

# Modelo relacional

Fernando Lobo

Base de Dados, Universidade do Algarve

# Modelo relacional

- Modelo de dados mais importante em bases de dados.
- Modelo simples baseado no conceito matemático de relação.
- Inventado em 1970 por Edgar Codd.
- Utilizado pela maioria dos SGBDs.
- SQL é baseado no modelo relacional.

# Modelo relacional

- Uma relação é como se fosse uma tabela com linhas e colunas.
- A diferença é que:
  - ▶ uma relação não pode ter linhas repetidas.
- Muitas vezes irei usar o termo tabela para me referir a uma relação (mas devem estar conscientes de que tabela e relação são coisas distintas).
- BD = conjunto de relações (ou conjunto de tabelas).

# O que é um modelo de dados?

- É uma notação para descrever dados ou informação.
- Consiste em 3 coisas:
  - ▶ Estrutura dos dados
  - ▶ Operações sobre os dados
  - ▶ Restrições sobre os dados

# Modelo relacional de dados

- É baseado em tabelas.
- Exemplo:

| nome               | ano  | duração | aCores |
|--------------------|------|---------|--------|
| Gone With the Wind | 1939 | 231     | true   |
| Star Wars          | 1977 | 124     | true   |
| Wayne's World      | 1992 | 95      | true   |

- Atribuimos um nome à tabela, por exemplo *Filmes*

# Algumas noções importantes

- Atributo
- Domínio
- Esquema
- Tuplo
- Chave primária

# Atributo

| nome               | ano  | duração | aCores |
|--------------------|------|---------|--------|
| Gone With the Wind | 1939 | 231     | true   |
| Star Wars          | 1977 | 124     | true   |
| Wayne's World      | 1992 | 95      | true   |

- Os atributos de uma tabela são os nomes das colunas.
- No exemplo, a tabela *Filmes* tem 4 atributos:
  - ▶ *nome, ano, duração, aCores*

# Domínio

- O domínio de um atributo é um tipo de dados associado a esse atributo.
  - ▶ inteiro, real, booleano, string, etc.
- No exemplo anterior:
  - ▶ nome: string
  - ▶ ano: inteiro
  - ▶ duração: inteiro
  - ▶ aCores: booleano

# Esquema

- O esquema de uma tabela é definido pelo nome da tabela e respectivos atributos.
- É costume usar a notação: nome-da-tabela(atrib<sub>1</sub>, ..., atrib<sub>n</sub>)
- No nosso exemplo...
  - ▶ Filmes(nome, ano, duração, aCores)
- Podemos especificar o domínio de cada atributo
  - ▶ Filmes(nome:string, ano:integer, duração:integer, aCores:boolean)

# Tuplo

- Um tuplo é uma generalização do conceito de par, triplo, ...
- Um tuplo corresponde a uma linha da tabela.
- No nosso exemplo a tabela tem 3 tuplos. Estes:
  - ▶ (Gone With the Wind, 1939, 231, true)
  - ▶ (Star Wars, 1977, 124, true)
  - ▶ (Wayne's World, 1992, 95, true)

| nome               | ano  | duração | aCores |
|--------------------|------|---------|--------|
| Gone With the Wind | 1939 | 231     | true   |
| Star Wars          | 1977 | 124     | true   |
| Wayne's World      | 1992 | 95      | true   |

# Chave

- É possível impor restrições à informação contida numa tabela.
- Uma dessas restrições é imposta pelo conceito de *chave* (em inglês *key*).
- Um conjunto de atributos de uma tabela é uma chave se e só se essa tabela não puder ter valores idênticos na totalidade desses atributos.

# Chave

- Nota importante: o conceito de chave é independente do conteúdo da tabela num determinado momento.
- Ou seja, o conceito de chave aplica-se qualquer que seja a instância da tabela.
- Pergunta: O que poderia ser uma chave da tabela Filmes?

# Tabela versus Relação

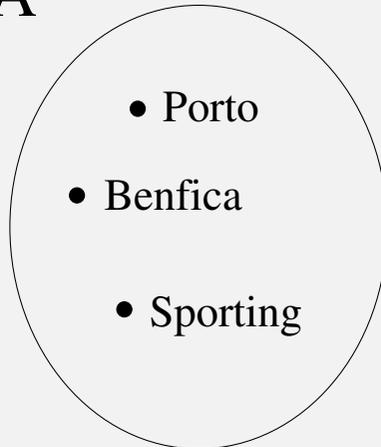
- São praticamente a mesma coisa
- A diferença entre uma coisa e outra é sutil

# Definição formal de relação

- Dados  $n$  conjuntos  $C_1, C_2, \dots, C_n$  (não necessariamente distintos),  $R$  é uma **relação** sobre estes  $n$  conjuntos, se for um conjunto de tuplos da forma  $(v_1, v_2, \dots, v_n)$  em que:
  - ▶  $v_1 \in C_1$
  - ▶  $v_2 \in C_2$
  - ▶ ...
  - ▶  $v_n \in C_n$
- Por outras palavras,  $R$  é subconjunto do Produto Cartesiano  $C_1 \times C_2 \times \dots \times C_n$ .

# Produto Cartesiano. Exemplo:

A



B



$A \times B$

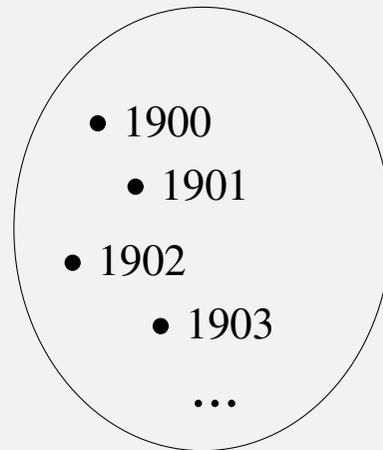


# Outro exemplo

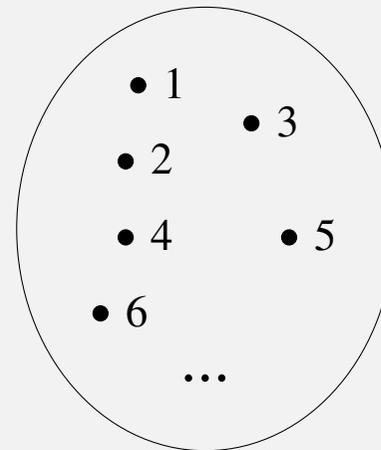
$C_{\text{nomes}}$



$C_{\text{anos}}$



$C_{\text{duracao}}$



$C_{\text{aCores}}$



- NOTA: Conjuntos podem ser infinitos.

## Outro exemplo (cont.)

$$C_{\text{nomes}} \times C_{\text{anos}} \times C_{\text{duracao}} \times C_{\text{aCores}}$$

- (Star Wars, 1900, 1, sim)
- (Star Wars, 1900, 1, nao)
- (Star Wars, 1900, 2, sim)
- (Star Wars, 1900, 2, nao)
- (Star Wars, 1900, 3, sim)

...

- Produto cartesiano dá o conjunto de todos os possíveis tuplos  $(v_1, v_2, v_3, v_4)$ , em que  $v_1 \in C_{\text{nomes}}$ ,  $v_2 \in C_{\text{anos}}$ ,  $v_3 \in C_{\text{duracao}}$ ,  $v_4 \in C_{\text{aCores}}$ .

# Uma possível relação sobre $C_{nomes}$ , $C_{anos}$ , $C_{duracao}$ , $C_{aCores}$

- (Star Wars, 1977, 124, sim)
- (King Kong, 1930, 120, nao)
- (Moulin Rouge, 2001, 124, sim)

## Dá jeito visualizar a relação como uma tabela

| nome         | ano  | duração | aCores |
|--------------|------|---------|--------|
| Star Wars    | 1977 | 124     | sim    |
| King Kong    | 1930 | 120     | não    |
| Moulin Rouge | 2001 | 124     | sim    |

- Nome das colunas são atributos.
- tuplo = linha.

