



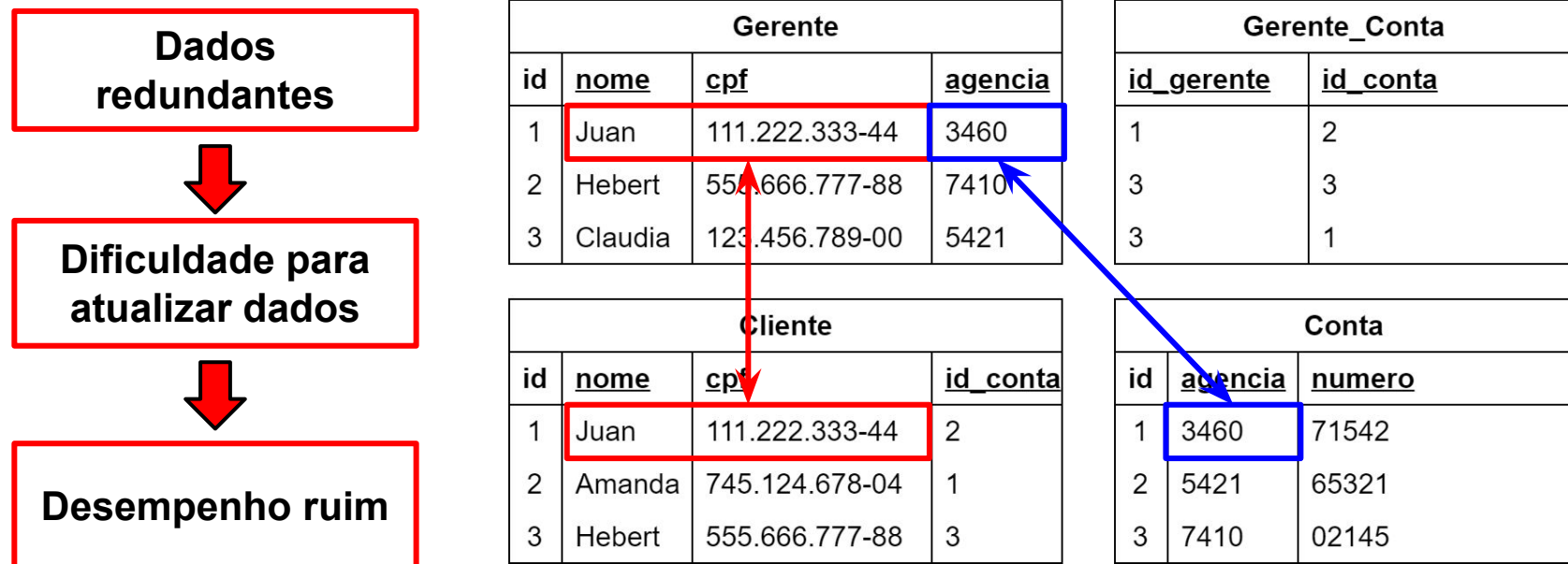
Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA
Departamento de Informática
Integrado / Análise e Desenvolvimento de Sistemas / Licenciatura em Computação

Dependências Funcionais

André L. R. Madureira <andre.madureira@ifba.edu.br>
Doutorando em Ciência da Computação (UFBA)
Mestre em Ciência da Computação (UFBA)
Engenheiro da Computação (UFBA)

Importância e Motivação

- Bancos de dados construídos usando o bom senso do projetista podem ter uma qualidade de projeto ruim:



Abordagens de Projetos de Bancos de Dados

- Bottom-Up (projeto por síntese)
 - Baseado no relacionamento entre atributos individuais
 - Grande número de atributos relacionamentos binários
- Top-Down (projeto por análise)
 - Atributos são agrupados em relações
 - Análise individual e coletiva de relações

Diretrizes de Projeto (boas práticas)

- **Semântica clara** dos atributos e tabelas
- **Não combinar atributos de entidades diferentes** em uma mesma relação
- **Reduzir valores NULL** (vazio) nas tuplas
- **Reduzir informações redundantes** nas tuplas
- **Evitar** relações com **tuplas falsas**

Semântica clara dos atributos e tabelas

- Um bom projeto de banco de dados deve ser fácil de entender
 - **Ex:** cod_produto, cpf, agencia, numero_conta
- Devemos evitar ambiguidades e nomes que dificultem o entendimento
 - **Ex:** cod_depto_func (codigo do departamento ou funcionario ?), Dnome (D significa o que? Departamento?)

