INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 2: Conceitos e Estrutura de Bases de Dados

Aula 1
Introdução às Bases de Dados

UPskill

Sobre o módulo

O módulo de Conceitos e Estrutura de Bases de Dados pretende dotar os alunos de um conjunto fundamental de noções para que estejam aptos a desenhar e implementar uma base de dados relacional bem estruturada, eficiente e que possibilite pesquisas flexíveis, bem como efetuar consultas, inserir e atualizar dados.

Programa do módulo

Aula	Tema
1	Introdução às Bases de Dados; UML, Relações; Associações; Exercícios
2	Classes Associativas; Agregações; Composições; Generalizações; Exercícios
3	Introdução ao modelo relacional; Chaves; Regras de Integridade; Otimizações e Índices; Exercícios
4	Conversão modelo conceptual para modelo relacional; Exercícios
5	**Exercícios desenho de Bases de Dados**
6	**Exercícios desenho de Bases de Dados**
7	**Teste 1 (3,5h) - Teste de Desenho de Bases de Dados**
8	Introdução ao SQL; Queries simples (Select, Insert, Update)
9	SQL: Funções de Agregação, Group by e Having; Subqueries; Introdução às transacções
10	**Exercícios de SQL**
11	**Exercícios de SQL**
12	**Teste 2 (3,5h) - Teste de SQL**

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

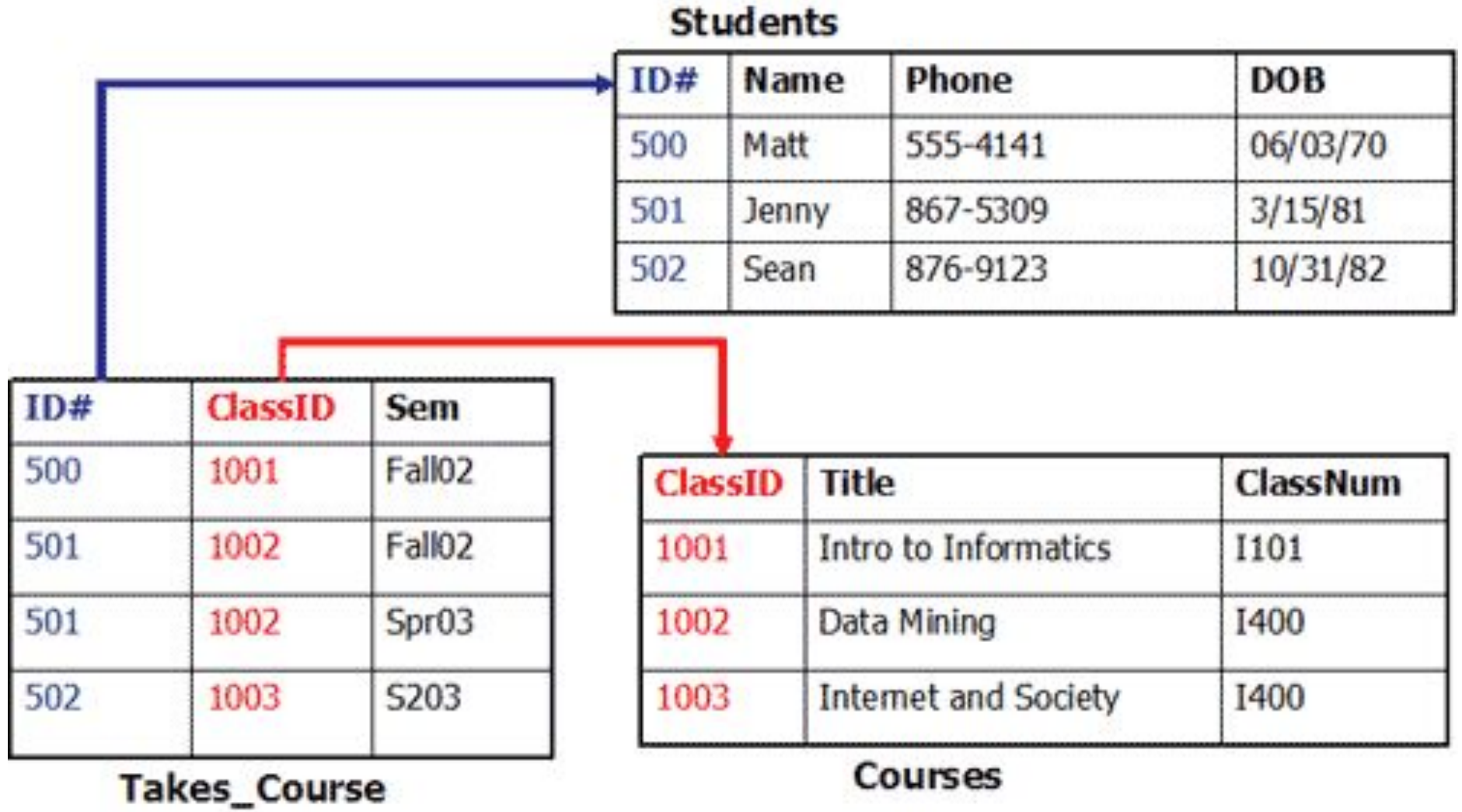
Avaliação do módulo

Teste 1 - Desenho de Bases de Dados Classificado de A-E	Aula 7 18 / 12 / 2020	**50%**
Teste 2 - SQL Classificado de A-E	Aula 12 08 / 01 / 2021	**50%**

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

O que é uma Base de Dados?

