

 **deti** departamento de  
electrónica, telecomunicações  
e informática

# Introdução aos Sistemas de Base de Dados

Base de Dados - 2020/21  
Carlos Costa

*(Adapted from several DB courses and Books - see bibliography)*

1



## Base de Dados - Conceito

- **Base de Dados (BD):** uma coleção organizada de dados que estão relacionados e que podem ser partilhados por múltiplas aplicações.

### Evolução

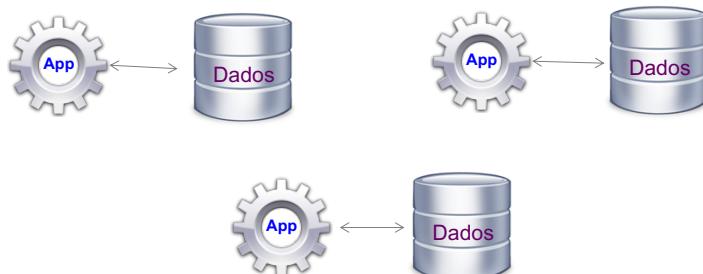


A timeline diagram illustrating the evolution of data processing over time. The horizontal axis is labeled 'tempo' (time) and has numerical markers at 50, 60, 70, and 80. Three blue rectangular boxes represent different stages: 'Processamento Aplicacional de Dados' (around 55), 'Sistema Partilhado de Ficheiros' (around 68), and 'Base de Dados' (around 82).

2

## Processamento Isolado de Dados

- **Dados isolados** - cada aplicação gera os seus próprios dados.
- Os mesmos **dados** podem estar **replicados**.
- Diferentes **organizações** e **formatos** de dados.
- Problemas de “**sincronismo**” -> **incoerências**.

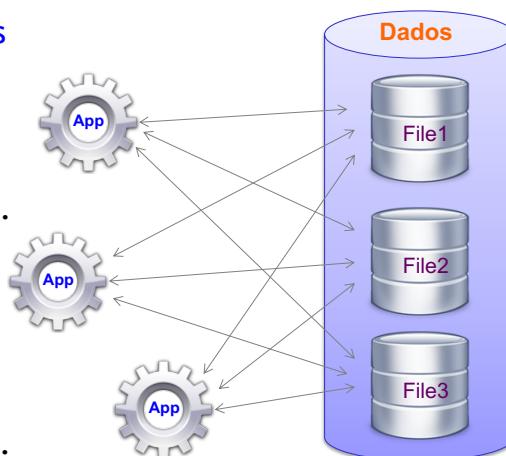


3

3

## Sistema de Gestão de Ficheiros

- Dados organizados e armazenados em **ficheiros partilhados** por várias aplicações.
- Cada aplicação acede diretamente aos ficheiros.
- Cada aplicação usa uma **interface proprietária**.
- Problemas de **acesso concorrente** aos dados.
- Problemas de **integridade**.
- Problemas de **segurança**.



4

4

**Sistema de Gestão de Base de Dados (SGBD)**

Database Management System (DBMS): “is a general-purpose software system that facilitates the processes of **defining**, **constructing**, **manipulating**, and **sharing** databases among various users and applications.”

5

5

**Sistema de Gestão de Base de Dados (SGBD)**

Base de Dados...

- Definição (*Defining*)
  - Especificação do tipo de dados, estruturas de dados e restrições
    - database catalog or dictionary
- Construção (*Constructing*)
  - Processo de armazenamento de dados
- Manipulação (*Manipulating*)
  - Envolve operações como a pesquisa e obtenção de dados
- Partilha (*Sharing*)
  - Acesso simultâneo aos dados por parte de vários utilizadores e programas

6

6

## SGBD - Características Gerais

- Entidade única que opera com a BD
  - O acesso à BD é sempre mediado pelo SGDB
- Existe uma interface de acesso que esconde os detalhes de armazenamento físico dos dados
- Elevada abstracção ao nível aplicacional
- Os dados estão integrados (nível lógico) numa mesma unidade de armazenamento
- Suporta uma ou mais BD
  
- Keyword - Data Independence

7

7

## SGBD - Vantagens

- Independência entre programas e dados
- Integridade dos dados
  - Controlo de alteração de dados de acordo com as regras de integridade definidas
- Consistência dos dados
  - Nos processos de transações e mesmo em falhas de software/hardware
- Eficiência no acesso aos dados
  - Especialmente em cenários de manipulação de grandes quantidades de dados, por um ou mais utilizadores
- Isolamento utilizadores
  - Cada utilizador tem a “sensação” de ser o único

8

8

## SGBD - Vantagens (cont.)

- Melhor gestão do acesso concorrencial
- Serviços de Segurança
  - Controlo de Acessos / Permissões
  - Codificação de Dados
- Mecanismos de backup e recuperação de dados
- Administração de dados
  - Disponibilidade de ferramentas desenvolvidas pelo fabricante e/ou terceiras entidades
- Linguagem de desenho e manipulação de dados

**Nota:** Muitas das vantagens anteriores são também requisitos funcionais de um SGBD.

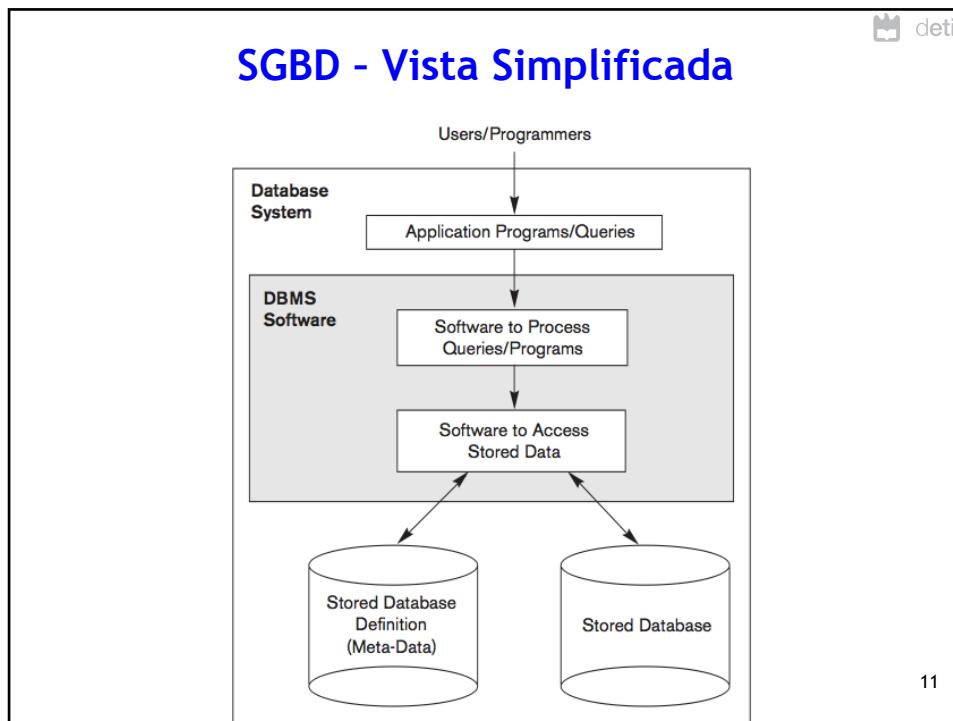
9

## SGBD - Desvantagens

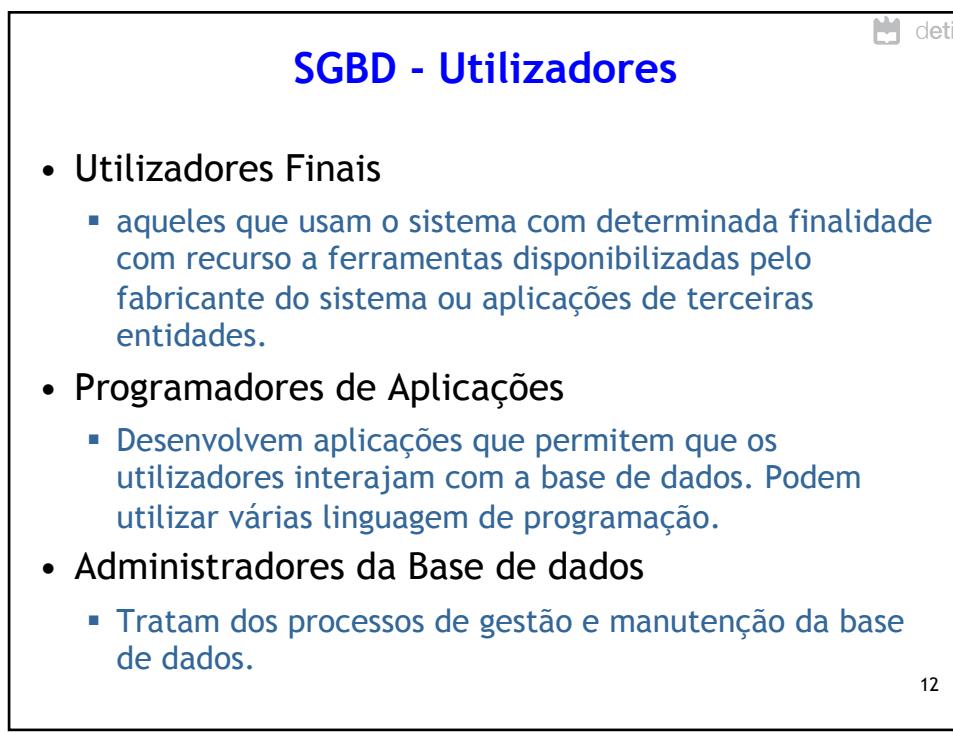
- Maiores custos e complexidade na instalação e manutenção
  - Especial em soluções empresariais
- Não respondem aos requisitos de alguns cenários aplicacionais como, por exemplo, pesquisa de texto
- Centralização dos dados mais suscetível a problemas de tolerância a falhas (software e hardware) e de escalabilidade

10

10



11



12

## SGBD - Dicionário de Dados

- O SGBD contém BD mas também informação relativa à descrição (definição) da própria estrutura da base de dados, incluindo as restrições
  - Metadados (dados sobre dados)
- Um dicionário contém:
  - Descritores de objetos da base de dados (tabelas, utilizadores, regras, vistas, indexes, etc)
  - Informação sobre dados em uso e por quem (locks).
  - *Schemas e mappings*

13

13

## Interfaces (Aplicações)

- Web-based
- Form-based (desktop)
- GUI (Graphical User Interface)
  - Manipulação visual de esquemas de BD com recurso a diagramas. Possibilidade de construção e execução de queries.
- Natural Query Language
- DBMS Command Line
  - Criar contas de utilizadores, parametrizar o sistema, definir permissões e privilégios, definir/alterar estruturas de dados, definir tipos de dados, etc.
  - Utilizando uma linguagem própria - SQL

14

14

**SGBD - Arquitetura ANSI/SPARC<sup>1</sup>**

Three-level architecture:

- External level  
database users
- Conceptual level  
database designers and administrators
- Internal level  
systems designers

External level

User 1    User 2    ...    User n

Conceptual level

Conceptual schema

Internal level

Internal schema

Physical data organization

Database

15

1. ANSI/X3/SPARC Study Group on Data Base Management Systems: (1975), Interim Report. FDT, ACM SIGMOD bulletin. Volume 7, No. 2

15

**ANSI/SPARC - Nível Interno**

- Lida com a implementação física da BD
  - Estrutura dos registos em disco - files, pages, blocks
  - Indexes e ordenação dos registos
- Domínio: Programadores de sistemas de BD
- Exemplo de Esquema
 

```
RECORD FUNCIONARIO
  LENGTH=44
  HEADER: BYTE(5)
  OFFSET=0
  NOME: BYTE(25)
  OFFSET=5
  SALARIO: FULLWORD
  OFFSET=30
  DEPARTAMENTO: BYTE(10)
  OFFSET=34
```

16

16

## ANSI/SPARC - Nível Conceptual

- Esquema Conceptual - descreve a estrutura da base de dados para os utilizadores
  - Descreve entidades, tipo de dados, relações, operações, restrições, etc
  - Utiliza (tipicamente) um modelo de dados para descrição do esquema conceptual
- Oculta detalhes de implementação física(abstração)
- Domínio: Administrador BD e prog. de aplicações
- Exemplo de esquema

```
CREATE TABLE FUNCIONARIO
  (Nome VARCHAR(25),
   Salario REAL, Dept_Nome VARCHAR(10))
```

17

17

## ANSI/SPARC - Nível Externo

- Oferece vistas da base de dados adaptadas a casa utilizador
  - Apresentação dos dados pode ser trabalhada, parte dos dados pode ser ocultada, etc.
- Domínio: Utilizadores finais e prog. de aplicações
- Exemplo de Esquema

```
FolhaPagamentos:
  char *Nome
  double Salario

Funcionarios:
  char *Nome
  char *Departamento
```

18

18

## ANSI/SPARC - Independência dos dados

- A alteração do esquema (*schema*) de um nível não tem impacto no esquema do nível acima.

=> [dois níveis de independência](#)

- Nível Físico

- Alterações do nível físico não devem ter impacto no esquema conceptual.
- Por exemplo, podemos alterar a forma como armazenamos os dados no sistema de ficheiros por razões de desempenho.

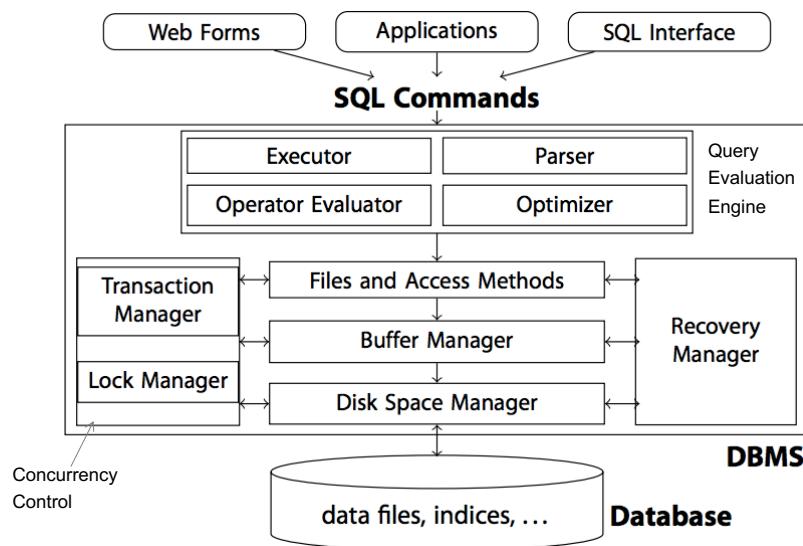
- Nível Lógico

- Alterações no esquema conceptual (modelo de dados) não devem repercutir-se nos esquemas externos ou aplicações já desenvolvidas.

19

19

## SGBD - Arquitetura Típica



20

R. Ramakrishnan and J. Gehrke, Database Management Systems. McGraw-Hill.

20

## Modelo de Base de Dados

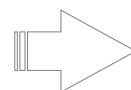
- Modelo de BD - coleção de conceitos para descrição lógica de dados (Modelo Lógico)
- Esquema (Schema): a descrição de um **conjunto particular de dados** com **recurso** a um determinado **modelo**
- Um bom **modelo** de dados é **fundamental** para garantir a **independência dos dados**
- O **Modelo Relacional** é um dos mais utilizados nos dias de hoje.
  - Bancos, Hospitais, Finanças, Seguradoras, etc

21

21

## Modelos de Base de Dados

- 1<sup>a</sup> Geração (Pré-relacional)
  - Hierárquico
  - Rede
- 2<sup>a</sup> Geração
  - **Relacional**
- 3<sup>a</sup> Geração (Pós-relacional)
  - Object-relational
  - Object-oriented
  - Key-value store
  - Document-oriented
  - Column-oriented
  - Graph database



Disciplina  
de  
Base de Dados

22

22

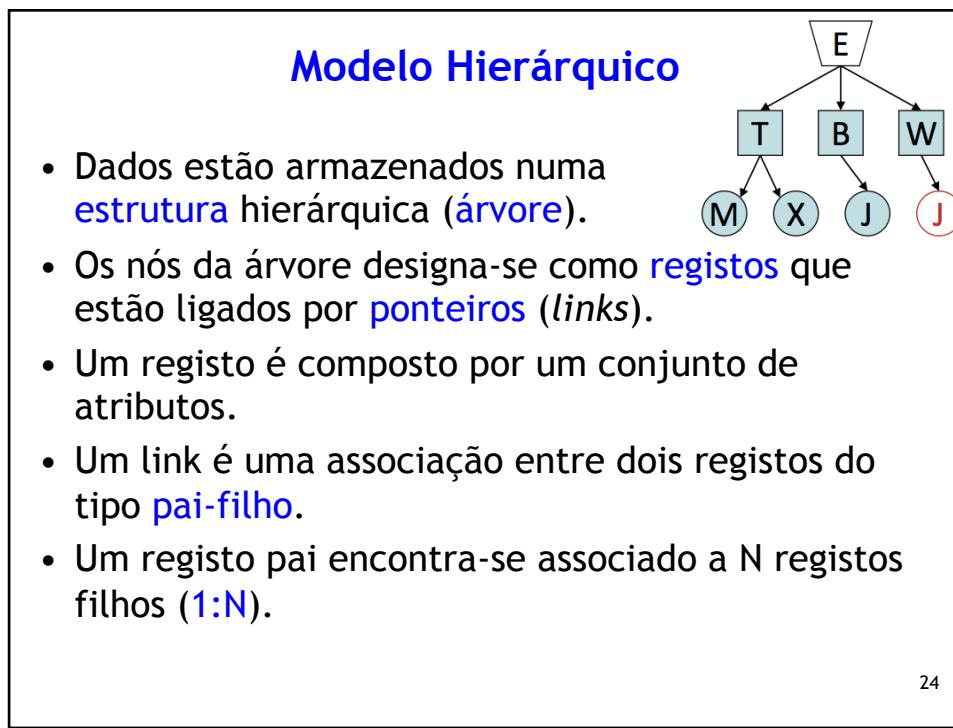
deti

## Modelos NoSQL

Phases	Targets		
Conceptual Design	Conceptual Data Model	ERD(Entity Relationship Diagram) UML(Unified Modeling Language) ORM(Object Role Modeling) FCO-IM(Fully Communication Oriented Information Modeling)	
Logical Design	NoSQL Data Model	Key-Value , Document, Column Family , Graph	
Physical Design	NoSQL Database	Key-Value	Riak, Redis, Memcached ,Berkeley DB ,Hamster DB, Amazon Dynamo DB ,Project Voldemort
		Document	MongoDB, Couch DB, Terrastore, Orient DB, Raven DB
		Column Family	Cassandra, HBase, Hypertable, Amazon Simple DB
		Graph	Neo4J, Infinite Graph, Orient DB, Flock DB

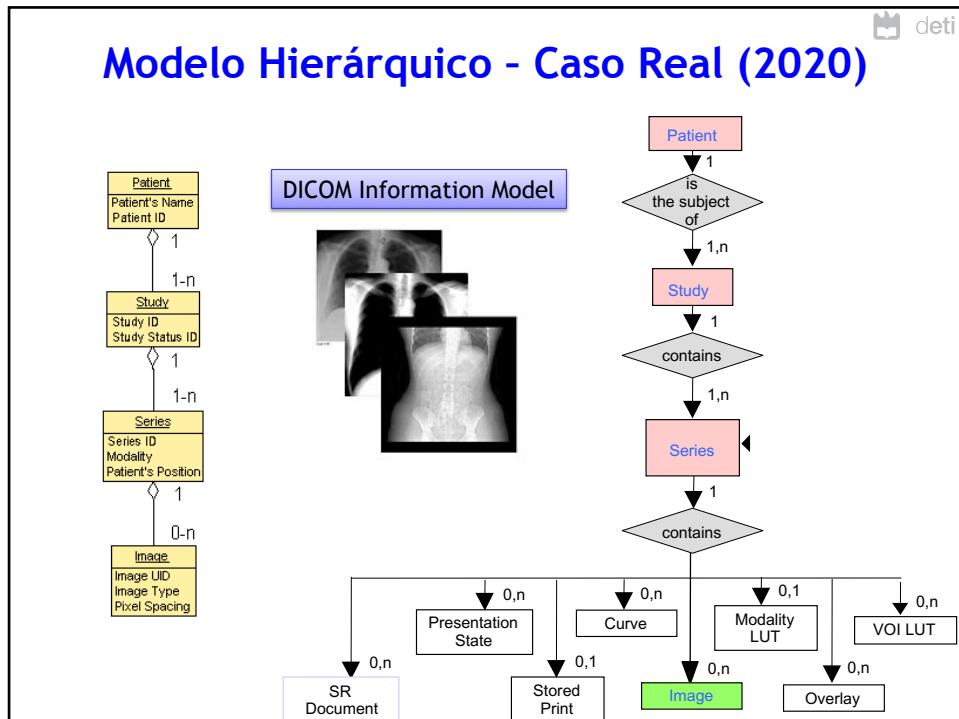
23

23

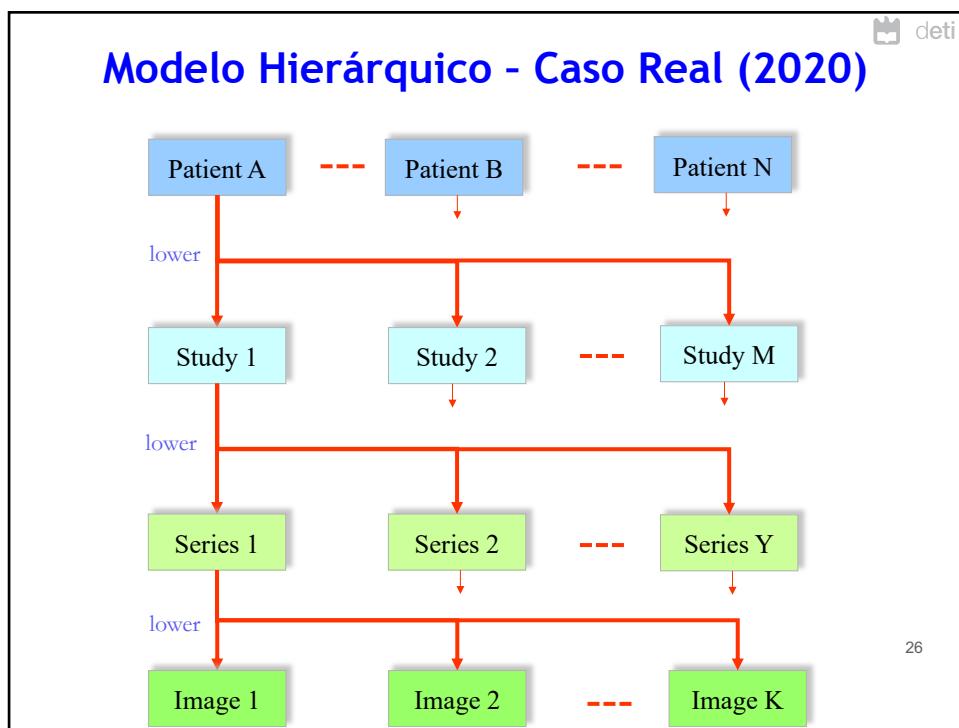


24

24



25



26

## Modelo Hierárquico - (Des)vantagens

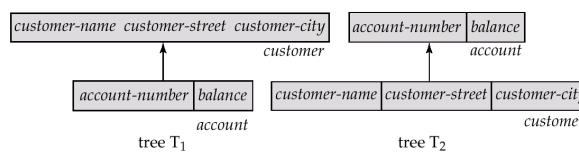
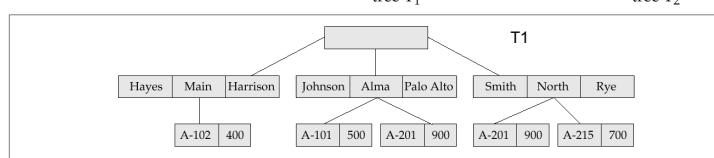
- Adaptado a cenários de acesso sequencial aos dados.
  - Qualquer acesso aos dados passa sempre pelo segmento raiz.
  - A maior parte das necessidades atuais requer acesso aleatório!
- Redundância de informação
  - Desperdício de espaço e inconsistências de dados
- Restrições de integridade, exemplo:
  - A eliminação de um segmento pai, implica a remoção de todos os segmentos filhos associados.
- Não permite estabelecer associações N:M

27

27

## Modelo Hierárquico - Relação N:M

Uma conta pode ter vários titulares (clientes).  
Um cliente pode ter várias contas.

tree T<sub>1</sub> tree T<sub>2</sub>

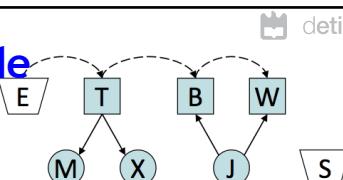
Temos necessidade  
de duas estruturas  
em árvore

Duplicação  
de Dados

28

28

**Modelo de Rede**

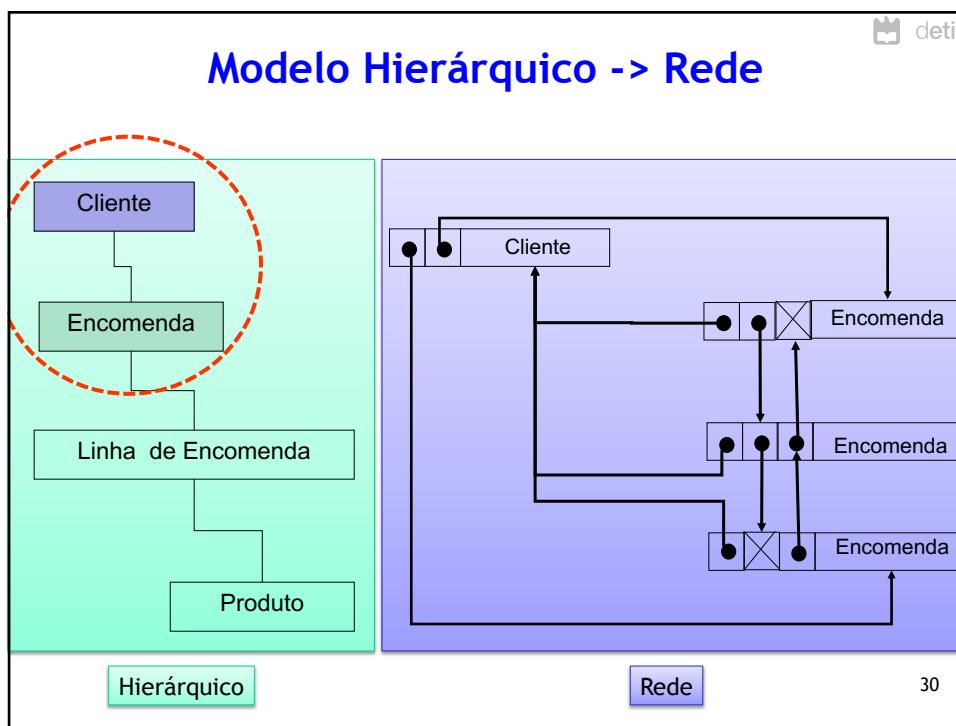


- Extensão do modelo hierárquico.
- Permite que um mesmo registo esteja envolvido em várias associações -> visão de rede.
- Melhorias na capacidade de navegação na estrutura de dados.
- Relações representadas através de grafos.
- Um conjunto (*set*) suporta associação entre registos do mesmo tipo
  - Tipicamente implementados com listas ligadas circulares
  - Relacionamento 1:N entre dois tipos de registo.

