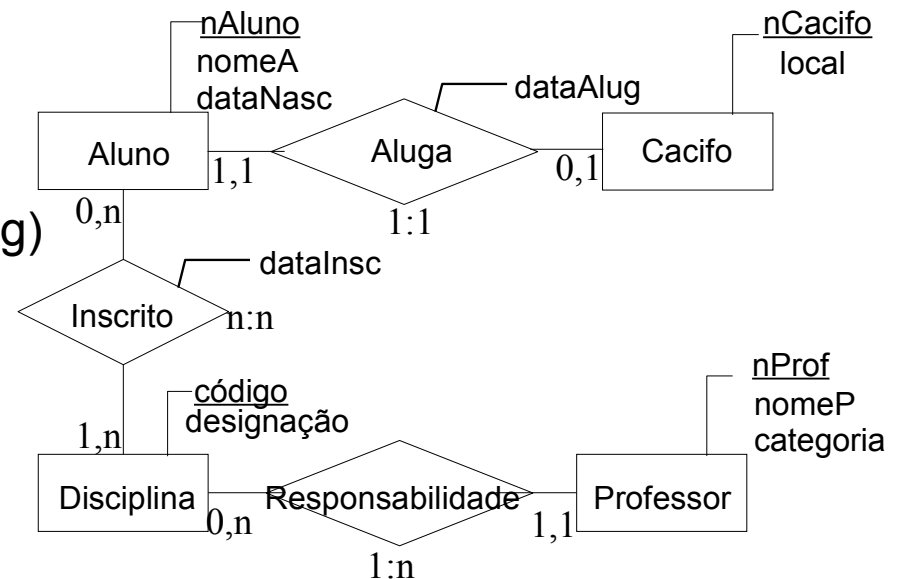
Aluno (nAluno, nomeA, dataNasc)

Disciplina (código, designação, nProf)

Professor (nProf, nomeP, categoria)

Inscrito (nAluno, código, dataInsc)

Cacifo (nCacifo, local, nAluno, dataAlug)



Na notação utilizada

- um duplo sublinhado indica que o conjunto de atributos é a chave primária da relação
- um sublinhado simples tracejado indica uma chave estrangeira.

Tradução do Modelo E-A no Modelo Relacional

Estas regras permitem que se proceda de uma forma rápida (e quase automática) à passagem de um modelo Entidade-Associação em que só existam associações binárias para um esquema relacional.

Exercício 2:

A partir dos modelos entidade – associação estabelecidos no exercício anterior, crie para cada um deles, um **esquema relacional** que respeite a terceira forma normal.

