TÉCNICO LISBOA

Bases de Dados

Sumário

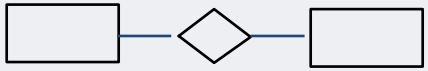
- Modelo Relacional: Definições e características
- Conversão Modelo E-A para Relacional
  - Entidades
  - Associações M:N
  - Associações 1:N
  - Associações 1:1
  - Associações Ternárias

Concepção de uma Base de Dados

Especificação de Requisitos

- requisito funcional 1:
- requisito funcional 2:
- …
- **regra de integridade 1:**
- **regra de integridade 2:**
- …

Modelo Conceptual
Modelo EA



TRADUÇÃO

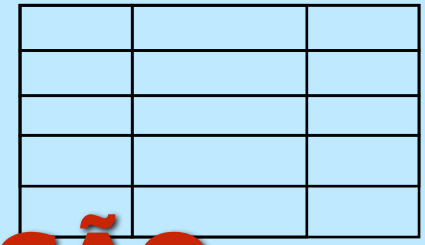
Modelo Relacional

R1(X, Y, Z)

R2(A, B, C)

TRADUÇÃO

Esquema Relacional



O Modelo Relacional

História

- Criado em 1970 por E. F. Codd

- Muito simples e elegante:
  - Baseado na teoria de conjuntos
  - Permite detectar rapidamente incoerências
  - Uma Base de Dados é vista com uma colecção de **relações**
  - Cada relação "é concretizada" num SGBD por uma tabela com linhas e colunas

Bibliografia

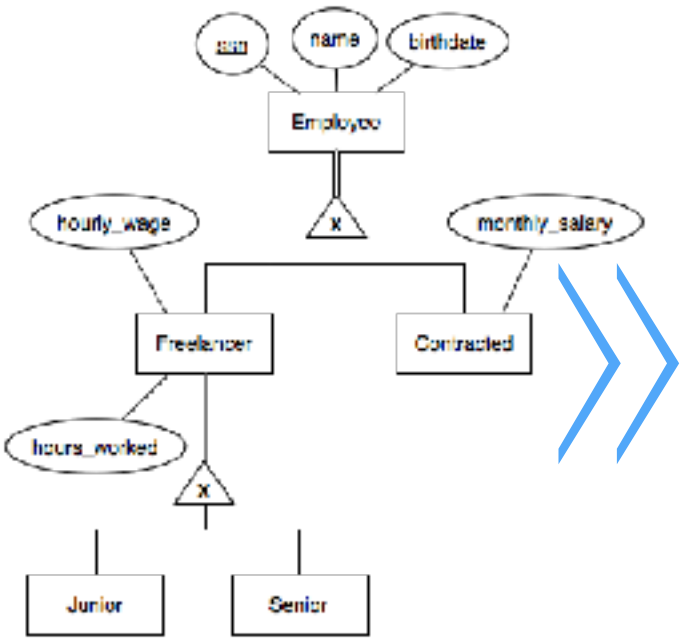
https://en.wikipedia.org/wiki/Edgar_F._Codd

Motivação

- O modelo relacional é um modelo de solução que é independente da implementação

- Pode ser facilmente realizado num SGBD relacional (ou noutro sistema)

Transformação E-A para Relacional



employee(<u>ssn</u>, name)

RI-1: ssn tem de existir em Freelancer e/ou em Contracted

RI-2: ssn não pode existir em Freelancer e Contracted

contracted(<u>ssn</u>, monthly_salary)

ssn: FK(employee.ssn)

freelancer(<u>ssn</u>, hours_worked, hourly_wages)

ssn: FK(employee.ssn)

RI-3: ssn não pode existir em Junior e Senior

junior(ssn)

ssn: FK(freelancer.ssn)

senior(ssn)

ssn: FK(freelancer.ssn)

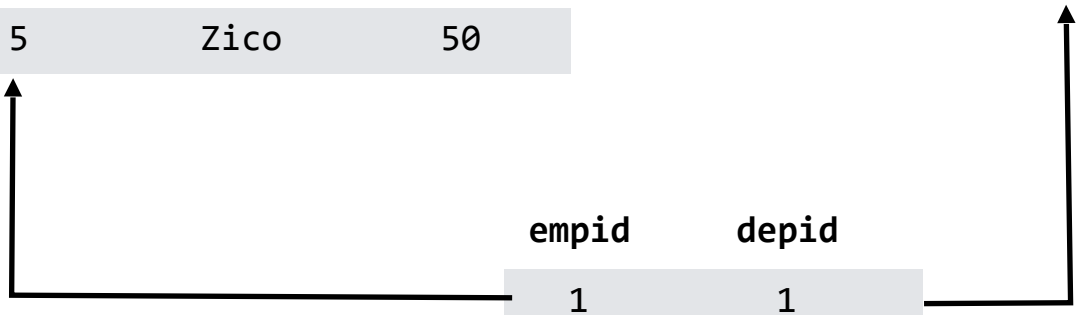
Fundamentos do modelo relacional

Tabelas

empid	name	age
1	Cajó	20
2	Quim	30
3	Xico	25
4	Zé	40
5	Zico	50

depid	name	location
1	Finance	Buraca
2	Chemistry	Damaia
3	Sales	Chelas

empid	depid
1	1
2	2
3	3
4	3
5	3



Representação colunar

employee = {⟨1, Cajó, 20⟩, ⟨1, Quim, 30⟩, ⟨3, Xico, 25⟩, ⟨4, Zé, 40⟩, ⟨5, Zico, 50⟩}

empid	name	age
1	Cajó	20
2	Quim	30
3	Xico	25
4	Zé	40
5	Zico	50

Relação: Definição Informal

- Uma **relação** consiste em:
  - Um **esquema de relação $R(A_1,\ldots,A_n)$**

Students(sid: string, name: string, login: string, dbirth: date, grade: float)

  - Um **conjunto *r*** designado por **relação**
  - **Onde todos os elementos relação *r*** tem a estrutura definida pelo esquema relação **R**

- Uma relação é representável como uma **tabela**

Relação: Definição Formal

- Dado um **esquema de relação** R(A_1,…, A_n), onde cada atributo A_i (distinto) tem o domínio D_i
  - Uma **relação** *(exemplar ou instância de relação)* *r* é um conjunto
    $r \subseteq D_1 \times \ldots \times D_n$

- *r* é um conjunto de *n*-tuplos na forma $\langle a_1, \ldots, a_n \rangle$, onde cada $a_i \in D_i$

Características das Relações

- O **grau** de uma relação é o seu número de atributos (colunas ou campos)

- A **cardinalidade** de um exemplar (*instance*) de uma relação é o seu número de tuplos (linhas)

Exercício

Quais os conjuntos que não são relações? Justifique.

