






---









## Conteúdos

---





### **Dos ficheiros de dados aos sistemas de gestão de base de dados**

-  Ficheiros de dados, registos e campos pág. 2
-  Limitações das bases de dados baseadas numa só tabela pág. 2
-  Sistemas de Gestão de Base de Dados pág. 4

### **Modelos de Base de Dados**

-  Modelos baseados em objectos pág. 7
-  Modelos baseados em registos pág. 8
-  Modelos Entidade – Relação pág. 9
-  Entidades e Atributos pág. 9
-  Valores e domínios dos atributos pág. 10
-  Tipos de Atributos pág. 11
-  Relacionamentos entre entidades pág. 12
-  Modelo Relacional pág. 15

### **Modelação da informação e design de uma Base de Dados**

-  Objectivos a atingir num projecto de base de dados pág. 21
-  Fases de um projecto de base de dados pág. 21
-  Estratégias para a concepção de bases de dados pág. 22
-  Desenho de base de dados pág.



## Dos ficheiros de dados aos sistemas de gestão de base de dados

### Limitações das Base de Dados baseadas numa só tabela

Este género de base de dados é também denominado por **base de dados flat file** ou **monotabela**.

Um dos exemplos deste género de base de dados é a folha de cálculo (Excel, por exemplo), onde a informação é organizada em linhas e colunas.

Para comprovar a limitação deste género de base de dados tomemos o seguinte exemplo:




*Uma empresa comercializa um conjunto de artigos e pretende criar uma base de dados para registar esses artigos, bem como as encomendas efectuadas pelos seus clientes, em relação aos quais também pretende ter registados os dados habituais (nome, endereço, telefone, etc.).*

O registo destes dados numa base de dados monotabela levaria a um resultado como o apresentado a seguir:

Cliente	Endereço	TelfFax	Produto	Modelo	Preço	DataEnc	Quantid
Silva	Lisboa	665544	Alicate	A1	400	4/3/2000	5
Santos	Porto	554433	Martelo	M1	250	4/3/2000	10
Costa	Coimbra	332211	Serra	S1	1500	4/3/2000	2
Castro	Faro	443322	Tesoura	T1	500	4/3/2000	6
Silva	Lisboa	665544	Martelo	M1	250	5/3/2000	10
Silva	Lisboa	665544	Tesoura	T1	500	5/3/2000	5
Costa	Coimbra	332211	Alicate	A1	400	6/3/2000	4
Costa	Coimbra	332211	Serra	S1	1500	6/3/2000	1
Castro	Faro	443322	Alicate	A1	400	7/3/2000	10
Silva	Lisboa	665544	Serra	S1	1500	7/3/2000	4

É de notar a redundância (repetição desnecessária de certos dados) quer a nível dos dados dos clientes, quer a nível dos dados dos produtos!

Surge assim a necessidade de se ter várias tabelas:

-  uma para os dados dos produtos;
-  uma para os dados dos clientes;
-  uma para as encomendas efectuadas pelos clientes em relação aos produtos da empresa.

Assim, de uma só tabela, passaríamos a ter três tabelas, evitando assim a redundância ou repetição desnecessária da informação. É de notar que a informação é armazenada apenas uma vez na sua respectiva e própria tabela.





<i>Cientes</i>			
Cód_Cliente	Nome_Cliente	Endereço	TelefFax
11	Silva	Lisboa	665544
12	Santos	Porto	554433
13	Costa	Coimbra	332211
14	Castro	Faro	443322

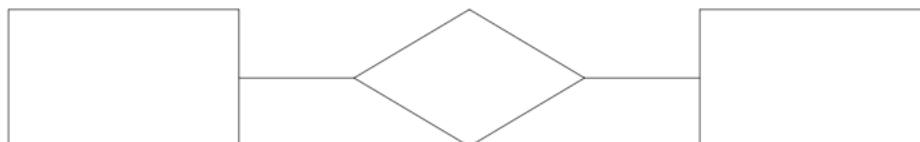
<i>Produtos</i>			
Cód_Produto	Produto	Modelo	Preço
101	Martelo	M1	250
111	Alicate	A1	400
121	Tesoura	T1	500
131	Serra	S1	1500

<i>Encomendas</i>			
Cód_Cliente	Cód_Produto	DataEnc	Quantid
11	111	4/3/2000	5
12	101	4/3/2000	10
13	131	4/3/2000	2
14	121	4/3/2000	6
11	101	5/3/2000	10
11	121	5/3/2000	5
13	111	6/3/2000	4
13	131	6/3/2000	1
14	111	7/3/2000	10
11	131	7/3/2000	4

Surge aqui o conceito de **relacionamento** entre duas tabelas ("*Cientes*" e "*Produtos*"), através dos seguintes campos comuns:

-  "*Cód\_Cliente*", comum às tabelas "*Cientes*" e "*Encomendas*";
-  "*Cód\_Produto*", comum às tabelas "*Produtos*" e "*Encomendas*".

Estes relacionamentos podem ser representados da seguinte forma com um **diagrama E-R** ou **Entidade - Relacionamento**:



## Sistemas de Gestão de Bases de Dados

Para criar e gerir base de dados (com uma ou várias tabelas) são necessários programas específicos: os **SGBD – Sistemas de Gestão de Bases de Dados** ou **DBMS – Database Management Systems**.

