Bases de Dados

Modelo relacional

Gabriel David

gtd@fe.up.pt

Universidade do Porto

Faculdade de Engenharia

FEUP

Sumário

- Modelo relacional
- noções básicas
- tradução do modelo de objectos

Objectivos

- Ted Codd, 1970
  - Fornecer um elevado grau de independência dos dados.
  - Proporcionar uma definição comum dos dados de simplicidade espartana, de forma que uma vasta gama de utilizadores numa empresa possa com ela lidar.
  - Simplificar o potencialmente gigantesco trabalho do administrador da BD.
  - Introduzir uma fundamentação teórica na gestão de BD.
  - Fundir os campos de pesquisa de factos e gestão de ficheiros, preparando a posterior adição de serviços de inferência no mundo comercial.
  - Elevar a programação de aplicações de BD para um novo nível, em que conjuntos sejam tratados como operandos, em vez de serem tratados elemento a elemento.

Sistemas

- Norma ANSI/SPARC
- Alguns sistemas relacionais
  - Rendez-vous
  - Oracle
  - DB2 (IBM)
  - SQLServer (MS)
  - PostgreSQL
  - Sybase
  - MySQL
  - Access (MS)
  - Paradox

Produto cartesiano

- domínios
  - D_1 = { rosa, cravo, tulipa }
  - D_2 = { branco, amarelo, vermelho, negro }
- produto cartesiano
  - $D_1 \times D_2$ = { (r,b), (r,a), (r,v), (r,n), (c,b), (c,a), (c,v), (c,n), (t,b), (t,a), (t,v), (t,n), (r,b) }
  - = { (x, y) | x ∈ D_1, y ∈ D_2 }

D_1 \ D_2	branco	amarelo	vermelho	negro
rosa	(r,b)	(r,a)	(r,v)	(r,n)
cravo	(c,b)	(c,a)	(c,v)	(c,n)
tulipa	(t,b)	(t,a)	(t,v)	(t,n)

Tuplo

- Generalização do produto cartesiano
  - $D_1 \times D_2 \times D_1 = \{ (x, y, z) \mid x \in D_1, y \in D_2, z \in D_1 \}$
  - $= \{$ (rosa, branco, rosa), (rosa, branco, cravo), … $\}$
- tuplo (n-uplo)
  - *sequência* ordenada, eventuais repetições;

  $\forall \neq$ *conjunto*, sem ordem nem repetições
  - tantos elementos quantos os domínios do produto cartesiano
    - (2 - par; 3 - terno; ...)

Relação

- Relação é subconjunto de produto cartesiano
  - $R \subseteq D_1 \times D_2$
  - extensão R = { (r,b), (c,v), (t,n) }
  - compreensão R = { (f, c) | existe no meu jardim a flor f com a cor c }
- Relação é conjunto
  - os tuplos são todos diferentes (existe chave)
  - a ordem dos tuplos na relação é irrelevante
  - a ordem dos componentes nos tuplos é importante (referência por posição)
    - se se atribuir um nome a cada elemento do tuplo (atributo), onde aquele for indicado a ordem não interessa
    - vê-se o tuplo como um mapeamento dos atributos para os respectivos valores
      - se t=(r, b) então flor(t) = r, cor(t) = b

Relação normalizada

- relação não normalizada: contém tuplos cujos valores são conjuntos, i.e., não atómicos
  - não normalizada

$$R = \{ (r, \{b,a\}), (c, \{b,v\}), (t,n)\}$$

  - normalizada correspondente

$$R= \{(r,b), (r,a), (c,b), (c,v), (t,n)\}$$

Tabela

R(Flor, Cor) = { (r,b), (c,v), (t,n) }

tuplo → linha

componente → coluna

R

Flor	Cor
r	b
c	v
t	n

← esquema de R [*plano*]

← instância de R [*conteúdo*]

- **atributo** : nome de uma coluna
- **esquema** : conjunto dos atributos
- **esquema da BD** : contém o conjunto dos esquemas das relações
- **BD** : conjunto das instâncias das relações

Chaves de relações

Um conjunto S de atributos de uma relação R é uma chave se

1. nenhuma instância de R, representando um estado possível do mundo, pode ter dois tuplos que coincidam em todos os atributos de S e não sejam o mesmo tuplo;
2. nenhum subconjunto próprio de S satisfaz (1).