A. {}

B. {⟨⟩}

C. {⟨1⟩}

D. {⟨1⟩, ⟨2⟩}

E. {⟨Alice⟩, ⟨1000⟩}

F. {⟨Alice, 100⟩, ⟨Bob, 200⟩}

G. {⟨Alice, 100⟩, ⟨Bob⟩}

H. {⟨Alice, 100⟩, ⟨Alice, 200⟩}

I. {⟨Alice, 100⟩, ⟨Alice, 100⟩}

Qual o resultado das seguintes expressões

A. $\{\langle \text{Alice}, 100\rangle, \langle \text{Bob}, 200\rangle\} \cup \{\langle \text{Alice}, 100\rangle\}$?

B. $\{\langle \text{Alice}, 100\rangle, \langle \text{Bob}, 200\rangle\} \cap \{\langle \text{Alice}, 200\rangle, \langle \text{Alice}, 100\rangle\}$?

Propriedades das Relações

- Não existem tuplos repetidos (duplicados)
- Não existem atributos repetidos (com o mesmo nome)
- A ordem dos tuplos é irrelevante
- A ordem dos atributos é irrelevante — por exemplo: *emp*(id, salary) não tem mais nem menos informação que *emp*(salary, id).

Base de Dados Relacional

- Uma **base de dados relacional** é uma colecção de relações com nomes distintos
- O **esquema relacional de uma base de dados** é uma colecção de esquemas para as relações na base de dados
- Um **exemplar** (*instance*) **de uma base de dados relacional** é uma colecção de exemplares (*instances*) de relações

Restrições

Restrições sobre Relações

Definição

- Uma **restrição de integridade sobre uma relação** é uma condição especificada sobre o esquema de uma relação que restringe as combinações dados que podem ser armazenados na relação

- **Restrição de Domínio**
- **Restrições de Chave**
- **Restrições de Unicidade/Chave Candidata**

Restrição de domínio

- O **domínio** de um campo/atributo restringe os valores que podem ocorrer nesse campo/atributo
- Propriedade importante que cada exemplar (instance) da relação tem que satisfazer:
  - Numa relação: Os valores que aparecem num atributo têm que pertencer ao domínio associado a essa atributo
- Um exemplar de relação satisfaz sempre as restrições de domínio de um esquema de relação

Restrições de Chave

- Uma **restrição de chave** indica que um certo conjunto mínimo de atributos de uma relação constitui um identificador único para qualquer tuplo
  - Ou seja, os seus valores identificam **univocamente** qualquer tuplo de uma instância de relação
  - Dois tuplos distintos não podem ter valores iguais para os atributos da chave
  - Nenhum sub-conjunto de atributos da chave pode ser uma chave

Restrições de chave candidata ou de unicidade

- Uma chave candidata deve ser indicada com a restrição de unicidade **unique**
- Por exemplo na relação

  Students(sid, name, login, bdate, grades

  - unique(login)

  - sid é chave
  - login é único

Restrições sobre Bases de Dados

Definição

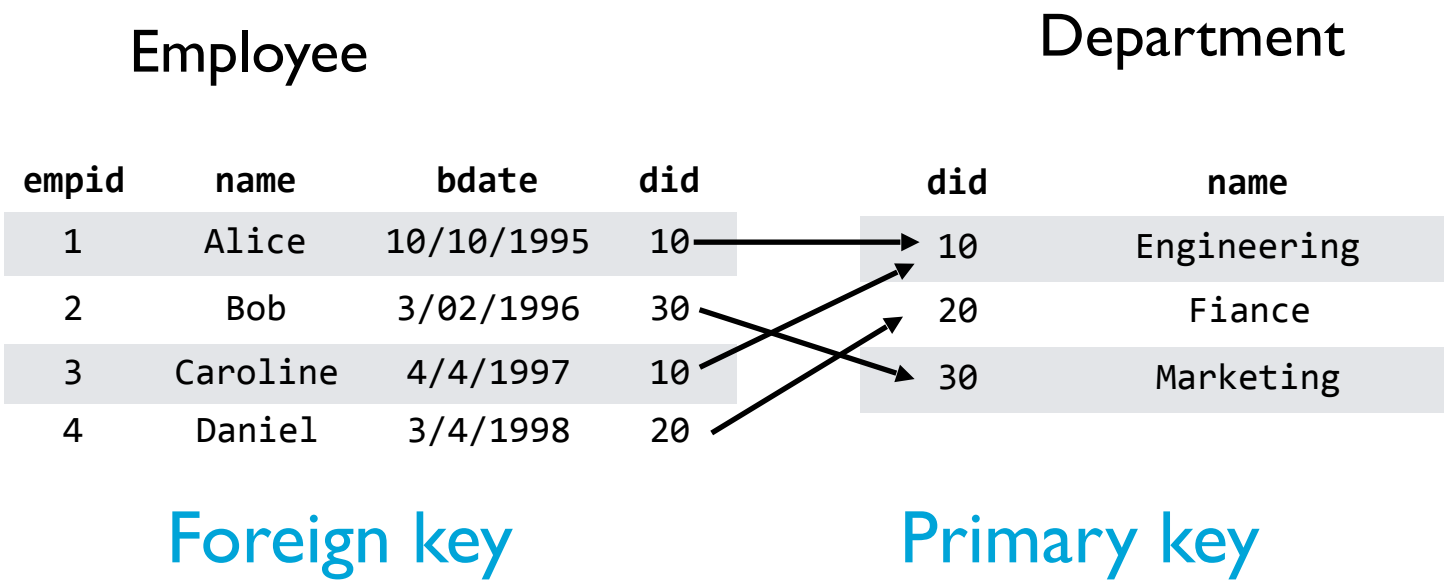
- Uma **restrição de integridade sobre uma base de dados** é uma condição especificada sobre o esquema de diversas relações que restringe as combinações dados que podem ser armazenados nas relações (da base de dados).