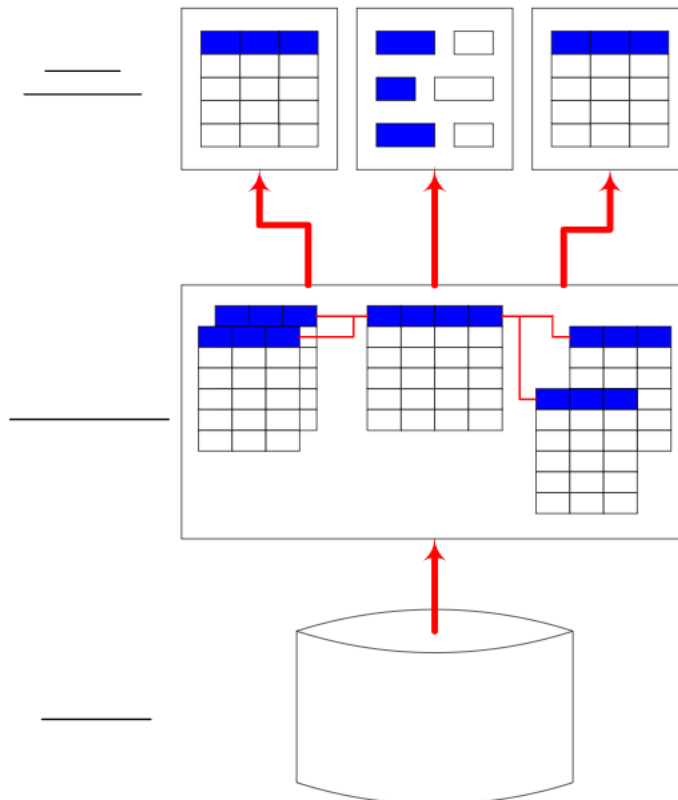
Nestes sistemas criamos e manipulamos bases de dados que estruturam os dados com independência em relação às aplicações que os manipulam, ou seja, podemos alterar a estrutura de uma base de dados sem que isso implique a necessidade de reformular o programa que opera com os dados.

### Os três níveis da arquitectura de um SGBD

1. **Nível Físico**: discos, disquetes, bandas magnéticas, etc.;



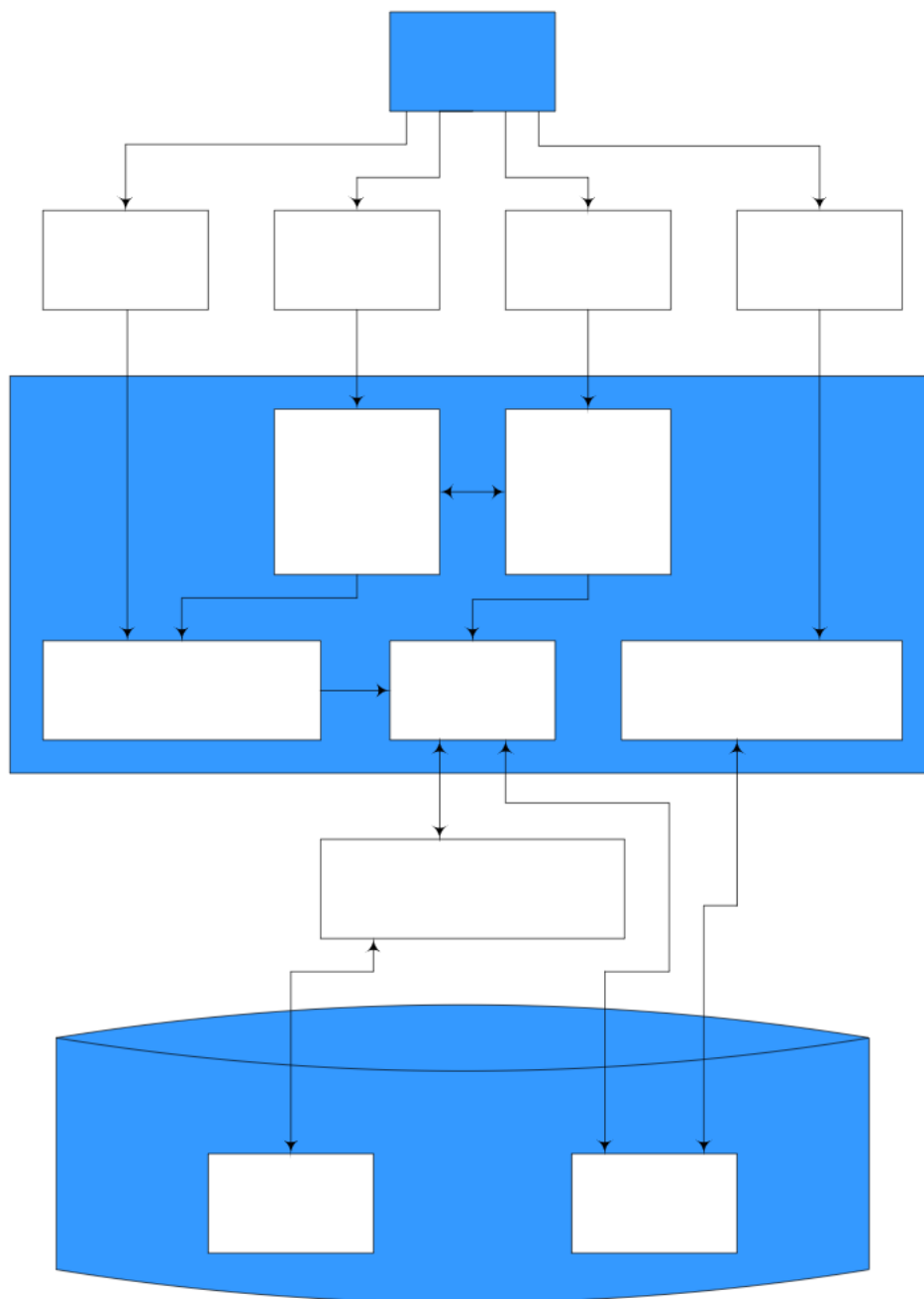
- Nível Conceptual:** onde o utilizador realiza a estruturação ou organização dos dados (definição dos campos de uma tabela e das relações entre tabelas, por exemplo);
- Nível de Visualização:** interfaces de comandos e visualização dos dados. É a forma como os dados são apresentados aos utilizadores finais, através de interfaces gráficas proporcionados pelo SGBD.



#### Principais tipos de operações com um SGBD

- Operações de definição e alteração da estrutura de uma base de dados** (chamada Linguagem de Definição de Dados): criação de uma nova base de dados, criação de um novo ficheiro ou tabela, alteração da estrutura de campos de uma tabela, criação e alteração de ficheiros e índices, eliminação de ficheiros ou tabelas de uma base de dados.
- Operações de manipulação de dados, sem alteração da estrutura da base de dados** (chamada Linguagem de Manipulação de Dados): consultas ou pesquisas de dados, inserção de novos dados (registos), alteração de dados existentes (campos e registos), eliminação de dados (registos).
- Operações de controlo dos dados:** atribuição ou supressão dos níveis de acesso aos dados em relação a utilizadores ou grupos de utilizadores.

