

Introdução às Bases de Dados

Course Presentation

FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Presentation

- Presentation and available resources
- Motivation
 - why do we require databases?
 - what are database management systems?
- Purpose and course program
- Project
- Evaluation and grading
- Bibliographic information

Presentation and communication



- Theoretical and T-Practical/Laboratory classes
 - Ana Paula Afonso
- Schedule
 - T classes: Tuesdays, **16h10** (16h00) - 18h00
room 6.2.53
 - T-Practical/Lab classes
 - Tuesdays: 13h30 – room 8.2.06 – computer lab (1.2.22)
 - Tuesdays: 18h00 – room 8.2.10 – computer lab (1.2.22)

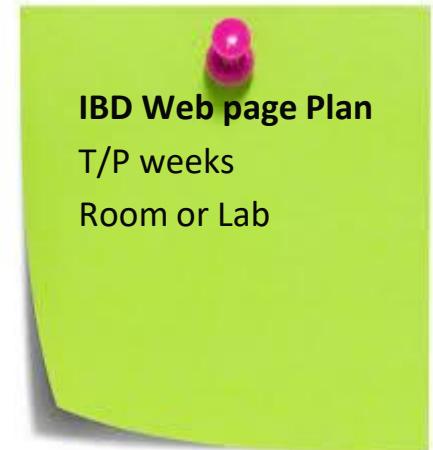
Presentation and communication



- Communication
 - docentes-ibd@listas.di.ciencias.ulisboa.pt
 - IBD Web page: MOODLE FCUL
<https://moodle.ciencias.ulisboa.pt/course/view.php?id=3148>
 - Announcements
 - Student Forum
 - Teacher office hours: Tuesdays, 15h00-16h00,
Office: 6.3.44 or
<https://videoconf-colibri.zoom.us/my/apafonso>
- before send an email to docentes-ibd@listas.di.ciencias.ulisboa.pt**

Schedule: rooms and labs

Horas	Segunda	Terça	Quarta	Quinta	Sexta
13:30 - 14:00					
14:00 - 14:30		[1MSIG-TeA; 1MEIO; 1MBBC; 1PGDS; 1MI; 1PGI; 1MCD] [8 2 06] [TP] TP12			
14:30 - 15:00		Lab. 1.2.22			
15:00 - 15:30					
15:30 - 16:00					
16:00 - 16:30					
16:30 - 17:00		[1MSIG-TeA; 1MEIO; 1MMAEG; 2MMAEG; 1MBioEst; 1PGEABCS; 1MBBC; 1PGDS; 1MEGE; 2MEGE; 1MI; 1PGI; 1MCD] [6 2 53] [T] T11			
17:00 - 17:30					
17:30 - 18:00					
18:00 - 18:30					
18:30 - 19:00		[1MSIG-TeA; 1MMAEG; 2MMAEG; 1MBioEst; 1PGEABCS; 1MBBC; 1PGDS; 1MEGE; 2MEGE; 1MI; 1PGI; 1MCD] [8 2 10] [TP] TP11			
19:00 - 19:30		Lab. 1.2.22			
19:30 - 20:00					



Motivation

data, database e database management systems

Example: YouTube

The image shows a screenshot of the YouTube mobile interface. At the top left, there's a navigation bar with icons for Home, Trending, Subscriptions, Library, History, Watch later, Purchases, Liked videos, Popular on YouTube, Music, Sports, Gaming, and More from YouTube (Premium and Movies & Shows). The main area has a search bar with the text "funny cats" and a result count of "About 12,100,000 results". Below the search bar, there are three video thumbnails highlighted with blue boxes:

- How to Get Rid of Cat Pee Stains** by BISSELL (2M views)
- CATS make us LAUGH ALL THE TIME! - Ultra FUNNY CATS compilation** by Tiger FunnyWorks (100K views, 3 days ago)
- Have you EVER LAUGHED HARDER? - Ultra FUNNY CATS** by Tiger FunnyWorks (124K views, 1 week ago)

The video player in the center is showing a clip titled "Baby Cats - Funny and Cute Baby Cat Videos Compilation (2018) Gatitos Bebes Video Recopilación" by Animal Planet Videos (Published on May 23, 2018). The video is at 0:34 / 15:25. The player interface includes a like button (4.7K), a dislike button (768), and a share button. To the right of the video player, there are several recommended videos and a sidebar with a green box around a thumbnail for "Top Cats Vs. Cucumbers Funny Cat Videos Compilation" and a red box around the "Upload video" and "Go live" buttons.

Motivation

data, database e database management systems

Other examples: Facebook ...

users, friends, activities, announcements, ...



clients, consumptions, billing

O meu perfil My Vodafone



clients, doctors, consultations, exams

Serviços mais procurados

- CONSULTAS
Marque a sua consulta >
- EXAMES
Marque o seu exame >
- ANÁLISES
Saiba mais >

Encontre um Médico

Por nome

 Escreva ou duplo clique para lista

Por especialidade

 Escreva ou duplo clique para lista

Por doença

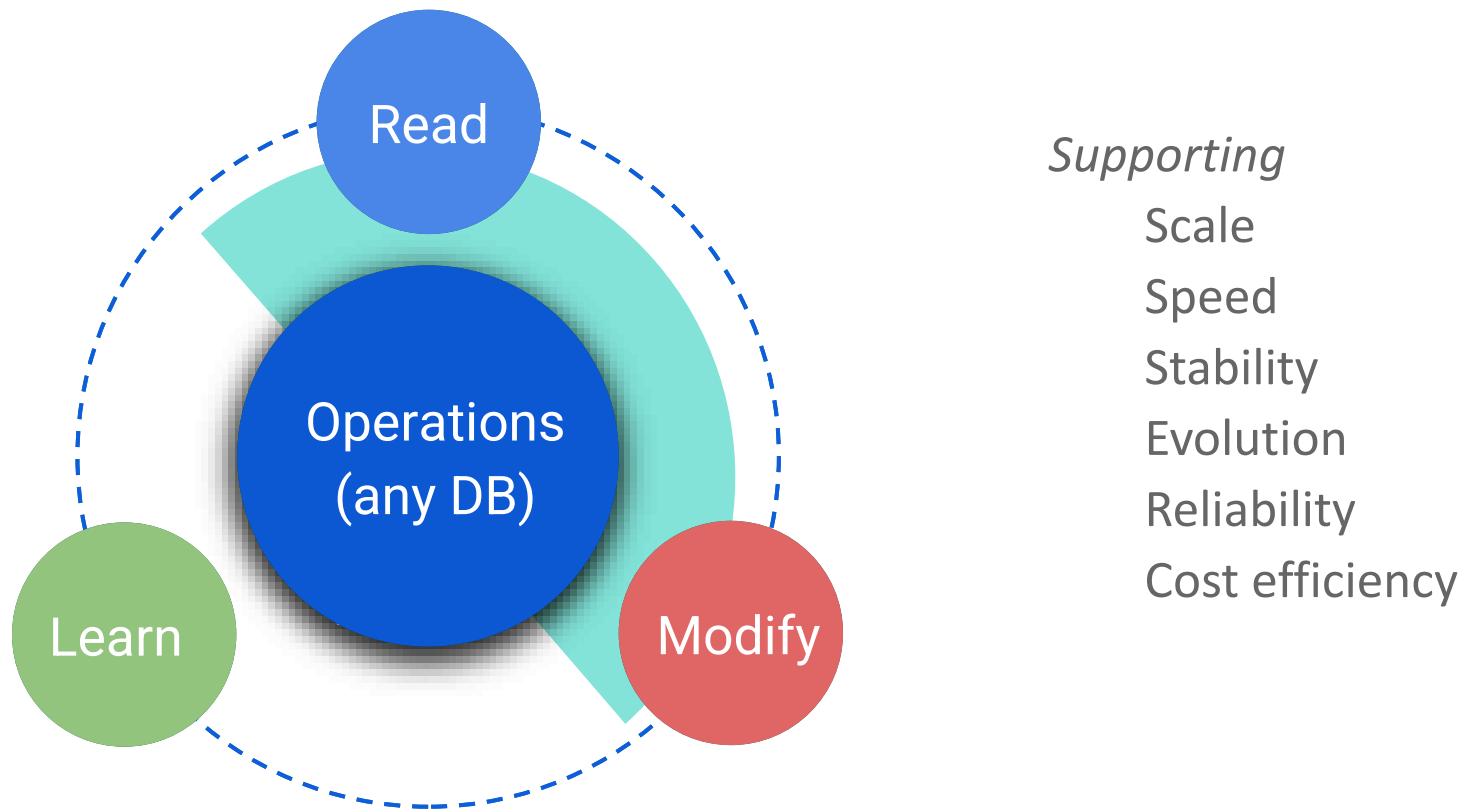
 Escreva ou duplo clique para lista

Por sistema/seguro de saúde

 Pesquisa...

Goals of standard databases

Store and manage information



Introduction

- Since the 90's, organizations become aware that information is one of their most critical and valuable assets
- Information value depends upon its validity, correctness and availability
- Database systems are essential tools for managing information

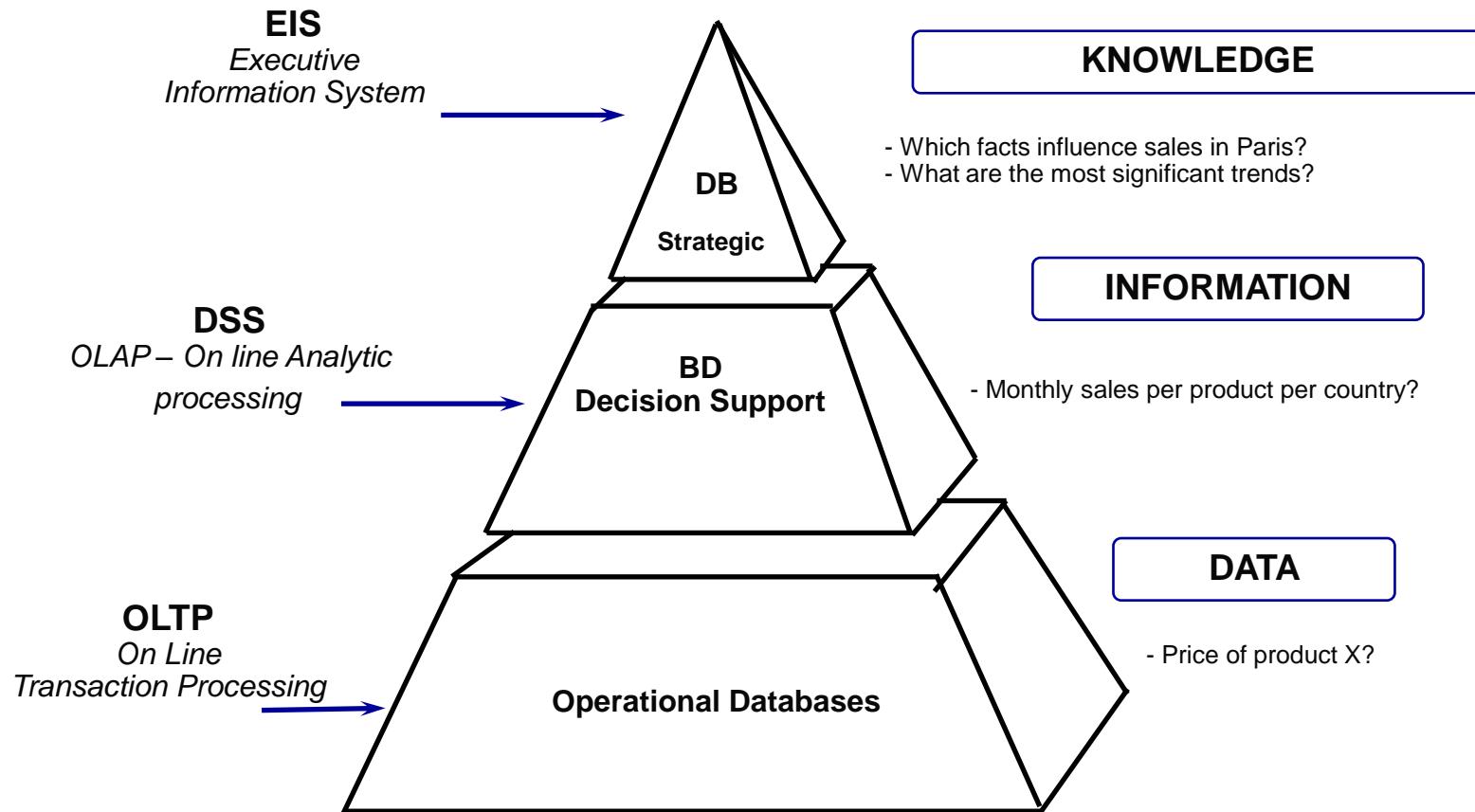
Data vs Information

Data are real world facts

Answers to a survey or a physical measurement are considered as “data”

Information results from data processing presented in such a way to allow interpretation and give the fundament for decision making

Information Systems



Database Management Systems (DBMS)

- What is a Database?
 - A large collection of integrated data
 - Model the intrinsic characteristics of the universe it tries to encompass:
 - Entities (e.g., Students, Courses)
 - Relationships (e.g., Ana is enrolled in IDB)
- What is a DBMS?
 - It is a software package designed to store and manage large amounts of data and coordinate user access

Advantages and disadvantages of using a DBMS

- **Advantages of using a DBMS**

- Data independence
- Efficient access
- Reduced time for developing and maintaining applications
- Easy and centralized **data integrity** mechanisms and **increased security**
- Uniform and simplified administration
- Allows for concurrent access and easy fault recovery

- **Disadvantages**

- Data sharing creates conflict
- It is harder to maintain than simple file systems
- Requires specialized training
- Typically larger investment in software and hardware
- They may not be as performant as some file based systems

Objectives

- To acquaint students with the fundamental principles of **data centered information systems** and information organization independent of any program language that manipulates it
- Explain and illustrate the full process of **database construction and management**. From conceptual modeling to logical models and actual implementation
 - Entity-relationship (ER) modeling
 - Relational modeling
- Explain how to manipulate data and extract data from databases using the Structured Query Language (SQL)
- Explain the principles of database interaction via a programming language with SQL

Course summary - goals

We'll learn how to...

Design “good” databases

conceptual design, logical design, schema normalization

Query over small-med-large data sets with SQL

On relational engines

Update data sets

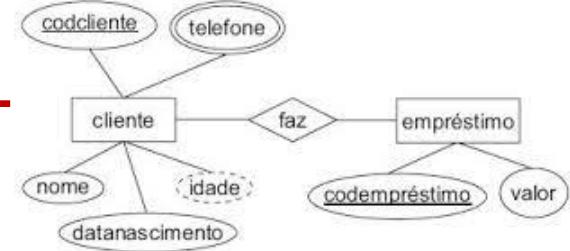
Writes, transactions, logging, ACID properties

Course planning (tentative)

Semana	Data	Teóricas	Teórica Prática	Local	Entregas
1	14-Sep	Apresentação da disciplina. Métodos de Avaliação. Programa da Disciplina; Bibliografia. Introdução aos Sistemas de Gestão de Bases de Dados (capítulo 1)			Formação grupos
2	21-Sep	Desenho Conceptual de BD: Construtores do modelo EA - entidades, associações, participação e multiplicidade nas associações, entidades fracas, generalizações e agregações (capítulo 2)	Apresentação das aulas TP. Exercícios de modelação conceitual	sala	Formação grupos
3	28-Sep	Desenho Conceptual de BD: Decisões no Desenho Conceptual de BD. Erros comuns, projetos complexos e verificação (capítulo 2)	Exercícios de modelação conceitual	sala	Formação grupos
4	05-Oct	FERIADO	FERIADO		Formação grupos (data limite: 10 out)
5	12-Oct	Modelo Relacional: história, relação, BD relacional, SQL. Conceitos de Chaves: Chaves primárias, chaves candidatas chaves estrangeiras. Integridade de Chave, Entidade e Referencial (capítulo 3 e 5)	Exercícios de modelação conceitual	sala	E1 - Simple E/R (17 out, 23:59)
6	19-Oct	Desenho Lógico de BD: Passagem do EA para relacional, entidades, associações, RI, ent. Fracas, generalizações e agregações (capítulo 3)	Introdução ao SGBD Mysql. SQL/DDL: criação tabelas, inserção de dados e regras de integridade	lab	
7	26-Oct	Desenho Lógico de BD: Vistas; Apagar e alterar tabelas e vistas (capítulo 3) Project - E1: Uma solução e erros comuns.	Passagem do diagrama EA para um esquema relacional	sala	
8	02-Nov	Introdução ao SQL/DML: Interrogações (capítulo 5) SQL/DML: Operadores de Agregação (capítulo 5)	Passagem do diagrama EA para um esquema relacional	sala	E2 - E/R, SQL/DDL (7 nov, 23:59)
9	09-Nov	SQL: Valores Nulos e Joins. Restrições de Integridade (capítulo 5)	Discussão de projetos: E2	sala	
10	16-Nov	Introdução à normalização no modelo relacional (capítulo 19)	SQL/DML - Interrogações	sala	
11	23-Nov	Conceitos de Gestão de Bases de Dados: Planeamento da Base de Dados, gestão de transações (capítulo 16)	SQL/DML - Interrogações	lab	
12	30-Nov	Conceitos de Gestão de Bases de Dados: Segurança e gestão de utilizadores (capítulo 21)	SQL/DML - Interrogações	lab	
13	07-Dec	Desenvolvimento de Aplicações com Bases de Dados (capítulo 6). Bases de dados não relacionais	SQL/DML - Interrogações	lab	E3 - SQL/DML (12 dec, 23:59)
14	14-Dec	Discussão de projetos	Discussão de projetos: E3	lab	

Detailed

- **Theoretical classes**
 - Overview of database management systems
 - Conceptual database design: Entity-Relationship model and UML
 - Logical database design: Relational model, SQL DDL, and normalization
 - DBMS queries: SQL DML
 - Overview of transaction management
 - Database application development
 - NoSQL databases
- **Theoretical-practical classes** (start next week)
 - Exercises about the subjects given in the theoretical component
 - Use of a programming language to access the database management system



Project

Elaboration of a project, where is requested the development of relational database. Components of the project:

1. A simple Universe of Discourse (UoD) is presented to students to make a conceptual model (E/R)
2. A more complex UoD is provided and new models are built, conceptual (E/R) and logical models (SQL/DDL) are built and implemented in SQL
3. A series of query problems is given to solve in SQL/DML to a solution of Phase 2

Evaluation method

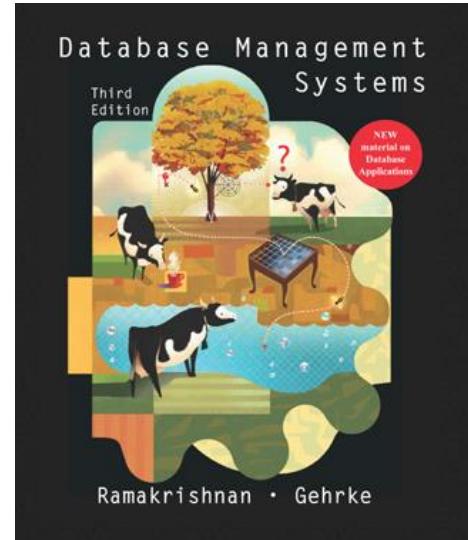
- **Project – 50%**
 - Project with 3 deliverables / 2 individual discussions (presential)
 - Teamwork
 - group size: 3 - 4 students per group
 - use the course moodle activity to create the groups (deadline: October 10)
- **Final exam – 50%**
 - Examination calendar available at: www.fc.ul.pt/

All components must score at least 9.5 for course approval

Bibliographic references

- **Essential**

- Ramakrishnan R. and Gehrke J.
Database Management Systems (3rd ed.), McGraw-Hill



Also important for specific topics

- Elmasri and Navathe - Fundamentals of Database Systems (7th ed.), Pearson

- **Course material at the FCUL's Moodle**

- Slides of the theoretical classes (in Portuguese)
 - Lab class materials and datasets
 - Tutorials of SQL and MySQL

Honor code rules

Any work submitted for grading should not be derived from or influenced by the work of others. All submissions are subject to plagiarism detection tools.

Examples of honor code violations include (but are not limited to):

- reusing your own or another student's assignment work from previous years
- sharing your responses/answers/code/design with other students nor publicly
- joint design/development/debugging
- use of web or public resources for public solutions
- copying code or answers
- posting up/dispersing your solutions or code on public repos

IBD Moodle web page

Introduction to Database Systems 2021-2022
(Introdução às Bases de Dados)



The main objective is to know the principles of relational database management systems, in order to develop and manage a real-world relational databases.

Contents: Overview of DataBase Management Systems; Conceptual Database Design; Logical Database Design; DBMS queries; Database Application Development and Overview of Transaction Management

Teachers

- Ana Paula Afonso, Regente - Theoretical and T/P
Office hours: Tuesdays, 15h00-16h00, 6.3.44 or [Zoom](#) (students must send an email before)

Schedule

Theoretical Classes

- Tuesday, 16h00-18h00, room 6.2.53

T/Practical and Labs Classes

- TP11: Tuesday, 18h00-19h30, room 8.2.10/lab 1.2.22 (see plan)
- TP12: Tuesday, 13h30-15h00, room 8.2.06/lab 1.2.22 (see plan)

Communication

E-mail: docentes-ibd@listas.di.ciencias.ulisboa.pt

Introdução às Bases de Dados

Modelo Entidade-Associação - I

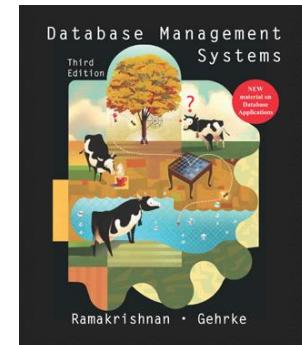
FCUL, Departamento de Informática

Ano Letivo 2021/2022

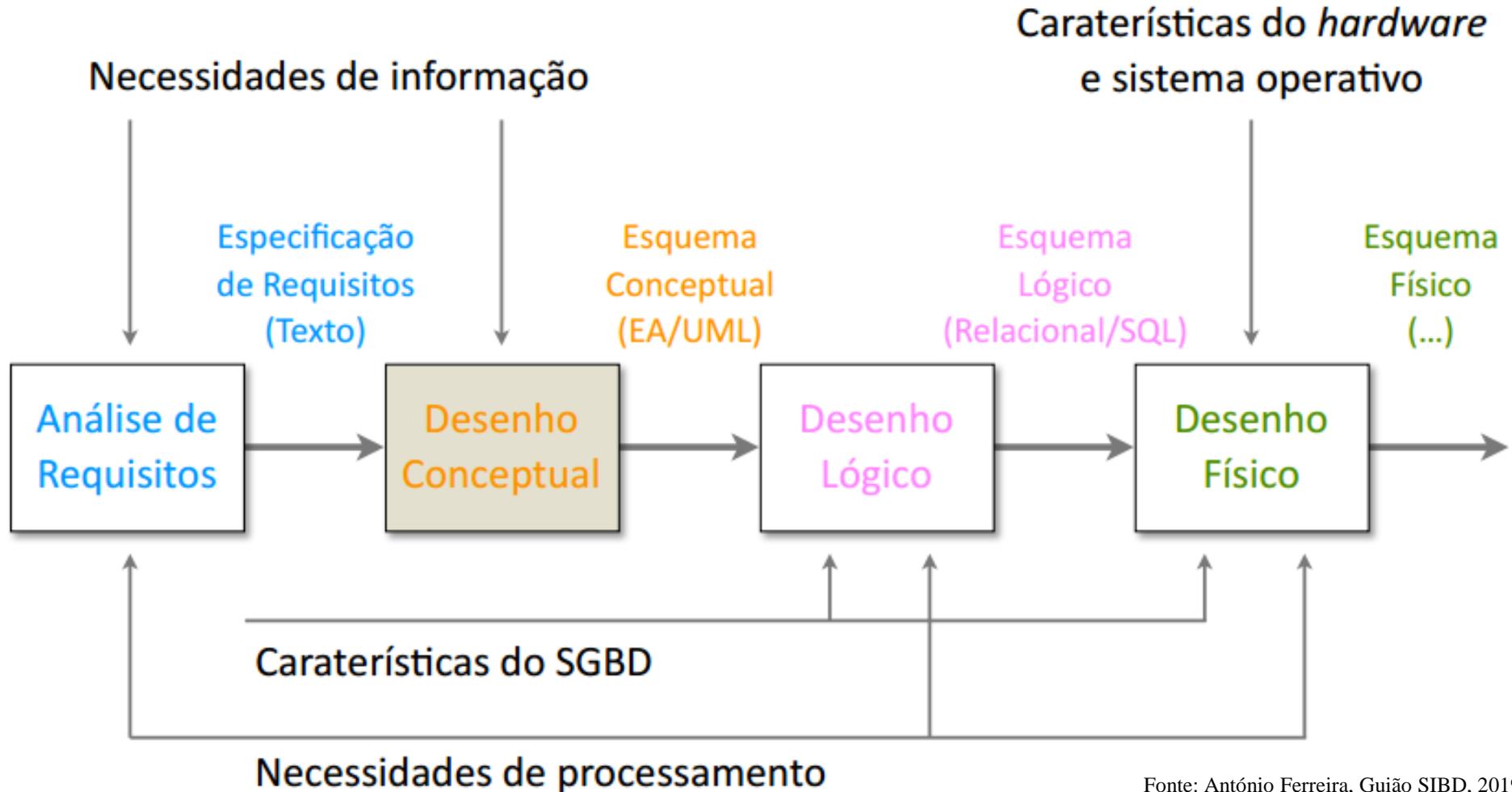
Ana Paula Afonso

Sumário e Referências

- Sumário
 - Processo de desenho de BD: etapas e conceitos
 - Modelo entidade-associação (EA)
 - Entidades
 - Associações
 - Associações ternárias
 - Papéis nas associações
 - Restrições em associações
 - Chave (Multiplicidade)
 - Participação
- Referências
 - R. Ramakrishnan (**capítulo 2**)



Processo de Desenho de BDs



Fonte: António Ferreira, Guião SIBD, 2019

Modelo de Dados e Esquema

- Modelo de dados
 - Colecção de construtores para descrever os dados a nível abstrato
 - Esconde detalhes de armazenamento
 - Define os dados que serão armazenados pelo SGBD
- O modelo de dados está mais perto da estrutura de armazenamento do SGBD do que da visão do utilizador
- A descrição dos dados em termos de um modelo de dados é chamado de **esquema (*schema*)**

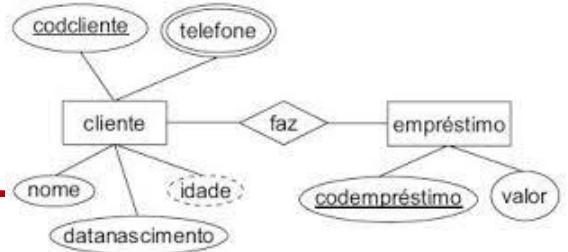
<i>sid</i>	<i>name</i>	<i>lZogin</i>	<i>age</i>	<i>gpa</i>
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Modelo Relacional

<i>sid</i>	<i>[name]</i>	<i>IZogin</i>	<i>age</i>	<i>gpa</i>
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

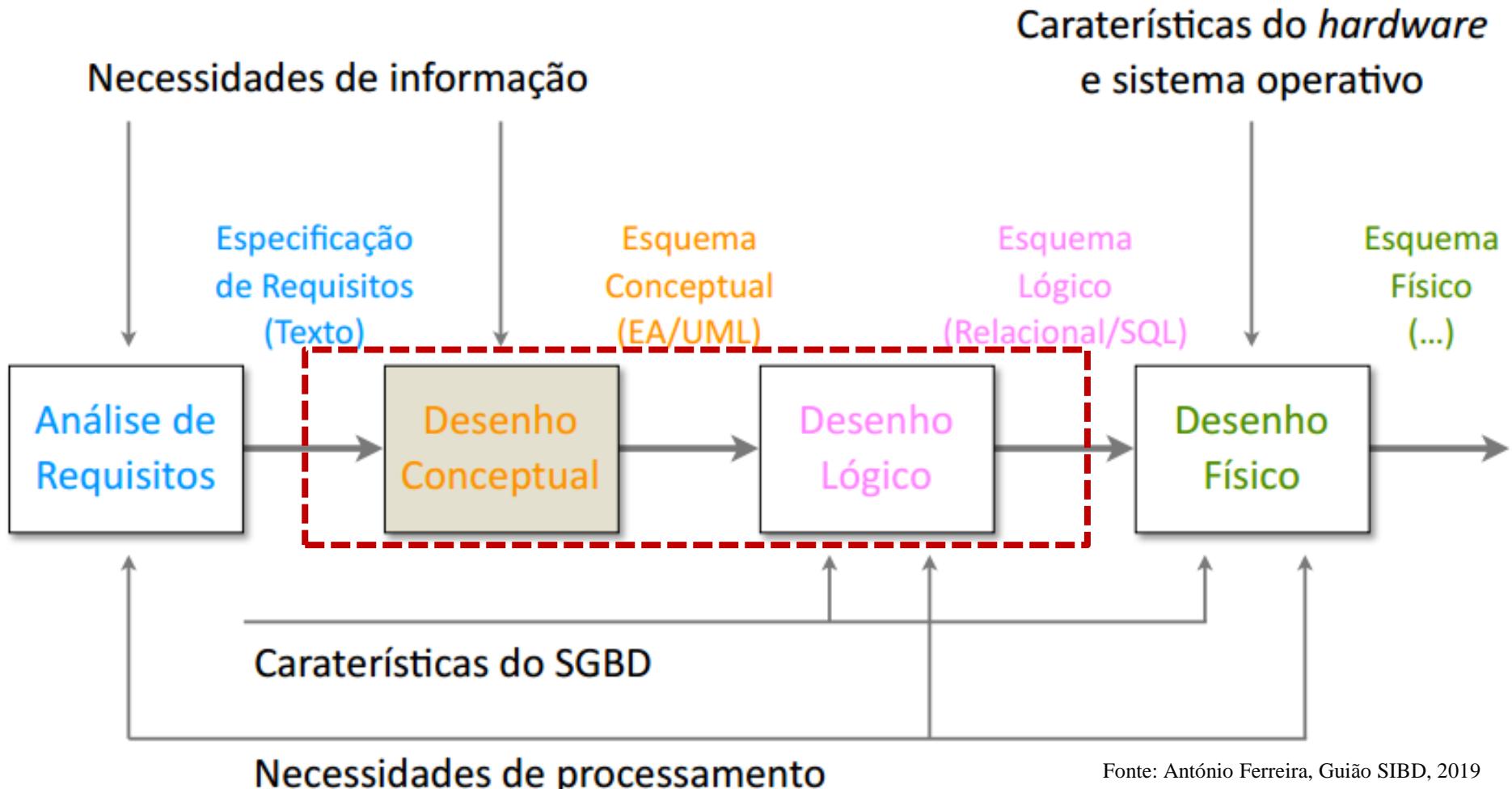
- O Modelo mais usado na atualidade
 - Conceito fundamental é a
Relação = tabela com **colunas** (atributos, campos) e **linhas** (registos)
- **O esquema relacional**
 - Descreve o nome das **relações** e suas **colunas**
 - *Exemplo: Students (sid: string, name: string, login: string, age: integer, gpa: real)*
Cada linha na relação *Students* é um **registro** que descreve um aluno
Cada linha segue o **esquema da relação** *Students*
- Inclui também **regras de integridade**, que são condições que os registos de uma relação têm de satisfazer
 - *Exemplo, Student.age > 0*

Modelo Semântico dos Dados



- Modelo relacional suportado diretamente pelos SGBDs relacionais
 - SGBDs têm comandos para criar tabelas e gerir registos de dados
 - Ex. CREATE TABLE student (...), INSERT INTO student (...)
- Contudo, modelo relacional nem sempre é o mais adequado
 - A primeira abordagem de modelação deve ser mais abstrata
- **Modelo semântico** dos dados
 - Modelo mais rico com construtores mais ricos para descrever a realidade
 - Modelo mais perto da visão do utilizador
 - No final é traduzido para um modelo de dados
- **Modelo Entidade-Associação (EA)** *Entity-Relationship (ER) model*
 - Descreve de uma forma gráfica as entidades e as relações entre elas (mas, existem outros modelos)
 - Ponto de partida da modelação de dados em BD

Processo de Desenho de BDs



Fonte: António Ferreira, Guião SIBD, 2019

Processo de Desenho de BDs

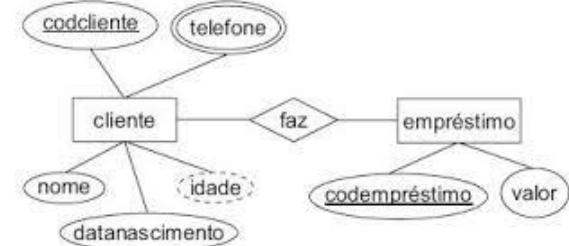
- Análise de Requisitos
 - Define o **espaço do problema**: identificar e descrever dados e processos pretendidos pela organização
 - Representação textual, com envolvimento dos utilizadores
- Desenho Conceptual da BD
 - Define o **espaço da solução**: sintetizar num **modelo semântico** as diferentes necessidades numa descrição de alto nível dos *dados* e *restrições*
 - Descrever entidades, associações e atributos, independentes da tecnologia de BD (ex. **modelo entidade-associação**)
- Desenho Lógico da BD
 - Define dados e regras usando o modelo suportado pelo SGBD
 - **Esquema lógico (schema)** dependente da tecnologia da BD (ex. **modelo relacional**)
- Desenho Físico da BD
 - Definir estruturas físicas dos dados, adequadas ao ambiente informático particular
 - Definir estratégias de segurança, desempenho, recuperação e salvaguarda (backup)

Conceitos

- **Universo do Discurso (UoD)**
 - Fragmento do mundo real para o qual se pretende conceber o SI
- **Estrutura de Conceitos**
 - Conjunto de abstrações usadas para simbolizar as entidades do UoD
- **Modelo Semântico**
 - Uma interpretação de um UoD através de uma estrutura de conceitos

Abordagens com notação em forma de diagrama:

- **Entidade-Associação (EA)**
Várias versões: várias simbologias, estrutura de conceitos semelhante
- Unified Modeling Language (UML)
- ...