- **Restrição de Integridade Referencial (ou de Chave Estrangeira)**
- **Restrição de Integridade genérica**

Restrição de **Integridade Referencial** (ou de Chave Estrangeira)

- Uma relação está normalmente relacionada com outras relações
- Se os dados de uma relação são alterados, as outras relações devem ser verificadas para manter os dados consistentes
- A restrição de integridade que envolve duas relações mais comum é a **restrição de integridade referencial**

Integridade Referencial: Exemplo



- Cada valor de ***did*** que aparece na tabela ***Employee*** tem de ocorrer na coluna da chave primária da tabela ***Department***

Violações da Integridade Referencial

- Operações que podem originar violações:
  - Inserir linhas em Employee
  - Remover linhas de Department
  - Alterar valores de campos chave em Department

"Chave Estrangeira"

- Não é necessariamente "Chave"
  - Não precisa de ser chave (em nenhum dos lados)
- Não é necessariamente "Estrangeira"
  - A chave estrangeira pode referenciar a própria tabela
  - Exemplo: Adicionar coluna *supervisor* à tabela *Employee*
    - Mas se o Employee não tem *supervisor*? (melhor colocar fora)

Especificação de restrições de Integridade Referencial

Student(*sid, name, login, bdate, gpa*)

Enrolled(*cid, sid, grade*)

- *sid:* ***FK****(Students.sid)*
- *sid:* ***FK****(Students)*

Outras RIs

RI-1: Um aluno só se poder inscrever a uma cadeira não tiver atingido o limite de créditos

- *Especificadas tal como em EA*

Comportamento de um SGBD racional

- O **SGBD rejeita actualizações** que violem as restrições que podem ser especificadas em SQL através de:
  - Restrições de tabela
    - Envolvem uma única tabela
  - Asserções
    - Envolvem várias tabelas
- Falaremos da sua implementação mais à frente:
  - *Restrições em SQL e Triggers*

SGBD

- O **SGBD rejeita actualizações** que violem as restrições

Tipo de restrição	Implementação no SGBD
Domínio	Domínios dos atributos no esquema
Chave Primária	Restrição de PRIMARY KEY e UNIQUE no esquema
Chave Candidatas	Restrição de UNIQUE no esquema
Integridade Referêncial	Restrição de FOREIGN KEY
Restrição de Integridadde	Assertions, Stored Procedures e Triggers

Resumo das ideias fundamentais

- Apenas existem **relações** (conjuntos estruturados)
- Todos os elementos de uma relação devem obedecer ao **esquema da relação**
- As operações sobre os dados correspondem a **operações sobre conjuntos**
- As situações que não podem ocorrer são especificadas com **restrições**

See also

The relational model

https://www.youtube.com/watch?v=F7j31IEBctU

Tradução E-A para Relacional

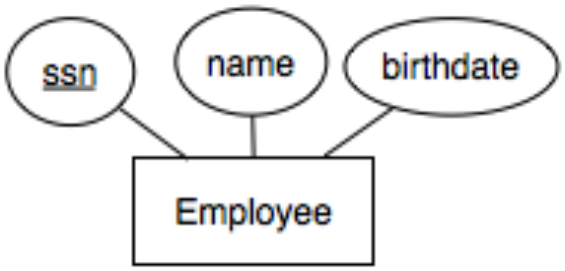
Tradução de Entidades e Atributos

Entidades

- **Entidades** (fortes) originam uma **relação** no esquema relacional

Atributos mapeados com o mesmo nome na relação

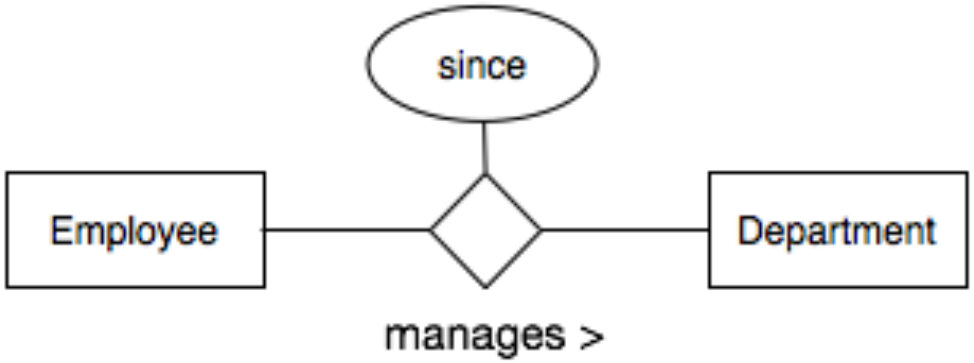
Chave primária é a mesma



Employee(ssn, name, birthdate)

Tradução de Associações (multiplicades)

Associações M:N



Employee(<u>ssn,</u> name)
Department(<u>did</u>, dname, budget)

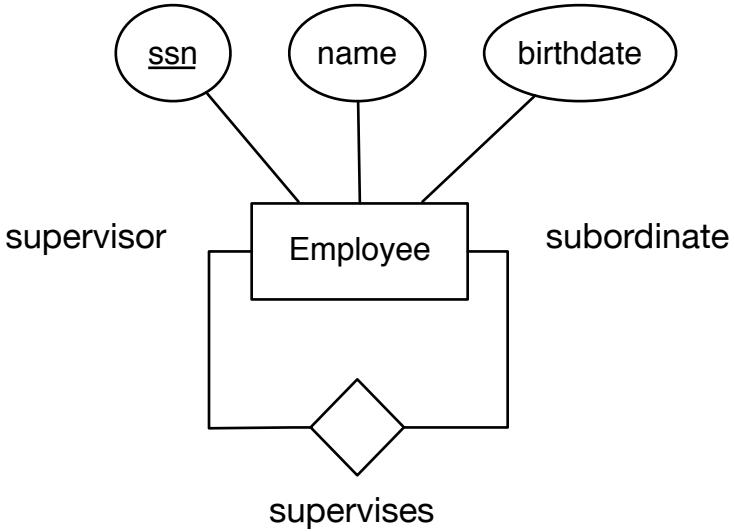
manages(<u>ssn, did</u>, since)
*ssn: **FK**(Employee)*
*did: **FK**(Department)*