29

29

**Modelo Hierárquico -> Rede**

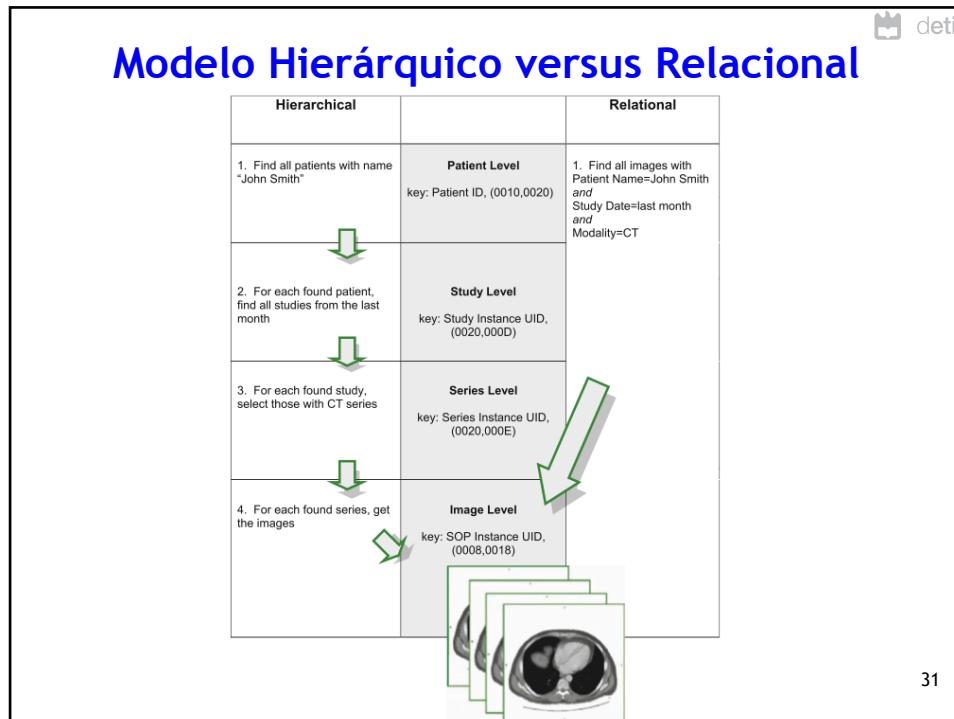


**Hierárquico**

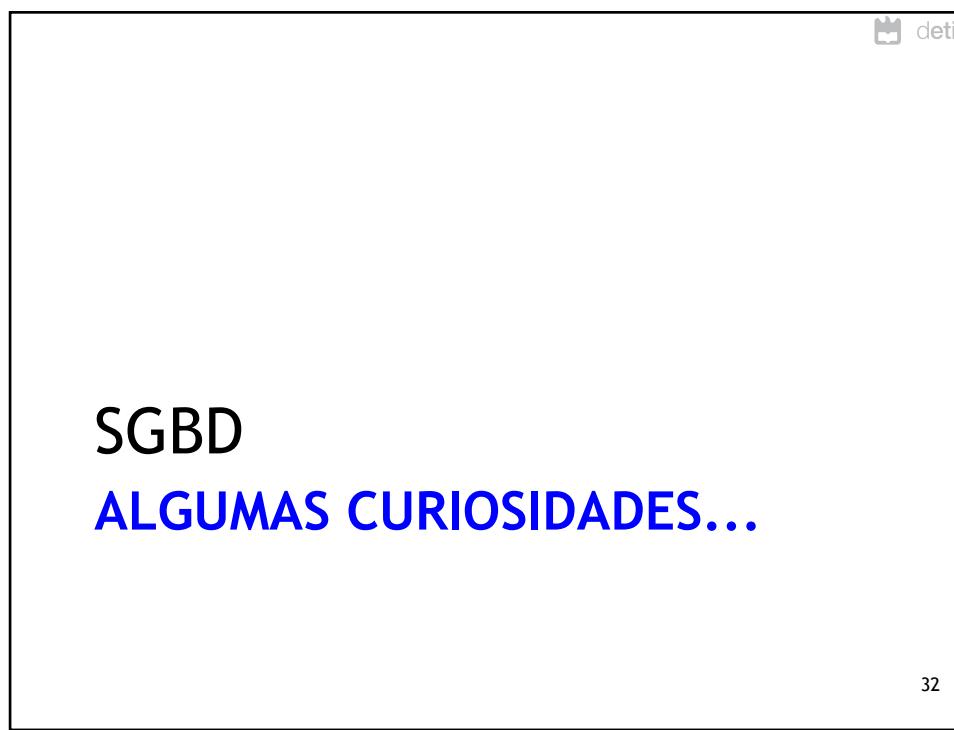
**Rede**

30

30



31



32

32

**DB-Engines Ranking - Engine**

364 systems in ranking, March 2021

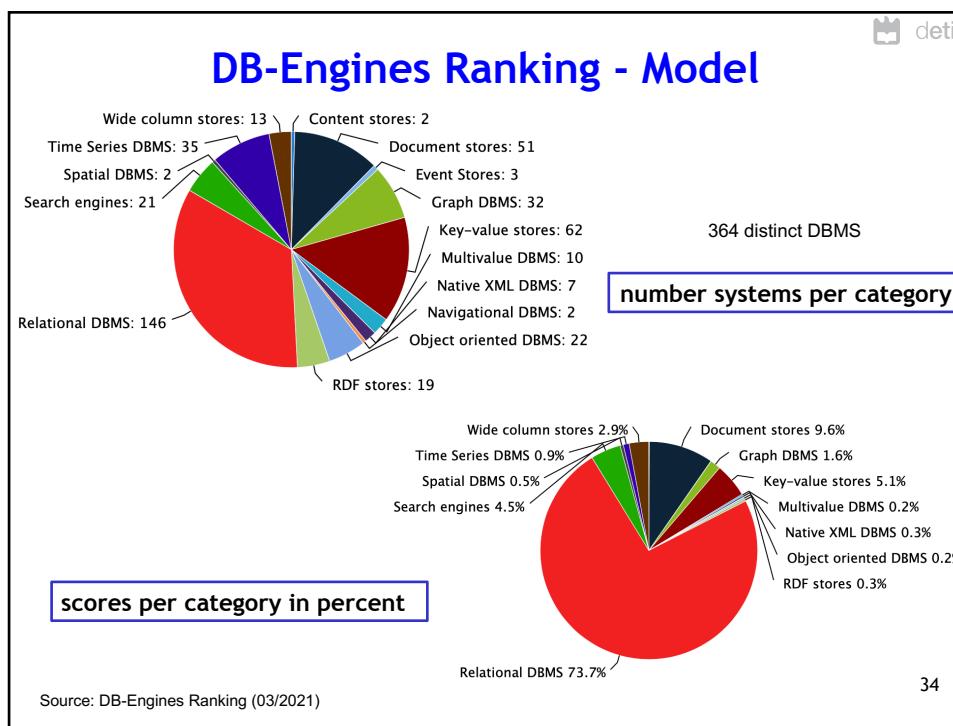
Rank	DBMS	Database Model	Score		
			Mar 2021	Feb 2021	Mar 2020
1.	1. Oracle +	Relational, Multi-model	1321.73	+5.06	-18.91
2.	2. MySQL +	Relational, Multi-model	1254.83	+11.46	-4.90
3.	3. Microsoft SQL Server +	Relational, Multi-model	1015.30	-7.63	-82.55
4.	4. PostgreSQL +	Relational, Multi-model	549.29	-1.67	+35.37
5.	5. MongoDB +	Document, Multi-model	462.39	+3.44	+24.78
6.	6. IBM Db2 +	Relational, Multi-model	156.01	-1.60	-6.55
7.	7. Redis +	Key-value, Multi-model	154.15	+1.58	+6.57
8.	8. Elasticsearch +	Search engine, Multi-model	152.34	+1.34	+3.17
9.	9. SQLite +	Relational	122.64	-0.53	+0.69
10.	11. Microsoft Access	Relational	118.14	+3.97	-7.00

Ranks database management systems according to their popularity.

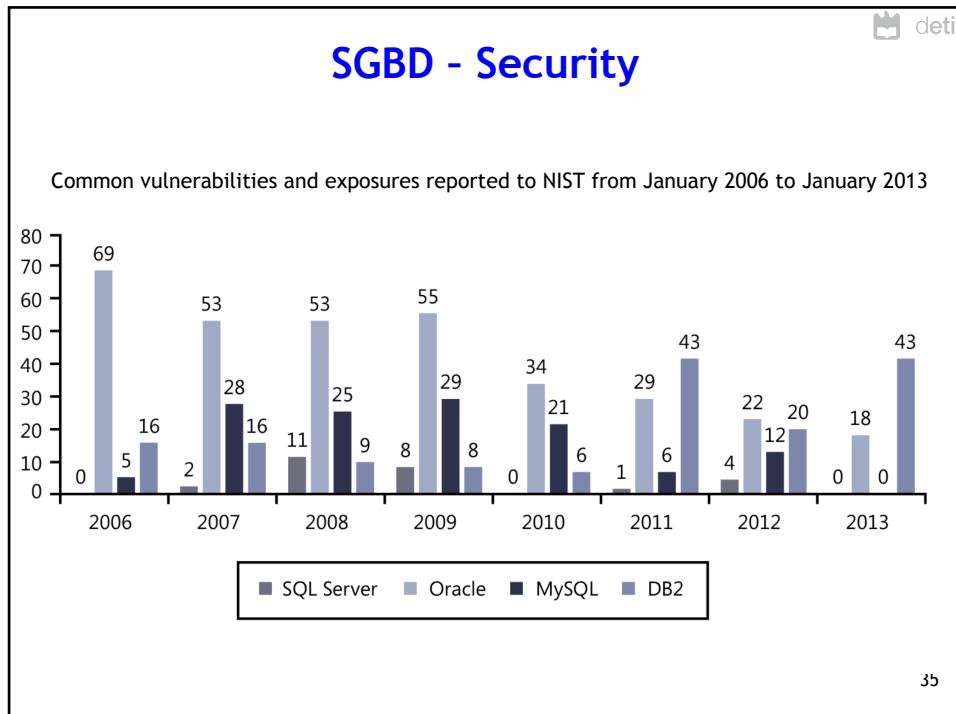
Source: DB-Engines Ranking (03/2021)

33

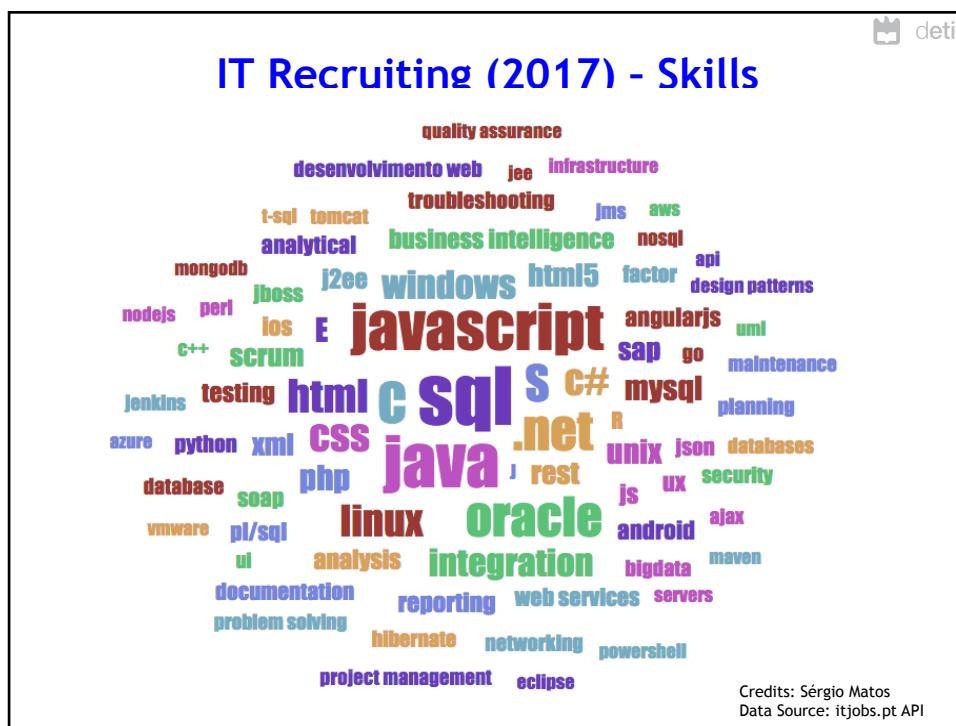
33



34



35



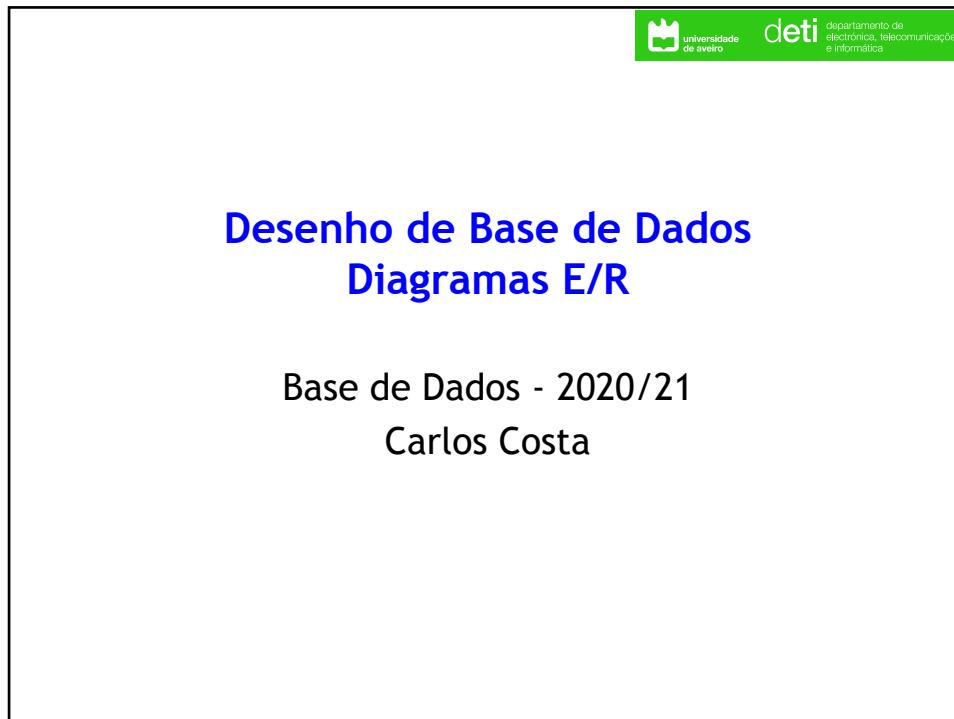
36

## Resumo

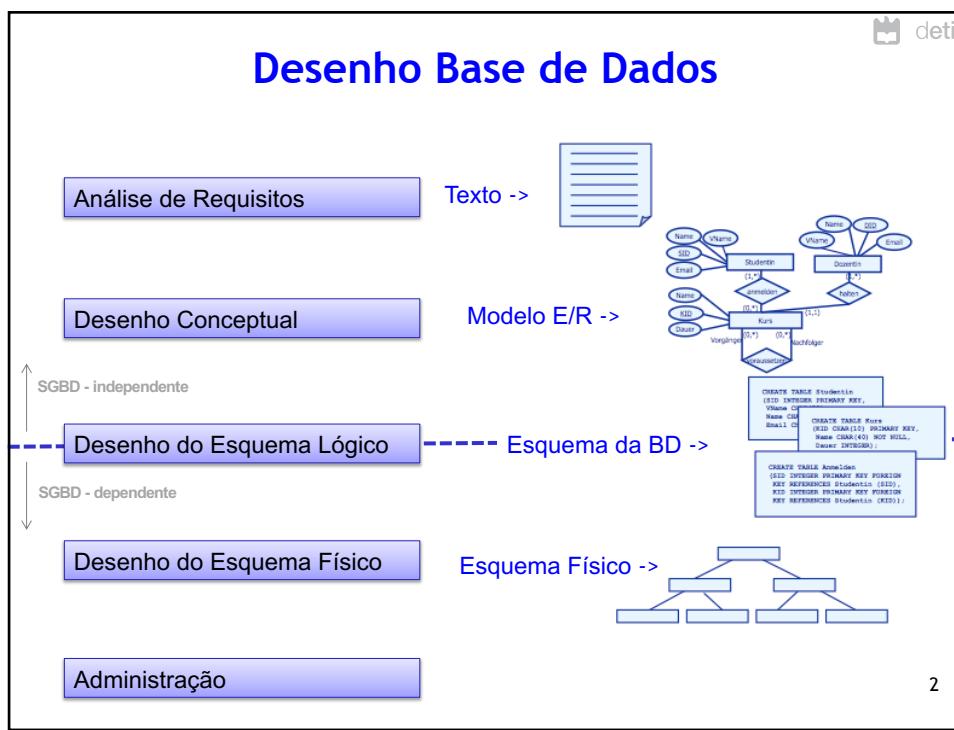
- Introdução aos Sistemas de Base de Dados
- Sistemas Gestores de Base de Dados
- Modelos de Base de Dados

37

37



1



2

## Análise de Requisitos

Obriga a um processo de **comunicação** com o **cliente** da solução de DB.

1. Levantamento detalhado de toda a informação (essencial) associada ao “problema” do mundo real: entidades, relações, restrições, etc.;
2. Filtragem da informação: remoção de redundâncias e “ruído” (informação pouco relevante);
3. Discussão para clarificar aspectos dúbios e eventuais falhas no levantamento do ponto 1;
4. Distinção entre dados e operações.

3

3

## Desenho Conceptual

- Modelo Conceptual
  - **Conceptualização** do mundo real (*structuring the problem*)
- Modelação trata do **mapeamento** das **entidades** e **relações** do mundo real para **conceitos** de base de dados.
  - não é determinístico.
  - nem sempre é claro (óbvio).
- Uma **visão abstracta** da **estrutura** de base de dados que suportará os dados reais.
- Técnica (típica):
 

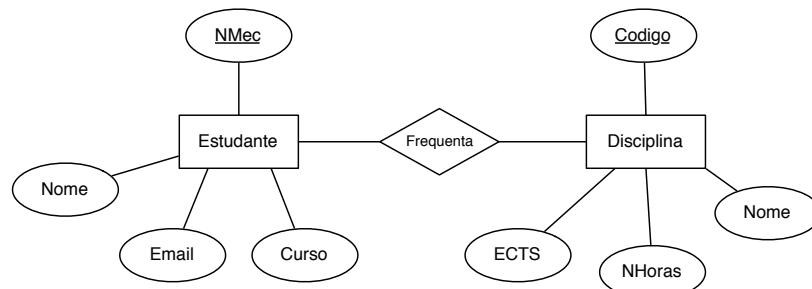
Modelo Entidade/Relacionamento

4

4

## Modelo Entidade/Relacionamento (E/R)

- alias: Modelo Entidade/Associação (E/A)
- Introduzido em 1976 por P.P. Chen  
[The Entity-Relationship Model - Toward a Unified View of Data. TODS 1\(1\): 9-36, 1976](#)
- Diagrama E/R (DER)



5

5

## Modelo E/R - Elementos Principais

- Entidades
  - algo que existe
  - ex: Pessoa, Carro, Filme
- Atributos
  - propriedades das entidades
  - ex: Pessoa tem um nome, Carro tem uma matrícula e Filme tem um título
- Relacionamento
  - relações entre duas ou mais entidades

6

6

 deti

## Diagramas E/R - Notação

- Entidade
  - Representada por um rectângulo.

Exemplos:



Entidade

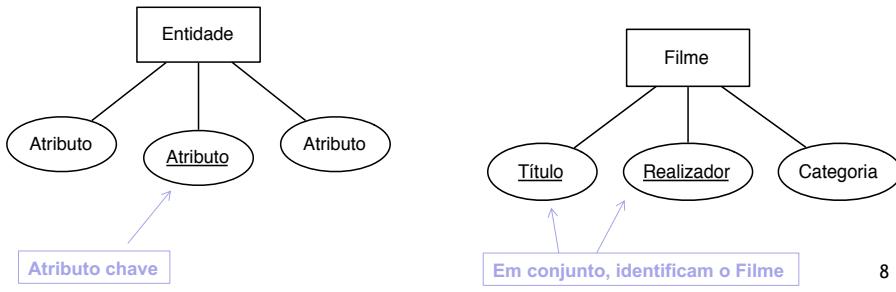
7

 deti

## Diagramas E/R - Notação

- As entidades tem um (ou mais) atributos chave que a identificam.
- O nome destes atributos aparece a sublinhado nos diagramas E/R.

Exemplo:

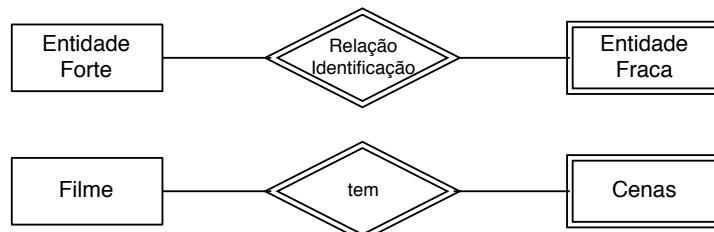


8

## DER - Entidades

- Fortes
  - Não dependem de outras entidades.
- Fracas
  - Dependem de outras entidades.

*"...do not have key attributes ... entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values... a weak entity cannot be identified without an owner entity..."*

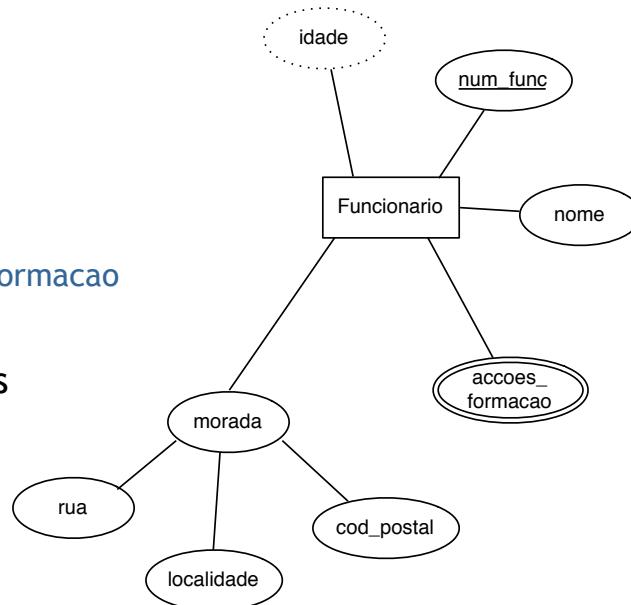


9

9

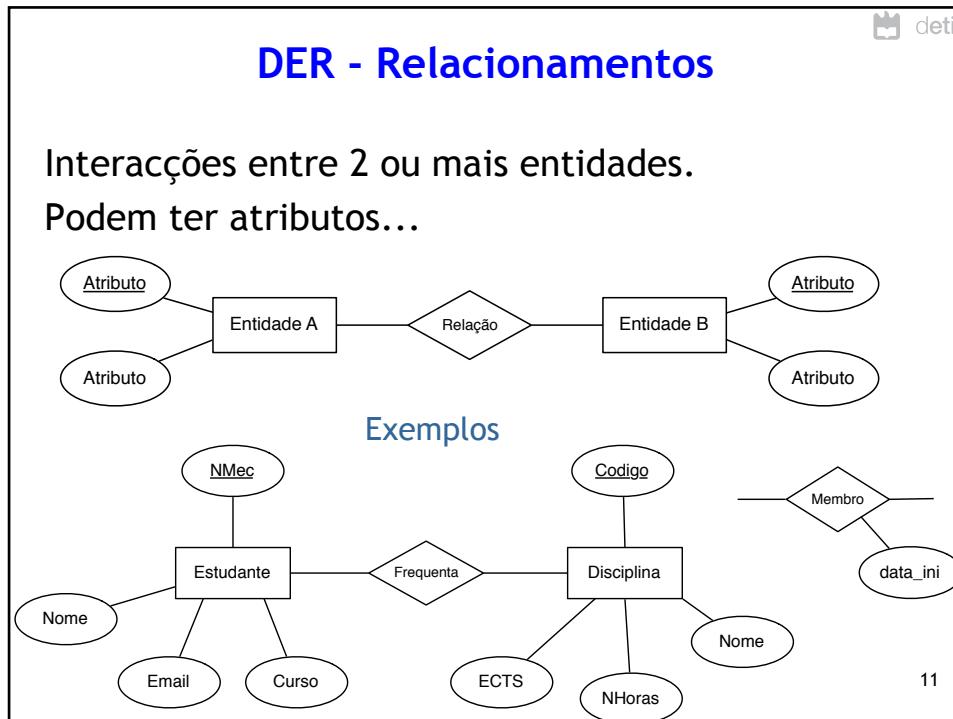
## DER - Atributos

- Derivados
  - *idade*
- Multivalor
  - *accoes\_formacao*
- Compostos
  - *morada*

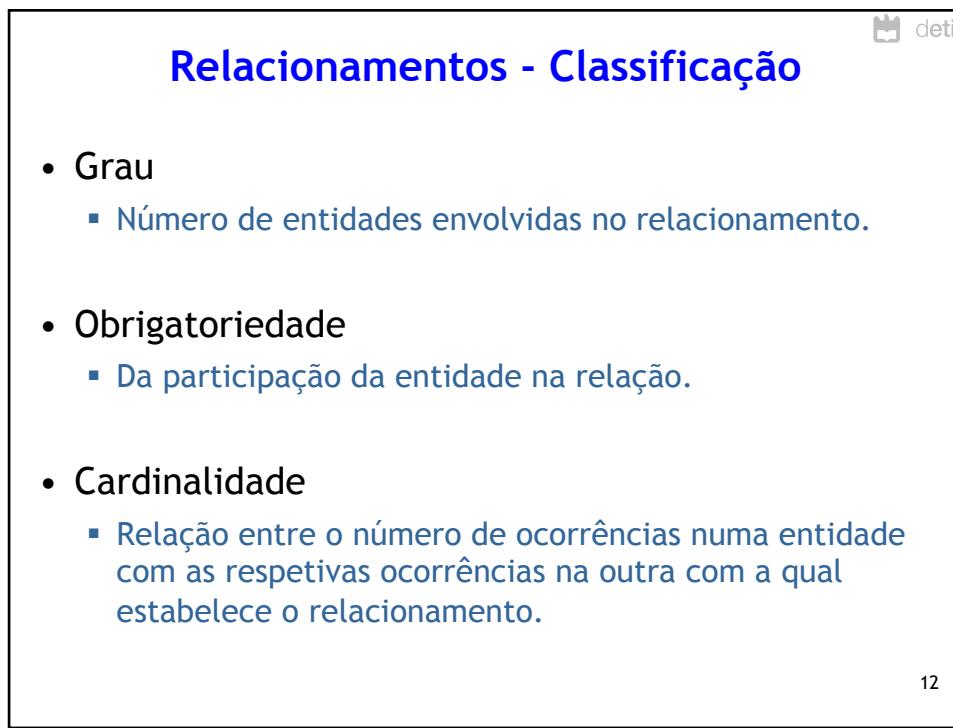


10

10



11



12

 deti

## Grau da Relação

Número de entidades participantes no relacionamento.

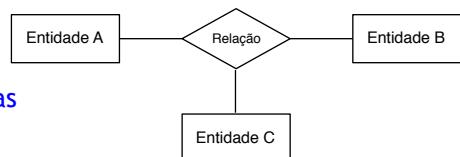
- Unária



- Binária  
(mais comuns)



- Ternária  
(podem ser convertidas em binárias)



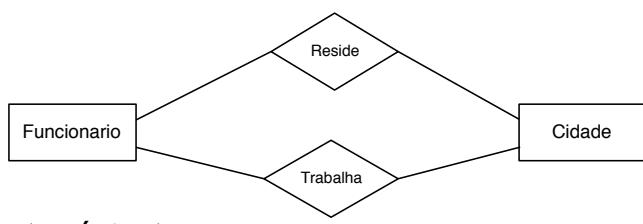
13

13

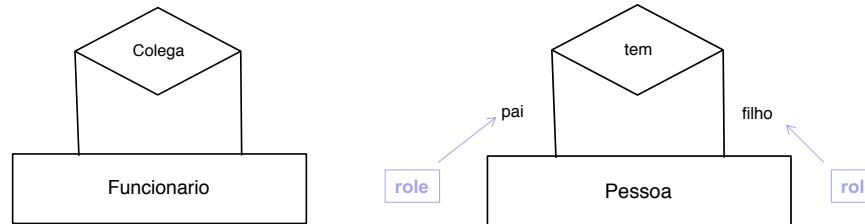
 deti

## DER - Relacionamentos

- Múltiplos



- Recursivos (unárias)
  - assimétricas - é necessário indicar os papéis (roles)



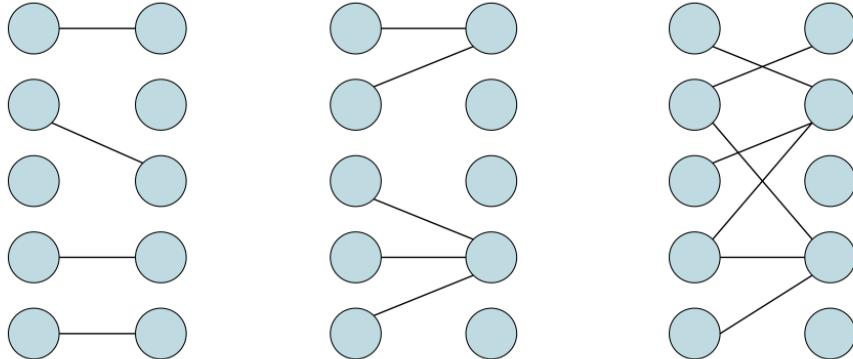
14

14

 deti

## Cardinalidade

- Relação entre o número de ocorrências numa entidade com as respectivas ocorrências na outra com que tem o relacionamento.



The diagram shows three examples of entity relationships:

- Relação 1:1 (um-para-um)**: Two entities from set A are connected to two specific entities in set B. This means each entity in A is paired with exactly one entity in B.
- Relação 1:N (um-para-muitos)**: One entity from set A is connected to multiple entities in set B. This means each entity in A can be paired with zero or more entities in B.
- Relação N:M (muitos-para-muitos)**: Entities from both sets A and B are interconnected in a many-to-many relationship. Each entity in A can be paired with zero or more entities in B, and vice versa.

15

 deti

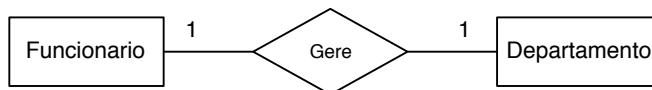
## Cardinalidade - Notação E/R

- Notação de Chen



The general Chen notation for cardinality in E/R diagrams shows two entities, Entidade A and Entidade B, connected by a diamond-shaped relationship symbol labeled "Relação". The cardinality is indicated by numbers K1 and K2 placed near the lines connecting the entities to the diamond.

- Exemplos**



Funcionario  $\xrightarrow{1}$  Gere  $\xrightarrow{1}$  Departamento

Um funcionário gera um departamento. Um departamento só tem um gestor.



Funcionario  $\xrightarrow{N}$  Trabalha Para  $\xrightarrow{1}$  Departamento

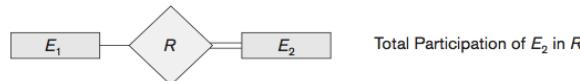
Um funcionário trabalham para um departamento. Um departamento tem vários funcionários.

16

## Obrigatoriedade de Participação na Relação

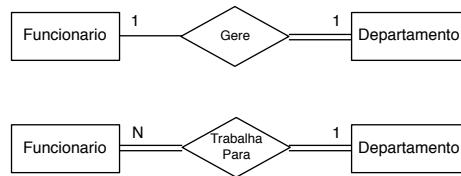
- Participação total (obrigatório)

- cada instância da entidade participa em pelo menos uma relação do conjunto de relações (linha dupla).



- Participação parcial (opcional)

- alguma(s) instância(s) da entidade podem não participar em qualquer relação do conjunto de relações.



17

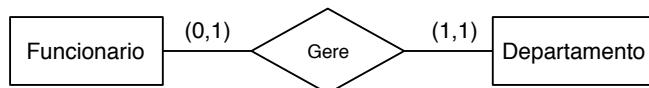
17

## Obrigatoriedade - Notação E/R (min,max)

- Existe uma notação alternativa com **(min,max)** para impor **restrições** à participação de cada entidade na relação.



- Exemplos



18

18

## Obrigatoriedade - Notação E/R (min,max)

- Mínimo

- Se “0”, é **opcional** a participação da entidade na relação.
- Se “1”, é **obrigatória** a a participação da entidade na relação.