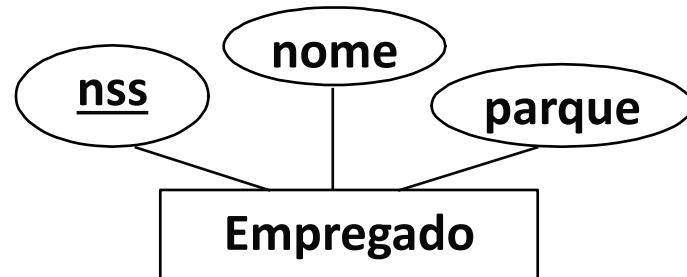


EA Conjunto de Entidades

- **Entidade:** objeto do mundo real
- **Conjunto de Entidades:** coleção de entidades semelhantes
 - Partilham atributos
 - Partilham associações
- Uma **chave** é o conjunto mínimo de atributos cujos valores identificam univocamente cada entidade do conjunto
 - Existem várias **chaves candidatas**
 - Uma é escolhida para ser **chave primária**

Representação de Conjuntos de Entidades

- Representado por um *retângulo*
- Os **atributos** por *elipses*
 - A **chave primária** está sublinhada

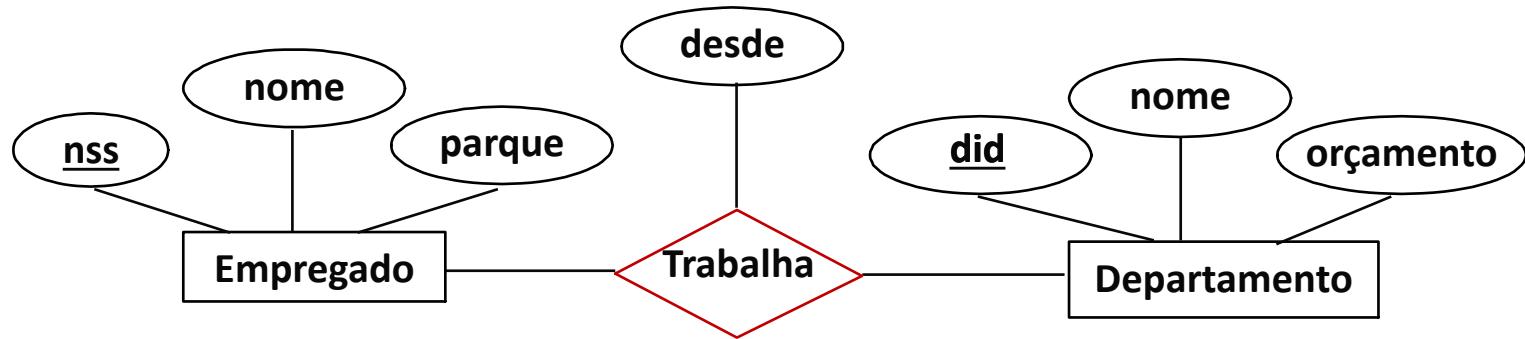


EA Conjuntos de Associações

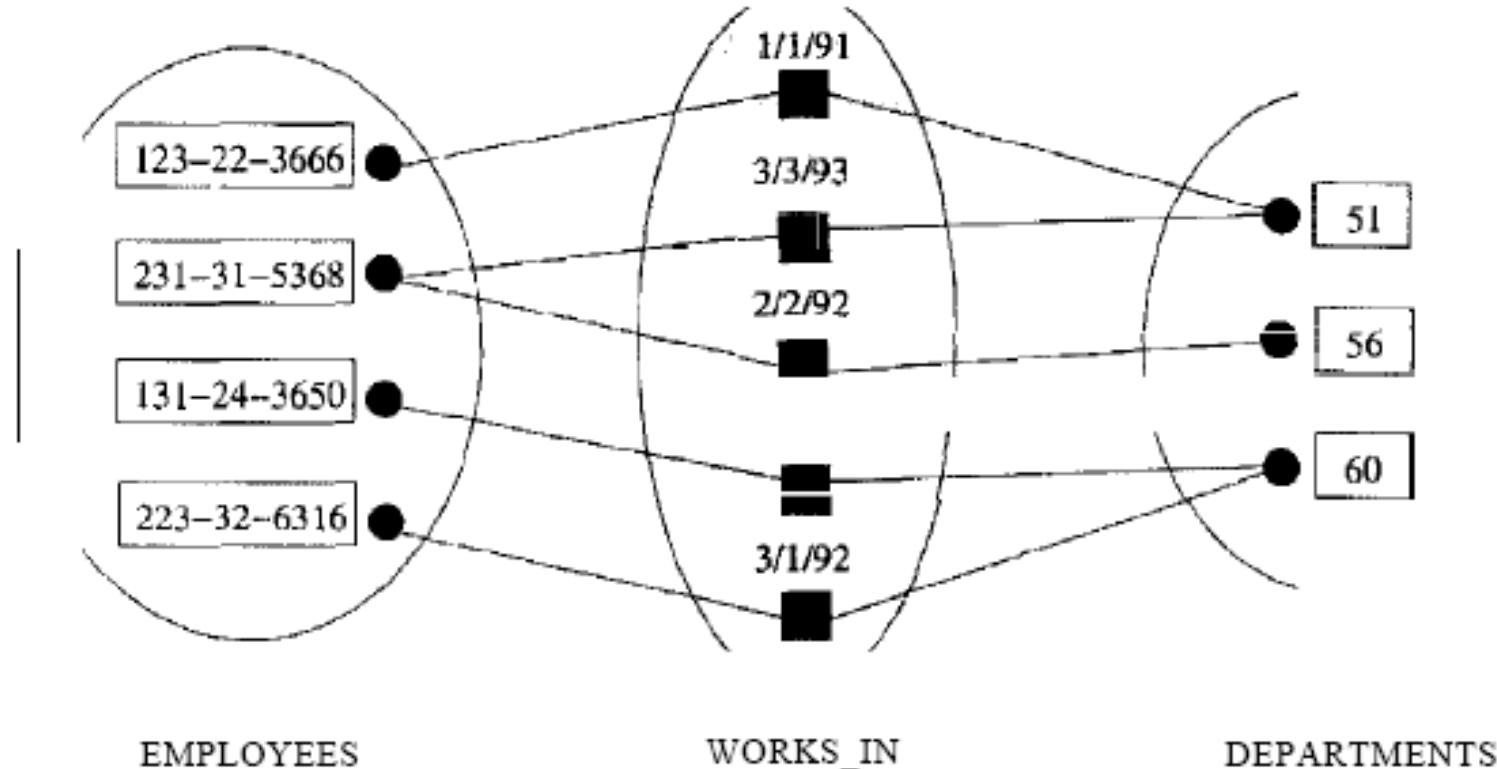
- **Associação:** relação entre duas ou mais entidades
- **Conjunto de Associações:** coleção de associações semelhantes
 - Relacionam os mesmos conjuntos de entidades:
 - Binárias: dois conjuntos de entidades
 - Não necessariamente distintos
 - Ternárias: três conjuntos de entidades
- A associação pode ter **atributos descritivos**
 - Com informação sobre a associação
 - Não servem para guardar historial

Representação do Conjunto de Associações

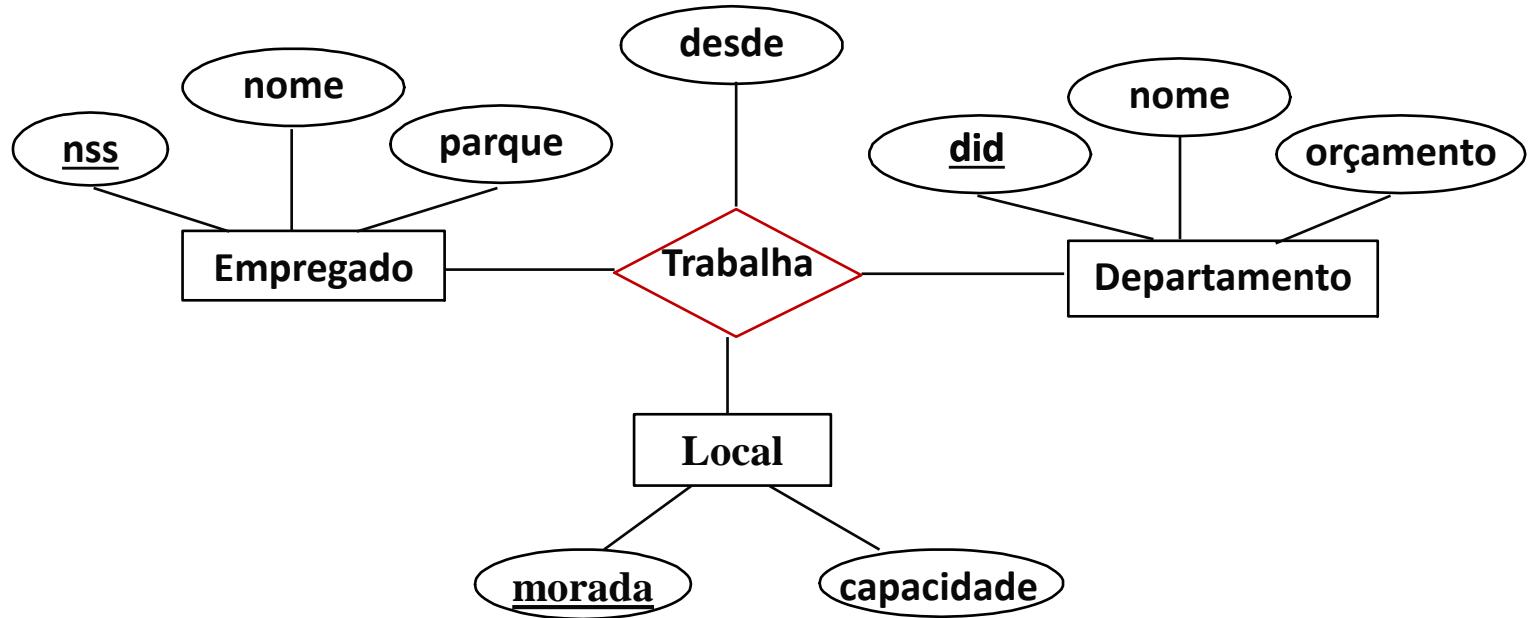
- O conjunto de associações é representado por um *losango*



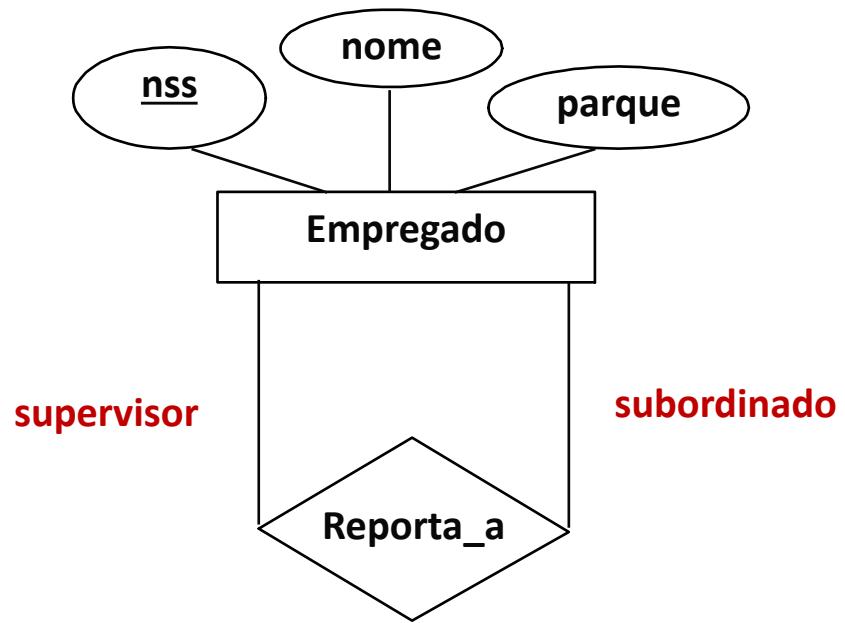
Instâncias das Associações



Conjunto de Associações Ternárias

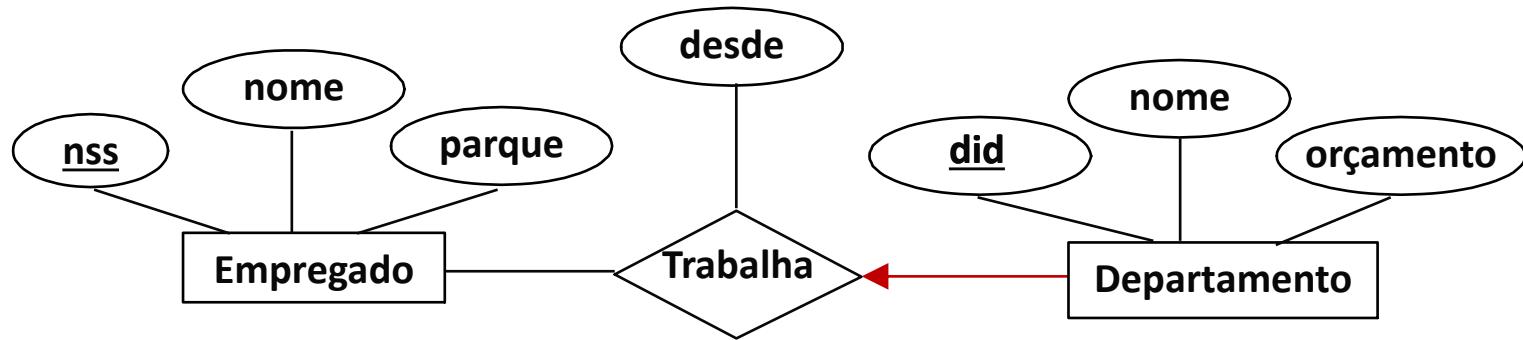


Papéis na Associação

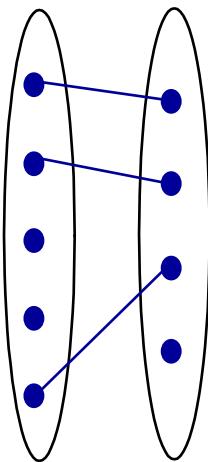


Restrição de Chave multiplicidade

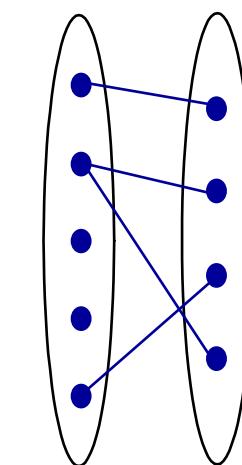
- No máximo uma associação por cada entidade
- Representado por uma seta



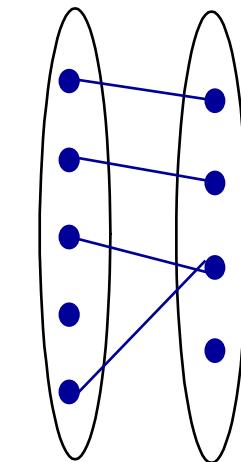
Combinações de Restrições



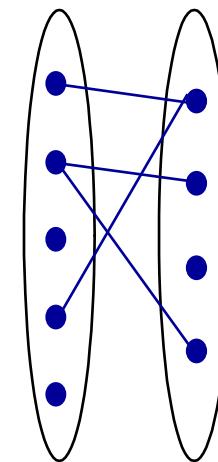
1-para-1



1-para-muitos

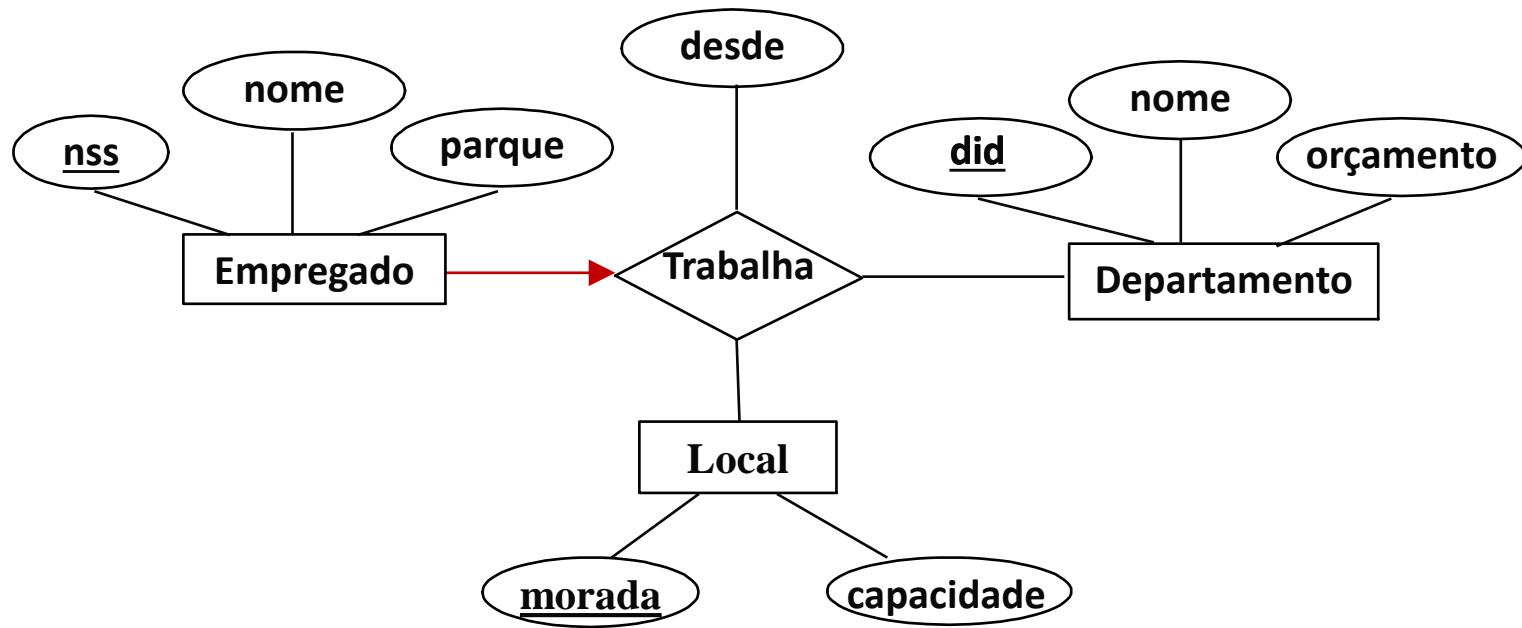


muitos-para-1

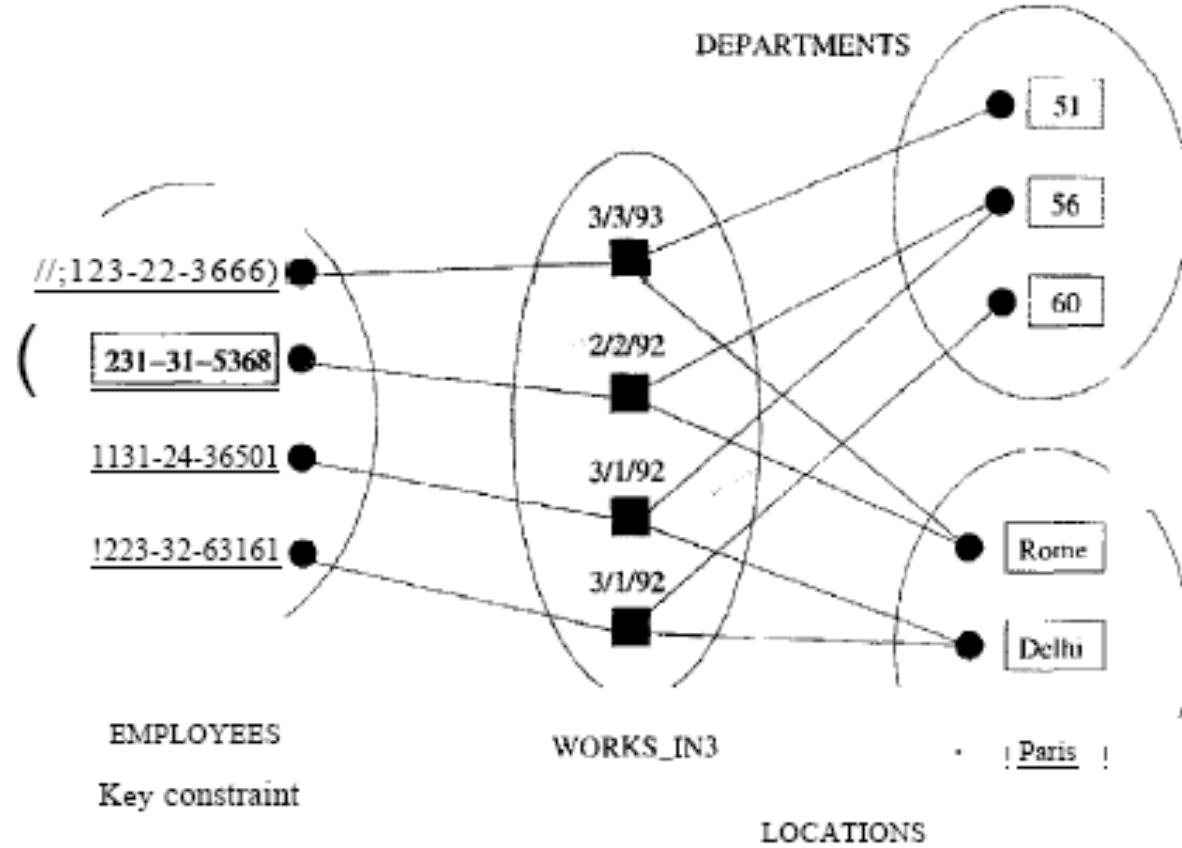


muitos-para-muitos

Restrição de Chave nas Ternárias

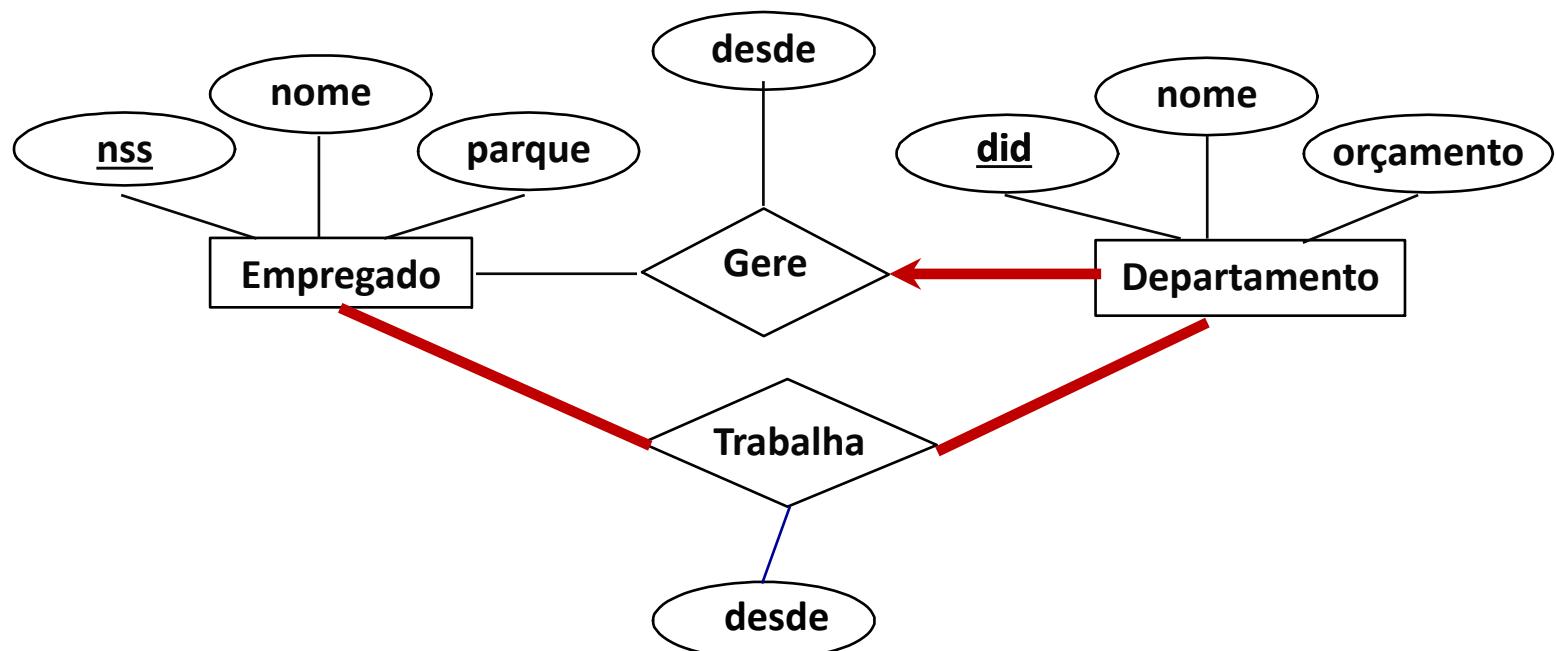


Instâncias da Ternária



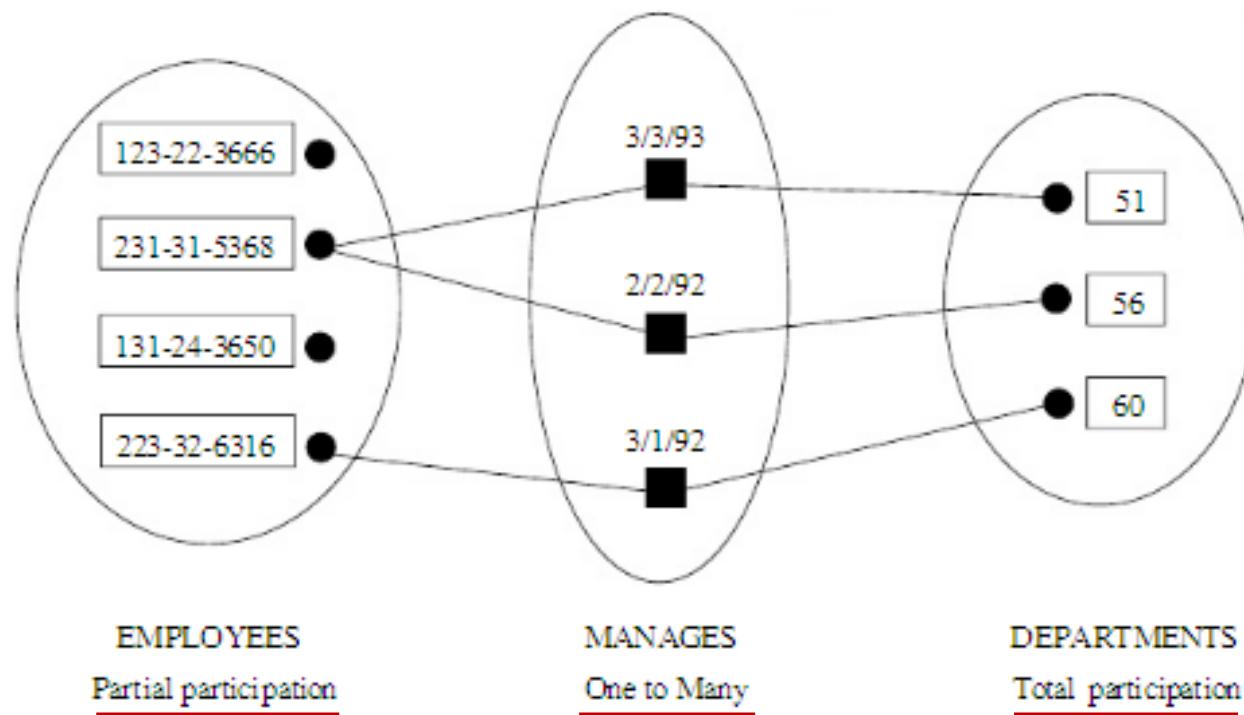
Restrição de Participação

- No **mínimo** uma associação por cada entidade
 - **Participação TOTAL** -Representada por uma *linha a cheio*
 - Por omissão: assume-se participação **PARCIAL**



Restrições de Chave e Participação

Instâncias



Ferramentas para desenho EA

- DIA Diagram Editor
 - <https://wiki.gnome.org/Apps/Dia/>
- Diagrams.net (Draw.io)
- Creately
 - <https://creately.com/>
- Lucidchart
 - <https://www.lucidchart.com/pages/>

Introdução às Bases de Dados

Modelo Entidade-Associação - II

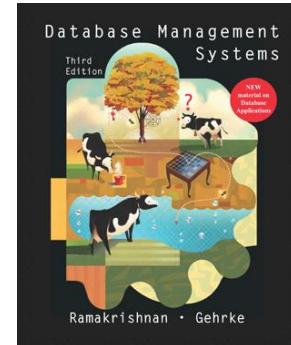
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Modelo entidade-associação (EA)
 - Entidades fracas
 - Generalizações
 - Restrições em generalizações
 - Sobreposição
 - Cobertura
 - Agregações
- Referências
 - R. Ramakrishnan ([capítulo 2](#))

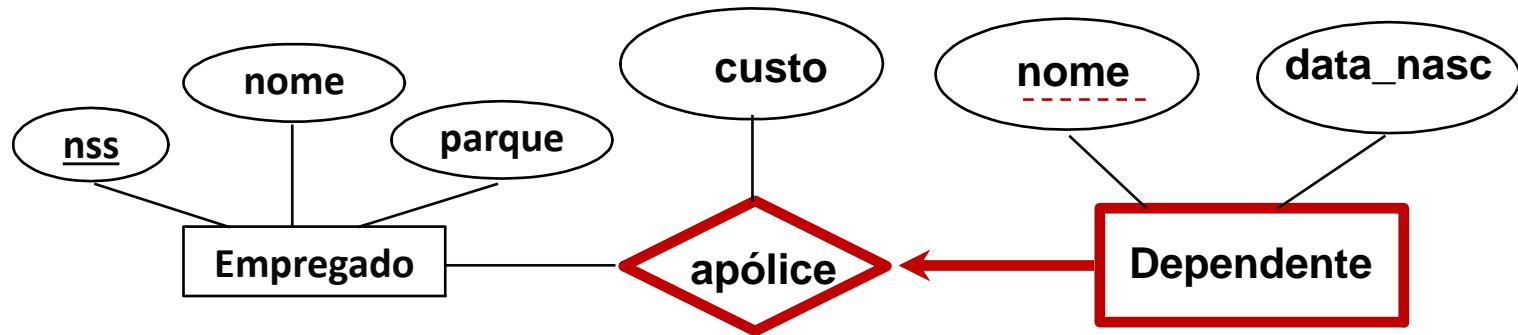


Entidades Fracas

- Uma **entidade fraca** só pode ser identificada
 - Através dos seus atributos
 - E da chave primária de outra entidade
 - Chamada de entidade forte
- As seguintes restrições têm de existir:
 - O conjunto de entidades fortes e o conjunto de entidades fracas devem participar numa associação **um-para-muitos**
 - Chamada de **associação identificadora**
 - O conjunto de entidades fracas deve ter **participação total** na associação identificadora

Exemplo de Entidade Fraca

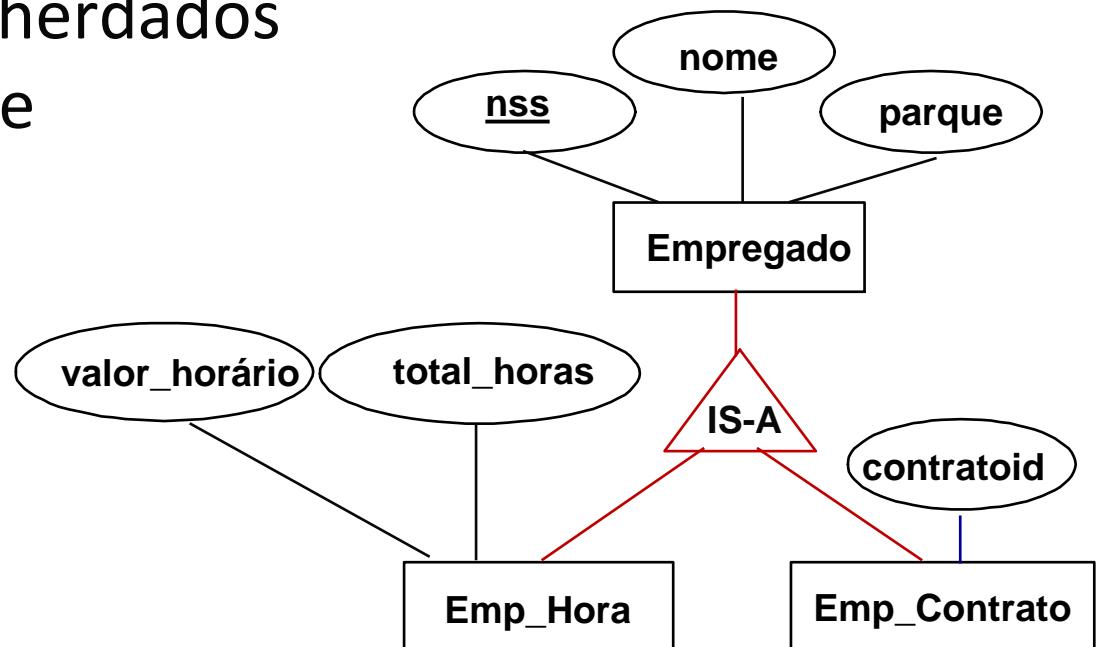
- Representada por *linhas a cheio*
- A **chave parcial** está sublinhada a *tracejado*



Qual a diferença se tiver apenas restrição de chave (multiplicidade) e participação total ?

Generalizações

- Classifica um conjunto de entidades em subconjuntos
- Os atributos do conjunto das **super-entidades** são herdados pelos subconjuntos de **sub-entidades**

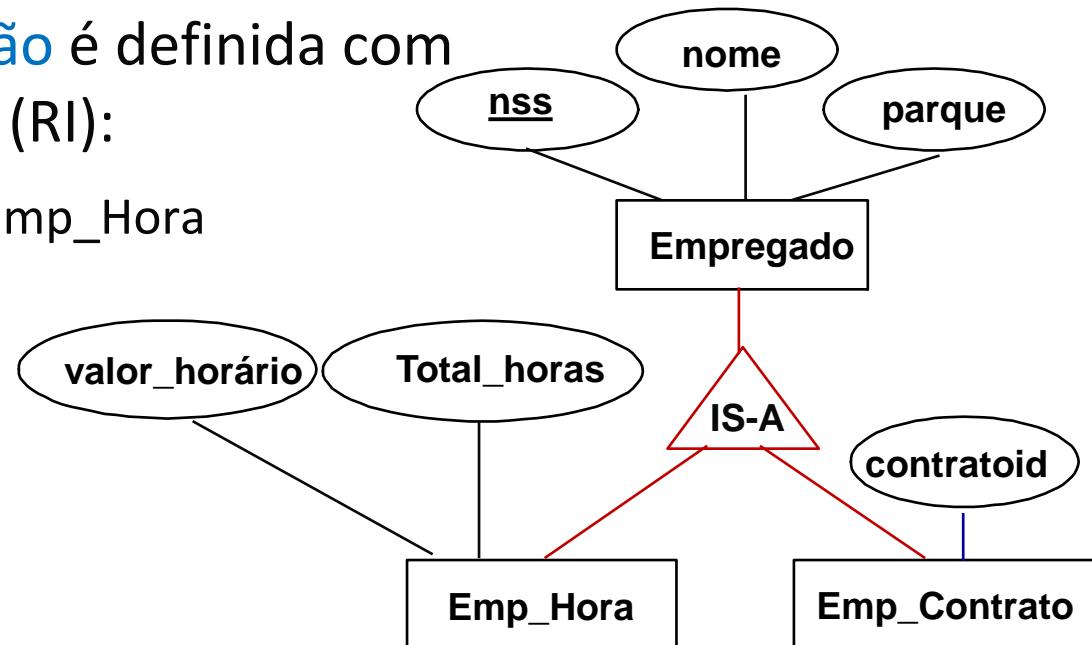


Especialização vs. Generalização

- Empregados estão **especializados** em subconjuntos
 - Especialização é o processo de identificação de subconjuntos de entidades que partilham características comuns: **atributos ou associações**
- Hourly_Emps e Contract_Emps são **generalizados** por Employees
 - Generalização consiste na identificação de algumas características comuns a vários conjuntos de entidades e representar essas características num novo conjunto de entidades

Restrições de Sobreposição

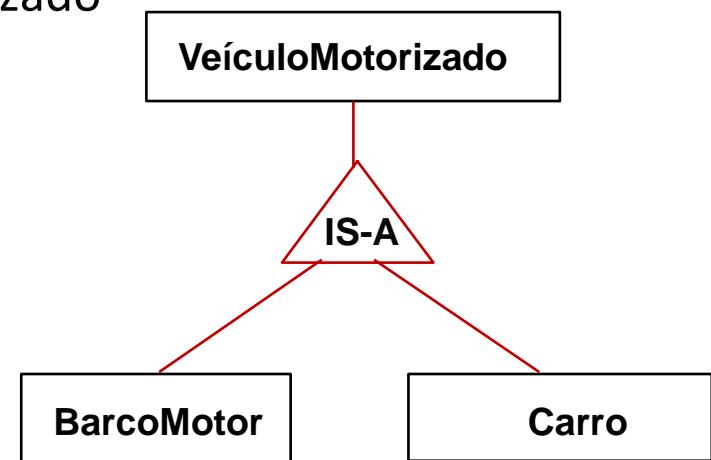
- Restrições de sobreposição determinam se dois subconjuntos podem conter a mesma entidade
 - Sobrepostos ou disjuntos
 - Por omissão os subconjuntos são **disjuntos**
 - A restrição de **sobreposição** é definida com uma regra de integridade (RI):
ex: Emp_Contrato **OVERLAPS** Emp_Hora