Associações M:N

- Associações "muitos-para-muitos" originam uma relação cujos esquema é composto por:
  - Chave constituída pelas chaves primárias das relações que correspondem às entidades participantes
  - Atributos descritivos da Associação
  - Restrições de chaves estrangeiras que correspondem às chaves primárias das entidades participantes

Associações M:N

Caso especial: Auto-associação

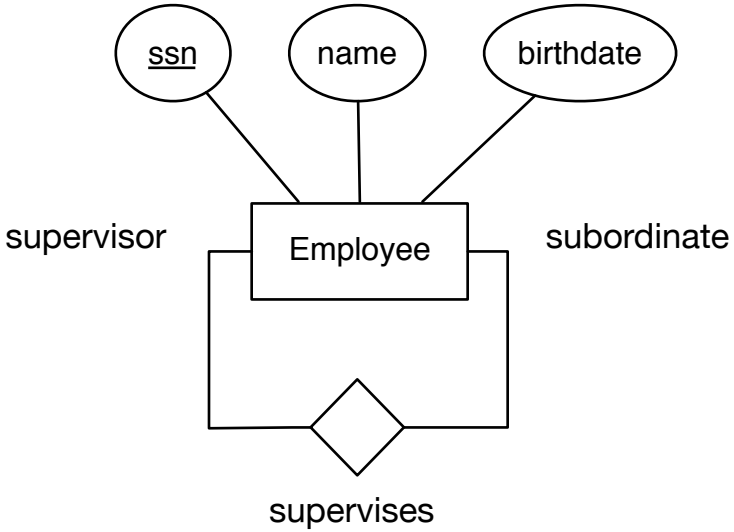


supervises(ssn, ssn)

- ssn: FK(Employee.ssn)
- ssn: FK(Employee.ssn)

Associações M:N

Caso especial: Auto-associação

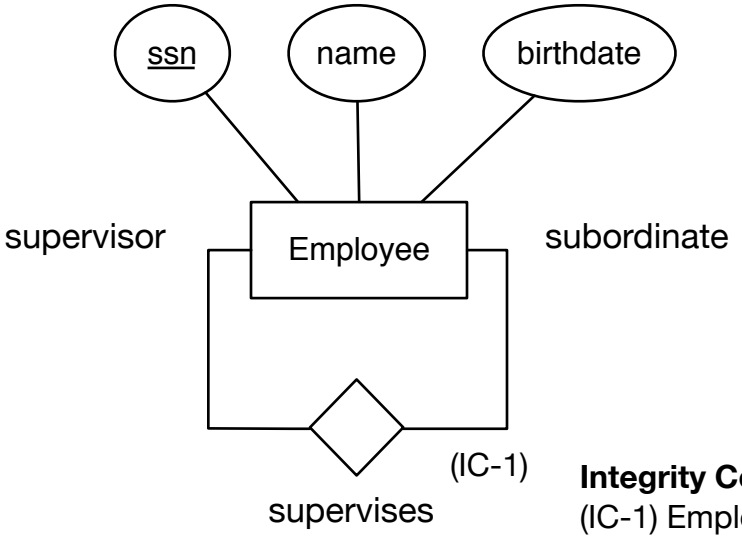


supervises(supervisor_ssn, subordinate_ssn)

- supervisor_ssn: FK(Employee.ssn)
- subordinate_ssn: FK(Employee.ssn)

Associações M:N

Caso especial: Auto-associação

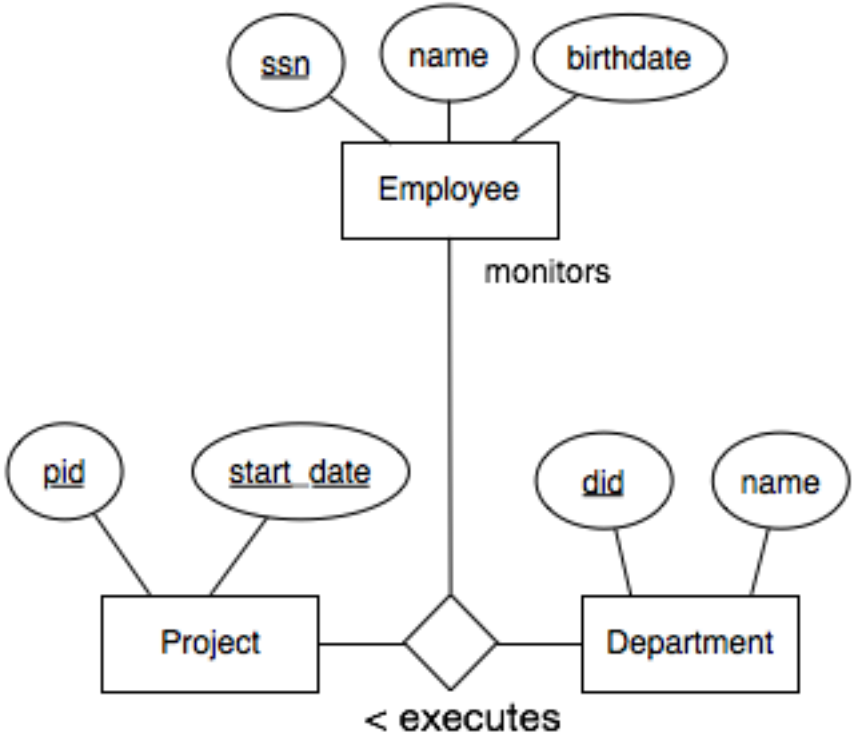


Integrity Constraints:
(IC-1) Employees are not allowed to **supervise** themselves

supervises(supervisor_ssn, subordinate_ssn)