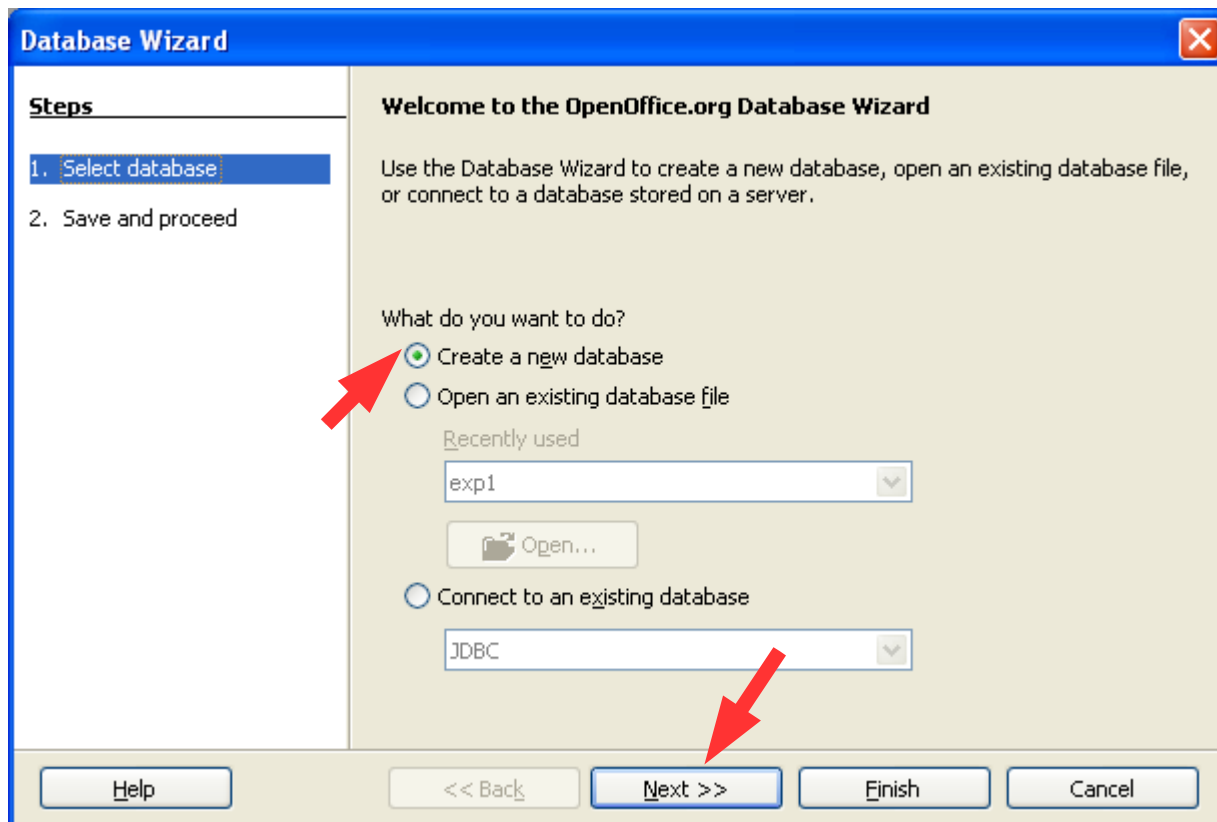
OpenOffice.org Base

- ☞ O Open Office.org é um conjunto de aplicações multiplataforma (Windows, Linux, Sun Solaris, Mac OS X, FreeBSD), *open source* e de distribuição gratuita (www.openoffice.org).
- ☞ O OpenOffice.org Base é um sistema de gestão de bases de dados relacionais que utiliza como motor da base de dados o HSQLDB (www.hsqldb.org) e que suporta a linguagem SQL.
- ☞ O **SQL**, Structured Query Language, é uma linguagem normalizada (ANSI) usada para manipular a informação de uma base de dados relacional.

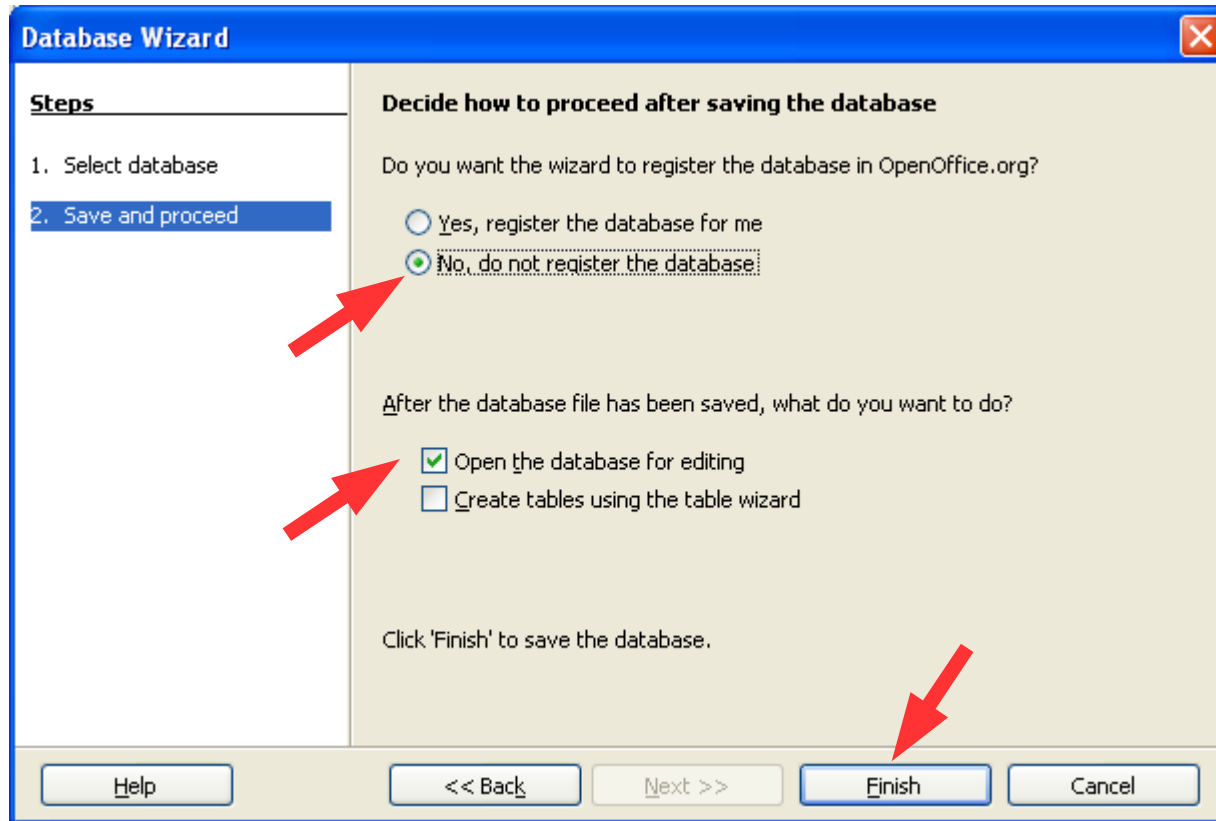
Criação de uma base de dados em Base

📄 Activar a aplicação:

Start → Programs → OpenOffice.org 3.0 →
OpenOffice.org Base



Criação de uma base de dados (cont.)



Exercício: Crie uma base de dados designada PrimeiraBD, salvando-a na pasta de trabalho BDados, numa pasta pessoal criada em D:\home.

Criação de uma tabela

The image shows two overlapping windows from the OpenOffice.org Base application. The background window is titled 'PrimeiraBD - OpenOffice.org Base' and displays the 'Tasks' pane with options like 'Create Table in Design View...', 'Use Wizard to Create Table...', and 'Create View...'. The foreground window is titled 'PrimeiraBD.odb : Trabalhador - OpenOffice.org Base: Table Design' and shows a table design grid with the following fields:

Field Name	Field Type	Description
número	Integer [INTEGER]	
nome	Text [VARCHAR]	
dataNasc	Date [DATE]	
especialidade	Text [VARCHAR]	