**Nível de  
Visualização**



Interfa  
de  
Aplica

Além das **Linguagem de Definição de Dados** (*DDL – Data Definition Language*) e **Linguagem de Manipulação de Dados** (*DML – Data Manipulation Language*) os sistemas de SGBD podem disponibilizar uma outra linguagem de programação chamada de **Linguagem Hospedeira**. No caso do sistema que vamos estudar (*Microsoft Access*) a linguagem utilizada é uma versão da linguagem de programação *Visual Basic for Applications* (*VBA*).



## Modelos de Bases de Dados

Quando falamos em modelos de bases de dados, referimo-nos a modelos conceptuais (modo como são feitas):

### Modelos baseados em objectos

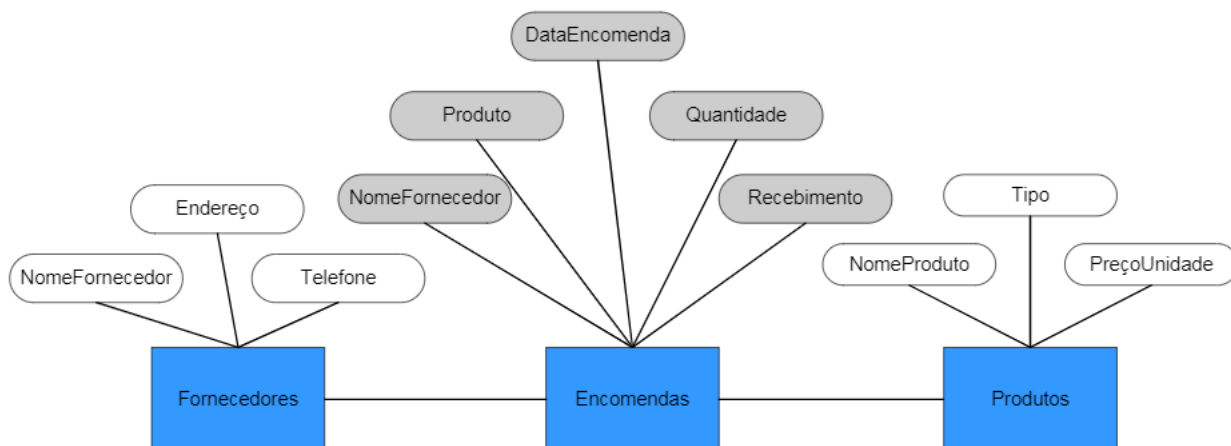
Procuram representar a realidade através de objectos (**entidades**), que podem ser transpostas para o campo da programação, contendo informação relevante sobre as entidades reais que representam.

Entre este género de modelos surgem:

- Modelo Entidade – Relacionamento (modelo E-R)
- Modelos orientados por objectos
- Modelos semânticos
- Modelos funcionais

Interessa-nos apenas focar o primeiro caso: o **Modelo Entidade – Relacionamento (modelo E-R)**, visto este nos proporcionar uma metodologia de abordagem e representação da realidade que se revela bastante interessante para a concepção e *design* de uma base de dados, mesmo utilizando um outro modelo conceptual, como é o caso do modelo relacional.

O **Modelo E-R** procura criar uma simulação ou representação da realidade através dos conceitos de **entidade** e **relacionamento**. As entidades podem representar pessoas (cidadãos, funcionários, alunos, etc.), organizações (empresas, escolas, departamentos, etc.), coisas (produtos, facturas, livros, etc.) e os relacionamentos procuram traduzir as relações entre as entidades consideradas (a relação entre os funcionários e os departamentos de uma organização ou os produtos de uma empresa e as encomendas dos seus clientes).





## Modelo Entidade – Relação (E – R)

Embora o modelo mais utilizado seja o relacional, isso não invalida que o modelo E – R igualmente o não seja. Assim:

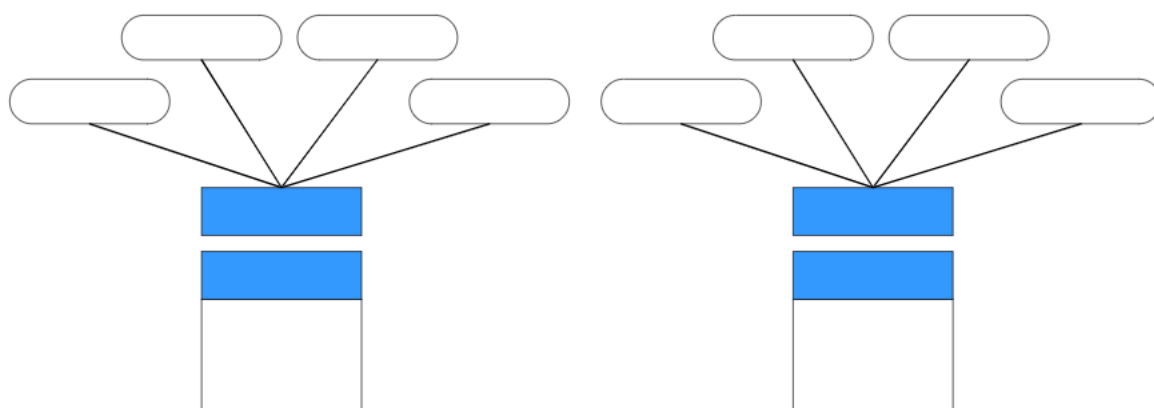
- o modelo E – R revela-se bastante útil para definir o esquema ou *design* geral de uma Base de Dados, aquando da primeira fase de concepção, antes da sua implementação num SGBD;
- alguns SGBDs utilizam alguns conceitos provenientes do modelo E – R (na secção de desenho da BD, por exemplo).

### Entidades e Atributos

Neste modelo, as entidades, são compostas ou caracterizadas por **atributos** – elementos que caracterizam as entidades.

Exemplos:

- Pessoas*: nº de BI, data de nascimento, morada, telefone, ...
- Empresas*: nome, nº de contribuinte, morada, telefone, fax, ...
- Produtos*: código de identificação, nome ou designação, categoria, modelo, tipo, cor, ...