Bases de Dados são conjuntos de dados relacionados entre si com registos sobre pessoas, lugares ou coisas. São coleções organizadas de dados que se relacionam de forma a criar algum sentido (Informação) e dar mais eficiência durante uma pesquisa. São de vital importância para empresas e há mais de duas décadas que se tornaram a principal peça dos sistemas de informação e segurança. Normalmente existentes por vários anos sem alterações na sua estrutura.



Exemplo de estrutura e relações numa Base de Dados

STORE

Store_key	City	Region
1	New York	**East**
2	Chicago	Central
3	Atlanta	**East**
4	Los Angeles	**West**
5	San Francisco	**West**
6	Philadelphia	**East**
.	.	.
.	.	.

PRODUCT

Product_key	Description	Brand
1	Beautiful Girls	MKF Studios
2	Toy Story	**Wolf**
3	Sense and Sensibility	Parabuster Inc.
4	Holiday of the Year	**Wolf**
5	Pulp Fiction	MKF Studios
6	The Juror	MKF Studios
7	From Dusk Till Dawn	Parabuster Inc.
8	Hellraiser: Bloodline	Big Studios
.	.	.
.	.	.

SALES_FACT

Store_key	Product_key	Sales	**Cost**	Profit
1	6	2.39	1.15	1.24
1	2	16.7	6.91	9.79
2	7	7.16	2.75	4.40
3	2	4.77	1.84	2.93
5	3	11.93	4.59	7.34
5	1	14.31	5.51	8.80
.	.	.	.	.
.	.	.	.	.

Exemplo de estrutura e relações numa Base de Dados

Resumindo...

- Armazenamento de informação
- As Bases de Dados são constituídas por várias tabelas
- Cada tabela representa uma entidade no nosso modelo de dados
    - Ex. tabela “produtos”, “clientes” ou “registos_de_compras”
    - Cada entidade tem um conjunto de dados, representados nas colunas (um cliente pode ter uma morada, localidade e um NIF, por exemplo)
- Cada linha de uma tabela representa uma nova entrada
    - Um novo produto, um novo cliente, ou uma nova compra

Resumindo...

- Todas as entradas precisam de um ID único:
  - Forma inequívoca de identificar o registo. Muitas vezes é um número sequencial, mas também outro valor único
  - O ID do cliente pode ser um número sequencial (cliente 1, cliente 2, cliente 3) mas também pode ser o email, ou o NIF.
- Existem relações entre os dados
  - Ex. id do cliente na tabela “registo_de_compras”, para registar o cliente que efetuou a compra

Como **criar** uma Base de Dados?

Comecemos pelo seu **desenho.**

UML

Unified
Modeling
Language

iscte
INSTITUTO
UNIVERSITÁRIO
DE LISBOA
emprego
digital

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UML: Unified Modeling Language

- É uma linguagem de modelação, baseada em **esquemas** e **diagramas**.
- É a linguagem padrão para a elaboração da estrutura de projetos de software, muito adequada à modelação de sistemas.
- Permite representar um sistema de forma padronizada, com o intuito de facilitar a compreensão pré-implementação.
- Poderá ser empregada para especificação, construção e documentação de projetos de software.
- **No âmbito deste módulo, será usada para a especificação do modelo de dados de uma base de dados.**

UML: Unified Modeling Language

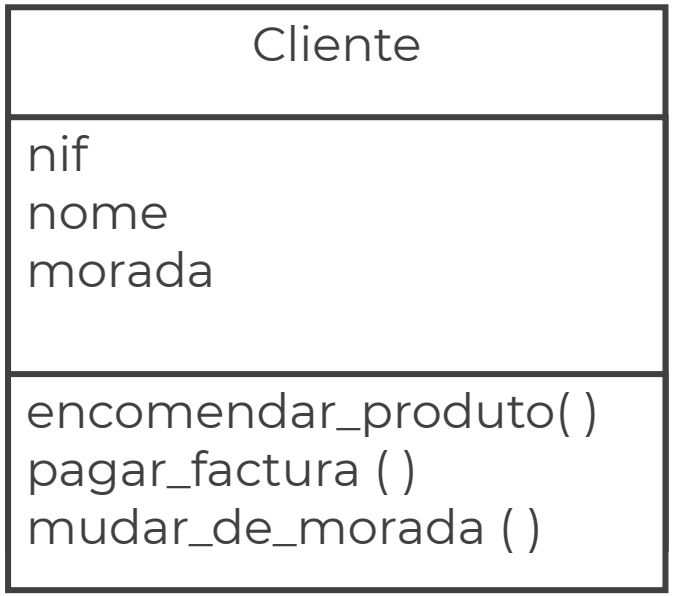
- Esta linguagem subdivise-se em:
  - **Diagramas de Classes**
  - Diagramas de Objetos
  - Diagramas de Componentes
  - (...)
  - Diagramas de Casos de Uso
  - Diagramas de Sequencia
  - Diagramas de Atividade
  - (...)
- **Para a modelação da estrutura de uma Base de Dados, é apenas necessário o recurso aos diagramas de Classes, pelo que serão os únicos abordados neste módulo.**

UML: Diagrama de Classes

- O Diagrama de Classes é utilizado para fazer a representação da estrutura das classes de um determinado sistema.