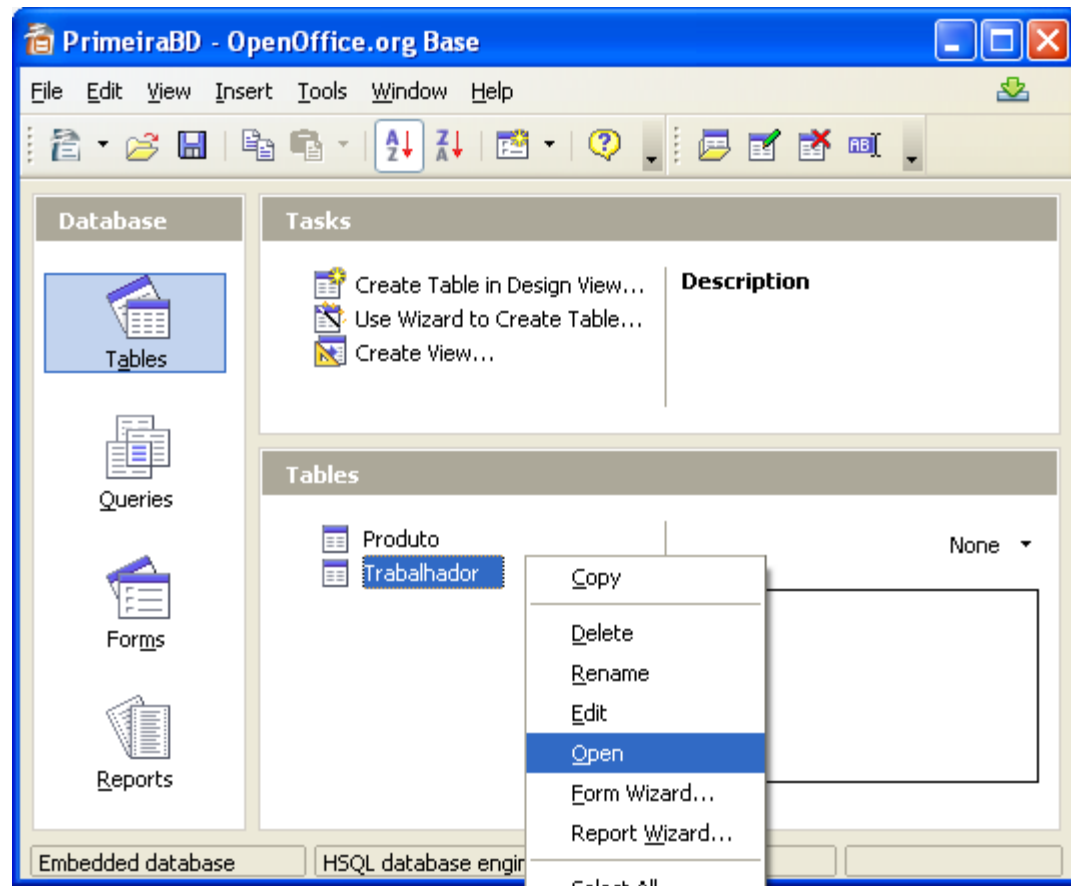
Below the table design grid is the 'Field Properties' section, which includes the following settings:

- AutoValue: Yes
- Auto-increment statement: (empty text box)
- Length: 0
- Format example: 0

Introdução de dados numa tabela

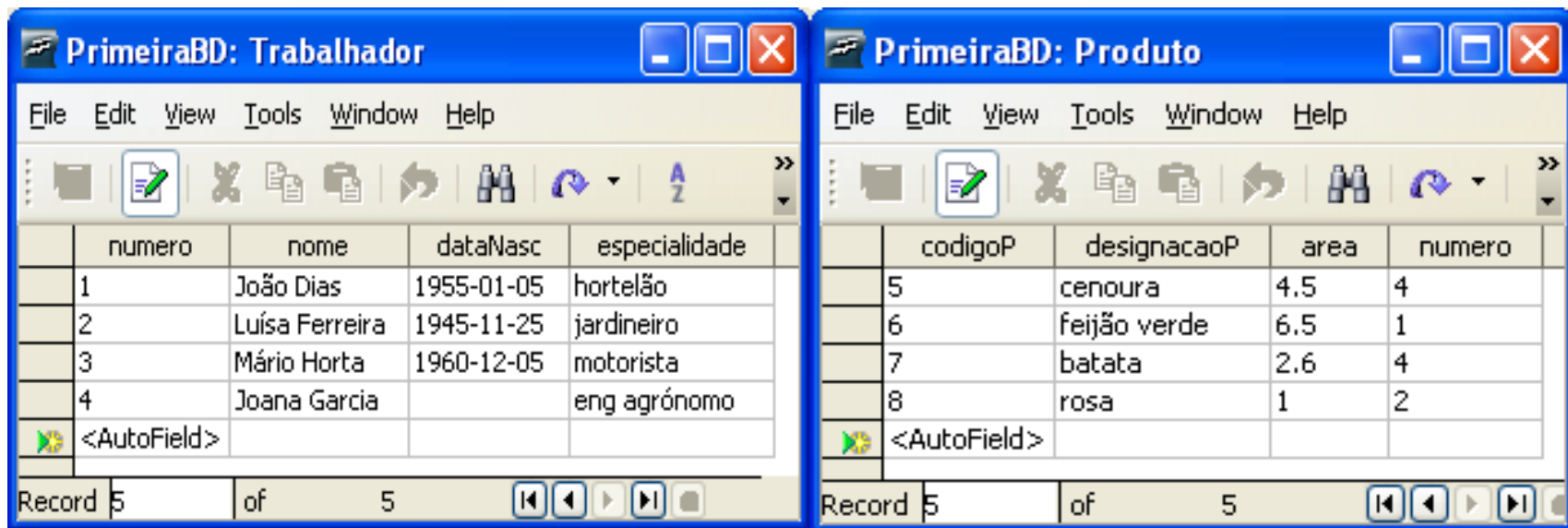
Para adicionar / alterar registos numa tabela é necessário abrir a tabela:

- duplo clique no nome da tabela, ou
- após clicar com o botão direito do rato, seleccionar Open, ou
- menu Edit → Open Database Object...