- Máximo

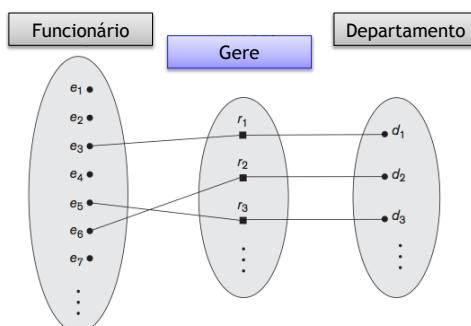
- Se “1”, cada instância da entidade está, no máximo, associada a uma única instância da relação.
- Se “N”, cada instância da entidade está associada a várias instâncias da relação.
  - Uma notação alternativa especifica o número máximo de associações, por exemplo: 4, 8, 20, etc

19

19

## Relacionamento 1:1

*Um funcionário gera um departamento e um departamento só tem um gestor (funcionário).*



20

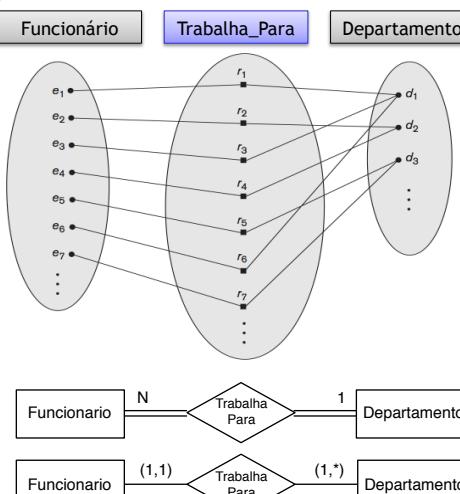
20

 deti

## Relacionamento 1:N

*Um funcionário trabalha para um só departamento. Um departamento tem um ou mais funcionários.*

Diagrama de entidades e relacionamentos:

- Entidades: Funcionário, Trabalha\_Para, Departamento.
- Relacionamento: Funcionário (N) — Trabalha\_Para —> Departamento (1).
- Diagrama de fluxo de dados:


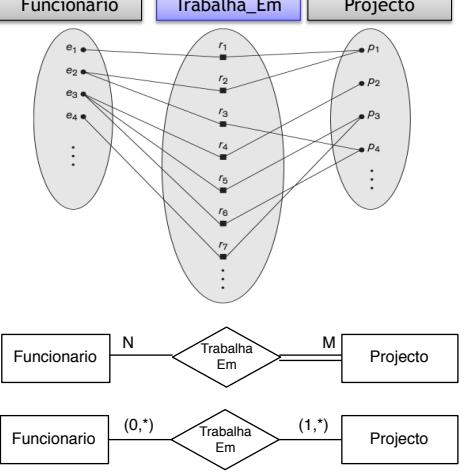
21

 deti

## Relacionamento N:M

*Um funcionário pode trabalhar em um ou mais projetos. Um projeto tem um ou mais funcionários a trabalhar nele.*

Diagrama de entidades e relacionamentos:

- Entidades: Funcionário, Trabalha\_Em, Projecto.
- Relacionamento: Funcionário (N) — Trabalha\_Em —> Projecto (M).
- Diagrama de fluxo de dados:


22

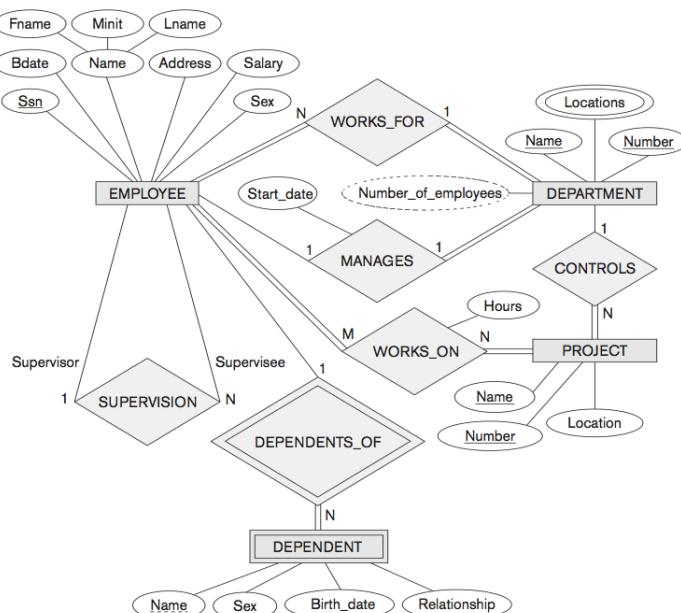
## Restrições de Integridade

- São invariantes que a base de dados deve garantir.
- Tipos de Restrições:
  - Atributos
    - Cada atributo só tem um valor
    - Atributos chave são únicos
    - Atributo (deve / pode ter) ter um valor
    - Valor do atributo pode ter restrições (>, <, !=, not null, etc)
  - Cardinalidade do Relacionamento
    - 1:1 (um-para-um)
    - 1:N (um-para-N)
    - N:M (muitos-para-muitos)
  - Obrigatoriedade de participação das entidades nas associações.

23

23

## Diagrama E/R - Exemplo



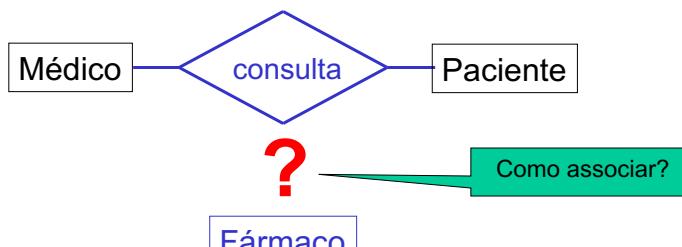
24

24

 deti

## DER - Agregação

- Às vezes temos necessidade de modelar uma **relação entre** uma **entidade e outra relação** envolvendo outras entidades.
- Exemplo: Como associar Fármacos prescritos numa Consulta médica?



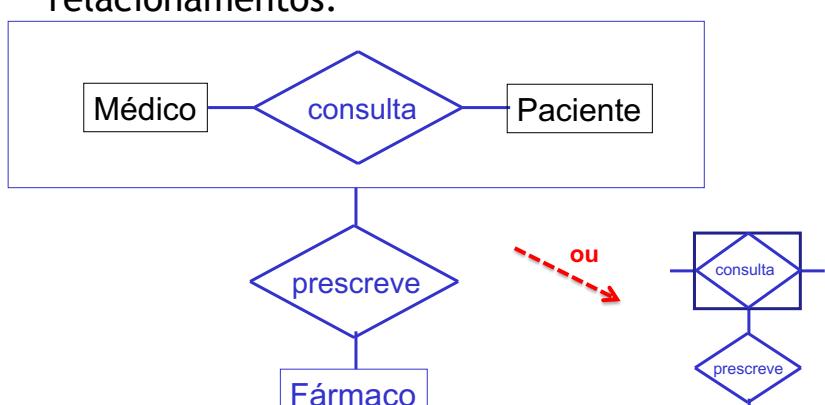
26

26

 deti

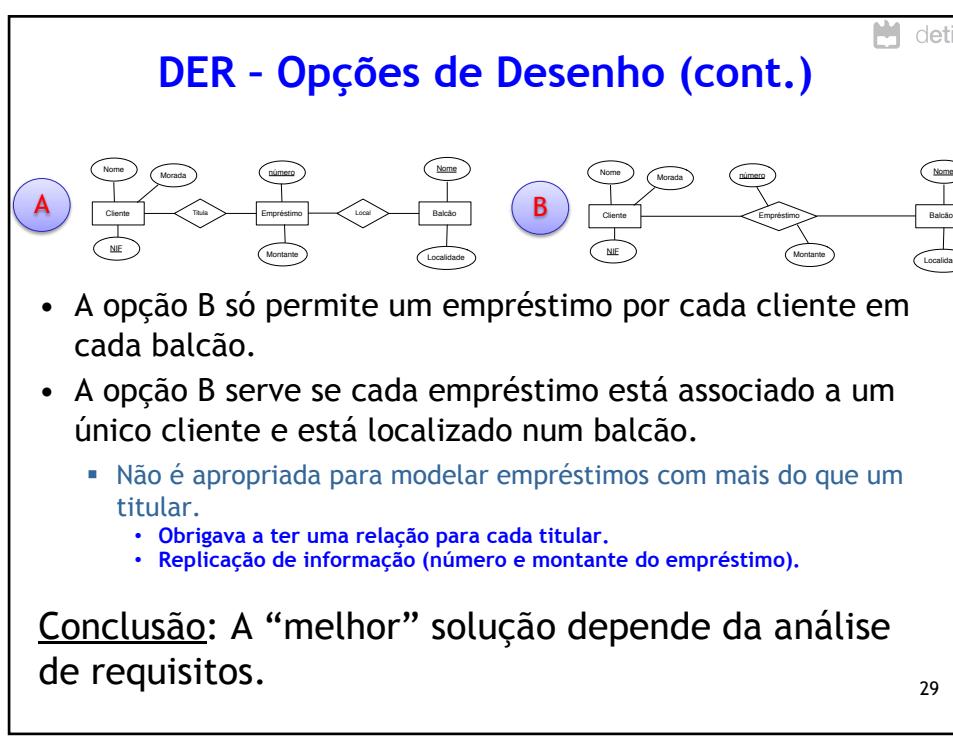
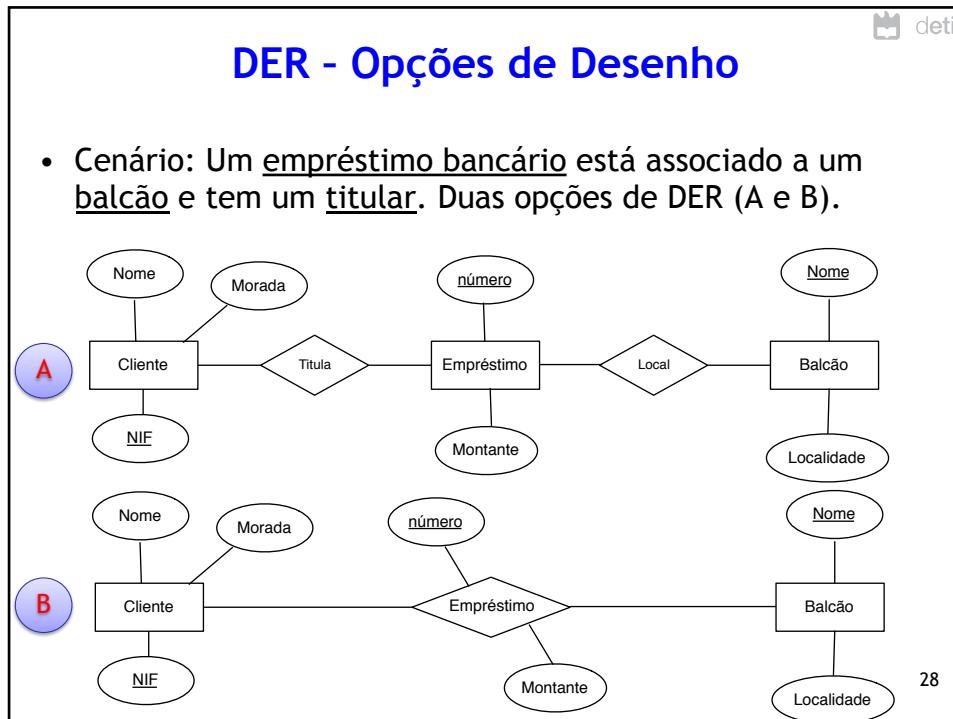
## DER - Agregação

- Solução:** Tornar uma relação numa entidade associativa.
- Entidade Associativa** - Permite associar entidades a relacionamentos.



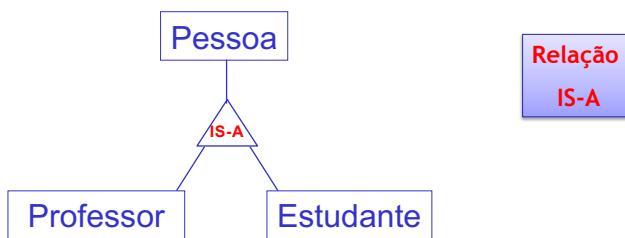
27

27



## Generalização versus Especialização

- Classificação de entidades em hierarquia de classes.  
As sub-entidades herdam os atributos das super-entidades.



### Restrições (tipo de especialização)

- Sobreposição (*overlapping*)
- Completude (*covering*)

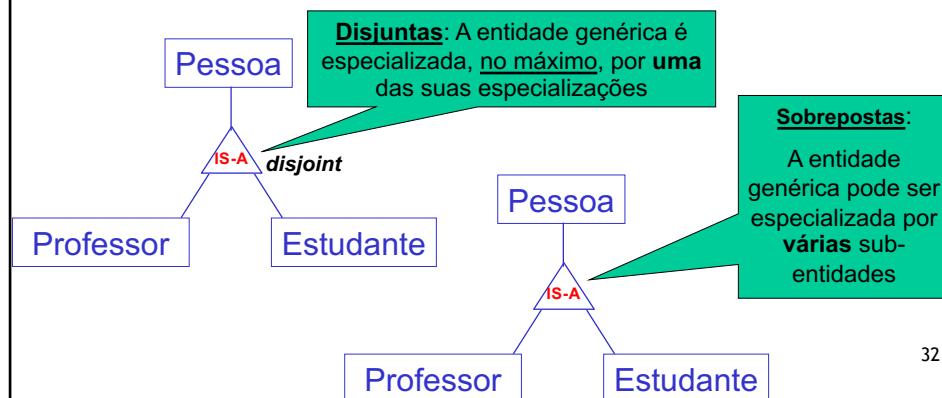
31

31

## Especialização - Tipos

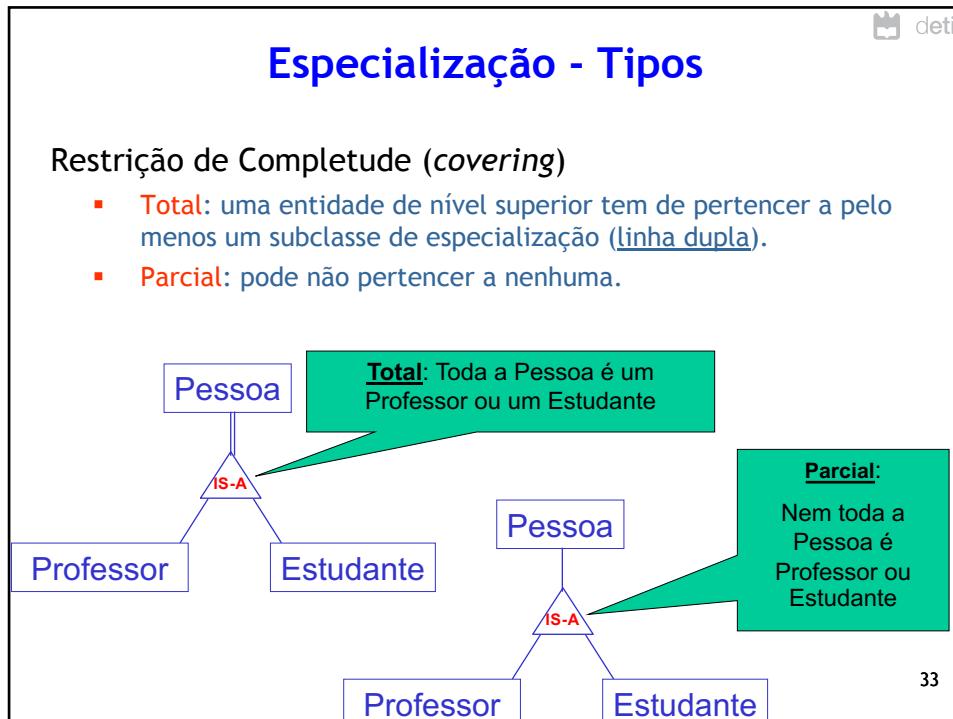
### Restrição de Sobreposição (*overlapping*)

- Disjuntas:** uma entidade só pode pertencer, no máximo, a uma subclasse de especialização (*disjoint* - ao lado do  $\Delta$ ).
- Sobrepostas:** uma ocorrência de entidade genérica pode ter mais de uma especialização.

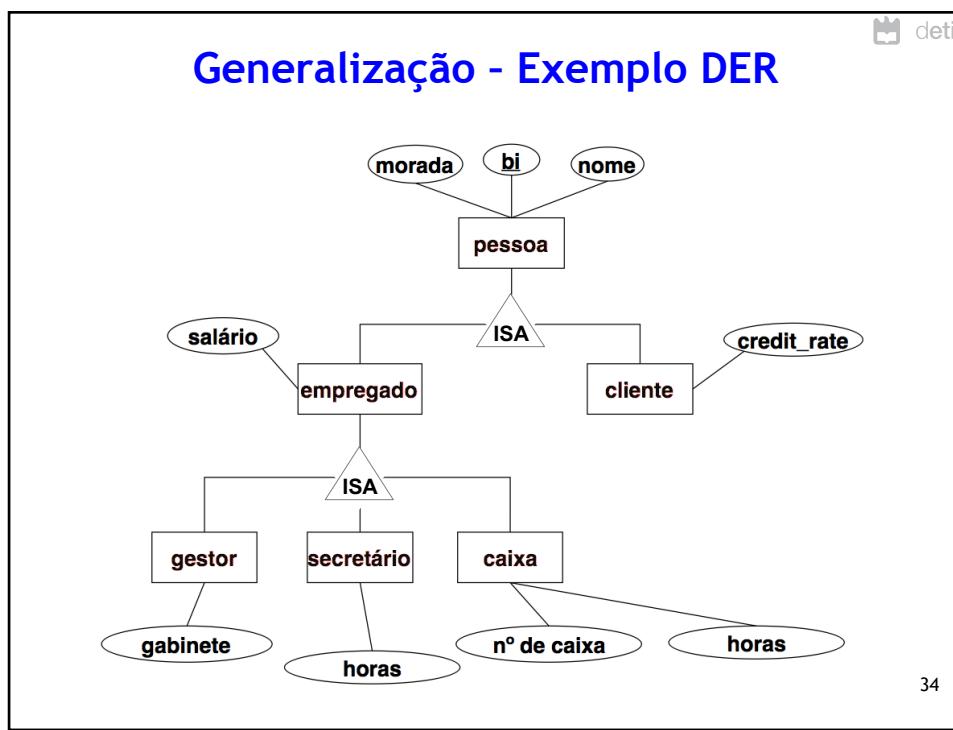


32

32



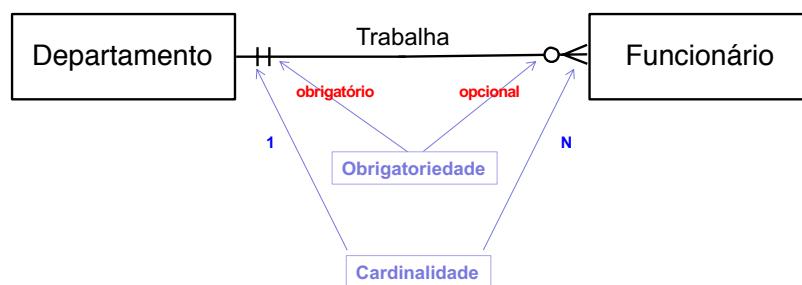
33



34

## Outras Notações DER

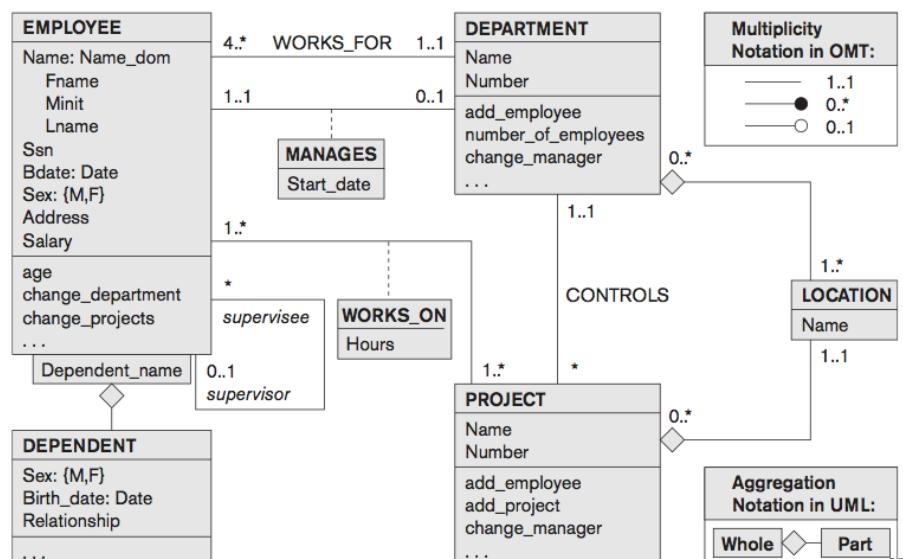
- Para além da notação utilizada por Chen, existem outras notações para Diagramas ER.
- Outra notação muito utilizada na literatura:  
**Crow's Foot (pé de galinha)**



36

36

## Outras Notações - Diagrama de Classes UML



“standard for conceptual object modeling”

37

37

## Diagramas E/R - Casos de Estudo

1 - Clínica Médica

2 - Empresa

39

39

### 1 - Clínica Médica

- Uma clínica médica pretende informatizar os seus serviços administrativos, começando por informatizar os dados referentes a médicos, pacientes e consultas.
- Cada médico é identificado internamente por um número de funcionário e a clínica pretende ainda registar o seu nome, especialidade, endereço e telefone.
- Os médicos dão consultas a pacientes que são identificados pelo seu número de utente. A clínica pretende ter sempre disponível a informação do nome, telefone e endereço dos seus pacientes.
- Uma consulta obriga à associação de um médico a um paciente num determinado dia e hora.
- As consultas são numeradas para cada um dos médicos, ou seja, para cada médico há uma consulta 1, 2, 3, etc.
- Associado a cada consulta existe um processo de prescrição de fármacos que tem de ficar registado no sistema de informação. Cada fármaco tem um nome e um código de identificação.

40

40

 deti

## 1 - Clínica Médica

- Identificação das entidades
  - médico
  - paciente
  - consulta
  - fármaco
- Identificação das relações entre entidades (cardinalidade)
  - médico dá consulta (1:N)
  - paciente marca consulta (N:1)
  - fármaco prescrito em consulta (N:M)
- Obrigatoriedade
  - uma consulta envolve sempre um médico / todos os médicos têm consultas
  - uma consulta envolve sempre um paciente / nem todos os pacientes têm consultas
  - nem todas as consultas prescrevem fármacos / nem todos os fármacos são prescritos em consultas
- Identificação dos atributos de cada entidade...

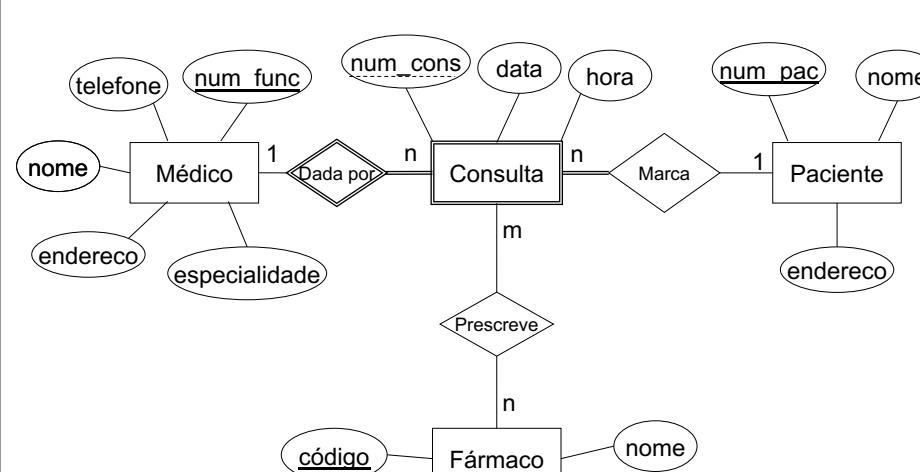
41

41

 deti

## 1 - Clínica Médica

### DER - Notação



```

    erDiagram
        {
            "Médico" ||--o "Consulta" : "Dada por"
            "Consulta" ||--o "Paciente" : "Marca"
            "Consulta" ||--o "Fármaco" : "Prescreve"
            "Médico" {
                string nome
                string telefone
                number num_func
                string endereço
                string especialidade
            }
            "Consulta" {
                number num_cons
                string data
                string hora
            }
            "Paciente" {
                number num_pac
                string nome
                string endereço
            }
            "Fármaco" {
                string código
                string nome
            }
        }
    
```

42

42

## 2 - Empresa

- Uma empresa está organizada em departamentos.
- Cada departamento tem um nome único, um número único e um gerente, devendo-se registar a data em que o gerente começou a gerir o departamento. Um departamento pode ter várias localizações.
- Um departamento controla um determinado número de projectos. Cada projeto tem um nome único, um número único e uma localização.
- Para cada empregado deve-se guardar o nome, o número da segurança social, o endereço, o salário, o sexo e a data de nascimento.
- Um empregado pertence a um departamento, trabalhar em um ou mais projetos, que não são necessariamente controlados pelo mesmo departamento.
- Deve-se registrar o número de horas (por semana) que um empregado trabalha num dado projeto.
- Deve-se registrar o supervisor direto de cada empregado.
- Devemos registrar os dependentes de cada empregado. Queremos guardar  $\frac{q}{3}$  nome do dependente, o sexo, data de nascimento e ligação ao empregado.

43

## 2 - Empresa

- Identificação das entidades
    - departamento
    - empregado
    - projeto
    - dependente
  - Identificação das relações entre entidades (cardinalidade)
    - empregado gere departamento (1:1)
    - empregado trabalha para departamento (N:1)
    - departamento controla projeto (1:N)
    - empregado trabalha em projeto (N:M)
    - supervisor supervisiona empregado (1:N)
    - empregado tem dependente (1:N)
- ...

44

44

deti

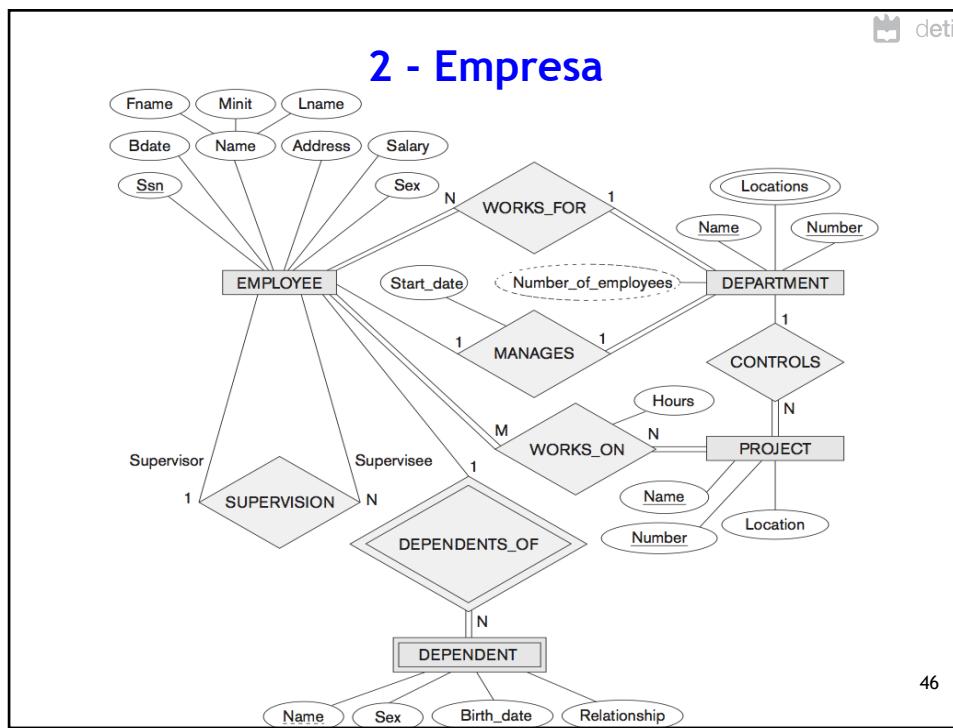
## 2 - Empresa

...

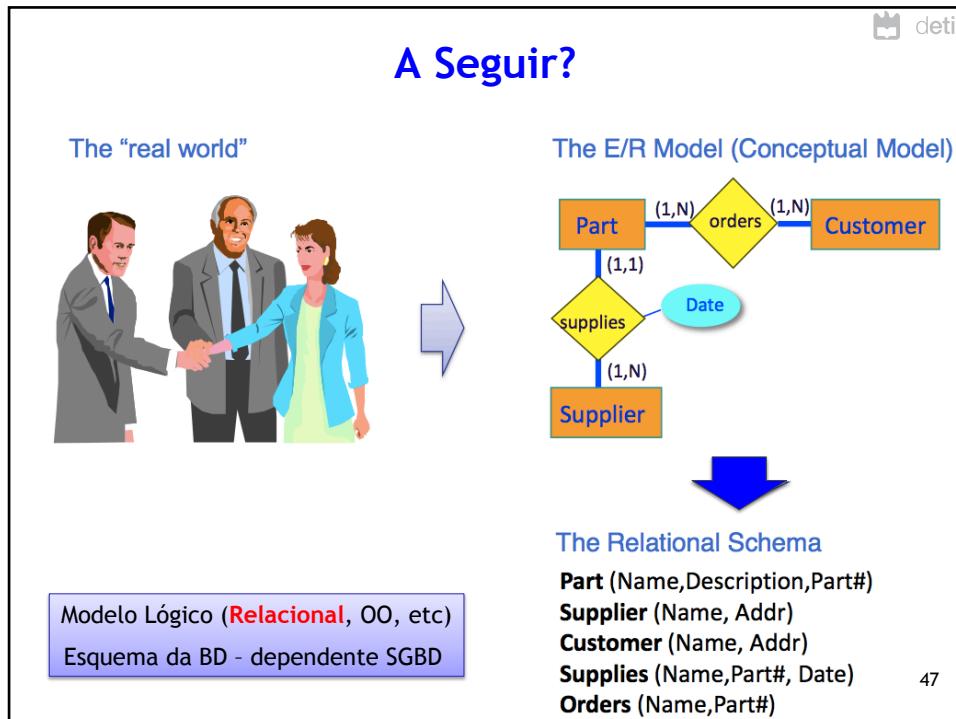
- Obrigatoriedade
  - todos os departamentos tem um gestor / nem todos os empregados são gestores.
  - um departamento tem pelo menos um empregado / um empregado trabalha sempre para um departamento.
  - todos os projetos têm um departamento a controlá-los / nem todos os departamentos controlam projetos.
  - um empregado trabalha em pelo menos um projeto / um projeto tem pelo menos um empregado.
  - todos os dependentes estão associados a um empregado / nem todos os empregados têm dependentes.
  - nem todos os empregados são supervisores / nem todos os empregados são supervisionados.
- Identificação dos atributos de cada entidade e relação...