- Em Bases de Dados, servem o propósito de representar as nossas entidades (ainda não são as tabelas, mas quase), e as suas relações.

UML: Representação de uma Classe



Nome da Classe
Deve ser único no seu domínio

Atributos da Classe

Operações da Classe
Omitidas no âmbito de BD, pois apenas nos interessam os dados / atributos)

UML: Atributos de uma Classe

- Um **atributo** numa classe representa uma **característica comum** dos objectos dessa classe.

- Deve especificar-se um tipo de dados para cada atributo.
  - Neste caso, os valores que podem ser atribuídos ao atributo estão condicionados à compatibilidade com o tipo.

No fundo, cada classe neste diagrama será uma tabela na nossa BD.

E cada entrada nesta tabela será um objeto.

Relações entre Classes

- Em qualquer sistema existem objectos que se relacionam entre si.
  - Sistema universitário
    - Objectos: alunos, cursos, exames;
    - Os alunos relacionam-se com os cursos que frequentam e com os exames que fazem.
  - Sistema bancário
    - objectos: clientes, contas, balcões;
    - Os clientes relacionam-se com as contas que possuem e as contas relacionam-se com os balcões em que estão sediadas.

As ligações entre os objetos relacionados também são informação.

Quando há interesse em guardar na base de dados as ligações entre os objectos do sistema, especificam-se relações entre as classes desses objectos.

Tipos de Relações

Em UML existem os seguintes tipos de relações, que expressam diferentes semânticas de ligação entre objectos:

- Associação

  Tem dois casos especiais:

  - Agregação
  - Composição
- Generalização
- Relação de dependência

Associações

- **Associação:** Relação que permite especificar que objetos de uma dada classe se relacionam com objetos de outra classe, sendo importante saber para cada objecto quais os objectos com que está relacionado.

Um cliente pode estar associado a **muitas faturas, ou a nenhuma**.

Uma fatura está sempre associada **a um e apenas um cliente**.

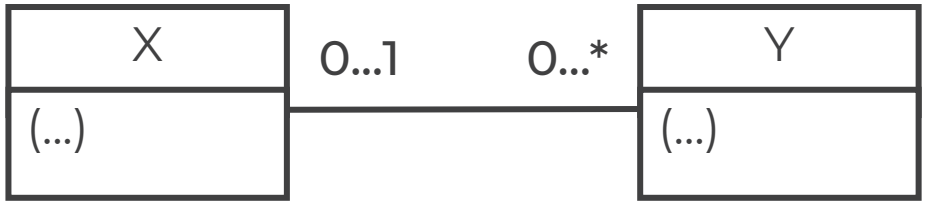
Multiplicidade das Associações

Indica **quantos** objetos da classe X...

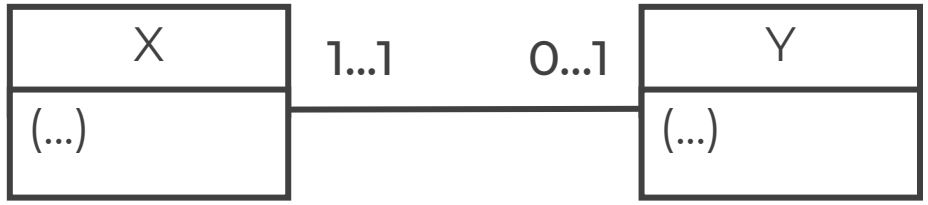
... podem estar associados a **um** objeto da classe Y

Multiplicidade das associações

... há muitas combinações possíveis. Eis as designações mais comuns:



Um-para-Muitos



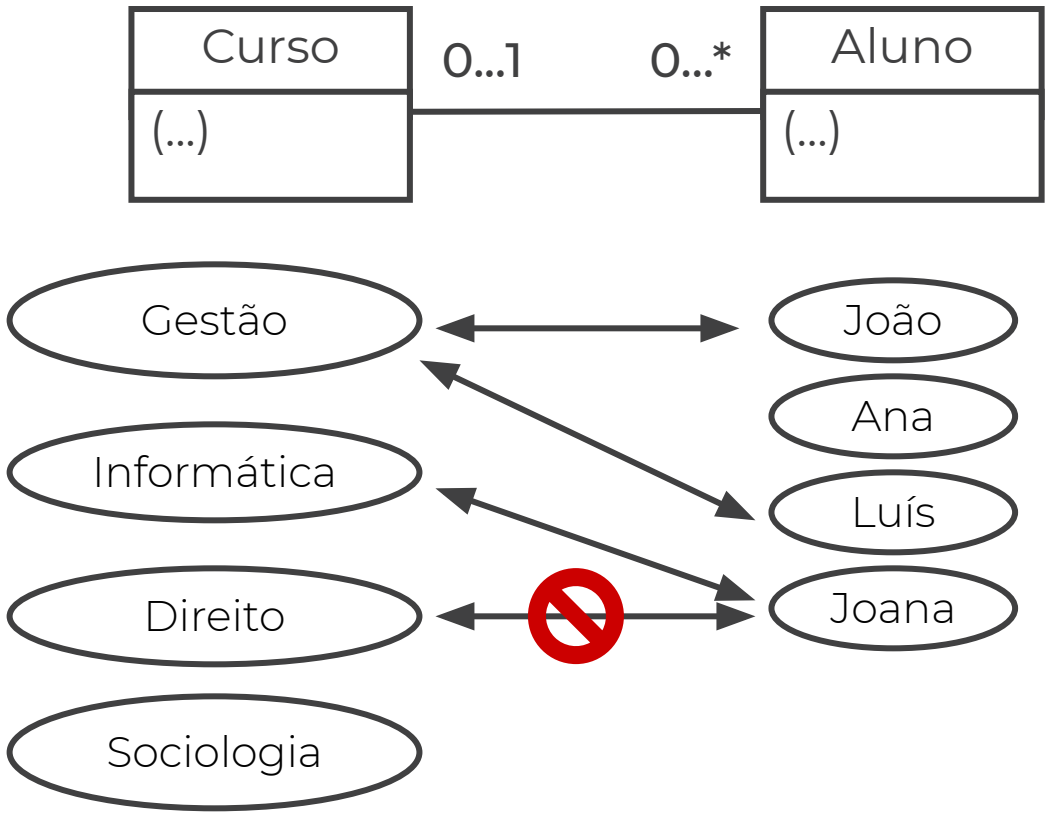
Um-para-Um

Muitos-para-Muitos

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

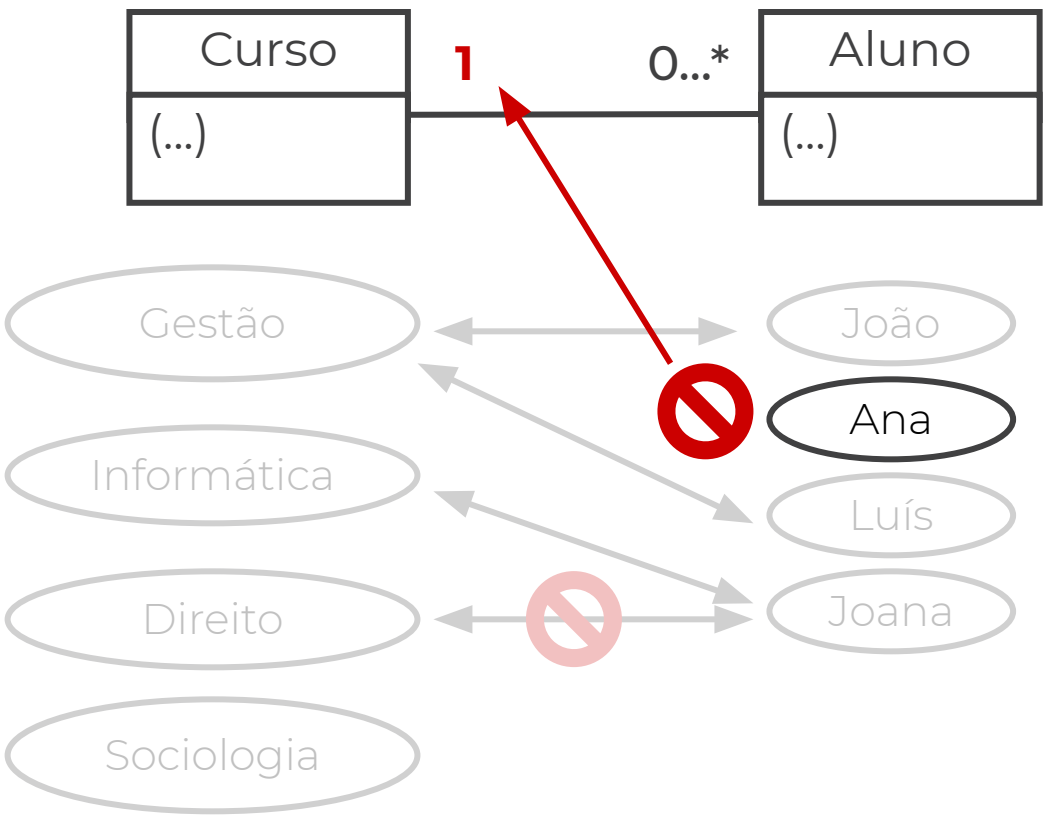
emprego digital

Associação **Um-para-Muitos**



- ***Semântica***
  - Um aluno pode estar associado a (inscrito em) apenas um curso
  - A um curso podem-se associar vários ou nenhum aluno.
- ***Funcional***
  - Dado um aluno é possível determinar em que curso está inscrito, e
  - Dado um curso é possível identificar os seus alunos.

