
Normalização

Sistemas de Informação I

Motivação I

Para o armazenamento de informação acerca de fornecedores e produtos, definiu-se a seguinte Relação R:

Código Fornecedor	Nome	Morada	Produto	Preço
12345	Pregos, Lda	Azinhaga Trinta e Um, 32, R/C Drt.	Cavilhas	0,05 €
12345	Pregos, Lda	Azinhaga Trinta e Um, 32, R/C Drt.	Parafusos 6mm	0,20 €
99912	Sizal & Filhos	Av. Das Descobertas 78	Cordel	0,50 €
14566	Borracheiro	Rua Bica funda, 12, Lote 3	Mangueira 30mm	0,25 €
12345	Pregos, Lda	Azinhaga Trinta e Um, 32, R/C Drt.	Anilhas	0,01 €
14566	Borracheiro	Rua Bica funda, 12, Lote 3	Botas Borracha	5,00 €

A Relação apresenta informação redundante (e.g. morada). **Quais os problemas que se colocam?**

Motivação II

Problemas de manutenção

- A redundância significa armazenamento repetido da mesma informação. Por essa razão, a alteração ou remoção de informação pode implicar o acesso a variadas partes da base de dados, tornando-se difícil manter a coerência dos dados.

Custos de armazenamento

- Uma vez que existe informação repetida, existe espaço que está a ser desperdiçado. Esse desperdício pode ser maior ou menor consoante exista muita ou pouca redundância. Mesmo com o preço do MB a descer, quando o volume de informação é grande, este é um problema a ter em conta.

Problemas de desempenho

- Embora menos evidente, se existe desperdício de espaço, é necessário um maior acesso ao disco para aceder à mesma informação. Como o suporte magnético é tendencialmente lento, esses acessos traduzem-se em tempo desperdiçado.

Anomalia de Alteração

- Como alterar a morada do fornecedor ‘Pregos Lda.’ ?
- Que aconteceria se essa alteração não fosse feita para um dos tuplos?

Anomalia de Remoção

- Como proceder para remover os tuplos relacionados com um determinado Fornecedor?
- Os produtos por eles fornecidos serão igualmente apagados?
- Como remover um produto?

Anomalia de Inserção

- Como se adiciona um novo fornecedor?
 - Será possível essa adição sem ele fornecer algum produto?
- Colocando NULL nos campos respeitantes aos Produtos
 - Mas se quando no fornecimento do primeiro produto esses valores a NULL ficassem esquecidos?
- Código Produto identifica univocamente um produto –
Não é permitido ter valor NULL!

Como resolver as anomalias
apresentadas ?

Decomposição I

- Acabando com a informação repetida, ou seja, redundante
- Essa redundância pode se ultrapassada decompondo as Relações noutras Relações mais pequenas
- A decomposição passa primeiro pela análise do que não pode ser decomposto
- No exemplo:
 - CódigoFornecedor não pode ser separado de nome e morada – Existe uma dependência entre eles
 - CódigoFornecedor não pode ser separado de produto e de preço
 - Cada fornecedor fornece um ou vários produtos

Decomposição II

- Sabendo que CódigoFornecedor determina o nome e a morada e que CódigoFornecedor determina o produto e preço, a Relação R será decomposta em:
 - FORNECEDOR(CódigoFornecedor, Nome, Morada)
 - PRODUTO_FORNECIDO(CódigoFornecedor, Produto, Preço)

Mas será que se consegue obter a mesma informação que estava na Relação R?