Restrições de Cobertura

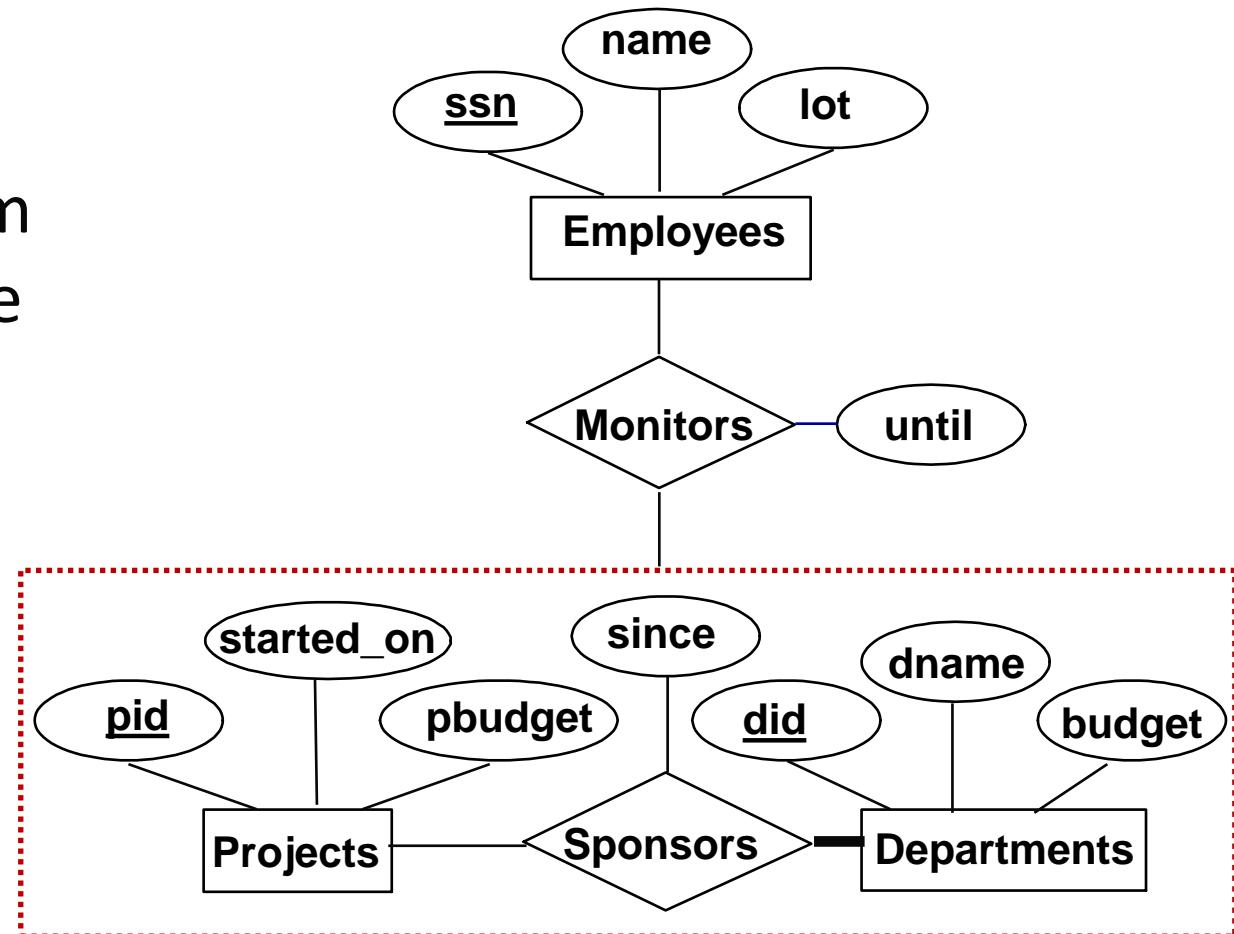
- **Restrições de cobertura** determinam se todas as entidades estão representadas nos subconjuntos
 - Total ou parcial
 - Por omissão a cobertura é **parcial**
 - A restrição de **cobertura total** é definida com uma regra de integridade (RI)

ex: BarcoMotor AND Carro **COVER** VeiculoMotorizado



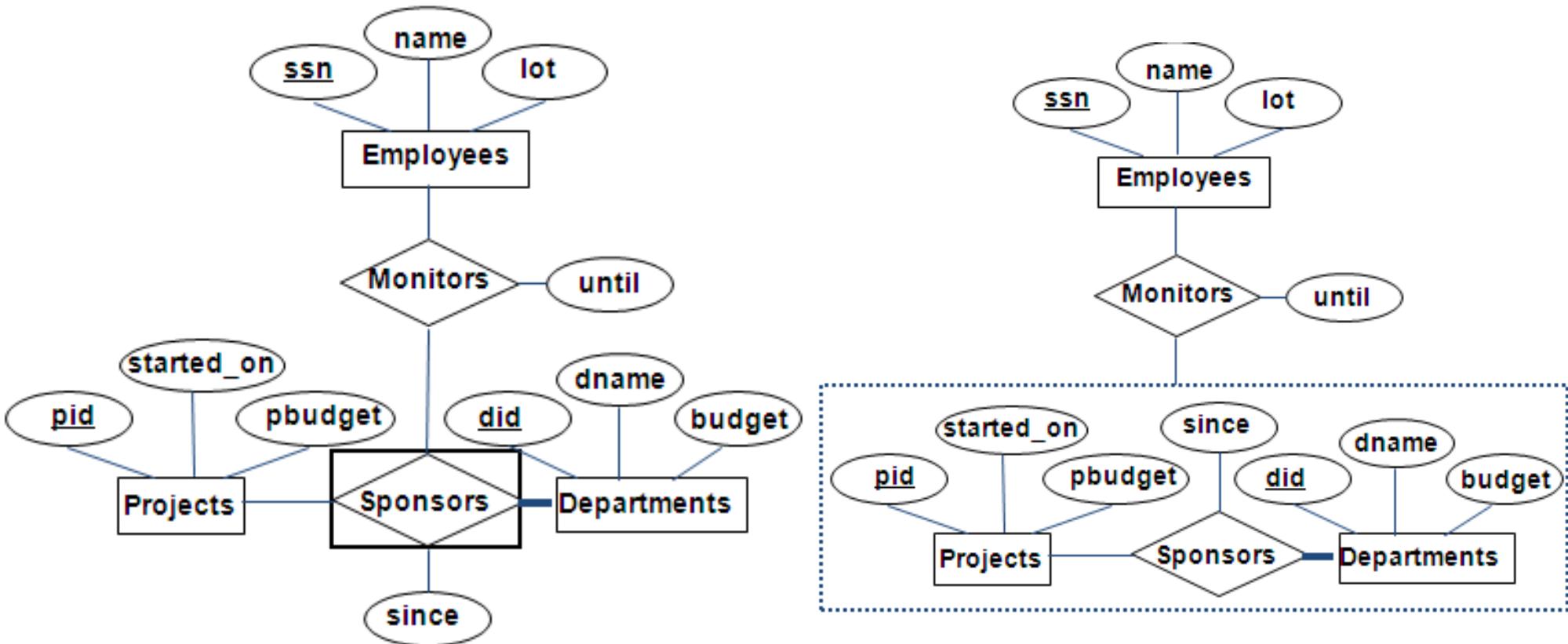
Agregação

- Agregação indica que um conjunto de associações está associado com um outro conjunto de associações



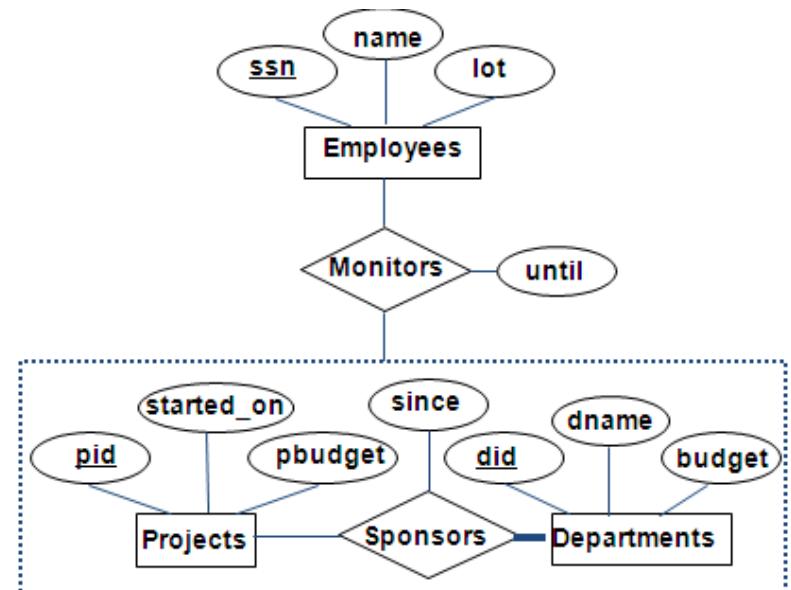
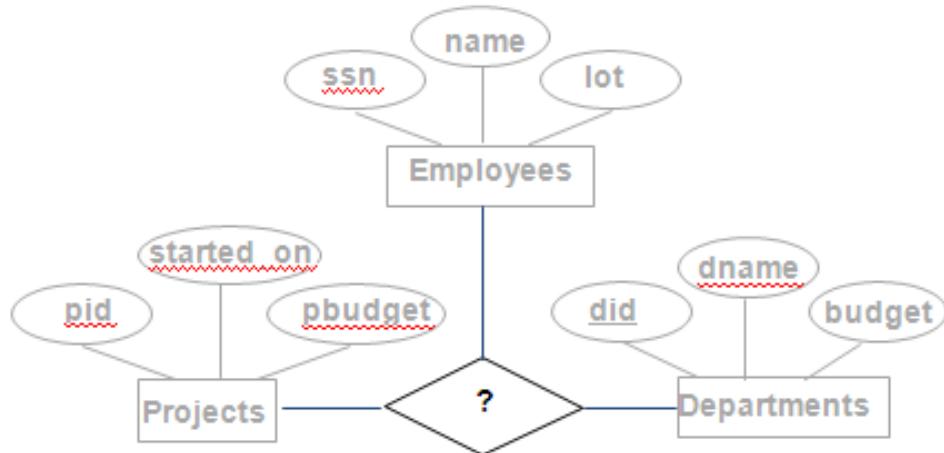
Aggregação

Notação alternativa



Agregação

- Porque não definir uma associação ternária entre *Projects*, *Departments* e *Employees* ?



A Seguir

Outras questões deste género:

- Um conceito deverá ser modelado como entidade ou atributo? Como entidade ou relação?
- Devemos usar associações binárias ou ternárias?
- Devemos usar associações ternárias ou agregações?
- E os projectos complexos?
- Que erros comuns?

em: *Decisões no Desenho Conceptual de BD*

Introdução às Bases de Dados

Modelo Entidade-Associação - III

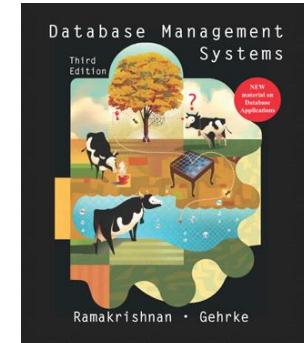
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Estudo de alternativas
 - Atributos vs. entidades
 - Associações vs. entidades
 - Associações ternárias vs. binárias
 - Agregações vs. associações ternárias
- Referências
 - R. Ramakrishnan (**capítulo 2**)



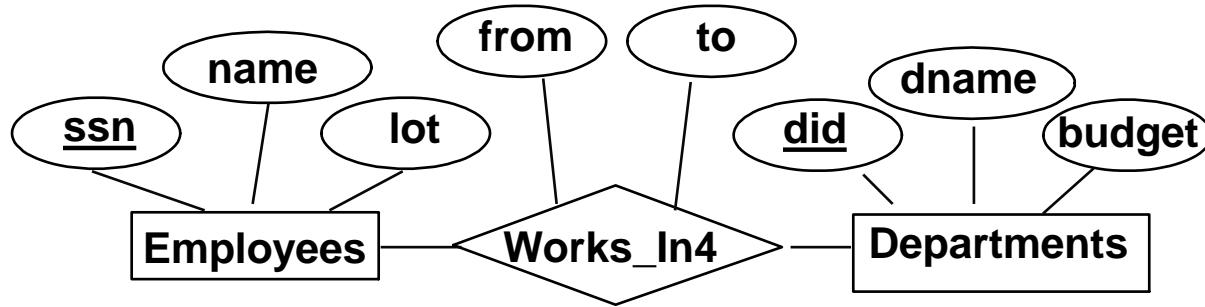
Desenho Conceptual

- Um conceito deverá ser modelado como entidade ou atributo? Como entidade ou relação?
- Devemos usar associações binárias ou ternárias?
- Devemos usar associações ternárias ou agregações?
- E os projetos complexos?
- Quais os erros comuns?
- Regras de Integridade

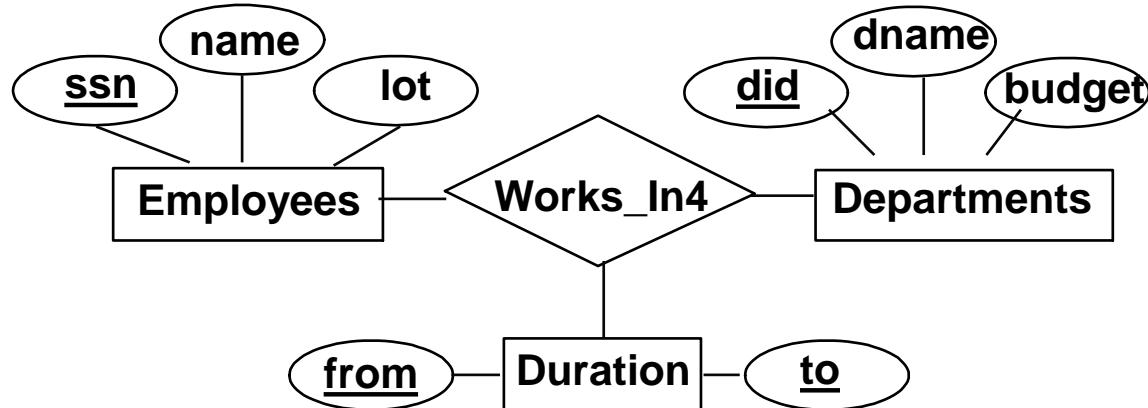
Entidade vs. Atributo

- Exemplo: endereço como atributo
 - Empregados têm vários endereços
 - Um endereço é composto pelos atributos:
Rua, cidade, país...
 - Alguns Empregados partilham o mesmo endereço
- Criar um conjunto de entidades a partir de um atributo quando:
 - Existe mais de um valor por cada entidade
 - O atributo tem a sua própria estrutura
 - Partilhado por vários conjuntos de entidades

Entidade vs. Atributo Descritivo



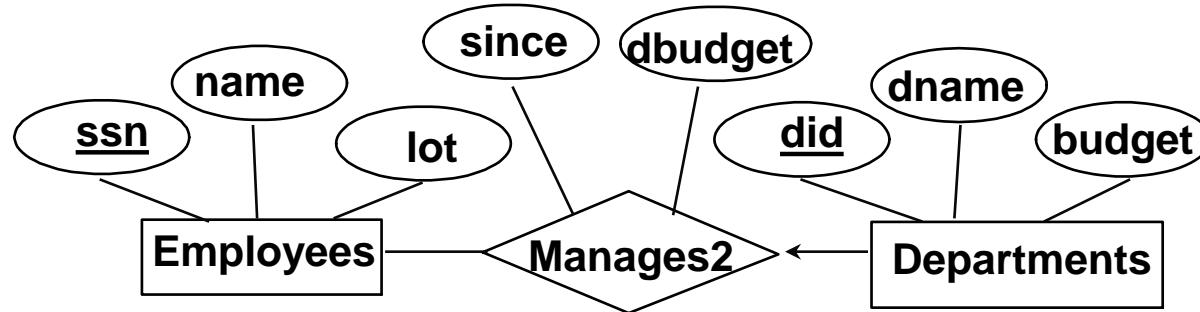
Se empregado puder trabalhar em vários períodos num departamento?



As duas soluções são possíveis. Mas no modelo relacional a solução é idêntica

Associação vs Entidade

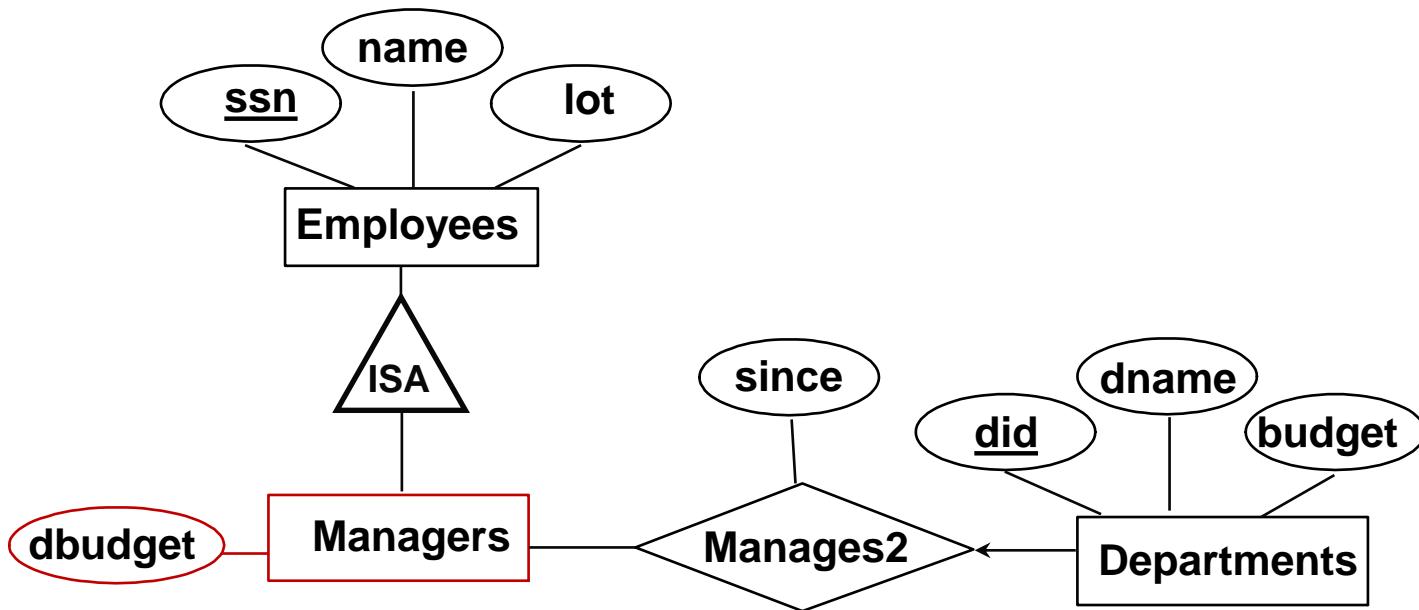
- Caso dos gestores de departamentos
 - A cada gestor é atribuído um **orçamento global** (dbudget)



- Problemas da solução
 - **Repetição** do mesmo dbudget para todos os departamentos de um gestor
 - Modelo leva a crer que o orçamento é específico de cada departamento

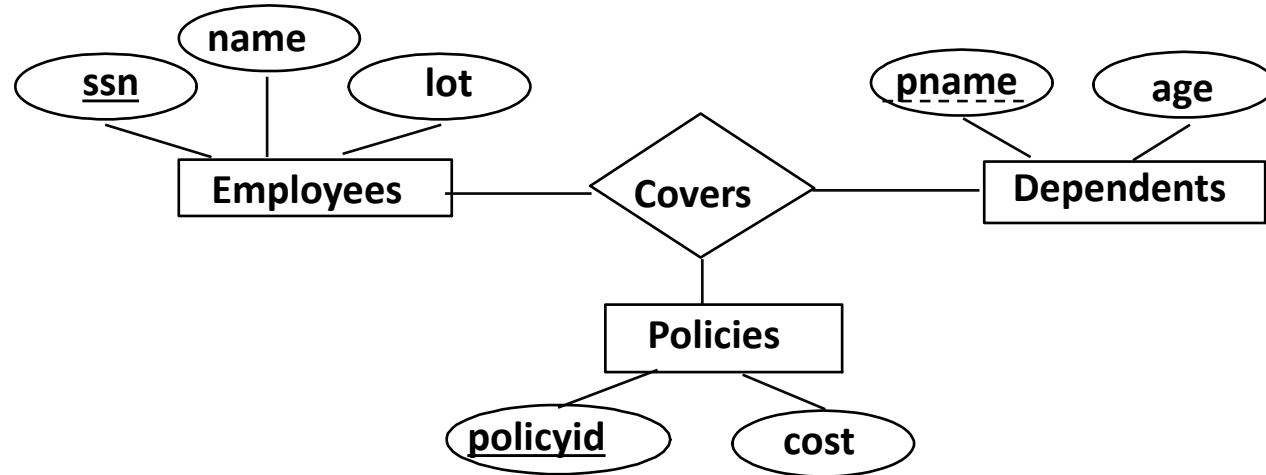
Associação vs Entidade

- Soluções
 - Colocar dbudget em Employees ?
 - Todos os empregados são gestores?

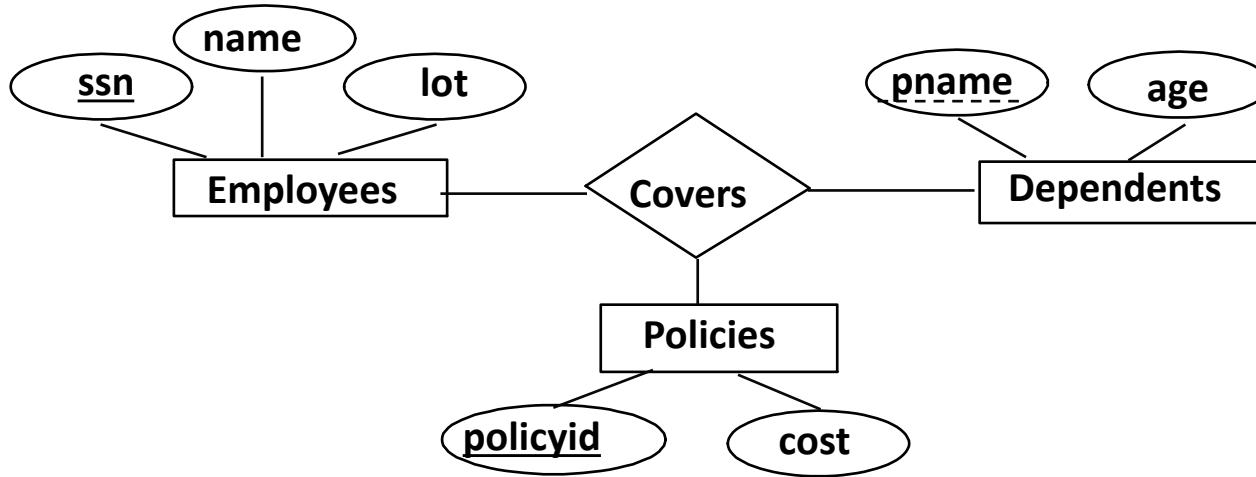


Associação Binária vs. Ternária

- Considere a solução para o caso dos dependentes e apólices de seguro

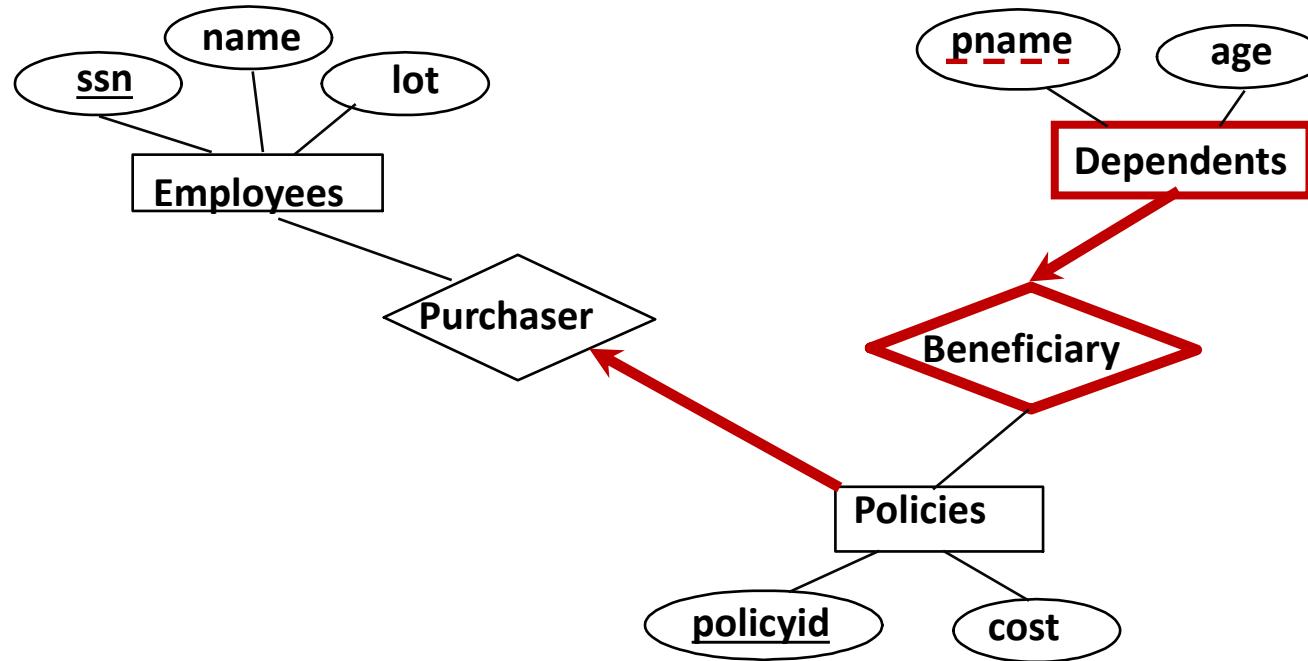


Associação Binária vs. Ternária



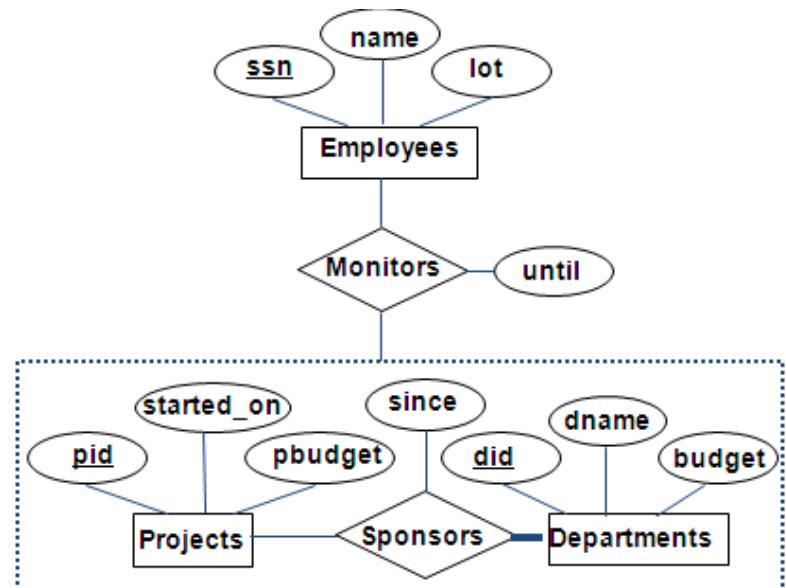
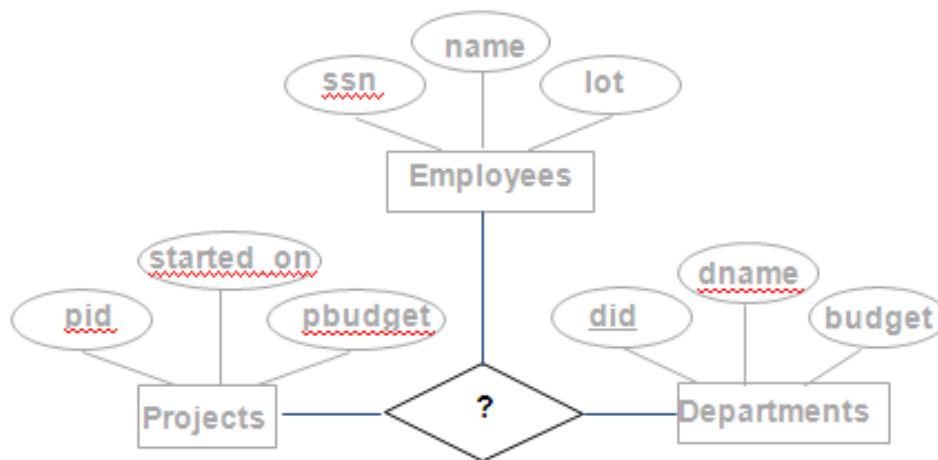
- Considerando as seguintes regras de integridade (RI) quais os problemas da solução
 - Apólice de seguro não pode pertencer a mais do que um empregado
 - Cada apólice tem de estar associada a um empregado
 - Dependentes são entidades fracas (dependem da apólice)

Associação Binária vs. Ternária



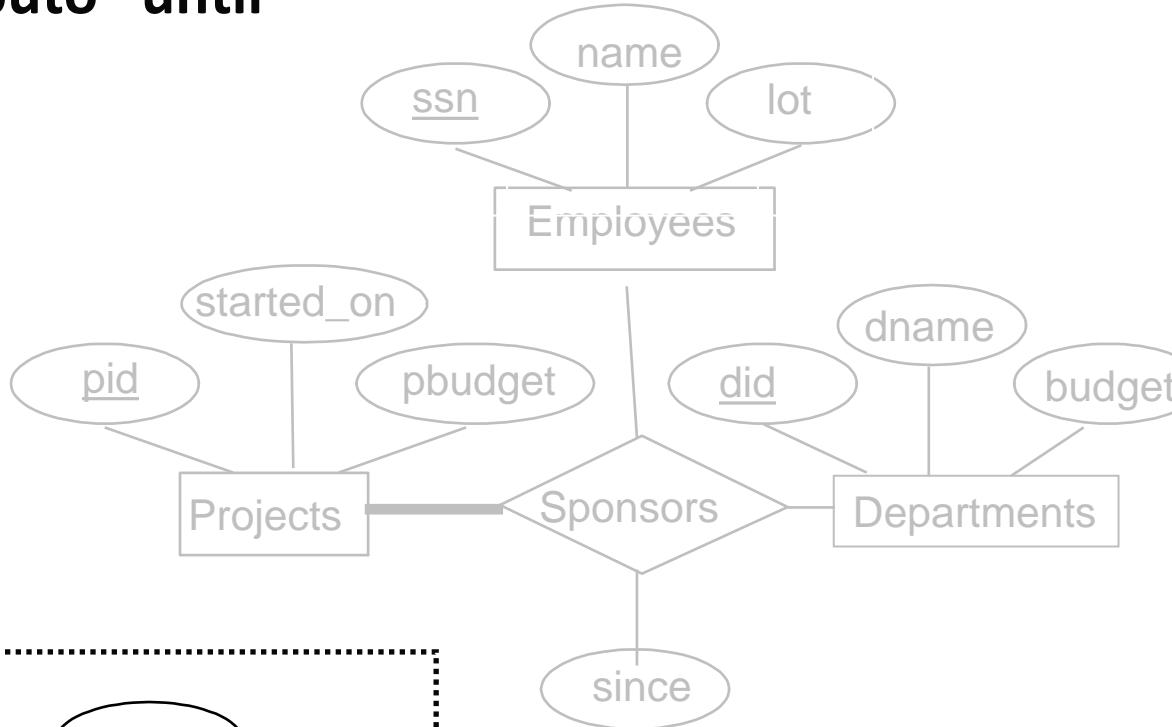
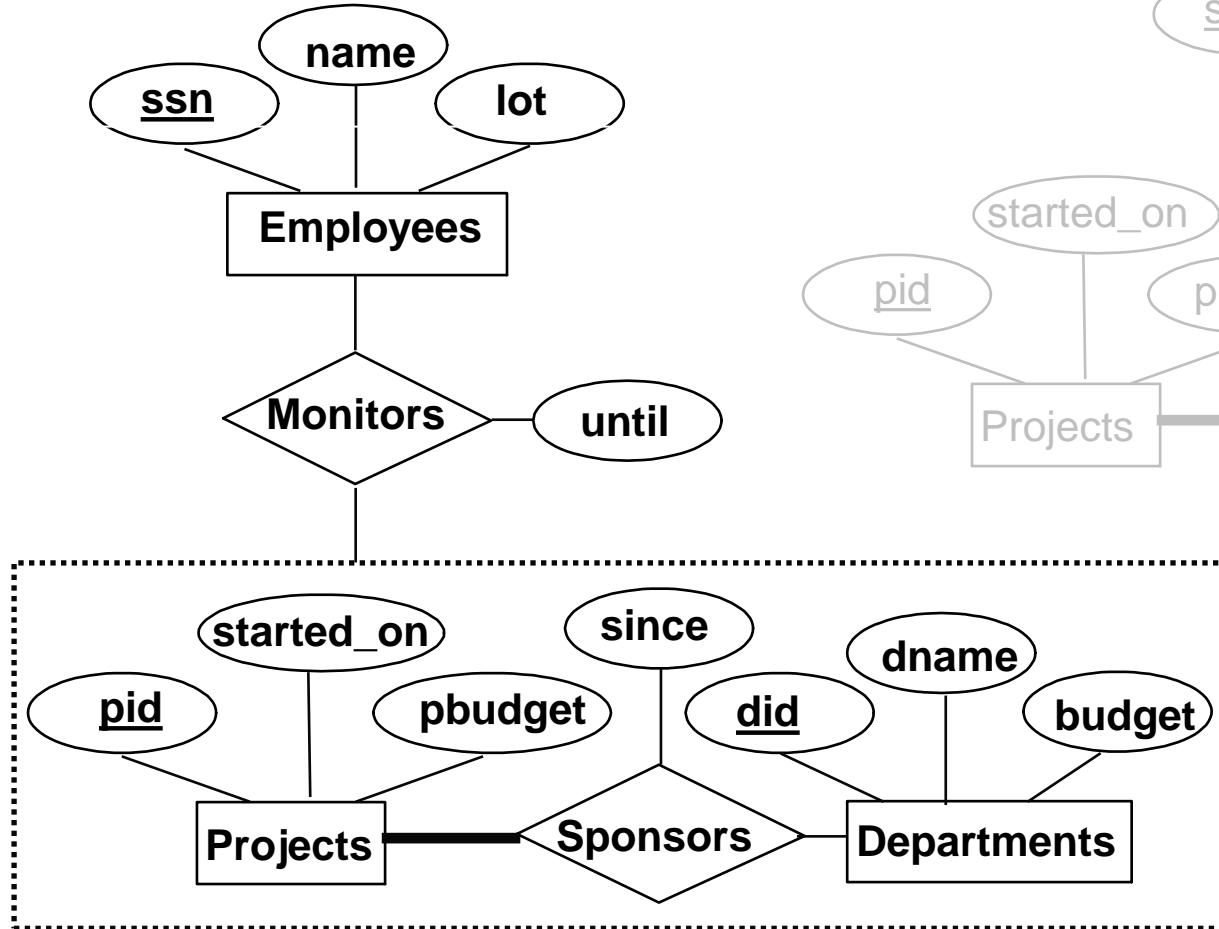
Agregação vs Ternária

Porque não definir uma associação ternária entre *Projects*, *Departments* e *Employees* ?



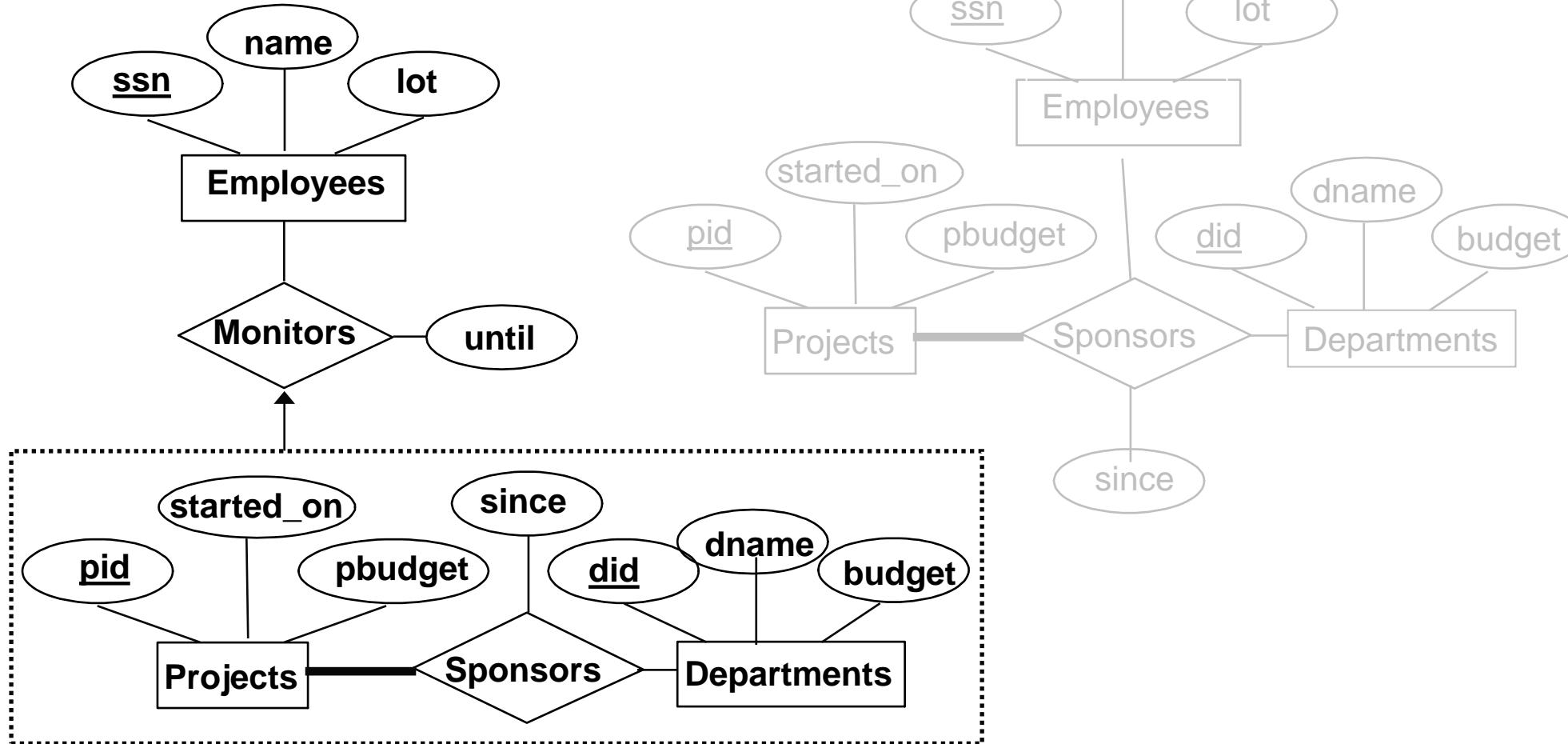
Agregação vs. Ternária

Necessidade de guardar o atributo “until”



Agregação vs. Ternária

Restrição: cada financiamento é monitorizado por um empregado



Agregação vs. Ternária

- Numa associação **ternária** é obrigatório ter **três** entidades:
 - (a,b,c)
- Numa **agregação**, as associações são **independentes**:
 - ((a,b),c)

Introdução às Bases de Dados

Modelo Entidade-Associação - IV

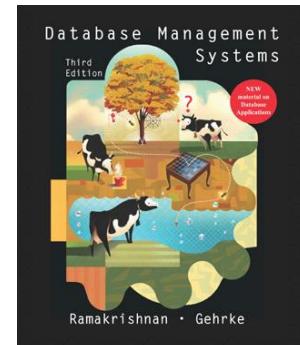
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Restrições de integridade
 - Projetos complexos
 - Erros comuns de modelação
 - Verificação do esquema conceptual
- Referências
 - R. Ramakrishnan (**capítulo 2**)

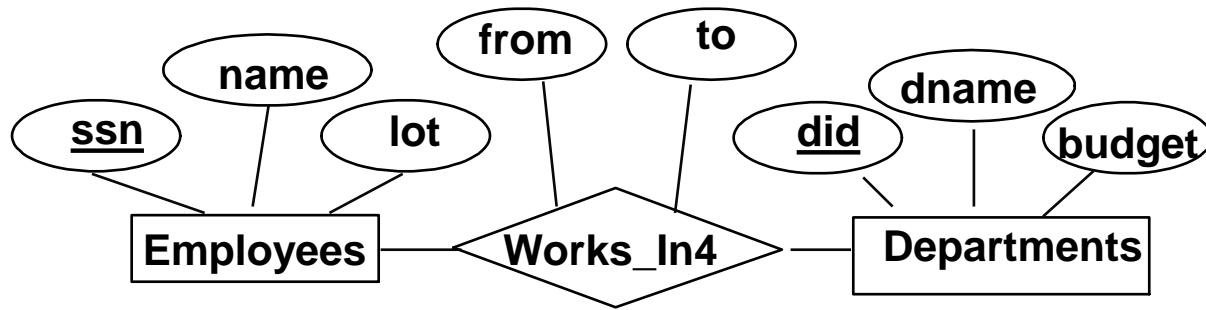


Regras de Integridade Adicionais

- Modelo EA é expressivo mas tem limitações
 - Poucas opções para o limite superior de participação numa associação
 - Restrição de chave limita a no máximo uma participação
 - Outros limites bem determinados (ex. 2, 3, ...) não se representam graficamente
 - Não suporta relações bem conhecidas entre valores de atributos
- **Restrições de integridade (RI)** adicionais escritas em forma de texto
 - Frases curtas, com os mesmos termos usados no diagrama
 - Numeradas (RI-1, RI-2, ...)
 - Agrupadas e colocadas logo abaixo do diagrama

Regras de Integridade Adicionais

Exemplo:



RI-1: Um empregado não pode estar num departamento por um período inferior a 3 meses.