- Ex: chave(R) = {flor}
  - (f, c_1), $(f, c_2) \Rightarrow c_1 = c_2$
  - Significa que só pode haver uma linha para uma dada flor
  - Diz-se que a flor determina a cor
    - Não pode haver duas cores para a mesma flor

Chave como restrição

disciplina
aluno

- Ex: chave(Inscrito) = {coddis, BIa}
  - (d, a, r_1), (d, a, r_2) $\Rightarrow r_1 = r_2$
  - só *coddis* ou só *BIa* não garantem a condição (1)
    - (d, a_1, r_1), (d, a_2, r_2), $r_1 \neq r_2$ podem ser todas verdade
  - {*coddis*, *BIa*, *resultado*} não é chave porque contém uma chave.
- Conceito de chave depende do esquema, não da instância
- Várias chaves numa relação — chaves candidatas
  - BI, NIF, número da segurança social, …
  - Escolhe-se uma delas como chave primária

Tradução do modelo de objectos

Do projecto UML à implementação

- Modelação conceptual usada no projecto
  - UML (modelo de classes)
  - Entidade - Associação
- O modelo de classes contém a maior parte da estrutura declarativa:
  - especificação das classes
  - atributos
  - hierarquia de herança
  - associações
- Implementação em BD relacionais
  - Traduzir de UML para modelo relacional

Relação com as BD

- A versatilidade do paradigma OO permite o projecto não só de sistemas e programas como também de BD, sejam elas hierárquicas, reticuladas, relacionais ou orientadas por objectos
- As BD relacionais são um tipo importante a considerar, devido à sua popularidade
- Embora não tão populares, as BDOO são importantes para diversas áreas de aplicação
  - Projecto
  - Multimédia

Características dos modelos OO

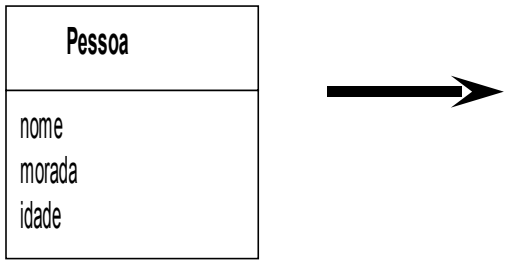
- Tipos de dados abstractos
  - Estrutura de dados + código (operações / métodos)
- Herança
  - Assente em hierarquia
- Identidade dos objectos
  - OID Imutável

Tradução do conceito de OID

- Mapear o conceito de identificador de objecto implícito na noção de objecto num atributo ID
  - Número único, só com significado interno
  - Surrogate, representante, chave artificial
- Utilização de ID's
  - *vantagens*: imutabilidade, independência, estabilidade
  - *desvantagens*: não são verdadeiros OID (conceito não suportado pelos SGBDR); conflituam com a intenção original das BDR de manipular informação com base apenas nos seus valores

Classes

- Cada classe é mapeada para uma tabela
  - tabela tem os mesmos atributos que a classe respectiva, mais o atributo referente ao identificador de objecto
  - decide-se para cada atributo da tabela se poderá ou não ter valores nulos
  - atribui-se a cada atributo um domínio, pré-definido ou definido pelo utilizador

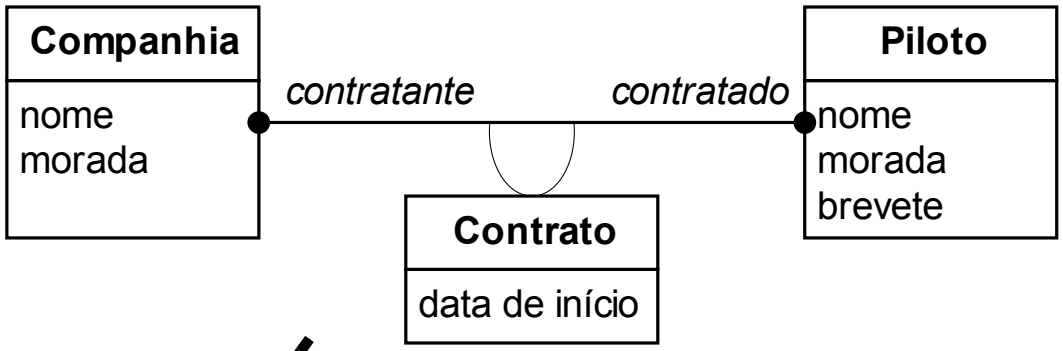


	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Pessoas	nome	N	Nome
	morada	Y	Endereço
	idade	Y	Idade
	Chave primária:	(pessoa-ID)	