- Pretende-se determinar o nome do fornecedor de cordel
- Através da chave estrangeira existente em PRODUTO_FORNECIDO é possível responder à questão
- Não se perdeu informação com a decomposição



Existe um equilíbrio que é necessário manter, entre performance e decomposição

Dependências entre dados

- Nos dados estão patentes algumas dependências que são necessárias conhecer para se proceder à decomposição de Relações
- Essas dependências podem ser:
 - **Dependências Funcionais**
 - Dependências Multivalor
 - Dependências de Junção
- Com base nestas dependências são definidas um conjunto de regras para resolver os problemas de redundância designadas de **Formas Normais**
 - Para as primeiras dependências são definidas quatro formas normais, nomeadamente **1^a, 2^a, 3^a** e Boyce-Cood Formas Normais
 - Para as dependências multivalor é definida a 4^a Forma Normal
 - Para as dependências de junção é definida a 5^a Forma Normal

Âmbito do nosso estudo



Dependências Funcionais

- Na definição dos problemas existem, subjacentes aos dados, algumas relações perfeitamente definidas
- Na Relação Fornecedor, codigoFornecedor identifica sempre o nome e a morada de um fornecedor
- Ou seja, existe um Dependência Funcional entre codigoFornecedor e os outros dois atributos: nome e morada



No entanto, o inverso pode não ser verdade !

Dependências Funcionais (cont.)

Definição

- Sejam A_1, A_2, \dots, A_n o conjunto de atributos de uma Relação
- Sejam X e Y dois conjuntos não vazios de $\{A_1, A_2, \dots, A_n\}$
- Diz-se que existe uma dependência funcional entre X e Y , denotada como $X \rightarrow Y$, quando uma instância de valores de X identifica univocamente uma instância de valores de Y
 - Quaisquer dois tuplos $T1$ e $T2$ verificam:
 $t1[X]=t2[X] \Rightarrow t1[Y]=t2[Y]$
- Ou seja, não existem duas instâncias de Y com o mesmo valor em X
- Formalmente: $\forall (t1, t2), t1[x]=t2[x] \Rightarrow t1[y]=t2[y]$

Convenções

- A sigla DF é normalmente usada para designar Dependência funcional

Se $X \rightarrow Y$

- Existe uma dependência funcional de X para Y
- Y é funcionalmente dependente de X
- Os valores de X determinam os valores de Y
- A cada valor de X está associado um e um só valor de Y
- X é o lado esquerdo da Dependência Funcional
- Y é o lado direito da Dependência Funcional

Identificação de Dependências Funcionais

- As Dependências Funcionais estão intrinsecamente ligadas ao problema em questão e não podem ser inferidas directamente a partir de alguns tuplos das relações 

- No entanto, através da **semântica** dos atributos, dado o domínio do problema, é possível extrair algumas dependências funcionais

numero	nome	dataNasc
12345	Antonio	10-10-1970
43321	Nuno	21-09-1970
12238	Antonio	22-10-1968
12231	Maria	12-05-1976

- nome depende de número?
 - Admitindo que cada aluno só pode ter um número, conhecendo esse número sabe-se o nome do aluno
 - Assim nome é funcionalmente dependente de número
 - No entanto foi necessário ter conhecimento do problema!
- nome → dataNasc
 - $t1[\text{nome}] = \text{'Antonio'} = t2[\text{nome}] = \text{'Antonio'} \rightarrow t1[\text{dataNasc}] = \text{'10-10-1970'} = t1[\text{dataNasc}] = \text{'22-10-1968'}$
 - Sendo falso, nome não determina dataNasc

Identificação de Dependências Funcionais (cont.)

Fornecedor	Nome	Produto	Preco
12345	Pregos, Lda	Cavilhas	0,05 €
99912	Sizal & Filhos	Cordel	0,50 €
1122	Latão Douro	Cavilhas	0,20 €
14566	Borracheiro	Mangueira 30mm	0,25 €
14566	Borracheiro	Botas Borracha	5,00 €

- Quais os atributos que determinam o preço dos produtos?
- Fornecedor → Preco
 - Não. Cada fornecedor tem um preço para os produtos que vende
- Produto → Preco
 - Não. Dependendo do fornecedor, o preço dos produtos variam
- Ou seja, preço é funcionalmente dependente de dois atributos
 - (fornecedor, produto) → preco



Como provar que não existe determinada Dependência Funcional?