Projetos Complexos

Melhor **metodologia** para **consistência**?

- Gerar uma lista única de requisitos é difícil
- **Separar** em várias partes (diagramas conceptuais)
 - Para diferentes utilizadores
 - E depois **integrar** os diagramas num só
 - É necessário encontrar correspondências entre as entidades, associações, atributos, e resolver os conflitos
- A maior parte dos Sistemas de Informação (SI) acedem hoje a vários conjuntos de dados **heterogéneos**

Erros Comuns I

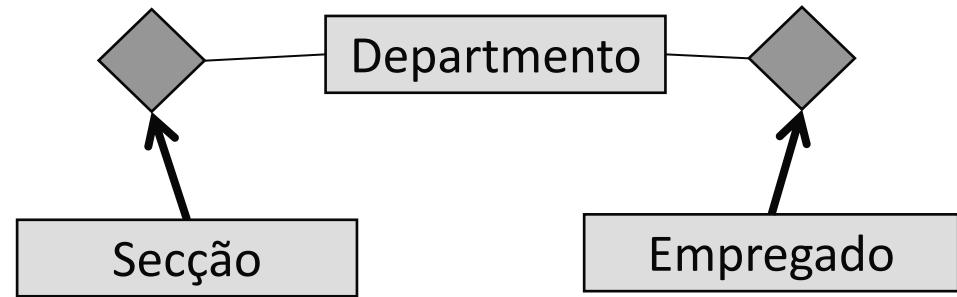
- **Idade** em vez de data de nascimento
- **Entidades** sem atributos
 - Sem chave primária
 - Com várias chaves primárias
- Entidades modeladas como atributo
 - Usada para associação implícita
- **RI** que podem ser mapeadas no EA
- Malhas sem RI

Erros Comuns II

- **Entidades fracas** sem:
 - Entidade que depende (forte)
 - Restrições de chave e participação
- **Associações ternárias** que podiam ser mapeadas como binárias
- **Agregações** mal representadas
- **Generalizações**
 - que podiam ser representadas como atributos
 - sem restrições de cobertura e sobreposição (RI)
- Texto com restrições de integridade que já vêm no diagrama

Verificação do Esquema I

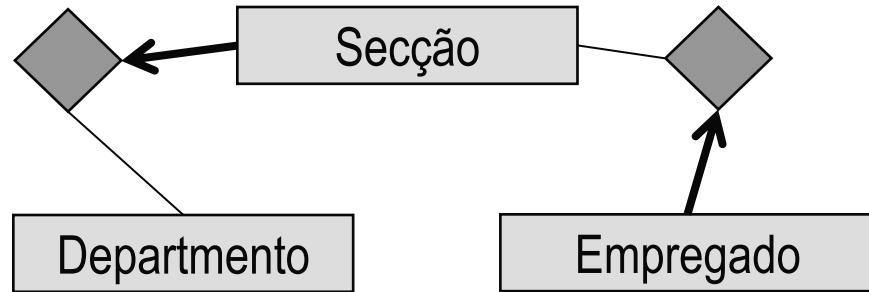
- Esquema permite obter:
 - O departamento de uma secção
 - O departamento de um empregado
 - As secções de um departamento
 - Os empregados de um departamento



- Problemas:
 - Quais os empregados de uma secção ?
 - Qual a secção de um empregado ?

Verificação do Esquema II

- O esquema permite:
 - Os empregados de uma secção
 - A secção de um empregado



- Problemas:
 - Quais os empregados que trabalham diretamente para um departamento (sem secção)

Verificação do Esquema III

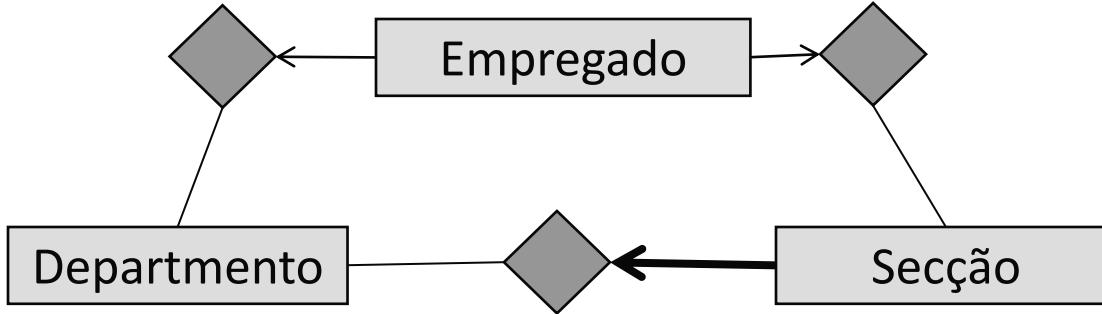
- Agora conseguimos:
 - Empregados que trabalham diretamente para um departamento (sem secção)



- Problemas:
 - Qual o departamento de uma secção (que não tem empregados)?

Verificação do Esquema IV

- A solução mais geral



- Mas pode conter RI associadas à **malha**
- Verificar se deve ser utilizada uma agregação

Introdução às Bases de Dados

Modelo Relacional - I

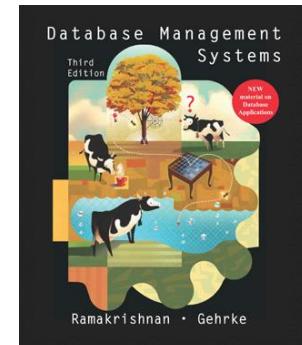
FCUL, Departamento de Informática

Ano Letivo 2021/2022

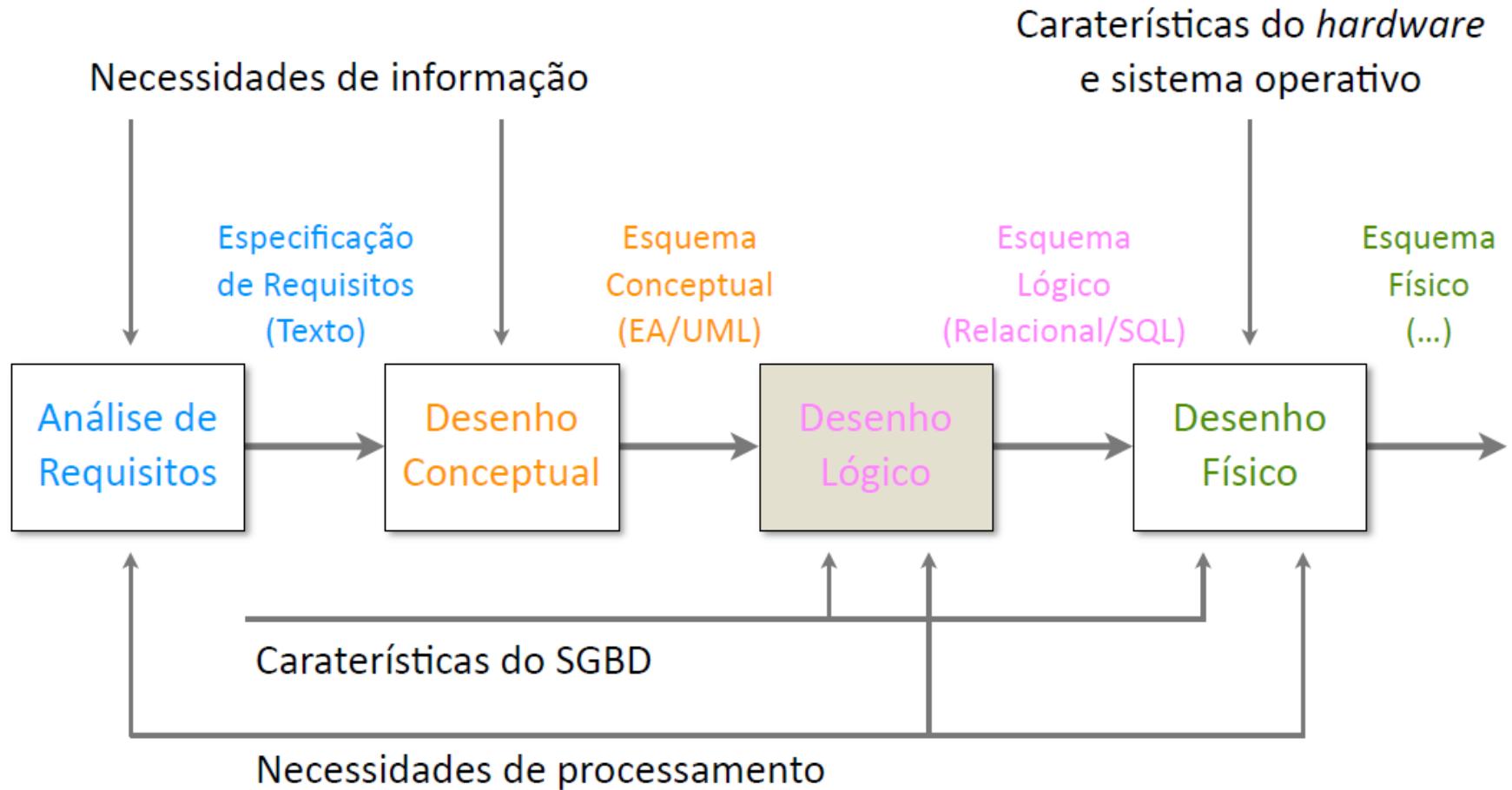
Ana Paula Afonso

Sumário e Referências

- Sumário
 - Enquadramento no processo de desenho de BD
 - Vertentes do modelo relacional
 - Estruturas de dados
 - Operadores relacionais
 - Restrições de integridade
 - Propriedades das relações
 - Comandos SQL
 - Criação de tabelas
 - Inserção, atualização e remoção de dados
- Referências
 - R. Ramakrishnan (**capítulo 3, secção 3.1**)



Processo de Desenho de BDs



Fonte: António Ferreira, Guião SIBD, 2019

Modelo Relacional: História

- Criado em 1970 por Edgar Codd
 - Muito simples e elegante
 - Estrutura de dados, operadores relacionais e regras de integridade
 - Modelo formal com forte **base matemática** a suportá-lo
 - Utilizado pela esmagadora **maioria dos SGBD**
1º comercial: Oracle; Outros: MS SQL Server, IBM DB2, MySQL, ...
- Não foi o primeiro modelo de dados
 - Nos anos 1960 já existia o modelo hierárquico e o modelo em rede
 - BD baseados em modelos de dados orientados a objetos alternativa, ainda problemas de concretização

Modelo Relacional: Vertentes

<i>sid</i>	<i>name</i>	<i>lZogin</i>	<i>age</i>	<i>gpa</i>
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- Estrutura de dados
 - Base de dados é uma **coleção de relações**
 - **Relação** é uma **tabela** com **linhas** e **colunas**
 - Cada coluna tem valores de um **domínio** de dados
- Operadores relacionais
 - Para gestão de tabelas e outras estruturas
 - Para inserção, remoção e pesquisa de dados
- Regras de Integridadde
 - Para garantia da coerência dos dados
 - Ex. um aluno para se inscrever em disciplinas tem de estar matriculado

SQL – Structured Query Language

- Linguagem universal/norma para o modelo relacional
 - 1986, norma do American National Standards Institute (ANSI)
 - 1987, também pela International Standard Organization (ISO)
- Apesar de ser uma norma (*standard*) os SGBD podem introduzir características específicas
- Componente nuclear estável, mas em evolução
 - 1986: primeira formalização do SQL standard (SQL-86)
 - 9 revisões:
 - SQL-89; SQL-92
 - SQL-99 uma grande extensão ao SQL-92
 - SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL:2016, SQL:2019

Relações

- Relações são **tabelas** com linhas e colunas
 - colunas = atributos = campos
 - linhas = registos = entidades = tuplos
- Esquema de relação inclui
 - Nome da relação, Nome e domínio de dados de cada coluna

Ex: *Students (sid: integer, name: string, login: string, age: integer, gpa: real)*

- Instância da relação
 - Conjunto de tuplos

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53831	Madayan	madayan@music	11	1.8
53832	Guldu	gllldll@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

Propriedades das Relações

- **Restrições de domínio**
 - Especificam o tipo de dados de cada atributo
 - Todas as colunas têm de ter um domínio
 - Ex. *string, integer, real*
 - SGBD oferecem domínios específicos
 - Ex. *char(n), int, smallint, number(n,m)*
- O **grau** da relação = número de colunas
- A **cardinalidade** de uma relação = número de linhas

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53831	Madayan	madayan@music	11	1.8
53832	Guldu	gllldll@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

Base de Dados

- **Base de dados (BD) relacional**
 - coleção de relações com nomes distintos
- **Esquema relacional da BD**
 - coleção dos esquemas de todas as relações
- **Instância da BD relacional**
 - coleção das instâncias de todas as relações

Síntese de Comandos SQL

- **SQL-DDL:** *Data Definition Language*
 - operações sobre a estrutura das tabelas e gestão de restrições de integridade
 - CREATE TABLE
 - DROP TABLE
 - ALTER TABLE
- **SQL-DML:** *Data Manipulation Language*
 - operações sobre os **dados** das tabelas
 - INSERT INTO
 - DELETE FROM
 - UPDATE
 - SELECT

CREATE TABLE: Criação de Tabelas

Dado um esquema de relação

Students (sid: integer, name: string, login: string, age: integer, gpa: real)

O comando CREATE TABLE cria uma nova tabela sem dados

```
CREATE TABLE Students ( sid integer,  
                      name char(30),  
                      login char(30),  
                      age integer,  
                      gpa real)
```

INSERT: Inserção de Linhas

INSERT

```
INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

UPDATE: Atualização de valores

```
UPDATE Students S  
SET      S.age = S.age + 1, S.gpa = S.gpa - 1  
WHERE    S.sid = 53688
```

Exemplo: UPDATE

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53831	Madayan	madayan@music	11	1.8
53832	Guldu	gllldll@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

UPDATE Students S
SET S.gpa = S.gpa - 0.1
WHERE S.gpa >= 3.3

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

DELETE: Apagar Linhas

```
DELETE  
FROM Students S  
WHERE S.name = 'Smith'
```

Introdução às Bases de Dados

Modelo Relacional - II

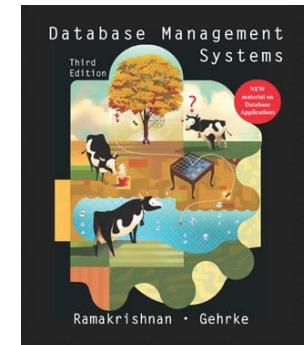
FCUL, Departamento de Informática

Ano Letivo 2020/2019

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Restrições de Integridade
 - Domínio e coluna
 - Entidade ou chave
 - Referencial ou chave estrangeira
 - Violações às Restrições de Chave
 - Verificação de Restrições de Integridade
- Referências
 - R. Ramakrishnan (**capítulo 3, secção 3.2, 3.3 e**
capítulo 5, secção 5.7)



Restrições de Integridade

- Condições especificadas sobre o esquema da BD
 - restringe os dados que podem ser armazenados numa instância
 - impede o armazenamento de informação incorreta ou incoerente
- Verificação automática pelo SGBD
- Tipos de restrições
 - Domínio
 - Coluna
 - Entidade
 - Referencial
 - Aplicacional (Adicional)

Integridade de Domínio e de Coluna

- Integridade de **domínio**
 - Cada coluna de uma tabela tem um domínio de dados (ex. INTEGER)
 - **Todos os valores dessa coluna têm de pertencer ao mesmo domínio**
- Integridade de **coluna**
 - Refinamento da integridade de domínio
 - Permite limitar gama de valores admissíveis
- Ex. tabela de empregados
 - Integridade de **domínio**
Número e vencimento são números
 - Integridade de **coluna**
Número: inteiro positivo, até 5 dígitos
Vencimento: real positivo, até 6 dígitos para parte inteira e 2 para decimal

```
CREATE TABLE Empregado (  
    numero INTEGER(5),  
    ...  
    vencimento DECIMAL(8,2),  
    PRIMARY KEY (numero),  
    CHECK (numero > 0),  
    CHECK (vencimento > 0,0))
```

SQL Tipos de Dados

<http://www.w3schools.com/sql/>

Data type	Description
CHARACTER(n)	Character string. Fixed-length n
VARCHAR(n) or CHARACTER VARYING(n)	Character string. Variable length. Maximum length n
BINARY(n)	Binary string. Fixed-length n
BOOLEAN	Stores TRUE or FALSE values
VARBINARY(n) or BINARY VARYING(n)	Binary string. Variable length. Maximum length n
INTEGER(p)	Integer numerical (no decimal). Precision p
SMALLINT	Integer numerical (no decimal). Precision 5
INTEGER	Integer numerical (no decimal). Precision 10
BIGINT	Integer numerical (no decimal). Precision 19
DECIMAL(p,s)	Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
NUMERIC(p,s)	Exact numerical, precision p, scale s. (Same as DECIMAL)

SQL Tipos de Dados

<http://www.w3schools.com/sql>

Data type	Description
FLOAT(p)	Approximate numerical, mantissa precision p. A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision
REAL	Approximate numerical, mantissa precision 7
FLOAT	Approximate numerical, mantissa precision 16
DOUBLE PRECISION	Approximate numerical, mantissa precision 16
DATE	Stores year, month, and day values
TIME	Stores hour, minute, and second values
TIMESTAMP	Stores year, month, day, hour, minute, and second values
INTERVAL	Composed of a number of integer fields, representing a period of time, depending on the type of interval
ARRAY	A set-length and ordered collection of elements
MULTISET	A variable-length and unordered collection of elements
XML	Stores XML data

MySQL Tipos de Dados

http://www.w3schools.com/sql/sql_datatypes.asp

- Numeric
 - INT(size), INTEGER(size), SMALLINT
 - DECIMAL(size, d), NUMERIC
 - Ex: salario NUMERIC(5,2) valores da coluna variam entre -999.99 e 999.99
 - ...
- Text
 - CHAR(size), VARCHAR(size)
 - BLOB
 - ...
- Date e Time

SGBD MySQL – Tipo de Dados DATE

http://www.w3schools.com/sql/sql_datatypes.asp

Data type	Description
DATE()	A date. Format: YYYY-MM-DD Note: The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS Note: The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS Note: The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS Note: The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

Restrições de Chave

- **Chaves candidatas** são colunas (atributos) com as seguintes propriedades
 - **Unicidade**: os seus valores identificam univocamente qualquer tuplo de uma instância, i.e., dois tuplos distintos não podem ter valores iguais para os atributos da chave
 - **Minimalidade**: conjunto mínimo de atributos que identificam univocamente qq tuplo de uma instância, i.e, nenhum sub-conjunto de atributos da chave pode ser uma chave

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53831	Madayan	madayan@music	11	1.8
53832	Guldu	gllldll@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

Chaves Candidatas e Primária

- Podem existir várias **chaves candidatas** por relação
 - Existe sempre uma chave candidata
- **Chave primária** é uma das chaves candidatas selecionada como a principal
 - A que otimiza a referência
 - Normalmente do tipo INTEGER
- Exemplo em SQL

```
CREATE TABLE Empregado (
    nid      INTEGER(4) PRIMARY KEY,
    nif      INTEGER(9) UNIQUE NOT NULL, ...)
```

↗ **chave candidata**

Chave Primária Composta

```
CREATE TABLE Emp_HDIA (
    nid          INTEGER (4) ,
    dia          DATE,
    horas        DECIMAL (3,1),
    PRIMARY KEY (nid, dia))
```

Nome de uma Restrição

```
CREATE TABLE Emp_HDIA (
    nid          INTEGER (4) ,
    dia          DATE,
    horas        DECIMAL (3,1),
    CONSTRAINT pk_horas_dia PRIMARY KEY (nid, dia))
```



Nome da restrição

Caso a restrição seja violada o SGBD indica este nome

Integridade Referencial

- Chave estrangeira
 - Restrição de integridade que envolve duas tabelas
 - Denominada **restrição de integridade referencial**
 - Coluna(s) cujos valores provêm da chave primária de outra tabela
 - Se os dados de uma relação são alterados, as outras relações devem ser verificadas para manter os dados consistentes

Exemplo de Chave Estrangeira

Students (**sid**: integer, name: string, login: string, age: integer, gpa: real)

Enrolled (**studid**: integer, cid: string, grade: string)

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae203	B	53832	53666	Jones	jones@cs	18	3.4
Topology112	A	53650	53688	Smith	smith@ee	18	3.2
History105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation)

Tabela referenciadora

Students (Referenced relation)

Tabela referenciada

Chave Estrangeira em SQL

```
CREATE TABLE Enrolled (  
    studid INTEGER(10),  
    cid     CHAR(20),  
    grade   CHAR(1),  
    PRIMARY KEY (studid, cid),  
    FOREIGN KEY (studid) REFERENCES Students (sid) )
```

Tabela referenciadora

```
CREATE TABLE Students (  
    sid    INTEGER(4) PRIMARY KEY,  
    name   VARCHAR(50), ...)
```

Tabela referenciada

Propriedades da Chave Estrangeira

- Cada valor de *studid* que aparece na tabela *Enrolled* tem de aparecer na coluna da chave primária da tabela *Students*

```
CREATE TABLE Enrolled(
    studid INTEGER(10),
    cid    CHAR(20),
    grade  CHAR(1),
    PRIMARY KEY (studid, cid),
    FOREIGN KEY (studid) REFERENCES Students (sid))
```

- Operações que podem originar violações
 - Inserir linhas em *Enrolled*
 - Remover linhas de *Students*
- A chave estrangeira pode referenciar a própria tabela
 - Ex: adicionar coluna *partner* à tabela *Students*
 - Mas se o aluno não tem *partner*?

Valor NULL

- NULL indica que para aquele campo o valor é **desconhecido ou não aplicável**
 - Por exemplo quando não existe *partner*
- NULL **pode** aparecer numa **chave estrangeira** sem violar a restrição de integridade referencial
 - Se a chave estrangeira for constituída por várias colunas, ou estão todas a NULL ou nenhuma
- NULL **não pode** aparecer na **chave primária**

Restrições Gerais

- Exemplo: as idades dos estudantes têm de ser maiores que 18
 - CHECK (age > 18)
- O SGBD rejeita remoções/atualizações que violem as restrições:
 - Restrições de tabela: envolvem uma única tabela
 - Asserções: envolvem várias tabelas
- Irão ser descritas mais à frente:
 - *SQL Constraints and Assertions (chapter 5)*

Introdução às Bases de Dados

Modelo Relacional - III

Passagem do EA para Relacional

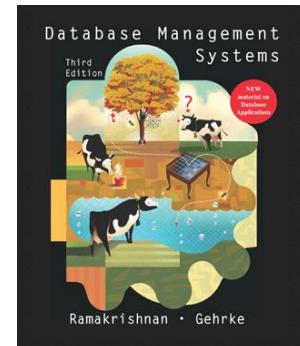
FCUL, Departamento de Informática

Ano Letivo 2021/2022

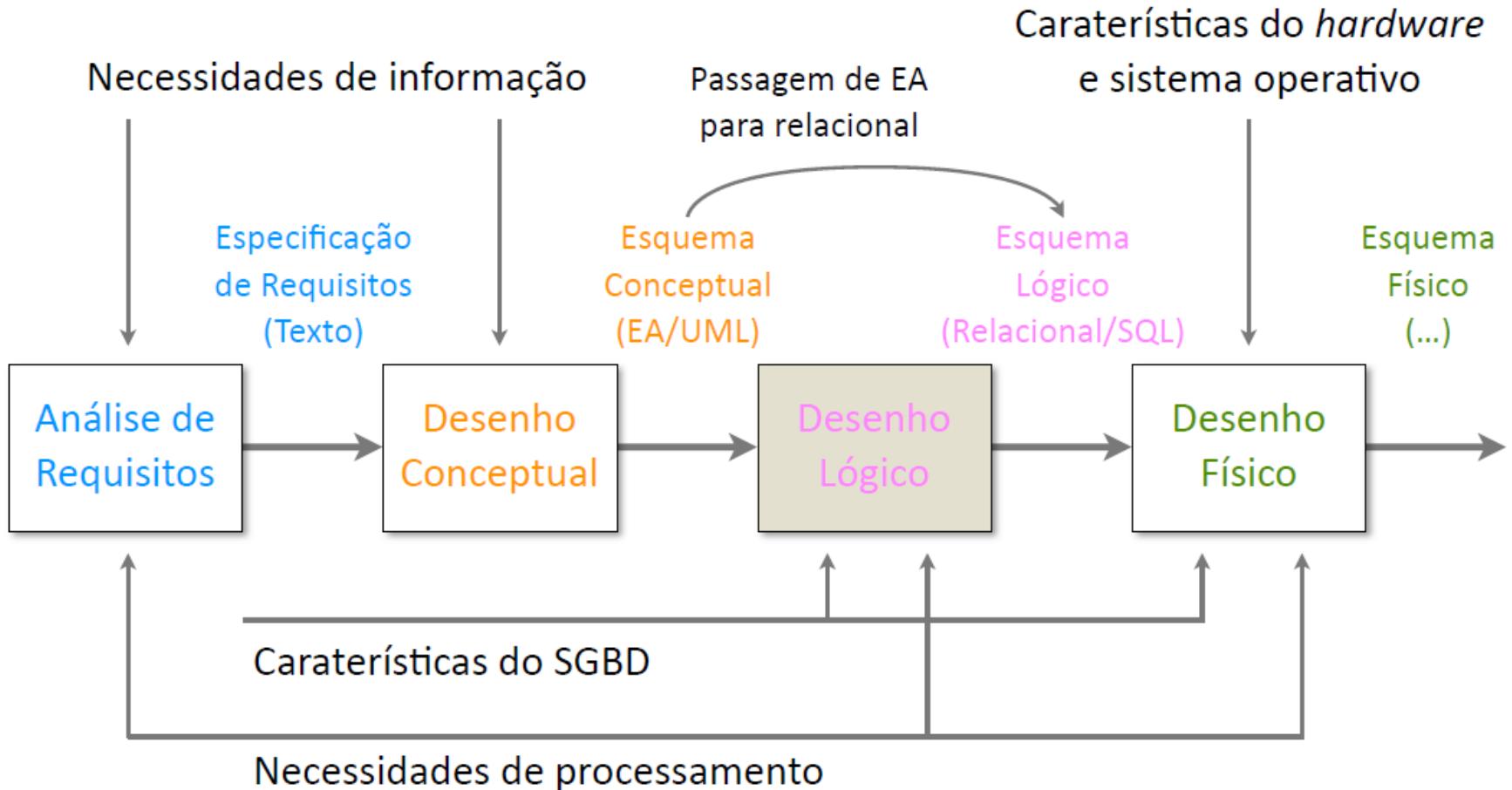
Ana Paula Afonso

Sumário e Referências

- Sumário
 - Passagem de EA para Relacional
 - Enquadramento no processo de desenho de BD
 - Entidades para Tabelas
 - Transformação de Associações
 - Associações com Restrições de Chave
 - Restrições de Participação
 - Entidades Fracas
 - Generalizações
 - Agregações
- Referências
 - R. Ramakrishnan (**capítulo 3, secção 3.5**)



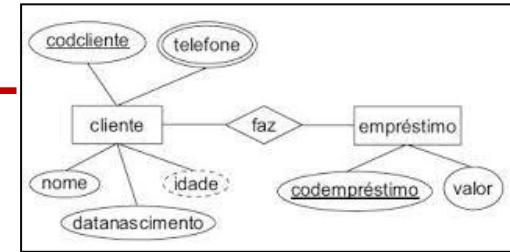
Processo de Desenho de BDs



Fonte: António Ferreira, Guião SIBD, 2016

Passagem de EA para Relacional

- Modelo entidade-associação (EA)
 - Adequado para o desenho inicial, de **alto nível**, da base de dados
 - Representação gráfica para facilitar discussão de alternativas por equipas
 - Mas não entendido pelos sistemas de gestão de bases de dados (SGBD)
- Modelo relacional
 - Suportado pelos SGBDs relacionais, muito populares
 - Mas de **baixo nível**, com comandos de texto, que dificultam discussão
 - Maior risco de perder a visão do todo, focando apenas nas partes
- Após discussão de alternativas e integração de diagramas EA
 - Esquema EA é traduzido num esquema relacional (ER) **aproximado**
 - Com **tabelas e restrições de integridade** escritas na linguagem SQL
 - Algumas restrições de integridade EA podem não ser concretizadas em SQL



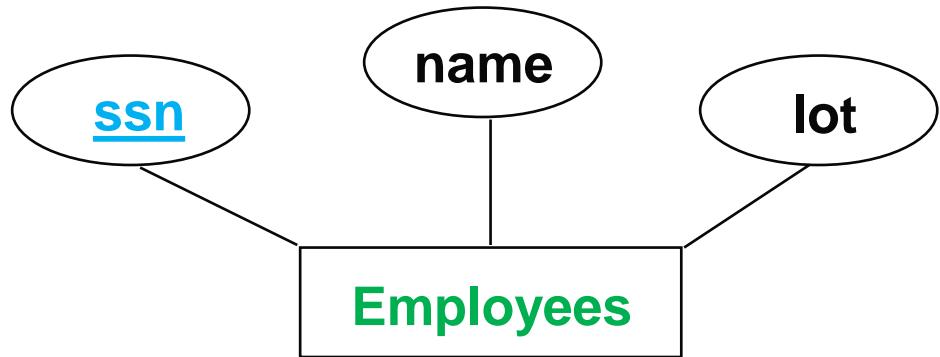
```
CREATE TABLE cliente (  
    codcliente integer,  
    ....)
```

Entidades para ER

Nome da tabela igual ao nome da entidade

Colunas da tabela são os atributos da entidade

Chave primária da tabela vem da chave primária da entidade



```
CREATE TABLE Employees (
    ssn CHAR(11),
    name CHAR(30),
    lot INTEGER,
    PRIMARY KEY (ssn)
)
```

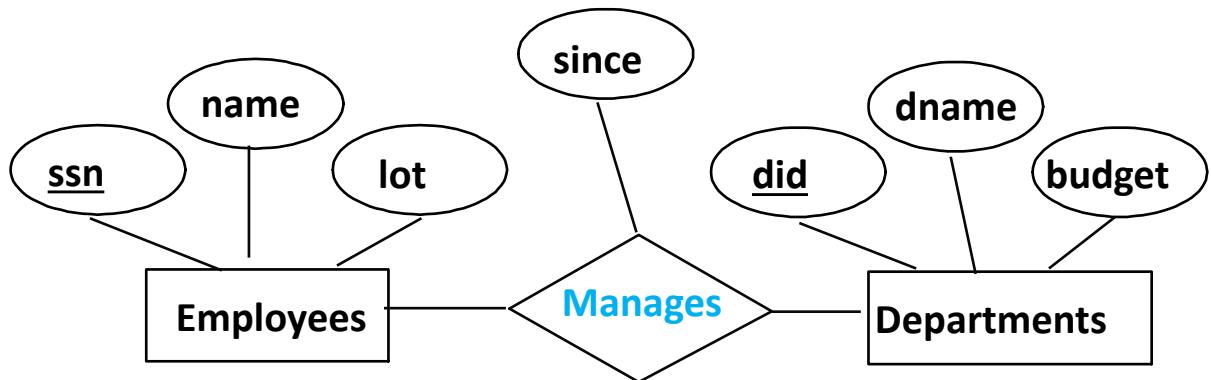
Associações para ER

Caso 1: sem restrições de chave e participação

Nome da tabela igual ao nome da associação

Chave primária da tabela é **composta** pelas chaves primárias das entidades participantes

Chaves estrangeiras referenciam as entidades

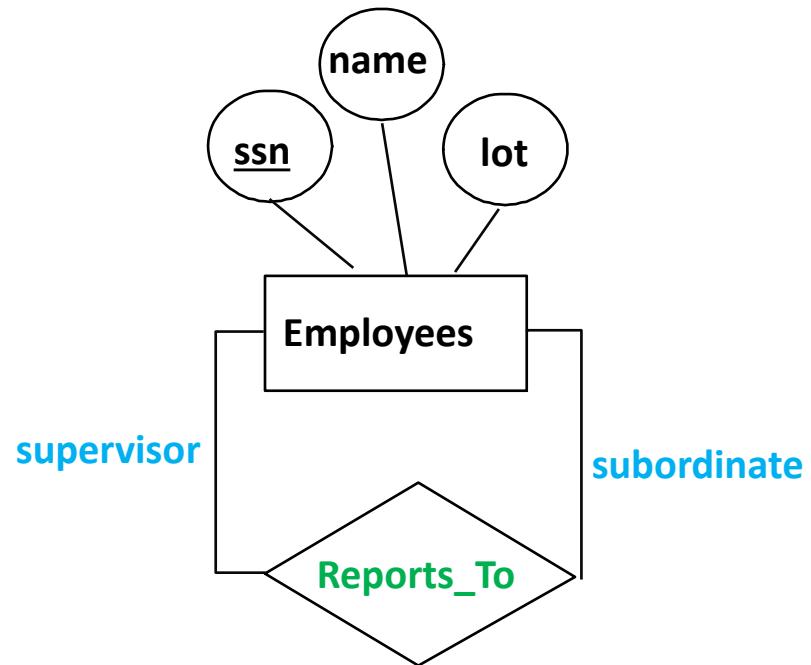


```
CREATE TABLE Manages (
    ssn  CHAR(11),
    did  INTEGER,
    since DATE,
    PRIMARY KEY (did, ssn),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