Exercício:

📄 Crie as tabelas Trabalhador e Produto, da base de dados do exercício 3, com os seguintes registos:

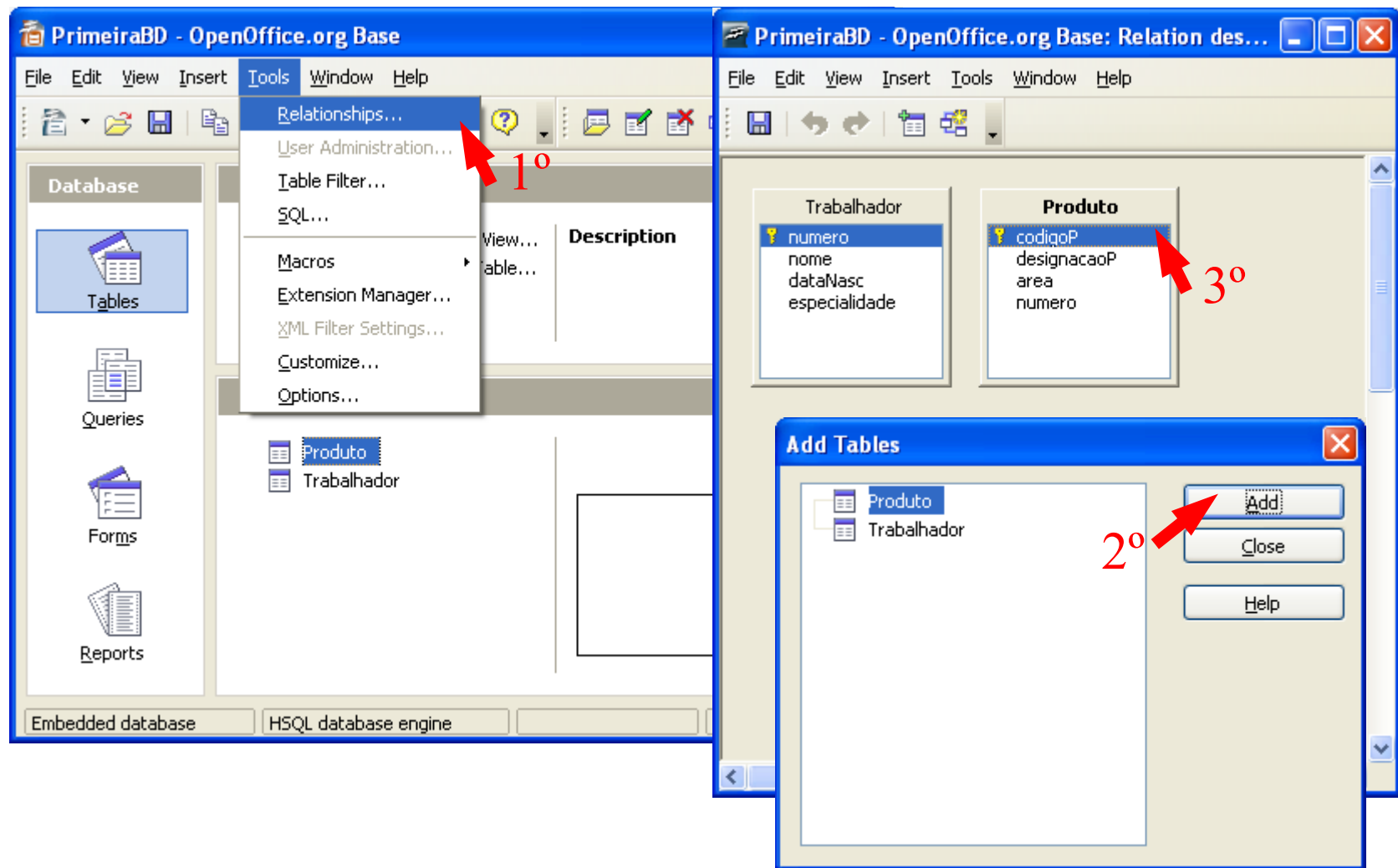


The image shows two side-by-side windows from a database application. The left window is titled 'PrimeiraBD: Trabalhador' and displays a table with 5 columns: 'numero', 'nome', 'dataNasc', and 'especialidade'. The right window is titled 'PrimeiraBD: Produto' and displays a table with 5 columns: 'codigoP', 'designacaoP', 'area', and 'numero'. Both windows show a status bar at the bottom indicating 'Record 5 of 5'.

