

# Introdução ao Modelo Relacional

## ■ Tópicos

- ✦ Estrutura das Bases de Dados Relacionais
- ✦ Redução ao modelo relacional de um Esquema ER

## ■ Bibliografia

- ✦ Secções 2.1 a 2.4 e 7.6 do livro recomendado
- ✦ Ou, não cobrindo necessariamente toda a matérias:
  - ❖ Secções 3.1 a 3.3 do livro de Ramakrishnan et al.
  - ❖ Capítulo 1 do livro de David Maier

# Modelo Relacional

- Os modelos ER ajudam na modelação dos dados
- Mas não ajudarão como modelo para armazenamento de dados e posterior “tratamento”
  - ✦ Como é que os dados estão armazenados?
  - ✦ Como consultar os dados?
  - ✦ Como alterar os dados?
- Ajudava mais ver os dados organizados em tabelas ou, usando nomenclatura matemática, em relações
- **Modelo Relacional**

# Atributos

- Todo o atributo de uma relação tem um nome
- O conjunto de valores que um atributo pode tomar é chamado de **domínio** do atributo.
- Normalmente, obriga-se a que os valores dos atributos sejam **atômicos**, ou seja, indivisíveis:
  - ✦ E.g. atributos multivalor não são atômicos
  - ✦ E.g. atributos compostos não são atômicos
- O valor especial *null* pertence a todos os domínios
- O valor *null* causa complicações na definição de muitas operações
  - ✦ Ignoraremos o efeito dos valores nulos em grande parte da apresentação mas consideraremos posteriormente as suas implicações

# Esquema de Relação e Instância

- $A_1, A_2, \dots, A_n$  são (nomes de) *atributos* com *domínios*  $D_1, D_2, \dots, D_n$ , respectivamente.
  - ★ E.g. *customer-name* com domínio {Jones, Smith, Curry, Lindsay}  
*customer-street* com domínio {Main, North, Park}  
*customer-city* com domínio {Harrison, Rye, Pittsfield}
- $R = (A_1, A_2, \dots, A_n)$  é um *esquema de relação*
  - ★ E.g. *Customer-schema* = (*customer-name*, *customer-street*, *customer-city*)
- $r(R)$  é uma *relação* no *esquema de relação*  $R$ 
  - ★ E.g. *customer(Customer-schema)* ou *vip(Customer-schema)*
- Formalmente, dados os conjuntos  $D_1, D_2, \dots, D_n$ , uma *relação*  $r$  é um subconjunto do produto cartesiano
$$D_1 \times D_2 \times \dots \times D_n$$
  - ★ i.e., uma relação é um conjunto de tuplos  $(a_1, a_2, \dots, a_n)$  em que  $a_i \in D_i$
  - ★ E.g. *customer* = {(Jones, Main, Harrison), (Smith, North, Rye), (Curry, North, Rye), (Lindsay, Park, Pittsfield)}

# Instância de Relação

- Os valores (*instância*) de uma relação podem ser representados por uma tabela
- Um elemento  $t$  de  $r$  é um *tuplo*, representado por uma *linha* da tabela

<b>id</b>	<b>nome</b>	<b>morada</b>	<b>cidade</b>
13123	Luís Trindade	Rue Central	Paris
43242	Pedro Silva	Rua da Sofia	Coimbra
36645	Joana Sobral	Rua D <sup>a</sup> Maria	Coimbra
21313	Susana Dias	Av do Brasil	Lisboa

# As relações não estão ordenadas

- A ordem dos tuplos é irrelevante (os tuplos podem ser armazenadas segundo qualquer ordem)
- E.g. relação *account* com os tuplos desordenados

<i>account-number</i>	<i>branch-name</i>	<i>balance</i>
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

# Base de Dados Relacional

- Uma base de dados relacional é constituída por diversas relações
- A informação acerca de uma empresa é dividida em partes, em que cada relação armazena uma parte dessa informação
  - ★ E.g.: *account*: armazena informação acerca de contas
  - depositor*: regista os clientes titulares das contas
  - customer*: guarda informação acerca de clientes
- O armazenamento da informação numa única relação *bank(account-number, balance, customer-name, customer-street, ...)* originaria
  - ★ Repetição de informação (e.g. dois clientes que detêm uma conta)
  - ★ A necessidade de valores nulos (e.g. para representar um cliente que não possui uma conta)
- A teoria da normalização (capítulo mais à frente!) especifica como se devem desenhar/analisar esquemas de relação

## A relação *customer*

<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

## *A relação depositador*

<i>customer-name</i>	<i>account-number</i>
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

# Chaves

- Seja  $K \subseteq R$
- $K$  é uma **super-chave** de  $R$  se os valores de  $K$  são suficientes para identificar todos os tuplos de cada relação  $r(R)$  possível. Por “relação possível” entende-se uma instância  $r$  que pode existir na base de dados que estamos a modelar.  
Exemplo:  $\{customer-name, customer-street\}$  e  $\{customer-name\}$   
são ambas super-chaves de *Customer*, se não for possível dois clientes terem o mesmo nome.
  - ★ Por vezes, em vez de *customer-name*, é utilizado um atributo *customer-id* para identificar univocamente os clientes. Omitiremos esse atributo para simplificar os exemplos.

# Chaves (cont.)

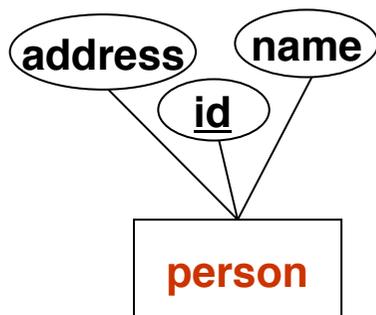
- $K$  é uma **chave candidata** se  $K$  é minimal  
Exemplo:  $\{customer-name\}$  é uma chave candidata para *Customer*, dado ser uma super-chave (assumindo que dois clientes não podem ter o mesmo nome), e nenhum subconjunto dela ser uma super-chave.
- **Chave primária**: uma chave candidata que é escolhida com o objectivo de identificar os tuplos numa relação.
  - ✦ Devem ser escolhidos atributos cujos valores nunca, ou raramente, variem.
  - ✦ Por exemplo, o e-mail é único, mas normalmente pode variar.

## Derivação de relações (tabelas) a partir de um DER

- Uma base de dados que seja representável por um DER pode ser também representada por intermédio de um conjunto de relações (tabelas).
- A conversão de um DER para um conjunto de relações (tabelas) constitui a base para a derivação do desenho de uma base de dados relacional a partir de um DER
- Para cada conjunto de entidades e para cada conjunto de relações do modelo ER gera-se uma única relação (tabela) com o nome do conjunto de entidades ou conjunto de relações respetivo.

## Conjuntos de Entidades como Tabelas

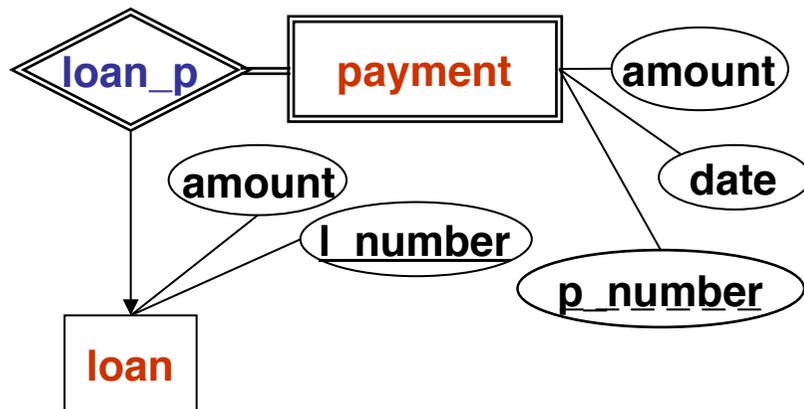
- Um conjunto de entidades forte reduz-se a uma relação (tabela) com exatamente os mesmos atributos
- Por exemplo, o conjunto de entidades person dá origem à relação (tabela) **person(id,name,address)**



<b>id</b>	<b>name</b>	<b>address</b>
13123	Luís Trindade	Paris
43242	Pedro Silva	Coimbra
36645	Joana Sobral	Coimbra
21313	Susana Dias	Lisboa

# Conjuntos de Entidades Fraco

- Um conjunto de entidades fraco é representado por uma relação (tabela) que tem como atributos:
  - ✦ a chave primária dos conjuntos de entidades dominantes (ou identificadores);
  - ✦ os restantes atributos para do conjunto de entidades fraco.
- Por exemplo, o conjunto de entidades fraco payment dá origem à relação (tabela) **payment(p\_number, l\_number, amount, date)**

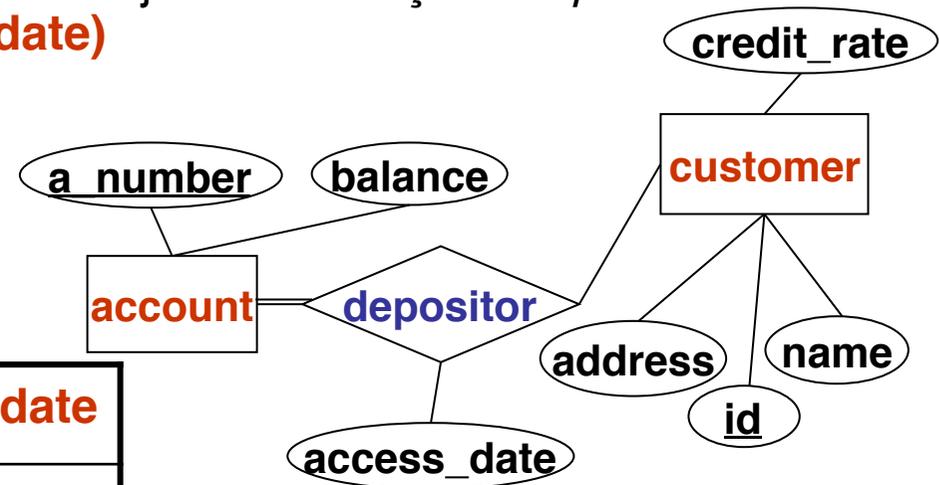


p_number	l_number	amount	date
1	L1233	2000	10-10-2005
2	L1234	1000	10-11-2005
2	L1233	1000	10-11-2005
3	L1433	500	22-12-2005

# Conjuntos de Relações

- Um conjunto de relações é representado por uma relação (tabela) com atributos para as chaves primárias dos conjuntos de entidades participantes, e com atributos adicionais para os atributos próprios (ou descritivos) do conjunto de relações.
- Por exemplo a relação (tabela) para o conjunto de relações *depositor* é **depositor(a\_number, id, access\_date)**

<b>a_number</b>	<b>id</b>	<b>access_date</b>
A122	Luís Trindade	12-12-2005
A133	Luís Trindade	14-12-2005
A122	Joana Sobral	13-11-2005
A144	Susana Dias	10-13-2004

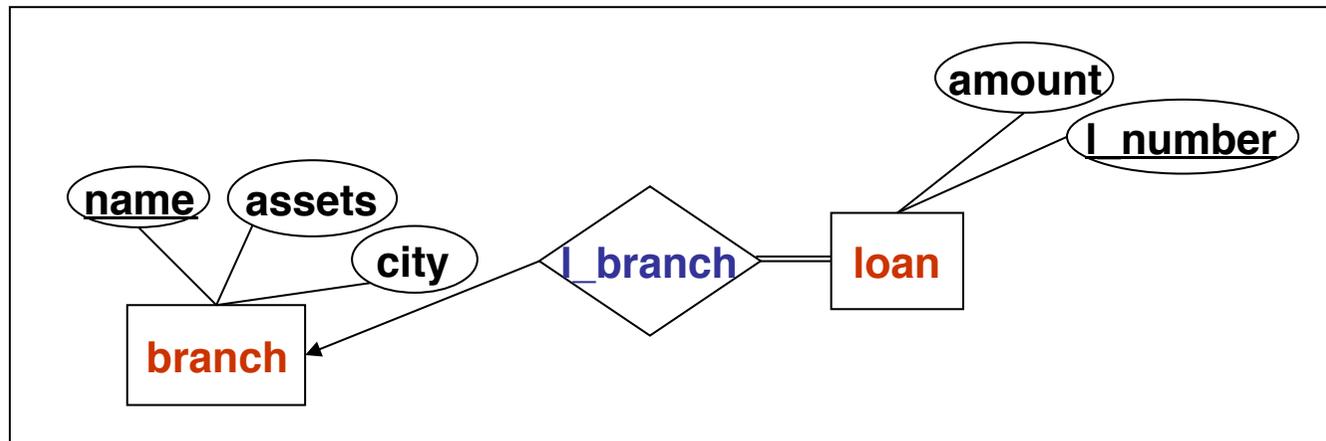


# Determinação de Chaves a partir do DER

- **Conjunto de entidades fortes.** A chave primária do conjunto de entidades é a chave primária da relação (tabela).
- **Conjunto de entidades fracas.** A chave primária da relação (tabela) consiste na união da(s) chave(s) primária(s) do(s) conjunto(s) de entidades dominante(s) com o discriminante do conjunto de entidades fracas.
- **Conjunto de relações.** A união das chave primárias dos conjuntos de entidades relacionados é uma super-chave da relação (tabela).
  - ✦ Para conjuntos de relações binários um-para-muitos, a chave primária do lado “muitos” é a chave primária da relação (tabela).
  - ✦ Para conjuntos de relações um-para-um, a chave primária de qualquer um dos conjuntos de entidades é chave candidata da relação (tabela).
  - ✦ Para conjuntos de relações muitos-para-muitos, a união das chaves primárias é a chave primária da relação (tabela).

# Tabelas Redundantes

- Conjuntos de relações muitos-para-um e um-para-muitos, totais no lado *muitos* podem ser representados adicionando atributos extra ao lado muitos contendo a chave primária do outro conjunto participante.
- E.g.: Em vez de se criar uma relação (tabela) para o conjuntos de relações *l-branch*, adicionar um atributo *name* à relação (tabela) derivada a partir do conjunto de entidades *loan*, obtendo **loan(l\_number,amount,name)**



# Tabelas Redundantes

loan

<u>I_number</u>	amount
A122	10
A133	20
A123	15
A144	10

I\_branch

<u>I_number</u>	name
A122	Sete Rios
A133	Benfica
A123	Sete Rios
A144	Sete Rios

branch

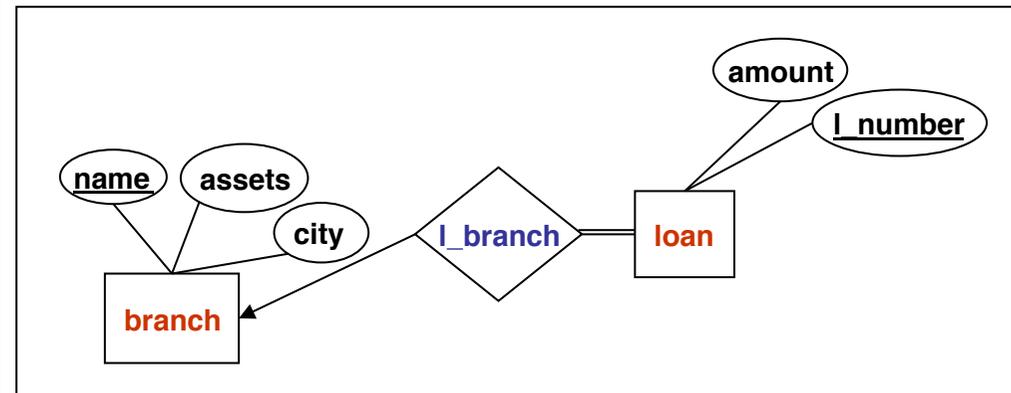
<u>name</u>	assets	city
Sete Rios	10000	Lisboa
Benfica	250000	Lisboa
Santa Clara	123000	Coimbra



loan



<u>I_number</u>	amount	name
A122	10	Sete Rios
A133	20	Benfica
A123	15	Sete Rios
A144	10	Sete Rios



# Redundância de Tabelas (Cont.)

- Se a participação é *parcial* no lado muitos, a substituição da relação (tabela) por atributos extra pode levar à ocorrência de **valores nulos**.
- Para conjuntos de relações um-para-um, qualquer dos lados pode receber a chave primária do outro lado.
- São redundantes as relações (tabelas) correspondentes aos conjuntos de relações entre o conjunto de entidades fracas e os seus conjuntos de entidades dominantes.
  - ★ E.g. A tabela *payment* já contém a informação que apareceria na tabela *loan\_p* (i.e., as colunas *l\_number* e *p\_number*).

# Derivação de Tabelas para a Especialização

## ■ Método 1:

- ✦ Formar uma relação (tabela) para a entidade de maior nível (mais geral)
- ✦ Criar uma relação (tabela) para cada conjunto de entidades de nível abaixo, incluindo a chave primária da entidade acima e os atributos locais.

<b>tabela</b>	<b>atributos</b>
person	id, name, address
customer	id, credit_rating
employee	id, salary

- ✦ Desvantagem: obter a informação acerca de *employee* (por exemplo) obriga à consulta de duas tabelas

# Derivação de Tabelas para a Especialização

Método 2:

- \* Formar uma relação (tabela) para cada conjunto de entidades com os atributos locais e herdados

<b>tabela</b>	<b>atributos</b>
person	id, name, address
customer	id, name, address, credit_rating
employee	id, name, address, salary

- \* Desvantagem:
  - ❖ *name e address* podem ser duplicados para pessoas que são clientes e/ou empregados

Se a especialização é total e não há relações com *person*, não há necessidade de criar uma relação (tabela) para a entidade mais geral (*person*)

- \* Desvantagem:
  - ❖ *address* podem ser duplicados para pessoas que são simultaneamente clientes e empregados

- Método a ser usado apenas quando a especialização é total, disjunta, e não há relações envolvendo o conjunto de entidades mais geral.

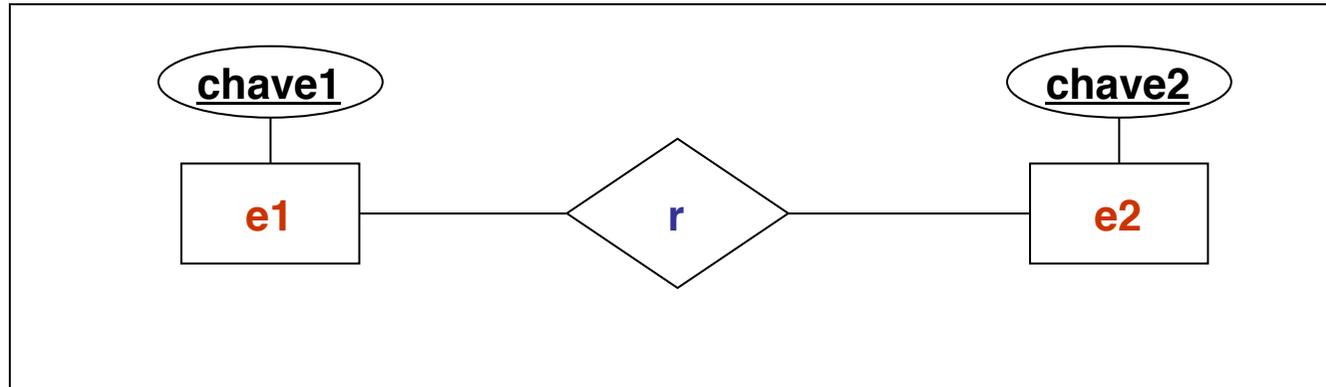
# Relações Correspondentes à Agregação

- Tratar o conjunto de relações agregado como se se tratasse de um conjunto de entidades, sendo a sua chave a chave do conjunto de relações.

# Chaves Estrangeiras

- *Um esquema de relação pode ter um (ou mais) atributo(s) que corresponda(m) à chave primária de outra relação. Esses atributos são designados por **chaves estrangeira (ou externa)**.*
  - ★ Exemplo: *customer-name* e *account\_number* da relação *depositor* são chaves estrangeira de *customer* e *account*, respectivamente.
- As chaves estrangeiras impõem restrições sobre os dados (integridade referencial ou de referência)
  - ★ Apenas os valores que ocorrem na **relação referenciada** podem ocorrer nos atributos da chave estrangeira da **relação referenciadora**
  - ★ No exemplo, em cada momento, na relação *depositor* o atributo *account-number* só pode conter valores que existam nesse momento no atributo *account-number* de algum elemento da relação *account*
- No modelo ER isto não era necessário, pois nos conjuntos relação só podiam participar entidades dos conjuntos que essa relação unia

# Integridade de referência derivada de ER

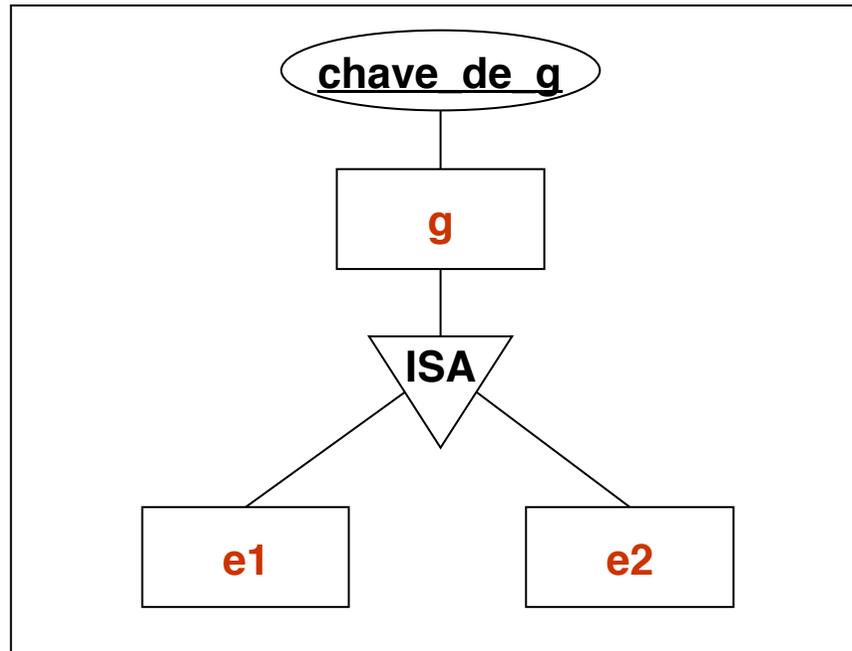


- Independentemente da cardinalidade, para o conjunto de relações  $r$  criar a tabela:

**$r(chave1, chave2)$**

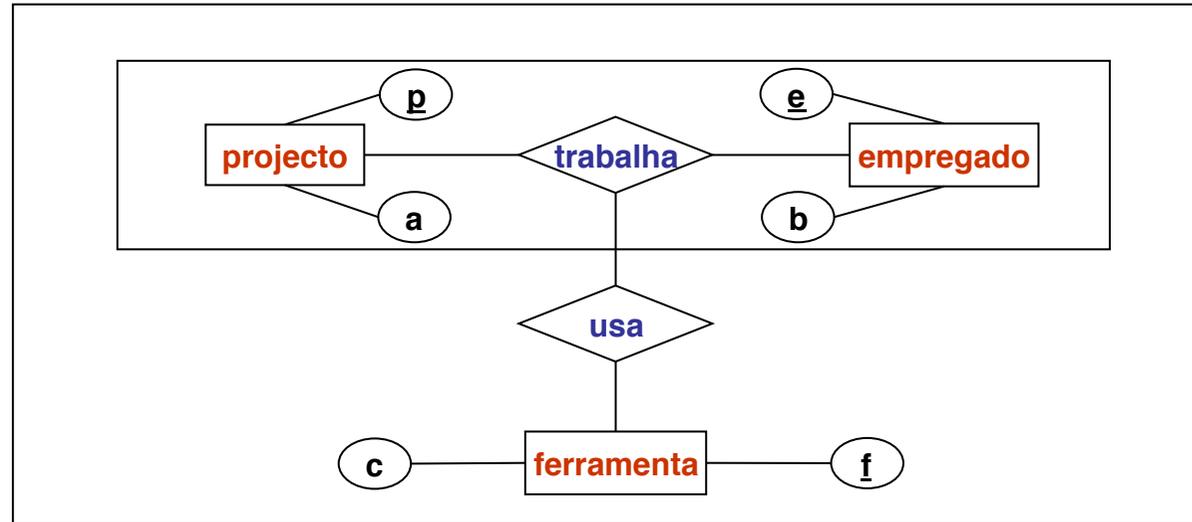
- \*  $chave1$  em  $r$  é chave estrangeira referindo  $e1$
- \*  $chave2$  em  $r$  é chave estrangeira referindo  $e2$

# Integridade de referência e ER



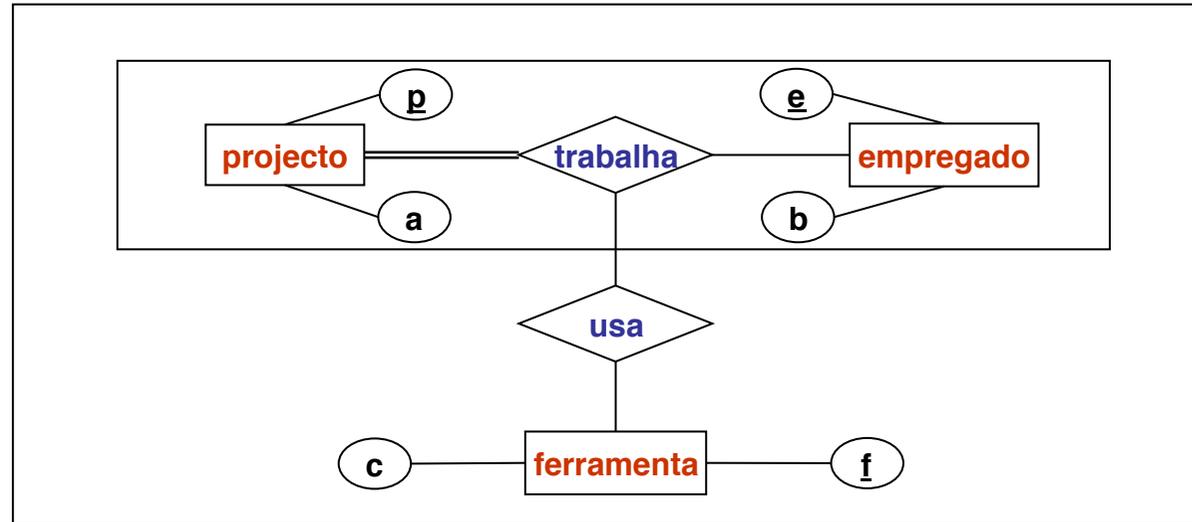
- A relação correspondente ao conjunto de entidades **e1** (resp. **e2**) tem como atributo **chave\_de\_g**, para além dos atributos locais de **e1** (resp. **e2**).
  - \* **chave\_de\_g** é chave primária de **e1** (resp. **e2**)
  - \* **chave\_de\_g** em **e1** (resp. **e2**) é chave estrangeira referindo **g**
- Nada impõe (ainda) sobre restrições de pertença ou completude!!
  - \* Para a pertença e completude, joga-se com a restrição do domínio, não admitindo valores *null* – veremos em detalhe (bem) mais à frente

# Integridade de referência e ER



- **projecto(p,a)**
- **empregado(e,b)**
- **ferramenta(f,c)**
- **trabalha(p,e)**
  - ★ **p** é chave estrangeira de **projecto**
  - ★ **e** é chave estrangeira de **empregado**
- **usa(p,e,f)**
  - ★ **f** é chave estrangeira de **ferramenta**
  - ★ **(p,e)** é chave estrangeira de **trabalha**

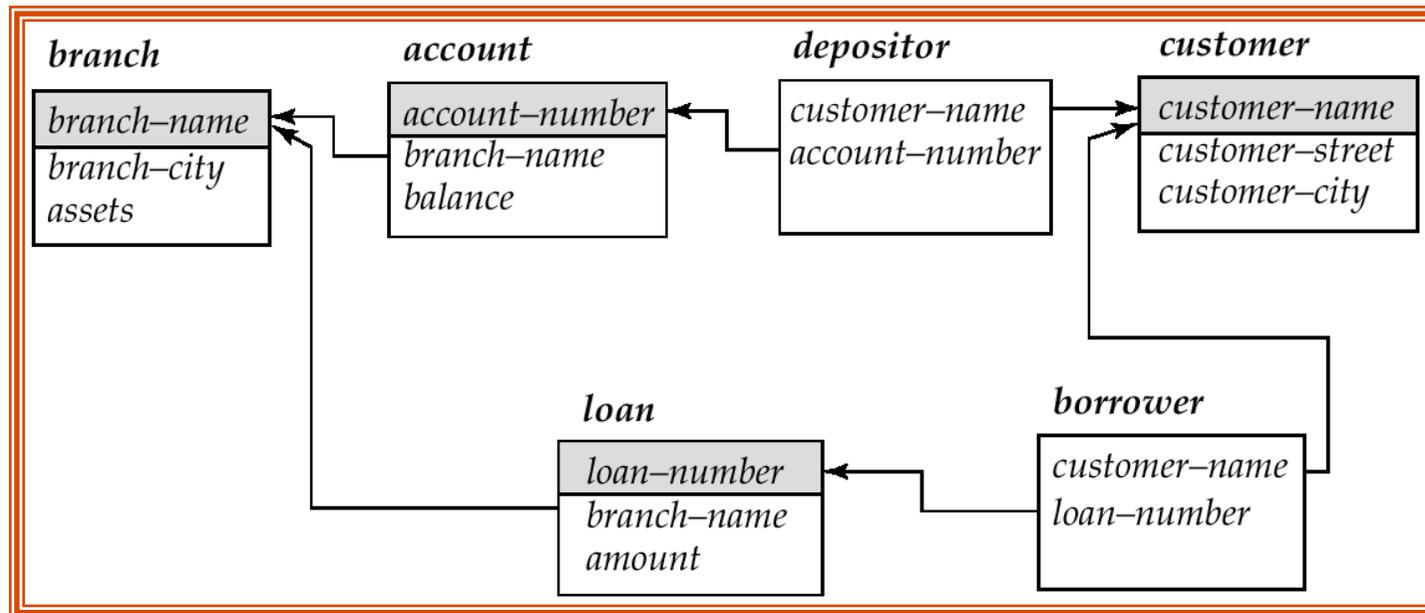
# Integridade de referência e ER



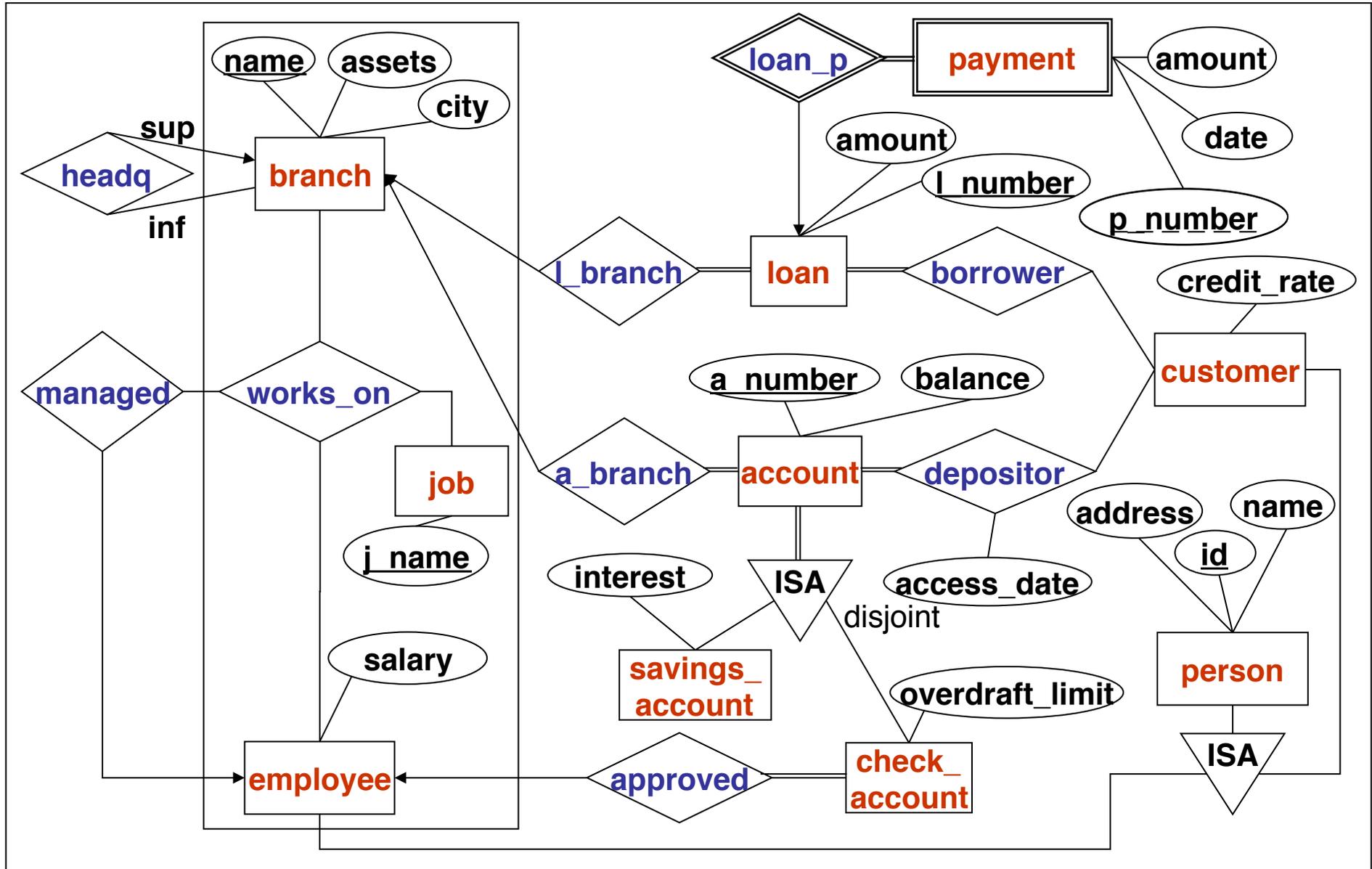
- E como tratar restrições de participação totais?
- O standard não impõem nada, mas podíamos adicionar uma chave estrangeira no sentido inverso:
  - ★ **p** em **projeto** é chave estrangeira referindo **trabalha**.
- (Bem) mais a frente vamos ver quando isso pode ser útil.

# Diagrama de Esquema

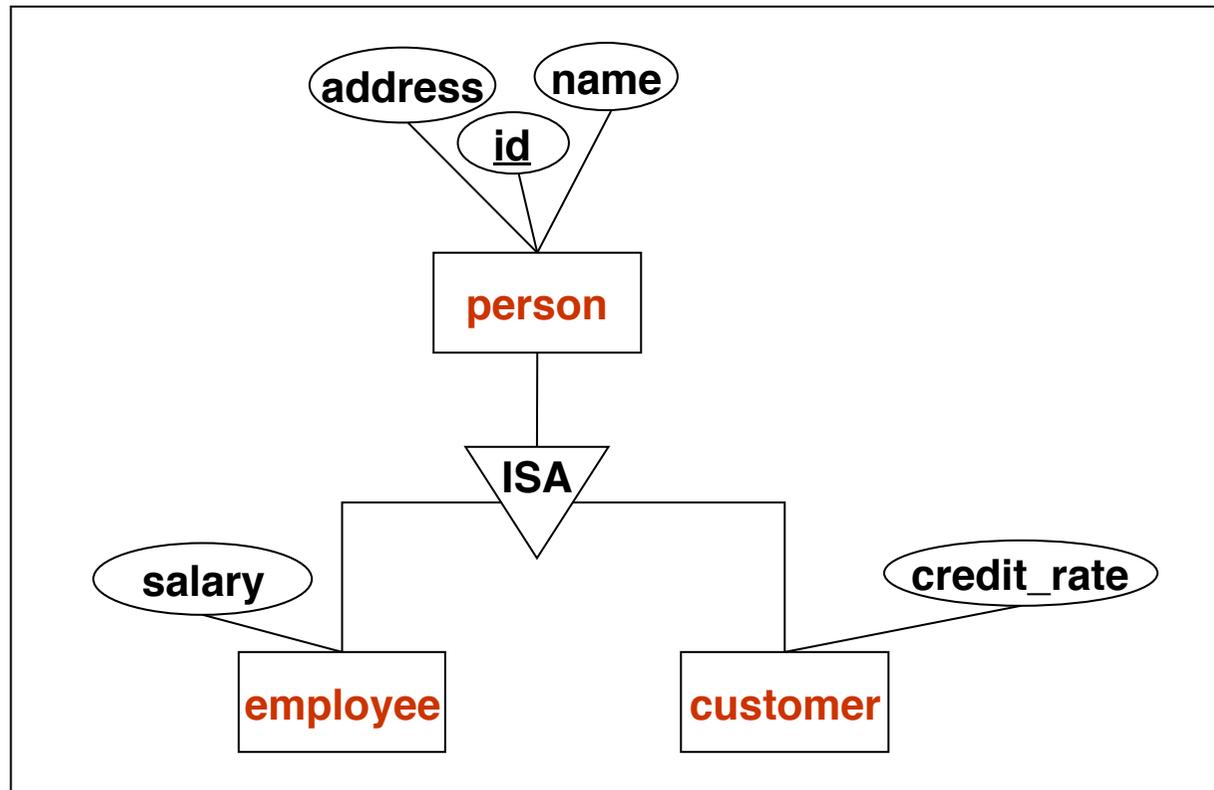
- Representação esquemática de uma Base de Dados: **relações**, **atributos**, **chaves primárias** e **chaves estrangeiras**.



# DER de um Banco

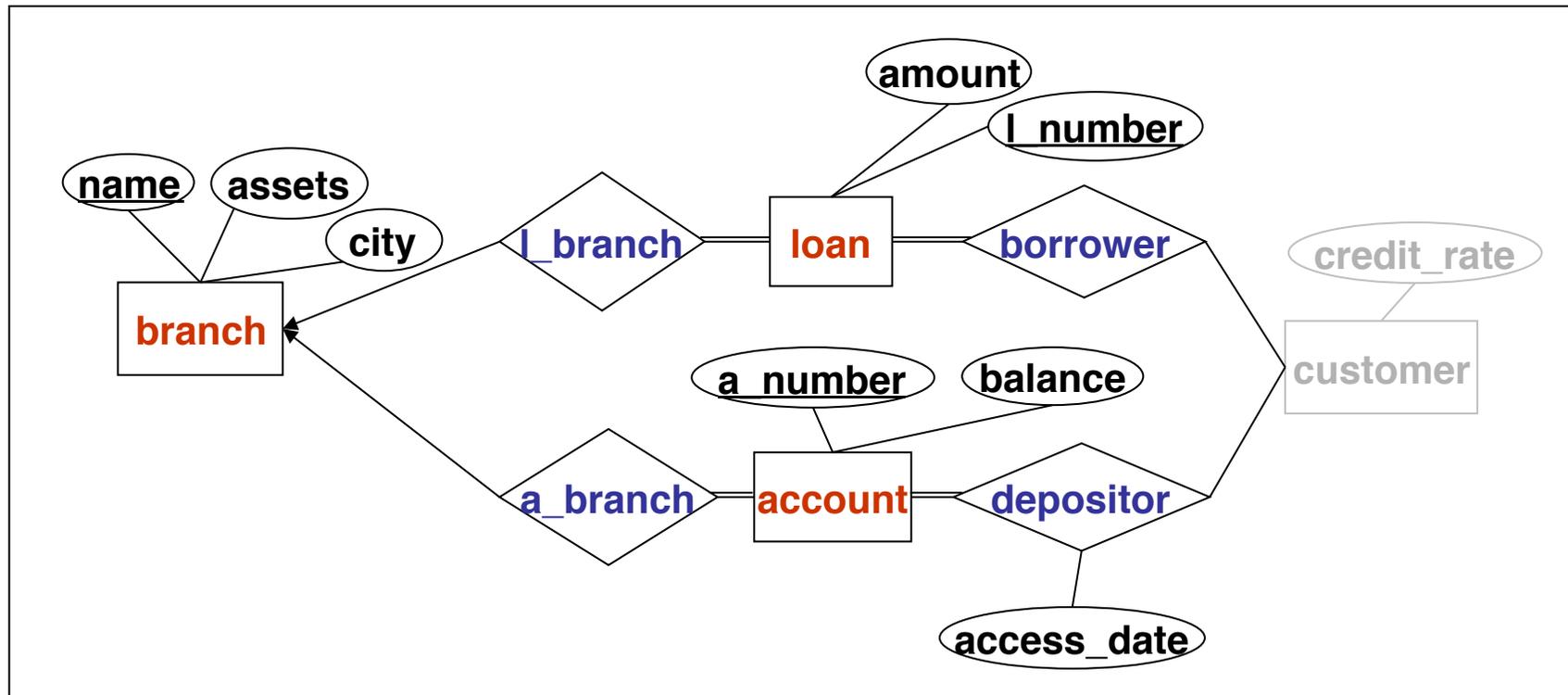


# Conversão em Esquemas de Relação do DER de um Banco



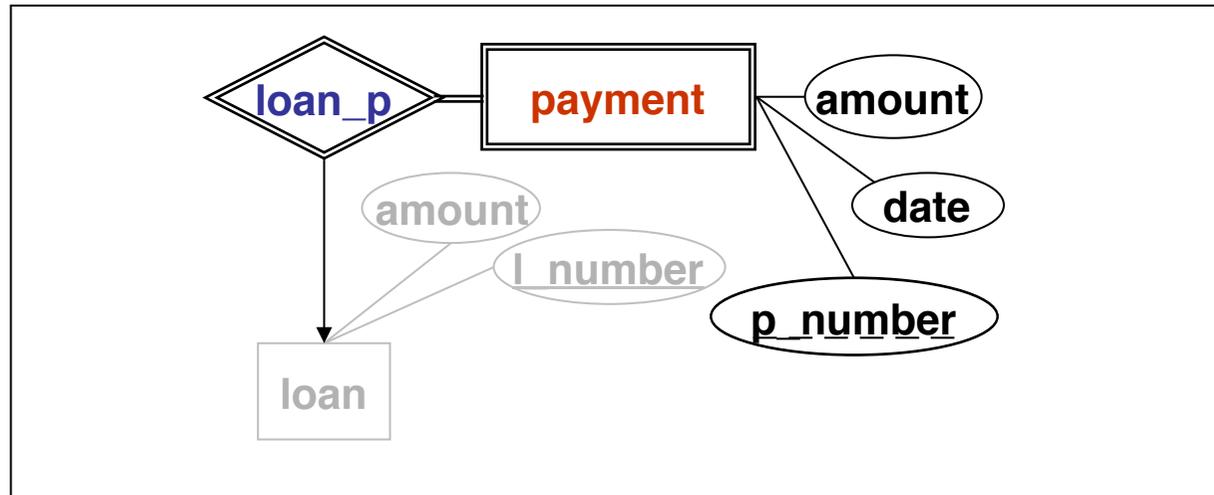
- *person(id, name, address)*
- *customer(id, credit\_rate)* id é chave estrangeira de *person*
- *employee(id, salary)* id é chave estrangeira de *person*

# Conversão em Esquemas de Relação do DER de um Banco



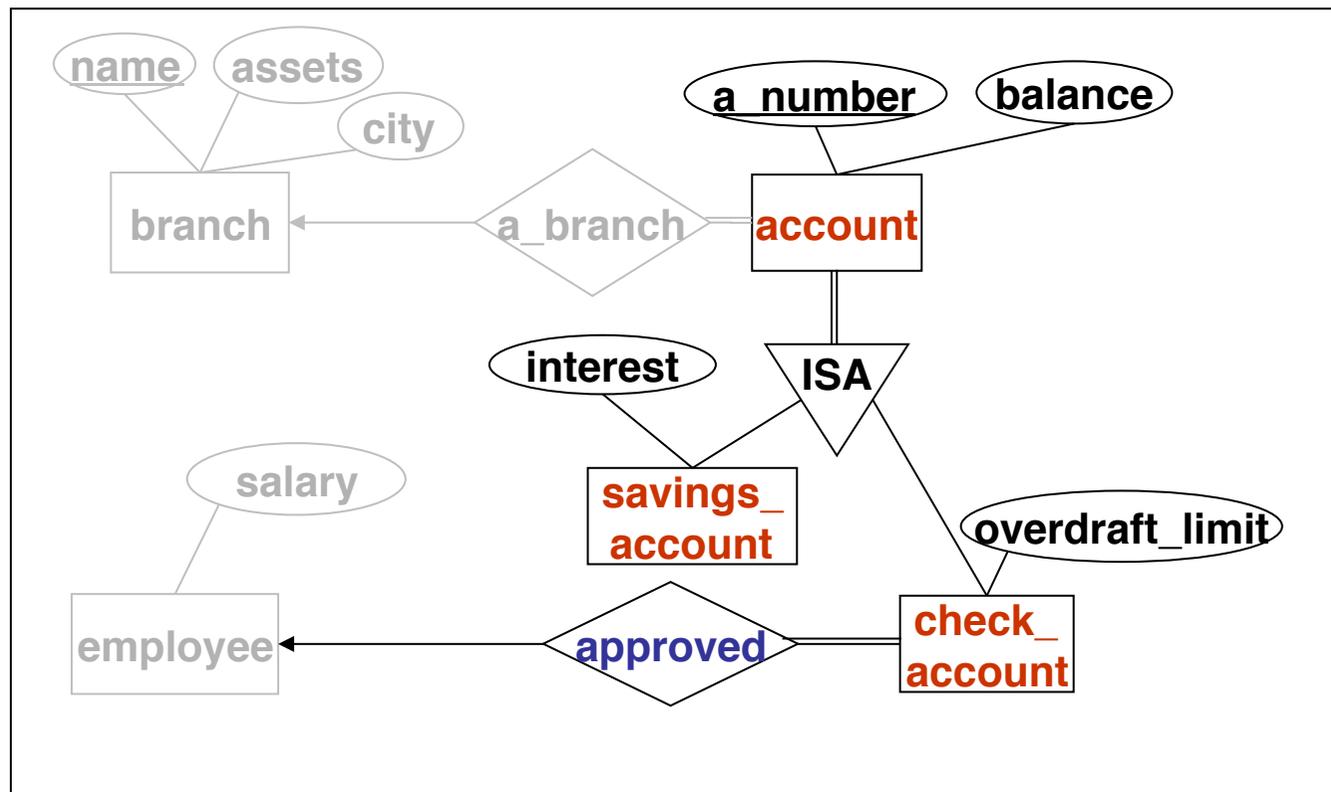
- branch(name,assets,city)
- loan(l\_number,amount,name) name é chave estrangeira de branch (e, btw, não pode ter valores null)
- account(a\_number,balance,name) name é chave estrangeira de branch (e, btw, não pode ter valores null)
- borrower(l\_number,id) id é chave estrangeira de cliente; l\_number é chave estrangeira de loan
- depositor(a\_number,id, access\_date) id é chave estrangeira de cliente; a\_number é chave estrangeira de account (e esta como fariam para garantir que é total? – veremos lá mais para a frente...)

# Conversão em Esquemas de Relação do DER de um Banco



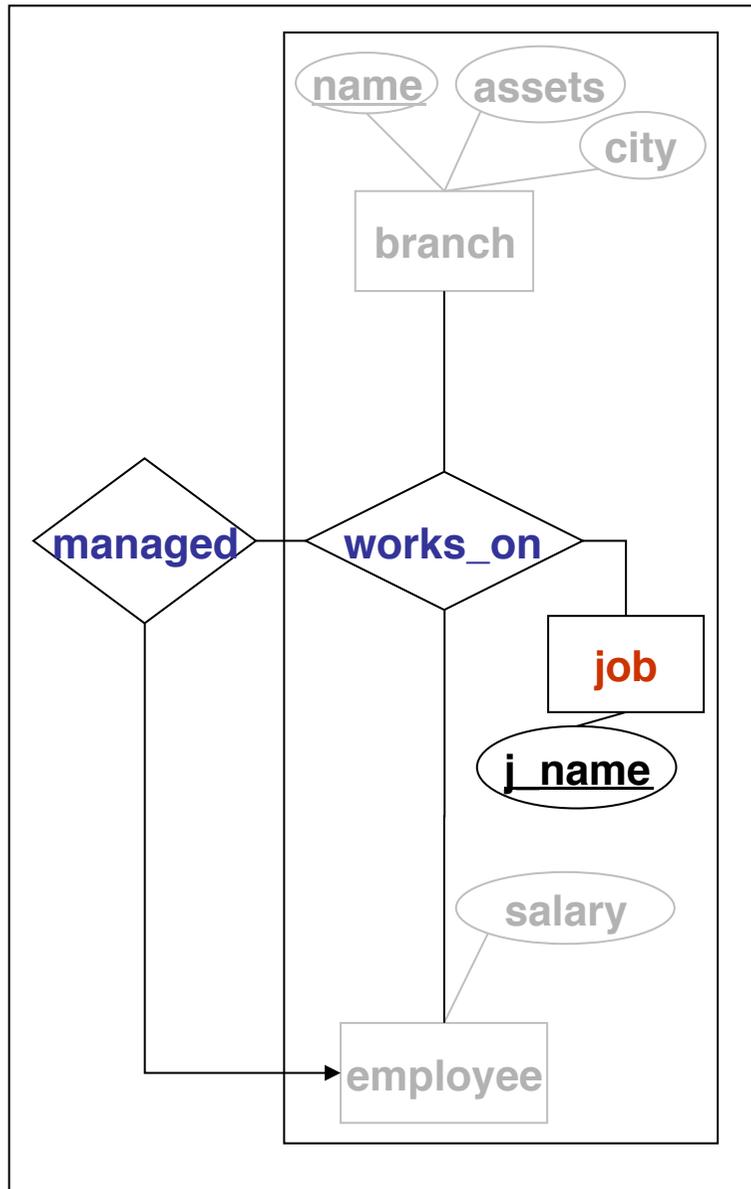
- payment(p\_number, l\_number, amount, date) l\_number é chave estrangeira de loan

# Conversão em Esquemas de Relação do DER de um Banco



- **account(a\_number,balance,name)** *name* é chave estrangeira de **branch**
- **savings\_account(a\_number,interest)** *a\_number* é chave estrangeira de **account**
- **check\_account(a\_number,overdraft\_limit,id)** *a\_number* é chave estrangeira de **account**, *id* é chave estrangeira de **employee**

# Conversão em Esquemas de Relação do DER de um Banco

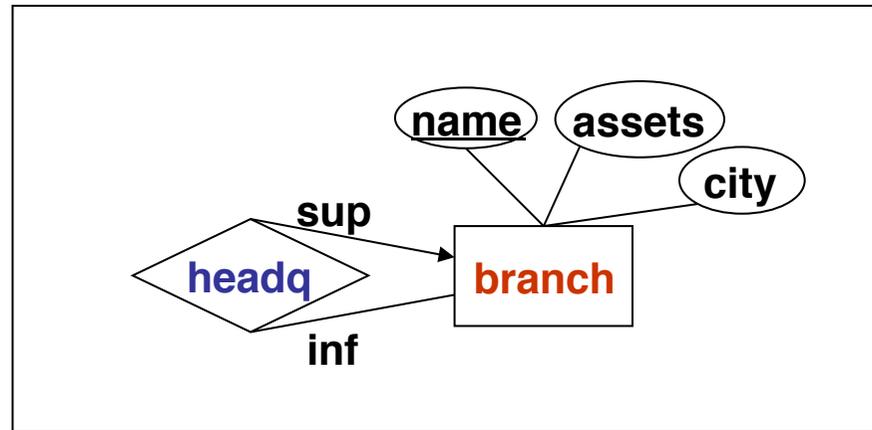


- `job(j_name)`
- `works_on(id,j_name,name)` `id` é chave estrangeira de `employee`, `j_name` é chave estrangeira de `job`, `name` é chave estrangeira de `branch`
- `managed(id,j_name,name,id.manager)` `(id,j_name,name)` é chave estrangeira de `works_on`; `id.manager` é chave estrangeira de `employee`

ou (ver.2)

- `job(j_name)`
- `works_on(id,j_name,name,manager.id)` `id` é chave estrangeira de `employee`; `j_name` é chave estrangeira de `job`; `name` é chave estrangeira de `branch`; `id.manager` é chave estrangeira de `employee`

# Conversão em Esquemas de Relação do DER de um Banco



- `branch(name,assets,city)`
- `headq(inf.name,sup.name)` `inf.name` é chave estrangeira de `branch`; `sup.name` é chave estrangeira de `branch`

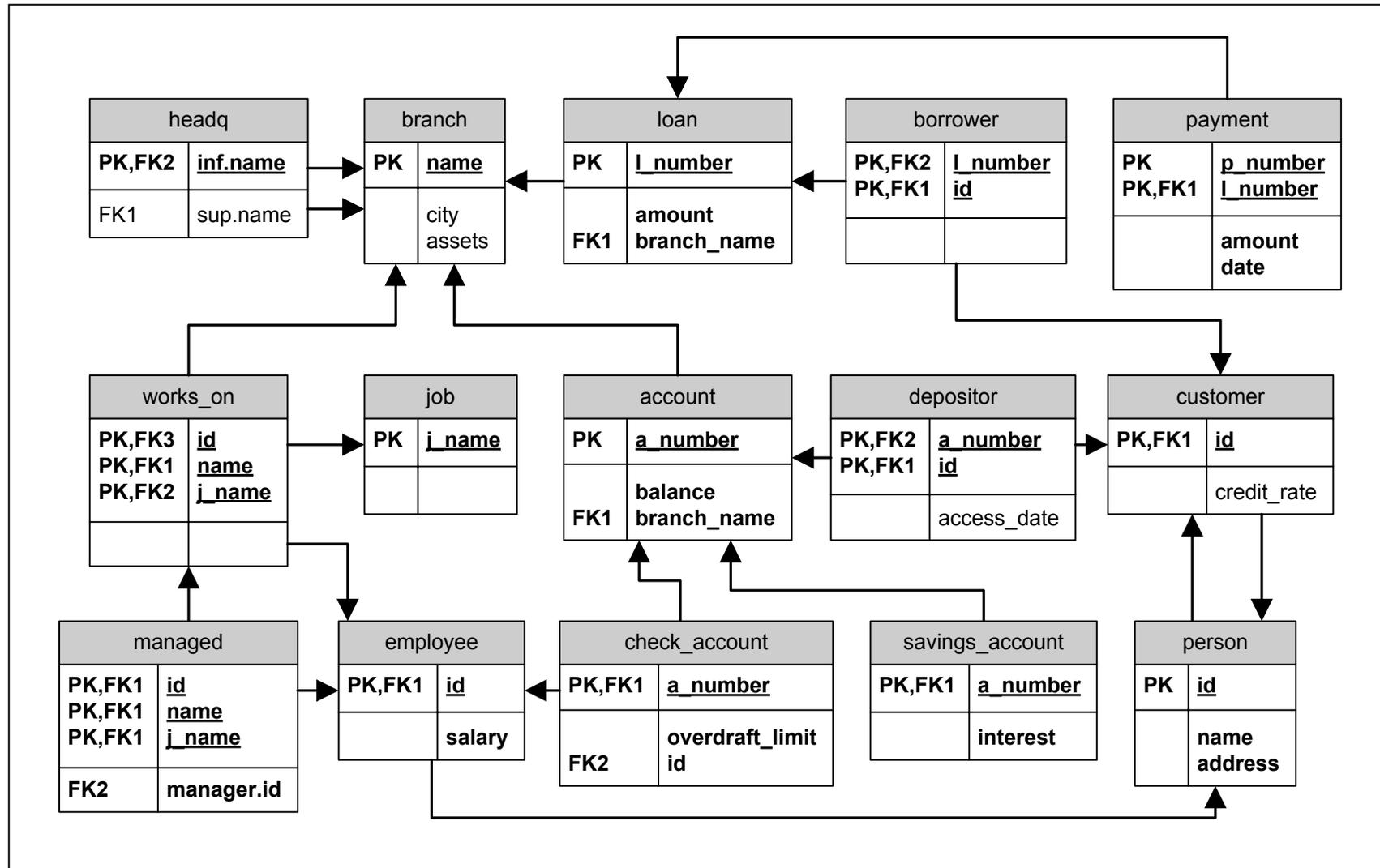
ou (ver.2)

- `branch(name,assets,city,sup)` `sup` é chave estrangeira de `branch`

# Conjunto de tabelas

- *person(id,name,address)*
- *customer(id,credit\_rate)* id é chave estrangeira de *person*
- *employee(id,salary)* id é chave estrangeira de *person*
- *branch(name,assets,city)*
- *loan(l\_number,amount,name)* name é chave estrangeira de *branch*
- *account(a\_number,balance,name)* name é chave estrangeira de *branch*
- *borrower(l\_number,id)* id é chave estrangeira de cliente; l\_number é chave estrangeira de *loan*
- *depositor(a\_number,id,access\_date)* id é chave estrangeira de cliente; a\_number é chave estrangeira de *account*
- *payment(p\_number,l\_number,amount,date)* l\_number é chave estrangeira de *loan*
- *savings\_account(a\_number,interest)* a\_number é chave estrangeira de *account*
- *check\_account(a\_number,overdraft\_limit,id)* a\_number é chave estrangeira de *account*, id é chave estrangeira de *employee*
- *job(j\_name)*
- *works\_on(id,j\_name,name)* id é chave externa de *employee*, j\_name é chave estrangeira de *job*, name é chave estrangeira de *branch*
- *managed(id,j\_name,name,id.manager)* (id,j\_name,name) é chave estrangeira de *works\_on*; id.manager é chave estrangeira de *employee*
- *headq(inf.name,sup.name)* inf.name é chave estrangeira de *branch*; sup.name é chave estrangeira de *branch*

# Esquema da BD de um banco



# Conjunto de tabelas (ver.2)

- *person(id,name,address)*
- *customer(id,credit\_rate)* id é chave estrangeira de *person*
- *employee(id,salary)* id é chave estrangeira de *person*
- *loan(l\_number,amount,name)* name é chave estrangeira de *branch*
- *account(a\_number,balance,name)* name é chave estrangeira de *branch*
- *borrower(l\_number,id)* id é chave estrangeira de cliente; l\_number é chave estrangeira de *loan*
- *depositor(a\_number,id,access\_date)* id é chave estrangeira de cliente; a\_number é chave estrangeira de *account*
- *payment(p\_number,l\_number,amount,date)* l\_number é chave estrangeira de *loan*
- *savings\_account(a\_number,interest)* a\_number é chave estrangeira de *account*
- *check\_account(a\_number,overdraft\_limit,id)* a\_number é chave estrangeira de *account*, id é chave estrangeira de *employee*
- *job(j\_name)*
- *works\_on(id,j\_name,name,manager.id)* id é chave externa de *employee*; j\_name é chave estrangeira de *job*; name é chave estrangeira de *branch*; id.manager é chave estrangeira de *employee*
- *branch(name,assets,city,sup)* sup é chave estrangeira de *branch*

# Esquema da BD de um banco (ver.2)