Prova da não existência de DF

- Recordando a definição de Dependência Funcional
 - $\forall (t1, t2), t1[x] = t2[x] \rightarrow t1[y] = t2[y]$
- E algumas equivalências entre expressões lógicas
 - $A \rightarrow B$ é equivalente a $\neg A \vee B$
 - $\neg \forall p(x)$ é equivalente a $\exists \neg p(x)$
- Para provar que $X \rightarrow Y$ é falso, ou seja, que Y não depende funcionalmente de X , pode-se escrever da seguinte forma
 - $\neg \forall (t1, t2), t1[X] = t2[X] \rightarrow t1[Y] = t2[Y]$, onde
 - $P(x)$ é a expressão $t1[X] = t2[X] \rightarrow t1[Y] = t2[Y]$

Prova da não existência de DF (cont.)

- Sabemos, através das equivalências entre expressões lógicas, que
 - $\neg \forall (t_1, t_2), p(x)$ e $\exists (t_1, t_2), \neg p(x)$ são equivalentes
- Se
 - A for dado por $t_1[X] = t_2[X]$
 - B por $t_1[Y] = t_2[Y]$
- Então
 - $\exists (t_1, t_2), \neg p(x)$ passa para $\exists (t_1, t_2), \neg (\neg A \vee B)$
- Sabendo que $\neg (\neg A \vee B)$ é equivalente a $(A \wedge \neg B)$, chega-se por fim à negação da definição de Dependência Funcional, aquilo que se pretende provar

$\exists (t_1, t_2), t_1[X] = t_2[X] \wedge t_1[Y] \neq t_2[Y]$

Exemplo

Considere a seguinte Relação

R1	A	B	C
	x	y	z
	s	t	u
	a	b	t
	x	y	t

- Como provar que é falsa a seguinte Dependência Funcional
 - $(A,B) \rightarrow C$
- Provando que se verifica, na Relação R1
 - $\exists (t_1, t_2), t_1[X] = t_2[X] \rightarrow t_1[Y] \neq t_2[Y]$
- $\exists (t_1, t_2), t_1[A,B] = t_2[A,B] \rightarrow t_1[C] \neq t_2[C]$, ou seja, existem dois tuplos com os mesmos valores para os atributos A e B e que têm valores distintos no atributo C
- Existe, A=x e B=y, logo a dependência funcional não existe

Manipulação de Dependências Funcionais

- A manipulação das Dependências Funcionais obedece aos *axiomas de Armstrong*, descritos de seguida
 1. **Reflexividade** – se $X \supseteq Y$ então $X \rightarrow Y$
 2. **Aumento** – se $X \rightarrow Y$ e $Z \supseteq W$ então $XZ \rightarrow YW$
 3. **Transitividade** – se $X \rightarrow Y$ e $Y \rightarrow Z$ então $X \rightarrow Z$
- Os axiomas acima descritos dão origem às seguintes regras derivadas
 4. **Decomposição** – se $X \rightarrow YZ$ então $X \rightarrow Y$ e $X \rightarrow Z$
 5. **União** – se $X \rightarrow Y$ e $X \rightarrow Z$ então $X \rightarrow YZ$
 6. **Pseudotransitividade** – se $X \rightarrow Y$ e $YW \rightarrow Z$ então $XW \rightarrow Z$

Provas

1. Reflexividade – se $X \sqsupseteq Y$ então $X \rightarrow Y$
 - Admita-se que $X \sqsupseteq Y$ e que existem dois tuplos t_1 e t_2 ;
 - Se se verificar que $t_1[X]=t_2[X]$ como $X \sqsupseteq Y$ então
 - $t_1[Y]=t_2[Y]$, logo $X \rightarrow Y$
2. Aumento – se $X \rightarrow Y$ e $Z \sqsupseteq W$ então $XZ \rightarrow YW$
 - Admita-se que $X \rightarrow Y$ é verdadeiro mas que $XZ \rightarrow YW$ é falso
 - Se existissem dois tuplos t_1 e t_2 tais que
 - a) $t_1[X]=t_2[X]$
 - b) $t_1[Y]=t_2[Y]$
 - c) $t_1[XZ]=t_2[XZ]$
 - d) $t_1[YW] \neq t_2[YW]$
 - e) $t_1[Z]=t_2[Z]$
 - Como se verifica existe um inconsistência
 - De a) e c) conclui-se e)
 - De b) e e) e sabendo que $Z \sqsupseteq W$ verifica-se que d) é inconsistente

Provas (cont.)