Associações para ER

Caso 1: sem restrições de chave e participação

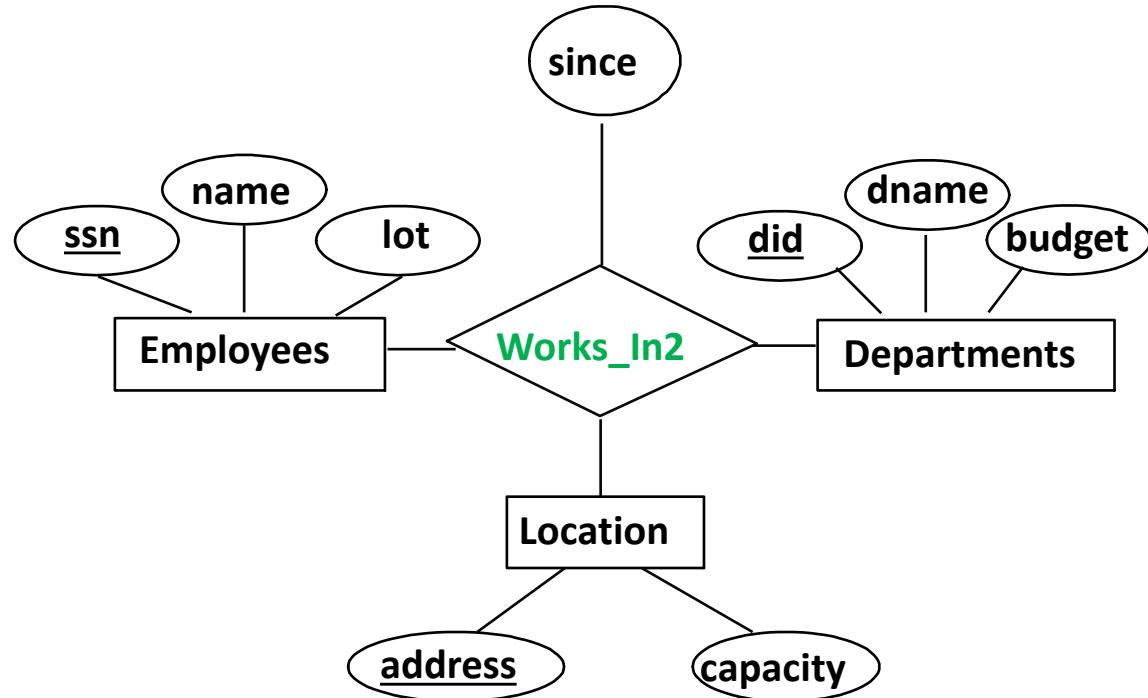
Papéis ajudam a dar nome às colunas



```
CREATE TABLE Reports_To (
    supervisor_ssn CHAR (11),
    subordinate_ssn CHAR (11),
    PRIMARY KEY (supervisor_ssn, subordinate_ssn),
    FOREIGN KEY (supervisor_ssn) REFERENCES Employees,
    FOREIGN KEY (subordinate_ssn) REFERENCES Employees)
```

Associações para ER

Caso 1: sem restrições de chave e participação

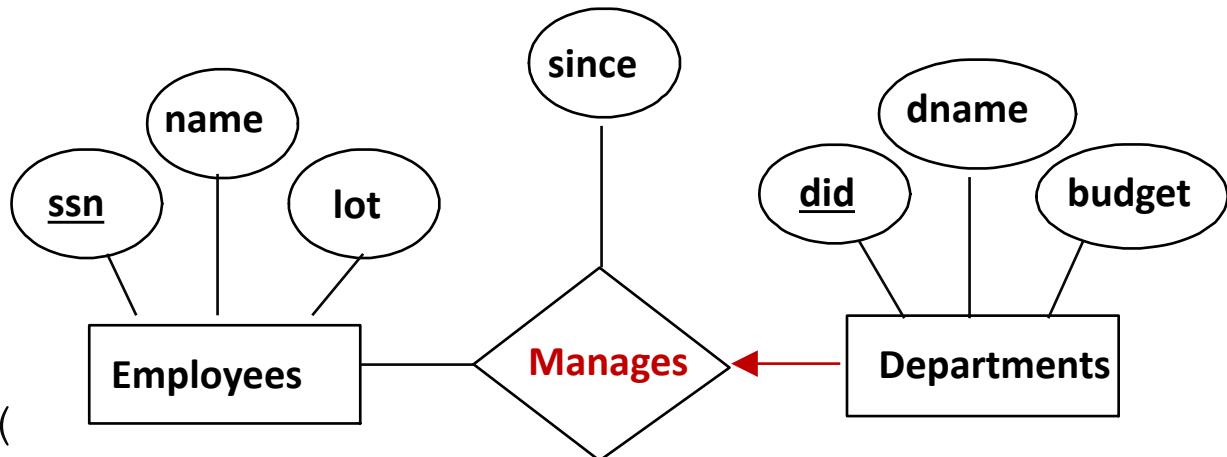


```
CREATE TABLE Works_In2 (
    ssn CHAR(11),
    did INTEGER,
    address CHAR(20),
    since DATE,
    PRIMARY KEY (ssn, did, address),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (address) REFERENCES Location,
    FOREIGN KEY (did) REFERENCES Departments)
```

Associações para ER

Caso 2: com restrição de chave

Abordagem 1: criação de uma nova tabela



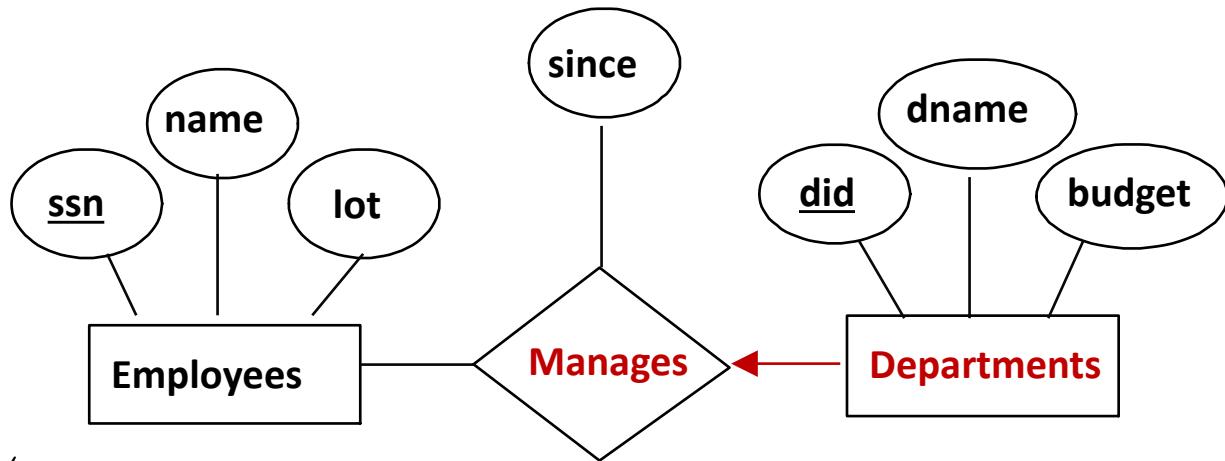
```
CREATE TABLE Manages (
    ssn  CHAR(11) NOT NULL,
    did  INTEGER,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

Nota: (ssn,did) como chave não cumpre a regra de ser o conjunto mínimo

Associações para ER

Caso 2: com restrição de chave

Abordagem 2: **Adição de chave estrangeira** à tabela Dept existente



```
CREATE TABLE Dept_Mgr (
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11),
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees)
```

Associações para ER

Caso 2: com restrição de chave

Duas abordagens

1. Criar uma nova tabela

Vantagens

- Atributos descritivos da associação na **sua própria tabela**
- **Restrição de participação parcial** facilmente suportada: basta não inserir linhas na tabela

Desvantagens

- **Mais uma tabela** no esquema relacional torna pesquisas mais complexas
- Restrição de participação total **custosa**: necessárias asserções

Usar em casos de associações com muitos atributos descritivos

Associações para ER

Caso 2: com restrição de chave

Duas abordagens

2. Adicionar chave estrangeira à tabela existente

Vantagens

- **Menos uma tabela** no esquema relacional permite pesquisas mais simples
- **Restrição de participação total** facilmente suportada: basta usar NOT NULL

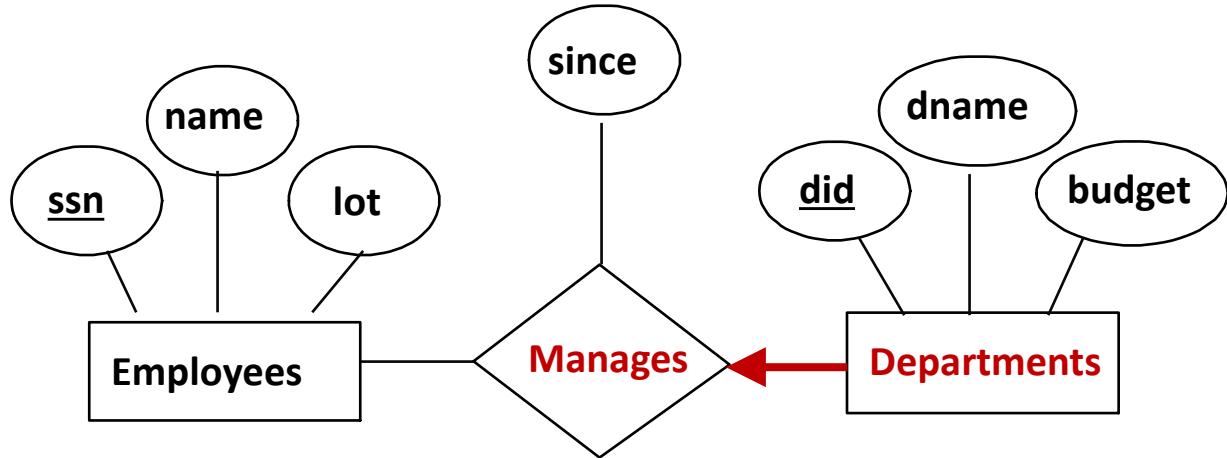
Desvantagens

- **Mistura atributos** da associação e entidade na mesma tabela
- Restrição de participação parcial pode levar a **muitos valores nulos** nas linhas da tabela

Associações para ER

Caso 3: com restrição de chave e participação

Adição de chave estrangeira à tabela existente e restrição NOT NULL



```
CREATE TABLE Dept_Mgr (
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11) NOT NULL,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees ON DELETE NO ACTION)
```

NO ACTION

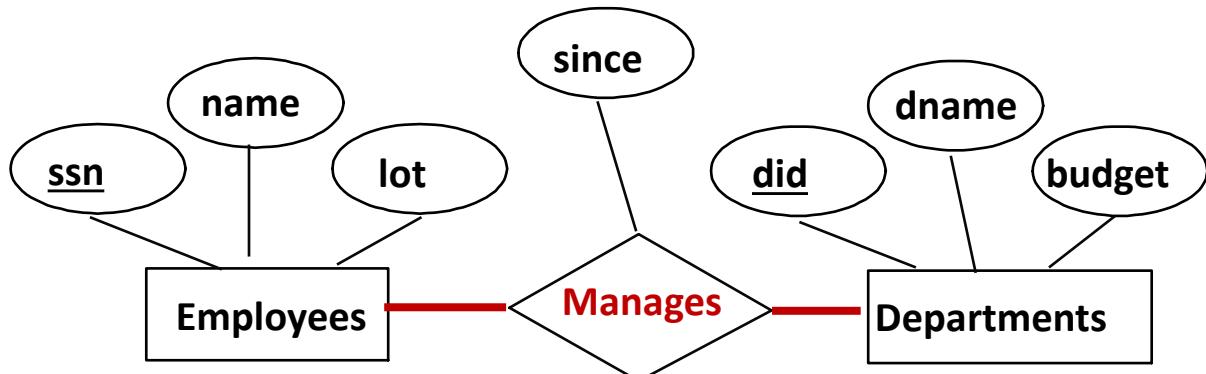
ação por defeito. Um empregado não pode ser removido se tiver um Dept_Mgr a referenciá-lo

Associações para ER

Caso 4: com restrição de participação

Criação de uma nova tabela e asserção

```
CREATE TABLE Manages (
    ssn  CHAR(11),
    did  INTEGER,
    since  DATE,
    PRIMARY KEY  (did, ssn),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```



Nota: Por agora, Aserção é uma restrição de integridade textual

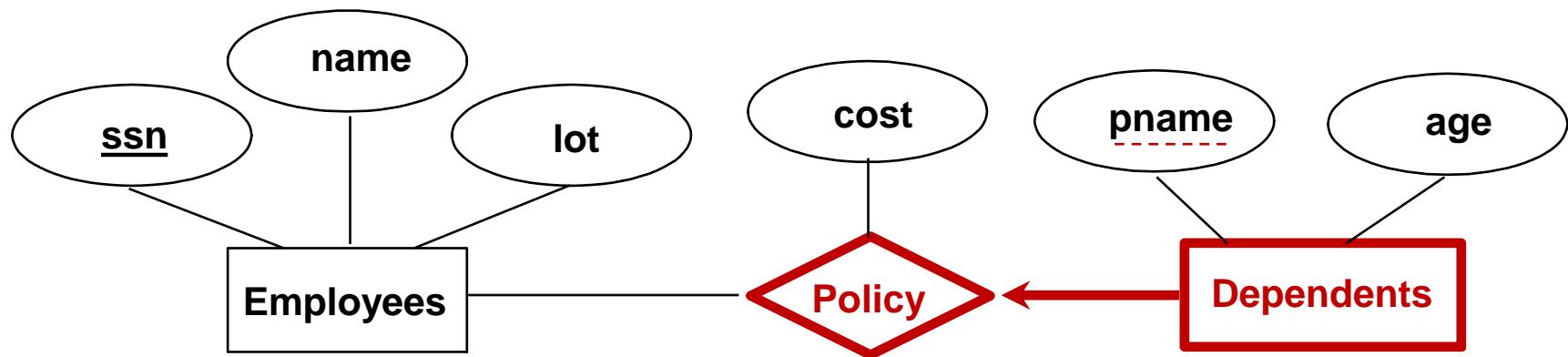
RI-1: cada departamento tem de ter pelo menos um empregado (e vice-versa)

Entidades Fracas para ER

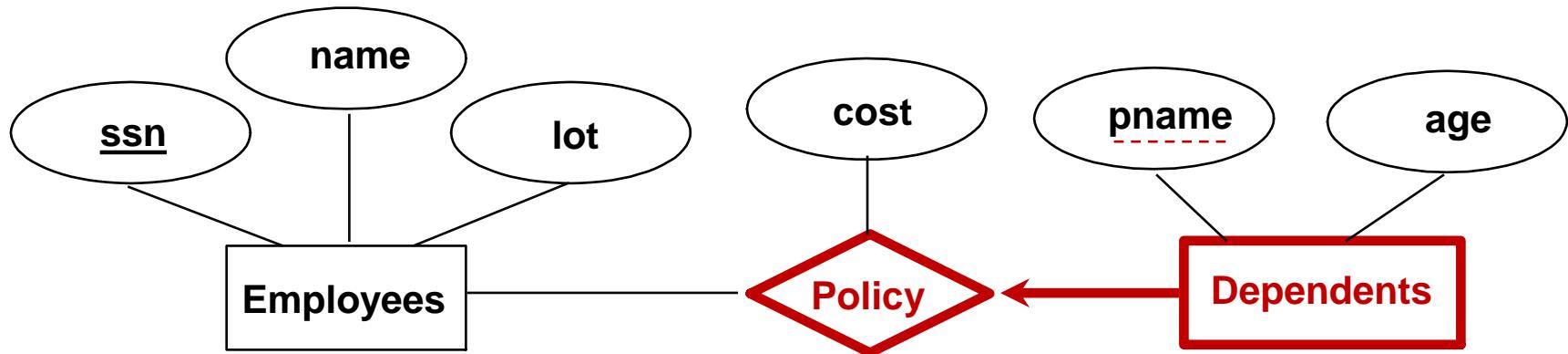
Criação de nova tabela e chave estrangeira para entidade forte

Chave primária da entidade fraca é **composta**...

... por chave parcial e chave primária da entidade forte



Entidades Fracas para ER



```
CREATE TABLE Dep_Policy (
  pname CHAR(20) ,
  age INTEGER,
  cost REAL,
  ssn CHAR (11) ,
  PRIMARY KEY (pname, ssn) ,
  FOREIGN KEY (ssn) REFERENCES Employees ON DELETE CASCADE)
```

CASCADE

Remoção de linha na entidade forte (employees) despoleta a remoção das respectivas linhas na entidade fraca (dependents)

Generalizações para ER

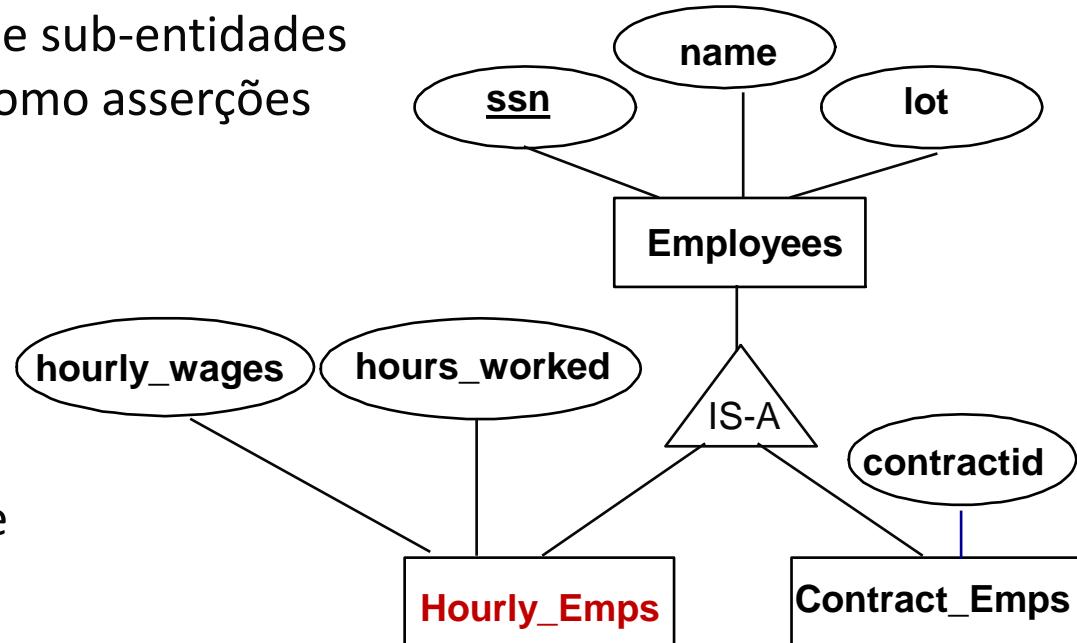
Abordagem 1

Criação de tabelas para a super-entidade e sub-entidades

Restrições de cobertura e sobreposição como asserções

Nas tabelas das sub-entidades

- Chave primária vem da super-entidade
E se a sub-entidade tiver uma chave própria?
- Chave estrangeira para a super-entidade
com propagação de remoções



```
CREATE TABLE Hourly_Emps (
    hours_worked INTEGER,
    hourly_wages REAL,
    ssn CHAR (11),
    PRIMARY KEY (ssn),
    FOREIGN KEY (ssn) REFERENCES Employees ON DELETE CASCADE )
```

Generalizações para ER

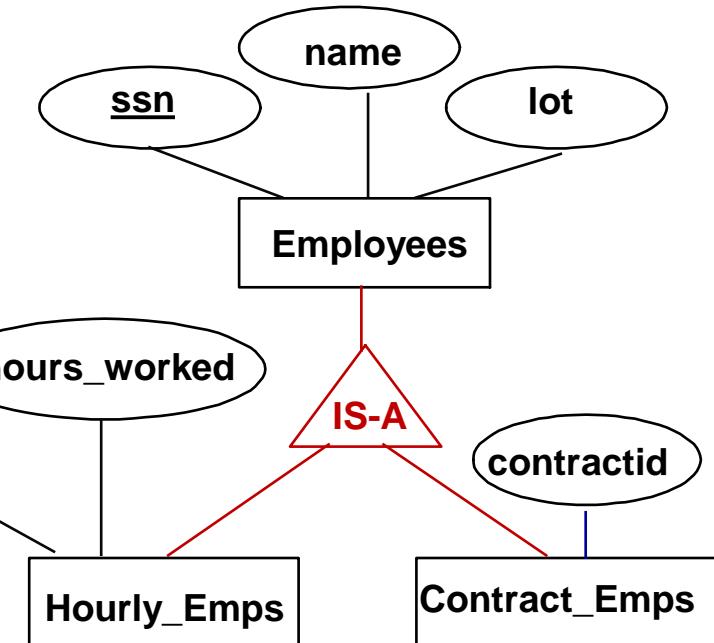
Abordagem 2

Criação de tabelas **apenas para as sub-entidades**

Restrições de cobertura e sobreposição como asserções

Apenas aplicável quando existe **cobertura total**

```
CREATE TABLE Hourly_Emps (
    ssn CHAR(11),
    name CHAR(30),
    lot INTEGER,
    hours_worked INTEGER,
    hourly_wages REAL,
    PRIMARY KEY (ssn))
```



```
CREATE TABLE Contract_Emps (
    ssn CHAR(11),
    name CHAR(30),
    lot INTEGER,
    contractid INTEGER,
    PRIMARY KEY (ssn))
```

Generalizações para ER

Duas abordagens

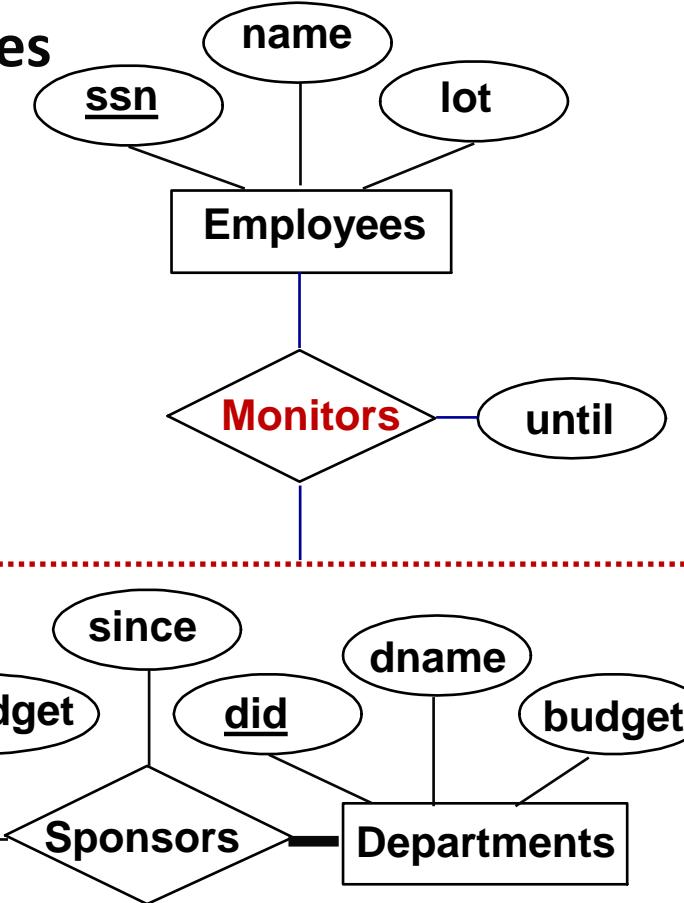
1. Criação de tabelas para a super-entidade e sub-entidades
 - Sempre aplicável
 - Necessário consultar **duas tabelas** para obter todos os dados de cada sub-entidade
2. Criação de tabelas apenas para as sub-entidades
 - Mais eficiente para interrogações a sub-entidades específicas
 - Apenas aplicável quando existe **cobertura total**

Restrições de cobertura e sobreposição como asserções

Aggregações para ER

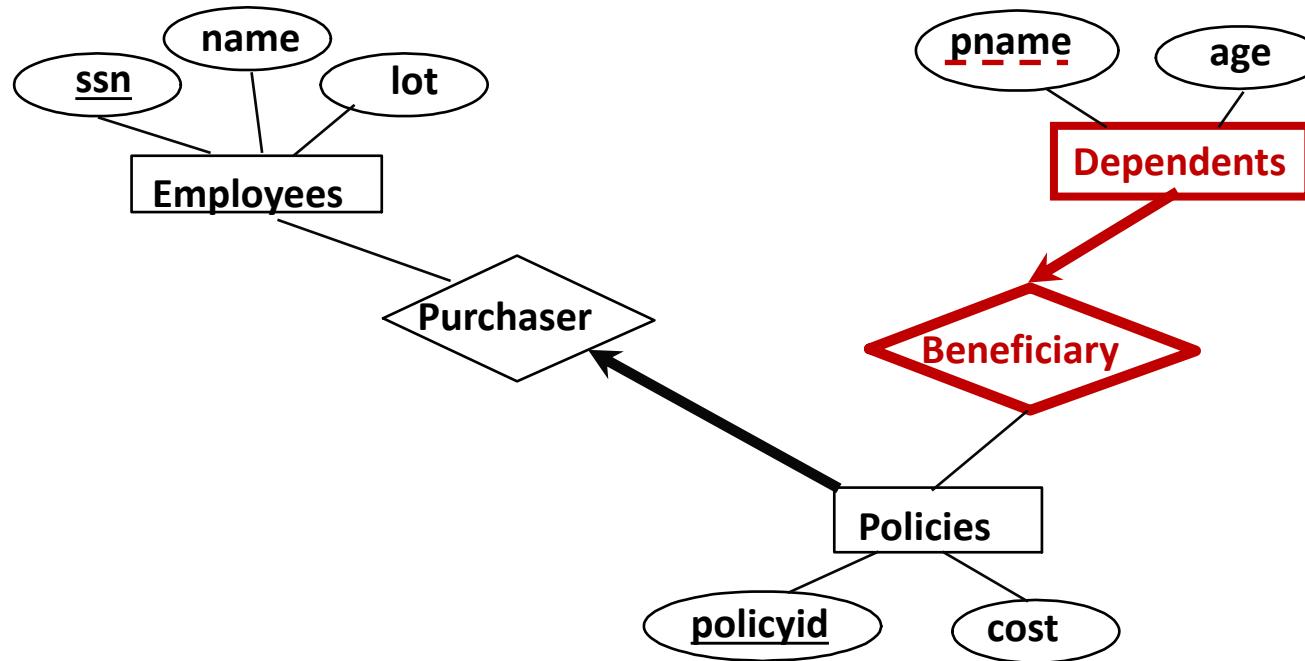
- Passagem **semelhante** à aplicada às associações
- Primeiro traduz-se o interior da agregação
 - Sponsors
- Depois a associação com a agregação
 - Monitors

```
CREATE TABLE Monitors (
    ssn CHAR(11),
    pid INTEGER,
    did INTEGER,
    until CHAR(11),
    PRIMARY KEY (ssn,pid,did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (pid,did) REFERENCES Sponsors)
```

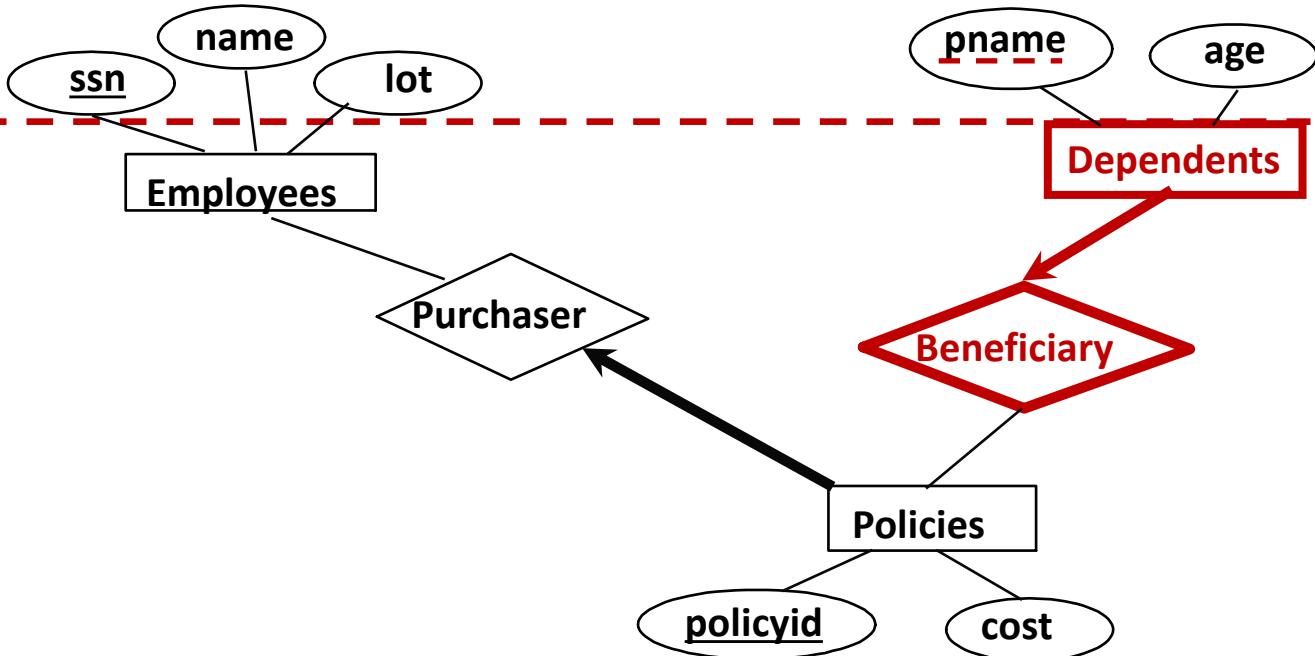


Exercício 1

Passagem do EA para ER



Exercício 1

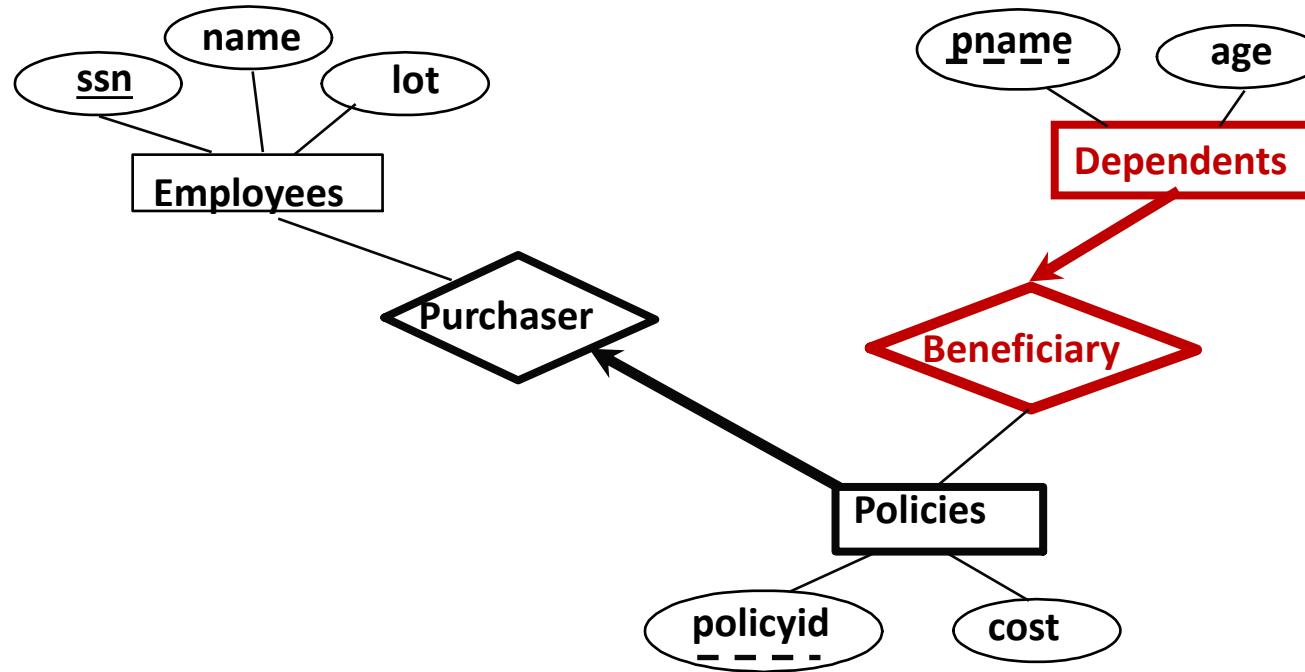


```
CREATE TABLE Policies (
    policyid INTEGER,
    cost REAL,
    ssn CHAR (11) NOT NULL,
    PRIMARY KEY (policyid),
    FOREIGN KEY (ssn)
        REFERENCES Employees
        ON DELETE CASCADE )
```

```
CREATE TABLE Dependents (
    pname CHAR(20),
    age INTEGER,
    policyid INTEGER,
    PRIMARY KEY (pname, policyid),
    FOREIGN KEY (policyid)
        REFERENCES Policies
        ON DELETE CASCADE)
```

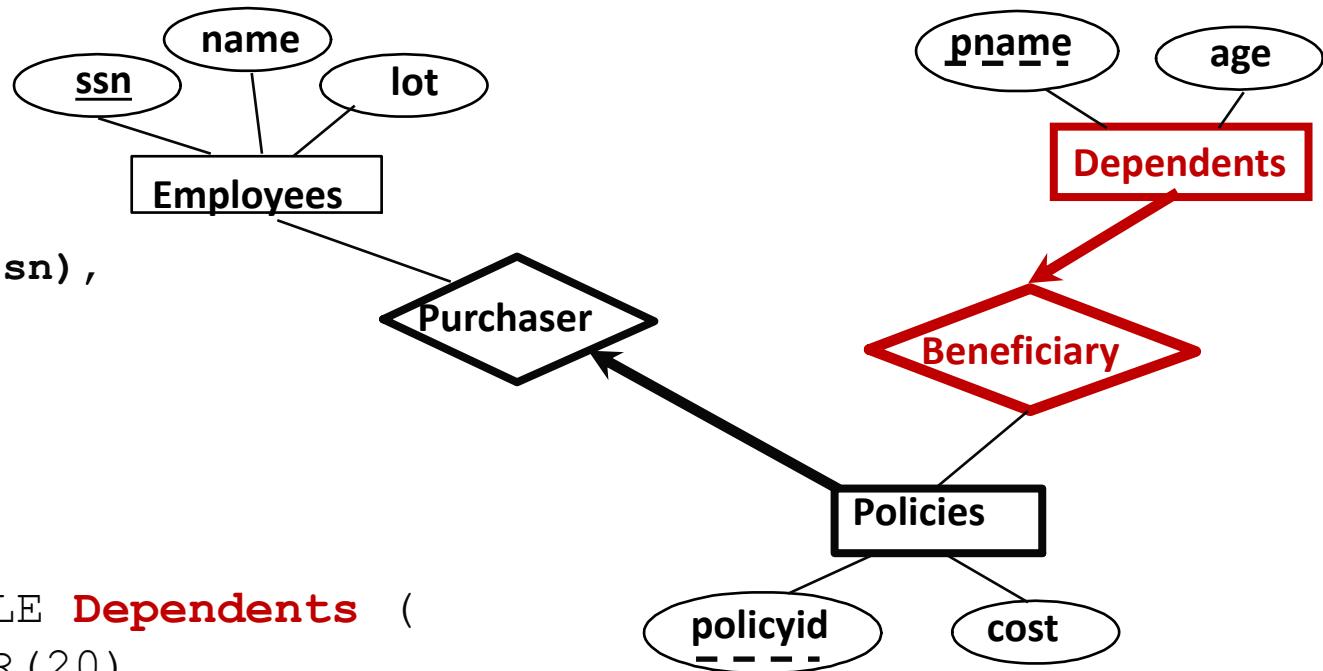
Exercício 2

Passagem do EA para ER



Exercício 2

```
CREATE TABLE Policies (
    policyid INTEGER,
    cost REAL,
    ssn CHAR (11),
    PRIMARY KEY (policyid, ssn),
    FOREIGN KEY (ssn)
        REFERENCES Employees
        ON DELETE CASCADE )
```



```
CREATE TABLE Dependents (
    pname CHAR(20),
    age INTEGER,
    ssn CHAR (11),
    policyid INTEGER,
    PRIMARY KEY (pname, policyid, ssn),
    FOREIGN KEY (policyid, ssn) REFERENCES Policies
        ON DELETE CASCADE ))
```

Introdução às Bases de Dados

Modelo Relacional - IV

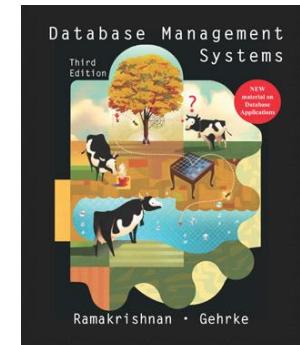
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Comandos SQL
 - Alteração de tabelas
 - Remoção de tabelas
 - Pesquisas de dados em tabelas (forma básica)
 - Vistas
 - Criação de vistas
 - Independência dos dados e privacidade
 - Atualização de dados
- Referências
 - R. Ramakrishnan (**capítulo 3, secção 3.4, 3.6 e 3.7**)