Associações

- Em geral, uma associação pode ou não ser mapeada para uma tabela, dependendo:
  - do tipo e multiplicidade da associação
  - das preferências do projectista em termos de extensibilidade, número de tabelas e compromissos de performance

Associações binárias (m - n)



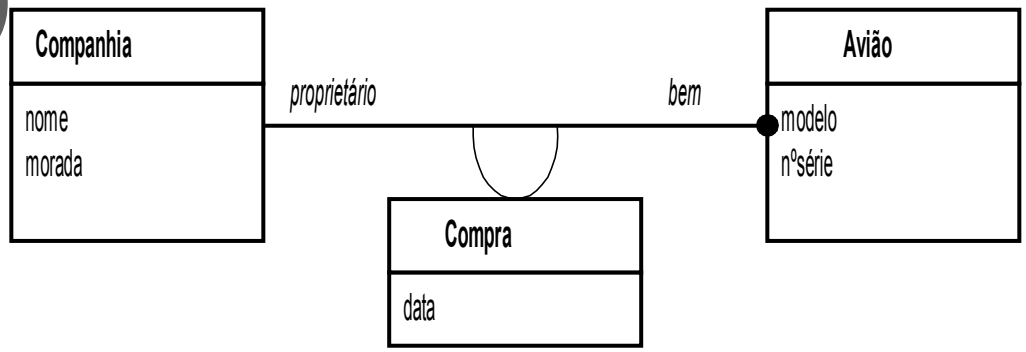
	Atributo	Nulos?	Domínio
Tabela **Contratos**	companhia-ID	N	ID
	piloto-ID	N	ID
	data-de-inicio	Y	Data
	Chave primária:	(companhia-ID, piloto-ID)	

Tabela Contratos, para além das tabelas Companhias e Pilotos.

Associações binárias (m – n)

- Uma associação muitos-para-muitos é sempre mapeada para uma tabela distinta
  - A chave primária será constituída, à partida, pela concatenação das chaves primárias de cada uma das tabelas envolvidas na associação
  - Mas outras chaves podem revelar-se mais interessantes, dependendo da realidade subjacente
    - Exemplo: (piloto-id, data_de_inicio)

Associações binárias (1 - n)

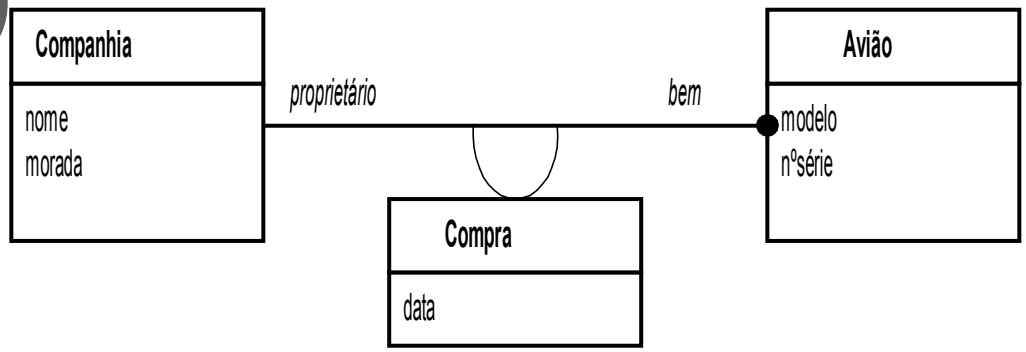


Solução 1:
criação de 1 tabela distinta

	Atributo	Nulos?	Domínio
Tabela	companhia-ID	N	ID
Compras	avião-ID	N	ID
	data	Y	Data
	Chave primária:	(avião-ID)	

Criação da tabela Compras, para além das tabelas Companhias e Aviões.