3. Transitividade – se $X \rightarrow Y$ e $Y \rightarrow Z$ então $X \rightarrow Z$
 - Para quaisquer dois tuplos t_1 e t_2 tem-se que
 - $t_1[X]=t_2[X]$ e $t_1[Y]=t_2[Y]$
 - Como $Y \rightarrow Z$ então $t_1[Y]=t_2[Y]$ e $t_1[Z]=t_2[Z]$
 - Logo $X \rightarrow Z$
4. Decomposição – se $X \rightarrow YZ$ então $X \rightarrow Y$ e $X \rightarrow Z$
 - Usando 1) tem-se que $YZ \rightarrow Y$, sabendo que $Y \supseteq YZ$
 - Usando 3) em $X \rightarrow YZ$ e $YZ \rightarrow Y$ tem-se $X \rightarrow Y$
 - Procedendo da mesma forma para Z , chega-se também à conclusão que $X \rightarrow Z$

5. União

se $X \rightarrow Y$ e $X \rightarrow Z$ então $X \rightarrow YZ$

- Usando 2) e aplicando a $X \rightarrow Y$, tem-se $X \rightarrow XY$ ($XX=X$)
- Usando 2) e aplicando a $X \rightarrow Z$, tem-se $XY \rightarrow YZ$
- Usando 3) em $X \rightarrow XY$ e $XY \rightarrow YZ$, tem-se que $X \rightarrow YZ$

6. Pseudotransitividade

se $X \rightarrow Y$ e $YW \rightarrow Z$ então $XW \rightarrow Z$

- Usando 2) em $X \rightarrow Y$ tem-se que $XW \rightarrow YW$
- Usando 3) em $XW \rightarrow YW$ e $YW \rightarrow Z$, tem-se que $XW \rightarrow Z$

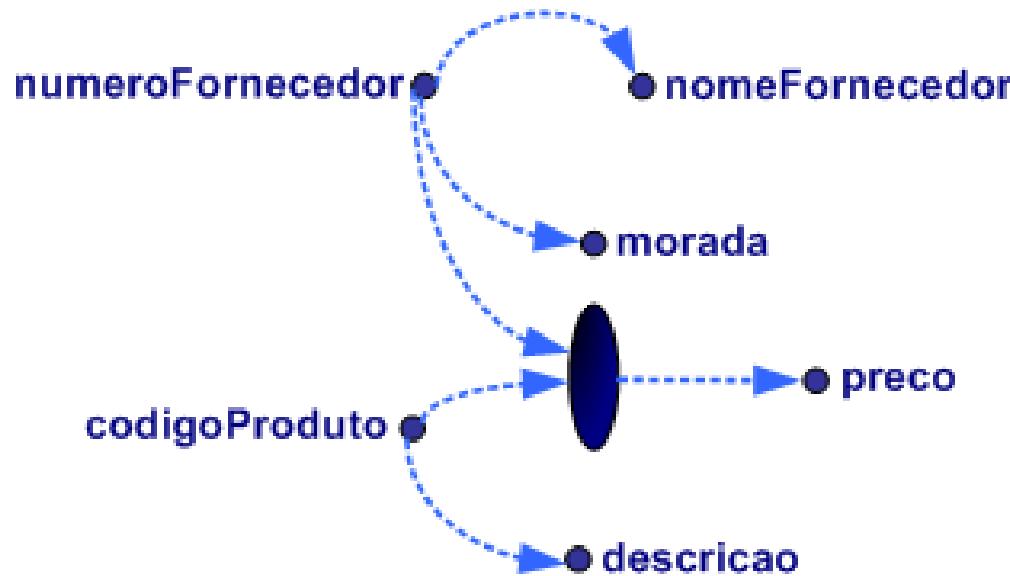
Representação de DF com grafos

- Seja
 - $\{A_1, A_2, \dots, A_n\}$ um conjunto de atributos
 - $F = \{X_i \rightarrow A_j\}$ um conjunto de DF
 - $X_i = \{A_{i1}, A_{i2}, \dots, A_{ip}\}$ com $p > 1$
- F pode ser representado sobre a forma de um grafo atendendo a que
 - A todo o atributo A_i é atribuído um nó etiquetado com A_i
 - A todas as DF $X_i \rightarrow A_j$ corresponde a um nó auxiliar etiquetado com X_i , constituído por um conjunto de arcos que ligam os vários $A_{i1}, A_{i2}, \dots, A_{ip}$ e um arco que liga X_i a A_j