Desta forma é fácil de perceber que:

- Entidade ou classe de entidades ⇔ **tabelas**;
- Vários elementos da entidade ⇔ **registos**;
- Atributos da entidade ⇔ **campos**.





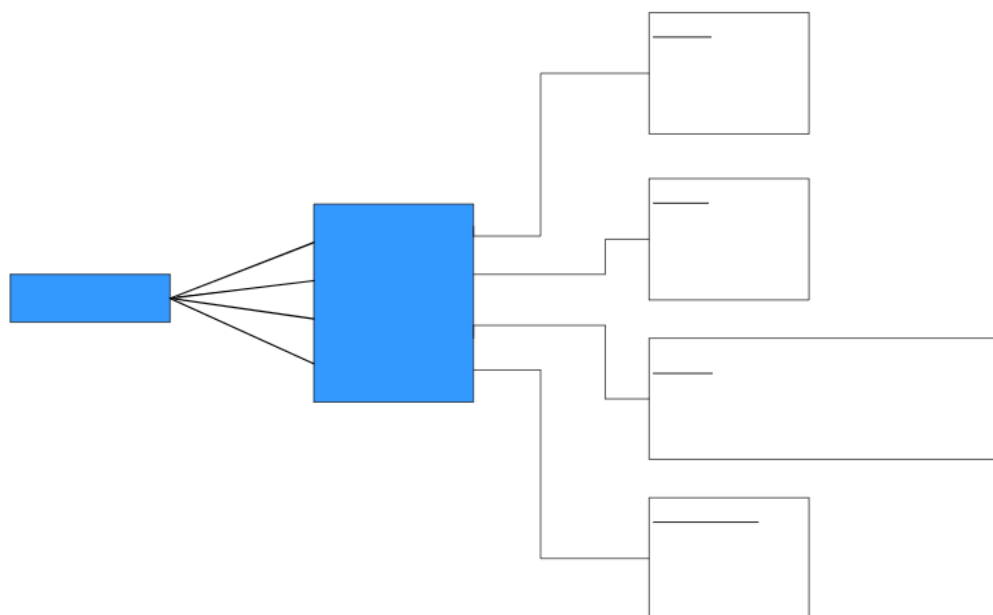


Nome	Idade	Cargo	Vencimento
Albano Silva	35	Chefe de Vendas	200
Ana Teixeira	30	Contabilista	150
Rui Fonseca	45	Electricista	100
Carla Pinto	25	Directora de Produção	200






### **Valores e Domínios dos Atributos**

Como é de prever, cada atributo das entidades contém um determinado valor. Estes **valores** identificam cada uma das entidades perante as demais.

Em cada atributo, cada um destes valores pertence a um determinado conjunto – o **domínio** do atributo.



O **domínio** de um atributo, corresponde num SGBD ao tipo de dados que o campo suportará, como por exemplo:

-  String ou texto – caracteres ou texto;
-  Number – números;
-  Date – datas;
-  Memo – texto mais extenso;
-  ...

Só assim se consegue evitar alguns erros usuais na introdução ou alteração de dados (introdução de caracteres não numéricos num campo que só suporte valores numéricos, etc.).





## Tipos de atributos

### Atributos Atómicos

Atributo que não é possível decompor em unidades mais elementares.

### Atributos Compostos

Atributos que podem ser decompostos em parcelas elementares, como por exemplo: primeiro nome, segundo nome, último nome, etc.

Todo o atributo de uma entidade deverá corresponder a valores elementares. Cada atributo não deve conter mais que um valor ao mesmo tempo para uma determinada entidade.

Quando se comete o erro de definir como atributo de uma entidade algo que pode corresponder a vários valores ao mesmo tempo, não estamos perante um atributo mas sim de um conjunto de atributos.

*Exemplo:*

Entidade Aluno

Atributo: Disciplinas

Neste caso o atributo iria conter vários valores para a mesma entidade.

Deverá existir sempre um atributo que identifique sem ambiguidades cada entidade concreta – o **atributo identificador**.

Estes atributos identificadores irão desempenhar posteriormente o papel de **chaves** (*keys*).

Esta chave é utilizada pelos SGBDs para criar um ficheiro de índice (*index*) com base no qual se torna possível uma acesso mais rápido aos dados.

É também com estas chaves que são feitos os relacionamentos entre diferentes entidades ou tabelas numa base de dados.

Por vezes é necessário, ou conveniente, utilizarmos chaves artificiais, como por exemplo, nº empregado, nº de processo, código do produto, etc.

## Relacionamentos entre entidades

### Tipos de relacionamentos quanto ao número de entidades na relação

#### 1. Relações Unárias

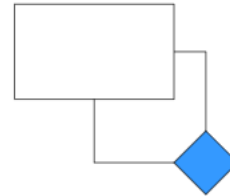
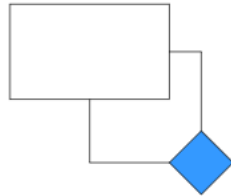
Classe de entidades que mantém um relacionamento consigo própria.

*Exemplo:*

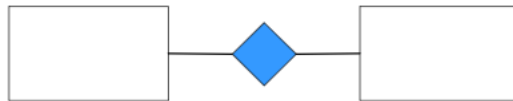
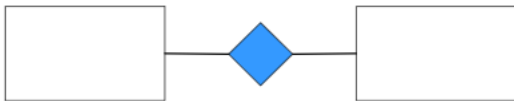
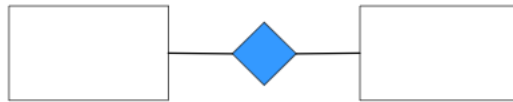
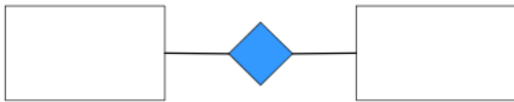
- Numa base de dados concebida para registar os jogos entre as equipas desportivas de um campeonato, podemos considerar que existe uma única entidade – Equipas – que jogam entre si;



- Se considerarmos a relação superior-subordinado que pode existir entre os empregados duma empresa, a entidade em questão é só uma: Empregados; a relação estabelece-se exclusivamente entre os elementos dessa mesma entidade.