Associações binárias (1 - n)



Solução 2:
não criação de 1 tabela distinta

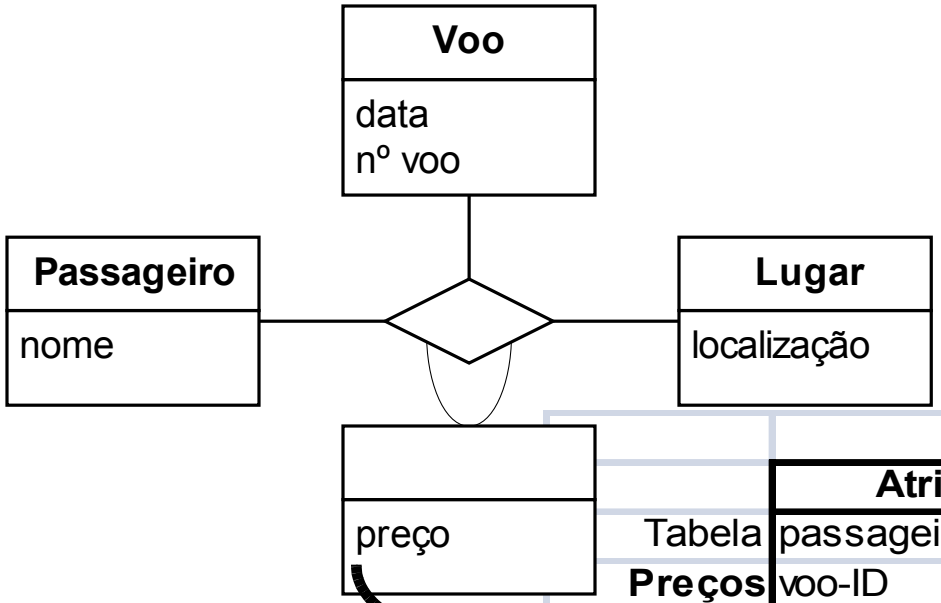
	Atributo	Nulos?	Domínio
Tabela	avião-ID	N	ID
Aviões	modelo	Y	Text
	no_serie	Y	Text
	companhia-ID	Y	ID
	data	Y	Data
	Chave primária:	(avião-ID)	

Só se criam as tabelas Companhias e Aviões, mas esta com duas colunas extra: a chave de Companhias (tabela do lado 1) e o atributo da associação

Associações binárias (1 - n)

- Uma associação um-para-muitos pode ser mapeada de 2 formas
  - criação de uma tabela distinta para a associação
  - adicionar uma chave-externa na tabela relativa a muitos
- Vantagens de não criar uma tabela distinta
  - menos tabelas no esquema final
  - maior performance devido a um menor número de tabelas a navegar
- Desvantagens de não criar uma tabela distinta
  - menos rigor em termos de projecto do esquema
  - extensibilidade reduzida
  - maior complexidade

Associações ternárias



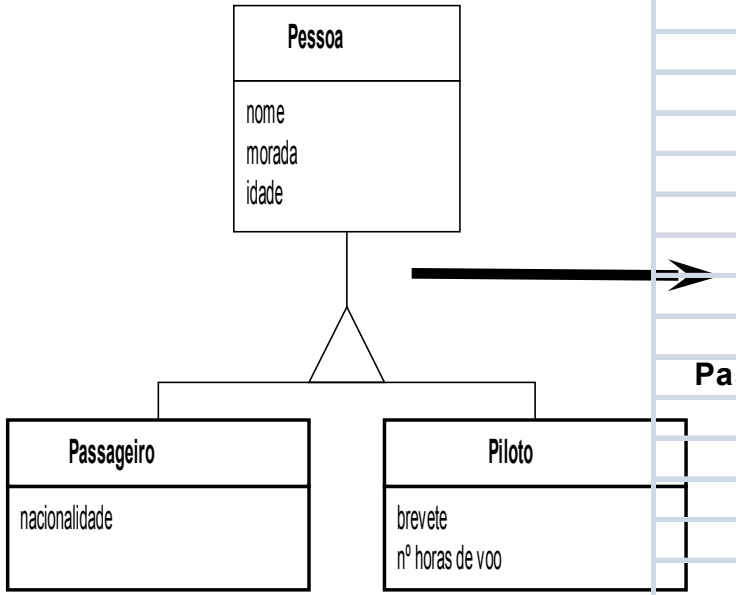
	Atributo	Nulos?	Domínio
Tabela	passageiro-ID	N	ID
Preços	voo-ID	N	ID
	lugar-ID	N	ID
	preço	Y	Montante
	Chave primária:	(passageiro-ID, voo-ID, lugar-ID)	