45

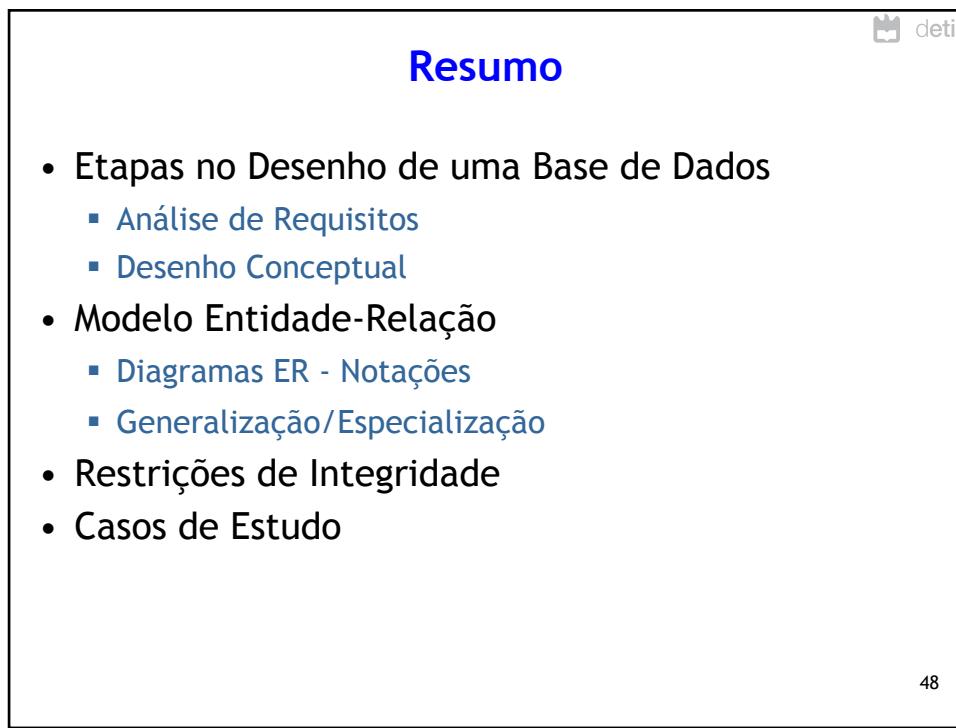
45



46



47



48

deti departamento de  
electrónica, telecomunicações  
e informática

## Modelo Relacional

Base de Dados - 2020/21  
Carlos Costa

1

deti

## Introdução

- Modelo proposto por Edgar F. Codd em 1970
  - garante uma grande independência de dados.

---

*Information Retrieval*

*P. BAXENDALE, Editor*

**A Relational Model of Data for Large Shared Data Banks**

E. F. CODD  
*IBM Research Laboratory, San Jose, California*

*Future users of large data banks must be protected from*

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Volume 13 / Number 6 / June, 1970

Communications of the ACM      377

2

## Modelo Relacional - Introdução

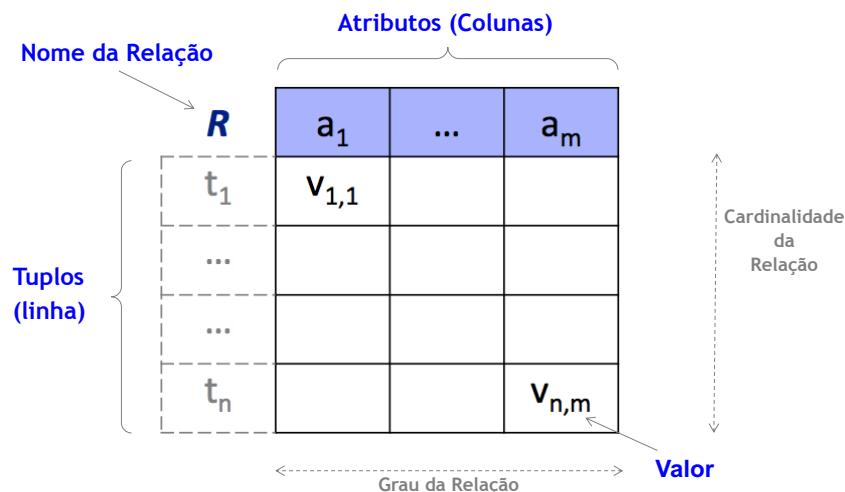
- Modelo baseado na Teoria dos Conjuntos.
  - Modelo matemático rigoroso
    - Anteriores evoluíram das técnicas de processamento de ficheiros
- Baseado na noção matemática de “**Relação**”, representadas por **Tabelas**.
- Dispõem de um sistema formal de manipulação das relações - **Álgebra Relacional** (próximas aulas).
- Utilização comercial no início dos anos 80.
  - Devido a restrições de hardware e linguagem de programação
- Contribuiu para a massificação das tecnologias de base de dados.

3

3

## Conceitos (1/4)

- Base do Modelo Relacional - **Relação (Tabela)**



4

4

## Conceitos (2/4)

- **Atributo** (A<sub>1</sub>, A<sub>2</sub>,..., A<sub>n</sub>)
  - Representam o tipo de dados a armazenar.
  - O número de atributos de uma relação define o **grau da relação**.
  - Os atributos de uma relação devem ter nomes distintos.
- **Domínio** (D<sub>1</sub>, D<sub>2</sub>,...,D<sub>n</sub>)
  - Tipo de dados
  - Gama de valores possíveis para determinado **atributo**.
    - Sexo** {'M', 'F'}
    - Cidade** {Porto, Aveiro, Coimbra,...}
    - Nome** {Maria, João, Ana, Sofia,...}
  - Valores desconhecidos ou não existentes.  
**NULL**

5

5

## Conceitos (3/4)

- **Esquema da Relação** - R(A<sub>1</sub>, A<sub>2</sub>,...,A<sub>n</sub>)
  - *Relational Schema*
  - Nome do esquema e lista de atributos,  
**Pessoa(nome, bi, idade)**
  - Opcionalmente: inclui o tipo dos atributos  
**Pessoa(nome:string, bi:integer, idade:integer)**
- **Relação** - r(R)
  - Estrutura bidimensional com determinado **esquema** e zero ou mais **instâncias (tuplos)**.  
**r = {t<sub>1</sub>, t<sub>2</sub>, ..., t<sub>m</sub>}**
  - Formalmente é um subconjunto do produto cartesiano  
**r(R) ⊆ (dom(A<sub>1</sub>) × dom(A<sub>2</sub>) × ... × dom(A<sub>n</sub>))**

6

6

## Conceitos (4/4)

- **Tuplo**

- Linha de uma relação.  
 $t = \langle v_1, v_2, \dots, v_n \rangle$
- Devem ser distintos (numa relação)  $\rightarrow$  Set
- A ordem das linhas é indiferente.
- O número de tuplos define a cardinalidade da relação.

- **Atomicidade**

- O valor de um atributo num tuplo é atómico (não é composto/multi-valor).

- **Esquema da Base de Dados (Database Schema)**

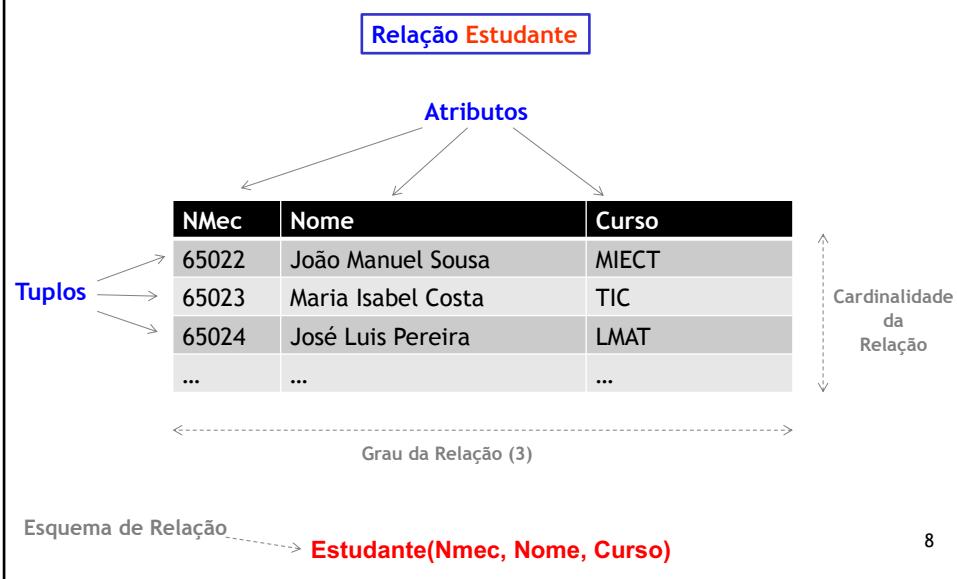
- conjunto de todos os esquemas da relação da BD.

$$D = \{R_1(X_1), \dots, R_n(X_n)\}$$

7

7

## Relação - Exemplo 1



8

8

**Relação - Exemplo 2**

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

- Esquema Relação
 

`STUDENT(Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa)`

`STUDENT(Name: string, Ssn: string, Home_phone: string, Address: string, Office_phone: string, Age: integer, Gpa: real)`
- Tuplo da Relação
 

`t = < (Name, Dick Davidson), (Ssn, 422-11-2320), (Home_phone, NULL), (Address, 3452 Elgin Road), (Office_phone, (817)749-1253), (Age, 25), (Gpa, 3.53) >`

9

**Relação - Chaves**

- **Superchave (superkey)**: conjunto de atributos que identificam de forma única os tuplos da relação.
- **Chave Candidata (candidate key)**: subconjunto de atributos de uma superchave que não pode ser reduzido sem perder essa qualidade de superchave.
- **Chave Primária (primary key)**: chave principal selecionada de entre as chaves candidatas.
- **Chave Única (unique key)**: chave candidata não eleita como primária.
- **Chave Estrangeira ou importada (foreign key)**: conjunto de um ou mais atributos que é chave primária noutra relação.<sup>10</sup>

10

 deti

## SuperChaves e Chaves Candidatas

- Cada relação tem pelo menos uma superchave
  - Conjunto de todos os atributos

Exemplo

**Estudante**(Nome, Email, NMec, Curso)

Superchaves:

{Nome, Email, NMec, Curso},	}	Chaves Candidatas ?	{Email}
{Nome, Email, NMec},			{NMec}
{Nome, Email},			
{Nome, NMec},			
{Email, NMec},			
{Email},			
{NMec}			

Lista não exaustiva

11

11

 deti

## Chave Primária

- A **escolha da chave primária** (de entre as candidatas) é **arbitrária**.
- As chaves candidatas não eleitas (primária) designam-se como **chaves únicas**.
- A chave primária **não pode** ter valor **NULL**.
- Recomendação: ter critério na escolha da chave primária. Por exemplo:
  - Elemento “natural” de identificação
  - Atributo cujo valor nunca (raramente) é alterado.

No exemplo do slide anterior, qual das chaves candidatas devo escolher para chave primária? Email ou NMec?

Mais razões... ?      Estudante(Nome, Email, NMec, Curso)<sup>12</sup>

12

 deti

## Chaves - Exemplo

**CAR**

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Duas chaves candidatas:

- Licence\_number e Engine\_serial\_number

Escolhemos com chave primária:

- Licence\_number

13

13

 deti

## Chaves - Relacionamento entre Tabelas

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	1

**Resumo:**

- Temos a relação EMPLOYEE (Funcionário) e a relação DEPARTMENT (Departamento)
- Um EMPLOYEE trabalha num DEPARTMENT
- Dnumber é **chave primária** na relação DEPARTMENT
- Dno é **chave estrangeira** na relação EMPLOYEE

**DEPARTMENT**

Dname	Dnumber
Research	5
Administration	4
Headquarters	1

14

Chave Primária

14

## Restrições de Integridade

- São regras que visam garantir a integridade dos dados.
  - Devem ser garantidas pelo próprio SGBD.

Tipos:

- **Domínio** - dos **atributos**. Forma mais elementar de integridade. Os campos devem obedecer ao tipo de dados e às restrições de valores admitidos para um atributo.
- **Entidade** - cada **tuplo** deve ser identificado de forma única com recurso a uma **chave primária** que não se repete e não pode ser null (condição de **set**).
- **Referencial** - o valor de uma **chave estrangeira** ou é **null** ou contém um valor que é **chave primária** na relação de onde foi importada.

15

15

## Regras de Codd - 1

- Como definir (verificar se) um SGBD é ou não relacional?
- Codd estabeleceu uma lista de 12 regras\* que definem/avaliam um sistema de modelo relacional.
- Vários autores (próprio Codd) reconhecem ser difícil encontrar implementações que, à luz das 12 regras, possam ser consideradas completamente relacional.
- No entanto foram muito importantes para combater posicionamentos proprietários da indústria de SGBD.

\*Codd, E. (1985). "Is Your DBMS Really Relational?" and "Does Your DBMS Run By the Rules?"  
ComputerWorld, October 14 and October 21.

16

16

## Regras de Codd - 2

### 1. Representação da Informação

- Numa base de dados relacional, todos os dados, incluindo o próprio dicionário de dados, são representados de uma só forma, em tabelas bidimensionais.

### 2. Acesso garantido

- Cada elemento de dados fica bem determinado pela combinação do nome da tabela onde está armazenado, valor da chave primária e respectiva coluna (atributo).

### 3. Suporte sistemático de valores nulos (NULL)

- Valores NULL são suportados para representar informação não disponível ou não aplicável, independentemente do domínio dos respectivos atributos.

### 4. Catálogo activo e disponível

- Os metadados são representados e acedidos da mesma forma que os próprios dados<sup>17</sup>

17

## Regras de Codd - 3

### 5. Linguagem completa

- Apesar de um sistema relacional poder suportar várias linguagens, deverá existir pelo menos uma linguagem com as seguintes características:
  - Manipulação de dados, com possibilidade de utilização interativa ou em programas de aplicação.
  - Definição de dados.
  - Definição de views.
  - Definição de restrições de integridade.
  - Definição de acessos (autorizações).
  - Manipulação de transações (commit, rollback, etc.).

### 6. Regra da atualização de vistas (view)

- Numa vista, todos os dados modificados (em atributos actualizáveis) devem ver essas modificações traduzidas nas tabelas base.

### 7. Operações de alto-nível

- Capacidade de tratar uma tabela (base ou virtual) como se fosse um simples operando (ou seja, utilização de uma linguagem set-oriented), tanto em operações de consulta como de atualização ou eliminação.

18

18

## Regras de Codd - 4

### 8. Independência física dos dados

- Alterações na organização física dos ficheiros da base de dados ou nos métodos de acesso a esses ficheiros (nível interno) não devem afectar o nível lógico.

### 9. Independência lógica dos dados

- Alterações no esquema da base de dados (nível lógico), que não envolvam remoção de elementos, não devem afectar o nível externo.

### 10. Restrições de integridade

- As restrições de integridade devem poder ser especificadas numa linguagem relacional, independentemente dos programas de aplicação, e armazenadas no dicionário de dados.

### 11. Independência da localização

- O facto de uma base de dados estar centralizada numa máquina, ou distribuída por várias máquinas, não deve repercutir-se ao nível da manipulação dos dados.

### 12. Não subversão

- Se existir no sistema uma linguagem de mais baixo-nível (tipo record-oriented), ela não deverá permitir ultrapassar as restrições de integridade e segurança.

19

## Conversão do DER em Modelo Relacional

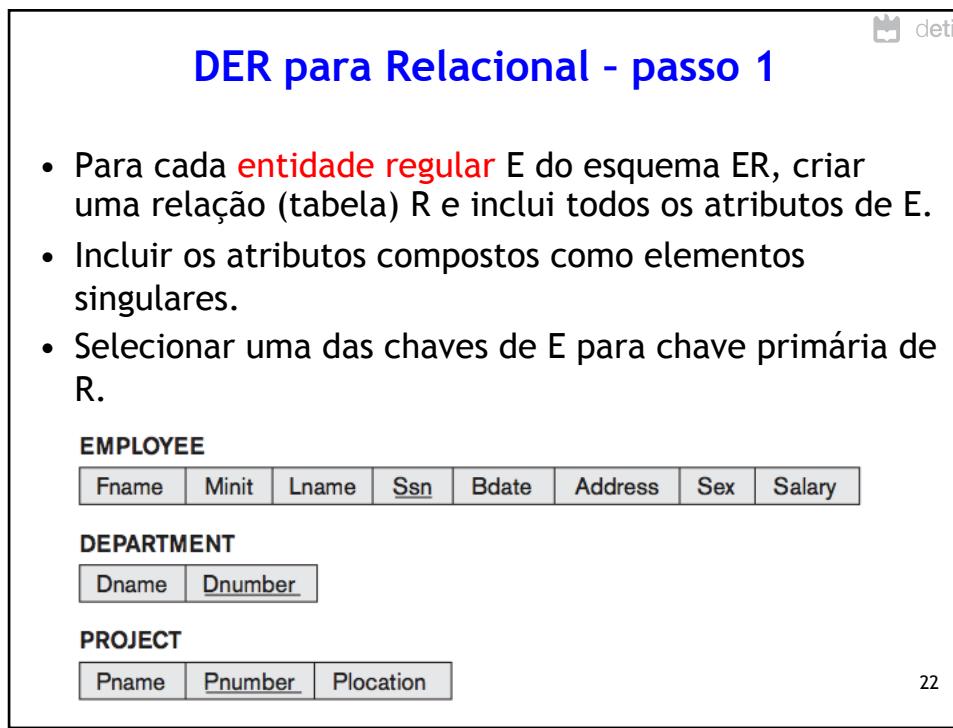
- Um desenho conceptual de uma base de dados, utilizando DER, pode ser representado por intermédio de um conjunto de relações (tabelas)
- Cada conjunto de entidades e relações do DER vai gerar uma única relação (tabela) com o nome do respectivo conjunto.
- *Mapping Process*
  - Vamos seguir um conjunto de regras.
- Caso Estudo: DER da Empresa

20

20



21



22

**DER para Relacional - passo 2**

- Cada **entidade fraca** W do esquema ER é representada por uma relação (tabela) R que inclui os seu atributos, assim como a chave primária da entidade dominante E que passará a ser chave estrangeira em R.
- Incluir os atributos compostos de W, caso existam, como elementos singulares.
- A chave primária de R é a combinação da chave primária de E e da chave parcial de W.

DEPENDENT				
Essn	Dependent_name	Sex	Bdate	Relationship

The diagram shows a partial dependency relationship between the DEPENDENT entity and the EMPLOYEE entity. The DEPENDENT entity has attributes Name, Sex, Birth\_date, and Relationship. The EMPLOYEE entity has attributes Essn, Dependent\_name, Sex, Bdate, and Relationship. There is a partial dependency relationship labeled 'N' from DEPENDENT to EMPLOYEE, indicated by a line connecting the two entities with a circle containing 'N' at the end of the line pointing to the EMPLOYEE entity.

23

23

**DER para Relacional - passo 3**

- Para cada **relacionamento 1:1** do esquema ER, envolvendo as relações S e T:
  - escolher uma das relações, digamos S, e incluir como chave estrangeira, a chave primária da outra relação.
  - incluir em S eventuais atributos do relacionamento.
  - devemos escolher como S uma relação com participação total.

DEPARTMENT			
Dname	Dnumber	Mgr_ssn	Mgr_start_date

The diagram shows a 1:1 relationship between the EMPLOYEE entity and the DEPARTMENT entity via the MANAGES diamond. The EMPLOYEE entity has attributes Essn, Dependent\_name, Sex, Bdate, and Relationship. The DEPARTMENT entity has attributes Dname, Dnumber, Mgr\_ssn, and Mgr\_start\_date. The MANAGES diamond has attributes Start\_date and a cardinality of 1 on both sides. There is also a partial dependency relationship from DEPARTMENT to EMPLOYEE.

Escolhemos com S a relação DEPARTMENT e incluímos a chave primária de EMPLOYEE como chave estrangeira.

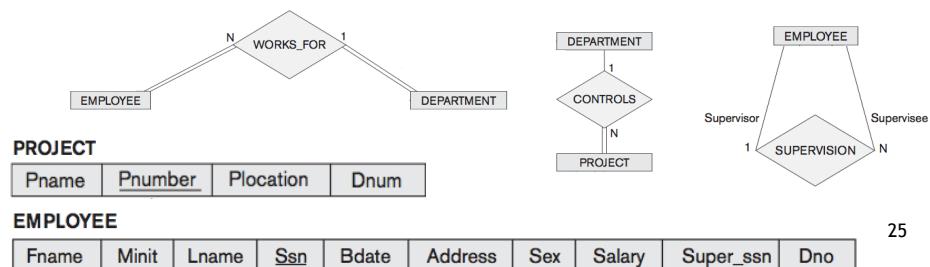
24

Nota: existem outras abordagens. Por exemplo, criar uma nova relação caso não exista participação total -> ver caso N:M

24

## DER para Relacional - passo 4

- Para cada **relacionamento 1:N** do esquema ER, envolvendo as relações S e T:
  - escolher como S a relação que representa a entidade do lado N e como T a que representa a entidade do lado 1.
  - incluir em S, como chave estrangeira, a chave primária da relação T.
  - incluir os atributos do relacionamento em S.

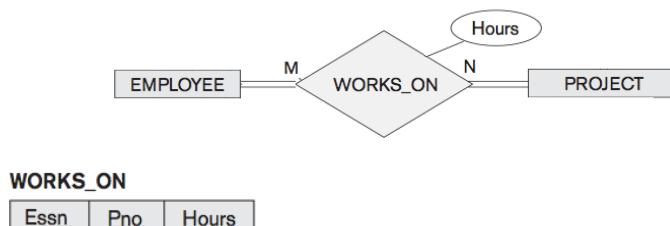


25

25

## DER para Relacional - passo 5

- Para cada **relacionamento N:M** do esquema ER, criar uma nova relação (tabela) R.
  - incluir como chave estrangeira as chaves primárias das relações que participam em R. Estas chaves combinadas formarão a chave primária da relação R.
  - incluir os atributos do relacionamento em R.

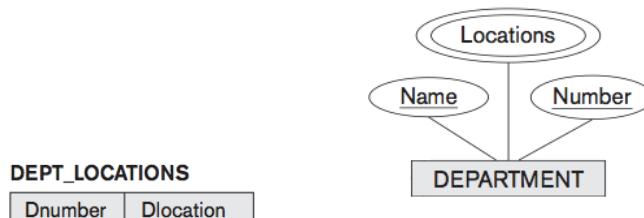


26

26

## DER para Relacional - passo 6

- Para cada **atributo multi-valor A** do esquema ER, criar uma nova relação (tabela) R.
  - incluir um atributo correspondendo a A.
  - incluir a chave primária K da relação que tem A como atributo.
  - a chave primária de R é a combinação de A e K.



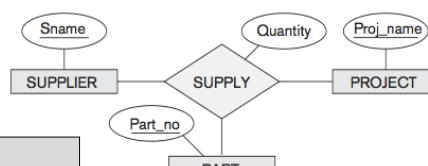
27

27

## DER para Relacional - passo 7

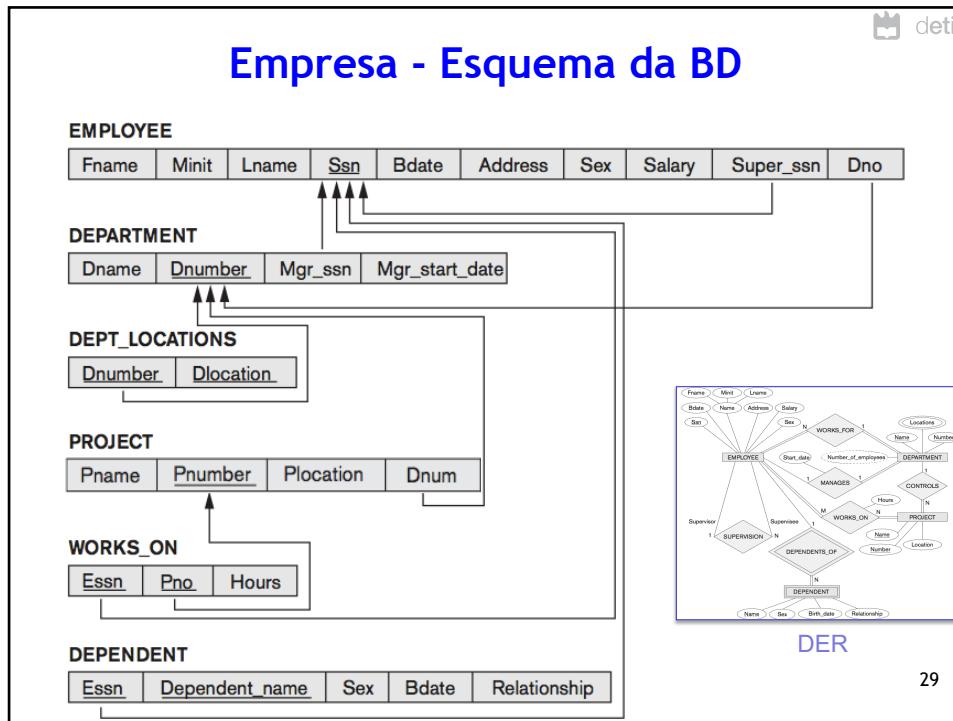
- Para cada **relacionamento n-ário** ( $n > 2$ ):
  - criar uma nova relação (tabela) R
  - incluir, como chaves estrangeiras, as chaves primárias das relações que representam as entidades participantes
  - incluir os eventuais atributos do relacionamento
  - a chave primária de R é normalmente a combinação das chaves estrangeiras

SUPPLIER
<u>Sname</u> ...
PROJECT
<u>Proj_name</u> ...
SUPPLY
<u>Sname</u> <u>Proj_name</u> <u>Part_no</u> <u>Quantity</u>



28

28



29

**Instância da BD Empresa - Exemplo**

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

**DEPARTMENT**

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

**DEPT\_LOCATIONS**

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

**WORKS\_ON**

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

**PROJECT**

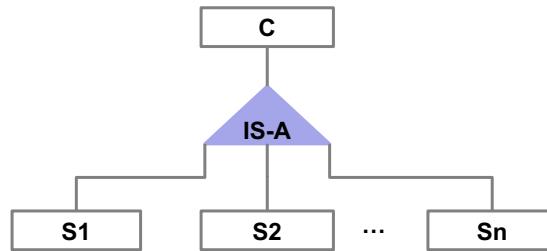
Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

30

30

## DER para Relacional - Especialização

- Várias aproximações possíveis... vamos apresentar duas usuais.



superclasse C {k, a<sub>1</sub>, ..., a<sub>n</sub>}, k é chave primária  
 n subclasses {S<sub>1</sub>, S<sub>2</sub>, ..., S<sub>n</sub>}

31

31

## DER para Relacional - Especialização

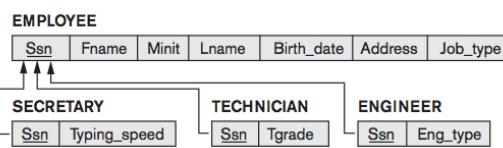
### Método 1

- Formar uma relação (tabela) L para a entidade de maior nível (C)

$\text{Attrs}(L) = \{k, a_1, \dots, a_n\}$  e  $\text{PK}(L) = k$

- Criar uma relação Li para cada entidades de nível inferior. Incluir em cada uma destas relações a chave primária de C e os atributos locais.

$\text{Attrs}(Li) = \{k\} \cup \{\text{attributes of } Si\}$  e  $\text{PK}(Li) = k$



32

Funciona com qualquer tipo de especialização: Total/Parcial, Disjunta/Sobreposta

32

## DER para Relacional - Especialização

### Método 2

- Criar uma relação Li para cada entidade de nível inferior. Incluir os atributos da superclasse e os atributos locais.

$$\text{Attrs}(Li) = \{\text{attributes of } Si\} \cup \{k, a_1, \dots, a_n\} \text{ e } \text{PK}(Li) = k$$

CAR
Vehicle_id   License_plate_no   Price   Max_speed   No_of_passengers
TRUCK
Vehicle_id   License_plate_no   Price   No_of_axles   Tonnage

Só funciona com especialização total.

Só se recomenda em especializações disjuntas pois nas sobrepostas há duplicação de informação da mesma entidade por várias relações (tabelas).