Síntese de Comandos SQL

- **SQL-DDL:** *Data Definition Language*
 - operações sobre a estrutura das tabelas e gestão de restrições de integridade
 - CREATE TABLE
 - DROP TABLE
 - ALTER TABLE
 - CREATE VIEW
- **SQL-DML:** *Data Manipulation Language*
 - operações sobre os **dados** das tabelas
 - INSERT INTO
 - DELETE FROM
 - UPDATE
 - SELECT

Tabelas de Exemplo

Students (sid: string, name: string, login: string, age: integer, gpa: real)

Enrolled (studid: string, cid: string, grade: string)

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae203	B	53832	53666	Jones	jones@cs	18	3.4
Topology112	A	53650	53688	Smith	smith@ee	18	3.2
History105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation)

Students (Referenced relation)

Alteração de Tabelas

- Alteração de tabelas com o comando ALTER TABLE
 - Adição, alteração, e remoção de **colunas**
- **Adição de coluna**
 - Linhas que já existem ficam a *NULL* nessa coluna
 - `ALTER TABLE Students ADD COLUMN maiden-name CHAR(10)`
- **Alteração de coluna**
 - Novo domínio deverá abranger valores já existentes nessa coluna
 - `ALTER TABLE Students MODIFY COLUMN maiden-name CHAR(20)`
- **Remoção de uma coluna**
 - `ALTER TABLE Students DROP COLUMN maiden-name`

Alteração de Tabelas

- Alteração de tabelas com o comando ALTER TABLE
 - Adição e remoção de **restrições de integridade**
- **Adição de restrição de integridade**
 - Apenas se todos os dados já existentes cumprirem a nova regra
 - `ALTER TABLE Enrolled ADD CONSTRAINT nn_enrolled_grade CHECK (grade IS NOT NULL)`
- **Remoção de restrição de integridade**
 - Tem efeito permanente
 - `ALTER TABLE Enrolled DROP CONSTRAINT nn_enrolled_grade`

Pesquisas de Dados

- Pesquisas de dados com o comando **SELECT**
 - Seleção de linhas e colunas de uma ou mais tabelas
- Exemplos com uma tabela

```
SELECT * FROM Students
```

```
SELECT name, login  
FROM Students  
WHERE age >= 18
```

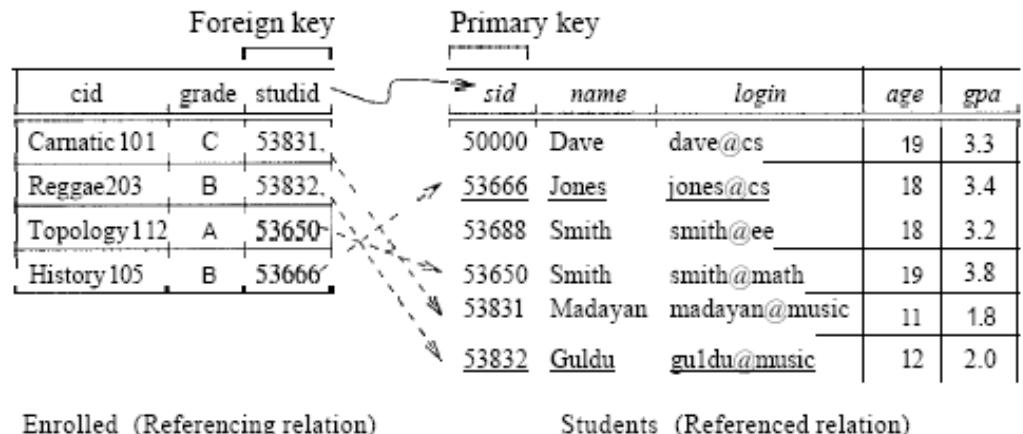
sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

name	login
Dave	dave@cs
Jones	jones@cs
Smith	smith@ee
Smith	smith@math

Pesquisas de Dados

- Exemplo que combina informação de duas tabelas
 - Nome dos estudantes e nome da disciplina para os quais grade = 'A' e S.sid = E.studid

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid = E.studid AND  
grade = 'A'
```



Vistas sobre Dados

- **Uma vista é uma tabela virtual**
 - Cujas linhas não são explicitamente armazenadas
 - Conteúdo determinado na criação da view por um comando SELECT
 - Pode mostrar apenas algumas colunas e linhas da(s) tabela(s) de base
- **Essencial para independência e privacidade de dados**
 - Pode abstrair alterações na tabela de base
 - Cada tipo de utilizador pode ter acesso a vistas específicas

Vistas sobre Dados

- Exemplo

```
CREATE VIEW Old_students (name, login, age) AS  
SELECT name, login, age  
FROM students  
WHERE age >= 18
```

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Consulta da *view* executa a interrogação SQL associada

```
SELECT * FROM Old_students
```

Vistas sobre Dados

- Exemplo

```
CREATE VIEW B-Students (name, sid, course) AS  
    SELECT S.sname, S.sid, E.cid  
    FROM Students S, Enrolled E  
    WHERE S.sid = E.studid AND E.grade = 'B'
```

<i>name</i>	<i>sid</i>	<i>course</i>
Jones	53666	History105
Guldu	53832	Reggae203

Vistas sobre Dados

- SQL distingue entre vidas
 - Cujas linhas podem ser modificadas (*updatable views*)
 - E vistas onde novas linhas podem ser inseridas (*insertable views*)
 - Tem de existir uma relação de um para um entre as linhas da vista e das linhas das respetivas tabelas

Inserção de Dados nas Views

VIEW good_student

TABLE student

```
CREATE VIEW good_student (sid, gpa) AS  
SELECT sid, gpa FROM student  
WHERE gpa >= 3
```

sid	gpa
50000	3.3
53650	3.8
53666	3.4
53688	3.2

sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
INSERT INTO good_student VALUES(10000, 3)  
INSERT INTO good_student VALUES(11000, 1.8)
```

sid	gpa
10000	3.0
50000	3.3
53650	3.8
53666	3.4
53688	3.2

sid	name	login	age	gpa
10000			0	3.0
11000			0	1.8
50000	Dave	dave@cs	19	3.3
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Inserção de Dados nas Views

VIEW **good_student**

```
CREATE VIEW good_student (sid, gpa) AS  
    SELECT sid, gpa FROM student  
    WHERE gpa >= 3  
WITH CHECK OPTION
```

sid	gpa
10000	3.0
50000	3.3
53650	3.8
53666	3.4
53688	3.2

TABLE **student**

sid	name	login	age	gpa
10000			0	3.0
11000			0	1.8
50000	Dave	dave@cs	19	3.3
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
INSERT INTO good_student values (12000, 3)  
INSERT INTO good_student values (13000, 1.8)
```

sid	name	login	age	gpa
10000			0	3.0
11000			0	1.8
12000			0	3.0
50000	Dave	dave@cs	19	3.3
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Atualização de Dados nas Views

```
CREATE VIEW authors_CA AS  
    SELECT * FROM Authors WHERE state='CA'
```

```
UPDATE authors_CA SET state='NJ'
```

--problema/desafio: deixa de pertencer à vista

```
CREATE VIEW authors_CA AS  
    SELECT * FROM Authors WHERE state='CA'  
WITH CHECK OPTION
```

```
UPDATE authors_CA SET state='NJ'
```

Remoção de Tabelas e Vistas

- Remoção de tabela e dos seus dados
 - As chaves estrangeiras para esta tabela têm de ser removidas antes
 - `DROP TABLE Student`

`DROP TABLE Students RESTRICT`

- Apaga a tabela exceto se existirem *views* ou restrições de integridade referencial

`DROP TABLE Students CASCADE`

- Apaga a tabela e todas as *views* e restrições de integridade referencial

- Remoção de view

- `DROP VIEW Good_students`

MySQL observations

- `DROP TABLE IF EXISTS` não é *standard*
 - Se a tabela não existir não dá erro
- `SET foreign_key_checks = 0;`
 - Lista de tabelas: drop table ...
- `SET foreign_key_checks = 1;`

Introdução às Bases de Dados

Interrogações SQL - I

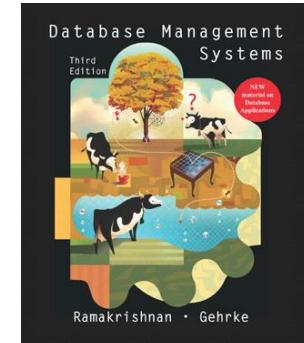
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - SQL/DML: Interrogação/Seleção
 - Forma básica e avaliação
 - Exemplos de interrogações SQL
 - Expressões no SELECT
 - Expressões aritméticas e chamadas a funções
 - Operador LIKE
 - Construtores de conjuntos
 - UNION, INTERSECT, EXCEPT
- Referências
 - R. Ramakrishnan (**capítulo 5, secção 5.2 e 5.3**)



Síntese de Comandos SQL - DML

- **SQL-DML:** *Data Manipulation Language*

operações sobre os **dados** das tabelas

- INSERT INTO
- DELETE FROM
- UPDATE
- SELECT

Forma Básica de uma Interrogação SQL

```
SELECT [DISTINCT] select-list  
FROM from-list  
[WHERE qualification]
```

select-list: lista de colunas a seleccionar

from-list: lista de uma ou mais tabelas de onde provêm os dados

qualification: condições para definir as linhas a seleccionar

condição booleana que admite AND, OR, NOT,

exp op exp em que op: {<, <=, =, <>, >=, >}

DISTINCT: elimina linhas duplicadas

Avaliação de uma Interrogação

Algoritmo genérico para selecionar dados

1. Calcula o **produto cartesiano** de todas as tabelas no **from-list** (FROM)
2. Elimina as linhas que falham a condição **qualification** (WHERE)
3. Elimina as colunas que não aparecem na **select-list** (SELECT)
4. Elimina linhas duplicadas se usar **DISTINCT**

Tabelas de Exemplo

Sailors

<u>sid</u>	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Boats

<u>bid</u>	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

Exemplo de DISTINCT

- Nomes e idades de todos os marinheiros

```
SELECT DISTINCT S.sname, S.age  
FROM Sailors S
```

<i>sname</i>	<i>age</i>
Dustin	45.0
Brutus	33.0
Lubber	55.5
Andy	25.5
Rusty	35.0
Horatio	35.0
Zorba	16.0
Art	25.5
Bob	63.5

Com DISTINCT

<i>sname</i>	<i>age</i>
Dustin	45.0
Brutus	33.0
Lubber	55.5
Andy	25.5
Rusty	35.0
Horatio	35.0
Zorba	16.0
Horatio	35.0
Art	25.5
Bob	63.5

Sem DISTINCT

Condição sobre Linhas de uma Tabela

- Marinheiros com um *rating* maior que 7

```
SELECT S.sid, S.sname, S.rating, S.age  
FROM Sailors S  
WHERE S.rating > 7
```

- Sinónimo **S** pode ser usado no contexto do SELECT em vez de Sailors (etiquetagem de tabela)
- SELECT * seria uma alternativa para mostrar todas as colunas
- SELECT *
 - É aceitável em modo interativo
 - Em programação de aplicações com BD é preferível indicar explicitamente

Interrogação com Duas Tabelas

- Nomes dos marinheiros que reservaram o barco 103

```
SELECT S.sname  
FROM Sailors S, Reserves R  
WHERE S.sid = R.sid AND R.bid=103
```

Com duas ou mais tabelas é essencial usar a **condição de junção**

<i>sid</i>	<i>sname</i>	<i>Tating</i>	<i>age</i>
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/96
58	103	11/12/96

<i>sname</i>
rusty

Produto Cartesiano e Condição de Junção

Produto cartesiano

```
SELECT *  
FROM Sailors S, Reserves R
```

sid	sname	Tating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

sid	bid	day
22	101	10/10/96
58	103	11/12/96

sid	sname	rating	age	sid	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Com condição de junção

```
SELECT *  
FROM Sailors S, Reserves R  
WHERE S.sid = R.sid
```

sid	sname	rating	age	sid	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Produto Cartesiano e Condição de Junção

<i>sid</i>	<i>sname</i>	<i>Tating</i>	<i>age</i>
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/96
58	103	11/12/96

Nomes dos marinheiros que reservaram o barco 103

```
SELECT S.sname  
FROM Sailors S, Reserves R  
WHERE S.sid = R.sid AND R.bid=103
```

sname
dustin
rusty

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>	<i>sid</i>	<i>bid</i>	<i>day</i>
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Interrogação com Três Tabelas

- **Identificadores** dos marinheiros que reservaram um barco vermelho
 - Acesso a 2 tabelas requer pelo menos **1 condição de junção**

```
SELECT R.sid  
FROM Boats B, Reserves R  
WHERE B.bid = R.bid AND B.color = 'red'
```

- **Nomes** dos marinheiros que reservaram um barco vermelho
 - Acesso a 3 tabelas requer pelo menos **2 condições de junção**

```
SELECT S.sname  
FROM Sailors S, Boats B, Reserves R  
WHERE S.sid = R.sid AND B.bid = R.bid  
      AND B.color = 'red'
```

Tabelas de Exemplo

Sailors

<u>sid</u>	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Boats

<u>bid</u>	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

Mais exemplos

- **Cores dos barcos** reservados pelo marinheiro Lubber

```
SELECT B.color  
FROM Sailors S, Reserves R, Boats B  
WHERE S.sid = R.sid AND R.bid = B.bid  
      AND S.sname = 'Lubber'
```

- **Nomes dos marinheiros** que reservaram pelo menos um barco

```
SELECT S.sname  
FROM Sailors S, Reserves R  
WHERE S.sid = R.sid
```

Expressões na Select-list e Qualification

- **Select-list** pode ter mais do que nomes de colunas de tabelas
 - Expressões aritméticas e chamadas a funções (ex. funções de agregação)
 - Cada expressão deve ter um nome fácil de interpretar
 - Exemplo
- Na **Qualification** as condições podem incluir
 - Expressões aritméticas e chamadas a funções
 - Ex. Nome dos marinheiros com o dobro do rating de outros marinheiros

```
SELECT S1.sname AS name1, S2.sname AS name2  
FROM Sailors S1, Sailors S2  
WHERE 2*S1.rating = S2.rating
```

Exemplo mais complexo

- Calcular **incrementos** dos *ratings* dos marinheiros que reservaram dois barcos para o mesmo dia

```
SELECT S.sname, S.rating+1 AS new_rating  
FROM Sailors S, Reserves R1, Reserves R2  
WHERE S.sid = R1.sid AND S.sid = R2.sid  
      AND R1.day = R2.day  
      AND R1.bid <> R2.bid
```

Operador LIKE

- O operador **LIKE** suporta uma variante de expressões regulares
 - O caracter **%** representa zero ou mais caracteres arbitrários
 - O caracter **_** representa um caracter arbitrário
 - O espaço é importante no LIKE
 - Ex. Idade dos marinheiros cujo nome começa por um qualquer caracter, seguido de um A, depois de um B e depois um qualquer outro caracter

```
SELECT S.age
FROM Sailors S
WHERE S.name LIKE '_AB%'
```
 - Ex. Idades dos marinheiros cujo nome comece e termine com um B e tenha no mínimo 3 caracteres

```
SELECT S.age
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

Construtores de conjuntos

- **União**, SELECT ... UNION SELECT ...
 - União das linhas dos dois conjuntos
- **Interseção**, SELECT ... INTERSECT SELECT ...
 - Linhas comuns a ambos os conjuntos
- **Diferença**, SELECT ... EXCEPT SELECT ...
 - Linhas de um conjunto às quais se retiraram as linhas de outro conjunto
- Por omissão são eliminadas as linhas duplicadas
 - Para manter os duplicados, UNION ALL, INTERSECT ALL, EXCEPT ALL

Exemplo de União Simples

- Nomes dos marinheiros que reservaram um barco verde **ou** vermelho

```
SELECT S.sname  
FROM Sailors S, Reserves R, Boats B  
WHERE S.sid = R.sid AND R.bid = B.bid  
AND (B.color = 'red' OR B.color = 'green')
```

- Alternativa com **UNION**

```
SELECT S.sname  
FROM Sailors S, Reserves R, Boats B  
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'  
UNION  
SELECT S2.sname  
FROM Sailors S2, Reserves R2, Boats B2  
WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green'
```

- Assume S.name como chave candidata
- Se não for considerado poderia não ser o mesmo marinheiro. Se selecionarmos sid, em vez de sname, ok
- Solução com sub-interrogações (próxima aula)

Exemplo de Interseção Complexa

- Nomes dos marinheiros que reservaram um barco verde e vermelho
 - Bastaria substituir OR por AND no exemplo da união simples?

```
SELECT S.sname
      FROM Sailors S, Reserves R1, Boats B1,
            Reserves R2, Boats B2
     WHERE S.sid = R1.sid AND R1.bid = B1.bid
           AND S.sid = R2.sid AND R2.bid = B2.bid
           AND (B1.color='red' AND B2.color = 'green')
```

- **R1** e **B1** são as linhas que provam que o marinheiro S.sid reservou um barco vermelho
- **R2** e **B2** são as linhas que provam que o marinheiro S.sid reservou um barco verde

Solução com INTERSECT

- Nomes dos marinheiros que reservaram um barco verde e vermelho

```
SELECT S.sname  
FROM Sailors S, Reserves R, Boats B  
WHERE S.sid = R.sid AND R.bid = B.bid  
      AND B.color = 'red'
```

INTERSECT

```
SELECT S2.sname  
FROM Sailors S2, Reserves R2, Boats B2  
WHERE S2.sid = R2.sid AND R2.bid = B2.bid  
      AND B2.color = 'green'
```

- Observações
 - Solução mais eficiente que a anterior
 - Assume S.name como chave candidata

Exemplo de EXCEPT

- **Identificadores** dos marinheiros que reservaram barcos vermelhos **mas não** verdes

```
SELECT R.sid  
FROM Boats B, Reserves R  
WHERE R.bid = B.bid AND B.color = 'red'
```

EXCEPT

```
SELECT R2.sid  
FROM Boats B2, Reserves R2  
WHERE R2.bid = B2.bid AND B2.color = 'green'
```

União de Tabelas Diferentes

- Identificadores dos marinheiros com um *rating* de 10 ou com uma reserva para o barco 104

```
SELECT S.sid  
FROM Sailors S  
WHERE S.rating = 10
```

UNION

```
SELECT R.sid  
FROM Reserves R  
WHERE R.bid = 104
```

- Observações: Para estas operações, os **conjuntos** têm de ser **compatíveis**
 - Mesmo número de colunas e com os mesmos tipos (domínios)

Introdução às Bases de Dados

Interrogações SQL – II (Subinterrogações)

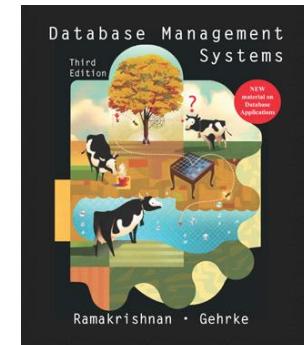
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Sub-Interrogações
 - Independentes (operadores IN e NOT IN)
 - Correlacionadas (operadores EXISTS e NOT EXISTS)
 - Múltiplas Sub-Interrogações
 - Comparações de conjuntos
 - Operadores ANY e ALL
 - Exemplos
 - Máximo e Intersecções
- Referências
 - R. Ramakrishnan (**capítulo 5, secção 5.4**)



Sub-Interrogações em Interrogações

- Uma interrogação pode conter outras interrogações
 - Um (ou mais) SELECT dentro de um SELECT
 - Também denominadas *nested queries*
 - A interrogação que está embebida chama-se sub-interrogação

```
SELECT S.sname  
FROM   Sailors S  
WHERE  S.sid IN
```

```
( SELECT R.sid  
  FROM Reserves R  
 WHERE R.bid = 103)
```

- Numa interrogação podem aparecer
 - Na cláusula WHERE (o mais frequente)

```
SELECT ...  
FROM ...  
WHERE ...OperadorConjunto (SELECT ... FROM ...)
```

OperadorConjunto: IN, NOT IN, EXISTS, NOT EXISTS, ANY, ALL

- Na Cláusula FROM (**não autorizados no âmbito desta disciplina**)
- Na cláusula HAVING (matéria futura)

Tabelas de Exemplo

Sailors

<u>sid</u>	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Boats

<u>bid</u>	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

Interrogação com Operador IN

- Nomes dos marinheiros que reservaram o barco 103

```
SELECT S.sname  
FROM   Sailors S  
WHERE  S.sid  IN  ( SELECT R.sid  
                      FROM Reserves R  
                     WHERE R.bid = 103)
```

- A **sub-interrogação** devolve o conjunto dos identificadores dos marinheiros que reservaram o barco 103
- A interrogação seleciona apenas os marinheiros que pertencem ao conjunto da sub-interrogação

Múltiplas Sub-Interrogações

- Nomes dos marinheiros que reservaram barcos vermelhos

```
SELECT S.sname  
FROM Sailors S  
WHERE S.sid IN ( SELECT R.sid  
                  FROM Reserves R  
                  WHERE R.bid IN (SELECT B.bid  
                                 FROM Boats B  
                                 WHERE B.color = 'red' ) )
```

- A **2ª sub-interrogação** devolve os identificadores dos barcos vermelhos
- A **1ª sub-interrogação** devolve os identificadores dos marinheiros que reservaram barcos vermelhos
- A interrogação devolve o nome dos marinheiros que reservaram barcos vermelhos

Interrogação com Operador NOT IN

- Nomes dos marinheiros que **não** reservaram barcos vermelhos

```
SELECT S.sname  
FROM Sailors S  
WHERE S.sid NOT IN( SELECT R.sid  
                      FROM Reserves R  
                      WHERE R.bid IN ( SELECT B.bid  
                                      FROM Boats B  
                                      WHERE B.color = 'red' ))
```

- Outras alternativas
 - **NOT IN** no segundo WHERE em vez de no primeiro
Nome dos marinheiros que reservaram barcos que não são vermelhos
 - **NOT IN** nos dois WHERE
Nome dos marinheiros que não reservaram barcos que não são vermelhos
Ou seja, que só reservaram barcos vermelhos ... juntamente com os que não fizeram qualquer reserva

Sub-Interrogações Correlacionadas

- Nomes dos marinheiros que reservaram o barco 103

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS ( SELECT *
    FROM Reserves R
    WHERE R.bid = 103 AND
        R.sid = S.sid )
```

- EXISTS verifica a existência de valores no resultado da sub-interrogação
 - A condição na sub-interrogação tem em conta o marinheiro atual na interrogação principal
 - A utilização de SELECT * é recomendada nestas situações
- Outro caso
 - NOT EXISTS no WHERE: Marinheiros que não reservaram o barco 103

Comparação de Conjuntos com ANY

- Marinheiros cujo *rating* é maior que o rating de **algum** dos marinheiros chamados *Horatio*

```
SELECT S.sid
  FROM Sailors S
 WHERE S.rating > ANY
       ( SELECT S2.rating
         FROM Sailors S2
        WHERE S2.sname = 'Horatio' )
```

- (1) Sub-interrogação devolve os ratings dos marinheiros Horatio
- Interrogação seleciona os marinheiros cujo rating seja superior a algum em (1)
- Se sub-interrogação devolve **conjunto vazio** então $> \text{ANY}$ (empty) = false
- SOME é sinónimo de ANY

Comparação de Conjuntos com ALL

- Marinheiros cujo *rating* é maior que o rating de **todos** os marinheiros chamados *Horatio*

```
SELECT S.sid
  FROM Sailors S
 WHERE S.rating > ALL
       ( SELECT S2.rating
         FROM Sailors S2
        WHERE S2.sname = 'Horatio' )
```

- (1) Sub-interrogação devolve os ratings dos marinheiros Horatio
- Interrogação seleciona os marinheiros cujo rating seja superior a todos em (1)
- Se sub-interrogação devolve **conjunto vazio** então $> \text{ALL}$ (empty) = true
- Se não há Horatios então devolve todos os marinheiros

Exemplo de Escolha do Valor Máximo

- Marinheiros com o maior *rating*?

```
SELECT S.sid  
FROM Sailors S  
WHERE S.rating >= ALL ( SELECT S2.rating  
                           FROM Sailors S2)
```

- Observações
 - WHERE ... IN equivalente a WHERE ... = ANY ...
 - WHERE ... NOT IN equivalente a WHERE ... <> ALL ...

Exemplo de Interseção de Conjuntos

- Quais os nomes dos marinheiros que reservaram (pelo menos) um barco verde e (um) vermelho?

```
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = 'red'
      AND S.sid IN (
                      SELECT S2.sid
                      FROM Sailors S2, Boats B2, Reserves R2
                      WHERE S2.sid = R2.sid
                            AND R2.bid = B2.bid
                            AND B2.color = 'green' )
```

- Observação:
 - Equivalente a INTERSECT, e NOT IN equivalente a EXCEPT
 - Concretização no MYSQL do INTERSECT e EXCEPT

Bases de Dados

Interrogações SQL – III (Operadores de Agregação)

FCUL, Departamento de Informática

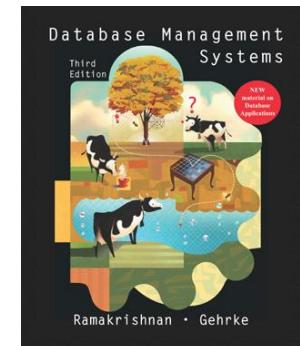
Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Operadores de agregação
COUNT, SUM, AVG, MAX, MIN
Exemplos
 - Sub-Interrogações com operadores de agregação
 - Agrupamento de linhas e Filtragem sobre grupos
GROUP BY e HAVING
Exemplos e Regras do GROUP BY
 - Operador de divisão

- Referências
 - R. Ramakrishnan (**capítulo 5, secção 5.5**)



Operadores de Agregação

- Os operadores de agregação
 - produzem sumários de dados tipicamente referentes a uma coluna da tabela
 - devolvem um valor único como resultado
- Operadores de agregação em SQL
 - COUNT ([DISTINCT] coluna)
Número de valores na coluna da tabela
 - SUM ([DISTINCT] coluna)
Soma dos valores na coluna A
 - AVG ([DISTINCT] coluna)
Média dos valores na coluna A
 - MAX (coluna) e MIN(coluna)
Máximo e mínimo valor na coluna

Observação: DISTINCT considera apenas os valores únicos, sem repetições

Tabelas de Exemplo

Sailors

<u>sid</u>	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Boats

<u>bid</u>	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

Interrogações com COUNT

- Número total de marinheiros

```
SELECT COUNT(*)  
FROM Sailors S
```

COUNT(*)
10

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

- Número de nomes distintos de marinheiros

```
SELECT COUNT(DISTINCT S.sname)  
FROM Sailors S
```

COUNT(DISTINCT S.sname)
9

Observação

- A utilização do * no COUNT é recomendada
- Quando estão a ser contadas linhas e não colunas específicas

Interrogações com AVG

- Média de idades dos marinheiros

```
SELECT AVG(S.age)  
FROM Sailors S
```

AVG(S.age)
36.90000

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

- Média de idades dos marinheiros com *rating* de 10

```
SELECT AVG(S.age)  
FROM Sailors S  
WHERE S.rating = 10
```

AVG(S.age)
25.50000

Interrogações com MAX

- Idade do marinheiro mais velho

```
SELECT MAX (S.age)  
FROM Sailors S
```

- Nome e idade do marinheiro mais velho

```
SELECT S.sname, MAX (S.age)  
FROM Sailors S
```

- Não é um comando válido

O operador de agregação MAX agrupa num só valor os valores das outras colunas não estão acessíveis
Excepto se GROUP BY (mais à frente)

```
SELECT S.sname, S.age  
FROM Sailors S  
WHERE S.age = ( SELECT MAX (S2.age)  
                FROM Sailors S2 )
```

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sname	MAX(S.age)
Dustin	63.5



sname	age
Bob	63.5



Sub-Interrogações com Operadores de Agregação

- Nomes dos marinheiros que são mais velhos que o **marinheiro mais velho com rating de 10?**

```
SELECT S.sname  
FROM Sailors S  
WHERE S.age > ( SELECT MAX(S2.age)  
FROM Sailors S2  
WHERE S2.rating = 10)
```

- Outra alternativa, com operador ALL

```
SELECT S.sname  
FROM Sailors S  
WHERE S.age > ALL( SELECT S2.age  
FROM Sailors S2  
WHERE S2.rating = 10)
```

- Observação: > ALL pode ser menos claro que MAX

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sname
Dustin
Lubber
Bob

Agrupamento e Filtragem

- Sintaxe do comando SELECT

```
SELECT [DISTINCT] select-list  
FROM from-list  
[WHERE qualification]  
GROUP BY grouping-list  
[HAVING group-qualification]
```

- GROUP BY permite criar grupos de linhas
 - Cada grupo de linhas tem o mesmo valor nas colunas do **grouping-list**
 - GROUP BY (age) cria tantos grupos quantas as idades existentes
- HAVING elimina os grupos que não satisfazem a condição da **group-qualification**
 - Tem de incluir colunas da **grouping-list** ou operadores de agregação
 - HAVING (age > 15) inclui grupos cujos marinheiros têm idade > 15

Interrogações com GROUP BY

- Idade do mais novo marinheiro para cada *rating*

```
SELECT S.rating, MIN(S.age)
FROM Sailors S
GROUP BY S.rating
```

rating	MIN(S.age)
1	33.0
3	25.5
7	35.0
8	25.5
9	35.0
10	16.0

- Idade do marinheiro mais novo com mais de 18 anos para cada *rating* com pelo menos 2 marinheiros (com mais de 18 anos)

```
SELECT S.rating, MIN(S.age) AS "Idade minima"
FROM Sailors S
WHERE S.age > 18
GROUP BY S.rating
HAVING COUNT(*) >= 2
```

rating	Idade mínima
3	25.5
7	35.0
8	25.5

Passos de Execução

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
9	35.0
3	25.5
3	63.5

Exclusão de marinheiros com idade não superior a 18 anos (e eliminação de colunas desnecessárias)

```
SELECT S.rating, MIN(S.age) AS "Idade minima"  
FROM Sailors S  
WHERE S.age > 18  
GROUP BY S.rating  
HAVING COUNT(*) >= 2
```

rating	age
1	33.0
3	25.5
3	63.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

Criação de grupos para cada *rating*

rating	minage
3	25.5
7	35.0
8	25.5

Exclusão de grupos com menos de 2 marinheiros e cálculo da idade mínima em cada grupo sobrevivente

(fonte: António Ferreira, SIBD19)

Agrupamento e Filtragem: Regras

- Sintaxe do comando SELECT

```
SELECT [DISTINCT] select-list  
FROM from-list  
[WHERE qualification]  
GROUP BY grouping-list  
[HAVING group-qualification]
```

- **Cada coluna na select-list só pode ter um valor único por grupo**
 - Podem ser colunas do grouping-list
 - Ou com dependência funcional
 - Podem ser operadores de agregação

Interrogações com GROUP BY e HAVING

- Número de reservas para cada barco vermelho

```
SELECT B.bid, COUNT (*) AS rescount  
FROM Boats B, Reserves R  
WHERE R.bid = B.bid  
GROUP BY B.bid  
HAVING B.color = 'red'
```



- Solução correta

```
SELECT B.bid, COUNT (*) AS rescount  
FROM Boats B, Reserves R  
WHERE R.bid = B.bid  
      AND B.color = 'red'  
GROUP BY B.bid
```



Interrogações com GROUP BY e HAVING

- Idade do marinheiro mais novo com **mais de 18 anos para cada rating** cuja **média de idades dos marinheiros** (com mais de 18 anos) **seja superior** à **média de idade de todos os marinheiros?**

```
SELECT S.rating, MIN(S.age) AS minage  
FROM Sailors S  
WHERE S.age >= 18  
GROUP BY S.rating  
HAVING AVG(S.age) > ( SELECT AVG(S2.age)  
                        FROM Sailors S2 )
```

rating	minage
3	25.5
7	35.0
8	25.5

Operador de divisão

- Utilidade
 - Permite expressar interrogações de um determinado tipo
 - p.e., "Nome dos marinheiros que reservaram todos os barcos"
- Exemplo

A	<i>sno</i>	<i>pno</i>	B1	<i>pno</i>	A / B1	<i>sno</i>
	s1	p1				s1
	s1	p2		p2		s2
	s1	p3				s3
	s1	p4				s4
	s2	p1				
	s2	p2				
	s3	p2				
	s4	p2				
	s4	p4				
B2	<i>pno</i>		A / B2	<i>sno</i>		
		p2				
		p4				
B3	<i>pno</i>		A / B3	<i>sno</i>		
		p1				
		p2				
		p4				

Interrogações com GROUP BY e HAVING

Exemplo de Divisão

Nomes dos marinheiros que reservaram **todos** os barcos

```
SELECT S.sname  
FROM Sailors S, Reserves R  
WHERE R.sid = S.sid  
GROUP BY S.sname  
HAVING COUNT(DISTINCT R.bid)=( SELECT COUNT(*)  
                                FROM Boats B)
```

Interrogações com NOT EXISTS

Exemplo de Divisão

Nomes dos marinheiros que reservaram **todos** os barcos

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (
    SELECT B.bid
    FROM Boats B
    WHERE NOT EXISTS (
        SELECT R. bid
        FROM Reserves R
        WHERE R.bid = B.bid
        AND R.sid = S.sid ))
```

Obs. All x: p(x) <=> Not Exists x: Not p(x)

Exemplo de Divisão

1. Nome dos marinheiros, para os quais
 - Qualquer que seja o barco
 - Existe uma reserva para esse barco e para esse marinheiro

2. Nome dos marinheiros:

$\forall \text{barco}: (\exists \text{reserva}: R.\text{bid} = B.\text{bid} \text{ AND } R.\text{sid} = S.\text{Sid})$

Nota: $\forall x: p(x) \Leftrightarrow \neg \exists x: \neg p(x)$

3. Nomes dos marinheiros:

$\neg \exists \text{barco}: \neg \exists \text{reserva}: R.\text{bid} = B.\text{bid} \text{ AND } R.\text{sid} = S.\text{sid}$

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (SELECT B.bid
                   FROM Boats B
                   WHERE NOT EXISTS (SELECT R. bid
                                     FROM Reserves R
                                     WHERE R.bid = B.bid AND R.sid = S.sid ))
```

Exemplo de Divisão com Restrição

- Nomes dos marinheiros que reservaram **todos** os barcos **vermelhos**

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (
    SELECT B.bid
    FROM Boats B
    WHERE B.color = 'red'
        AND NOT EXISTS (
            SELECT R. bid
            FROM Reserves R
            WHERE R.bid = B.bid
                AND R.sid = S.sid ))
```

Introdução às Bases de Dados

Interrogações SQL – IV

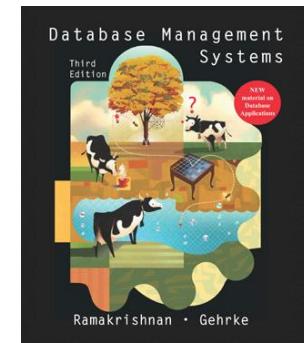
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Ordenação dos resultados
ORDER BY ASC, ORDER BY DESC
 - Valores Nulos
 - Comparações e Operações
 - Verificação de existência de valores Nulos
 - Impacto nos comandos SQL
 - Operadores de Agregação
 - Junções Exteriores
- Referências
 - R. Ramakrishnan (**capítulo 5, secção 5.6**)



Tabelas de Exemplo

Sailors

<u>sid</u>	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	NULL
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Boats

<u>bid</u>	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

Nota: Zorba tem idade desconhecida

Ordenação de Resultados de Interrogações

- Sintaxe do comando SELECT

```
SELECT [DISTINCT] select-list  
FROM from-list  
[WHERE qualification]  
[GROUP BY grouping-list]  
[HAVING group-qualification]  
ORDER BY order-list
```

- ORDER BY permite ordenar os resultados de uma interrogação
 - Sem ORDER BY a ordem das linhas no resultado é arbitrária
 - As colunas na order-list devem constar na select-list
Para facilitar a compreensão do(s) critério(s) de ordenação
 - Ordenação ascendente (ASC) ou descendente (DESC)
Aplica-se a cada coluna da order-list

Ordenação de Resultados de Interrogações

- Todos os marinheiros ordenados alfabeticamente e depois por *rating* (do maior para o menor)

```
SELECT S.sname, S.rating  
FROM Sailors S  
ORDER BY S.sname ASC, S.rating DESC
```

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	NULL
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sname	rating
Andy	8
Art	3
Bob	3
Brutus	1
Dustin	7
Horatio	9
Horatio	7
Lubber	8
Rusty	10
Zorba	10

Valores Nulos

- Usados quando se **desconhece** o valor de uma coluna ou quando **não aplicável**
 - Em SQL são representados por **NULL**
 - Ex. desconhece-se o telemóvel de um cliente, mas pode vir a saber-se
 - Ex. nome de solteira só é relevante para mulheres casadas (*maiden name*)
- Aplicabilidade
 - Podem ser usados em colunas com qq domínio de dados
 - Não podem ser usados em chaves candidatas, chave primária
 - Não podem ser usados em colunas NOT NULL

Operações com Valores Nulos

- Operações de comparação ($<$, \leq , $=$, \neq , \geq , $>$) e aritméticas ($+$, $-$, $*$, $/$)
 - Basta um dos argumentos ser NULL, para resultado ser *unknown*
 - arg1 Op arg2 , se arg1 ou arg2 NULL \Rightarrow resultado *unknown*
- Operações conjunção e disjunção (AND, OR)
 - NULL **AND** FALSE = FALSE, cc. resultado é sempre *unknown*
 - NULL **OR** TRUE = TRUE, cc. resultado é sempre *unknown*
- Verificação de valores nulos (IS NULL)
 - NULL **IS NULL** = TRUE (ex. s.age IS NULL)
 - NULL **IS NOT NULL** = FALSE (ex. s.age IS NOT NULL)

Impacto nos Comandos SQL

- A condição na cláusula **WHERE**
 - Só aparecem no resultado as linhas que verificam a condição ou seja, para a qual a condição seja TRUE
 - Elimina as linhas cujo resultado seja FALSE ou *unknown*
- Ex. Marinheiros com idade superior a 18

```
SELECT *  
FROM Sailors  
WHERE age > 18
```

 - O marinheiro Zorba aparece no resultado?