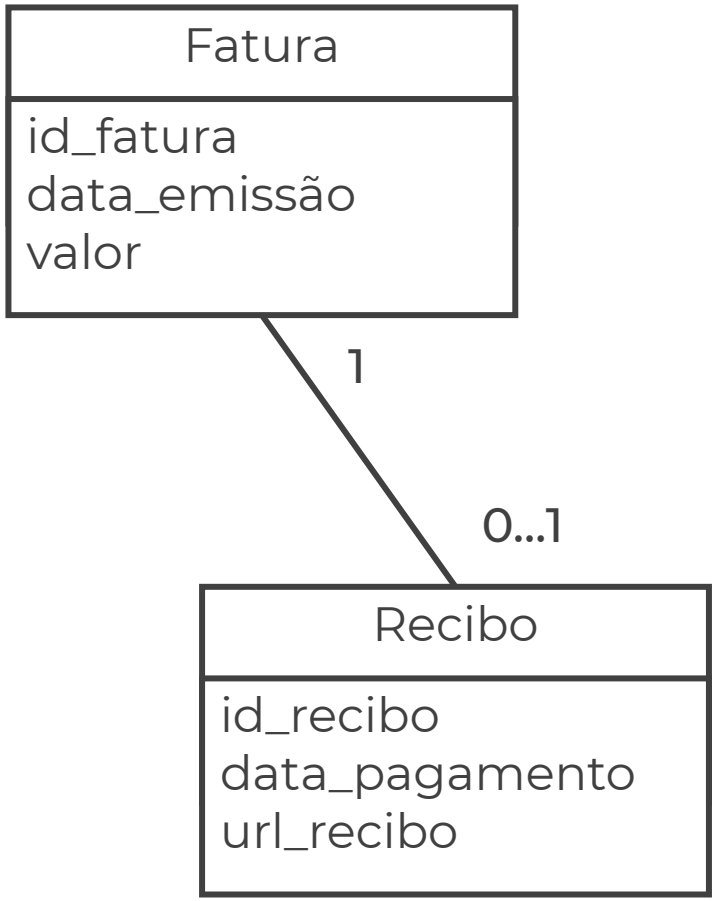
Associação **Um-para-Muitos**



- ***Semântica***
  - Um aluno pode estar associado a (inscrito em) apenas um curso
  - A um curso podem-se associar vários ou nenhum aluno.
- ***Funcional***
  - Dado um aluno é possível determinar em que curso está inscrito, e
  - Dado um curso é possível identificar os seus alunos.

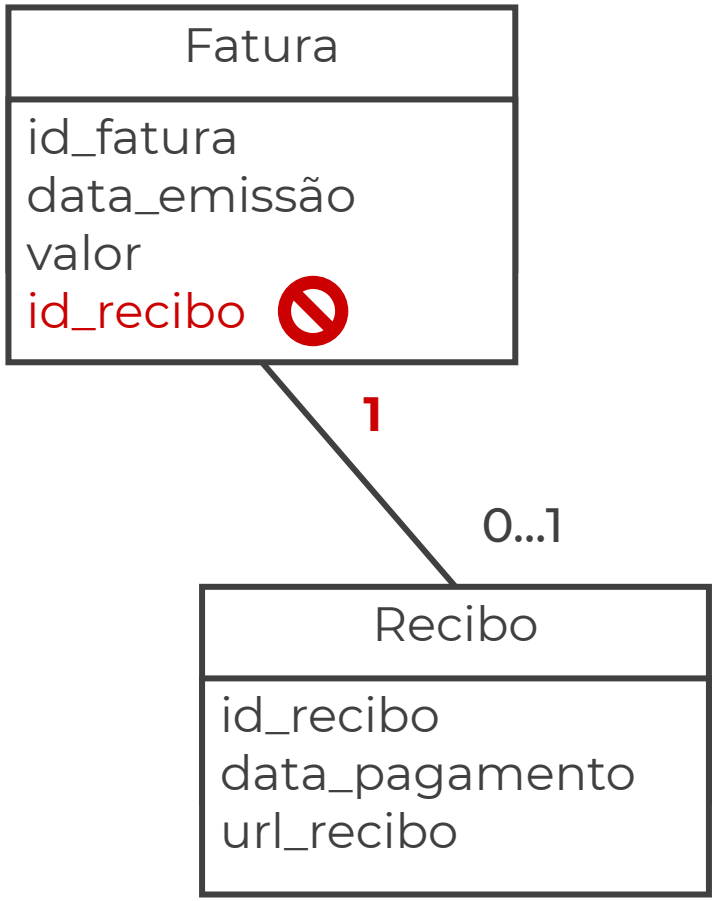
iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

Associação **Um-para-Um**



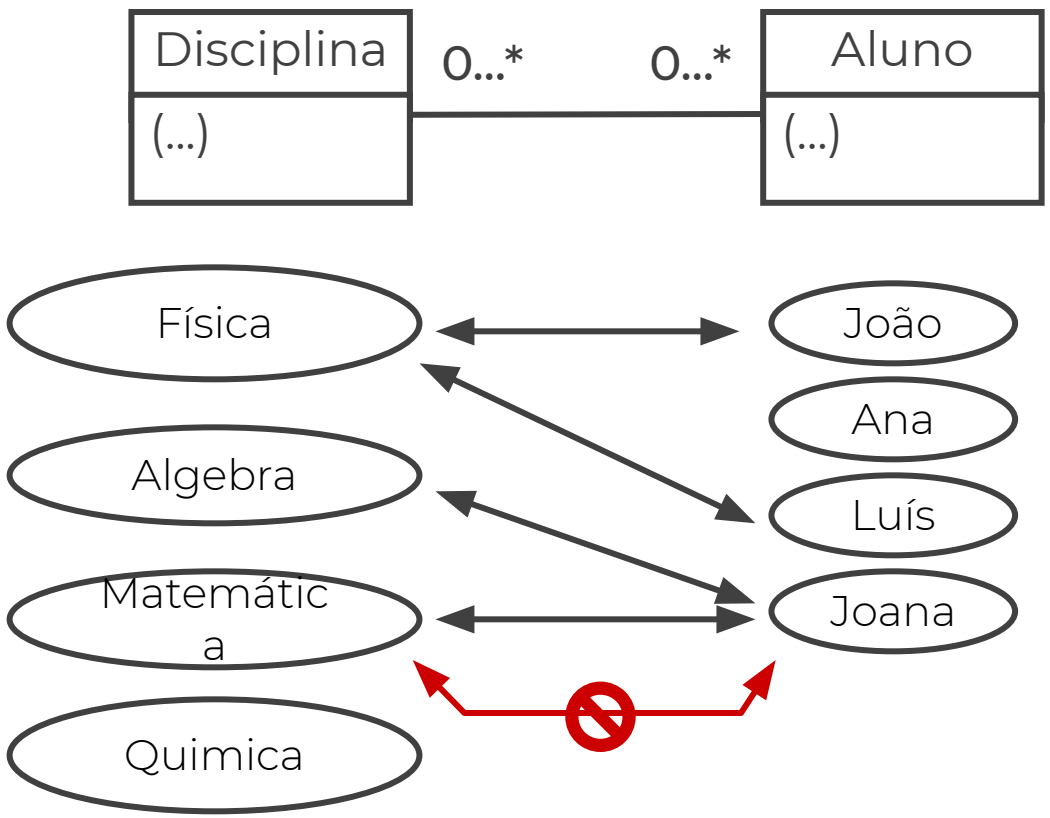
- Pelas multiplicidades, percebe-se que uma factura será necessariamente introduzida antes do respectivo recibo (quanto muito, simultaneamente).