Nos diagramas E-R, um relacionamento é representado por um losango e por linhas que interligam as entidades entre si através de losangos. No caso de uma Relação Unária, a linha parte da entidade para ela mesma, passando pelo losango que representa a relação.



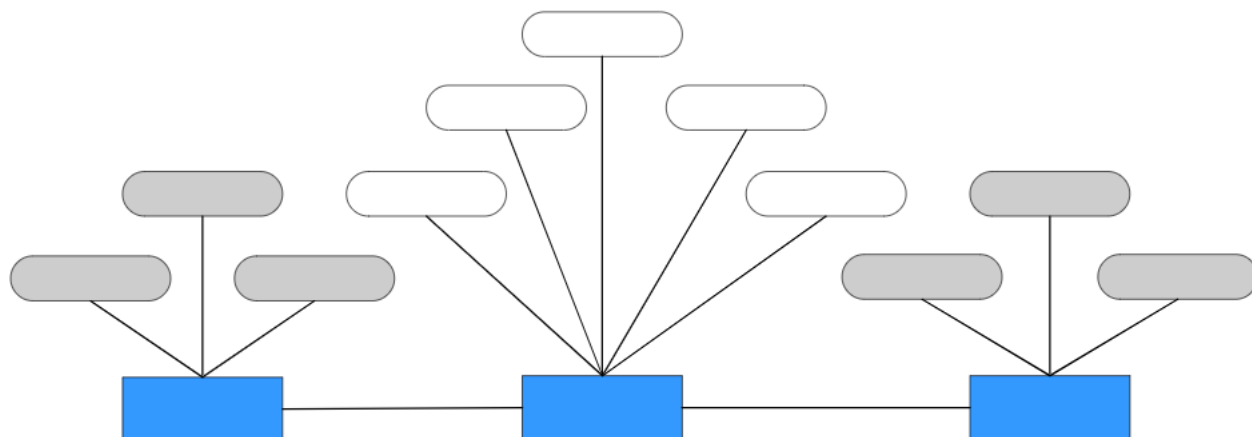
## 2. Relações Binárias

Sempre que temos duas entidades ou classes de entidades entre as quais existe qualquer forma de relacionamento.

Exemplo:

- Os clientes de um banco e as contas dos clientes
- Os fornecedores de uma empresa e os produtos fornecidos ou adquiridos pela empresa
- Os produtos vendidos por uma empresa e os clientes dessa mesma empresa
- Os alunos de um curso e as disciplinas em que eles estão inscritos

Equip



Embora se apresente nos exemplos, o relacionamento representado por um losango com um R, em muitas situações este losango pode ter uma designação própria.

Também é costume indicar os atributos através de elipses ligadas aos rectângulos das entidades.

Em algumas situações o relacionamento é efectuado através de uma outra entidade, ou seja, o relacionamento passa a ter três tabelas e não duas, onde duas são as entidades e uma terceira será o relacionamento. Nesse caso, podemos incluir também os atributos desse relacionamento, ou seja, por outras palavras, os campos da tabela correspondente ao relacionamento.

Exemplo:

A relação binária que representa os fornecedores de uma empresa e os produtos fornecidos. Se quisermos representar os atributos das duas entidades e também os atributos do relacionamento, poderemos ter algo semelhante ao que é mostrado na figura seguinte:

## Endereço

### 3. Relações Ternárias

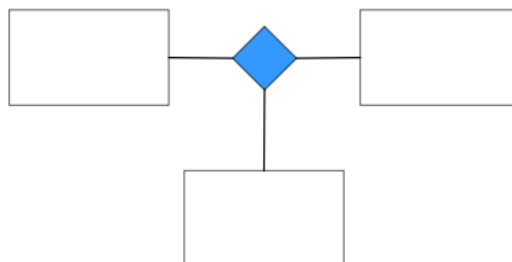
Envolvem simultaneamente três entidades no mesmo relacionamento.

Exemplo:

Suponhamos que nos interessava uma base de dados que nos desse conta da seguinte situação: registo de filmes publicitários feitos por uma agência de publicidade, os realizadores desses filmes e os actores ou modelos intervenientes em cada filme.

Se definirmos uma relação ternária entre as entidades filmes, realizadores e actores, a base de dados poderia dar-nos respostas a perguntas como:

- quais os filmes realizados pelo realizador x em que entrou o actor y?
- quais os actores que entraram nos filmes realizados pelo realizador z?



### Tipos de relacionamentos binários

A relação mais comum é a do tipo binário.

Há ainda a considerar várias situações distintas, dependendo do modo como as entidades de cada lado participam na relação.

Esta questão interfere na estrutura da base de dados, nomeadamente no número de tabelas que teremos em cada caso.

Considerando o **número de elementos** de uma entidade que se relacionam com os elementos da outra entidade:

1. Relações Um-para-Um
2. Relações Um-para-Vários ou Vários-para-Um
3. Relações Vários-para-Vários

Considerando a **obrigatoriedade de participação** de todos os elementos:

1. Participação obrigatória de ambas as partes
2. Participação não obrigatória de uma das duas entidades
3. Participação não obrigatória de nenhuma das entidades

### Modelo Relacional

#### Chaves de uma tabela

Considerando a seguinte tabela:

Fornecedores			
Nome	Morada	Localidade	Telefone
Silva	R. Nova	Lisboa	6633
Nunes	R. Velha	Lisboa	6699
Metalin	R. Nova	Lisboa	9966
Silva	R. Nova	Porto	4488
Sousa	R. Velha	Porto	6699



Embora não existam duas linhas ou registos iguais, não há nenhum atributo que, por si só, seja capaz de definir de modo unívoco (sem ambiguidades) cada uma das linhas ou registos da tabela, visto que em qualquer campo da tabela, aparecem valores repetidos.

Surge assim o conceito de **chave** – atributo ou conjunto de atributos que nos permite identificar de modo único ou unívoco cada entidade concreta ou registo da tabela.



### Chave

Atributo ou conjunto de atributos que nos permite identificar de modo único ou unívoco cada entidade concreta ou registo da tabela.

Esta **chave** corresponde ao que anteriormente designámos de **atributo identificador**, mas com a seguinte diferença:

- ☞ uma chave pode ser constituída por um atributo: **chave simples**;
- ☞ ou por mais do que um atributo: **chave composta** ou chave concatenada.