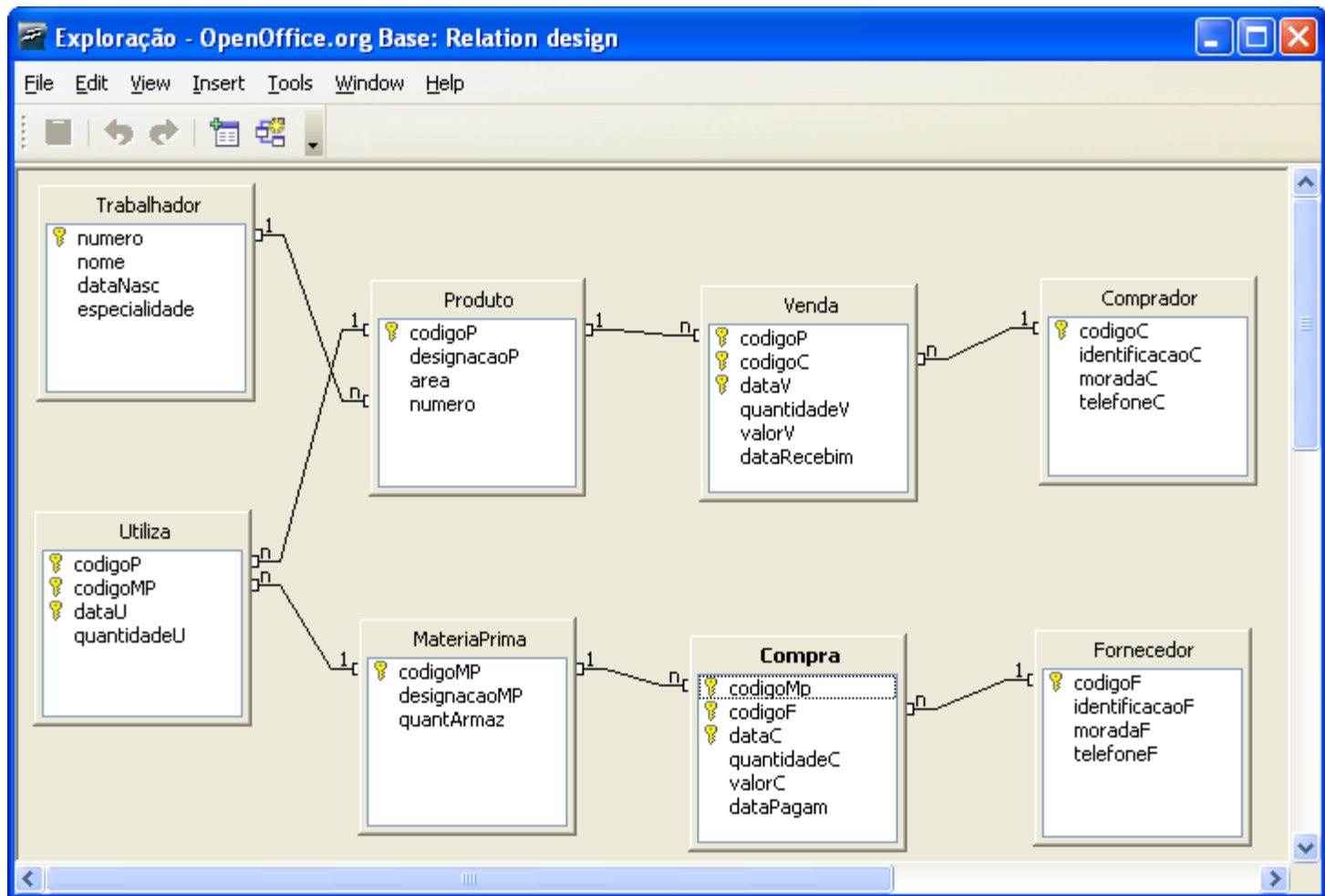
numero	nome	dataNasc	especialidade
1	João Dias	1955-01-05	hortelão
2	Luísa Ferreira	1945-11-25	jardineiro
3	Mário Horta	1960-12-05	motorista
4	Joana Garcia		eng agrónomo
<AutoField>			

codigoP	designacaoP	area	numero
5	cenoura	4.5	4
6	feijão verde	6.5	1
7	batata	2.6	4
8	rosa	1	2
<AutoField>			

Implementação de restrições de integridade de referência

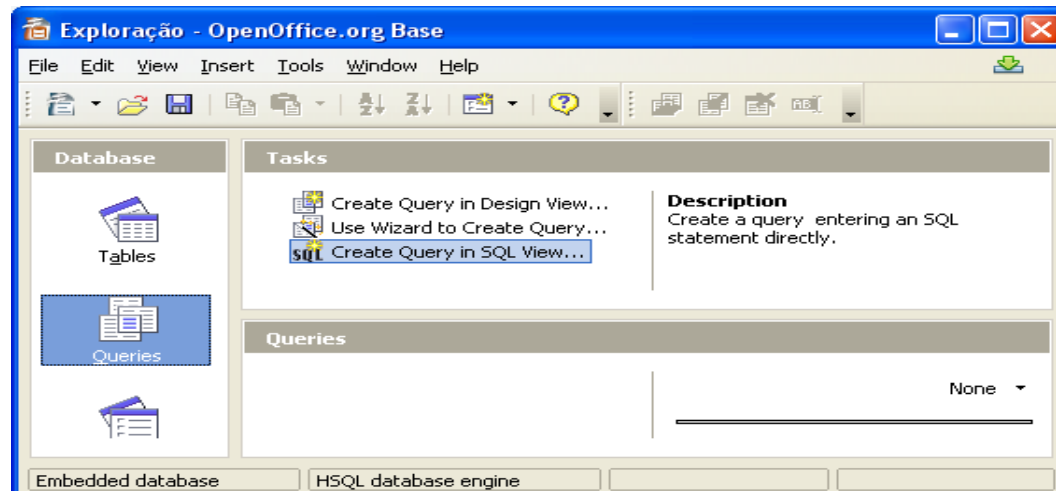


Restrições de integridade de referência: exemplo da base de dados Exploração



Consultas a uma base de dados

- As operações de consulta (e actualização) de uma base de dados denominam-se *queries*.
- As regras do modelo relacional levaram ao desenvolvimento de uma linguagem, o SQL, que permite efectuar eficientemente *queries*.
- Iniciar a criação de uma *query* em SQL:



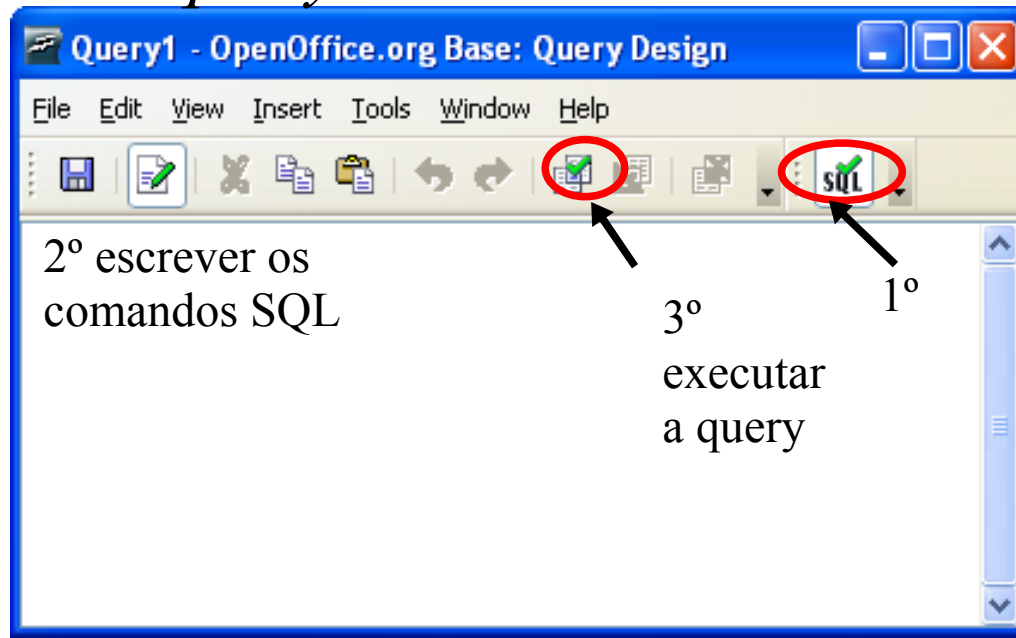
Execução de uma *query* em SQL

1º Seleccionar o icon **SQL** (deste modo os comandos são directamente executados no motor da base de dados);

2º Escrever os comandos SQL

– nas versões mais recentes do OpenOffice.org Base é necessário colocar entre aspas os nomes das tabelas e dos campos;

3º Executar a *query*.



A instrução SELECT

- As consultas a uma base de dados relacional fazem-se em SQL recorrendo à instrução SELECT. Esta instrução permite criar conjuntos de registos de uma ou mais tabelas da base de dados seleccionados segundo diversos critérios.
- A sintaxe completa da instrução SELECT é complexa.

A forma mais simples da instrução

SELECT

utiliza apenas a cláusula FROM

selecciona todos os registos de uma tabela

Sintaxe — 1ª variante:

```
SELECT { * | table.* | [table.]field1 [, [table.]field2 [, ...]] }  
FROM table;
```

onde

* especifica que todos os campos devem ser seleccionados

table especifica o nome da tabela que contém os campos e os registos seleccionados

field especifica os nomes dos campos que são seleccionados

Exemplo

☰ Considere a base de dados Exploração¹ com o seguinte esquema relacional:

Trabalhador (número, nome, dataNasc, especialidade)

Produto (códigoP, designaçãoP, área, numero)

Comprador (códigoC, identificaçãoC, moradaC, telefoneC)

Venda (códigoP, códigoC, dataV, quantidadeV, valorV, dataRecebi)

MateriaPrima (códigoMP, designaçãoMP, quantArmaz)

Utiliza (códigoP, códigoMp, dataU, quantidadeU)

Fornecedor (códigoF, identificaçãoF, moradaF, telefoneF)

Compra (códigoMp, códigoF, dataC, quantidadeC, valorC, dataPagam)

¹ \\prunus\home\cadeiras\MatInf\BDados\Exploração.odt

Exemplo (cont.)

Tabela Trabalhador

número	nome	dataNasc	especialidade
1	João Dias	1955-01-05	hortelão
2	Luísa Ferreira	1945-11-25	jardineiro
3	Mário Horta	1960-12-05	motorista
4	Joana Garcia		eng agrónomo

Tabela Produto

códigoP	designaçãoP	area	número
5	cenoura	4.5	4
6	feijão verde	6.5	1
7	batata	2.6	4
8	rosa	1	2

Tabela Comprador

códigoC	identificaçãoC	moradaC	telefoneC
1	Manel Maria	Casais de Cima	749658365
2	Luisa Fraga	Casais de Baixo	589016587
3	Duarte Silva	Vila Nova	456123789

Tabela Venda

códigoP	códigoC	dataV	quantidadeV	valorV	dataRecebib
5	1	1999-04-07	50	45000	1999-05-07
5	2	1999-02-20	5.6	1000	
6	2	1999-04-05	100	60000	
6	1	1999-03-29	450	90000	

Tabela Fornecedor

códigoF	identificaçãoF	moradaF	telefoneF
1	Ana Sousa	ISA - Tapada da Ajuda	213638161

A forma mais simples da instrução SELECT (exemplo)


📄 Seleccionar todos os produtos e os valores de todos os seus atributos.

```
SELECT *  
FROM "Produto";
```

📄 Conjunto de registos seleccionados:

códigoP	designaçãoP	área	número
5	cenoura	4.5	4
6	feijão verde	6.5	1
7	batata	2.6	4
8	rosa	1	2

A cláusula WHERE

 permite especificar uma condição que os registos seleccionados verificam.