Associação **Um-para-Um**



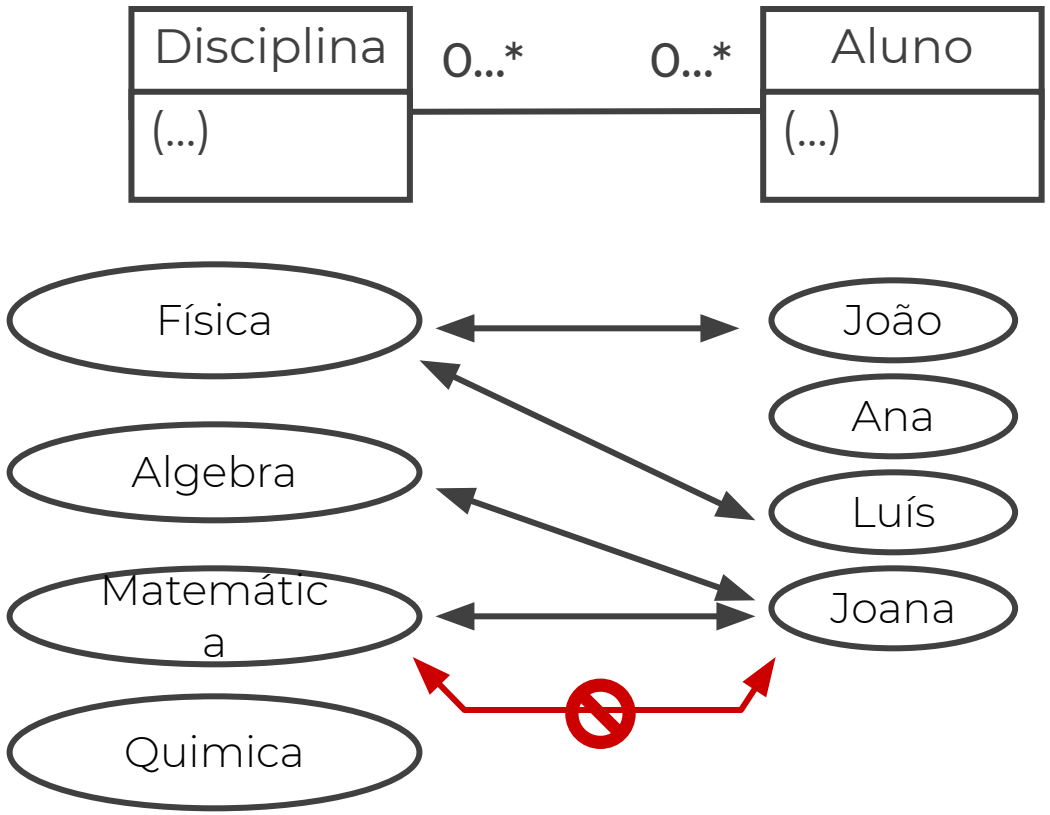
- **Não devemos colocar a chave estrangeira como atributo.** É a associação que atribui um número de recibo à factura.
  - Caso contrário, uma factura poderia ficar associada a dois números de recibo (o indicado no atributo da factura e o indicado no objecto *Recibo* ao qual a factura estivesse ligada).