No caso da tabela apresentada poderíamos por a hipótese de utilizar dois campos, que combinados nos permitissem identificar univocamente os vários registos. Assim, a chave composta formada pelos atributos **Nome** e **Localidade** poderia desempenhar esse papel. Outras combinações poderiam também ser soluções.

Todas as chaves possíveis – simples ou compostas – são designadas por **chaves candidatas**.



### Chaves Candidatas

Todas as chaves possíveis – simples ou compostas.

Entre estas chaves candidatas existe uma que será a mais indicada ou a escolhida para desempenhar o papel de chave – a **chave primária**: atributo ou conjunto de atributos que assume a função de identificar de modo unívoco as entidades ou registos de uma tabela.



### Chave Primária

Atributo ou conjunto de atributos que assume a função de identificar de modo unívoco as entidades ou registos de uma tabela.



É aconselhável, embora não obrigatório que as tabelas de entidades possuam pelo menos um atributo identificador, que possa desempenhar o papel de chave primária (chave primária simples), evitando assim ter de recorrer a uma combinação de atributos (chave composta).

Podem surgir também situações em que os campos existentes (considerados *naturais* a uma entidade), não são considerados seguros para desempenhar o papel de chave, como é o caso de um campo “*Nome*” (duas pessoas podem ter o mesmo nome).

Nesta situação recorre-se a um **atributo artificial**.

Voltando à tabela anterior, poderíamos acrescentar-lhe um campo de código, como, por exemplo: ***CodForn***.

#### Regras a respeitar com as chaves primárias:

1. ***Ser unívoca***: tem de ter um valor único para cada entidade concreta (registo na tabela);
2. ***Não ser nula***: não poderá conter um valor nulo em nenhum registo;
3. ***Não ser redundante***: no caso de ser uma chave composta, esta não deverá conter mais do que os campos necessários para identificar os registos de modo unívoco

#### **Relacionamentos e chaves externas**

Uma das características essenciais do **modelo relacional** é que permite estabelecer **relacionamentos** entre entidades ou tabelas de entidades.

Estas relações são estabelecidas através das chaves primárias nas respectivas tabelas.

Exemplo:

Fornecedores(CodForn, Nome, Endereço, Telef)

Produtos(CodProd, NomeProd, Modelo)



Na descrição das entidades e atributos no modelo relacional, é costume indicar-se o nome da entidade seguido pela lista dos seus atributos entre parêntesis; os atributos que funcionam como chave são indicados a sublinhado ou a negro.

Para que a base de dados nos dê resposta a questões como:

“Que produtos são fornecidos pelo fornecedor X?”

“Que fornecedores podem fornecer o produto P?”

Para isto ser possível, é necessário estabelecer um relacionamento entre as duas tabelas.

Teremos então um relacionamento do tipo *vários-para-vários* (*N-para-N*), já que:



*“Cada fornecedor pode fornecer mais que um produto”*

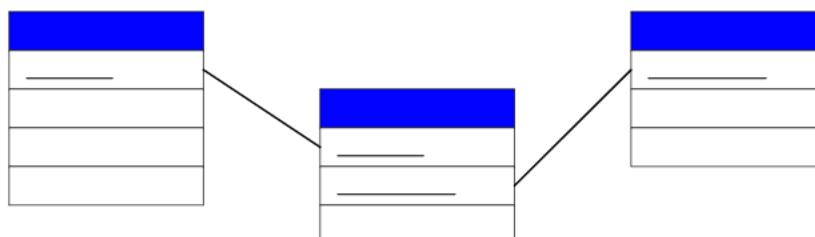
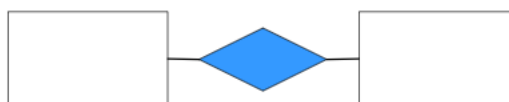
*“Cada produto pode ser fornecido por mais que um fornecedor”*

Para este relacionamento necessitamos de três tabelas: Fornecedor, Produtos e uma outra onde iremos registar o relacionamento entre as duas entidades, ou seja, os produtos fornecidos por cada fornecedor. Enquanto que às primeiras damos o nome de **tabelas de entidades**, à outra, que não representa propriamente uma classe de entidades, mas ocorrências de relacionamentos, **tabela de relacionamento**.



### Tabela de Relacionamento

Tabela que não representa uma classe de atributos, mas ocorrências de relacionamentos.



Esta **tabela de relacionamentos** deverá incluir, entre os seus campos, as chaves das tabelas que entram no relacionamento.

Quando isto acontece, do ponto de vista desta última tabela, diz-se que se trata de uma **chave externa** – campo de uma tabela que é chave primária de uma outra entidade ou tabela.



### Chave Externa

Campo de uma tabela que é chave primária de uma outra entidade ou tabela.

Na maioria dos casos, é através da inclusão de campos que são chaves primárias nas suas tabelas de origem em outras tabelas (onde são chaves externas) que são estabelecidos os relacionamentos entre as diferentes entidades de uma base de dados relacional.





Numa tabela de relacionamentos, a chave primária é normalmente uma chave composta, constituída por chaves externas.



Numa tabela de relacionamentos, a chave primária é normalmente uma chave composta, constituída por chaves externas.

### **Preservação da integridade da informação**

A actualização de uma base de dados deve ser feita de forma a não se perder a consistência dos dados. Assim, esta, deve ser assegurada pelo SGBD ou pelas aplicações que forem criadas a partir desse mesmo SGBD.

Desta forma, devem estar assegurados dois tipos de integridade:

1. Integridade de Entidade
2. Integridade Referencial

#### **1. Integridade de Entidade**

Impõe que os valores dos campos que correspondem à chave primária de uma entidade não podem ser nulos nem iguais a outros já existentes na tabela.