Associação n-árias dão sempre origem a tabela distinta.
Chave aqui poderia ser (voo-ID, lugar-ID)

Mapeamento de Generalizações

- Há três maneiras diferentes (ou mais...) de mapear uma hierarquia
  - A escolha depende das características específicas

Generalizações - 1



	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Pessoas	nome	N	Nome
	morada	Y	Endereço
	idade	Y	Idade
	tipo-de-pessoa	N	Tipo-Pessoa
	Chave primária:	(pessoa-ID)	

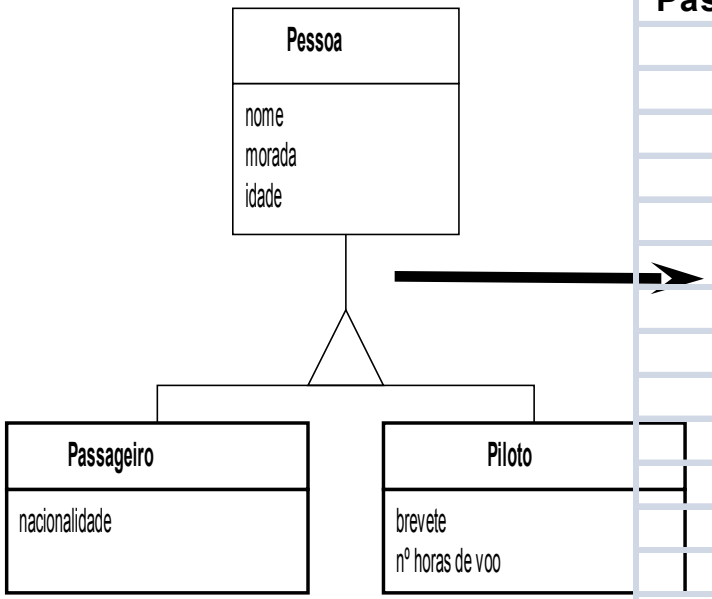
	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Passageiros	nacionalidade	Y	País
	Chave primária:	(pessoa-ID)	

	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Pilotos	brevete	Y	Text
	horas_de_voo	Y	Inteiro
	Chave primária:	(pessoa-ID)	

Generalizações 1

- Tanto a superclasse como cada uma das subclasses são mapeadas para tabelas distintas
  - abordagem logicamente correcta e extensível
  - envolve várias tabelas, pelo que a navegação da superclasse para as subclasses pode tornar-se lenta

Generalizações - 2



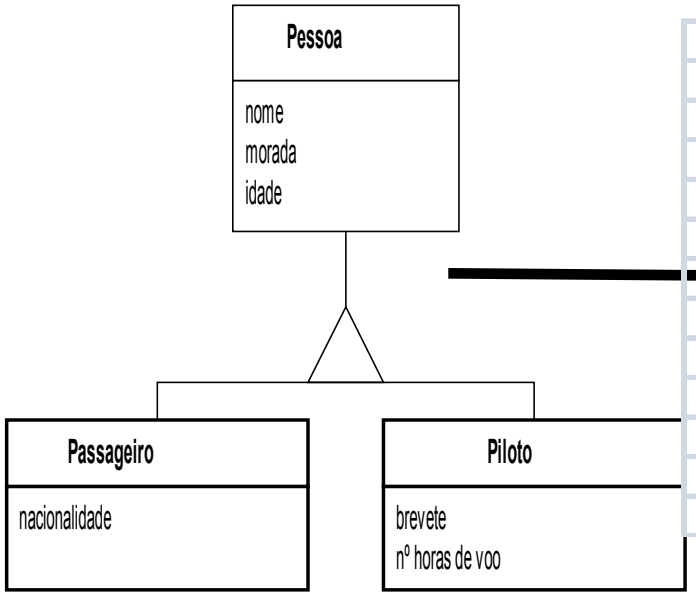
	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Passageiros	nome	N	Nome
	morada	Y	Endereço
	idade	Y	Idade
	nacionalidade	Y	País
	Chave primária:	(pessoa-ID)	