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	NULL
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Impacto nos Comandos SQL

- A condição na cláusula **WHERE**
 - Só aparecem no resultado as linhas que verificam a condição ou seja, para a qual a condição seja TRUE
 - Elimina as linhas cujo resultado seja FALSE ou *unknown*

- Ex. Marinheiros com idade superior a 18

```
SELECT *
FROM Sailors
WHERE age > 18


- O marinheiro Zorba aparece no resultado?
- NULL > 18 = unknown, pelo que não é selecionado

```

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Impacto nos Comandos SQL

- Operadores de agregação (resultados inesperados)
 - COUNT(*) , na contagem inclui os valores NULL
 - Restantes operadores (com ou sem DISTINCT) ignoram NULL
COUNT(coluna), SUM(coluna), AVG(coluna), MAX/MIN(coluna)
 - Se todos os valores na coluna forem nulos, resultado é NULL
Exceto COUNT(coluna) que devolve 0

```
SELECT COUNT (*)  
FROM Sailors
```

count(*)

10

```
SELECT COUNT (age)  
FROM Sailors
```

COUNT(age)

9

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	NULL
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Junções Naturais (*natural join*)

- Marinheiros e as suas reservas (junção interior)

```
SELECT *
FROM Sailors S, Reserves R
WHERE S.sid = R.sid
```

```
SELECT *
FROM Sailors S INNER JOIN Reserves R
          ON (S.sid = R.sid)
```

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
61	Horatio	7	35.0
71	Zorba	10	NULL
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/96
58	103	11/12/96

sid	sname	rating	age	sid	bid	day
22	Dustin	7	45.0	22	101	1996-10-10
58	Rusty	10	35.0	58	103	1996-12-11

Observação: E os marinheiros que não fizeram reservas?

Não devolve marinheiros sem reservas

Junções Exteriores (*outer join*)

- As junções exteriores
 - Devolvem as linhas que satisfazem a condição de junção e as linhas numa das tabelas sem correspondência com as da outra tabela
 - FULL OUTER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN
- Ex. Marinheiros e as suas reservas

```
SELECT *
FROM Sailors S FULL OUTER JOIN Reserves R
ON (S.sid = R.sid)
```

- Devolve todos os marinheiros mesmo os que não fizeram reservas
- Devolve também as reservas, mesmo as que não tiverem marinheiros (...)

Junções Exteriores

- Qual o número de barcos reservados para cada marinheiro?

```
SELECT S.sname, COUNT(R.bid)
FROM Sailors S LEFT OUTER JOIN Reserves R
      ON (S.sid = R.sid)
GROUP BY S.sname
```

sname	COUNT(R.bid)
Andy	0
Art	0
Bob	0
Brutus	0
Dustin	1
Horatio	0
Lubber	0
Rusty	1
Zorba	0

- LEFT indica que todas as linhas de Sailors vão aparecer mesmo as que não têm reserva essas são juntas com valores nulos nas colunas de R

Junções exteriores

- Variantes - Considerando a junção de T1 com T2, por esta ordem
 - Junção exterior esquerda: inclui todos os valores de T1
 - Junção exterior direita: inclui todos os valores de T2
 - Junção exterior completa: inclui todos os valores de T1 e T2
- Exemplos
 - Todos os marinheiros e suas reservas, mesmo os que não fizeram reservas
`SELECT * FROM sailors S LEFT OUTER JOIN reserves R ON (S.sid = R.sid)`
 - Todas as reservas e seus marinheiros, mesmo as que não têm marinheiros
`SELECT * FROM sailors S RIGHT OUTER JOIN reserves R ON (S.sid = R.sid)`

Como todas as reservas têm de ter um marinheiro (ver chave primária) esta junção exterior direita é equivalente à junção interior

MySQL exemplos: tabelas t1 e t2

```
-- t1  
id name  
1 Tim  
2 Marta
```

```
-- t2  
id name  
1 Tim  
3 Katarina
```

```
SELECT *  
FROM `t1`  
INNER JOIN `t2` ON `t1`.`id` = `t2`.`id`;
```

1	Tim	1	Tim
---	-----	---	-----

```
SELECT *  
FROM `t1`  
LEFT OUTER JOIN `t2` ON `t1`.`id` = `t2`.`id`;
```

1	Tim	1	Tim
2	Marta	NULL	NULL

```
SELECT *  
FROM `t1`  
RIGHT OUTER JOIN `t2` ON `t1`.`id` = `t2`.`id`;
```

1	Tim	1	Tim
NULL	NULL	3	Katarina

```
SELECT *  
FROM `t1`  
LEFT OUTER JOIN `t2` ON `t1`.`id` = `t2`.`id`
```

UNION

```
SELECT *  
FROM `t1`  
RIGHT OUTER JOIN `t2` ON `t1`.`id` = `t2`.`id`;
```

1	Tim	1	Tim
2	Marta	NULL	NULL
NULL	NULL	3	Katarina

Introdução às Bases de Dados

Modelo Relacional – II

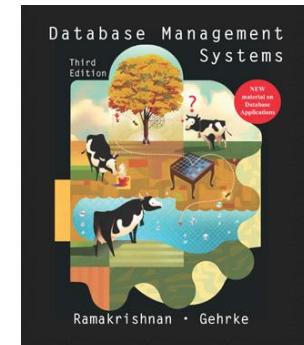
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Restrições de Integridade
 - Domínio e coluna
 - Entidade ou chave
 - Referencial ou chave estrangeira
 - Violações às Restrições de Chave
 - Verificação de Restrições de Integridade
- Referências
 - R. Ramakrishnan (**capítulo 3, secção 3.2, 3.3 e**
capítulo 5, secção 5.7)

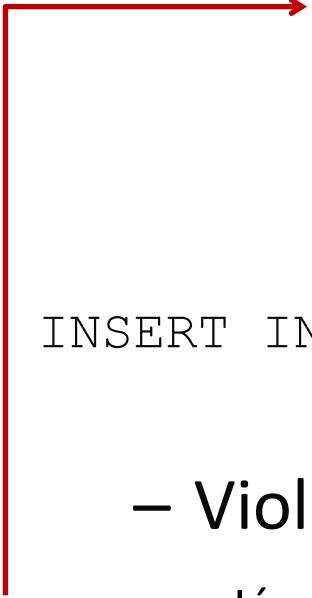


Tentativas de Violação às Restrições

```
CREATE TABLE Enrolled(  
    studid INTEGER(10),  
    cid      CHAR(20),  
    grade    CHAR(1),  
    CONSTRAINT pk_enrolled PRIMARY KEY (studid, cid),  
    CONSTRAINT fk_enrolled_sid  
        FOREIGN KEY (studid) REFERENCES Students (sid) )
```

```
CREATE TABLE Students(  
    sid      INTEGER(4),  
    name     VARCHAR(50),  
    ...  
    CONSTRAINT pk_student PRIMARY KEY (sid) )
```

Tentativas de Violação às Restrições de Chave



<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
INSERT INTO Students (sid, name, login, age, gpa)
    VALUES (53688, 'Mike', 'mike@ee', 17, 3.4)
```

- Viola a restrição de chave primária

 Já existe uma linha com *sid*=53688

- O comando é rejeitado pelo SGBD

 MySQL said: #1062 - Duplicate entry '53688' for key 'PRIMARY'

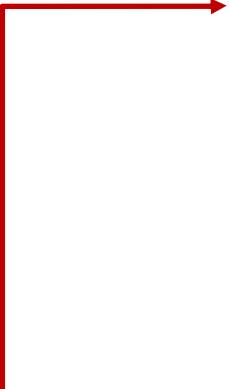
Tentativas de Violação às Restrições de Chave

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
INSERT INTO Students (sid, name, login, age, gpa)
    VALUES (null, 'Mike', 'mike@ee', 17, 3.4)
```

- Violação da restrição de chave primária
 - chave primária não pode conter NULL
 - MySQL said: #1048 - Column 'sid' cannot be null
- Violação semelhante para restrições de domínio

Tentativas de Violação às Restrições de Chave



<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

```
UPDATE Students S SET S.sid = 50000 WHERE S.sid = 53688
```

- Viola a restrição de chave primária

Já existe uma linha com *sid*=50000

- O comando é rejeitado pelo SGBD

MySQL said: #1062 - Duplicate entry '50000' for key 'PRIMARY'

Tentativas de Violação às Restrições de Chave

Students (**sid**: integer, name: string, login: string, age: integer, gpa: real)

Enrolled (**studid**: integer, cid: string, grade: string)

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae203	B	53832	53666	Jones	jones@cs	18	3.4
Topology112	A	53650	53688	Smith	smith@ee	18	3.2
History105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation)

Tabela referenciadora

Students (Referenced relation)

Tabela referenciada

Tentativas de Violação às Restrições

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic 101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae 203	B	53832	53666	Jones	jones@cs	18	3.4
Topology 112	A	53650	53688	Smith	smith@ee	18	3.2
History 105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation) Students (Referenced relation)

```
DELETE FROM Students WHERE (sid = 53831)
```

- Violação da restrição de integridade referencial
O estudante *sid=53831* não é removido por estar aprovado numa disciplina
- O comando é rejeitado pelo SGBD
MySQL said: #1451 - Cannot delete or update a parent row: a foreign key constraint fails
- Assumindo opção FOREIGN KEY ... REFERENCES ... **ON DELETE NO ACTION**

Tentativas de Violação às Restrições

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae203	B	53832	53666	Jones	jones@cs	18	3.4
Topology112	A	53650	53688	Smith	smith@ee	18	3.2
History105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation) Students (Referenced relation)

```
DELETE FROM Students WHERE (sid = 53831)
```

- E se existir a opção FOREIGN KEY ... REFERENCES ... **ON DELETE CASCADE**
- Não há violação da restrição de integridade referencial
 - O estudante *sid=53831* é removido e todas as linhas respetivas de *Enrolled*

Tentativas de Violação às Restrições

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic 101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae 203	B	53832	53666	Jones	jones@cs	18	3.4
Topology 112	A	53650	53688	Smith	smith@ee	18	3.2
History 105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation) Students (Referenced relation)

```
UPDATE Students SET sid = 80000 WHERE (sid = 53831)
```

- Violação da restrição de integridade referencial
 - A chave estrangeira tem de referenciar um estudante que exista
- O comando é rejeitado pelo SGBD
 - MySQL said: #1451 - Cannot delete or update a parent row: a foreign key constraint fails
- Assumindo opção FOREIGN KEY ... REFERENCES ... **ON UPDATE NO ACTION**

Tentativas de Violação às Restrições

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae203	B	53832	53666	Jones	jones@cs	18	3.4
Topology112	A	53650	53688	Smith	smith@ee	18	3.2
History105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation) Students (Referenced relation)

```
INSERT INTO Enrolled (cid, grade, studid)
VALUES ('Hindi101', 'B', 51111)
```

- Violação da restrição de integridade referencial
Não existe uma linha em *Students* com *sid*=51111
- O comando é rejeitado pelo SGBD
MySQL said: #1452 - Cannot add or update a child row: a foreign key constraint fails

Tentativas de Violação às Restrições

Foreign key			Primary key				
cid	grade	studid	sid	name	login	age	gpa
Carnatic 101	C	53831	50000	Dave	dave@cs	19	3.3
Reggae 203	B	53832	53666	Jones	jones@cs	18	3.4
Topology 112	A	53650	53688	Smith	smith@ee	18	3.2
History 105	B	53666	53650	Smith	smith@math	19	3.8
			53831	Madayan	madayan@music	11	1.8
			53832	Guldu	guldu@music	12	2.0

Enrolled (Referencing relation) Students (Referenced relation)

- **Remoções** de linhas da tabela ***Enrolled*** não violam a restrição de integridade referencial
- Mas **inserções e atualizações** podem violar

Tentativas de Violação de Integridade Referencial

- Remoção (DELETE) e atualização (UPDATE) de linhas em tabelas referenciadas
 - Pode gerar problemas nas chaves estrangeiras das tabelas referenciadoras
 - Exemplo: Aluno deixa de existir ou muda de número
- Opções para preservação da integridade referencial
FOREIGN KEY REFERENCES... ON DELETE (ou UPDATE)
 - NO ACTION -- opção por omissão
Linhos apenas são removidas/atualizadas se não referenciadas na chave estrangeira
 - CASCADE
Linhos na tabela referenciadora também são apagadas/atualizadas
 - SET NULL
Os valores correspondentes da chave estrangeira são colocados a NULL

Verificação de Restrições de Integridade

- Modo imediato
 - O SGBD verifica as RI após a execução de cada comando SQL
 - Comando SQL: SET CONSTRAINTS ... IMMEDIATE
- Modo diferido
 - O SGBD verifica as RI no final da execução de uma transação
 - Comando SQL: SET CONSTRAINT ... DEFERRED
 - Útil para operações interdependentes que temporariamente criam incoerência

Exemplo: Departamento tem obrigatoriamente um chefe. Empregado pertence obrigatoriamente a um departamento

Regras de Integridade - Resumo

- **Integridade de Domínio**
 - Cada atributo de uma relação tem um domínio
 - O valor desse atributo para todos os tuplos terá de pertencer SEMPRE a esse domínio ou ser NULL
- **Integridade da Chave**
 - Dois tuplos distintos de uma relação não podem ter um conjunto de valores iguais nos atributos da chave
 - Nenhum subconjunto de atributos de uma chave é uma chave candidata
- **Integridade de Entidade**
 - Nenhum atributo componente de uma chave primária poderá em algum momento ter valor NULL

Regras de Integridade - Resumo

- **Integridade Referencial**

Numa relação, qualquer ocorrência de uma chave estrangeira deverá obrigatoriamente:

- Existir como ocorrência da chave primária da relação à qual se refere.
- Ou ter todos os atributos NULL.

- **Integridade Aplicacional (adicional ou semântica)**

- Qualquer outra regra a que as ocorrências de uma determinada base de dados deverão obedecer e que não é abrangida pelos tipos atrás mencionados.

Introdução às Bases de Dados

Restrições de Integridade Complexas

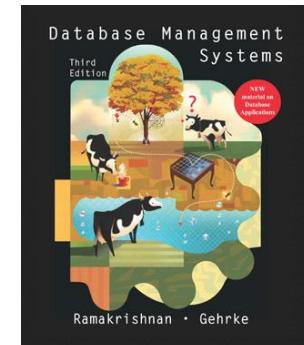
FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

Sumário e Referências

- Sumário
 - Restrições complexas em SQL (CHECK...)
 - Criação de domínios
 - Criação de tipos
 - Restrições numa tabela
 - Asserções - restrições sobre várias tabelas
 - *Triggers* e bases de dados ativas
 - Definição, exemplos e problemas com *triggers*
 - CHECK vs *triggers*
- Referências
 - R. Ramakrishnan (**capítulo 5, secção 5.7, 5.8 e 5.9**)



Regras de Integridade Complexas

- Modelo relacional fornece restrições de integridade
 - Domínio, Coluna, Entidade (ou chave), Referencial, *User-Defined*
- **Suporte para restrições complexas**
 - Domínio: definição de domínios para além dos fornecidos pelo SGBD
 - Coluna: uso de SELECTs em CHECKs de tabelas
 - *User-defined*: definição de asserções e *triggers*
- Asserções
 - CHECKs independentes de tabelas
- *Triggers*
 - Procedimentos invocados aquando de escritas em tabelas

Definição de Novos Domínios

- Definição de novos domínios

- Para manter coerência nas colunas
 - Ex. novo domínio com nºs inteiros entre 1 e 10

```
CREATE DOMAIN ratingval INTEGER  
    DEFAULT 1  
CHECK ( VALUE >= 1 AND VALUE <= 10 )
```

- Aplicação do novo domínio em coluna de tabela

- Ex. CREATE TABLE Sailors (... , rating **ratingval**, ...)
 - Se INSERT em Sailors omitir o valor de rating, este é preenchido com 1

- Valores de rating podem ser comparados com os de colunas do tipo INTEGER

- Limitação conceptual, pois os domínios são diferentes

Definição de Tipos

- Comando CREATE TYPE define um novo tipo de dados abstrato
 - Necessita de métodos próprios para suportar comparações, adições, ... mesmo que baseado em domínios simples, como INTEGER
 - Para evitar comparações entre tipos diferentes
 - Exemplo:
- ```
CREATE TYPE ratingtype AS INTEGER
```
- Domain vs Type
  - Colunas ratingtype **não** podem ser comparadas (ou operadas) com colunas do tipo INTEGER
  - Enquanto, colunas ratingval podem ser comparadas (ou operadas) com colunas do tipo INTEGER

# Restrições Complexas numa Tabela

---

- Definição de tabelas pode incluir cláusulas CHECK

```
CREATE TABLE Sailors (sid INTEGER,...,rating INTEGER,
 CHECK (rating >= 1 AND rating <= 10))
```

- Definição de tabelas pode incluir restrições mais complexas

- Com a consulta a outras tabelas
- Exemplo: reservas de barcos, **exceto para barcos com nome Interlake**

```
CREATE TABLE Reserves (sid INTEGER, bid INTEGER, day DATE,
 PRIMARY KEY (sid, bid),
 FOREIGN KEY (sid) REFERENCES Sailors (sid),
 FOREIGN KEY (bid) REFERENCES Boats (bid),
 CONSTRAINT noInterlakeRes
 CHECK ('Interlake' <> (SELECT B.bname
 FROM Boats B
 WHERE B.bid = Reserves.bid)))
```

- Condição verificada para cada INSERT ou UPDATE na tabela Reserves

# Restrições Complexas numa Tabela

---

- Outro exemplo com restrição complexa
  - O **número de barcos** e de **marinheiros** não pode ultrapassar os 100

```
CREATE TABLE Sailors (
 ...
CONSTRAINT smallClub
CHECK ((SELECT COUNT(S.sid) FROM Sailors S)
 +
 (SELECT COUNT(B.bid) FROM Boats B) < 100))
```

- Problemas?

# Restrições Complexas numa Tabela

---

- Outro exemplo com restrição complexa
  - O **número de barcos** e de **marinheiros** não pode ultrapassar os 100

```
CREATE TABLE Sailors (
 ...
CONSTRAINT smallClub
CHECK ((SELECT COUNT(S.sid) FROM Sailors S)
 + (SELECT COUNT(B.bid) FROM Boats B) < 100))
```

- Problemas
  - Restrição associada a **Sailors**, apesar de envolver também **Boats**  
Decisão de colocar restrição em Sailors é arbitrária
  - Enquanto Sailors estiver vazia, condição não precisa de ser verificada  
Número de barcos pode crescer indefinidamente (e ficar superior a 100)
  - Solução: criação de condições não associadas a uma qualquer tabela - asserções

# Asserções

---

- Restrições que não estão associadas a uma qualquer tabela
  - Definidas ao mesmo nível das tabelas no esquema de dados
  - Apropriadas para restrições que abrangem múltiplas tabelas
- Exemplo para a restrição complexa do slide anterior

```
CREATE ASSERTION smallClub
 CHECK ((SELECT COUNT (S.sid) FROM Sailors S)
 +(SELECT COUNT (B. bid) FROM Boats B)<100)
```

- Não está associada a nenhuma tabela
- Condição verificada para cada INSERT ou UPDATE em Sailors ou Boats

# Triggers

---

- Procedimento que é automaticamente despoletado quando se realizam escritas específicas
  - **Evento** de escrita ativa **condição** que permite, ou não, execução de **ação**
- **Evento**
  - Tipo de escrita na base de dados que faz ativar o *trigger*
  - Tipos de escrita: qualquer combinação de INSERT, UPDATE, e DELETE
  - Escritas podem ser numa tabela inteira ou em colunas específicas
  - Opções de ativação do *trigger*
    - Antes ou depois da escrita se concretizar
    - Uma só vez para um bloco inteiro de escritas ou para cada linha escrita
- **Condição** (opcional)
  - Uma interrogação ou um teste verificado aquando da ativação do *trigger*
- **Ação**
  - Código do procedimento executado quando o *trigger* é ativado e a condição anterior satisfeita

# Exemplo de *Trigger*

---

```
CREATE TRIGGER init_count
BEFORE INSERT ON Students ← Evento
```

```
DECLARE
 count INTEGER
BEGIN
 count := 0; ← Ação
END
```

Nota: trigger sem condição

# Exemplo de *Trigger*

---

```
CREATE TRIGGER incr_count
AFTER INSERT ON Students
WHEN (new.age < 18)
 -- 'new' is just-inserted tuple
FOR EACH ROW
BEGIN
 count := count + 1;
END
```

← **Evento**

← **Condição**

← **Ação** (Oracle PL/SQL Syntax)

-- Row-level trigger

## Observações

1. Também existe old e new para referir o valor antes e depois de um UPDATE

# Triggers: MySQL syntax

---

```
CREATE
 [DEFINER = user]
 TRIGGER trigger_name
 trigger_time trigger_event
 ON tbl_name FOR EACH ROW
 [trigger_order]
 trigger_body

trigger_time: { BEFORE | AFTER }

trigger_event: { INSERT | UPDATE | DELETE }

trigger_order: { FOLLOWS | PRECEDES } other_trigger_name
```

# MySQL: exemplo de Trigger

---

```
CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));

CREATE TRIGGER ins_sum
 BEFORE INSERT ON account
 FOR EACH ROW SET @sum = @sum + NEW.amount;

SET @sum = 0;

INSERT INTO account VALUES
 (137,14.98),
 (141,1937.50),
 (97,-100.00);

SELECT @sum AS 'Total amount inserted';
Total amount inserted
1852.48
```

# MySQL: exemplo de trigger

---

```
CREATE TRIGGER upd_check
BEFORE UPDATE ON account
FOR EACH ROW
BEGIN
 IF NEW.amount < 0 THEN
 SET NEW.amount = 0;
 ELSEIF NEW.amount > 100 THEN
 SET NEW.amount = 100;
 END IF;
END;
```

# Oracle: Exemplo de Trigger

```
CREATE TRIGGER verifica_aumento_vencimento
 BEFORE UPDATE OF vencimento ON empregado
 FOR EACH ROW
 WHEN (new.vencimento > old.vencimento)
DECLARE
 vencimento_do_chefe NUMBER := NULL;
BEGIN -- Supõe-se que só existe um chefe.
 SELECT vencimento INTO vencimento_do_chefe
 FROM empregado WHERE (categoria = 'chefe');
 IF (:new.vencimento > vencimento_do_chefe) THEN
 RAISE_APPLICATION_ERROR(-20001, 'Não pode ganhar mais que o chefe!')
 END IF;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
 RAISE_APPLICATION_ERROR(-20002, 'A empresa não tem chefe!');
 WHEN OTHERS THEN RAISE;
END;
```

The diagram illustrates the components of the trigger definition:

- Evento que ativa o trigger**: Points to the `BEFORE UPDATE` clause.
- Condição após ativação (opcional)**: Points to the `WHEN` clause.
- Ação do trigger (procedimento)**: Points to the `DECLARE`, `BEGIN`, and `EXCEPTION` blocks.
- Variável new guarda a linha que resultaria do UPDATE, e old refere a linha antes do UPDATE**: Points to the `RAISE_APPLICATION_ERROR` statement in the `IF` block.
- Exceções, se lançadas, cancelam o UPDATE**: Points to the `RAISE_APPLICATION_ERROR` statement in the `EXCEPTION` block.

(fonte: António Ferreira, SIBD 2016)

# Bases de Dados Ativas e *Triggers*

---

- Base de dados ativa
  - Base de dados com *triggers* associados
- Usos típicos de *triggers*
  - Restrições de integridade complexas
  - Autorizações de acesso e auditoria de escritas em tabelas
    - Ex. que utilizadores escreveram em certa tabela e a que horas
  - Réplicas síncronas de tabelas
- Definição de *triggers* – responsável e problemas
  - Tipicamente definidos (ou autorizados) pelo DBA = *DataBase Administrator*
  - Razão: consequências do uso de *triggers* podem ser difíceis de entender
    - Vários *triggers* podem ser ativados em simultâneo, por ordem arbitrária
    - Ação de um *trigger* pode ativar outros *triggers* (*triggers* recursivos)
  - Um uso criterioso de **restrições de integridade pode frequentemente substituir/evitar uso de triggers**

# CHECKS vs. Triggers

---

- **CHECKs**
  - Declarativos
  - Mais fáceis de entender
  - Mais eficientes pois podem ser otimizados pelos SGBDs
  - Restrições de integridade verificadas em permanência
    - Para quaisquer operações de escrita na base de dados
- **Triggers**
  - Procedimentais (requer saber programar)
  - Desempenho depende da qualidade do programador
    - Sempre que possível, devem ser usadas restrições de integridade declarativas
  - Podem ser usados para outros fins (além de manter integridade)
    - Autorizações de acesso, auditoria, estatísticas, replicação de dados...
  - Restrições de integridade verificadas para escritas específicas em tabelas
    - Necessário cuidado para cobrir todos os cenários possíveis de escrita de dados

# **Introdução às Bases de Dados**

Normalização de Esquemas Relacionais

FCUL, Departamento de Informática

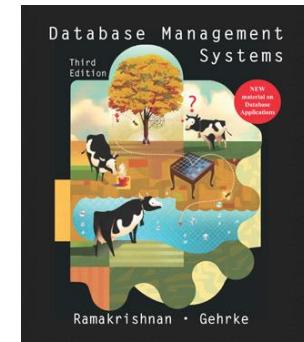
Ano Letivo 2021/2022

Ana Paula Afonso

# Sumário e Referências

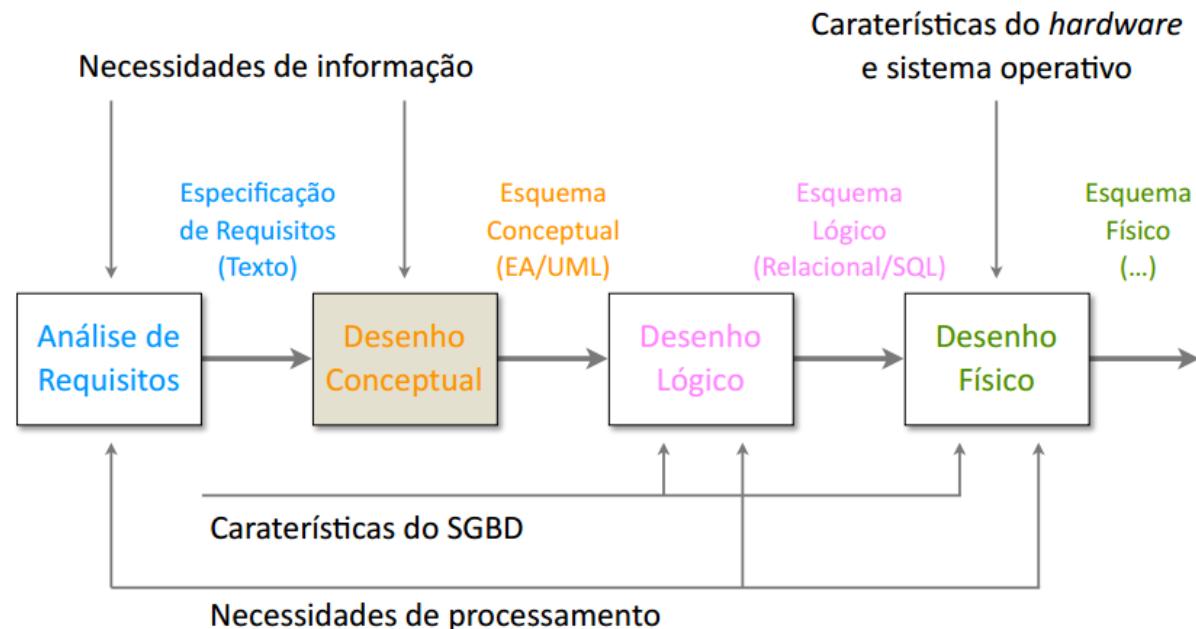
---

- Sumário
  - Motivação para a normalização de esquemas relacionais
  - Problemas da redundância de dados
    - Exemplo de relação com redundância
  - Dependências funcionais
  - Etapas da normalização
    - Principais formas normais
    - Exemplo de normalização
- Referências
  - R. Ramakrishnan (**capítulo 19, secção 19.1 - 19.4**)



# Motivação da Normalização

- Após a construção do modelo conceptual dos dados (Modelo E/A) é feita a transformação para um modelo lógico (Esquema Relacional)
- O esquema relacional obtido representa a estrutura da informação de um modo natural e completo. Mas terá o mínimo de redundância possível?



Fonte: António Ferreira, Guião SIBD, 2016

# Normalização

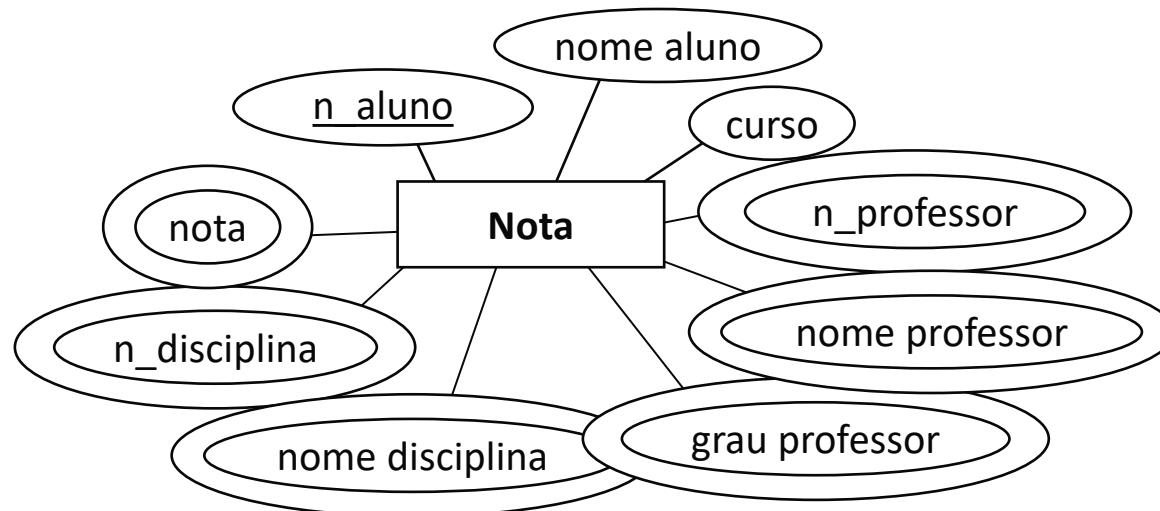
---

- Dado um esquema relacional por vezes
  - Os mesmos dados existem armazenados em múltiplos locais (**redundância**)
  - Existência de **anomalias** aquando da escrita de dados (incoerências dos dados)
- **Normalização** permite melhorar a qualidade do esquema através da
  - Eliminação da redundância dos dados
  - Prevenção de anomalias
- **Normalização** é uma abordagem que envolve a
  - Decomposição sucessiva de relações até se obter um conjunto de relações
  - ... sem redundâncias e que permitam inserções, atualizações e remoções sem incoerências

# Exemplo de Relação não Normalizada

---

Possível EA para a tabela Nota (**INCORRETO**)



Notação: atributo com vários valores



# Anomalias de Relação não Normalizada

| <u>n_aluno</u> | nome aluno | curso       | <u>n_disciplina</u> | nome disciplina | <u>n_professor</u> | nome professor | grau professor | nota |
|----------------|------------|-------------|---------------------|-----------------|--------------------|----------------|----------------|------|
| 21934          | Antunes    | Informática | 04                  | Álgebra         | 21                 | Gil Algébrico  | PA             | 15   |
|                |            |             | 14                  | Análise Sist.   | 87                 | Ana Listada    | PC             | 12   |
|                |            |             | 23                  | P.Linear        | 43                 | Plínio         | AS             | 16   |
| 42346          | Bernardo   | Matemática  | 08                  | Topologia       | 32                 | Topo Lógico    | AE             | 10   |
|                |            |             | 04                  | Álgebra         | 21                 | Gil Algébrico  | PA             | 12   |
|                |            |             | 12                  | Geometria       | 21                 | Gil Algébrico  | PA             | 18   |
|                |            |             | 16                  | Lógica          | 32                 | Topo Lógico    | AE             | 13   |
| 54323          | Correia    | Estatística | 04                  | Álgebra         | 21                 | Gil Algébrico  | PA             | 11   |
|                |            |             | 08                  | Topologia       | 32                 | Topo Lógico    | AE             | 10   |
| ...            | ...        | ...         | ...                 | ...             | ...                | ...            | ...            | ...  |

- **Inserção** de um novo professor requer indicação de outros dados
- **Atualização** do grau do professor tem de afetar várias linhas
- **Remoção** de um professor elimina os dados da disciplina a que está afeto

# Problemas da Redundância dos Dados

---

- A redundância introduz problemas (**anomalias**) de coerência e manutenção
  - **Anomalia de inserção** - informação que é independente não pode ser inserida de forma separada na Base de Dados
  - **Anomalia de atualização** - a modificação de informação num conjunto de ocorrências implica a criação de inconsistências ou a necessidade de alterar informação noutras instâncias da Base de Dados que são independentes das primeiras
  - **Anomalia de remoção** - a remoção de informação acarreta a perda de outra informação independente contida na Base de Dados

# Principais Formas Normais

---

- **Primeira forma normal: 1FN**
  - Colunas da relação guardam apenas **um** valor por linha
  - Tipicamente verificado em BD relacionais
- **Segunda forma normal: 2FN = 1FN +**
  - Colunas não pertencentes às chaves candidatas da relação...
  - ...dependem da **totalidade** das colunas de cada chave
  - Trivial, se chaves da relação tiverem apenas uma coluna
- **Terceira forma normal: 3FN = 2FN +**
  - Colunas não pertencentes às chaves candidatas...
  - ...dependem **apenas** das chaves candidatas

# Exemplo de Relação não Normalizada

---

## Nota

| <u>n_aluno</u> | nome aluno | curso       | <u>n_disciplina</u> | nome disciplina | <u>n_professor</u> | nome professor | grau professor | nota |
|----------------|------------|-------------|---------------------|-----------------|--------------------|----------------|----------------|------|
| 21934          | Antunes    | Informática | 04                  | Álgebra         | 21                 | Gil Algébrico  | PA             | 15   |
|                |            |             | 14                  | Análise Sist.   | 87                 | Ana Listada    | PC             | 12   |
|                |            |             | 23                  | P.Linear        | 43                 | Plínio         | AS             | 16   |
| 42346          | Bernardo   | Matemática  | 08                  | Topologia       | 32                 | Topo Lógico    | AE             | 10   |
|                |            |             | 04                  | Álgebra         | 21                 | Gil Algébrico  | PA             | 12   |
|                |            |             | 12                  | Geometria       | 21                 | Gil Algébrico  | PA             | 18   |
|                |            |             | 16                  | Lógica          | 32                 | Topo Lógico    | AE             | 13   |
| 54323          | Correia    | Estatística | 04                  | Álgebra         | 21                 | Gil Algébrico  | PA             | 11   |
|                |            |             | 08                  | Topologia       | 32                 | Topo Lógico    | AE             | 10   |
| ...            | ...        | ...         | ...                 | ...             | ...                | ...            | ...            | ...  |

# Passagem/Decomposição para a 1<sup>a</sup> Forma Normal (1FN)

|                           |               |       |                               |                    |                               |                   |                   |      |
|---------------------------|---------------|-------|-------------------------------|--------------------|-------------------------------|-------------------|-------------------|------|
| <u>n_</u><br><b>aluno</b> | nome<br>aluno | curso | <u>n</u><br><b>disciplina</b> | nome<br>disciplina | <u>n_</u><br><b>professor</b> | nome<br>professor | grau<br>professor | nota |
|---------------------------|---------------|-------|-------------------------------|--------------------|-------------------------------|-------------------|-------------------|------|

**Aluno**

| Chave                     |          |             |
|---------------------------|----------|-------------|
| <u>n_</u><br><b>aluno</b> | nome     | curso       |
| 21934                     | Antunes  | Informática |
| 42346                     | Bernardo | Matemática  |
| 54323                     | Correia  | Estatística |
| ...                       | ...      | ...         |