Sintaxe — 2ª variante:

```
SELECT fieldlist  
FROM table  
WHERE condition;
```

onde

fieldlist ...

table ...

condition uma condição que os registos seleccionados verificam; podem ser utilizados operadores relacionais (<, <=, >, >=, =, <>), operadores lógicos (NOT, AND, OR) e os operadores IN, IS BETWEEN e LIKE.

A cláusula WHERE (exemplo)

- Seleccionar os códigos dos produtos vendidos desde 1999-04-01, os códigos dos compradores que os compraram, as datas destas vendas e os respectivos valores.

```
SELECT "códigoP", "códigoC", "dataV", "valorV"  
FROM "Venda"  
WHERE "dataV" > {D '1999-04-01'};
```

- Conjunto de registos seleccionados:

códigoP	códigoC	dataV	valorV
5	1	07-04-1999	45000
6	2	05-04-1999	60000

A cláusula FROM

- ☞ A cláusula FROM especifica o(s) nome(s) da(s) tabela(s) em que se encontram os registos a seleccionar.
- ☞ Quando é indicada mais do que uma tabela, a instrução SELECT produz um conjunto de registos cujos campos são os indicados na lista de campos.
- ☞ Os registos seleccionados correspondem aos tuplos do produto cartesiano das tabelas especificadas, isto é, cada registo é composto por um registo de cada uma dessas tabelas.

A cláusula FROM

Sintaxe — 3ª variante:

```
SELECT fieldlist  
FROM tableexpression  
[ WHERE condition ] ;
```

onde

tableexpression especifica o(s) nome(s) da(s) tabela(s) que contém os campos e os registos seleccionados.

Exemplo:

```
SELECT “designaçãoP”, “área”, “Produto”.”número”,  
          “Trabalhador”.”número”, “nome”, “especialidade”  
FROM “Produto”, “Trabalhador”;
```

A cláusula FROM (exemplo)

designaçãoP	área	Produto.número	Trabalhador.número	nome	especialidade
cenoura	4.5	4	1	João Dias	hortelão
feijão verde	6.5	1	1	João Dias	hortelão
batata	2.6	4	1	João Dias	hortelão
rosa	1	2	1	João Dias	hortelão
cenoura	4.5	4	2	Luísa Ferreira	jardineiro
feijão verde	6.5	1	2	Luísa Ferreira	jardineiro
batata	2.6	4	2	Luísa Ferreira	jardineiro
rosa	1	2	2	Luísa Ferreira	jardineiro
cenoura	4.5	4	3	Mário Horta	motorista
feijão verde	6.5	1	3	Mário Horta	motorista
batata	2.6	4	3	Mário Horta	motorista
rosa	1	2	3	Mário Horta	motorista
cenoura	4.5	4	4	Joana Garcia	eng agrónomo
feijão verde	6.5	1	4	Joana Garcia	eng agrónomo
batata	2.6	4	4	Joana Garcia	eng agrónomo
rosa	1	2	4	Joana Garcia	eng agrónomo

A cláusula FROM e a cláusula WHERE

Quando a cláusula FROM refere mais do que uma tabela, a cláusula WHERE é frequentemente utilizada para seleccionar no resultado do produto cartesiano os tuplos que correspondem a registos em que o valor de uma chave estrangeira é igual ao **valor** de uma chave primária.

Neste caso, a instrução SELECT produz **um subconjunto do produto cartesiano** das tabelas referidas na cláusula FROM e é usual dizer-se que se efectua um *join* ou cruzamento de tabelas.

A cláusula FROM e a cláusula WHERE

(exemplos)

☞ Para cada produto seleccionar a designação, a área e o número, nome e especialidade do respectivo responsável.

```
SELECT "designaçãoP", "área", "Trabalhador"."número", "nome",  
       "especialidade"
```

```
FROM "Produto", "Trabalhador"
```

```
WHERE "Produto"."número"="Trabalhador"."número";
```

☞ Conjunto de registos seleccionados:

designaçãoP	área	número	nome	especialidade
feijão verde	6.5	1	João Dias	hortelão
rosa	1	2	Luísa Ferreira	jardineiro
cenoura	4.5	4	Joana Garcia	eng agrónomo
batata	2.6	4	Joana Garcia	eng agrónomo

Eliminação de registos duplicados

☰ Algumas queries podem conduzir a conjuntos de registos com duplicações. Os registos duplicados podem ser eliminados recorrendo à utilização de DISTINCT.

Sintaxe — 4^a variante:

```
SELECT DISTINCT fieldlist  
FROM tableexpression  
WHERE condition;
```

Eliminação de registos duplicados (exemplo)

- Seleccionar o número, nome e especialidade dos trabalhadores que são responsáveis por algum produto.

```
SELECT DISTINCT "Trabalhador"."número", "nome",  
                "especialidade"  
FROM "Produto", "Trabalhador"  
WHERE "Produto"."número"="Trabalhador"."número";
```

- Conjunto de registos seleccionados:

número	nome	especialidade
1	João Dias	hortelão
2	Luísa Ferreira	jardineiro
4	Joana Garcia	eng agrónomo

Sub-query numa cláusula WHERE

- ☰ Na cláusula WHERE pode ainda figurar uma *query*, neste caso denominada *sub-query*, constituída por outra instrução SELECT.
- ☰ O resultado da *sub-query* define um conjunto de valores que são utilizados na condição especificada pela cláusula WHERE.