Associação **Muitos-para-Muitos**



- Um objecto não pode estar duplamente associado a outro objecto (Joana / Matemática).
- À semelhança das classes (em que os objetos são distintos), as associações também têm que ter ocorrências distintas.
  - **Não há nada que distinga as duas ligações entre *Joana* e *Matemática*;**

Associação **Muitos-para-Muitos**



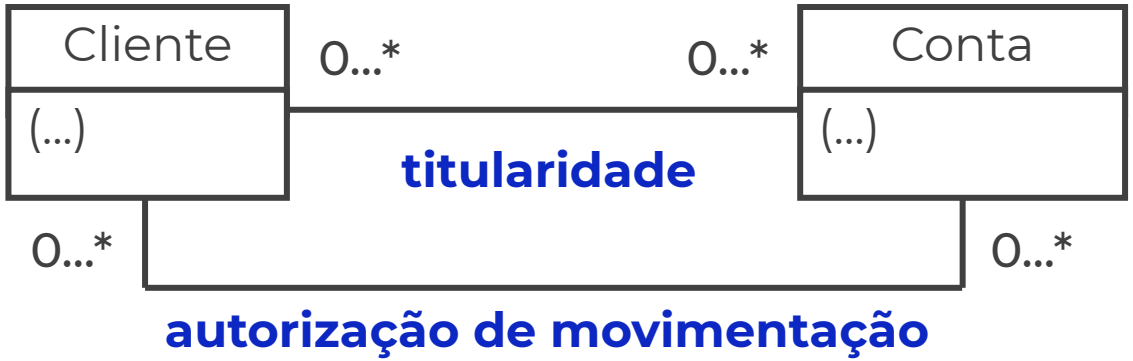
- O sistema de base de dados deverá considerar a 2ª ligação como redundante: Não interessa registar duas vezes que *Joana* e *Matemática* estão ligados
  - **Se interessa, então a associação muitos-para-muitos não é representação correcta.**

Nomes para Associações

- As associações podem (e devem) ter nomes

Normalmente usamos substantivos

- Especialmente relevante quando há mais do que uma associação entre duas classes



0...* ou 1...*

- Usar o 1...* quando, **de todo**, não se quer a informação do lado *1* sem a informação do lado *muitos*.

    - Exemplo 1:
        - Um curso tem pelo menos uma disciplina (portanto, na realidade: 1...*).
        - Mas podemos querer manter informação acerca do curso sem ainda termos indicado as suas disciplinas, **então usamos: 0...***

0...* ou 1...*

- Exemplo 2:
  - Uma receita médica prescreve um ou mais medicamentos.
  - O médico não deve poder guardar a receita no sistema sem ter indicado um dos medicamentos.
  - Sem pelo menos um medicamento, a informação sobre a receita não tem qualquer utilidade, **então usamos: 1...*.**

Resumindo,
Cuidado com o 1...* porque **pode bloquear a introdução de informação provisória** que pode ser útil (curso ainda sem disciplinas).

Estamos agora prontos para desenhar a primeira base de dados simples.

Exercício 1

Pretende desenvolver-se um sistema para gerir o *Youth SPA*. O descritivo abaixo descreve as necessidades do sistema.

"Pretendemos desenvolver um sistema que nos permita fazer uma gestão básica do nosso dia-a-dia no SPA. O sistema deve permitir registar os nossos clientes, os tratamentos disponíveis (com os respetivos preços). A informação relevante a guardar dos clientes é o seu nome, a idade e o NIF. O processo de faturação deve estar automatizado e o sistema deve permitir consultar o histórico de faturas de um determinado cliente (com a data de emissão de valor que o cliente pagou)."

Elabore o diagrama de classes da base de dados necessária para suportar o sistema descrito.

Nota 1.

Distinguir **funcionalidade** de **informação**.

Muitas vezes, descritivos de clientes como o do slide anterior misturam funcionalidade com informação.

Na concepção do modelo de dados, é importante distinguir.

No âmbito das bases de dados, apenas nos importa a **informação**, e podemos ignorar todas as funcionalidades, **nunca ignorando a informação de suporte à funcionalidade pedida**.

Nota 2.
Optimizar.