	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Pilotos	nome	N	Nome
	morada	Y	Endereço
	idade	Y	Idade
	brevete	Y	Text
	horas_de_voo	Y	Inteiro
	Chave primária:	(pessoa-ID)	

Generalizações 2

- Eliminação da navegação da superclasse-para-subclasse
  - elimina a tabela da superclasse e replica todos os atributos dela em cada subclasse
  - ideal, quando a superclasse possui poucos atributos e as subclasses muitos atributos
  - não se pode garantir a unicidade dos valores dos atributos da superclasse

Generalizações - 3



	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Pessoas	nome	N	Nome
	morada	Y	Endereço
	idade	Y	Idade
	tipo-de-pessoa	N	Tipo-Pessoa
	nacionalidade	Y	País
	brevete	Y	Text
	horas_de_voo	Y	Inteiro
	Chave primária:	(pessoa-ID)	

Generalizações 3

- Uma tabela para a superclasse
  - criação de apenas uma tabela para a superclasse
  - todos os atributos das subclasses são acrescentados à tabela da superclasse
  - cada subclasse utiliza apenas os atributos que lhe pertencem, deixando os restantes com valores nulos
  - acrescenta-se uma coluna Tipo para indicar a que subclasse pertence cada registo
  - não suporta atributos obrigatórios nas subclasses
  - abordagem interessante quando em presença de poucas subclasses (2 ou 3)