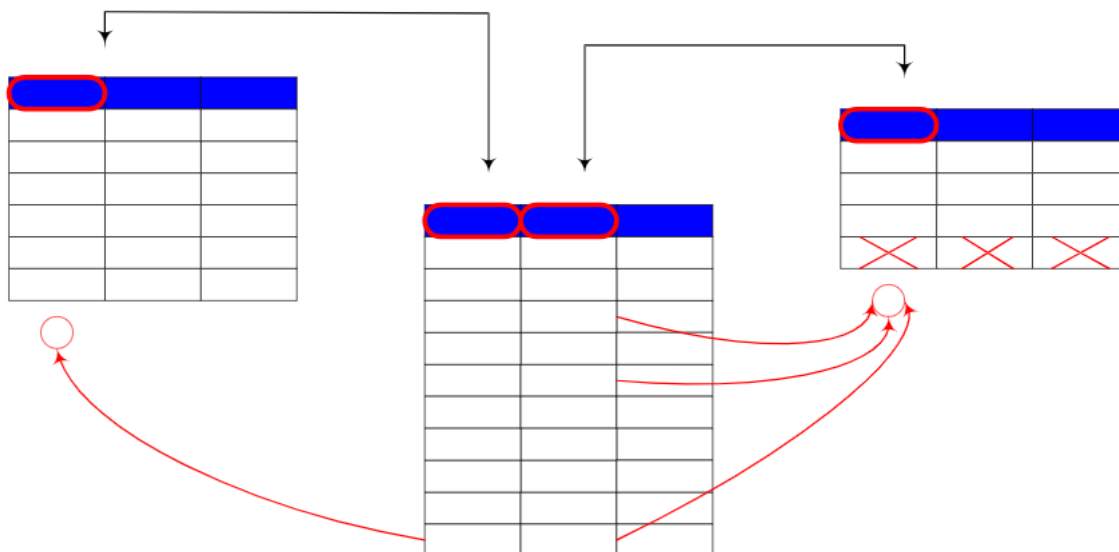
Estas situações fariam com que a chave deixasse de identificar de modo unívoco os registos da tabela, portanto, deixaria de poder funcionar como chave.

#### **2. Integridade Referencial**

Impõe que um valor de uma chave externa tem obrigatoriamente de existir na chave primária da tabela relacionada com aquela chave externa.

Sempre que se introduz um valor num campo que é chave externa de uma tabela, o SGBD tem de certificar-se que esse valor existe na chave primária da tabela referenciada por aquela chave externa – caso contrário, a base de dados passaria a ter uma inconsistência ou uma falha de integridade referencial.

Por outro lado, a operação de eliminação de registos numa tabela (relacionado com outra tabela através de um mesmo valor na chave primária da primeira tabela e na chave externa da segunda), poderá levar-nos a uma outra possibilidade de violação da integridade referencial – ficarmos com registos que referenciam outro registo que já não existe.



Vejam, através de exemplos concretos, os casos mais comuns de violação da integridade referencial:

1. Na tabela *ForneceProdutos*, pretendemos introduzir um registo, em que o campo *CodForn* tem o valor 108. Estamos perante um caso de violação da integridade referencial. O campo *CodForn* é chave externa na tabela *ForneceProdutos* e chave primária na tabela *Fornecedores*; ora, nesta última, não existe nenhum registo com o código 108; portanto, a informação que pretendíamos introduzir tem uma falha de integridade referencial (referencia algo que não existe na base de dados).
2. Pretendemos eliminar um registo, na tabela *Produtos*, correspondente ao produto com o código C2. Se o fizermos, o que é que deve acontecer aos registos da tabela *ForneceProdutos* que dizem respeito a esse produto? Se os mantivermos, estamos a criar uma falha de integridade referencial (a tabela *ForneceProdutos* tem registos que referenciam produtos que não existem na tabela *Produtos*).
3. Pretendemos alterar, na tabela *Produtos*, o registo com o código T2 para T3. Trata-se de uma situação semelhante à anterior, pois que desaparece um valor (T2) numa tabela (*Produtos*), enquanto, numa outra tabela (*ForneceProdutos*) continuam a existir registos que faziam referência a esse valor.

O SGBD ou aplicação que estiver a gerir a base de dados deve estar preparado para evitar estas situações em que a informação perde consistência ou integridade referencial, enviando mensagens ao utilizador que lhe permitam rectificar a operação.

103	Metallin
104	Silva
105	Sousa
106	Xavier

?



## Modelação da informação e design de uma base de dados

### Abordagem Bottom-Up – o processo de normalização

Esta abordagem propõe um conjunto de normas tendo em vista uma boa estruturação das bases de dados relacionais, de forma a evitar as típicas anomalias de redundância ou perda de integridade da informação.

Este conjunto de normas é composto pelas chamadas **formas normais** e constitui o **processo de normalização**.



Inicialmente foram estabelecidas três formas normais:

- 1ª forma normal (1FN)
- 2ª forma normal (2FN)
- 3ª forma normal (3FN)

Posteriormente, surgiram outras formas normais, para além daquelas:

- Forma Normal de Boyce-Codd (FNBC)
- 4ª forma normal (4FN)
- 5ª forma normal (5FN)