- supervisor_ssn: FK(Employee.ssn)
- subordinate_ssn: FK(Employee.ssn)
- IC-1: supervisor_ssn is always different form subordinate_ssn

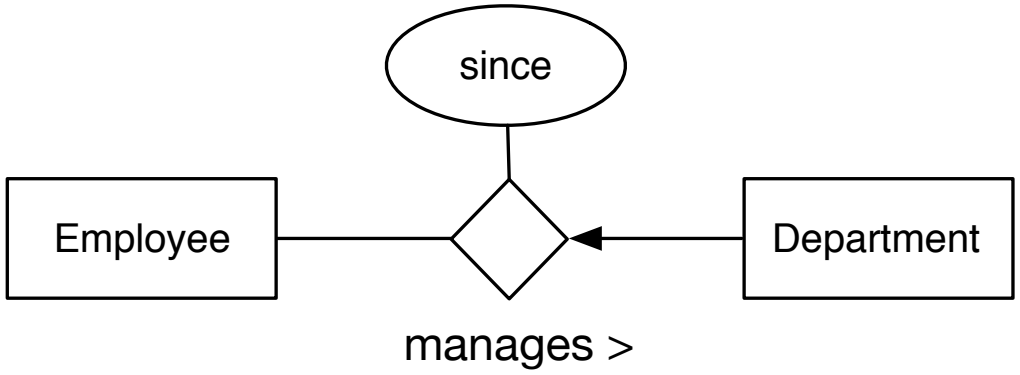
Associações Ternárias



executes(ssn, pid, did)

- *ssn: FK(Employee)*
- *pid: FK(Project)*
- *did: FK(Department)*

Multiplicidade: One-to-many

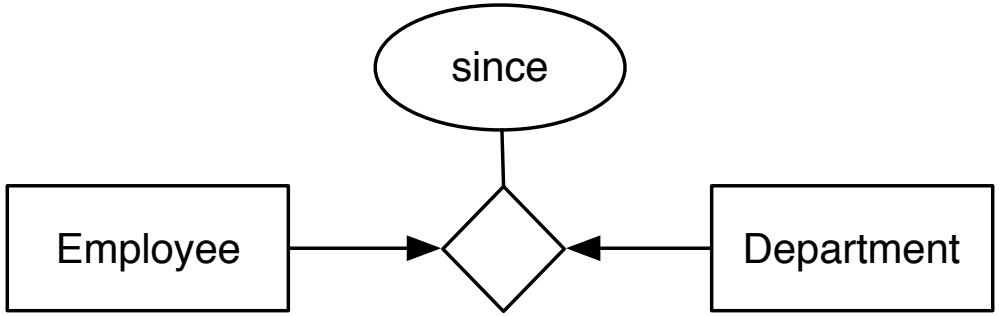


employee(ssn, name)
department(did, dname, budget)

manages(ssn, did, since)
- *ssn: FK(Employee)*
- *did: FK(Department)*

did nunca se repete!
Logo: Uma vez que um '*department*' seja associado a um '*employee*' através da relação '*manages*', já não pode ser associado novamente.

Multiplicidade: One-to-one



employee(ssn, name)
department(did, dname, budget)

manages(ssn, did, since)

- *ssn: FK(Employee)*
- *did: FK(Department)*
- *unique(ssn)*
- *unique(did)*

nem *ssn* nem *did* se repetem!

Logo: Uma vez que um '*department*' ou '*employee*' sejam associados através da relação '*manages*', já não podem ser associados novamente.

TÉCNICO LISBOA

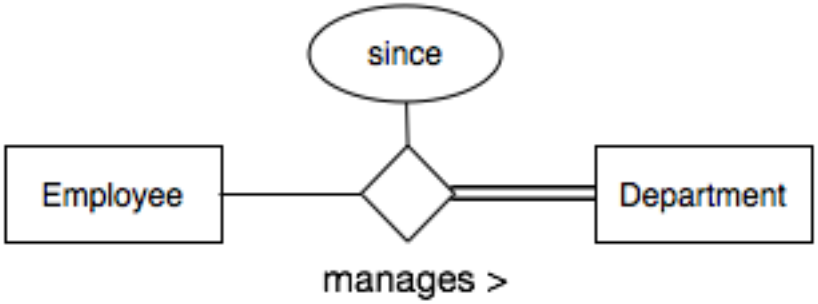
Bases de Dados

Aula 07: Modelo Relacional e Tradução E-A para Relacional (cont)

Prof. Paulo Carreira

Tradução de Associações (Obrigatoriedade)

Participação Total M:M

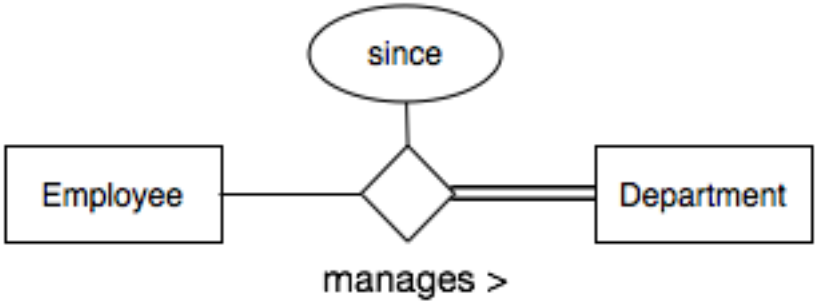


employee(<u>ssn,</u> name)
department(<u>did</u>, dname, budget)