33

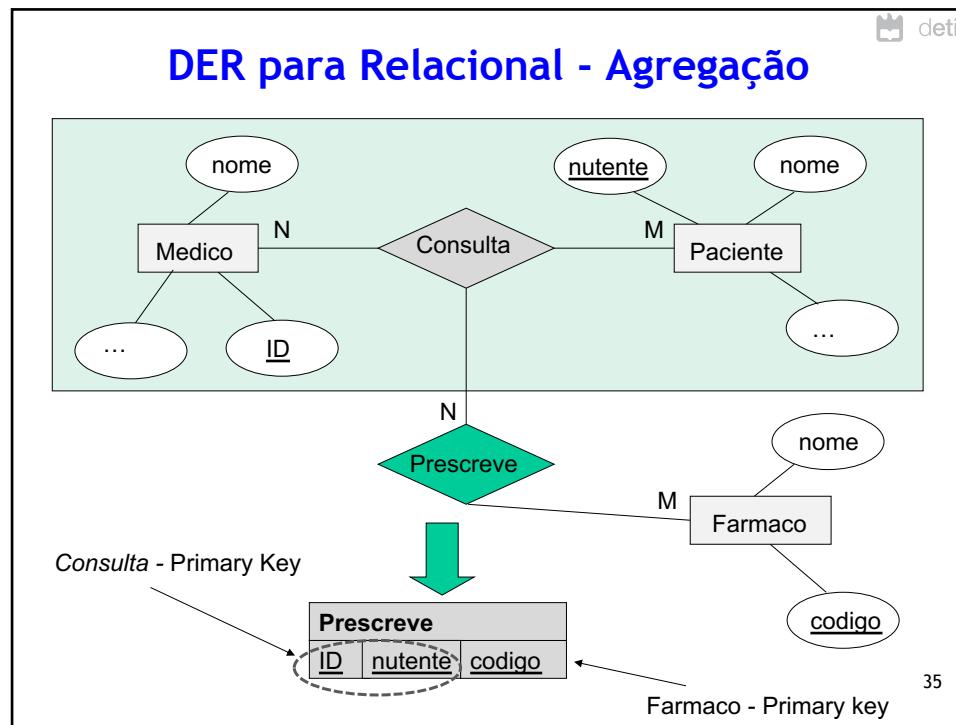
33

## DER para Relacional - Resumo

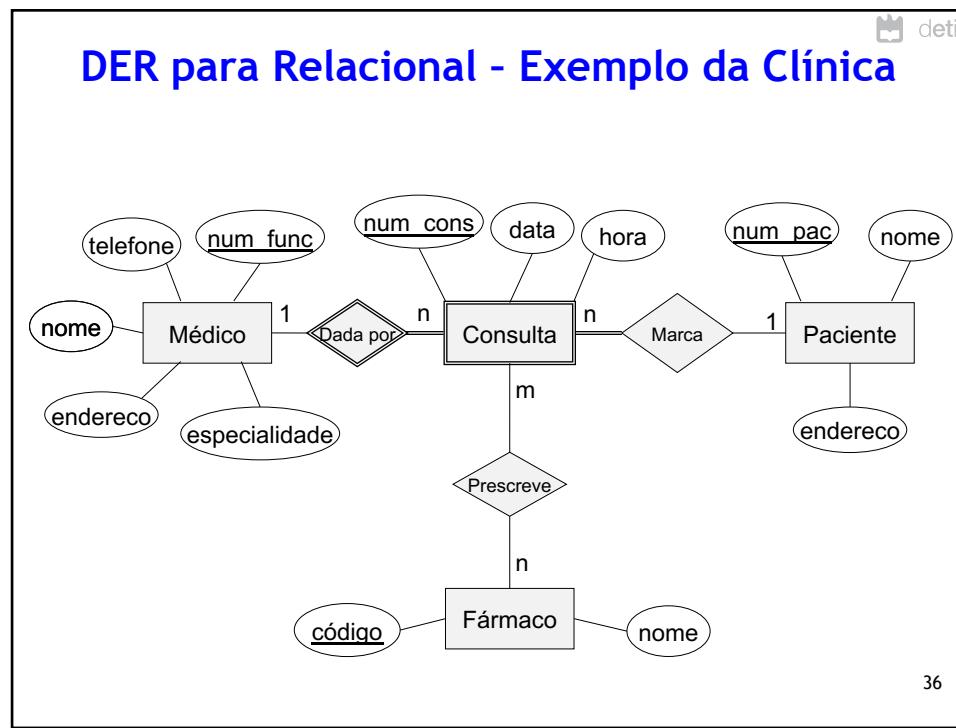
ER MODEL	RELATIONAL MODEL
Entity type	Entity relation
1:1 or 1:N relationship type	Foreign key (or <i>relationship</i> relation)
M:N relationship type	<i>Relationship</i> relation and <i>two</i> foreign keys
<i>n</i> -ary relationship type	<i>Relationship</i> relation and <i>n</i> foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

34

34



35



36

 deti

## DER para Relacional - Exemplo da Clínica

- Passo 1 (entidades regulares)

Médico
<u>num_func</u> (PK)   nome   telefone   endereco   especialidade

Paciente	Fármaco
<u>num_pac</u> (PK)   nome   endereco	<u>codigo</u> (PK)   nome

- Passo 2 (entidades fracas)

Consulta
<u>medico</u> (FK)(PK)   <u>num_consulta</u> (PK)   data   hora

37

37

 deti

## DER para Relacional - Exemplo da Clínica

- Passo 3 (rel. 1:1)
  - Não se aplica
- Passo 4 (rel. 1:N)

Consulta
<u>medico</u> (FK1) (PK)   <u>num_consulta</u> (PK)   <u>paciente</u> (FK2)   data   hora

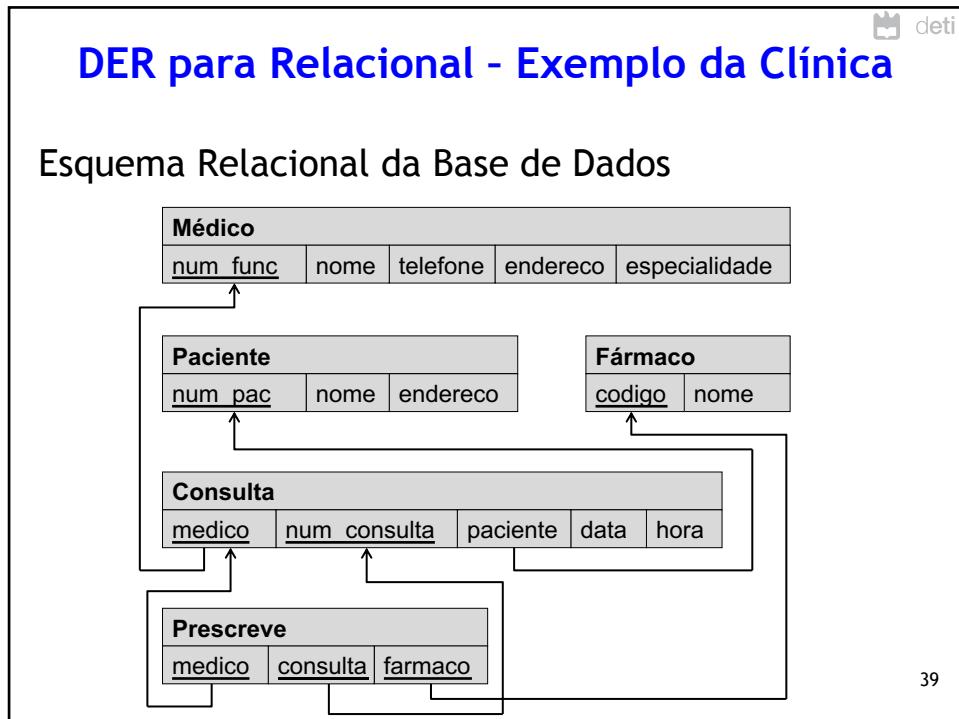
- Passo 5 (rel. N:M)

Prescreve
<u>medico</u> (FK1)(PK)   <u>consulta</u> (FK1)(PK)   <u>farmaco</u> (FK2)(PK)

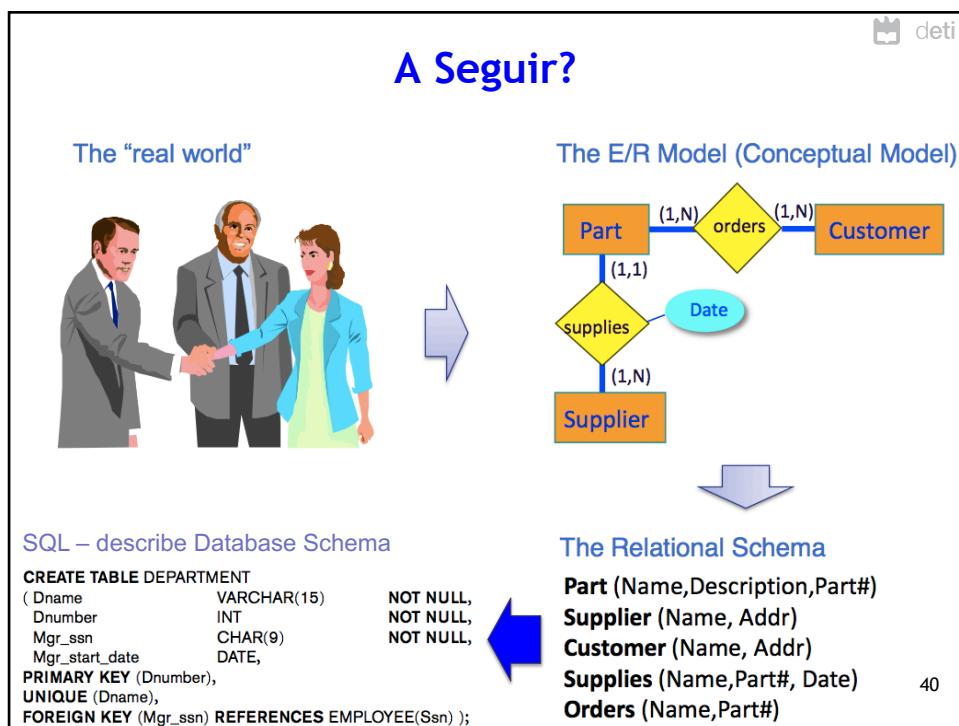
- Passo 6 e 7
  - Não se aplicam

38

38



39



40

## Resumo

- Desenho Lógico de BD
- Modelo Relacional
- Restrições de Integridade
- Conversão de Diagramas Entidade-Relação para Esquema Relacional
- Casos de Estudo

41

41

## Linguagem SQL - DDL

Base de Dados - 2020/21

Carlos Costa

1

## Linguagem SQL

- Structured Query Language (SQL)
  - SEQUEL
- Linguagem para definir, manipular e questionar uma Base de Dados Relacional.
  - É uma linguagem orientada ao processamento de conjuntos
- 2 sublinguagens principais
  - DDL - Data Definition Language.
  - DML - Data Manipulation Language.
- 1 sublinguagem de controlo BD
  - DCL - Data Control Language

2

2

 deti

## SQL - Versões

- 1986 (SQL-86 e SQL-87)
  - Publicado pela ANSI e ratificado pela ISO.
- 1989 (SQL-89)
- 1992 (SQL-92)
  - conhecido como SQL2.
- 1999 (SQL:1999)
  - conhecido como SQL 3.
  - inclui expressões regulares, queries recursivas, triggers, tipos não escalares, procedimentos, funcionalidades orientadas a objectos, etc.
- 2003 (SQL:2003)
  - Inclui suporte a XML e colunas com numeração automática.
- 2006 (SQL:2006)
  - Define formas de interacção SQL-XML: como importar e armazenar XML em BD SQL, XQuery, etc.
- 2008
- 2011

3

3

 deti

## SQL - SQL Server

- Vamos utilizar, como ferramenta de trabalho, a versão SQL Server (>=2012)

### Transact-SQL

“Microsoft SQL Server team has extended the ANSI definition with several enhancements and new commands, and has left out a few commands because SQL Server implemented them differently. The result is Transact-SQL, or T-SQL – the dialect of SQL understood by SQL Server”

“Missing from T-SQL are very few ANSI SQL commands, primarily because Microsoft implemented the functionality in other ways.”

4

Microsoft® SQL Server® 2008 Bible

4

## SQL - Hierarquia de Objetos

A diagram illustrating the SQL object hierarchy. It consists of four concentric ovals. The innermost oval is labeled "column". Surrounding it is a larger oval labeled "table". This is followed by another oval labeled "schema". The outermost oval is labeled "catalog".

Mas há mais elementos como, por exemplo, triggers, vistas, índices, stored procedures, funções, etc.

5

5

## SQL - catalog, schema e database

The screenshot shows two side-by-side database management interfaces. On the left is the MySQL Workbench interface, which displays a tree view of databases (information\_schema, def, mysql, performance\_schema, test, Hospital) and tables (WL). On the right is the Microsoft SQL Server Management Studio interface, which shows the Object Explorer with a table named 'dbo.WL' selected. The Properties window on the right details the table's structure, including columns like AccessionNumber, PatientID, PatientName, PatientBirthDate, PatientSex, Modality, MedicalAlerts, and StudyInstanceUID. A note indicates that the term 'Catalog' is used in MySQL, while 'Database' is used in SQL Server.

O significado destes termos varia de acordo com SGBD  
SQL Server: database\_name . schema\_name . table\_name

6

6

## SQL - Notas introdutórias

- SQL utiliza...
  - tabela, linha e coluna (table, row and column)
  - ... para designar os termos formais:
  - relação, tuplo e atributo do modelo relacional
- Cada instrução SQL termina com um ponto e vírgula (“;”)
- Comentar uma linha “--”
- Comentar um bloco de instruções /\* ... \*/

7

7

## SQL - Data Definition Language (DDL)

- Permite definir várias entidades da BD
- Utilizada para especificar a informação acerca de cada relação:
  - O esquema de cada relação.
  - O domínio de valores associados com cada atributo.
  - Restrições de integridade (entidade e referencial)
  - O conjunto de índices a manter para cada relação
  - ...
- Notas importantes:
  - Há comandos não disponíveis em alguns SGBD...
  - Devemos consultar o manual do SGBD para uma sintaxe mais completa dos comandos.

8

8



## Criar e Eliminar uma Base de Dados

- Criar uma base de dados

```
CREATE DATABASE dbname;
```

**dbname** - nome da base de dados a criar

```
CREATE DATABASE COMPANY;
```

- Eliminar uma base de dados

```
DROP DATABASE dbname;
```

**dbname** - nome da base de dados a eliminar

```
DROP DATABASE COMPANY;
```

9

9



## Schema

- Schema é um “namespace” que agrupa tabelas e outros elementos pertencentes à mesma aplicação.
- Criar um Schema

```
CREATE SCHEMA schemaname [AUTHORIZATION username];
```

```
CREATE SCHEMA COMPANY AUTHORIZATION 'CCosta';
```

- Eliminar um Schema

```
DROP SCHEMA schemaname;
```

```
DROP SCHEMA COMPANY;
```

10

MySQL - sinónimo de “CREATE DATABASE” !

10

## SQL - Tipo de Dados

- Tipos de dados básicos:
  - Numbers
  - Characters, strings
  - Date e time
  - Binary objects
- Os tipos de dados podem variar de acordo com o SGDB!
- Recomendação: Utilizar, na medida do possível, tipos de dados compatíveis com o standard.
  - Aumenta a portabilidade da solução...

11

11

## SQL - Tipos de dados (SQL:1999)

- Numeric
    - NUMERIC(p,s) e.g. 300.00
    - DECIMAL(p,s)
    - INTEGER (alias: INT) e.g. 32767
    - SMALLINT small integers
    - FLOAT(p) e.g. -1E+03
    - REAL (for short floats) DOUBLE (for long floats)
  - String
    - CHARACTER(n) (fixed length)
    - CHARACTER (variable lenght)
    - CHARACTER VARYING(n) (alias: VARCHAR(n))
    - CLOB (Character Large Object, e.g., for large text)
  - Date
    - DATE e.g. '1993-01-02'
    - TIME e.g. '13:14:15'
    - TIMESTAMP e.g. '1993-01-02 13:14:15.000001'
  - Binary
    - BIT[(n)] e.g. B'01000100'
    - BLOB[(n)] e.g. X'49FE' (Binary Large Objects, e.g., for multimedia)
  - Boolean
    - Boolean
- Listagem não exaustiva...

12

12

deti

## SQL - Tipo de Dados

Alguns mais utilizados...

- **char(n)**
  - cadeia de caracteres de tamanho fixo n
- **varchar(n)**
  - cadeia de caracteres com tamanho máximo n
- **int**
  - números inteiros (4 bytes)
- **numeric(precisão, escala)**
  - números reais “sem limite” de tamanho
- **date e time**
  - data e hora
- **boolean\***
  - valores booleanos

\* Não existe em SQL Server

13

13

deti

## SQL Server - Tipos de Dados

**Numeric Data Types**

Data Type	Description	Length
int	Stores integer values ranging from -2,147,483,648 to 2,147,483,647	4 bytes
tinyint	Stores integer values ranging from 0 to 255	1 byte
smallint	Stores integer values ranging from -32,768 to 32,767	2 bytes
bigint	Stores integer values ranging from -253 to 253-1	8 bytes
money	Stores monetary values ranging from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
smallmoney	Stores monetary values ranging from -214,748.3648 to 214,748.3647	4 bytes
decimal(p,s)	Stores decimal values of precision p and scale s. The maximum precision is 38 digits	5-17 bytes
numeric(p,s)	Functionally equivalent to decimal	5-17 bytes
float(n)	Stores floating point values with precision of 7 digits (when n=24) or 15 digits (when n=53)	4 bytes (when n=24) or 8 bytes (when n=53)
real	Functionally equivalent to float(24)	4 bytes

14

14

## SQL Server- Tipos de Dados (cont.)

Character String Data Types

Data Type	Description	Length
char(n)	Stores n characters	n bytes (where n is in the range of 1–8,000)
nchar(n)	Stores n Unicode characters	2n bytes (where n is in the range of 1–4,000)
varchar(n)	Stores approximately n characters	Actual string length +2 bytes (where n is in the range of 1–8,000)
varchar(max)	Stores up to $2^{31}-1$ characters	Actual string length +2 bytes
nvarchar(n)	Stores approximately n characters	2n(actual string length) +2 bytes (where n is in the range of 1–4,000)
nvarchar(max)	Stores up to $((2^{31}-1)/2)-2$ characters	2n(actual string length) +2 bytes

Binary Data Types

Data Type	Description	Length
bit	Stores a single bit of data	1 byte per 8 bit columns in a table
binary(n)	Stores n bytes of binary data	n bytes (where n is in the range of 1–8,000)
varbinary(n)	Stores approximately n bytes of binary data	Actual length +2 bytes (where n is in the range of 1–8,000)
varbinary(max)	Stores up to $2^{31}-1$ bytes of binary data	Actual length +2 bytes

15

15

## SQL Server- Tipos de Dados (cont.)

Date and Time Data Types

Data Type	Description	Length	Example
date	Stores dates between January 1, 0001, and December 31, 9999	3 bytes	2008-01-15
datetime	Stores dates and times between January 1, 1753, and December 31, 9999, with an accuracy of 3.33 milliseconds	8 bytes	2008-01-15 09:42:16.142
datetime2	Stores date and times between January 1, 0001, and December 31, 9999, with an accuracy of 100 nanoseconds	6–8 bytes	2008-01-15 09:42:16.1420221
datetimeoffset	Stores date and times with the same precision as datetime2 and also includes an offset from Universal Time Coordinated (UTC) (also known as Greenwich Mean Time)	8–10 bytes	2008-01-15 09:42:16.1420221 +05:00
smalldatetime	Stores dates and times between January 1, 1900, and June 6, 2079, with an accuracy of 1 minute (the seconds are always listed as “00”)	4 bytes	2008-01-15 09:42:00
time	Stores times with an accuracy of 100 nanoseconds	3–5 bytes	09:42:16.1420221

16

Listagem não exaustiva. Há outros tipo como o cursor, sql\_variant, table, xml, ...

16

## SQL - Definição de Domínio

- O comando `create domain` permite definir novos tipos de dados.
- Um domain pode conter um valor de defeito (default) e restrições do tipo not null e check.

`CREATE DOMAIN domainname`

**Criação...**

```
CREATE DOMAIN compsalary INTEGER
    NOT NULL CHECK (compsalary > 475);
```

**Utilização...**

```
CREATE TABLE EMPLOYEE (
    ...
    Salary          compsalary,
    ...);
```

Nota: Não disponível em SQL SERVER.

17

17

## SQL - Definição de Novo Tipo

- Como alternativa ao domain, podemos criar só um novo tipo (alias) com o comando `create type`.

`CREATE Type... em SQL SERVER`

**Criação...**

```
CREATE TYPE SSN FROM varchar(9) NOT NULL;
```

**Utilização...**

```
CREATE TABLE EMPLOYEE (
    ...
    Ssn           SSN,
    ...);
```

- Nota: Em geral, é mais limitado que o create domain.

18

18

 deti

## DDL - Criar uma Tabela

```
CREATE TABLE tbname ( A1 D1, A2 D2, ..., An Dn,
                      (integrity-constraint1),
                      ...
                      (integrity-constraintK) );
tbname - nome da relação (tabela)
```

CREATE TABLE COMPANY.EMPLOYEE (...)  
CREATE TABLE EMPLOYEE (...)

COMPANY - nome do schema

A1 D1, A2 D2, ..., An Dn  
A1...An - Atributos da relação  
D1...Dn - Domínio dos atributos

Restrições de Integridade  
integrity-constraint1,  
...,  
integrity-constraintN

19

 deti

## Criar uma Tabela (exemplo)

```
CREATE TABLE...
definindo atributos e respectivo domínio.
```

```
CREATE TABLE EMPLOYEE (
    Fname           VARCHAR(15),
    Minit          CHAR,
    Lname           VARCHAR(15),
    Ssn            CHAR(9),
    Bdate          DATE,
    Address        VARCHAR(30),
    Sex             CHAR,
    Salary          DECIMAL(10,2),
    Super_ssn      CHAR(9),
    Dno             INT);
```

20

20

## Atributos - Valores por Omissão

- Podem ser definidos valores por omissão para cada coluna
  - utilizando o termo “default”

**CREATE com default ...**

```
CREATE TABLE EMPLOYEE (
    Fname           VARCHAR(15),
    ...
    Salary          DECIMAL(10,2)      DEFAULT 0,
    ...
    Dno             INT);
```

21

21

## Restrições de Integridade

- **check (P)**
  - impor uma regra a um atributo
- **not null**
  - atributo não pode ser null
- **primary key (A<sub>1</sub>, ..., A<sub>n</sub>)**
  - definir chave primária
- **unique (A<sub>1</sub>, ..., A<sub>n</sub>)**
  - chaves candidatas não primárias
- **foreign key**
  - definir chave estrangeira

As restrições podem ser de:

- **coluna** - referem-se a apenas uma coluna e são descritas em frente à coluna
- **tabela** - referem-se a mais do que a uma coluna e ficam separadas da definição das colunas

22

22

 deti

## Restrição CHECK

**Restrição CHECK na coluna...**

```
CREATE TABLE EMPLOYEE (
    ...
    Salary           DECIMAL(10,2)      CHECK (Salary > 12),
    ...);
```

**Restrição CHECK na tabela...**

```
CREATE TABLE DEPARTMENT (
    ...
    Dept_create_date   DATE            NOT NULL,
    Mgr_start_date     DATE,
    ...
    CHECK (Dept_create_date <= Mgr_start_date);
```

Restrição aplicada a cada atributo referenciado sempre que um tuplo é introduzido ou modificado.

23

23

 deti

## Restrição PRIMARY KEY

- Só podemos definir uma chave primária na tabela.
  - Por definição, a chave primária não pode conter valores repetidos ou nulos.

**Restrição PRIMARY KEY na coluna...**

```
CREATE TABLE EMPLOYEE (
    ...
    Ssn             CHAR(9)          PRIMARY KEY,
    ...);
```

**Restrição PRIMARY KEY na tabela...**  
(obrigatório se PK for composta por mais do que um atributo)

```
CREATE TABLE EMPLOYEE (
    ...
    Ssn             CHAR(9),
    ...
    PRIMARY KEY (Ssn));
```

24

## Restrição UNIQUE

- Utilizada para as chaves candidatas alternativas.
  - Não pode conter valores repetidos mas pode ter valores null.

### Restrição UNIQUE na coluna...

```
CREATE TABLE DEPARTMENT (
    Dname          VARCHAR(15)  UNIQUE NOT NULL,
    Dnumber        INT          NOT NULL,
    PRIMARY KEY (Dnumber),
    ... );
```

### Restrição UNIQUE na tabela...

```
CREATE TABLE DEPARTMENT (
    Dname          VARCHAR(15)      NOT NULL,
    Dnumber        INT              NOT NULL,
    PRIMARY KEY (Dnumber),
    UNIQUE (Dname), ... );
```

25

25

## Restrição FOREIGN KEY

- Utilizada para declarar chaves estrangeiras.
- Uma chave estrangeira deve referenciar uma chave primária ou única.

### Restrição FOREIGN KEY na coluna...

```
CREATE TABLE EMPLOYEE (
    ...
    Super_ssn   CHAR(9)   REFERENCES EMPLOYEE(Ssn),
    Dno         INT       REFERENCES DEPARTMENT(Dnumber) NOT NULL,
    ...);
```

### Restrição FOREIGN KEY na tabela...

```
CREATE TABLE EMPLOYEE (
    ...
    Ssn          CHAR(9),
    Dno          INT          NOT NULL,
    ...
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
```

26

26

## Restrição FOREIGN KEY

### Integridade Referencial

- Pode haver uma violação quando são inseridos ou eliminados tuplos ou quando os atributos chave estrangeira ou primária são modificados, resultando numa rejeição da operação.
- Podemos definir as seguintes ações alternativas: “[on delete](#)” e “[on update](#)”, com as seguintes opções:
  - restrict - não deixa efetuar a operação
  - cascade - apaga os registos associados (delete) ou altera a chave estrangeira (update)
  - set null - a chave estrangeira passa a null.
  - set default - a chave estrangeira passa a ter o valor por <sup>27</sup> omissão.

27

## Restrição FOREIGN KEY

### Integridade Referencial

#### Restrição FOREIGN KEY

```
CREATE TABLE EMPLOYEE (
  ...
  Ssn          CHAR(9),
  Dno          INT              NOT NULL,
  ...
  FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
    ON DELETE SET NULL  ON UPDATE CASCADE,
  FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
    ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Se o tuplo do supervisor é eliminado, a coluna Super\_ssn dos supervisionados passa automaticamente a Null.

Se o Ssn do supervisor é atualizado, a coluna Super\_ssn dos supervisionados é atualizada em cascata.

28

28

## Restrições - atribuição de nome

- Imaginando que queremos alterar uma restrição de uma tabela... Como referenciá-la?
- Nestas situações temos de “baptizar” a restrição com um nome próprio.

### Restrições com nome...

```
CREATE TABLE EMPLOYEE (
    ...
    ...
    CONSTRAINT EMPPK
        PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
            ON DELETE SET NULL ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
            ON DELETE SET DEFAULT ON UPDATE CASCADE);
    )
```

29

## Tabela - Drop

- O comando **drop table** remove da base de dados toda a informação sobre a tabela e os dados (tuplos).

### Eliminar a tabela EMPLOYEE

```
DROP TABLE EMPLOYEE;
```

- Caso haja violação de restrições de integridade referencial, a operação é rejeitada.
- No entanto, a opção **CASCADE\*** permite eliminar a tabela e os elementos referenciados na restrição.

### Eliminar a tabela EMPLOYEE com opção CASCADE

```
DROP TABLE EMPLOYEE CASCADE;
```

\* Não está disponível em SQL Server. Solução: eliminar primeiro o constraint.

30

30

## Tabela - Alter

- O comando **alter table** é utilizado para modificar o esquema da tabela ou restrições existentes.
- Adicionar atributos à tabela:

```
ALTER TABLE tablename ADD Attribute Domain
```

```
ALTER TABLE EMPLOYEE ADD nofiscal INT;
```

- Todos os tuplos existentes ficam com valor null no novo atributo.

- Adicionar restrições à tabela:

```
ALTER TABLE tablename ADD CONSTRAINT name theconstraint
```

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT salarymin CHECK (Salary >475);
```

31

## Tabela - Alter

- Eliminar atributos da tabela:

```
ALTER TABLE tablename DROP COLUMN attributename
```

```
ALTER TABLE EMPLOYEE DROP COLUMN nofiscal;
```

- Eliminar restrições da tabela:

```
ALTER TABLE tablename DROP CONSTRAINT name
```

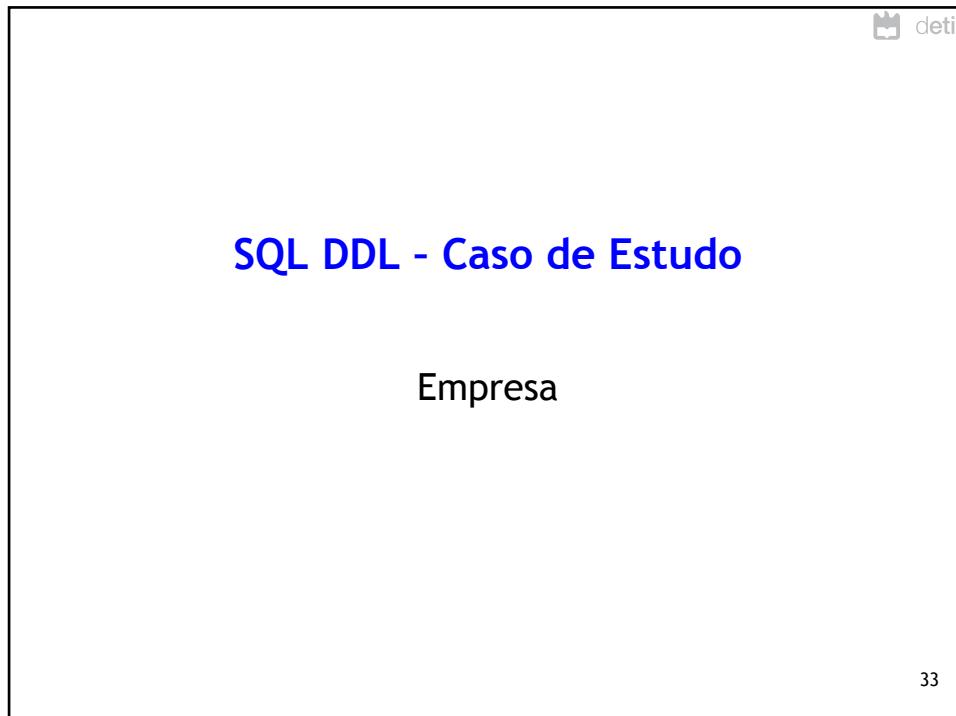
```
ALTER TABLE EMPLOYEE DROP CONSTRAINT salarymin;
```

- Alterar um atributo de uma tabela:

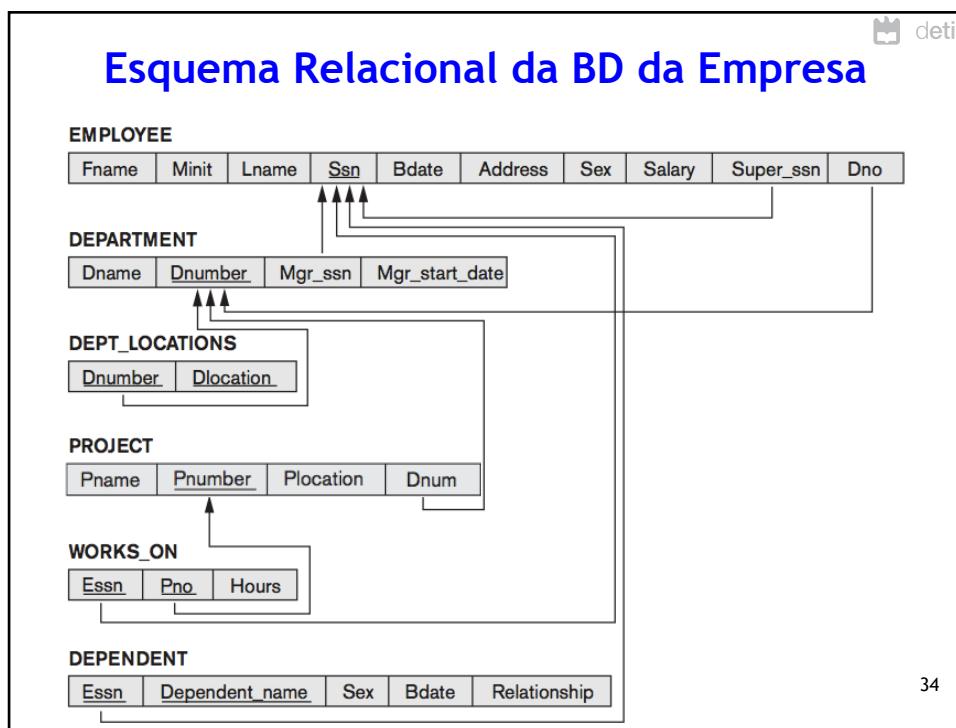
```
ALTER TABLE tablename ALTER Attribute Domain
```

```
ALTER TABLE EMPLOYEE ALTER COLUMN noFiscal CHAR(9);
```