Não combinar atributos de entidades diferentes

- Quando combinamos **atributos de entidades diferentes em uma mesma relação** temos:
 - Ambiguidade semântica
 - **Ex:** cod_depto_func (codigo do depto ou do funcionário?)
 - Redundância de dados
 - **Ex:** nome de uma pessoa repetido em tabelas diferentes
 - Dificuldade para atualizar dados
 - **Ex:** atualizar nome de uma pessoa em várias tabelas diferentes

Não combinar atributos de entidades diferentes

Gerente			
id	nome	cpf	agencia
1	Juan	111.222.333-44	3460
2	Hebert	555.666.777-88	7410
3	Claudia	123.456.789-00	5421

Gerente_Conta	
id_gerente	id_conta
1	2
3	3
3	1

Cliente			
id	nome	cpf	id_conta
1	Juan	111.222.333-44	2
2	Amanda	745.124.678-04	1
3	Hebert	555.666.777-88	3

Conta		
id	agencia	numero
1	3460	71342
2	5421	65321
3	7410	02145

Agência aparece em tabelas diferentes



Indica que

Agência é uma entidade à parte



Solução

Criar tabela agencia e usar chave estrangeira nas outras tabela

Não combinar atributos de entidades diferentes

Gerente			
id	nome	cpf	id_agencia
1	Juan	111.222.333-44	4
2	Hebert	555.666.777-88	7
3	Claudia	123.456.789-00	8

Gerente_Conta	
id_gerente	id_conta
1	2
3	3
3	1

Cliente			
id	nome	cpf	id_conta
1	Juan	111.222.333-44	2
2	Amanda	745.124.678-04	1
3	Hebert	555.666.777-88	3

Conta		
id	id_agencia	numero
1	7	71542
2	8	65321
3	4	02145

Agencia		
id	agencia	endereco
7	3460	Rua Oswaldo 74
4	5421	Av ACM 20
8	7410	Rua Arlinda 60

Agora é mais fácil entender a semântica das tabelas

Gerente possui dados exclusivos de gerente, idem pra Conta e pra Agencia

Reduzir valores NULL (vazio) nas tuplas

- NULL pode significar muitas coisas (pode gerar ambiguidade)
 - O dado existe, porém não é desconhecido (**ex:** endereço)
 - O dado não existe (**ex:** numero_pis_pasep)
 - O dado não se aplica (**ex:** cpf_gerente para um funcionário que já gerente)
 - Produz resultados imprevisíveis (**ex:** como buscar uma pessoa pelo endereço dela se o endereço é NULL ?)

Reduzir valores NULL (vazio) nas tuplas

“Jessica” sabe quem é o seu gerente, porém não sabe qual é seu departamento (“Dnome”)

“cpf_gerente” não se aplica para “Carlos”, pois ele é o gerente do departamento “Venda”

“Felipe” foi recém contratado e não tem um departamento ou gerente ainda

Funcionario				
id	<u>Fnome</u>	<u>cpf</u>	<u>Dnome</u>	<u>cpf_gerente</u>
1	Jessica	415.254.784-00	NULL	661.457.745-87
2	Carlos	661.457.745-87	Venda	NULL
3	Felipe	325.471.154-64	NULL	NULL

Reduzir valores NULL (vazio) nas tuplas

- NULL dificulta consultas de dados e funções de agregação (SOMAR, CONTAR)
 - **Ex:** como contar a quantidade de pessoas que moram na Av ACM, se existem pessoas com endereço NULL no banco de dados ?
 - Ignoro as pessoas que tem endereço NULL ?
 - Considero que NULL representa pessoas que não moram em Valença ?

Exercício - Diretrizes de banco de dados I

- Descreva de que forma o projeto de banco de dados abaixo pode ser melhorado seguindo as diretrizes de banco de dados que vimos até agora

Chefe		
<u>id</u>	nome	cpf
1	Henrique	111.777.888-00
2	Carla	000.111.333-44
3	Josué	222.333.000-77

Departamento			
<u>id</u>	nome	n_func	chefe
1	Estoque	10	Henrique
2	Vendas	8	Carla
3	Marketing	13	Josué

Funcionario			
<u>id</u>	nome	cpf	id_depto
1	Jessica	415.254.784-00	NULL
2	Jessica	661.457.745-87	8
3	Felipe	325.471.154-64	7

Reduzir informações redundantes nas tuplas

- Otimiza o uso de espaço em disco dos bancos de dados
- Melhora no desempenho do sistema
- Facilita a atualização de dados (sabemos quais tabelas e registros precisarão ser atualizados no banco de dados)
- Evitar incoerências (inconsistências) no banco de dados
 - Dados de uma mesma instância de entidade com valores diferentes

Reduzir informações redundantes nas tuplas

Gerente			
<u>id</u>	<u>nome</u>	<u>cpf</u>	<u>id_agencia</u>
1	Juan	111.222.333-44	4
2	Hebert	555.666.777-88	1
3	Claudia	123.456.789-00	8

Gerente_Conta	
<u>id_gerente</u>	<u>id_conta</u>
1	2
3	3
3	1

Cliente			
<u>id</u>	<u>nome</u>	<u>cpf</u>	<u>id_conta</u>
1	Juan	111.222.333-44	2
2	Amanda	745.124.678-04	1
3	Hebert	555.666.777-88	3

Conta		
<u>id</u>	<u>id_agencia</u>	<u>numero</u>
1	7	71548
2	8	65321
3	4	02145

Agencia		
<u>id</u>	<u>agencia</u>	<u>endereço</u>
7	3460	Rua Oswaldo 74
4	5421	Av ACM 20
8	7410	Rua Arlinda 60

Nome e CPF aparecem
em tabelas diferentes



Indica que

Nome e CPF pertencem
a uma entidade à parte

Reduzir informações redundantes nas tuplas

Pessoa		
<u>id</u>	nome	cpf
80	Juan	111.222.333-44
81	Hebert	555.666.777-88
82	Claudia	123.456.789-00
83	Amanda	745.124.678-04

Gerente		
<u>id</u>	<u>id_pessoa</u>	<u>id_agencia</u>
1	80	4
2	81	7
3	82	8

Gerente_Conta	
<u>id_gerente</u>	<u>id_conta</u>
1	2
3	3
3	1

Cliente		
<u>id</u>	<u>id_pessoa</u>	<u>id_conta</u>
1	80	2
2	83	1
3	81	3

Conta		
<u>id</u>	<u>id_agencia</u>	numero
1	7	71542
2	8	65321
3	4	02145

Agencia		
<u>id</u>	<u>agencia</u>	endereco
7	3460	Rua Oswaldo 74
4	5421	Av ACM 20
8	7410	Rua Arlinda 60

Agora é mais fácil atualizar os dados da agência (ex: endereço)

Não há redundância de dados (melhora no uso de espaço em disco pelo DB)

Reduzir informações redundantes nas tuplas

- **Porque reduzir dados redundantes?**

- Melhora na utilização de espaço em disco por DBs
- Melhora no desempenho no uso do sistema
- Mitiga problemas de **anomalias de atualização**, tais como:

- **Anomalias de inserção**
- **Anomalias de exclusão**
- **Anomalias de modificação**

Anomalias de atualização:
Problemas que podem causar
perdas de dados, além de
reduzir o desempenho do DB.

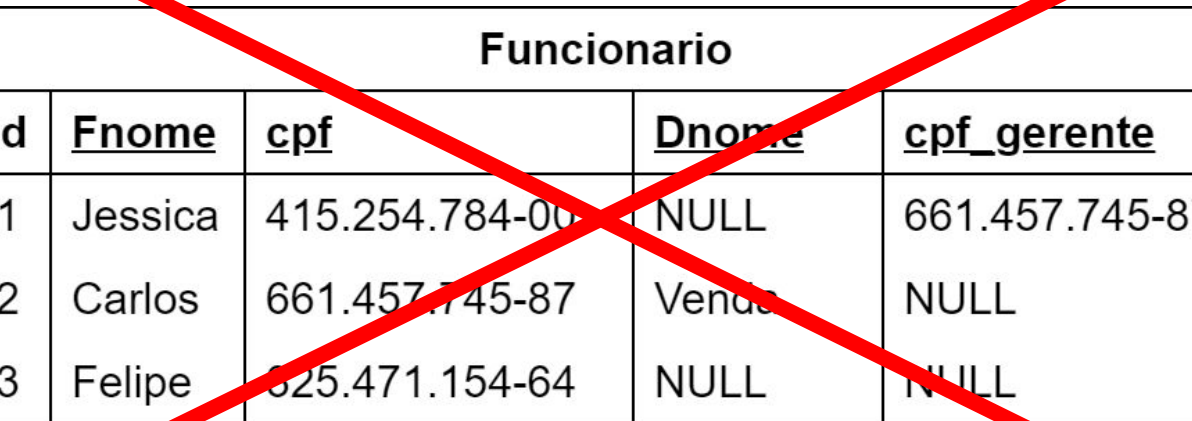
Anomalia de inserção

- Incluir uma nova tupla pode exigir armazenar valores NULL
 - **Ex:** gerente “Claudia” recém contratada pelo banco, porém ainda não foi lotada em nenhuma agência

Gerente			
<u>id</u>	<u>nome</u>	<u>cpf</u>	<u>agencia</u>
1	Juan	111.222.333-44	3460
2	Hebert	555.666.777-88	7410
3	Claudia	123.456.789-00	NULL

Anomalia de inserção

- Esquema ERRADO:



Funcionario				
id	<u>Fnome</u>	<u>cpf</u>	<u>Dnome</u>	<u>cpf_gerente</u>
1	Jessica	415.254.784-00	NULL	661.457.745-87
2	Carlos	661.457.745-87	Vendas	NULL
3	Felipe	525.471.154-64	NULL	NULL

Anomalia de inserção

- Esquema CORRETO:

Funcionario			
<u>id</u>	<u>nome</u>	<u>cpf</u>	<u>id_depto</u>
1	Jessica	415.254.784-00	NULL
2	Carlos	661.457.745-87	8
3	Felipe	325.471.154-64	7

Departamento		
<u>id</u>	<u>nome</u>	<u>cpf_gerente</u>
6	Vendas	NULL
7	Estoque	887.665.451-99
8	Adm	661.457.745-87

Anomalia de inserção

- Inserir informações incorretas nas tuplas
 - **Ex:** gerente “Claudia” foi recém contratada pelo banco para a agência Valença
 - O número da agência digitado confere com o de outros gerentes da mesma agência?

Gerente			
id	<u>nome</u>	<u>cpf</u>	<u>agencia</u>
1	Juan	111.222.333-44	3460
2	Hebert	555.666.777-88	3460
3	Claudia	123.456.789-00	???

Necessidade de conferir os valores dos atributos a cada inserção no DB

Teremos dados duplicados (**ex:** número da agência)

Anomalia de exclusão

- Ao remover uma tupla, todos os valores são deletados
 - Se a tupla contiver valores de mais de uma entidade, esses também serão removidos
 - **Ex:** remover o último funcionário de um departamento faz com que todos os dados do departamento também sejam removidos

Funcionario				
id	Fnome	cpf	Dnome	cpf_gerente
1	Jessica	415.254.784-00	Estoque	661.457.745-87
2	Carlos	661.457.745-87	Venda	777.142.654-71
3	Felipe	325.471.154-64	Venda	777.142.654-71

O funcionário “*Felipe*” foi removido

Se “*Carlos*” for removido também, perderemos os dados do departamento “*Venda*”

Anomalia de modificação

- Ao alterarmos os valores de uma tupla, temos de atualizar todas as tuplas que contém esses valores também
 - **Ex:** se o gerente do depto “Vendas” mudar, temos de alterar essa informação em todas as tuplas da tabela “Funcionário” associadas ao depto “Vendas”

Funcionario				
id	Fnome	cpf	Dnome	cpf_gerente
1	Jessica	415.254.784-00	Estoque	661.457.745-87
2	Carlos	661.457.745-87	Venda	777.142.654-71
3	Felipe	325.471.154-64	Venda	452.170.883-64

O **CPF do gerente** do funcionário “*Felipe*” é **diferente** do CPF do gerente de “*Carlos*”, sendo que **ambos representam o mesmo gerente** do depto de “*Vendas*”

Qual das duas informações é a correta ?

Anomalia de modificação

- **Ex:** se o gerente do depto “Vendas” mudar, temos de alterar essa informação em todas as tuplas da tabela “Funcionário” associadas ao depto “Vendas”

Funcionario				
id	<u>Fnome</u>	<u>cpf</u>	<u>Dnome</u>	<u>cpf_gerente</u>
1	Jessica	415.254.784-00	Estoque	661.457.745-87
2	Carlos	661.457.745-87	Venda	777.142.654-71
3	Felipe	325.471.154-64	Venda	452.170.883-64

Nesse caso, o **banco de dados se tornou incoerente** pois ele possui informações divergentes

Verificar todas as tuplas associadas a um depto é uma tarefa muito cara



Desempenho ruim do sistema!

Evitar relações com tuplas falsas



<u>cpf_funcionario</u>	<u>depto</u>	<u>cpf_gerente</u>
415.254.784-00	Adm	661.457.745-87
661.457.745-87	Vendas	NULL
325.471.154-64	Vendas	NULL

<u>nome</u>	<u>depto</u>
Jessica	Adm
Carlos	Vendas
Felipe	Vendas

Como descobrir o nome do funcionário associado ao CPF ?

Existem dois funcionários para o mesmo departamento de Vendas!

Será que podemos usar o nome do departamento como chave estrangeira (**FK**)?

“Funcionario.depto” NÃO pode ser FK, pois “Funcionario_Departamento.depto” NÃO é chave primária

Evitar relações com tuplas falsas

Funcionario_Departamento		
<u>cpf_funcionario</u>	<u>depto</u>	<u>cpf_gerente</u>
415.254.784-00	Adm	661.457.745-87
661.457.745-87	Vendas	NULL
325.471.154-64	Vendas	NULL

PK

Funcionario	
<u>nome</u>	<u>depto</u>
Jessica	Adm
Carlos	Vendas
Felipe	Vendas

PK

É possível criar relações cujas tuplas dificultam a obtenção de dados de múltiplas tabelas, através de consultas com chave estrangeira

Isto é, **não existe um par de chaves estrangeira e primária** que permita associar as tuplas das duas tabelas

Evitar relações com tuplas falsas

Funcionario_Departamento		
<u>cpf_funcionario</u>	<u>depto</u>	<u>cpf_gerente</u>
415.254.784-00	Adm	661.457.745-87
661.457.745-87	Vendas	NULL
325.471.154-64	Vendas	NULL

PK

Funcionario	
<u>nome</u>	<u>depto</u>
Jessica	Adm
Carlos	Vendas
Felipe	Vendas

PK

Nesse exemplo em específico, na tentativa de associar o nome do funcionário ao seu CPF, podemos criar tuplas que contém informações erradas (**tuplas falsas**).

Ex:

Podemos criar as seguintes tuplas $t_1 = \langle 661.457.745-87, \text{"Carlos"}, \text{"Vendas"} \rangle$ e $t_2 = \langle 325.471.154-64, \text{"Carlos"}, \text{"Vendas"} \rangle$. Sabemos que uma delas é uma tupla falsa, pois cada pessoa possui no máximo um CPF!

Atividade - Diretrizes de banco de dados

- Projete um banco de dados para atender as necessidades do IFBA, seguindo as diretrizes de projeto mencionadas, conforme os requisitos do IFBA abaixo:
 - O sistema deve armazenar os dados dos Alunos, Técnicos e Professores
 - O sistema deve armazenar as notas dos alunos por unidade, frequência, turma e disciplinas cursadas
 - Devemos ter uma forma de associar o professor a turma e a disciplina ministrada

Dependência funcional (DF ou d.f.)

- As diretrizes (boas práticas) de projetos de bancos de dados são informais
 - Precisamos de ferramentas mais precisas para avaliar a qualidade de um projeto
 - **Solução:** Usar dependências funcionais

Dependência funcional (DF)

- **Ex:** CPF -> nome

Consigo descobrir o “**nome**”
a partir do “**CPF**”

Note que o contrário não é
verdadeiro

Pode existir duas pessoas
com mesmo nome
Ex: duas “Jessicas”

The diagram shows a table named 'Funcionario' with columns: id, nome, cpf, and id_depto. A blue arrow points from the 'cpf' column to the 'nome' column, indicating a functional dependency. A red arrow points from the 'nome' column to the 'cpf' column, and it is crossed out with a large black 'X', indicating that the reverse dependency does not hold. The table data is as follows:

Funcionario			
id	<u>nome</u>	<u>cpf</u>	<u>id_depto</u>
1	Jessica	415.254.784-00	NULL
2	Jessica	661.457.745-87	8
3	Felipe	325.471.154-64	7

Dependência funcional (DF)

- **$X \rightarrow Y$**
 - Y é **funcionalmente dependente** de X
 - Valores de Y são determinados pelos valores de X
 - X determina Y
 - Existe dependência funcional de X para Y

- **Ex:** CPF \rightarrow nome

Consigo descobrir o “*nome*” a partir do “*CPF*”

Funcionario			
<u>id</u>	<u>nome</u>	<u>cpf</u>	<u>id_depto</u>
1	Jessica	415.254.784-00	NULL
2	Jessica	661.457.745-87	8
3	Felipe	325.471.154-64	7

Dependência funcional (DF ou d.f.)

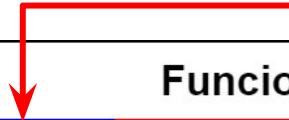
- É uma restrição entre dois conjuntos de atributos (**X** e **Y**)
- **Formalmente:**
 - “Uma dependência formal **X** \rightarrow **Y**, entre dois conjuntos de atributos **X** e **Y**, especifica uma restrição sobre tuplas, que podem formar um estado **r** da relação **R**”

Dependência funcional (DF)

- É uma propriedade da tabela (esquema) e não das tuplas (instâncias)

CPF \rightarrow nome é válido para qualquer *nome* ou *CPF*

Se existir um “*nome*” que não conseguimos descobrir a partir de um “*CPF*”, então a dependência funcional **CPF \rightarrow nome** NÃO existe



Funcionario			
id	<u>nome</u>	<u>cpf</u>	<u>id_depto</u>
1	Jessica	415.254.784-00	NULL
2	Jessica	661.457.745-87	8
3	Felipe	325.471.154-64	7

Dependência funcional (DF)

- **Ex:** (cpf_funcionario, depto) -> cpf_gerente

Consigo descobrir o
“**cpf_gerente**” a partir da
chave primária
“(b>cpf_funcionario, depto)”

Note que o contrário não é
verdadeiro
(um mesmo gerente pode
supervisionar vários
funcionários)

Funcionario_Departamento		
<u>cpf_funcionario</u>	<u>depto</u>	<u>cpf_gerente</u>
415.254.784-00	Adm	661.457.745-87
661.457.745-87	Vendas	NULL
325.471.154-64	Vendas	661.457.745-87

PK

X

Exercício - Dependência funcional

- Descubra quais dependências funcionais existem nas tabelas abaixo:

Funcionario		
nome	cpf	id_depto
Jessica	415.254.784-00	NULL
Jessica	661.457.745-87	8
Felipe	325.471.154-64	7

Departamento		
nome	n_func	chefe
Estoque	10	Henrique
Vendas	8	Carla
Marketing	13	Josué

Chefe	
nome	cpf
Henrique	111.777.888-00
Carla	000.111.333-44
Josué	222.333.000-77

Propriedades de Dependência Funcional

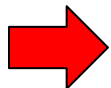
- Semelhante às operações matemáticas básicas (+, -, /, x) , dependências funcionais também têm propriedades:
 - **Separação**
 - **Acumulação**
 - **Transitividade**
 - **Pseudo-transitividade**

Separação

- Se **A -> BC** então:
 - **A -> B**
 - **A -> C**
- **Ex:** CPF -> (NOME, ENDERECO) **então** CPF -> NOME e CPF -> ENDERECO

Com o **CPF** eu consigo obter
o **NOME** e o **ENDERECO** da
pessoa

então



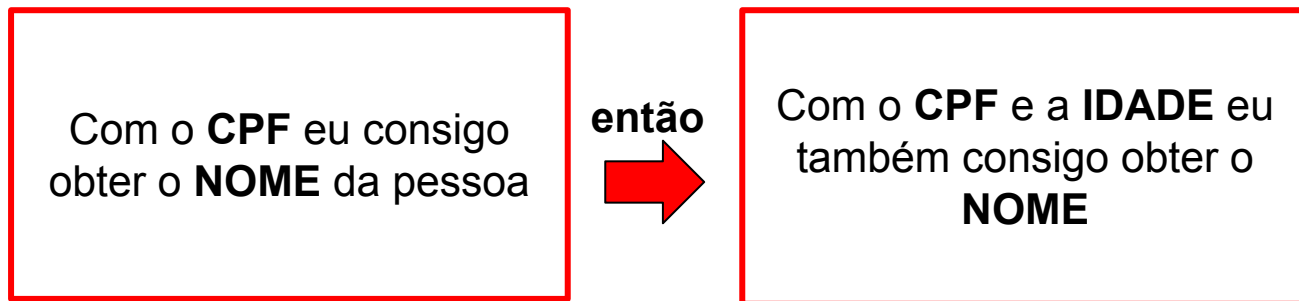
Com o **CPF** eu consigo obter somente o **NOME**

E

Com o **CPF** eu consigo obter somente o **ENDERECO**

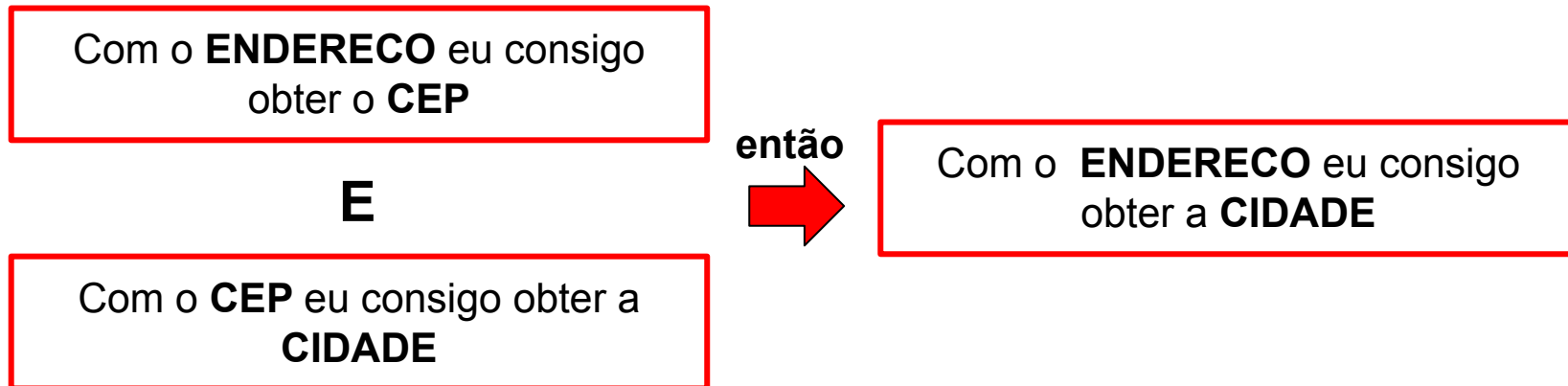
Acumulação

- Se **A** -> **B** então:
 - **AC** -> **B**
- **Ex:** CPF -> NOME **então** (CPF, IDADE) -> NOME



Transitividade

- Se **A -> B** e **B -> C** então:
 - **A -> C**
- **Ex:** ENDERECO-> CEP e CEP -> CIDADE **então** ENDERECO -> CIDADE



Pseudotransitividade

- Se **A -> B** e **B C -> D** então:
 - **AC -> D**
- **Ex:** CPF -> COD_FUNC e (COD_FUNC, CARGO) -> SALARIO **então** (CPF, CARGO) -> SALARIO

Com o **CPF** eu consigo obter o
COD_FUNC

E

Com o **(COD_FUNC, CARGO)** eu
consigo obter a **SALARIO**

então


Com o **(CPF, CARGO)** eu consigo
obter a **SALARIO**

Exercício - Dependência funcional

- Forneça exemplos das propriedades das dependências funcionais que podemos aplicar nas tabelas abaixo:

Funcionario			Chefe	
nome	cpf	id_depto	nome	cpf
Jessica	415.254.784-00	NULL	Henrique	111.777.888-00
Jessica	661.457.745-87	8	Carla	000.111.333-44
Felipe	325.471.154-64	7	Josué	222.333.000-77

Departamento		
nome	n_func	chefe
Estoque	10	Henrique
Vendas	8	Carla
Marketing	13	Josué

Referencial Bibliográfico

- KORTH, H.; SILBERSCHATZ, A.; SUDARSHAN, S. **Sistemas de bancos de dados**. 5. ed. Rio de Janeiro: Ed. Campus, 2006.
- DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Ed. Campus, 2004. Tradução da 8ª edição americana.