Sub-query numa cláusula WHERE

(exemplo)


- Seleccionar as designações dos produtos dos quais já se tenham efectuado vendas, e o nome e a especialidade dos respectivos reponsáveis.

```
SELECT "nome", "especialidade", "designaçãoP"  
FROM "Trabalhador", "Produto"  
WHERE "Trabalhador"."número"="Produto"."número"  
      AND "Produto"."códigoP" IN (SELECT "códigoP"  
                                   FROM "Venda");
```

- Conjunto de registos seleccionados:

nome	especialidade	designaçãoP
Joana Garcia	eng agrónomo	cenoura
João Dias	hortelão	feijão verde

A cláusula ORDER BY

 A cláusula ORDER BY apenas ordena os registos seleccionados pelos valores de um conjunto de campos especificado.

Sintaxe — 5ª variante:

SELECT fieldlist

FROM tableexpression

WHERE condition

ORDER BY field1 [ASC | DESC][, field2 [ASC | DESC][, ...]];

onde

ASC especifica a ordem ascendente

DESC especifica a ordem descendente

Aliases

É possível definir novos nomes (*aliases*) para o nome das tabelas e/ou dos campos utilizados numa instrução SELECT.

Exemplo: Para cada produto seleccionar a designação, a área e o nome e especialidade do respectivo responsável. O resultado deve ser apresentado por ordem decrescente da área do produto.

```
SELECT "designaçãoP" AS prod, "área", "nome" AS responsável,  
"especialidade"  
FROM "Produto" AS P, "Trabalhador" AS T  
WHERE P."número"=T."número"  
ORDER BY "área" DESC;
```

Funções de agregação

Estas funções possibilitam o resumo dos resultados de uma selecção de registos, como alternativa à selecção de registos. Podem ser utilizadas as funções SUM, AVG, MAX, MIN e COUNT.

Funções de agregação (exemplo)

- Seleccionar a área total de todos os produtos, a área média por produto e o número de produtos existentes

```
SELECT SUM ("área") AS "Área total",  
        AVG ("área") AS "Área média",  
        COUNT(*) AS "Nº de produtos"  
FROM "Produto";
```

- Conjunto de registos seleccionados:

Área total	Área média	Nº de produtos
14.6	3.65	4

Cláusula GROUP BY

- ☰ A utilização de funções de agregação é feita frequentemente em conjunto com a cláusula GROUP BY. Esta especifica os conjuntos de registos seleccionados que são objecto da(s) função(ões).
- ☰ Quando é utilizada a cláusula GROUP BY, só podem ser indicados na cláusula SELECT os atributos incluídos na cláusula GROUP BY (para além daqueles que são objecto de uma função).

Cláusula GROUP BY (exemplo)


☰ Para cada produto seleccionar o respectivo código, número de vendas e a quantidade total dessas vendas.

```
SELECT "códigoP", COUNT("códigoP") AS "Nº de vendas",  
        SUM("quantidadeV") AS "Quantidade Tot"  
FROM "Venda"  
GROUP BY "codigoP";
```

☰ Conjunto de registos seleccionados:

códigoP	Nº de vendas	Quantidade Tot
5	2	55.6
6	2	550

Cláusula **GROUP BY ... HAVING**

 A cláusula **GROUP BY** pode ser utilizada em conjunto com **HAVING** para restringir os registos seleccionados áqueles que verificam a condição especificada em **HAVING**.

Cláusula GROUP BY ... HAVING (exemplo)

☞ Para cada produto com vendas não pagas numa quantidade total superior ou igual a 250, seleccionar o respectivo código, número de vendas por pagar e a quantidade total dessas vendas.

```
SELECT "códigoP", COUNT("códigoP") AS "Nº de vendas",  
       SUM("quantidadeV") AS "Quantidade Tot"  
FROM "Venda"  
WHERE "dataRecebim" IS NULL  
GROUP BY "códigoP"  
HAVING SUM("quantidadeV")>=250;
```

☞ Conjunto de registos seleccionados:

códigoP	Nº de vendas	Quantidade Tot
6	2	550

O operador UNION

- UNION é usado para combinar o resultado de duas ou mais instruções SELECT num único conjunto de resultados.
- Cada instrução SELECT tem de ter como resultado o mesmo número de colunas.
- As colunas correspondentes nas várias instruções SELECT têm de ter tipos de dados compatíveis.
- No resultado final, o nome das colunas corresponde ao do primeiro SELECT.

SELECT ...

UNION

SELECT ...

[UNION

SELECT ...]

[ORDER BY field1 [ASC | DESC][, field2 [ASC | DESC][, ...]];

O operador UNION (exemplo)

Seleccionar os códigos e as identificações de todos os compradores e fornecedores indicando os que são clientes e os que são fornecedores.

```
SELECT “códigoC” AS código, “identificaçãoC” AS Nome,  
'Cliente' AS Tipo
```

```
FROM “Comprador”
```


```
UNION
```

```
SELECT “códigoF”, “identificaçãoF”, 'Fornecedor'
```

```
FROM “Fornecedor”;
```

código	nome	Tipo
1	Ana Sousa	Fornecedor
1	Manel Maria	Cliente
2	Luisa Fraga	Cliente
3	Duarte Silva	Cliente

Outras instruções SQL

 A linguagem SQL inclui outras instruções para definição/alteração do esquema de uma base de dados:

- CREATE TABLE
- ALTER TABLE
- INSERT
- DELETE
- UPDATE