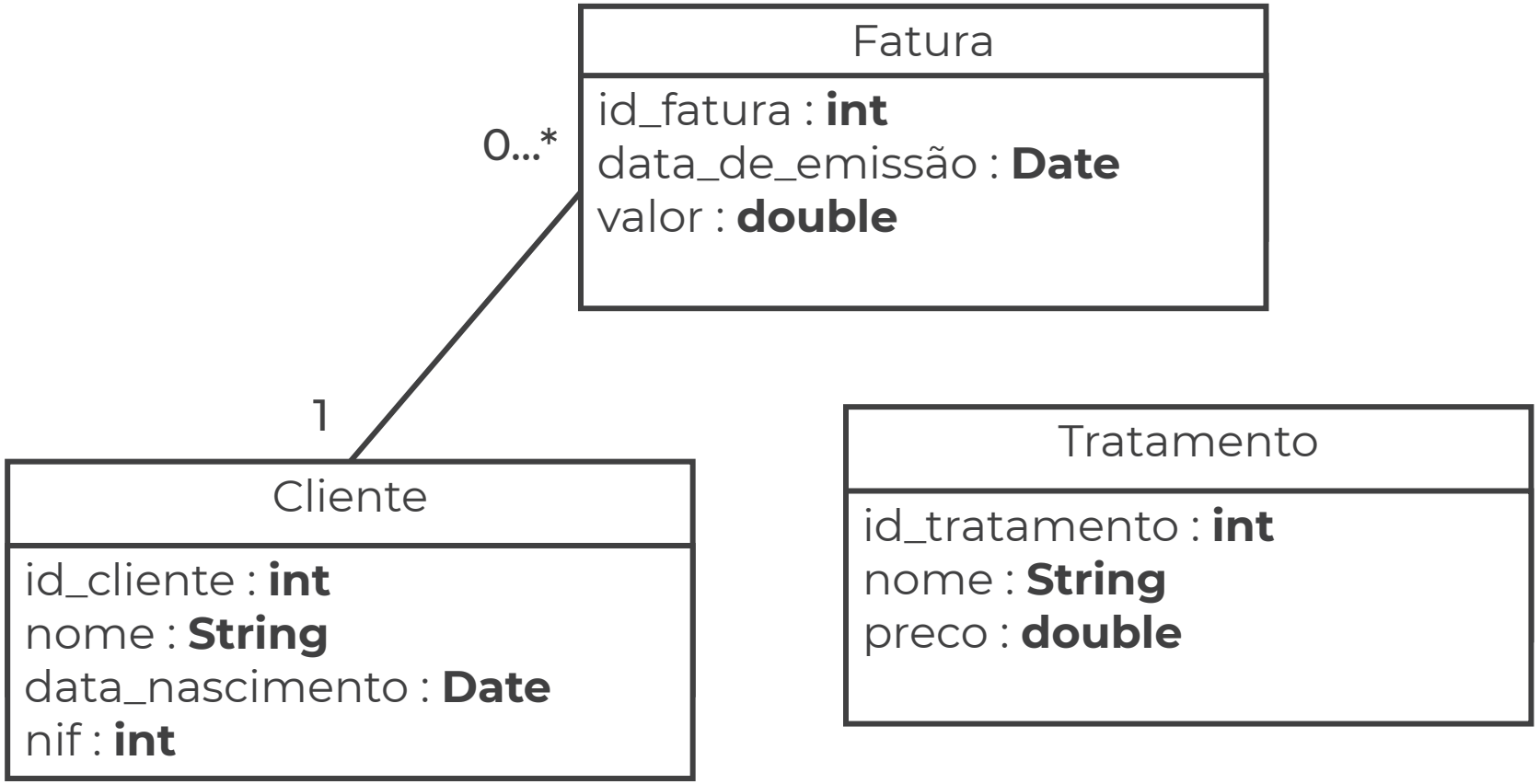
Muitas vezes, descritivos de clientes como o do slide anterior são redigidos de uma forma que apesar de ser a que o cliente descreve, não é necessariamente a que melhor permite definir o sistema.

Um bom exemplo disso no descritivo anterior é "*A informação relevante a guardar dos clientes é o seu nome, a* ***idade*** *e o NIF.*" **Será que devemos guardar a idade? Ou antes a data de nascimento**?

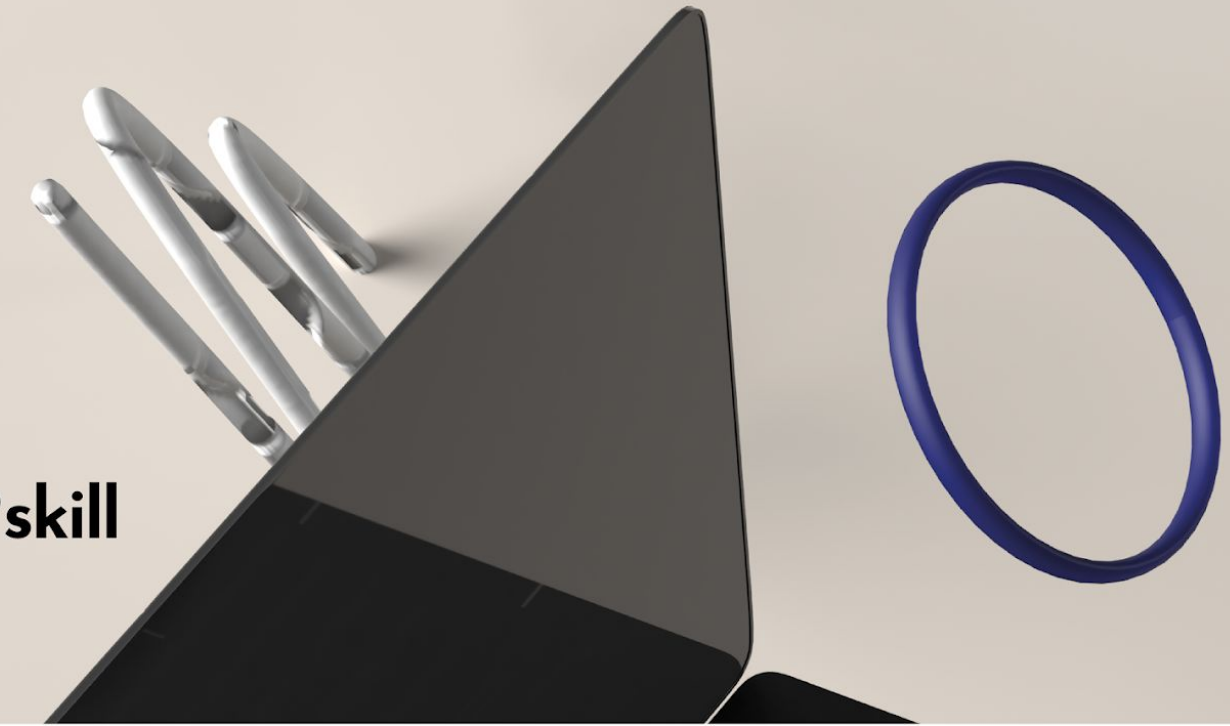
Devemos guardar a data de nascimento, caso contrário a base de dados vai ter de ser alterada anualmente para refletir essa mudança.

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA emprego digital

Exercício 1 - Resolução



O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill