

## Valores NULL

- Valor desconhecido ou inaplicável.
  - Ex 1: Não se conhece a duração de um filme.
  - Ex 2: Valor do atributo mulher/marido de uma pessoa solteira.
- Dá origem a um comportamento estranho em SQL.
- Lógica a 3 valores: True, False, Unknown.

## Operações envolvendo valores NULL

Se  $x$  tiver o valor NULL, então:

1.  $x + \text{qqcoisa} = \text{NULL}$   
(a mesma coisa para  $-$ ,  $*$ ,  $/$ ).
2. Quando comparamos  $x$  com outro valor, o resultado é Unknown.

## Lógica com 3 valores

- Como é que funcionam os operadores lógicos AND, OR, e NOT, numa lógica com 3 valores?
- True=1, False=0, Unknown= $\frac{1}{2}$
- $a$  AND  $b = \min(a, b)$   
 $a$  OR  $b = \max(a, b)$   
NOT  $a = 1 - a$
- Exemplo:  
True AND (False OR NOT(Unknown))  
=  $\min(1, \max(0, (1 - \frac{1}{2})))$   
=  $\min(1, \max(0, \frac{1}{2}))$   
=  $\min(1, \frac{1}{2})$   
=  $\frac{1}{2}$   
= Unknown

## Lógica com 3 valores (cont.)

nome	ano	duracao
Star Wars	1977	124
King Kong	1933	NULL
Return of the Jedi	1983	165

```
SELECT *  
FROM Filmes  
WHERE duracao<100 OR duracao>=100;
```

- “King Kong” não aparece no output.
- $NULL < 100 = \text{Unknown}$ .  
 $NULL > 100 = \text{Unknown}$ .  
 $\text{Unknown OR Unknown} = \text{Unknown}$ .
- Para o output só vão os tuplos para os quais a cláusula WHERE tem o valor True.

## Lógica com 3 valores (cont.)

Para verificar se um atributo  $x$  tem ou não o valor NULL, deve utilizar-se:  $x$  IS NULL (ou  $x$  IS NOT NULL).

```
-- Errado
SELECT *
FROM Filmes
WHERE duracao=NULL;

-- Correcto
SELECT *
FROM Filmes
WHERE duracao IS NULL;
```

## Joins explícitos

- podemos fazer joins usando a forma `SELECT-FROM-WHERE`.
- mas também podemos utilizar expressões de `JOIN` explícitas.
- essas expressões aparecem na cláusula `FROM`.

## Produto cartesiano

Em SQL: CROSS JOIN

```
SELECT *  
FROM Filmes, Estudios;
```

é equivalente a:

```
SELECT *  
FROM Filmes CROSS JOIN Estudios;
```

## Theta Join

Em SQL: JOIN ... ON

```
SELECT *  
FROM Filmes, Participa  
WHERE nome = nomeFilme  
      AND ano = anoFilme;
```

é equivalente a:

```
SELECT *  
FROM Filmes JOIN Participa ON  
      nome = nomeFilme AND ano = anoFilme;
```

## Natural Join

Em SQL: NATURAL JOIN

```
SELECT Actores.*, categoria
FROM Actores, Realizadores
WHERE Actores.nome = Realizadores.nome
```

é equivalente a:

```
SELECT *
FROM Actores NATURAL JOIN Realizadores;
```

O que significa a query em português corrente?

## Outerjoins

- adiciona ao output os tuplos que não conseguem fazer “join” .
- esses tuplos ficam com NULL nos restantes atributos.
- em SQL: OUTER JOIN.
- existem variantes do OUTER JOIN:
  - LEFT: os tuplos da 1<sup>a</sup> tabela vão para o output.
  - RIGHT: os tuplos da 2<sup>a</sup> tabela vão para o output.
  - FULL: os tuplos de ambas as tabelas vão para o output.

## Exemplos

```
--  
-- actores que não são realizadores aparecem  
-- com categoria NULL. Realizadores que não  
-- são actores aparecem com morada, sexo, e  
-- data de nascimento a NULL.
```

```
--
```

```
SELECT *  
FROM Actores NATURAL FULL OUTER JOIN Realizadores;
```

```
--
```

```
-- filmes sem actores aparecem no output  
-- actores sem filmes também.
```

```
--
```

```
SELECT *  
FROM Filmes FULL OUTER JOIN Participa ON  
    nome = nomeFilme AND ano = anoFilme;
```

## Exemplos (cont.)

```
--  
-- Filmes sem actores aparecem no output,  
-- mas actores que não tenham participado  
-- em filmes não aparecem.
```

```
--  
SELECT *  
FROM Filmes LEFT OUTER JOIN Participa ON  
    nome = nomeFilme AND ano = anoFilme;
```

```
--  
-- Actores que nunca participaram em nenhum  
-- filme aparecem no output, mas os filmes  
-- sem actores não aparecem.
```

```
--  
SELECT *  
FROM Filmes RIGHT OUTER JOIN Participa ON  
    nome = nomeFilme AND ano = anoFilme;
```