- RI: todo o *departamento (did)* tem de participar na associação

manages(<u>ssn, did</u>, since)
- *ssn: FK(Employee)*
- *did: FK(Departments)*

Participação Total M:M

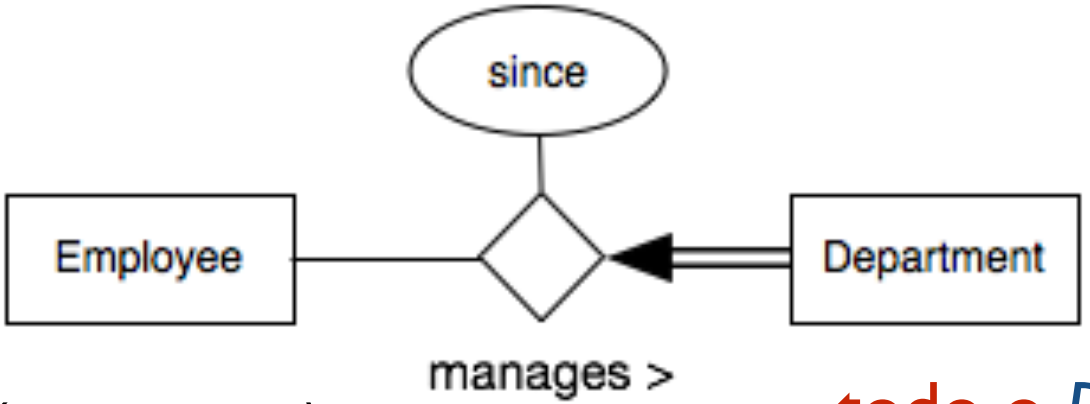


employee(ssn, name)
department(did, dname, budget)

- RI: um *departamento (did)* é válido se participar na relação 'manages'

manages(ssn, did, since)
- *ssn: FK(Employee)*
- *did: FK(Departments)*

Participação Total: 1:M
(com apenas duas tabelas)



employee(ssn, name)
department(did, dname, budget)

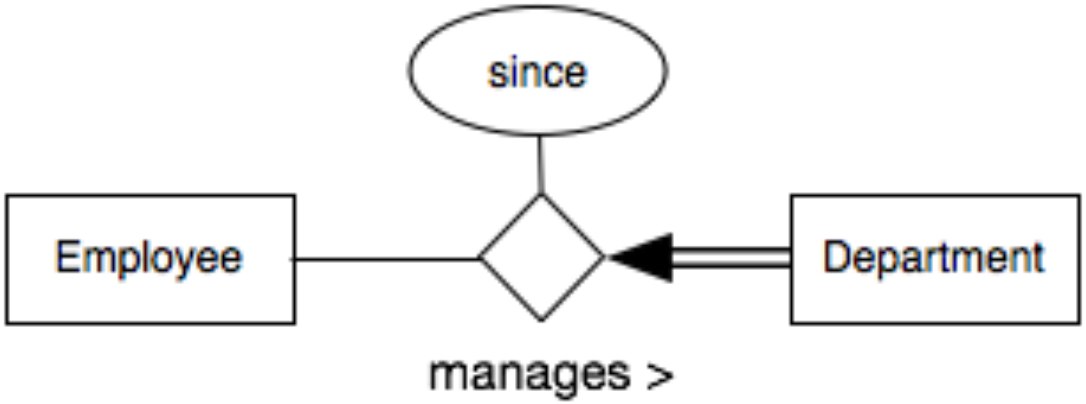
todo o *Departament (did)* tem de participar na associação!

- RI: um *departamento (did)* é válido se participar na relação 'manages'

cada *Departament (did)* só pode ser gerido por um Manager

manages(ssn, did, since)
- *ssn: FK(Employee)*
- *did: FK(Department)*

Participação Total: 1:M
(com apenas duas tabelas)



employee(ssn, name)

department(did, dname, budget, manager, since)

- *manager: FK(Employee.ssn)*

cada *Departament (did)* só pode ser gerido por um Manager

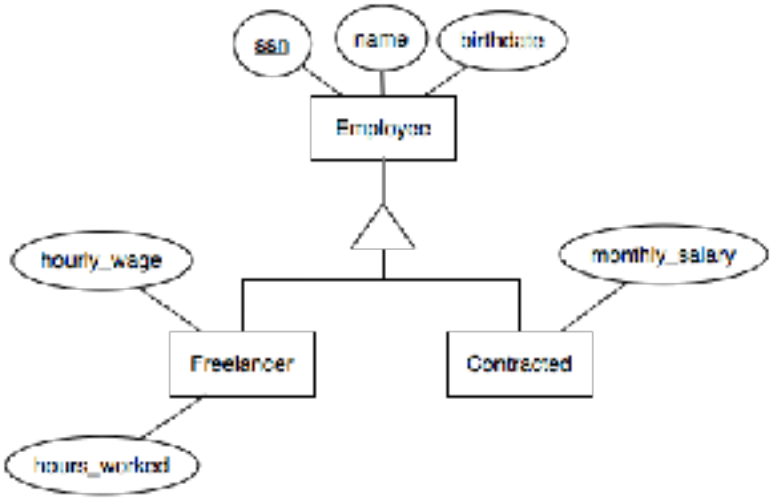
todo o *Departament (did)* tem de ter um manager
- e logo, participar na associação, obrigatoriamente

Tradução da Generalização

Generalização

- Mapear a super-entidade numa relação
- Mapear as sub-entidades em relações distintas onde:
  - A chave de cada sub-entidade é a chave da super-entidade (com a correspondente restrição de FK)
  - Restrições de "disjoint" ou de especialização obrigatória são mapeadas através de RIs sobre a super entidade

Generalização simples



employee(<u>ssn</u>, name, birthdate)