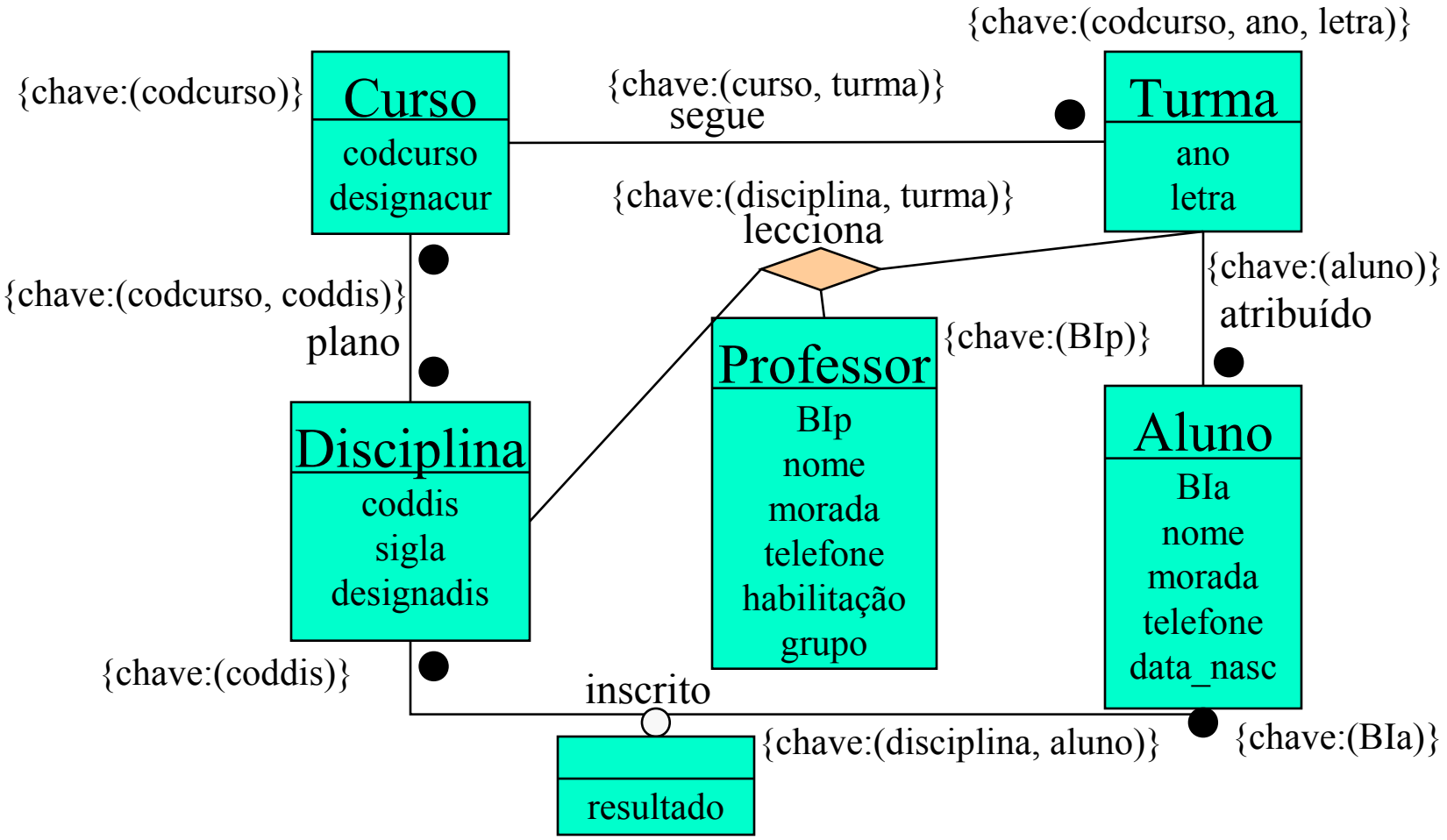
Bases de Dados Relacionais

- O mapeamento para BD Relacionais não é único
  - existem duas formas de mapear uma associação
  - existem três formas de mapear uma generalização
  - é necessário adicionar detalhes que não existem no modelo de objectos, tais como, chaves primárias e chaves candidatas, se um atributo pode ter valores nulos ou não
  - obriga à atribuição de um domínio a cada atributo

Tradução UML - BDR

- Resumindo
  - criar um atributo *id* correspondente a cada classe
  - *classe* implementada por *relação*
  - *associação* muitos-para-muitos ou ternária implementada por *relação*
  - *associação* muitos-para-um implementada por *atributo extra* (*id* do lado 1) na relação do lado muitos
  - traduzir hierarquias de uma das três formas apresentadas

Exemplo dos cursos



Tradução

Relações obtidas automaticamente

1 Cursos(curso-ID, *codcurso*, designacur)

2 Disciplinas(disciplina-ID, *coddis*, sigla, designadis)

3 Turmas(turma-ID, *curso-ID, ano, letra*)

4 Professores(prof-ID, *bip*, nome, morada, telefone, habilitação, grupo)

5 Alunos(aluno-ID, *bia*, nome, morada, telefone, data_nasc, turma-ID)

6 Planos(curso-ID, disciplina-ID)

7 Inscritos(disciplina-ID, aluno-ID, resultado)

8 Lecciona(prof-ID, disciplina-ID, turma-ID)

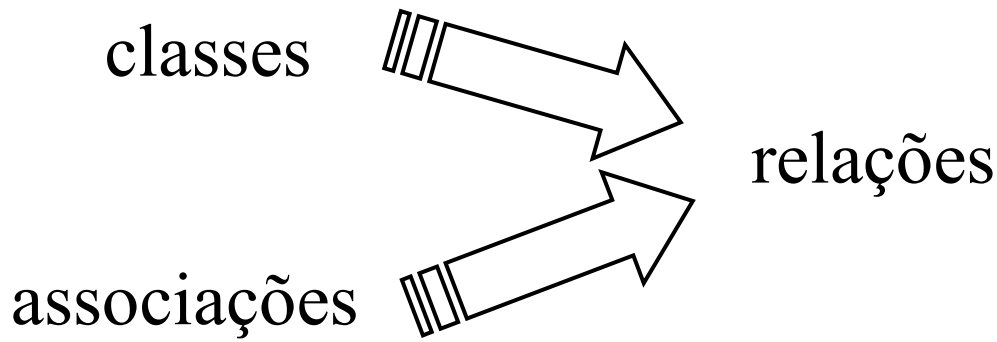
Em *itálico,* chaves alternativas

Caso relacional

Só um conceito, maior uniformidade, menos operadores.