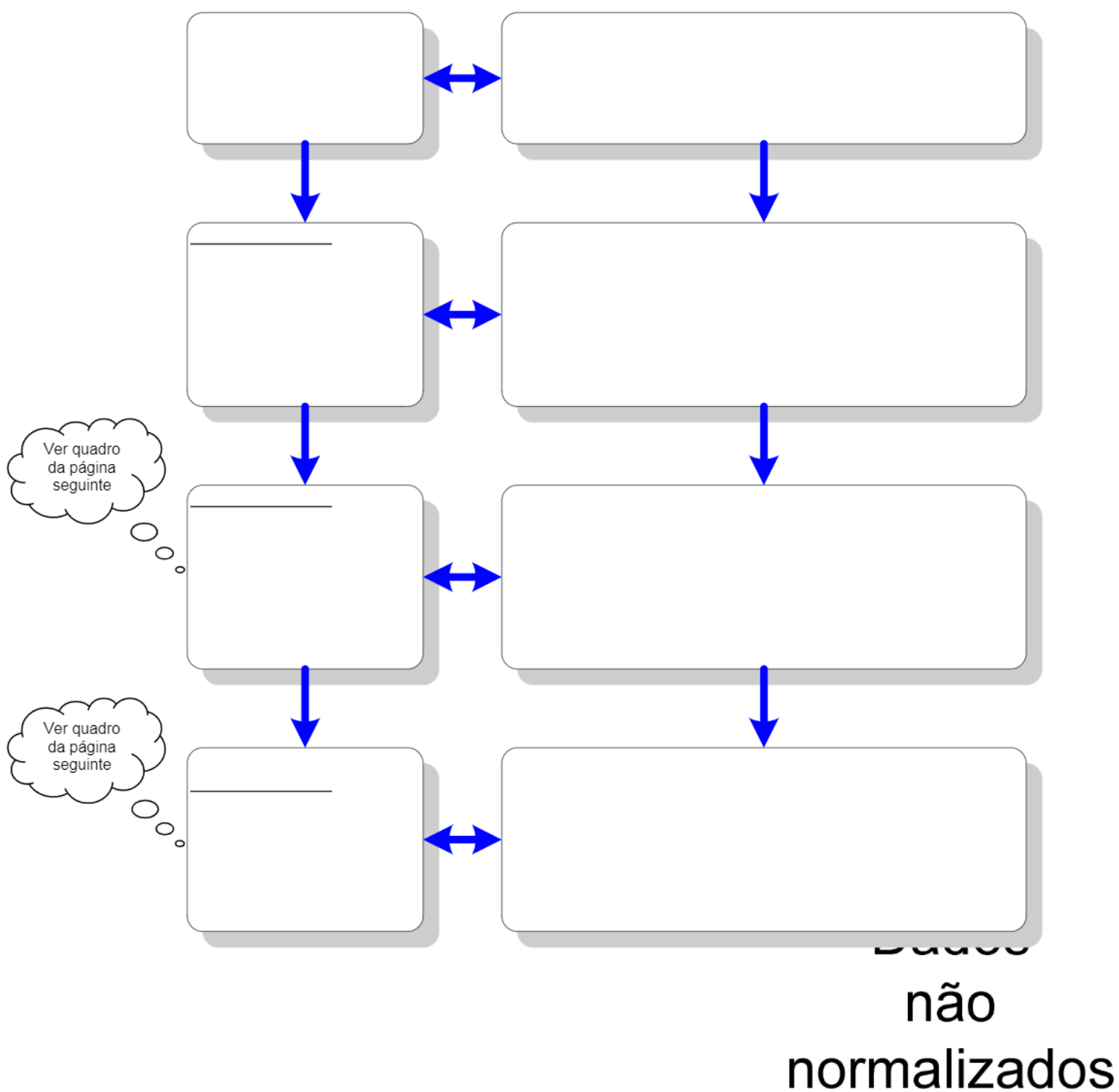
Na maioria dos casos, uma base de dados que respeite a 3ª forma normal, é considerada adequadamente elaborada para funcionar num SGBD relacional. As formas superiores destinam-se a situações mais complexas e menos comuns.

Na maioria dos casos, o **processo de normalização** consiste no seguinte:

1. **Definição das entidades**, com todos os seus atributos;
2. **Análise das relações e dependências** entre os atributos das entidades, comparando-se a estrutura analisada com as referidas formas normais;
3. **Reestruturação dos atributos** sempre que uma entidade ou tabela apresentar uma característica não conforme a uma forma normal. Estes atributos poderão ser, inclusive, separados da entidade original para formar com eles uma nova entidade ou tabela;
4. **Repetição do processo** até que todas as entidades estejam na forma normal pretendida.



Vamos então representar, esquematicamente, todo o processo de normalização até à 3ª forma normal:





A compreensão da 2ª e 3ª formas normais implica a compreensão do conceito de

### Dependência Funcional

Consideremos a seguinte situação: os alunos e as disciplinas em que se encontram inscritos. Especifiquemos um pouco melhor a parte relativa às disciplinas – em que cada disciplina é identificada por um código, o nome da disciplina e o ano do curso a que ela diz respeito.

*Alunos(N.º Aluno, Nome, Morada, CodDiscip, Disciplina, AnoCurso)*

Nota: os campos que surgem a sublinhado representam a chave da entidade ou tabela, ou seja, o conjunto de atributos que identificam cada registo de modo único na tabela.

Se criarmos uma tabela a partir dos atributos enumerados, a chave dessa tabela não pode ser constituída apenas pelo atributo "N.º Aluno", mas pelo conjunto "N.º Aluno, CodDiscip".

Como cada aluno pode inscrever-se em várias disciplinas, cada inscrição dá origem a um registo na nossa tabela.

Assim sendo, só podemos diferenciar um registo de outro através do par de atributos "N.º Aluno, CodDiscip".

O par de atributos "N.º Aluno, CodDiscip" identifica de modo único todos os registos da tabela – por isso é que é chave.

Nestas situações, diz-se que o conjunto de atributos "N.º Aluno, CodDiscip" **determina funcionalmente** os outros atributos (Nome, Morada, Disciplina, AnoCurso).

Reciprocamente, diz-se que os atributos Nome, Morada, Disciplina, AnoCurso são **funcionalmente dependentes** do par "N.º Aluno, CodDiscip".

Em geral:

- Um atributo ou conjunto de atributos é **determinante** de outros atributos quando os identifica de modo unívoco;
- Os atributos identificados de modo unívoco por um outro atributo ou conjunto de atributos são **funcionalmente dependentes** deste último (atributo ou conjunto).

No caso da entidade ou tabela definida pelos atributos (N.º Aluno, Nome, Morada, CodDiscip, Disciplina, AnoCurso), em que a chave é constituída pelo par (N.º Aluno, CodDiscip), podemos constatar que existem várias dependências funcionais:

1. Os atributos Nome e Morada são funcionalmente dependentes do atributo N.º Aluno;
2. Os atributos Disciplina e AnoCurso são funcionalmente dependentes do atributo CodDiscip;
3. Cada um dos atributos não-chave (Nome, Morada, Disciplina, AnoCurso) é funcionalmente dependente do par que constitui a chave (N.º Aluno, CodDiscip).

A segunda e a terceira formas normais vão colocar restrições em relação a estas dependências funcionais:

- A 2ª Forma Normal diz que cada atributo não-chave tem de ser funcionalmente dependente da chave na sua totalidade e não apenas de uma parte dessa chave;
- A 3ª Forma Normal diz que um atributo não-chave não pode depender funcionalmente de nenhum outro atributo que não seja chave.