32



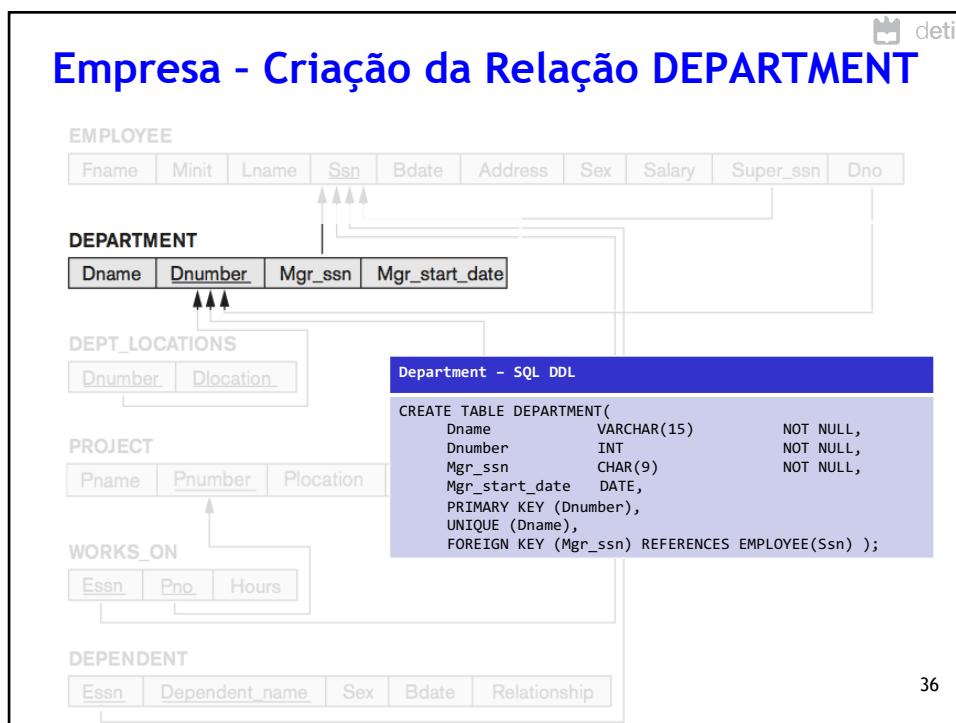
33



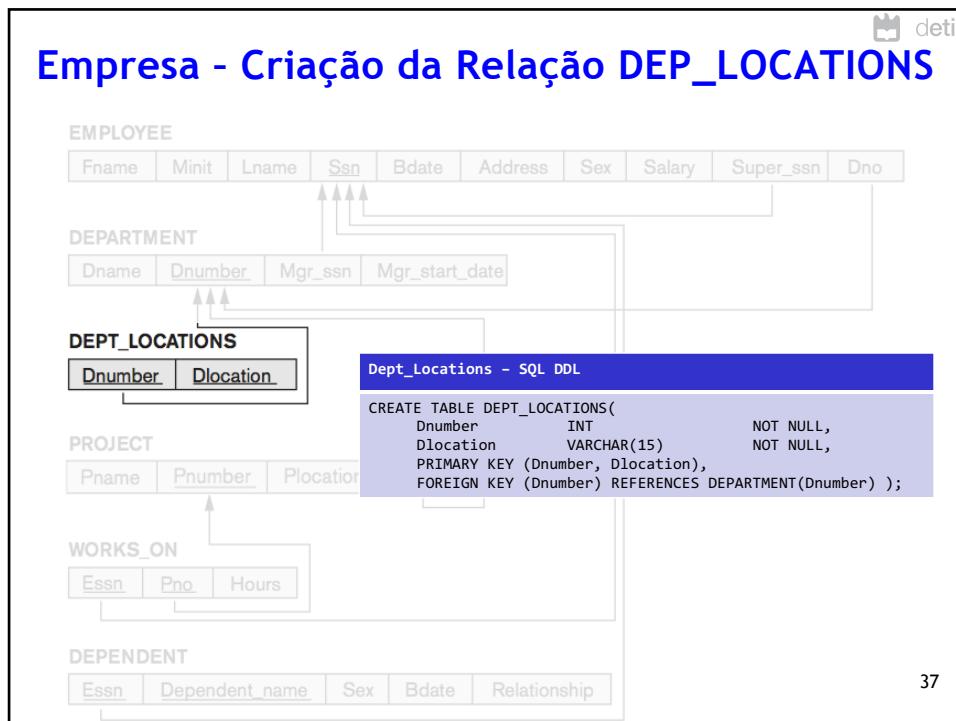
34



35



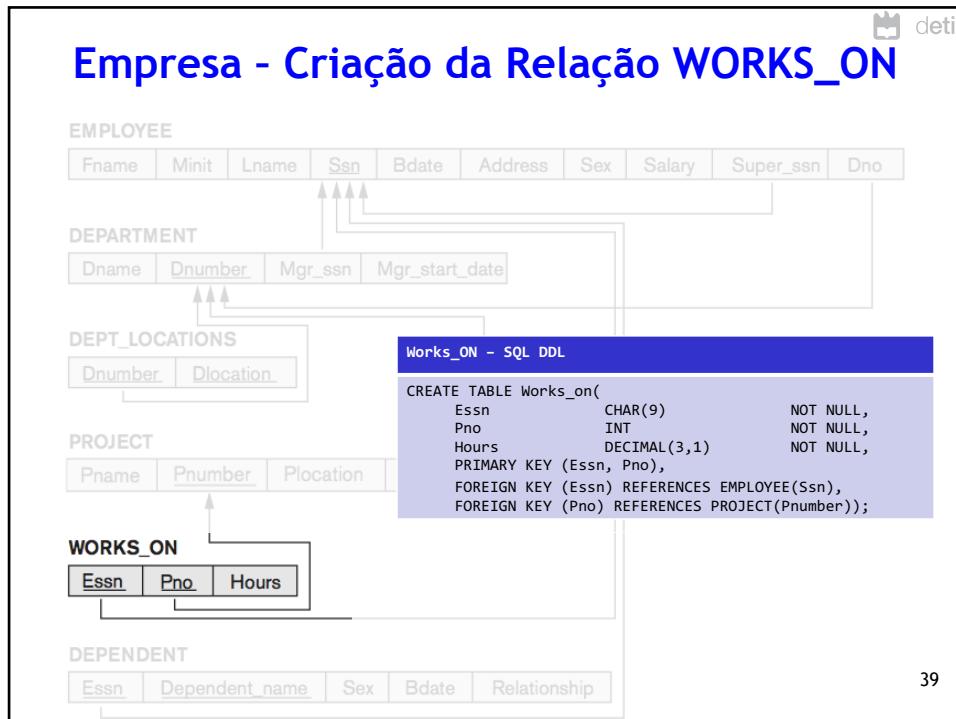
36



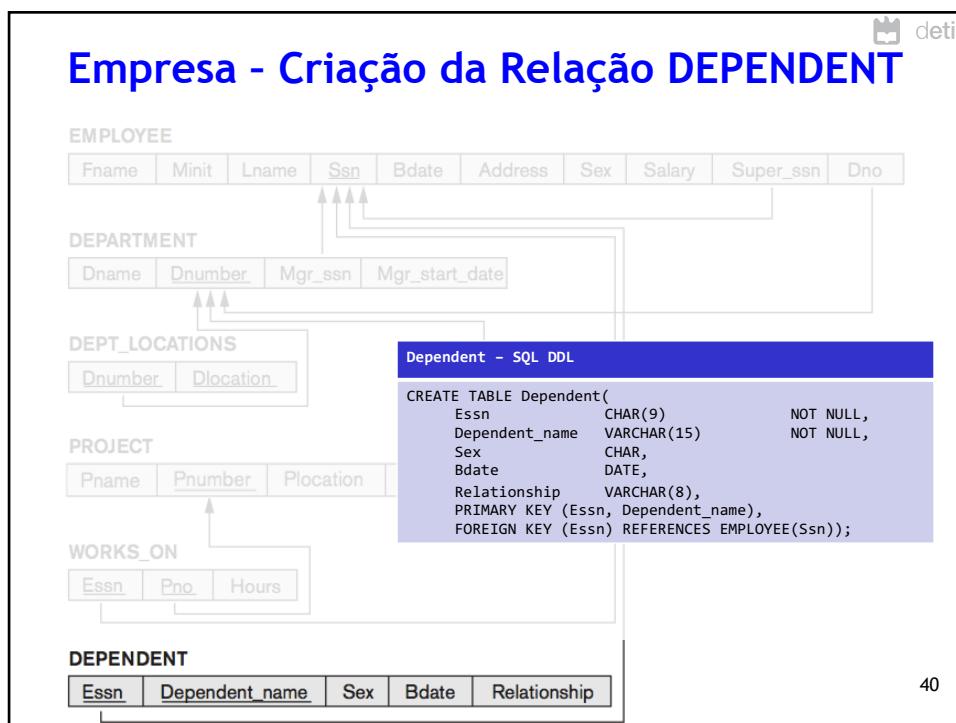
37



38



39



40

**deti**

## Empresa DDL - Considerações Práticas

**EXEMPLO: Employee, Department and Foreign Keys**

```

CREATE TABLE EMPLOYEE (
    Ssn           CHAR(9)          NOT NULL,
    Super_ssn    CHAR(9),
    Dno          INT              NOT NULL,
    ...
    PRIMARY KEY (Ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn));

CREATE TABLE DEPARTMENT(
    Dnumber      INT              NOT NULL,
    ...
    PRIMARY KEY (Dnumber),
    ...);

ALTER TABLE EMPLOYEE
    ADD CONSTRAINT EMPDEPTFK FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber);

ALTER TABLE DEPARTMENT
    ADD CONSTRAINT DEPTMGRFK FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn);

```

- Na prática só podemos criar restrições de integridade referencial, com recurso a chaves estrangeiras, quando temos as duas relações criadas.
- Assim, devemos começar por criar cada uma das relações (tabelas) e só depois definir as restrições.
  - Ou pelo menos uma delas...

41

41

**deti**

## SQL Server - Database Diagram

```

graph TD
    employee[employee] <--> works_on[works_on]
    employee <--> department[department]
    employee <--> dept_locations[dept_locations]
    project[project] <--> works_on
    project <--> department
    works_on <--> dept_locations

```

The diagram illustrates the relationships between five tables in a database:

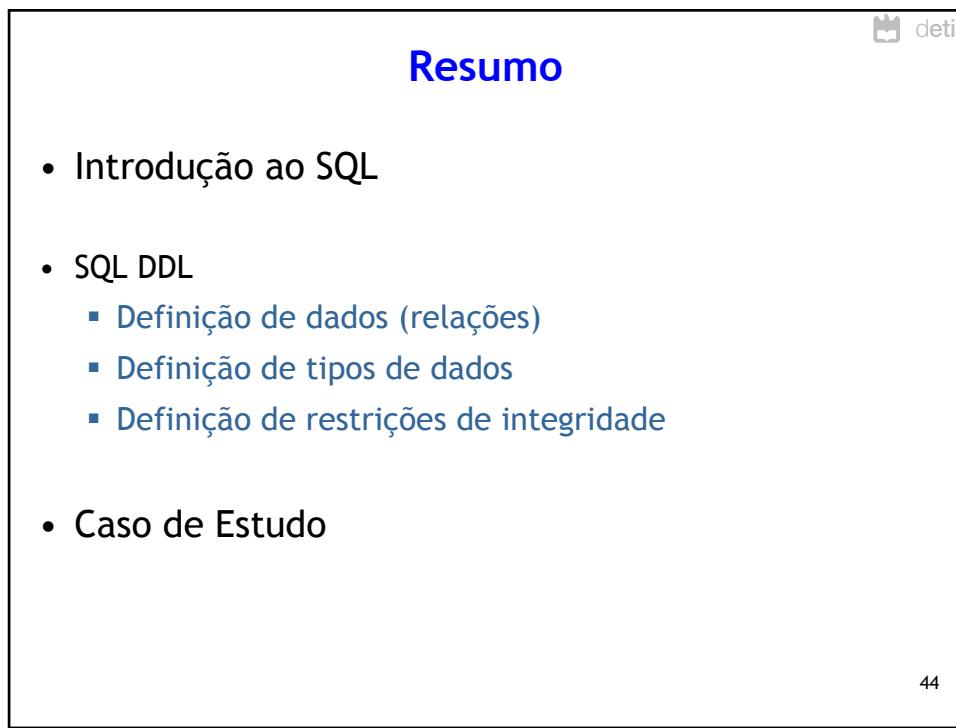
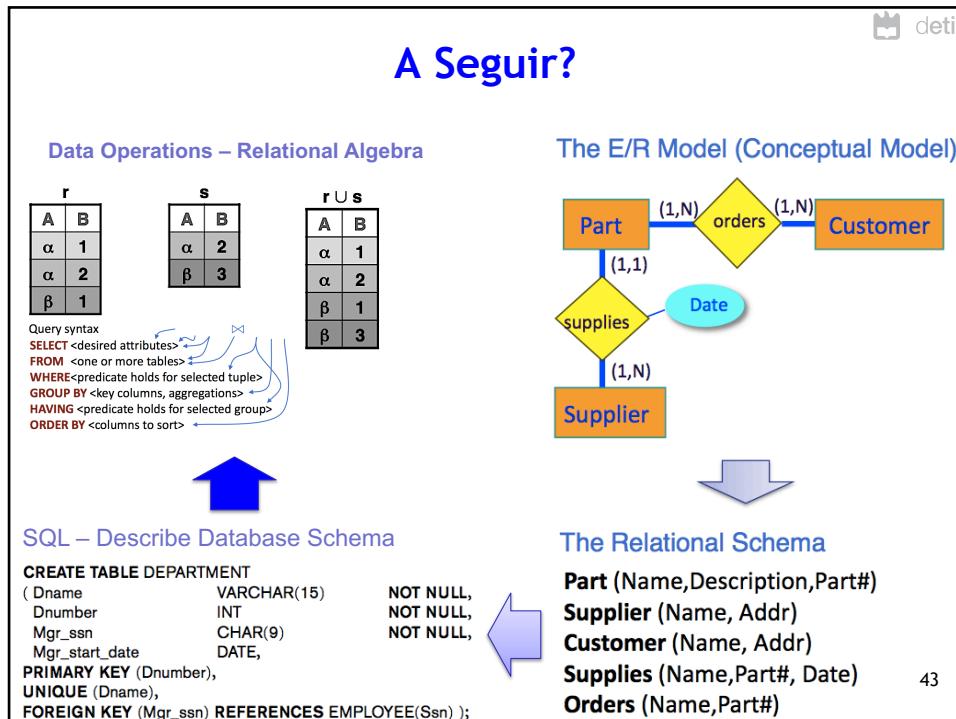
- employee**: Contains columns Name, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super\_ssn, and Dno.
- project**: Contains columns Pname, Pnumber, Plocation, and Dnum.
- works\_on**: Contains columns Essn, Pno, and Hours.
- department**: Contains columns Dname, Dnumber, Mgr\_ssn, and Mgr\_start\_date.
- dept\_locations**: Contains columns Dnumber and Dlocation.

Relationships are defined by foreign key constraints:

- Employee** has a self-referencing relationship via the **SUPER\_SSN** column.
- Employee** has three relationships:
  - A many-to-many relationship with **works\_on** via the **SSN** column.
  - A many-to-many relationship with **department** via the **DNO** column.
  - A many-to-many relationship with **dept\_locations** via the **DNO** column.
- Project** has a many-to-many relationship with **works\_on** via the **PNUMBER** column.
- Project** has a many-to-many relationship with **department** via the **DNUM** column.
- works\_on** has a many-to-many relationship with **dept\_locations** via the **DNUMBER** column.

42

42



44

44

## Álgebra Relacional

Base de Dados - 2020/21

Carlos Costa

1

## Introdução

### Linguagem de Consulta/Interrogação de BD

- Álgebra Relacional
  - Linguagem formal do Modelo Relacional
  - Um conjunto básico de operações
- Outras linguagem formais: *relational calculus*
- As linguagens formais oferecem uma base teórica para a linguagem de consulta utilizada na prática.
- Linguagem prática do Modelo Relacional
  - SQL

2

2

## Álgebra Relacional

### Questões?

- Como deve ser uma linguagem de interrogação da BD?
- Que tipo de interrogações existem?
- Como é que são os resultados?
- Expressões de álgebra relacional (linguagem).
  - Sequência de operações de álgebra relacional.
  - Permitem formular pedidos básicos de recuperação de informação sobre uma ou mais relações.
- Formulação da interrogação:
  - conjunto de operadores que operam sobre as relações
  - devolvem uma nova relação
- Vamos estudar um conjunto de operações...

3

3

## Álgebra Relacional - Operações Básicas

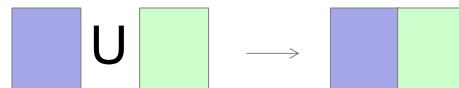
- Seleção



- Projeção



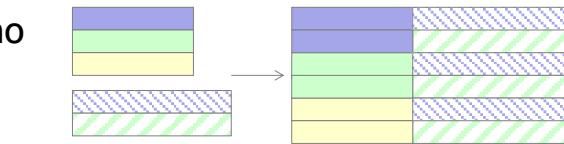
- União



- Diferença



- Produto Cartesiano



- Renomeação

4

## Seleção



- Notação:  $\sigma_{<\text{selection condition}>}(R)$ 
    - Utilizada para selecionar um subconjunto de tuplos da relação ( $t \in R$ ) que satisfazem os critérios de seleção.
    - “selection condition” é uma expressão booleana.
- $\text{Relation2} \leftarrow \sigma_{<\text{selection condition}>}(\text{Relation1})$
- O resultado é uma nova relação (Relation2) que tem um esquema relacional igual à original (Relation1).

5

5

## Seleção - Predicado

- Operadores de Comparação
  - Permitem comparar dois atributos ou um atributo com um valor.
  - Operandos: Nomes dos atributos e constantes.
  - Operadores:  $=, \neq, \leq, \geq, <, >$
  - Exemplos:
    - $\sigma_{Dno=4}(\text{EMPLOYEE})$
    - $\sigma_{Salary>30000}(\text{EMPLOYEE})$
- Condições Booleanas
  - Utilização de AND, OR e NOT.
  - Exemplo:
    - $\sigma_{(Dno=4 \text{ AND } Salary>25000) \text{ OR } (Dno=5 \text{ AND } Salary>30000)}(\text{EMPLOYEE})$

6

6

deti

## Seleção - Exemplo

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

**SQL query (próxima aula...)**

$\sigma_{(Dno=4 \text{ AND } Salary > 25000) \text{ OR } (Dno=5 \text{ AND } Salary > 30000)}(\text{EMPLOYEE})$

`SELECT * FROM EMPLOYEE  
WHERE Dno=4 AND Salary>25000  
OR Dno=5 AND Salary>30000;`

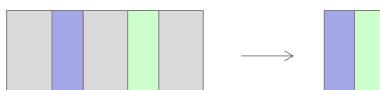
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

7

7

deti

## Projeção



- Notação:  $\Pi_{<\text{attribute list}>}(\text{R})$ 
  - $<\text{attribute list}> = A_1, A_2, \dots, A_k$
  - $A_1 \dots A_k$  são nomes dos atributos da relação R
- O resultado é uma nova relação só com os k atributos selecionados.
- São removidas as linhas duplicadas do resultado.
  - Condição de conjunto (set)

8

8

**Projeção - Exemplo**

**EMPLOYEE**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

$\Pi_{\text{Lname}, \text{Fname}, \text{Salary}}(\text{EMPLOYEE})$

**SQL query:**

```
SELECT DISTINCT Lname, Fname, Salary
FROM EMPLOYEE;
```

**Result:**

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

9

9

**Encadeamento de Operações**

- $\Pi_{\text{Fname}, \text{Lname}, \text{Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$
- Se quisermos renomear os atributos e a relação:  
 $\text{TEMP} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$   
 $\text{R}(\text{First\_name}, \text{Last\_name}, \text{Salary}) \leftarrow \Pi_{\text{Fname}, \text{Lname}, \text{Salary}}(\text{TEMP})$

**TEMP**

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

**R**

First_name	Last_name	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

10

10

## Renomeação

- Notação:  $\rho_{R_2(B_1, B_2, \dots, B_n)}(R_1)$  ou  $\rho_{R_2}(R_1)$   
ou  $\rho_{(B_1, B_2, \dots, B_n)}(R_1)$

- No primeiro caso o resultado é uma nova relação R2 com os atributos renomeados (B1, B2, ..., Bn).
- No segundo caso só renomeamos a relação.
- No terceiro só renomeamos os atributos.

### SQL query:

```
SELECT E.Fname AS First_name, E.Lname AS Last_name, E.Salary AS Salary
FROM EMPLOYEE AS E
WHERE E.Dno=5;
```

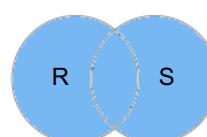
R1: EMPLOYEE  
R2: E  
Fname -> First\_name  
Lname -> Last\_Name  
...

11

Seleção

11

## União



- Notação:  $R \cup S = \{t : t \in R \vee t \in S\}$
- As tabelas têm de ser compatíveis
  - Mesmo número de atributos
  - Atributos com domínios compatíveis
- O resultado é uma relação que inclui todos os tuplos de R e de S
  - Os tuplos duplicados são eliminados

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

U

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

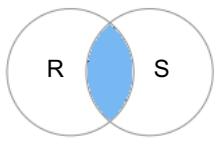


Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

12

12

## Intersecção



- Notação:  $R \cap S = \{t : t \in R \wedge t \in S\}$
- As tabelas têm de ser compatíveis
  - Mesmo número de atributos
  - Atributos com domínios compatíveis
- O resultado é uma relação que inclui os tuplos que existem simultaneamente em R e S
  - Os tuplos duplicados são eliminados

STUDENT	
Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

$\cap$

INSTRUCTOR	
Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

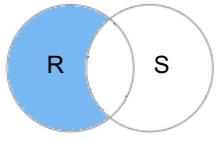


Fn	Ln
Susan	Yao
Ramesh	Shah

13

13

## Diferença



- Notação:  $R - S = \{t : t \in R \wedge t \notin S\}$
- As tabelas têm de ser compatíveis
  - Mesmo número de atributos
  - Atributos com domínios compatíveis
- O resultado é uma relação que inclui os tuplos de R que não existem em S

STUDENT	
Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

-

INSTRUCTOR	
Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah



Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

14

14

## União, Intersecção e Diferença

- Em SQL existem os seguintes comandos
  - UNION (ALL), INTERSECT (ALL) e EXCEPT (ALL)

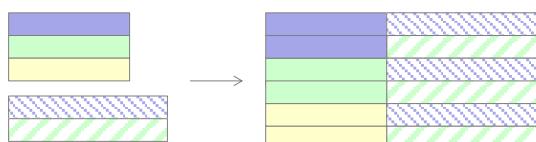
Propriedades:

- União e Intersecção são operações comutativas:
  - $R \cup S = S \cup R$  e  $R \cap S = S \cap R$
- A diferença não é comutativa:
  - $R - S \neq S - R$
- União e Intersecção são operações associativas:
  - $R \cup (S \cup T) = (R \cup S) \cup T$  e  $(R \cap S) \cap T = R \cap (S \cap T)$

15

15

## Produto Cartesiano



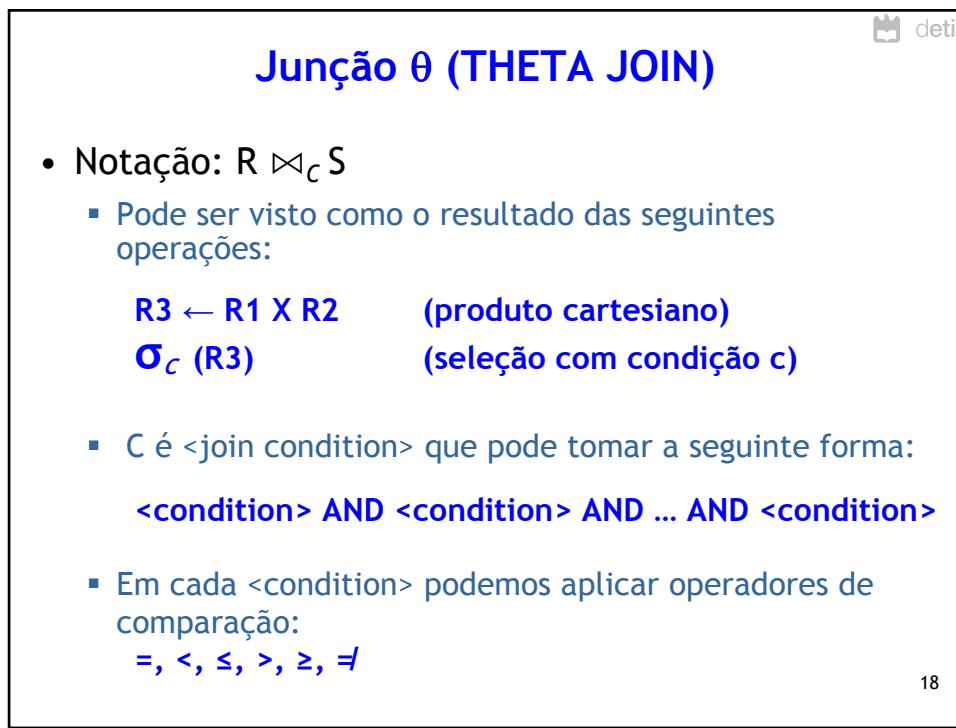
- Notação:  $R \times S$
- Permite-nos combinar tuplos de relações diferentes.
  - O resultado é uma nova relação (Q) que combina cada elemento (tuplo) de uma relação (R) com um elemento (tuplo) da outra relação (S):
 
$$Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m) = R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$$
  - O número de tuplos de Q é  $n * m$ .
- UK: “CROSS JOIN”

16

16



17



18

**Junção θ (THETA JOIN) - Exemplo**

deti

- Pretendemos saber os nomes dos funcionários gestores de departamentos

EMPLOYEE									
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1985-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1985-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1988-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Balaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1982-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT			
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

Para obter o nome dos gestores temos de combinar cada tuplo do departamento (Department) com um tuplo dos funcionários (Employee) cujo Ssn é igual ao Mgr\_ssn.

$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE}$

DEPT_MGR								
Dname	Dnumber	Mgr_ssn	...	Fname	Minit	Lname	Ssn	...
Research	5	333445555	...	Franklin	T	Wong	333445555	...
Administration	4	987654321	...	Jennifer	S	Wallace	987654321	...
Headquarters	1	888665555	...	James	E	Borg	888665555	...

Depois só temos de utilizar projeção para obter os atributos desejados:

$\text{RESULT} \leftarrow \pi_{\text{Dname}, \text{Lname}, \text{Fname}}(\text{DEPT\_MGR})$

19

**Junção - Variações da Junção θ**

deti

- Equi-Junção (EquiJoin)**
  - É utilizado o operador = na condição de junção.
  - Exemplo anterior:  $\text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE}$ .
  - Vamos ter sempre duas colunas repetidas.
- Junção Natural (Natural Join):  $R \bowtie S$** 
  - Condição implícita: igualdade dos atributos com o mesmo nome.
  - Os atributos repetidos são removidos.
  - Nota: Muitas vezes opta-se por renomear colunas de modo a facilitar junções naturais.

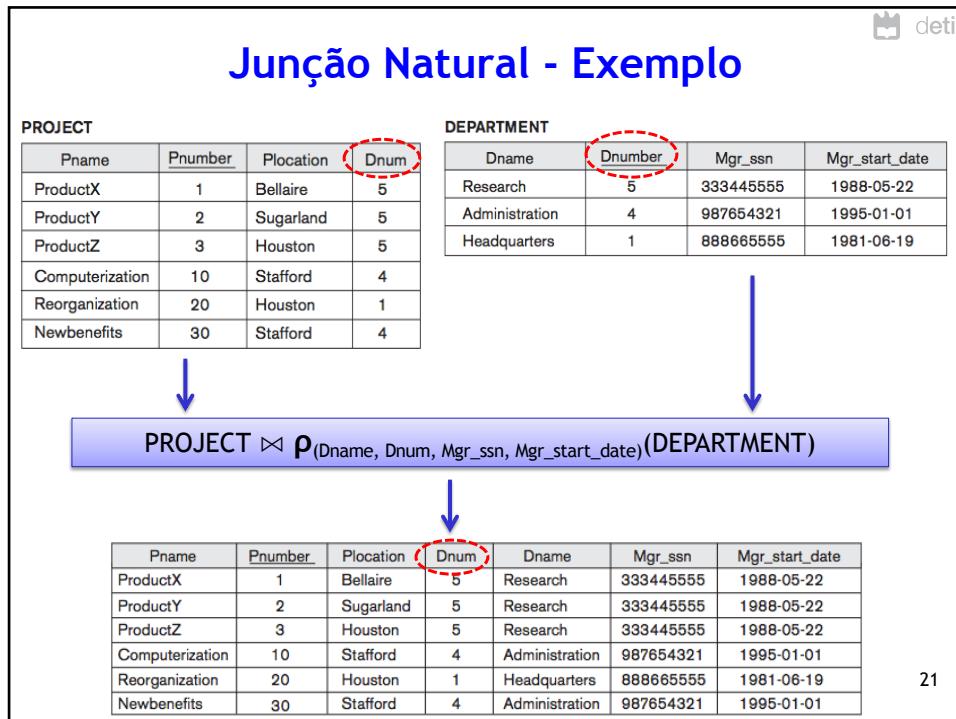
R		S				
X	Y	Y	Z			
a	c	d	g			
b	d	e	h			

$\bowtie \longrightarrow$

X	Y	Z
b	d	g

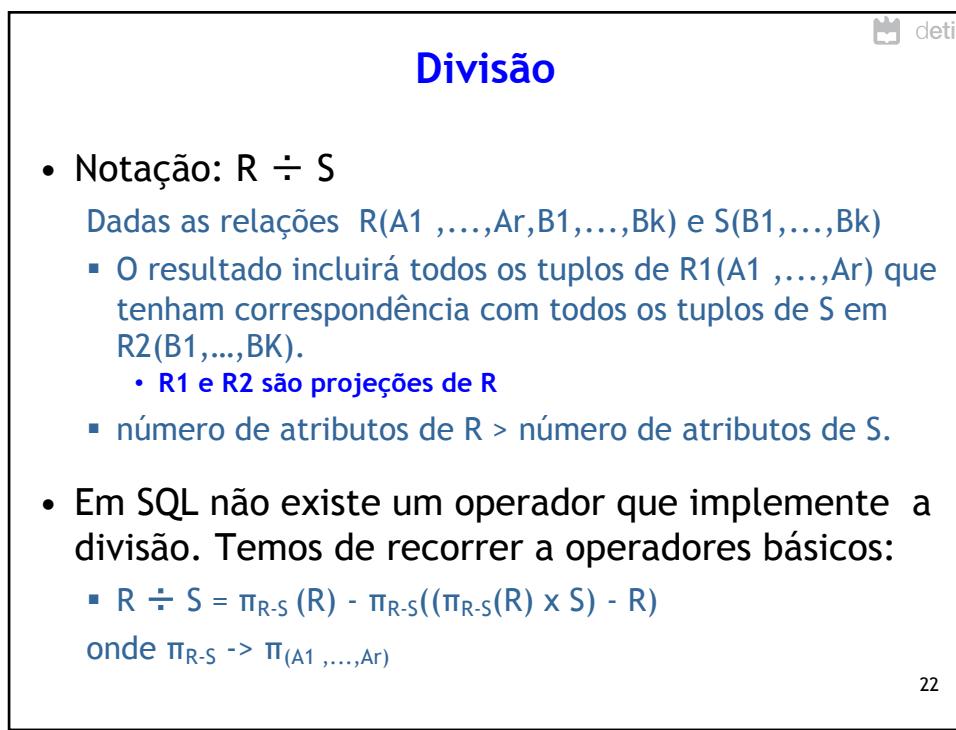
20

20



21

21



22

22

**Divisão - Exemplos**

$$R \div S = T$$

Department

Dno	Name	Location
1	Research	Houston
2	Commercial	Bellaire
3	Administration	LA
2	Commercial	Houston
4	Headquarters	Bellaire
2	Commercial	LA

÷

Location

Location
Houston
Bellaire
LA

→

Dno	Name
2	Commercial

**Departamentos que existem em todas as localizações?**

23

23

**Operações Álgebra Relacional - Resumo**

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation $R$ .	$\sigma_{<\text{selection condition}>}(R)$
PROJECT	Produces a new relation with only some of the attributes of $R$ , and removes duplicate tuples.	$\pi_{<\text{attribute list}>}(R)$
THETA JOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.	$R_1 \bowtie_{<\text{join condition}>} R_2$
EQUIJOIN	Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{<\text{join condition}>} R_2$ , OR $R_1 \bowtie_{(<\text{join attributes 1}>), (<\text{join attributes 2}>)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{<\text{join conditions}>} R_2$ , OR $R_1 *_{(<\text{join attributes 1}>), (<\text{join attributes 2}>)} R_2$ OR $R_1 *_{R_2} R_2$
UNION	Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of $R_1$ and $R_2$ , and includes tuples all possible combinations of tuples from $R_1$ and $R_2$ .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $R_1$ in combination with every tuple from $R_2(Y)$ , where $Z = X \cup Y$ .	$R_1(Z) \div R_2(Y)$

24

24

## Álgebra Relacional - Operações Estendidas

- Semi-Join (Semi Junção)
  - Left Semi Join
  - Right Semi Join
- Outer Join (Junção Externa)
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
- Agregação
  - Funções de Agregação

25

25

## Semi Join

- Left Semi Join:  $R \ltimes S = \Pi_R(R \bowtie S)$

Projeção dos atributos de R na junção natural de R com S

R	S	
x	y	
a	c	→
b	d	

S		
y	z	
d	g	→
e	h	

- Right Semi Join:  $R \rtimes S = \Pi_S(R \bowtie S)$

Projeção dos atributos de S na junção natural de R com S

R	S	
x	y	
a	c	→
b	d	

S		
y	z	
d	g	→
e	h	

26

26

## Inner Join vs Outer Join

### Inner Join

- As operações de junção anteriores combinam dados de duas tabelas para que estes possam ser apresentados na forma de uma única tabela.
- Os tuplos que não estão relacionados (*matching*) são descartados.
  - Incluindo os tuplos com valores Null nos atributos de junção.

### Outer Join

- Incluímos no resultado todos os tuplos de uma (ou de ambas) das relações componentes.
- Os atributos que não fazem *matching* são preenchidos com *Null*.

27

27

## Outer Join

- Left Outer Join:  $R \bowtie S$

R		S			Result			
A1	A2	B1	B2	$\bowtie_{A2=B1}$	A1	A2	B1	B2
a	c	d	g		a	c	null	null
b	d	e	h		b	d	d	g

- Right Outer Join:  $R \bowtie S$

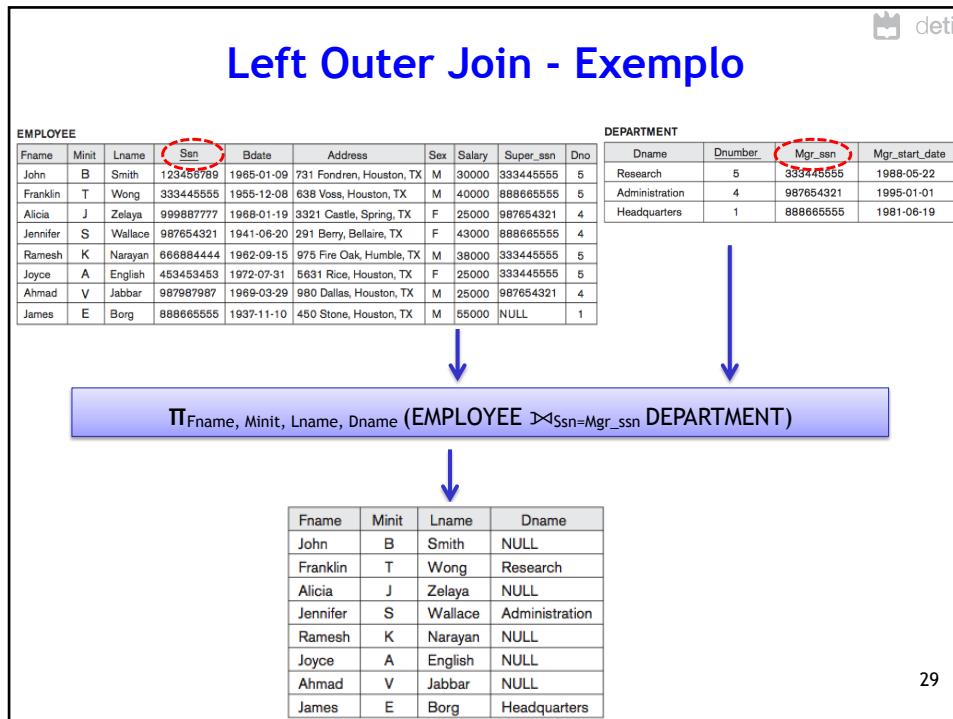
R		S			Result			
A1	A2	B1	B2	$\bowtie_{A2=B1}$	A1	A2	B1	B2
a	c	d	g		b	d	d	g
b	d	e	h		null	null	e	h

- Full Outer Join:  $R \bowtie S$

R		S			Result			
A1	A2	B1	B2	$\bowtie_{A2=B1}$	A1	A2	B1	B2
a	c	d	g		a	c	null	null
b	d	e	h		b	d	d	g

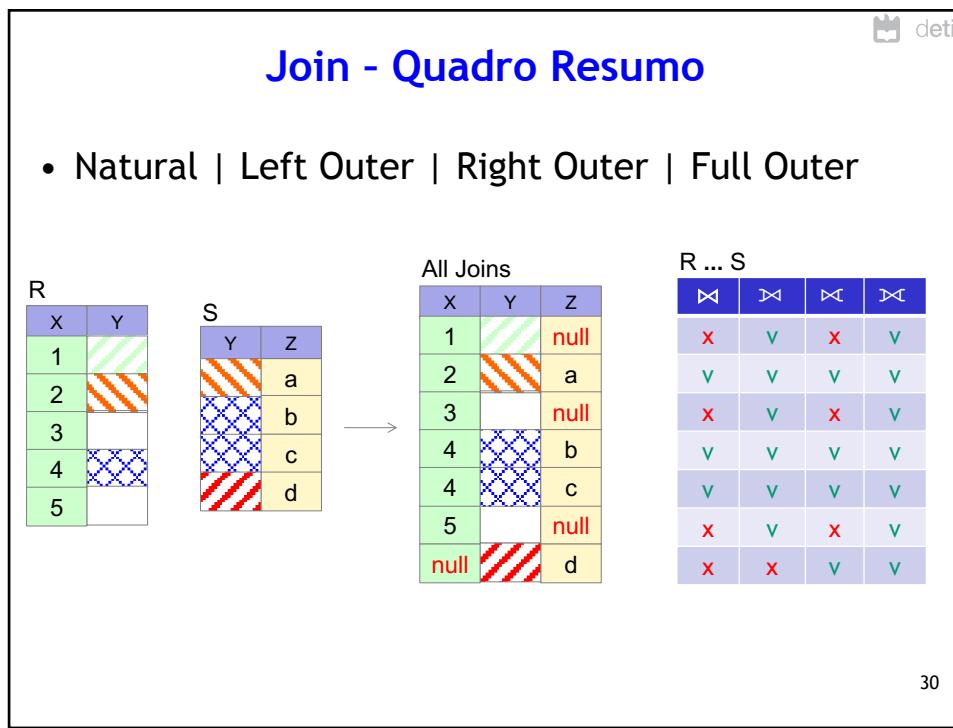
28

28



29

29



30

30

**Agregação**

- Operação de Agregação  
 $\langle \text{grouping attributes} \rangle \Sigma \langle \text{function list} \rangle (R)$ 

$\Sigma$  - Script F symbol
- Operações sobre vários tuplos da relação
- Lista de Funções de Agregação:
  - avg: média dos valores
  - min: mínimo dos valores
  - max: máximo dos valores
  - sum: soma dos valores
  - count: número dos valores

31

31

**Funções de Agregação**

- Também podem ser usadas em projeções
  - criar atributos agregados
  - os atributos não agregados são agrupados de forma a não haver valores repetidos.
- Exemplos:

$\Pi_{A1, A2, M = \text{avg}(A3)} (R)$

EMPLOYEE										
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4	
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5	
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5	
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	

$\downarrow$

$\Pi_{Dno, Avg\_Salary=\text{avg}(Salary)}(\text{EMPLOYEE})$

Dno	Avg_Salary
1	55000
4	31000
5	33250

32

32

deti

## Agregação (*Grouping*) - Exemplos

**EMPLOYEE**

Name	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

$\exists \text{ count}(Ssn), \text{ avg}(Salary)(\text{EMPLOYEE})$

Count_ssn	Average_salary
8	35125

$Dno \exists \text{ count}(Ssn), \text{ avg}(Salary)(\text{EMPLOYEE})$

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

$\rho_{R(Dno, \text{No\_of\_employees}, \text{Average\_sal})} (Dno \exists \text{ count}(Ssn), \text{ avg}(Salary)(\text{EMPLOYEE}))$

R		
Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

33

33

deti

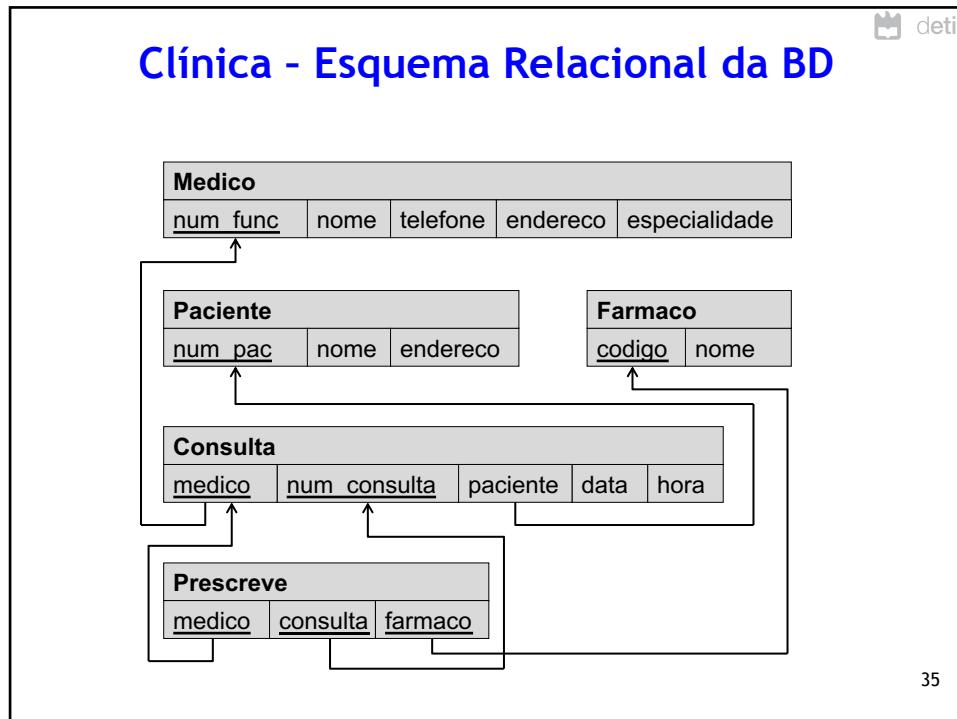
## Álgebra Relacional - Queries Caso de Estudo

### Clínica Médica

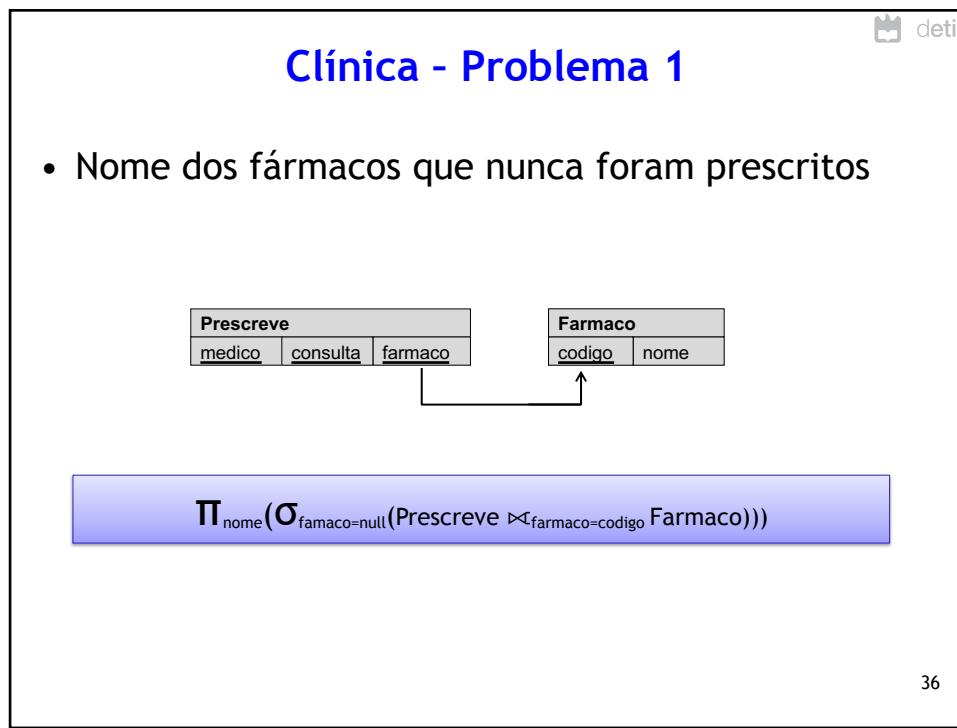
34

34

17



35



36

36

**Clínica - Problema 2**

• O número de fármacos prescritos em cada consulta

Prescreve			Farmaco	
medico	consulta	farmaco	codigo	nome

↑

$\Pi_{medico, consulta, num\_farm=count(farmaco)} (\text{Prescreve})$

**Ou**

$\text{medico, consulta } \Sigma_{count(farmaco)} (\text{Prescreve})$

37

37

**Clínica - Problema 3**

• Para cada médico, a quantidade média de fármacos receitados por consulta

Prescreve			Farmaco	
medico	consulta	farmaco	codigo	nome

↑

$\text{temp} \leftarrow \Pi_{medico, consulta, num\_farm=count(farmaco)} (\text{Prescreve})$

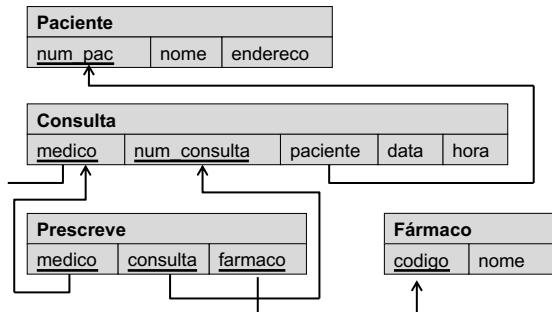
$\Pi_{medico, avg\_farmaco=avg(num\_farm)} (\text{temp})$

38

38

## Clínica - Problema 4

- O nome de todos os fármacos prescritos, incluindo a quantidade, para o paciente número 35312161



$\text{temp} \leftarrow \Pi_{\text{medico}, \text{num\_consulta}} (\sigma_{\text{paciente}=35312161} (\text{Consulta}))$

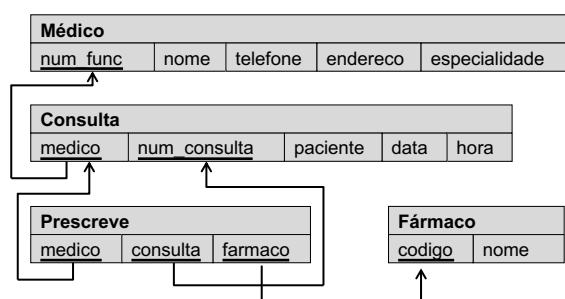
$\text{temp2} \leftarrow \Pi_{\text{farmaco}, \text{quantidade}=\text{count}(\text{farmaco})} (\text{temp} \bowtie_{\text{medico}=\text{medico} \text{ AND } \text{num\_consulta}=\text{consulta}} \text{Prescreve})$

$\Pi_{\text{nome}, \text{quantidade}} (\text{temp2} \bowtie_{\text{farmaco}=\text{codigo}} \text{Farmaco})$

39

## Clínica - Problema 5

- O nome dos fármacos que já foram prescritos por todos os médicos da clínica

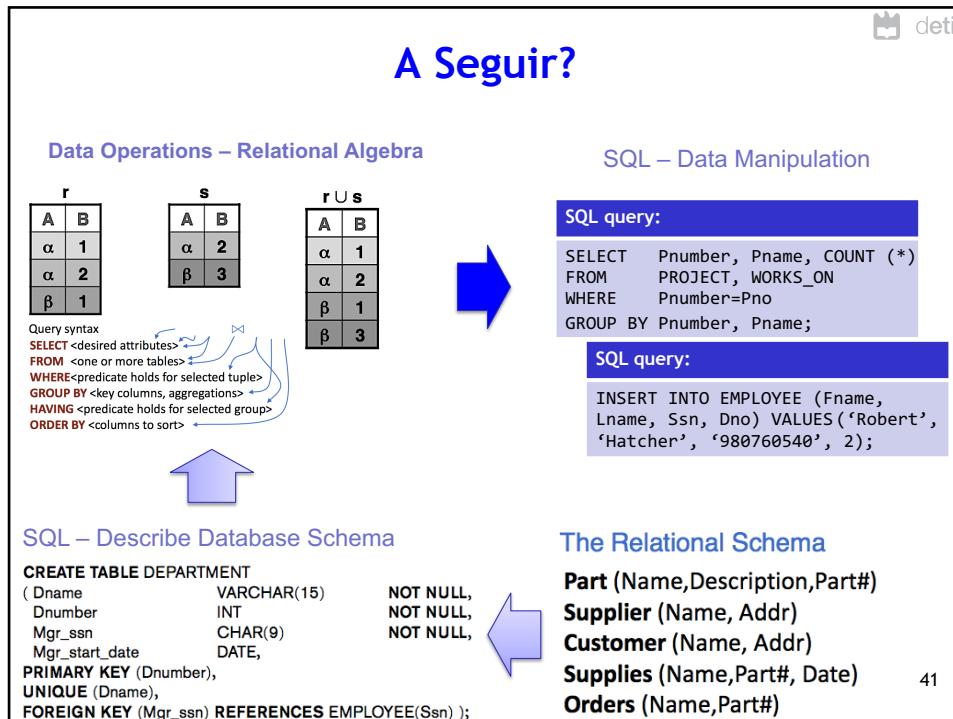


$\text{temp} \leftarrow (\Pi_{\text{farmaco}, \text{medico}} (\text{Prescreve})) \div (\rho_{\text{medico}} (\Pi_{\text{num_func}} (\text{Médico})))$

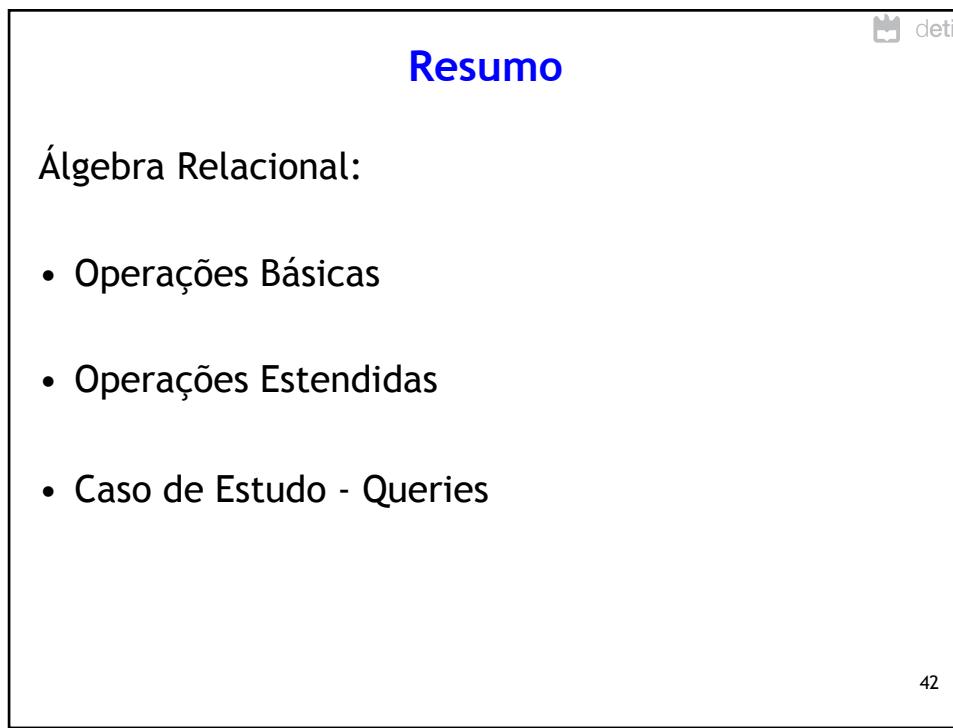
$\Pi_{\text{nome}} (\rho_{\text{codigo}, \text{medico}} (\text{temp}) \bowtie \text{Farmaco})$

40

40



41



42

42

## Linguagem SQL - DML

Base de Dados - 2018/19

Carlos Costa

## SQL DML - Introdução

- DML - Data Manipulation Language
- Os comandos SQL DML permitem:
  - Inserir, eliminar e atualizar dados
  - Efetuar consultas:
    - Simples
    - Avançadas

## SQL DML

### INSERT, DELETE e UPDATE

3

## Inserção - INSERT INTO

- Utilizado para inserir um novo tuplo numa relação.
  - Sintaxe 1: Não se indicam as colunas, tendo os valores inseridos de respeitar a ordem de criação dos atributos. Podemos utilizar os termos NULL ou DEFAULT:

```
INSERT INTO tablename VALUES (v1,v2,...,vn);
INSERT INTO EMPLOYEE VALUES
  ('Richard', 'K', 'Marini', '653298653', NULL, '98
   Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);
```

- Sintaxe 2: Indicamos as colunas em que queremos inserir os dados. As restantes ficam com o seu valor nulo ou por defeito (caso tenha sido definido):

```
INSERT INTO tablename (A1,A4,A8,...,An) VALUES (v1,v4,v8,...,vn);
INSERT INTO EMPLOYEE (Dno, Fname, Lname, Ssn) VALUES
  (4, 'Richard', 'Marini', '653298653');
```

## Eliminação - DELETE

- Utilizado para remover um ou mais tuplos de uma relação.

```
DELETE FROM tablename WHERE match_condition;
```

-- remoção (potencial) de um tuplo:

```
DELETE FROM EMPLOYEE WHERE Ssn='123456789';
```

-- remoção (potencial) de n tuplos:

```
DELETE FROM EMPLOYEE WHERE Dno = 5;
```

-- ou

```
DELETE FROM EMPLOYEE WHERE Dno > 5 AND Dno < 8;
```

-- remoção de todos os tuplos da relação:

```
DELETE FROM EMPLOYEE;
```

Só afecta uma relação. No entanto, a ação pode propagar-se a outras relações devido às definições de integridade referencial (on delete cascade).

## Actualização - UPDATE

- Utilizado para atualizar um ou mais tuplos de uma relação.

```
UPDATE tablename SET A1=v1,...,An=vn WHERE match_condition;
```

-- atualiza um tuplo:

```
UPDATE PROJECT  
SET Plocation = 'Bellaire', Dnum = 5  
WHERE Pnumber=10;
```

-- atualização (potencial) de n tuplos:

```
UPDATE EMPLOYEE  
SET Salary = Salary * 1.1  
WHERE Dno = 5;
```

Só afecta uma relação. No entanto, a ação pode propagar-se a outras relações devido às definições de integridade referencial (on update cascade). 6

## SQL DML

### Consultas Simples

7

## Operações com Conjuntos

- A linguagem SQL é baseada em operações de conjuntos e de álgebra relacional.
- No entanto, existem particularidades:
  - modificações e extensões
- SQL define formas de lidar com tuplos duplicados
  - Especifica quantas cópias dos tuplos aparecem no resultado.
    - Existem comandos para eliminar duplicados
  - Versões Multiconjunto de operadores (AR)
    - i.e. as relações podem ser multiconjuntos

8

**Projeção - SELECT FROM**

- **SELECT FROM**
  - Permite selecionar um conjunto de atributos (colunas) de uma ou mais tabelas.

$\Pi_{\langle \text{attribute\_list} \rangle} (R1)$

-- Forma Básica:  
**SELECT <attribute\_list> FROM <table\_list>;**

**SELECT \* FROM EMPLOYEE;** -- Todas as colunas

**SELECT Fname, Ssn FROM EMPLOYEE;** -- Duas colunas

EMPLOYEE							
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	5
Alicia	J	Zelaya	999887777	1968-01-10	3321 Castle, Spring, TX	F	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	1

Fname	Ssn
John	123456789
Franklin	333445555
Alicia	999887777
Jennifer	987654321
Ramesh	666884444
Joyce	453453453
Ahmad	987987987
James	888665555

9

**SELECT ALL vs DISTINCT**

- Podemos selecionar todos os tuplos ou eliminar os duplicados.
  - Tendo em atenção que, ao selecionarmos só algumas colunas da tabela, o resultado pode não ser um conjunto (set) mas um multiconjunto.

-- Todos os tuplos (por defeito):  
**SELECT All <attribute\_list> FROM <table\_list>;**

-- Eliminar tuplos repetidos:  
**SELECT DISTINCT <attribute\_list> FROM <table\_list>;**

**SELECT ALL Salary FROM EMPLOYEE;**

**SELECT DISTINCT Salary FROM EMPLOYEE;**

Salary
30000
40000
25000
43000
38000
25000
25000
55000

Salary
30000
40000
25000
43000
38000
55000

10

DISTINCT não pode ser aplicado a cada atributo individualmente. Deve aparecer depois do SELECT e aplica-se ao tuplo.

**Seleção - WHERE**

- WHERE permite selecionar um subconjunto de tuplos da(s) tabela(s) de acordo com uma expressão condicional.

$$\Pi_{\text{attribute\_list}}(\sigma_{\text{condition}}(R1))$$

```
SELECT <attribute_list> FROM <table_list> WHERE <condition>;
```

```
SELECT Bdate, Address FROM EMPLOYEE
WHERE Fname='John' AND Minit='B' AND Lname='Smith';
```

A condição pode conter operadores de comparação ( $=$ ,  $<$ ,  $\leq$ ,  $>$ ,  $\geq$ ,  $\neq$ ) e ser composta usando AND, OR e NOT.

EMPLOYEE								
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Dno	
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	5	
Franklin	T	Wong	333445555	1955-12-08	838 Voss, Houston, TX	M	5	
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	4	
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	5	
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	5	
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	4	
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	1	

Bdate	Address
1965-01-09	731Fondren, Houston, TX

11

**Renomeação - Relação, Atributo e Aritmética**

- Podemos renomear:
  - relações e atributos;
  - resultado de uma operação aritmética.

-- Renomear

```
-- Renomear Tabela*
SELECT E.Fname, E.Ssn FROM EMPLOYEE AS E;  $\rho_{R2}(R1)$ 
```

ou

```
SELECT E.Fname AS Fn, E.Ssn AS Ssname FROM EMPLOYEE AS E;
```

-- Renomear Atributo

```
SELECT Dno AS DepNumber FROM EMPLOYEE;  $\rho_{B1,\dots,Bn}(R1)$   $\rho_{R2(B1,\dots,Bn)}(R1)$ 
```

-- Renomear Resultado de Operação Aritmética\*\*
SELECT Salary \* 0.35 AS SalaryTaxes FROM EMPLOYEE;

\* ver mais à frente a importância de renomear tabelas em operações de junção.  
\*\* qual o resultado de não renomear? Depende de SGBD. SQL Server não dá nome à coluna!!!

## Reunião, Intersecção e Diferença

- Requisitos:
  - as duas relações têm de ter o mesmo número de atributos.
  - o domínio de cada atributo deve ser compatível.
  
- Operadores SQL:
  - UNION, INTERSECT e EXCEPT
  - devem ser colocados entre duas queries.
  - tuplos duplicados são eliminados.
  
- Para manter os tuplos duplicados devemos utilizar as suas versões multiconjunto.
  - UNION ALL, EXCEPT ALL\* e INTERSECT ALL\*

$R1 \cup R2$

$R1 \cap R2$

$R1 - R2$

\* Não disponível em SQL SERVER

13

## UNION - Exemplo

- Quais os projetos (número) que têm um funcionário ou um gestor do departamento que controla o projeto com o último nome Smith?

```

SELECT FROM .....
UNION (ALL)
SELECT FROM .....
(SELECT DISTINCT Pnumber
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn AND Lname='Smith' )
UNION
(SELECT DISTINCT Pnumber
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Essn=Ssn AND Lname='Smith' );

```

14

**deti**

## Produto Cartesiano

- Podemos utilizar mais do que uma relação na instrução SELECT FROM.
- O resultado é o produto cartesiano dos dois conjuntos.

**R1 X R2 X .. X RN**

```
SELECT * FROM table1, table2, ..., tableN;
-- Exemplo de Produto Cartesiano
SELECT * FROM EMPLOYEE, DEPARTMENT;

-- Exemplo de Produto Cartesiano só com dois atributos
-- >> Pode ser visto com Prod. Cartesiano seguido de Projeção
SELECT Ssn, Dname FROM EMPLOYEE, DEPARTMENT;
```

15

**deti**

## Junção de Relações - WHERE

- O Produto Cartesiano tem pouco interesse prático...
- No entanto, a associação do operador WHERE permite a junção de relações.

```
SELECT <attribute_list> FROM <table_list> WHERE <join_condition>;
```

-- Exemplo de “*select-project-join query*”

```
SELECT Fname, Lname, Address
FROM   EMPLOYEE, DEPARTMENT
WHERE  Dname='Research' AND Dnumber=Dno;
```

**ANSI SQL 89**

Join Condition

Fname	Lname	Address
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

EMPLOYEE			
Fname	Minit	Lname	Ssn
John	B	Smith	123456789
Franklin	T	Wong	333445555
Alicia	J	Zelaya	989887777
Jennifer	S	Wallace	987654321
Ramesh	K	Narayan	666884444
Joyce	A	English	453453453
Ahmad	V	Jabbar	987987987
James	E	Borg	888665555

DEPARTMENT			
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

## Junção de 3 Relações - Exemplo

- Caso com três relações e duas *join conditions*:

```

/* Questão: Para cada projeto localizado em 'Stafford', queremos
saber o seu número, o número do departamento que o controla e
último nome, endereço e data de nascimento do gestor desse
departamento. */

```

```

SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM   EMPLOYEE, DEPARTMENT, PROJECT
WHERE  Dnum=Dnumber AND Mgr_ssn=Ssn AND Plocation='Stafford';

```

Join Condition 1      Join Condition 2

EMPLOYEE										
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	Dnum
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4	
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5	
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5	
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	

Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

PROJECT			
Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPARTMENT			
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

## Junção - Ambiguidade de Nomes de Atributos

- Quando existem nomes de atributos iguais em distintas relações da junção, podemos utilizar o *full qualified name (fqn)*:

relation\_name.attribute

```

/* Exemplo: Vamos pegar num dos exemplos anteriores e imaginar
que o atributo Dno de EMPLOYEE se chamava Dnumber... */

```

```

SELECT Fname, Lname, Address
FROM   EMPLOYEE, DEPARTMENT
WHERE  Dname='Research' AND EMPLOYEE.Dnumber=DEPARTMENT.Dnumber;

```

Podemos também utilizar o fqn em situações em que não há ambiguidade de nomes.

18

## Junção - Ambiguidade + Renomeação

- Há situações em que ambiguidade de nomes de atributos resulta de termos uma relação recursiva.
- Nesta situação temos de renomeação as relações (*alias*).

```
/* Exemplo: Para cada Funcionário, pretendemos obter o seu
primeiro e último nome, assim como do seu supervisor. */
```

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
FROM   EMPLOYEE AS E, EMPLOYEE AS S
WHERE  E.Super_ssn=S.Ssn;
```

Muitas vezes a renomeação envolvendo várias relações ajuda a melhorar a legibilidade da instrução.

19

## Queries - Comparaçāo de Strings

- Operador LIKE permite comparar *Strings*
- Podemos utilizar wildcards.
  - % - significa zero ou mais caracteres.
  - \_ - significa um qualquer carácter.

### Exemplos:

```
/* Obter o primeiro e último nome dos funcionários cujo endereço contém a
substring 'Houston,TX'. */
```

```
SELECT Fname, Lname
FROM   EMPLOYEE
WHERE  Address LIKE '%Houston,TX%';
```

```
/* Obter o primeiro e último nome dos funcionários nascidos nos anos 50 */
```

```
SELECT Fname, Lname
FROM   EMPLOYEE
WHERE  Bdate LIKE '_ _ 5 _ _ _ _ _';
```

## Queries - Comparações de Strings

- Podemos pesquisar os próprios wildcards na string.
  - Para isso utilizamos um carácter especial a preceder o wildcard
  - Devemos definir esse carácter com a instrução ESCAPE

### LIKE ... ESCAPE

```
/* Nome dos funcionários cujo endereço contém a substring 'Houston%,TX'. */
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Address LIKE '%Houston@%,TX%' ESCAPE '@';
```

- Alguns SGBD permitem utilizar outros Wildcards.

Description	SQL Wildcard	MS-DOS Wildcard	Example
Any number (zero or more) of arbitrary characters	%	*	'Able' LIKE 'A%'
One arbitrary character	_	?	'Able' LIKE 'Ab_'
One of the enclosed characters	[ ]	n/a	'a' LIKE '[a-g]' 'a' LIKE '[abcddefg]'
Match not in range of characters	[^ ]	n/a	'a' LIKE '[ ^ w-z]' 'a' LIKE '[ ^ wxyz] '

SQL SERVER

21

## Queries - Operadores Aritméticos e BETWEEN

- Operações Aritméticas:
  - Operadores: adição (+), subtração (-), multiplicação (\*), divisão (/)
  - Operandos: valores numéricos ou atributos com domínio numérico.
- BETWEEN
  - Verificar se um atributo está entre uma gama de valores.

### Exemplos:

```
/* Obter o salário, com um aumento de 10%, de todos os trabalhadores do projeto GalaxyS. */
```

```
SELECT E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE E.Ssn=W.Essn AND W.Pno=P.Pnumber AND P.Pname='GalaxyS';

/* Funcionários do departamento nº 5 com salário entre 3k e 4k */
SELECT * FROM EMPLOYEE
WHERE (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

## Queries - Ordenação de Resultados

- Podemos ordenar os resultados segundo uma ou mais colunas.
- Sintaxe: **ORDER BY A<sub>1</sub>, ..., A<sub>k</sub>**
  - A<sub>1</sub>, ..., A<sub>k</sub> - atributos a ordenar.
  - 1,2,...,k - também podemos usar o número da coluna
- Podemos definir se é ascendente (ASC) ou descendente (DESC).
  - Por omissão as colunas são ordenadas ascendentemente.

### Exemplo:

```
/* Lista de funcionários e projetos em que trabalham, ordenado por
departamento e, dentro deste, pelo último nome (descendente) e depois o
primeiro */

SELECT    D.Dname, E.Lname, E.Fname, P.Pname
FROM      DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE     D.Dnumber= E.Dno AND E.Ssn= W.Essn AND W.Pno= P.Pnumber
ORDER BY  D.Dname, E.Lname DESC, E.Fname;
/* ... ORDER BY 1, 2 DESC, 3; */
```

## SQL DML

### Consultas Avançadas

## Tratamento dos NULL

- NULL
  - significa um valor desconhecido ou que não existe.
- SQL tem várias regras para lidar com os valores null.
- O resultado de uma expressão aritmética com *null* é *null*:  $5+null$  é *null*
- Temos possibilidade de verificar se determinado atributo é nulo: IS NULL
- Por norma, as funções de agregação ignoram o null.

25

## NULL - Lógica de 3 Valores

- Quando se faz uma comparação lógica temos duas possibilidades de retorno: TRUE, FALSE
- SQL - comparação com NULL retorna UNKNOWN.
  - $12 < null$ ,  $null <> null$ ,  $null = null$ , etc.
- Assim temos uma lógica de 3 valores em SQL:

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN
OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN
NOT			
TRUE	FALSE		
FALSE	TRUE		
UNKNOWN	UNKNOWN		

26

**deti**

## NULL Lógica 3 Valores - Exemplo

EMPLOYEE										
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	
Alicia	J	Zeloya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4	
Ramesh	K	Narayan	666884444	1982-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5	
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5	
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	NULL	NULL	1	

**Exemplos:**

```
/* Exemplo 1 */
SELECT Fname, Salary
FROM EMPLOYEE
WHERE Salary > 40000;
```

NULL > 40000  
UNKNOWN

Fname	Salary
Jennifer	43000

```
/* Exemplo 2 */
SELECT Fname, Salary
FROM EMPLOYEE
WHERE Salary > 40000 OR Fname='James';
```

UNKNOWN OR TRUE

Fname	Salary
Jennifer	43000
James	NULL

27

**deti**

## IS (NOT) NULL - Exemplo

- **IS NULL**: selecionar tuplos com determinado atributo a NULL;
- **IS NOT NULL**: selecionar tuplos com determinado atributo diferente de NULL;

**Exemplos:**

```
-- IS NOT NULL
SELECT * FROM EMPLOYEE
WHERE Super_ssn IS NOT NULL;
```

```
-- IS NULL
SELECT * FROM EMPLOYEE
WHERE Super_ssn IS NULL;
```

EMPLOYEE										
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	
Alicia	J	Zeloya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4	
Ramesh	K	Narayan	666884444	1982-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5	
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5	
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4	
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1	

28

 deti

## Junções - JOIN ON

- WHERE
  - Já vimos que o produto cartesiano associado ao operador “where” permite juntar várias relações. (ANSI SQL 89)
- ANSI SQL 92: JOIN ON utilizar sempre a partir de agora...
  - Permite especificar simultaneamente as tabelas a juntar e a condição de junção.

R  $\bowtie_c$  S

```
SELECT ... FROM (... [INNER] JOIN ... ON ...) ...;
-- [INNER] é opcional
-- exemplo de Equi-join:
SELECT Fname, Lname, Address
FROM (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE Dname='Research';
```

.9

 deti

## NATURAL JOIN

- Junção Natural - os atributos de junção têm todos o mesmo nome nas duas relações.
- Os atributos repetidos são removidos.
- Podemos renomear os atributos de uma relação para permitir a junção natural.

R  $\bowtie$  S

```
SELECT ... FROM (... NATURAL JOIN ...) WHERE <condition>;
```

-- exemplo de Natural Join com renomeação:

```
SELECT Fname, Lname, Address
FROM (EMPLOYEE NATURAL JOIN
      (DEPARTMENT AS DEPT (Dname, Dno, Mssn, Msdate)))
WHERE Dname='Research';
```

Não disponível em  
SQL Server!

30

## OUTER JOIN

- As junções externas podem ser à esquerda, à direita ou totais (LEFT, RIGHT, FULL).

```
SELECT ... FROM (... LEFT|RIGHT|FULL [OUTER] JOIN ...) ...;
```

```
/* exemplo de Outer Join com renomeação das relações e
atributos */
```

```
SELECT E.Lname AS Employee_name, S.Lname AS Supervisor_name
FROM   (EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE AS S
        ON E.Super_ssn=S.Ssn);
```

-- RIGHT OUTER JOIN  
-- FULL OUTER JOIN

R  $\bowtie_{A1=B2}$  S

R  $\bowtie_{A1=B2}$  S

R  $\bowtie_{A1=B2}$  S

31

Nota: Em Oracle utiliza-se o operador (+) à frente do atributo na cláusula WHERE.

## JOIN - Encadeamento

- Podemos ter várias operações JOIN encadeadas envolvendo 3..N relações.
  - uma das relações da junção resulta de outra operação de junção.

```
SELECT ... FROM (... JOIN ... JOIN ... JOIN ...) ...;
```

```
/* Exemplo do slide 17: Para cada projeto localizado em
'Stafford', queremos saber o seu número, o número do
departamento que o controla e último nome, endereço e data de
nascimento do gestor desse departamento. */
-- Nota: Neste caso as join conditions estão à frente do ON
```

```
SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM   ((PROJECT JOIN DEPARTMENT ON Dnum=Dnumber)
        JOIN EMPLOYEE ON Mgr_ssn=Ssn)
WHERE  Plocation='Stafford';
```

12

## Agregações

- Funções de agregação introduzidas em álgebra relacional.
- Funções de Agregação
  - Exemplos\*: COUNT, SUM, MAX, MIN, AVG
  - Em geral, não são utilizados os tuplos com valor NULL no atributo na função.
- Efetuar agregação por atributos
  - GROUP BY <grouping attributes>
- Efetuar seleção sobre dados agrupados
  - HAVING <condition>

\* Existem outras funções de agregação específicas do SGBD

33

## Funções de Agregação - Exemplo

### Exemplos... sem agrupamento de atributo(s)

```
/* Exemplo 1: relativamente aos salários dos funcionários, obter o valor total, o máximo, o mínimo e o valor médio */
SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM EMPLOYEE;

/* Exemplo 2: Nº de funcionários do departamento 'Research' */
SELECT COUNT (*)
FROM EMPLOYEE JOIN DEPARTMENT ON DNO=DNUMBER
WHERE DNAME='Research';

/* Exemplo 3: Nº de vencimentos distintos */
SELECT COUNT (DISTINCT Salary)
FROM EMPLOYEE;
```

Nota1: O operador COUNT(A1) conta o número de valores não NULL do atributo A1.  
O operador COUNT(\*) conta o número de linhas.

Nota2: Min, Max, Count(...) e Count(\*) podem ser utilizadas com qualquer tipo de dados. SUM e AVG só podem ser aplicadas a campos numéricos.

34

**Agregação (GROUP BY) - Exemplo**

**Exemplos... agregação de atributo(s)**

```
/* Exemplo 1: para cada departamento, obter o seu número, o
número de funcionários e a sua média salarial */
SELECT Dno, COUNT(*), AVG(Salary)
FROM EMPLOYEE
GROUP BY Dno;
```

Os “grouping attributes” devem aparecer na cláusula SELECT  
Exemplo: Dno

Fname	Minit	Lname	San	...	Salary	Super_ssn	Dno
John	B	Smith	123456789	...	30000	333445555	5
Franklin	T	Wong	333445555	...	40000	888665555	5
Ramesh	K	Narayan	666884444	...	38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777	...	25000	987654321	4
Jennifer	S	Wallace	987654321	...	43000	888665555	4
Ahmad	V	Jabbar	987987987	...	25000	987654321	4
James	E	Bong	888665555	...	55000	NULL	1

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

```
/* Exemplo 2: agregação com junção de duas relações */
SELECT Pnumber, Pname, COUNT(*)
FROM PROJECT JOIN WORKS_ON ON Pnumber=Pno
GROUP BY Pnumber, Pname;
```

Nota: Se existirem valores NULL nos “grouping attribute”, então é criado um grupo com todos os tuplos contendo NULL nesses atributos.

35

**Agregação (GROUP BY.. HAVING) - Exemplo**

**Exemplo... agregação de atributo(s) com seleção**

```
/* Exemplo 1: Para cada projeto, com mais de dois funcionários,
obter o seu nome e nº de funcionários que trabalham no projeto
*/
SELECT Pname, COUNT(*)
FROM PROJECT join WORKS_ON
ON Pnumber=Pno
GROUP BY Pname
HAVING COUNT(*) > 2;
```

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Junção

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1	...	123456789	1	32.5
ProductX	1	...	453453453	1	20.0
ProductY	2	...	123456789	2	7.5
ProductY	2	...	453453453	2	20.0
ProductY	2	...	333445555	2	10.0
ProductZ	3	...	666884444	3	40.0
ProductZ	3	...	333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10	...	987987987	10	35.0
Reorganization	20	...	333445555	20	10.0
Reorganization	20	...	987654321	20	15.0
Reorganization	20	...	888665555	20	NULL
Newbenefits	30	...	987987987	30	5.0
Newbenefits	30	...	987654321	30	20.0
Newbenefits	30	...	999887777	30	30.0

Nota1: A condição da cláusula WHERE é aplicada antes da criação dos grupos. A condição do HAVING é executada depois da criação dos grupos.

Nota2: Na cláusula HAVING só podemos ter atributos que aparecem em GROUP BY ou funções de agregação.

36

 deti

## Agregação - Resumo

**SQL**

```
SELECT      A1,..,An, FAgri1,..Fagrh
FROM        R1,R2,..,Rm
WHERE       <condition_W>
GROUP BY   A1,..,An
HAVING     <condition_H>;
```

Expressão equivalente em álgebra relacional

$$\Pi_{A1,..,An, FAgri1,..Fagrh} (\sigma_{<\text{condition\_H}>} (A1,..,An \bowtie_{FAgri1,..,Fagrh} (\sigma_{<\text{condition\_W}>} (R1 \times .. \times Rm))))$$

Importante para se perceber  
a ordem das operações

37

 deti

## SubConsultas (SubQueries)

- É possível usar o resultado de uma query, i.e. uma relação, noutra query.
  - Nested Queries
- Subconsultas podem aparecer na cláusula:
  - FROM - entendidas como cálculo de relações auxiliares.
  - WHERE - efetuar testes de pertença a conjuntos, comparações entre conjuntos, calcular a cardinalidade de conjuntos, etc.

38

## Cláusula FROM - Subquery como Tabela

- Podemos utilizar o resultado de uma subquery como uma tabela na cláusula FROM, dando-lhe um nome (alias).

### Exemplo... agregação de atributo(s) com seleção

```
/* Exemplo 1: Obter uma lista de funcionários com mais de dois dependentes */
SELECT      Fname, Minit, Lname, Ssn
FROM        Employee JOIN (
    SELECT      Essn
    FROM        DEPENDENT
    GROUP BY   Essn
    HAVING     count(Essn)>2) AS Dep
    ON Ssn=Dep.Essn;
```

39

## Operador IN - Pertença a Conjunto

- WHERE A1,...,An IN (SELECT B1,...,Bn FROM ...)
  - Permite selecionar os tuplos em que os atributos indicados (A1,...,An) existem na subconsulta.
  - B1,...,Bn são os atributos retornados pela subconsulta
- A1,...,An e B1,...,Bn
  - têm de ter o mesmo número atributos e domínios compatíveis.
- NOT IN
  - permite obter o resultado inverso.

40

**Operador IN - Exemplo**

**Exemplos...**

```

/* Exemplo 1: Obter o nome de todos os funcionários que não têm
dependentes */
SELECT      Fname, Minit, Lname
FROM        EMPLOYEE
WHERE       Ssn NOT IN (SELECT Essn FROM DEPENDENT);

/* Exemplo 2: Obter o Ssn de todos os funcionários que trabalham
no mesmo projeto, e o mesmo número de horas, que o funcionário
com o Ssn = '123456789'*/
SELECT      DISTINCT Essn
FROM        WORKS_ON
WHERE       (Pno, Hours) IN ( SELECT Pno, Hours
                             FROM  WORKS_ON
                             WHERE  Essn='123456789');

/* Exemplo 3: Obter o Ssn de todos os funcionários que trabalham
no projeto nº 1, 2 ou 3 */
SELECT      DISTINCT Essn
FROM        WORKS_ON
WHERE       Pno IN (1, 2, 3);

```

SQL Server não suporta múltiplas colunas!

**Comparação de Conjuntos**

- Existem **operadores** que pode ser utilizados para **comparar** um **valor simples** (tipicamente um atributo) **com** um **set** ou **multiset** (tipicamente uma subquery).
- ANY (= CASE)**
  - Permite selecionar os resultados cujos atributos indicados sejam iguais (=), maiores (>), menores(<) ou diferentes (<>) do que pelo menos um tuplo da subquery.
  - =ANY é o mesmo que IN
- ALL**
  - Também pode ser combinada com os operadores iguais (=), maiores (>), menores(<) ou diferentes (<>).

42

 deti

## ANY e ALL - Exemplos

**Exemplos...**

```

/* Exemplo 1: Obter o nome dos funcionários cujo salário é
maior do que o salário de todos os trabalhadores do departamento
5 */
SELECT      Lname, Fname
FROM        EMPLOYEE
WHERE       Salary > ALL ( SELECT  Salary
                           FROM    EMPLOYEE
                           WHERE   Dno=5);

/* Exemplo 2: Obter o nome dos funcionários cujo salário é
maior do que o salário de algum trabalhador do departamento 5 */
SELECT      Lname, Fname
FROM        EMPLOYEE
WHERE       Salary > ANY ( SELECT  Salary
                           FROM    EMPLOYEE
                           WHERE   Dno=5);

```

13

 deti

## Teste de Relações Vazias - EXISTS

- O operador EXISTS retorna
  - TRUE, se subconsulta não é vazia.
  - FALSE, se subconsulta é vazia.
- Existe a possibilidade de utilizar o NOT EXISTS

**SQL - (NOT) EXISTS**

```

/* Exemplo 1: Nomes dos funcionários que não têm dependentes */
SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       NOT EXISTS ( SELECT  *
                           FROM    DEPENDENT
                           WHERE   Ssn=Essn );

```

44

## Existem Tuplos Duplicados? - UNIQUE

- Unique permite verificar se o resultado de uma subconsulta possui tuplos duplicados.
- Permite verificar se determinado resultado (relação) é um conjunto ou um multiconjunto.

### SQL - (NOT) EXISTS

```
/* Exemplo 1: Nomes dos funcionários que gerem um departamento.
(supondo que o mesmo funcionário pode gerir mais do que um
departamento...) */
SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       UNIQUE      ( SELECT  Mgr_ssn
                           FROM   DEPARTMENT
                           WHERE   Ssn=Mgr_ssn );
```

Não disponível em  
SQL Server!

45

## SubConsultas Não Correlacionadas

- A subquery (query interior) não depende de dados lhe são fornecidos pela query exterior.
  - Nestes casos, a query interior é executada uma única vez e o resultado é utilizado no SELECT exterior.

### SubConsulta Correlacionada

```
/* Exemplo 1: Nome dos funcionário que são gestores de
departamento */

SELECT      Fname, Lname
FROM        EMPLOYEE
WHERE       Ssn IN      (           SELECT  Mgr_ssn
                           FROM   DEPARTMENT
                           WHERE   Mgr_ssn IS NOT NULL);
```



46

## SubConsultas Correlacionadas

- A subquery (query interior) depende de dados lhe são fornecidos pela query exterior.
  - Nestes casos, a query interior é executada uma vez para cada resultado do SELECT exterior.

### SubConsulta Correlacionada

```
/* Exemplo 1: Nome dos funcionários que tem um dependente com o
primeiro nome e sexo igual ao próprio funcionário */
```

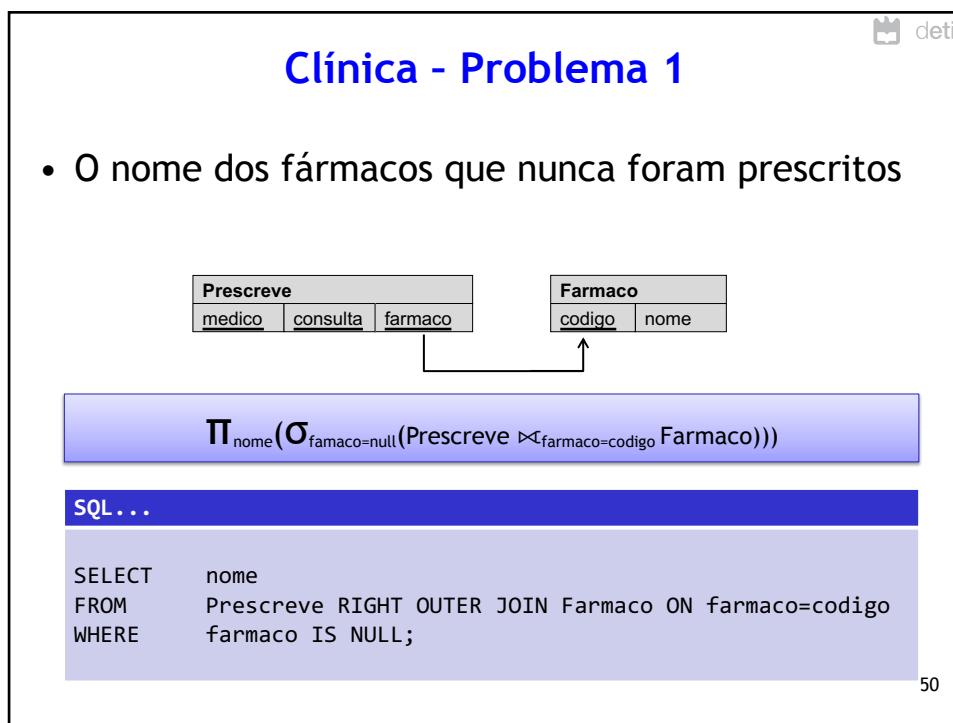
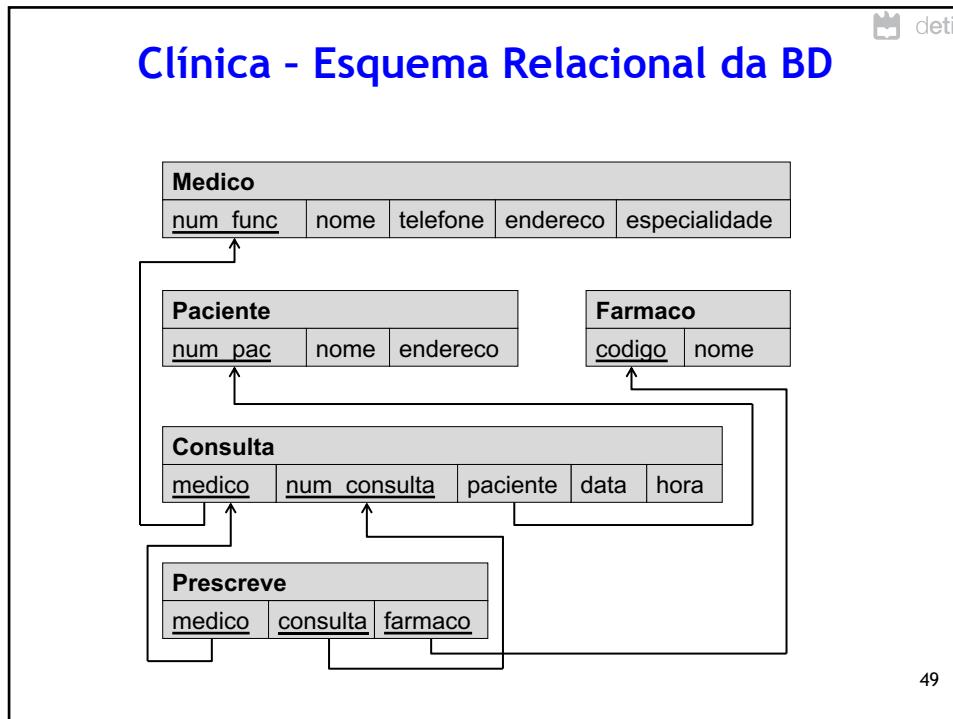
```
SELECT      E.Fname, E.Lname
FROM        EMPLOYEE AS E
WHERE       E.Ssn IN (      SELECT      Essn
                           FROM        DEPENDENT AS D
                           WHERE       E.Fname=D.Dependent_name
                                      AND E.Sex=D.Sex );
```

47

## SQL DML - Caso de Estudo

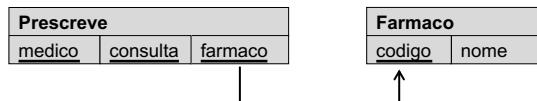
Clínica  
(Conversão das Queries AR para SQL)

48



## Clínica - Problema 2

- O número de fármacos prescritos em cada consulta



$\Pi_{medico, consulta, num\_farm=count(farmaco)} (\text{Prescreve})$

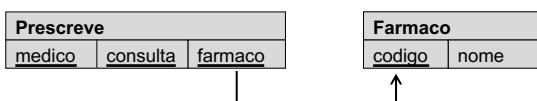
SQL...

```
SELECT      medico, consulta, count(farmaco) AS num_farm
FROM        Prescreve
GROUP BY    medico, consulta;
```

51

## Clínica - Problema 3

- Para cada médico, a quantidade média de fármacos receitados por consulta



$\Pi_{medico, avg\_farmaco=avg(num\_farm)} (\Pi_{medico, consulta, num\_farm=count(farmaco)} (\text{Prescreve}))$

SQL...

```
SELECT      medico, avg(num_farm) AS avg_farmaco
FROM        (SELECT      medico, consulta, count(farmaco) AS num_farm
            FROM        Prescreve
            GROUP BY    medico, consulta) AS T
GROUP BY    medico;
```

52

**Clínica - Problema 4**

- O nome de todos os fármacos prescritos, incluindo a quantidade, para o paciente número 35312161

```

temp ← πmedico, num_consulta(σpaciente=35312161(Consulta))
temp2 ← πfarmaco, quantidade=count(farmaco)(temp ⋈medico=medico AND num_consulta=consulta Prescreve)
πnome, quantidade(temp2 ⋈farmaco=codigo Farmaco)

```

SQL...

```

SELECT nome, quantidade
FROM Farmaco JOIN (SELECT farmaco, count(farmaco) AS quantidade
                     FROM Prescreve AS T1
                     JOIN (SELECT medico, num_consulta
                           FROM Consulta
                           WHERE paciente=35312161) AS T
                           ON (T1.medico=T.medico AND T.num_consulta=T1.consulta) AS T2
                     GROUP BY farmaco)
ON farmaco=codigo;

```

**Clínica - Problema 5**

- O nome dos fármacos que já foram prescritos por todos os médicos da clínica

```

temp ← ρcodigo, num_func(πfarmaco, medico(Prescreve)) ÷ πnum_func(Medico)

```

π<sub>nome</sub>(temp ⋈ Farmaco) ÷ não existe em SQL

SQL... Uma Implementação Alternativa da Query:

```

SELECT      farmaco, count(DISTINCT medico) as num_medicos
FROM        Prescreve
GROUP BY    farmaco
HAVING      count(DISTINCT medico)=(SELECT count(*) from Medico);

```