UML

relacional



Chaves naturais

- Simplificação: quando existir uma chave alternativa, o ID pode ser substituído por esta, em todas as ocorrências, e ignorado
- é preferível, no modelo relacional, usar chaves naturais em vez de artificiais

1 Cursos(codcurso, designacur)

2 Disciplinas(coddis, sigla, designadis)

3 Turmas(codcurso, ano, letra)

4 Professores(bip, nome, morada, telefone, habilitação, grupo)

5 Alunos(bia, nome, morada, telefone, data_nasc, codcurso, ano, letra)

6 Planos(codcurso, coddis)

7 Inscritos(coddis, bia, resultado)

8 Lecciona(bip, coddis, codcurso, ano, letra)

Conteúdo das chaves

- Quando os esquemas das relações são uma tradução do modelo de objectos:
  - relação vinda de uma **classe**: a chave de utilizador da classe é a chave da relação (*Bla* em *Aluno*)
  - relação vinda de uma **associação muitos-para-muitos**: chave tem, frequentemente, todos os atributos (*Plano*)
  - associação muitos-para-um de E_1, ..., E_{k-1} em E_k: as chaves de E_1, ..., E_{k-1} formam uma chave (*Lecciona*)
- Quando os atributos vêm de associações
  - **associação muitos-para-um** de E para F: o atributo em E é uma chave externa para F (*Atribuído*)
  - **associação um-para-um** entre E e F: tanto E como F podem conter a chave chave externa para a outra relação

Associação característica

- A classe D tem **chave emprestada** se, para definir a sua chave de utilizador, for necessário recorrer à chave de outra classe C que seja o lado um de uma associação de que D é o lado muitos
  - caso da turma num curso: turma 2B do MIEIC e 2B do MIEIG
- a existência de objectos em D depende da existência de objectos em C, enquanto que os objectos em C existem independentemente de D
- D corresponde a uma **entidade fraca;** entre D e C existe uma **associação característica**
  - exemplo: classes de Facturas e de Linhas das facturas

Elementos activos na BD

- **Esquema da BD**

  = esquemas das tabelas +
  restrições de integridade

- Restrição
  - função booleana cujo valor têm que ser verdadeiro
  - uma alteração que conduza a um estado que viole uma restrição é rejeitada pelo SGBD

- Elementos **activos** complementam os estruturais
  - Correspondem a rotinas invocadas automaticamente

Restrições de integridade

- **Chave primária** - identifica uma entidade; não pode haver duas entidades com a mesma chave; não pode ser nula
- **Valor único** - impede a repetição da mesma combinação de valores nos atributos envolvidos (**chave alternativa**)
- **Valor não nulo** - preenchimento obrigatório
- **Integridade referencial** - exige que um valor referido por uma entidade de facto exista na BD (**chave externa**)
- **Domínio** - exige que o valor pertença a um determinado conjunto ou gama
- **Genérica** - asserção que tem que ser verdadeira

Tuplos pendentes e valor nulo

- **Tuplo pendente** - tuplo numa tabela que não tem par na outra, ao combinar duas relações
- Forma de reduzir o problema
  1) Restrições de integridade referencial
     - se o valor V aparece em A de R_1, também tem que existir em B de R_2, sendo B chave de R_2; A é **chave externa**
     - atributo coddis em Inscritos só deve admitir valores que existam na chave de Disciplinas
  2) Nas linhas em que a ligação não exista, colocar **valor nulo** nos atributos desconhecidos (⊥)
     - atributos da chave externa (codcurso, ano, letra) devem estar a nulo nos alunos em que não se conheça a turma
     - valores nulos não podem aparecer na chave primária
     - dois valores nulos são sempre diferentes (não são 0 ou "")

Indicação de chaves externas

- Esquema relacional de Cursos
  - **1** Cursos(codcurso, designacur)
  - **2** Disciplinas(coddis, sigla, designadis)
  - **3** Turmas(codcurso➔Cursos, ano, letra)
  - **4** Professores(bip, nome, morada, telefone, habilitação, grupo)
  - **5** Alunos(bia, nome, morada, telefone, data_nasc, [codcurso, ano, letra]➔Turmas)
  - **6** Planos(codcurso➔Cursos, coddis➔Disciplinas)
  - **7** Inscritos(coddis➔Disciplinas, bia➔Alunos, resultado)
  - **8** Lecciona(bip ➔Professores, coddis➔Disciplinas, [codcurso, ano, letra]➔Turmas)

Restrições genéricas

- Trigger
  - código armazenado na BD que espera por um evento (inserção, apagamento, …)
  - SGBD executa as acções no código disparado pelo evento
  - usado para verificar regras de integridade suplementares
- Vantagem relativamente a procedimento no código da aplicação
  - é executado mesmo para alterações directas na BD, ou via qualquer aplicação
  - funciona como uma extensão às regras de integridade mantidas automaticamente pelo SGBD