Estas 2 tabelas representam a mesma relação. Porquê?

| nome      | ano  | duração | aCores |
|-----------|------|---------|--------|
| Lion King | 1997 | 108     | sim    |
| Star Wars | 1977 | 124     | sim    |
| King Kong | 1930 | 120     | não    |
| Lion King | 1997 | 108     | sim    |
| Lion King | 1997 | 108     | sim    |

| ano  | nome      | aCores | duração |
|------|-----------|--------|---------|
| 1930 | King Kong | não    | 120     |
| 1977 | Star Wars | sim    | 124     |
| 1997 | Lion King | sim    | 108     |

# Esquema da BD = descrição completa da estrutura da BD

- Nome das relações/tabelas
- Nome dos atributos
- Domínio dos atributos (integer, string, ...)
- Chaves
- ...

# Instância da BD

- O conteúdo (os tuplos) de todas as relações de uma BD constitui uma instância da BD.
- A instância da BD muda com frequência.
- O esquema da BD é feito uma vez (por pessoas como vocês) e não costuma mudar.

# Definição de tabelas em SQL

```
CREATE TABLE <nome> (  
    <lista de elementos>  
);
```

- cada elemento consiste num atributo e no respectivo tipo de dados.
- os tipos mais comuns são:
  - ▶ INT ou INTEGER
  - ▶ REAL ou FLOAT
  - ▶ CHAR(n), VARCHAR(n)
  - ▶ BOOLEAN
  - ▶ DATE
  - ▶ TIME

## Exemplo: Tabela de actores

```
CREATE TABLE Actores(  
    nome            VARCHAR(50),  
    morada          VARCHAR(70),  
    sexo            CHAR,  
    dataNascimento DATE  
);
```

# DATE e TIME

- o formato de DATE é 'yyyy-mm-dd'.
- o formato de TIME é 'hh:mm:ss'.
  - ▶ poderá ter ainda fracções de segundo.
- Exemplos:
  - ▶ DATE '2008-01-05'
  - ▶ TIME '15:32:04.5'

# Declaração da chave

- um atributo ou lista de atributos pode ser declarado como chave em SQL utilizando **PRIMARY KEY**.
- Exemplo:

```
CREATE TABLE Actores(  
    nome            VARCHAR(50) PRIMARY KEY,  
    morada          VARCHAR(70),  
    sexo            CHAR,  
    dataNascimento DATE  
);
```

## Declaração da chave (cont.)

- No caso da chave ser composta, temos de especificar um elemento à parte.
- Exemplo:

```
CREATE TABLE Filmes(  
    nome            VARCHAR(50),  
    ano             INTEGER,  
    duracao         INTEGER,  
    aCores          BOOLEAN,  
    PRIMARY KEY (nome, ano)  
);
```

# Do mundo real para o modelo relacional

- Dado um problema real, como representá-lo de acordo com o modelo relacional?
- Teremos de mapear a informação que queremos armazenar e manipular num conjunto de tabelas.
- Mais adiante iremos aprender a fazer uma modelação conceptual usando uma notação chamada UML (Unified Modeling Language)
  - ▶ Depois é possível converter essa notação para o modelo relacional
- ... mas por agora vamos usar o modelo relacional directamente.

# Exemplo: BD para os serviços académicos da UAlg

- Fazer uma BD para os SA da UAlg.
- Informação sobre alunos, disciplinas a que estão inscritos, curso a que pertencem, etc.
- Como representar essa informação na forma de tabela(s)?

# 1ª abordagem: uma tabela com “tudo ao molho”

| nº    | nome         | morada | dataNasc | curso | disciplinas  |
|-------|--------------|--------|----------|-------|--|
| 77888 | José Antunes | ...    | ...      | LEI   | Programação Imperativa, Bases de Dados, Análise Numérica |
| 77891 | Maria Silva  | ...    | ...      | LEI   | Empreendedorismo, Bases de Dados, Física II              |
| 77892 | Teresa Sousa | ...    | ...      | LEI   | Bases de Dados, Algoritmos e Estruturas de Dados         |
| 77993 | Pedro Lima   | ...    | ...      | LA    | Fruticultura, Agricultura Biológica                      |

- Isto é uma boa abordagem?

## 2ª abordagem

- Manter uma só tabela, mas ter uma coluna (booleana) para cada disciplina possível

| nº    | nome         | morada | dataNasc | curso | Programação Imperativa | Bases de Dados | ... |
|-------|--------------|--------|----------|-------|------------------------|----------------|-----|
| 77888 | José Antunes | ...    | ...      | LEI   | true                   | true           | ... |
| 77891 | Maria Silva  | ...    | ...      | LEI   | false                  | true           | ... |
| 77892 | Teresa Sousa | ...    | ...      | LEI   | false                  | true           | ... |
| 77993 | Pedro Lima   | ...    | ...      | LA    | false                  | false          | ... |

## 3ª abordagem

- Manter várias tabelas: Alunos, Cursos, Disciplinas

- Alunos

| nº    | nome         | morada | dataNasc | curso |
|-------|--------------|--------|----------|-------|
| 77888 | José Antunes | ...    | ...      | LEI   |
| 77891 | Maria Silva  | ...    | ...      | LEI   |
| 77892 | Teresa Sousa | ...    | ...      | LEI   |
| 77993 | Pedro Lima   | ...    | ...      | LA    |

- Cursos

nome  
LEI  
LA

- Disciplinas

nome  
Programação Imperativa  
Bases de Dados  
Análise Numérica  
Empreendedorismo  
Física II  
Algoritmos e Estruturas de Dados  
Fruticultura  
Agricultura Biológica

## 3ª abordagem (cont.)

- Conseguimos saber o curso de cada aluno
- Mas não sabemos quais as disciplinas a que cada aluno está inscrito
- Como podemos resolver isto?

## 3ª abordagem (cont.)

- Basta acrescentar mais uma tabela, que indica quais as disciplinas a que cada aluno está inscrito.
- A tabela necessita de ter 2 atributos:
  - ▶ Um para identificar o aluno, outro para identificar a disciplina
  - ▶ Um tuplo  $(x, y)$  terá o significado que o aluno  $x$  está inscrito à disciplina  $y$
- AlunoDisciplina

| aluno | disciplina                       |
|-------|----------------------------------|
| 77888 | Programação Imperativa           |
| 77888 | Bases de Dados                   |
| 77888 | Análise Numérica                 |
| 77891 | Empreendedorismo                 |
| 77891 | Bases de Dados                   |
| 77891 | Física II                        |
| 77892 | Bases de Dados                   |
| 77892 | Algoritmos e Estruturas de Dados |
| 77993 | Fruticultura                     |
| 77993 | Agricultura Biológica            |

## 3ª abordagem (cont.)

- Podemos poupar espaço atribuindo um código único a cada disciplina
- Disciplinas

| código | nome                             |
|--------|----------------------------------|
| 1      | Programação Imperativa           |
| 2      | Bases de Dados                   |
| 3      | Análise Numérica                 |
| 4      | Empreendedorismo                 |
| 5      | Física II                        |
| 6      | Algoritmos e Estruturas de Dados |
| 7      | Fruticultura                     |
| 8      | Agricultura Biológica            |

- AlunoDisciplina

| aluno | disciplina |
|-------|------------|
| 77888 | 1          |
| 77888 | 2          |
| 77888 | 3          |
| 77891 | 4          |
| 77891 | 2          |
| 77891 | 5          |
| 77892 | 2          |
| 77892 | 6          |
| 77993 | 7          |
| 77993 | 8          |

# Chave estrangeira

- Os atributos *aluno* e *disciplina* na tabela AlunoDisciplina funcionam como uma espécie de apontadores
- No modelo relacional (e também em SQL) podemos especificar que esta espécie de apontador tem de apontar para um valor válido.
  - ▶ Isto é, se  $(x, y)$  for um tuplo de tabela AlunoDisciplina, podemos impor a restrição de que  $x$  tem de forçosamente estar presente na tabela Aluno, e que  $y$  tem de forçosamente estar presente na tabela Disciplina.
    - ★ Dizemos que o atributo aluno é uma **chave estrangeira** que faz referência à tabela Aluno
    - ★ e disciplina também é uma **chave estrangeira** que faz referência à tabela Disciplina.

# Resumindo

- A nossa tabela “tudo ao molho” foi substituída por 4 tabelas:
  - ▶ Alunos
  - ▶ Cursos
  - ▶ Disciplinas
  - ▶ AlunoDisciplina
- As 3 primeiras dizem respeito a conceitos/entidades do nosso problema concreto
- A tabela AlunoDisciplina permite associar alunos às disciplinas a que estão inscritos