**Nota**

| Chave                     |                                |                    |                               |                   |                   |      |
|---------------------------|--------------------------------|--------------------|-------------------------------|-------------------|-------------------|------|
| <u>n_</u><br><b>aluno</b> | <u>n_</u><br><b>disciplina</b> | nome<br>disciplina | <u>n_</u><br><b>professor</b> | nome<br>professor | grau<br>professor | nota |
| 21934                     | 04                             | Álgebra            | 21                            | Gil Algébrico     | PA                | 15   |
| 21934                     | 14                             | Análise Sist.      | 87                            | Ana Listada       | PC                | 12   |
| 21934                     | 23                             | P.Linear           | 43                            | Plínio            | AS                | 16   |
| 42346                     | 08                             | Topologia          | 32                            | Topo Lógico       | AE                | 10   |
| 42346                     | 04                             | Álgebra            | 21                            | Gil Algébrico     | PA                | 12   |
| 42346                     | 12                             | Geometria          | 21                            | Gil Algébrico     | PA                | 18   |
| 42346                     | 16                             | Lógica             | 32                            | Topo Lógico       | AE                | 13   |
| 54323                     | 04                             | Álgebra            | 21                            | Gil Algébrico     | PA                | 11   |
| 54323                     | 08                             | Topologia          | 32                            | Topo Lógico       | AE                | 10   |
| ...                       | ...                            | ...                | ...                           | ...               | ...               | ...  |

# Anomalias da Relação Nota

---

*Chave*

| <u>n</u><br><u>aluno</u> | <u>n</u><br><u>disciplina</u> | nome<br>disciplina | <u>n</u><br><u>professor</u> | nome<br>professor | grau<br>professor | <b>Nota</b> |
|--------------------------|-------------------------------|--------------------|------------------------------|-------------------|-------------------|-------------|
| 21934                    | 04                            | Álgebra            | 21                           | Gil Algébrico     | PA                | 15          |
| 21934                    | 14                            | Análise Sist.      | 87                           | Ana Listada       | PC                | 12          |
| 21934                    | 23                            | P.Linear           | 43                           | Plínio            | AS                | 16          |
| 42346                    | 08                            | Topologia          | 32                           | Topo Lógico       | AE                | 10          |
| 42346                    | 04                            | Álgebra            | 21                           | Gil Algébrico     | PA                | 12          |
| 42346                    | 12                            | Geometria          | 21                           | Gil Algébrico     | PA                | 18          |
| 42346                    | 16                            | Lógica             | 32                           | Topo Lógico       | AE                | 13          |
| 54323                    | 04                            | Álgebra            | 21                           | Gil Algébrico     | PA                | 11          |
| 54323                    | 08                            | Topologia          | 32                           | Topo Lógico       | AE                | 10          |
| ...                      | ...                           | ...                | ...                          | ...               | ...               | ...         |

# Anomalias da Relação Nota

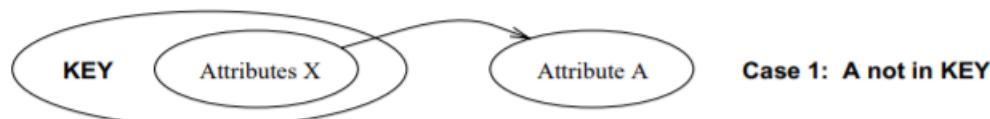
---

- **Inserção**
  - Não se pode inserir informação sobre uma nova disciplina ...
  - .... enquanto não existirem alunos inscritos para a nova disciplina  
(o atributo n\_aluno faz parte da chave da relação)
- **Remoção**
  - Quando se apaga a informação sobre todos os alunos que têm uma determinada disciplina ....
  - ... perde-se toda a informação dessa disciplina e do respetivo professor
- **Atualização**
  - Quando se modifica o nome de uma disciplina ...
  - ... é necessário percorrer toda a relação e fazer essa modificação para todos os alunos que tivessem essa disciplina
  - No caso de falhar a aplicação de modificação em alguma ocorrência, então ter-se-iam dados inconsistentes

# Principais Formas Normais

---

- Primeira forma normal: 1FN
  - Colunas da relação guardam apenas **um** valor por linha
  - Tipicamente verificado em bases de dados relacionais
- Segunda forma normal: 2FN = 1FN +
  - Se **não existirem dependências funcionais entre subconjuntos próprios da chave e atributos não chave**
  - Colunas não pertencentes às chaves candidatas da relação...
  - ...dependem da **totalidade** das colunas de cada chave
  - As relações com chaves simples (1 único atributo) que estejam na 1FN estão sempre na 2FN

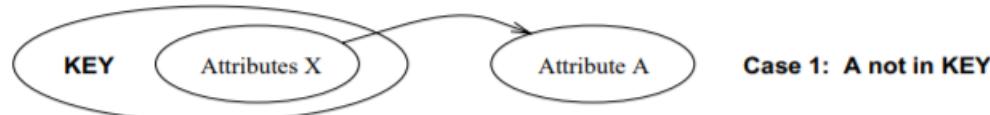
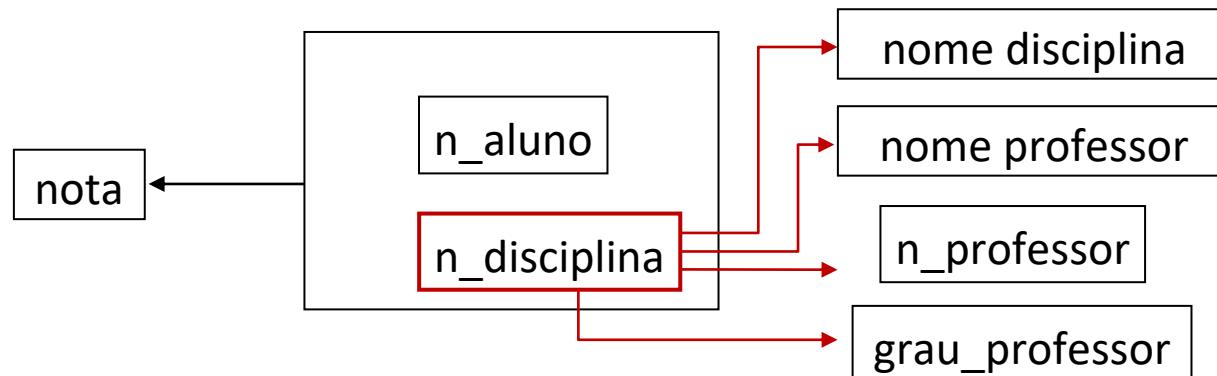


# Decomposição na 2FN

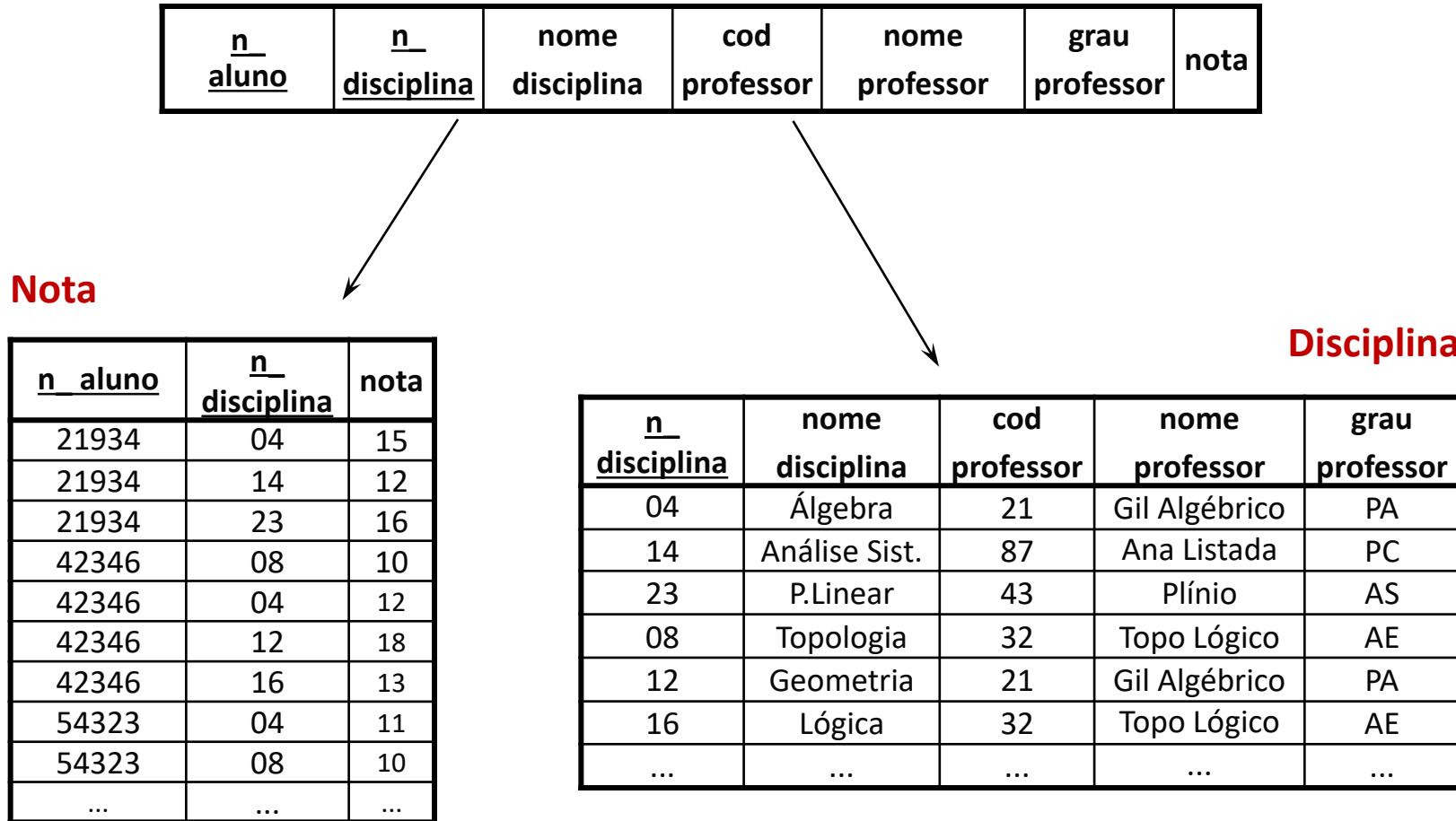
Disciplina (n\_disciplina, n\_aluno, nota,  
nome\_disciplina, n\_professor, nome\_professor, grau\_professor)

Existem **DFs** entre atributo subconjunto da chave (n\_disciplina) e atributos não chave

n\_disciplina → nome\_disciplina, n\_professor, nome\_professor, grau\_professor



# Decomposição na 2FN



# Decomposição na 2FN

---

- O esquema atual tem atualmente 3 relações
  - Aluno, Disciplina e Nota
  - Inicialmente era apenas 1 relação (Nota)
  - As 3 relações estão na 2FN ...
  - ... colunas não chave dependem da totalidade da chave
- Contudo, ainda existem anomalias na relação **Disciplina**

# Anomalias na Relação Disciplina

---

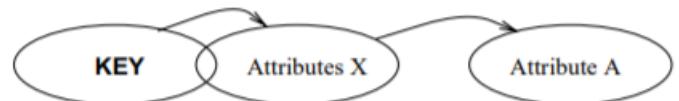
- **Inserção** de um novo professor exige que lhe seja atribuída pelo menos uma disciplina (a chave é n\_disciplina)
- **Remoção** de uma disciplina elimina a informação do professor  
se esse professor não dá aulas a outras disciplinas, então perde-se a sua informação
- **Atualização** do grau de um professor exige efetuar a alteração em todas as disciplinas atribuídas a esse professor

| n_disciplina | nome_disciplina | n_professor | nome_professor | grau_professor |
|--------------|-----------------|-------------|----------------|----------------|
| 04           | Álgebra         | 21          | Gil Algébrico  | PA             |
| 14           | Análise Sist.   | 87          | Ana Listada    | PC             |
| 23           | P.Linear        | 43          | Plínio         | AS             |
| 08           | Topologia       | 32          | Topo Lógico    | AE             |
| 12           | Geometria       | 21          | Gil Algébrico  | PA             |
| 16           | Lógica          | 32          | Topo Lógico    | AE             |
| ...          | ...             | ...         | ...            | ...            |

# Principais Formas Normais

---

- **Primeira forma normal: 1FN**
  - Colunas da relação guardam apenas **um** valor por linha
  - Tipicamente verificado em bases de dados relacionais
- **Segunda forma normal: 2FN = 1FN +**
  - Colunas não pertencentes às chaves candidatas da relação...
  - ...dependem da **totalidade** das colunas de cada chave
  - Trivial, se chaves da relação tiverem apenas uma coluna
- **Terceira forma normal: 3FN = 2FN +**
  - Colunas não pertencentes às chaves candidatas...
  - ...dependem **apenas** das chaves candidatas
  - As relações com um único atributo não chave que estejam na 2FN estão sempre na 3FN

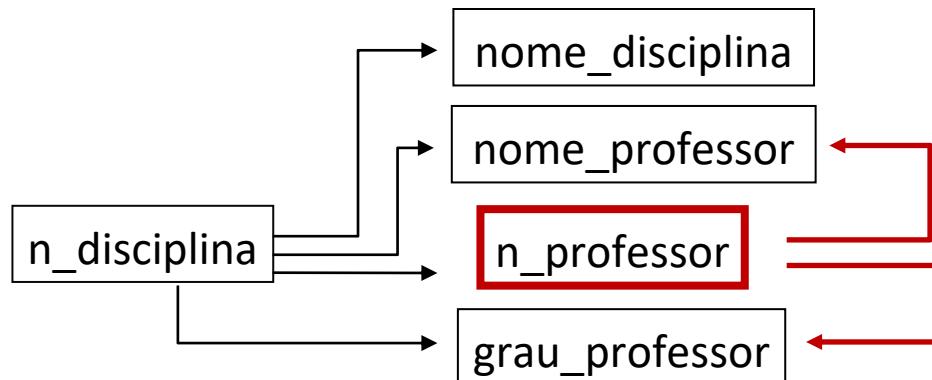


Case 1: A not in KEY

# Decomposição na 3FN

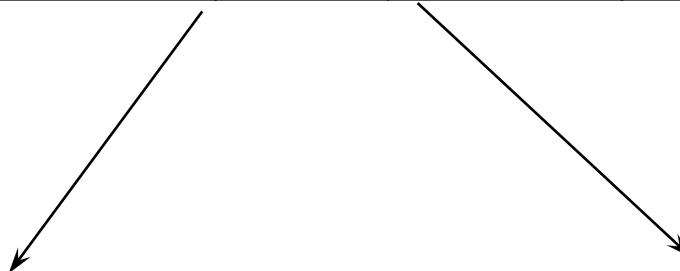
Disciplina (n\_disciplina, nome\_disciplina,  
n\_professor, nome\_professor, grau\_professor)

Existem as **DFs** entre um atributo não pertencente à chave (n\_professor) e atributos não chave  
 $\text{cod\_professor} \rightarrow \text{nome\_professor}, \text{grau\_professor}$



# Decomposição na 3FN

| <u>n</u><br>disciplina | nome<br>disciplina | cod<br>professor | nome<br>professor | grau<br>professor |
|------------------------|--------------------|------------------|-------------------|-------------------|
|------------------------|--------------------|------------------|-------------------|-------------------|



**Disciplina**

| <u>n</u><br>disciplina | nome<br>disciplina | cod<br>professor |
|------------------------|--------------------|------------------|
| 04                     | Álgebra            | 21               |
| 14                     | Análise Sist.      | 87               |
| 23                     | P.Linear           | 43               |
| 08                     | Topologia          | 32               |
| 12                     | Geometria          | 21               |
| 16                     | Lógica             | 32               |
| ...                    | ...                | ...              |

**Professor**

| cod<br>professor | nome<br>professor | grau<br>professor |
|------------------|-------------------|-------------------|
| 21               | Gil Algébrico     | PA                |
| 87               | Ana Listada       | PC                |
| 43               | Plínio            | AS                |
| 32               | Topo Lógico       | AE                |
| ...              | ...               | ...               |

# Decomposição na 3FN

---

- Esquema atual tem 4 relações

Aluno (n\_aluno, nome, curso)

Nota (n\_aluno, n\_disciplina, nota)

Disciplina (n\_disciplina, nome\_disciplina, n\_professor)

Professor (n\_professor, nome\_professor, grau\_professor)

- Esquema relacional está na 3FN

- Para cada relação, não existem dependências entre colunas não chave
  - Todas as dependências são das chaves candidatas para as outras colunas

- Esquema normalizado é que seria traduzido para comandos SQL

# **Introdução às Bases de Dados**

Tópicos de Gestão de Bases de Dados

FCUL, Departamento de Informática

Ano Letivo 2021/2022

Ana Paula Afonso

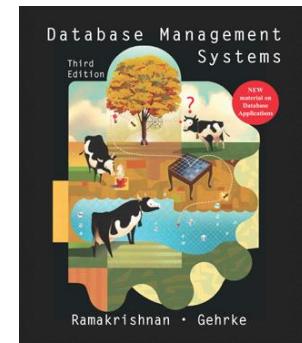
# Sumário e Referências

---

- Sumário
  - Bases de dados e SGBDs

Razões de utilização, vantagens e desvantagens
  - Componentes de um SGBD e Utilização de SGBD

Conceção de BD e desenvolvimento de aplicações  
Análise de dados  
Concorrência e robustez  
Eficiência e escalabilidade
- Referências
  - R. Ramakrishnan ([capítulo 1](#))



# Razões de Uso de SGBDs

---

- Suportam **conceitos úteis para a gestão de BDs**
  - Linguagem para criar, alterar, e consultar/analisar dados
  - Restrições para manter a integridade dos dados (ex. idade > 0)
  - Utilizadores e privilégios de acesso aos dados
  - Transações: programas que transformam o estado da BD
- Oferecem **mecanismos automáticos** de gestão de BDs
  - Aplicação de restrições de integridade aquando de alterações nos dados
  - Controlo de acesso à BD por vários utilizadores em simultâneo
  - Recuperação de faltas (ex. falta de energia elétrica)
  - Otimização do acesso aos dados, para respostas rápidas
- Suportam **grandes quantidades de dados e de transações**
  - Mecanismos desenhados para permitir expansão de capacidade

# Vantagens dos SGBD (1)

---

- **Independência dos dados**
  - Aplicações não estão expostas aos detalhes de como os dados estão representados e armazenados
  - SGBD disponibiliza uma visão abstrata dos dados
- **Acesso eficiente aos dados**
  - O SGBD utiliza uma variedade de técnicas sofisticadas para armazenar e recolher dados de uma forma eficiente

# Vantagens dos SGBD (2)

---

- **Integridade dos dados e segurança**
  - O SGBD pode aplicar restrições de integridade durante o acesso aos dados
- **Administração dos dados**
  - Profissionais experientes podem organizar a representação dos dados, por forma a minimizar a redundância e melhorar o armazenamento e recolha dos dados

# Vantagens dos SGBD (3)

---

- **Acesso concorrente e recuperação de falhas**
  - Acesso aos dados como se fosse acedido por um utilizador de cada vez
  - Minimiza os efeitos de falhas no sistema
- **Redução do tempo de desenvolvimento de aplicações**
  - Disponibiliza funções de acesso comuns
  - Interface de alto nível para os dados
  - Mais robusto:  
Tarefas executadas pelo SGBD não precisam de ser verificadas

# SGBD Desvantagens

---

- **Aplicações complexas de software**
- **Desempenho** inaceitável para algumas aplicações
  - Aplicações de **tempo-real**
- **Não** disponibiliza análise **flexível** de dados **texto**
- **Nem sempre** os benefícios dos SGBD são **necessários**

# Quando Não usar um SGBD

---

- Custos iniciais demasiado elevados
  - Investimentos em hardware, software e formação
  - Necessidade de definição e aplicação de políticas de segurança
- Bases de dados simples ou imutáveis (\*)
- Ausência de acessos concorrentes (\*)
- Requisitos de tempo-real incontornáveis
  - SGBDs não garantem execução num intervalo de tempo fixo
- Custo inicial do software pode ser reduzido
  - *Open source*: MySQL, MariaDB, PostgreSQL, ...
  - Versões grátis de produtos: Oracle Express, SQL Server Express, ...

(\*) uso de ficheiros - pode ser suficiente

# Bases de Dados noSQL

---

- NoSQL é uma abordagem recente para criar bases de dados
  - o termo NoSQL é usado para referir bases de dados não relacionais
  - são BD não tabulares e armazenam os dados de maneira diferente das tabelas relacionais
  - as BD NoSQL surgem em vários tipos com base no seu modelo de dados

p. e.: documento, chave-valor (*key-value*), *graph*,...
  - desenhadas para serem escaláveis para  
grandes quantidades de dados  
grandes quantidades de utilizadores
- Existem vários produtos NoSQL
  - exs: MongoDB, Neo4j, Apache Cassandra, Google Cloud BigTable

# Ranking dos SGBD

| Rank        |             |             | DBMS                                                                                                                                                                           | Database Model                                                                                                   |
|-------------|-------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| Dec<br>2021 | Nov<br>2021 | Dec<br>2020 |                                                                                                                                                                                |                                                                                                                  |
| 1.          | 1.          | 1.          | Oracle                                                                                        | Relational, Multi-model       |
| 2.          | 2.          | 2.          | MySQL                                                                                         | Relational, Multi-model       |
| 3.          | 3.          | 3.          | Microsoft SQL Server                                                                         | Relational, Multi-model       |
| 4.          | 4.          | 4.          | PostgreSQL   | Relational, Multi-model       |
| 5.          | 5.          | 5.          | MongoDB                                                                                       | Document, Multi-model         |
| 6.          | 6.          | ↑ 7.        | Redis                                                                                         | Key-value, Multi-model        |
| 7.          | 7.          | ↓ 6.        | IBM Db2                                                                                                                                                                        | Relational, Multi-model       |
| 8.          | 8.          | 8.          | Elasticsearch                                                                                                                                                                  | Search engine, Multi-model  |
| 9.          | 9.          | 9.          | SQLite                                                                                      | Relational                                                                                                       |
| 10.         | ↑ 11.       | ↑ 11.       | Microsoft Access                                                                                                                                                               | Relational                                                                                                       |

<https://db-engines.com/en/ranking>

# Utilização de SGBDs - questões essenciais

---

1. Conceção de Bases de Dados e Desenvolvimento de Aplicações
2. Análise de Dados
3. Concorrência e Robustez
4. Eficiência e Escalabilidade

# Concepção de Bases de Dados e Análise dos Dados

---

## Conceção de BDs

- Como se pode **descrever um caso real**
  - (ex. uma universidade) em termos de **dados** a guardar num SGBD?
- Que fatores devem ser considerados na decisão de como **organizar** os dados armazenados?
- Como **desenvolver aplicações** que recorrem a um SGBD?

## Análise dos Dados

- Como o utilizador pode obter resposta às suas questões através de **interrogações** aos dados no SGBD?

# Concorrência e Robustez / Eficiência e Escalabilidade

---

## Concorrência e Robustez

- Como é que um SGBD permite o **acesso concorrente** aos dados?
- Como é que este protege os dados num caso de uma **falha do sistema**? (ex. falta eletricidade)

## Eficiência e Escalabilidade

- Como é que um SGBD armazena **grandes coleções de dados** e executa interrogações nesses dados de uma forma eficiente? (ex. índices)
- Como lida com grande **quantidade de utilizadores e acessos**?

# SQL em Aplicações

---

- Os comandos SQL podem ser chamados através de uma **aplicação informática**
  - Escrita numa qualquer **linguagem host**
    - Java, C, PHP, Perl, Python, ...

# Um exemplo: PHP

---

```
<?php

 // Estabelece uma ligação com a base de dados usando o programa openconn.php
 // A variável $conn é inicializada com a ligação estabelecida
 include "openconn.php";

 $query = "INSERT INTO pessoa VALUES ('O_seu_nome', 50)";
 $res = mysqli_query($conn, $query);

 if ($res) {
 echo "Um novo registo inserido com sucesso";
 } else {
 echo "Erro: insert failed" . $query . "
" . mysqli_error($conn);
 }

 // Termina a ligação com a base de dados
 mysqli_close($conn);

?>
```

# Em exemplo: PHP

---

```
<?php

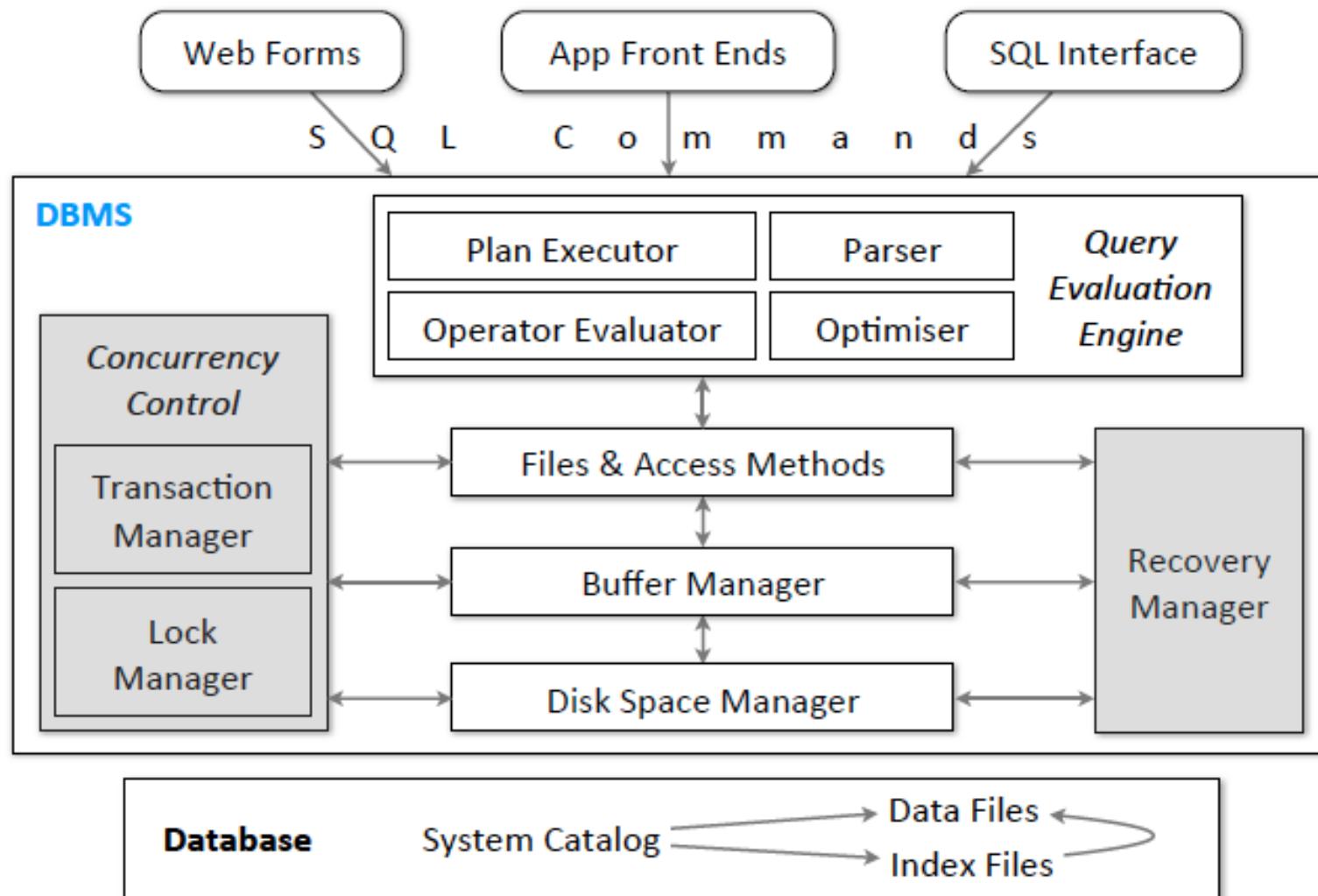
$dbhost = "appserver-01.alunos.di.fc.ul.pt";
$dbuser = "ibd000";
$dbpass = "XXXXXXX";
$dbname = "ibd000";

$conn = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);

if (mysqli_connect_error()) {
 die("Database connection failed:".mysqli_connect_error());
}

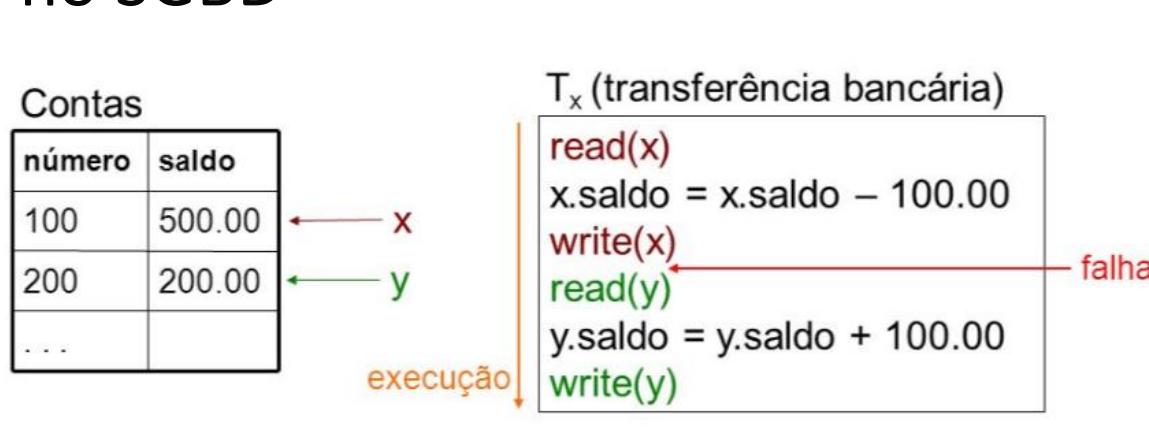
?>
```

# Componentes de um SGBD



# Gestão de Transacções

- Exemplos:
  - Transferência bancária de dinheiro com falha do sistema
  - Compra de produtos com acesso concorrente
- Uma **transação** é uma qualquer execução de uma tarefa unitária no SGBD



# Gestão de Transações - Motivação

---

Produto

| pid | pnome | preço | stock |
|-----|-------|-------|-------|
| 1   | P1    | 20    | 25    |
| 2   | P2    | 12    | 10    |
| 3   | P3    | 15    | 10    |
| 4   | P4    | 20    | 10    |

## Cenário

9:00:00 - uma aplicação cliente (C1) efetua uma pesquisa sobre a quantidade de produtos “P1” em stock

```
SELECT stock FROM Produto
WHERE pnome='P1'
```

9:00:01 - outra aplicação (C2) efetua uma pesquisa semelhante

O SGBD responde a ambos “25”

C1 reserva 20 unidades, pelo que efetua uma dedução da respetiva quantidade à BD

C2 efetua uma operação semelhante

# Transação - Definição

---

- Uma **transação** num SGBD é uma abstração de um procedimento
  - Uma sequência de operações de leitura e/ou escrita SQL ...
  - ..... executada de forma atómica pelo SGBD

Início da transação

    Instrução 1

    Instrução 2

    ...

    Instrução N

Fim da transação --   OK (**COMMIT**) ou Erro (**ROLLBACK**)

- O mecanismo de transações é essencial
  - sempre que a BD servir vários utilizadores simultaneamente
  - recuperação de falhas

# Concorrência e Consistência

---

- Requisitos de um sistema transacional
  - Gestão de múltiplas **transações em simultâneo**
  - Gestão das regras de **integridade**
- Concorrência
  - Múltiplos utilizadores e respetivos pedidos em simultâneo
- Consistência (Integridade)
  - A BD está num estado consistente quando cumpre ...
  - ... as regras de integridade
- **Um SGBD é um sistema transacional**

# Execução Concorrente de Transacções

---

- Um **protocolo de locking** é um conjunto de regras que devem ser seguidas por cada transacção (garantido pelo SGBD)
  - Apesar das operações das transacções serem executadas intercaladamente
  - O resultado é o mesmo de executar as transacções numa determinada ordem
- Dois tipos de *locks*:
  - **Partilhados** por múltiplas transacções (*leitura*)
  - **Exclusivos** a uma transacção (*escrita e leitura*)

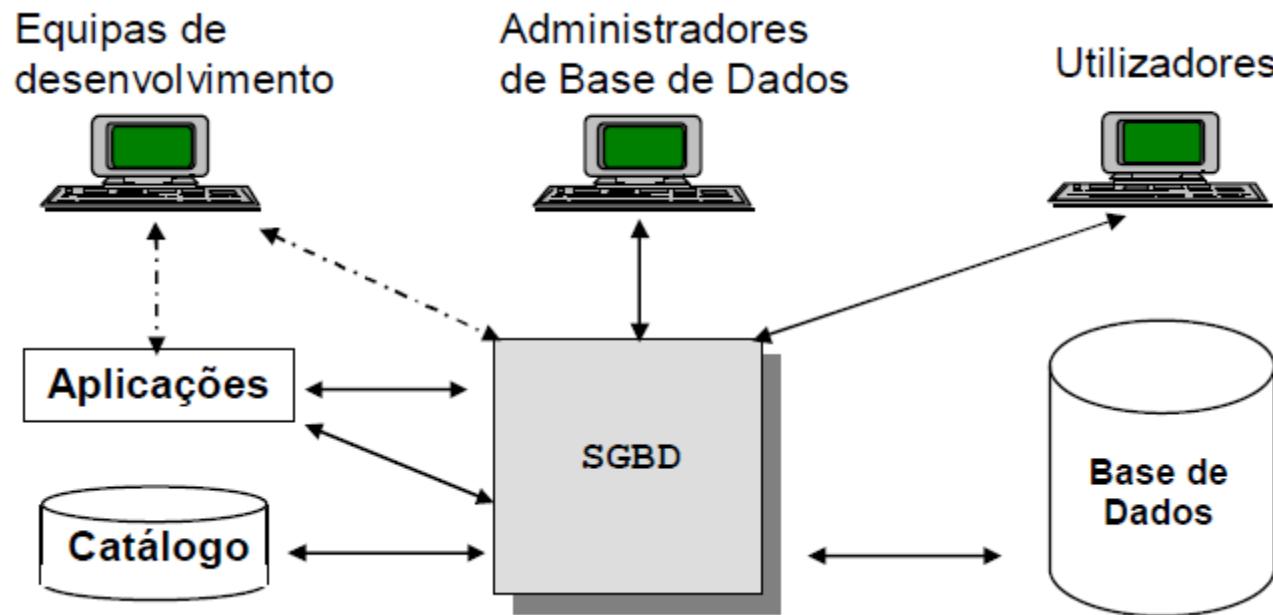
# Transacções Incompletas

---

- Alterações feitas por transacções incompletas devem ser anuladas
- O SGBD mantém um *log* de todas as operações de escrita:
  - Para **anular** transacções **incompletas**
  - Ou **refazer** transacções **completas** depois de uma falha  
Checkpoints (forçam escrita em disco) para minimizar o tempo de recuperação (dim do log)

# Desenvolvimento e Uso de SGBDs

---



**Catálogo:** contém a definição da BD. Ex: num SGBD relacional, contém definições das tabelas, vistas, regras de integridade, etc.

# Intervenientes

---

- Empresas que desenvolvem os SGBDs (*DB implementors*)
- Programadores (*DB application programmers*)
  - Criam pacotes de *software* que permitem aos utilizadores aceder aos dados
- Utilizadores de aplicações (*end users*)
  - Indústrias, serviços, fora ou dentro da Web
  - Fazem entrar dados na organização e analisam dados
- Administradores de bases de dados (*DB Administrator - DBAs*)
  - Manipulam esquemas físico e lógico (ex. para afinação)
  - Criam utilizadores e definem privilégios de acesso aos dados
  - Garantem disponibilidade e recuperação em caso de faltas

# Tarefas Críticas do Administrador BD

---

- Concepção do modelo lógico e físico
  - Interage com os utilizadores para decidir que relações (tabelas) armazenar e como
- Fiabilidade
  - Faz *backups* periódicos
  - e mantém *logs* da actividade do sistema
- Afinação
  - Adequa o desempenho às alterações de requisitos

# Tarefas Críticas do Administrador BD

---

- **Segurança e autorização**

- Atribui diferentes permissões de acesso
- 4 tipos de acesso aos recursos (table/view): CRUD
  - Create, Read, Update, Delete
- Utilização diferenciada para diferentes *roles*
  - Um utilizador pode ter vários roles
    - p.e., app\_programador, app\_read, app\_write
  - Cada role está associado a privilégios

```
CREATE ROLE 'app_programador', 'app_read', 'app_write'
GRANT ALL ON appdb.* TO 'app_programador';
GRANT SELECT ON appdb.* TO 'app_read';
GRANT INSERT, UPDATE, DELETE ON appdb.* TO 'app_write';
```