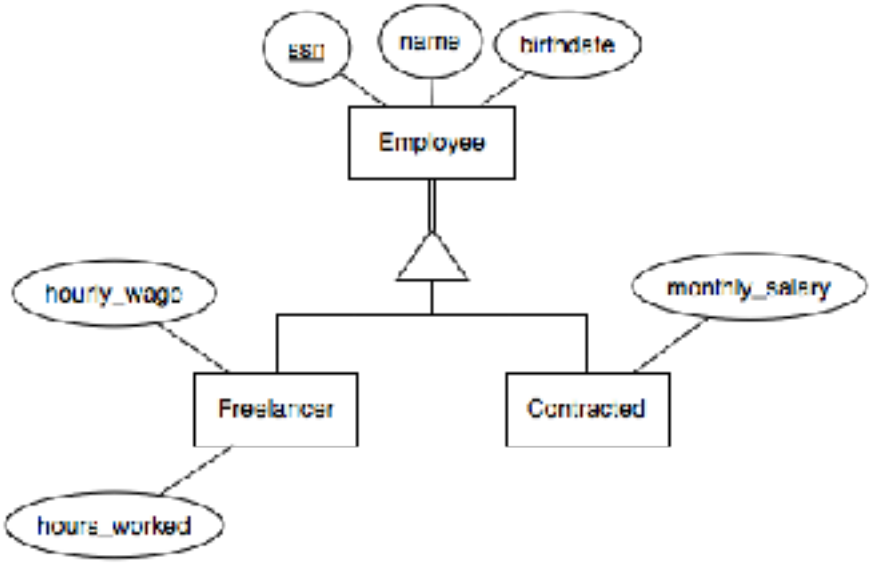
contracted(<u>ssn</u>, monthly_salary)

- *ssn: FK(Employee.ssn)*

freelancer(<u>ssn</u>, hours_worked, hourly_wages)

- *ssn: FK(Employee.ssn)*

Especialização obrigatória



employee(<u>ssn</u>, name, birthdate)

- *RI-1: ssn tem existir em Freelancer e/ou Contracted*

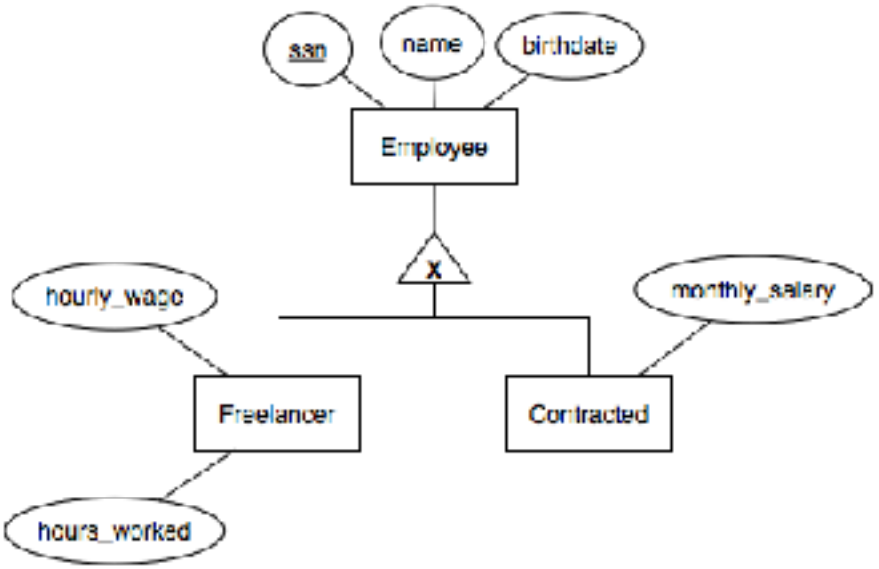
contracted(<u>ssn</u>, monthly_salary)

- *ssn: FK(Employee.ssn)*

freelancer(<u>ssn</u>, hours_worked, hourly_wages)

- *ssn: FK(Employee.ssn)*

Especialização disjunta



employee(ssn, name, birthdate)

- *RI-1: ssn não pode existir em Freelancer e Contracted simultaneamente*

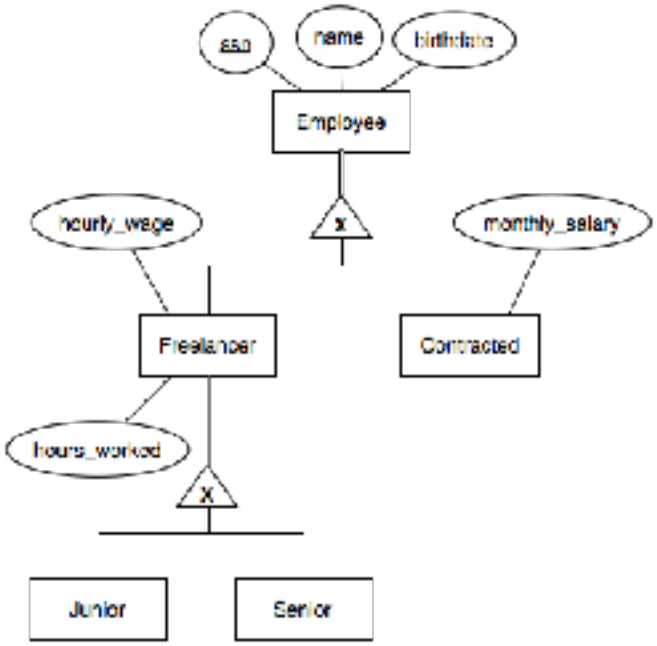
contracted(ssn, monthly_salary)

- *ssn: FK(Employee.ssn)*

freelancer(ssn, hours_worked, hourly_wages)

- *ssn: FK(Employee.ssn)*

Especializações Aninhadas



employee(<u>ssn</u>, name)

- *RI-1: ssn tem de existir em Freelancer e/ou em Contracted*
- *RI-2: ssn não pode existir em Freelancer e Contracted simultaneamente*

contracted(<u>ssn</u>, monthly_salary)

- *ssn: FK(employee.ssn)*

freelancer(<u>ssn</u>, hours_worked, hourly_wages)

- *ssn: FK(employee.ssn)*
- *RI-3: ssn não pode existir em Junior e Senior simultaneamente*

junior(<u>ssn</u>)

- *ssn: FK(freelancer.ssn)*

senior(<u>ssn</u>)

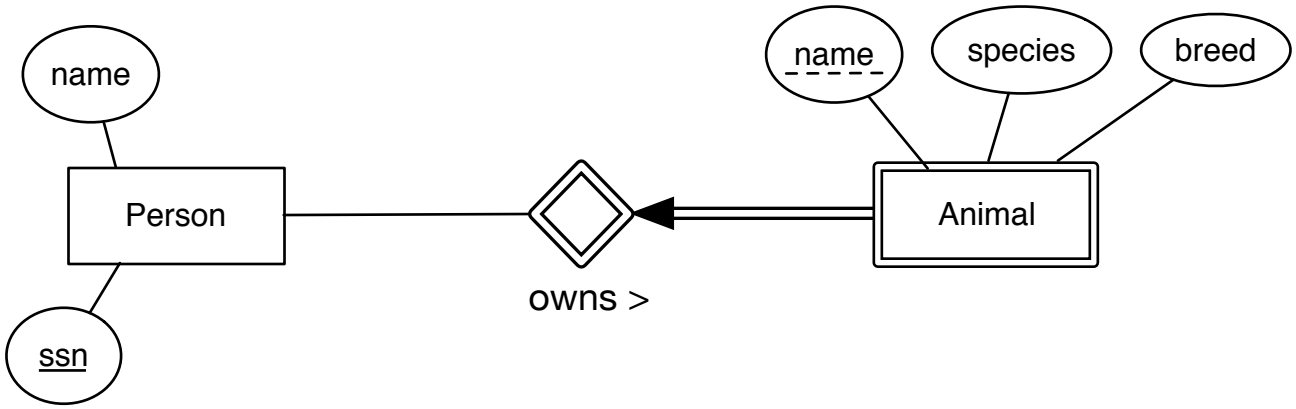
- *ssn: FK(freelancer.ssn)*

Tradução de Entidades Fracas

Entidades Fracas

- As **Entidades Fracas** originam uma relação que tem uma chave composta por:
  - Chave da relação que corresponde à entidade forte
  - Chave parcial especificada
  - Atributos da entidade fraca
- Dispensa conversão da associação identificadora em tabela

Entidade Fraca: Exemplo



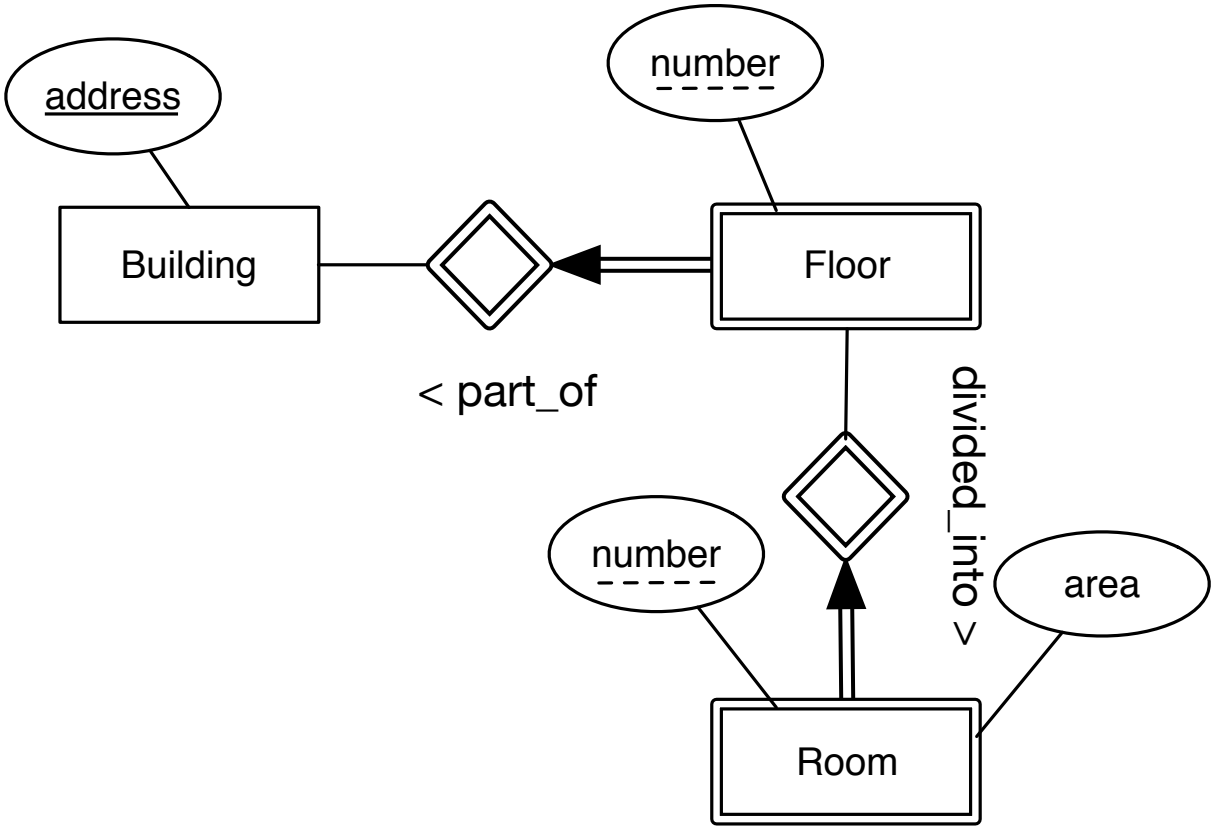
Person(ssn, name)

Animal*(ssn, name, species, breed)*

- *ssn: FK(Person)*

Exercício A.

Converta para relacional



Solução

Building(address)

Floor*(address, number, net_area)*

- *address: FK(Building.address)*

Room(address, floor_number, room_number)

- *address: FK(Floor.address)*
- *floor_number: FK(Floor.number)*

Tradução de Agregações

Agregações

- A agregação é mapeada como um associação onde:
  - **Primeiro**: mapeia-se as entidades exteriores
  - **Segundo:** mapeia-se interior da agregação
  - **Terceiro**: Mapeia-se a associação (da agregação) contra a relação resultante da associação que está a ser agregada

employee(ssn, name, birthdate)

project(pid, start_date)
- *RI-1: Every pid must exist in the relation executes*

department(did, name)

executes(pid, did)
- *pid: FK(Project)*
- *did: FK(Department)*

monitors(ssn, pid, did, until)
- *ssn: FK(Employee)*
- *pid, did: FK(executes.pid, executes.did)*