Representação de DF com grafos - Exemplo

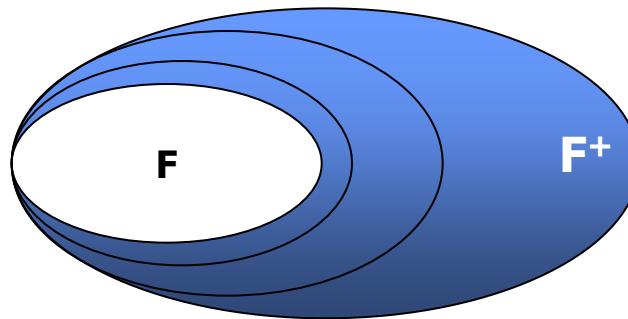
- Seja

$F = \{ \text{numeroFornecedor} \rightarrow \text{nameFornecedor}, \text{numeroFornecedor} \rightarrow \text{morada}, (\text{numeroFornecedor}, \text{codigoProduto}) \rightarrow \text{preco}, \text{codigoProduto} \rightarrow \text{descricao} \}$



Fecho de um conjunto de Dependências Funcionais

- Seja F um conjunto de dependências funcionais elementares
- Designa-se de Fecho de F , representado por F^+ , ao conjunto de todas as Dependências Funcionais que podem ser derivadas de F utilizando para isso os axiomas de Armstrong e as regras derivadas dos axiomas
- Dessa forma são geradas “novas” Dependências Funcionais a partir das elementares



Fecho de um conjunto de DF (cont.)

- Considere-se o Esquema de Relação R definido como $R(A,B,C)$
- Considere-se o conjunto $F=\{A \rightarrow B, B \rightarrow C\}$
- O Fecho de F, ou seja, o conjunto de Dependências Funcionais que se podem derivar é dado por:
- $F^+ = \{A \rightarrow A, B \rightarrow B, C \rightarrow C, AB \rightarrow A, AB \rightarrow B, AB \rightarrow C, AC \rightarrow A, AC \rightarrow B, AC \rightarrow C, \dots, A \rightarrow C, A \rightarrow \emptyset, B \rightarrow \emptyset, C \rightarrow \emptyset, AB \rightarrow \emptyset, \dots\}$
- É bastante difícil construir um fecho de um conjunto e não deixar escapar alguma DF

Lidar com o volume de DF do fecho é extremamente difícil !

Fecho de um conjunto de Atributos

- Seja X um conjunto de atributos
- O Fecho do atributo X, representado por X^+ , é o conjunto de todos os atributos funcionalmente dependentes de X
- Algoritmo para o cálculo de X^+

```
X+=X
Repetir
{
    antigoX+ := X+
    Para cada DF Y → Z
        Se Y ⊆ X+ então X+ := X+ ∪ z
}enquanto (X+<> antigoX+)
```

Fecho de um conjunto de Atributos (cont.)

- $F = \{ \text{numeroEmpregado} \rightarrow \text{nameEmpregado},$
 $\text{numeroProjecto} \rightarrow (\text{nameProjecto}, \text{localizacao}),$
 $(\text{numeroEmpregado}, \text{numeroProjecto}) \rightarrow \text{horasEstimadas} \}$
 - $\text{numeroEmpregado}^+ = \{\text{numeroEmpregado}, \text{nameEmpregado}\}$
 - $\text{numeroProjecto}^+ = \{\text{numeroProjecto}, \text{nameProjecto}, \text{localizacao}\}$
 - $(\text{numeroEmpregado}, \text{numeroProjecto})^+ = \{\text{numeroProjecto},$
 $\text{numeroEmpregado}, \text{nameEmpregado}, \text{nameProjecto},$
 $\text{localizacao}, \text{horasEstimadas}\}$

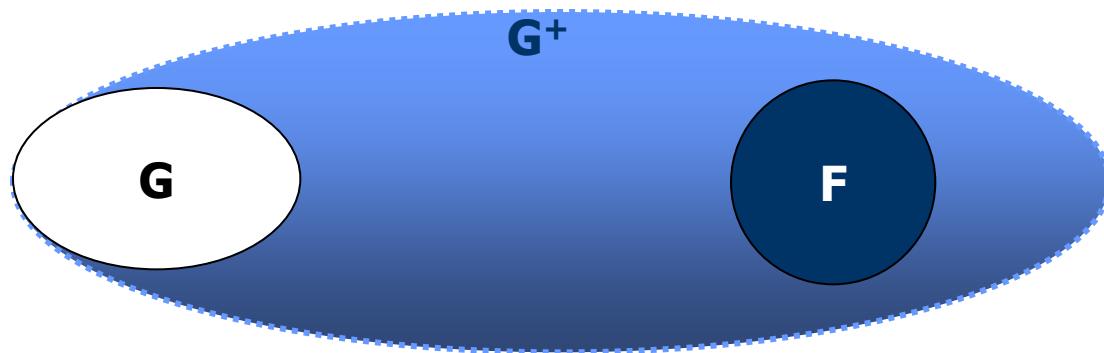
Verificação se uma DF pertence a F^+

Como provar que $X \rightarrow Y$ pertence a F^+ ?

- Um caminho será calcular F^+ e verificar se $X \rightarrow Y$ pertence a esse conjunto – como foi visto, tal tarefa pode ser quase impraticável!
- Outro caminho passa por utilizar as Regras de Armstrong
 - Considere-se o lado esquerdo da DF – X
 - Partindo de F e aplicando as Regras de Armstrong, encontrar o conjunto W que contem todos os atributos funcionalmente dependentes de X
 - Se Y estiver contido em W então pode concluir-se que $X \rightarrow Y$ pertence a F^+ .
- Exemplo:
 - Sendo $F=\{A \rightarrow BD, AB \rightarrow C\}$ verificar se $A \rightarrow C$ pertence a F^+
 - $A^+ = \{A, B, C, D\}$ como $C \subset A^+$ então $A \rightarrow C$ pertence a F^+

Cobertura de um conjunto de DF

- Sejam G e F dois conjuntos de Dependências Funcionais
- Diz-se que G é cobertura de F se
 - G^+ contiver todas as DF de F



- G^+ pode ser maior que F^+ ($G^+ \supset F^+$)

Verificação da Cobertura de F

Como verificar se G é ou não cobertura de F?

- Um primeiro caminho pode ser calcular G^+ e F^+
 - Se G^+ contiver F^+ , então G é cobertura de F
 - **O cálculo de G^+ e F^+ pode ser muito difícil!**
- Pode ser verificado de uma forma sistemática, fazendo:
 - Para cada Dependência Funcional $X \rightarrow Y$ em F
 - Usando as DF de G calcular X^+
 - Se $X^+ \supseteq Y$ então continuar o ciclo
 - Caso contrário parar. G não é cobertura de F

Verificação da Cobertura de F - exemplo

- Para
 - $G = \{A \rightarrow BC, C \rightarrow D\}$
 - $F = \{AB \rightarrow C, CE \rightarrow D, CA \rightarrow ACD\}$
- Verificar se G é cobertura de F
 - Lados esquerdos de F – { AB, CE, CA }
 - Usando G, determinar o Fecho da cada um
 - $AB^+ = \{ABCD\}$ – inclui C que é lado direito em F
 - $CE^+ = \{CDE\}$ – inclui D que é lado direito em F
 - $CA^+ = \{ABCD\}$ – inclui ACD que é lado direito em F

Conclui-se que G é cobertura de F!

Equivalências entre Dependências Funcionais

- Se tivermos dois conjuntos G e F, diz-se que são equivalentes ($G \equiv F$) quando $G^+ = F^+$



- Se se conseguir determinar que G é cobertura de F ($G^+ \supseteq F^+$) e que F é cobertura de G ($F^+ \supseteq G^+$), então $G \equiv F$

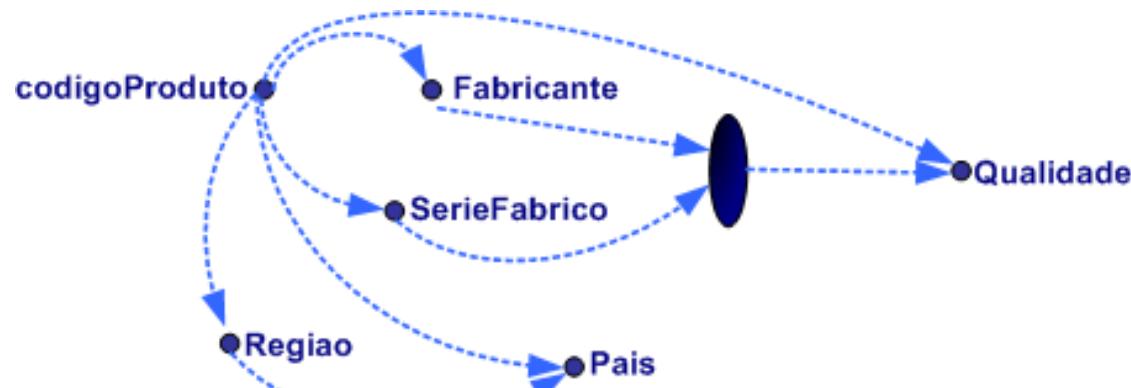
Cobertura Mínima - Motivação

- Trabalhar com F^+ revela-se difícil, pois tem um grande número de DF
 - quanto menos DF se tiver para trabalhar melhor
- F^+ tem um grande número de DF redundantes
 - é necessário eliminá-las
- Partindo de F é possível encontrar um conjunto de DF mínimo, designado de **Cobertura Mínima** (CM), a partir do qual é possível derivar todas as DF de F
- Ou seja $F \equiv CM$, que quer dizer que $F^+ = CM^+$
- CM não contêm DF redundantes
- CM é o conjunto mais pequeno que se consegue obter sem perder nenhuma DF

Cobertura Mínima (cont.)

- Diz-se que G é Cobertura Mínima de F se
 - G é cobertura de F
 - Em G o lado direito de todas as DF apenas têm um atributo
 - Em G o lado esquerdo de todas as DF não têm nenhum atributo redundante
 - Qualquer DF que se tire de F, faz com que G deixe de ser equivalente a F
- Formalmente
 - $G^+ \supseteq F$
 - $\forall X \rightarrow A \in G$ então A é um atributo singular (ou seja é o único atributo)
 - $\neg \exists X \rightarrow A \in G$, tal que $Z \subset X \{ \{G - \{X \rightarrow A\}\} \cup \{Z \rightarrow A\} \}^+ = F^+$
 - $\neg \exists X \rightarrow A \in G$, tal que $Z \subset X \{G - \{X \rightarrow A\}\}^+ = F^+$

Cobertura Mínima - Exemplo



Conjunto F

Cobertura Mínima



Algoritmo da Cobertura Mínima

- Entrada – Conjunto de Dependências Funcionais F
 - Saída – Cobertura Mínima
1. considerar F uma Cobertura dele mesmo e chamar-lhe G
 - $G := F$
 2. tornar singular cada Dependência Funcional de G
 - Substituir cada DF $X \rightarrow (y_1, y_2, \dots, y_n)$ por n DF: $X \rightarrow Y_1, X \rightarrow Y_2, \dots, X \rightarrow Y_n$
 3. eliminar atributos redundantes nos lados esquerdos das DF de G
 - para cada $X \rightarrow A$ em G,
 - $Z := X$
 - Para cada $B \in X$
 - se, com base em G, tivermos $A \in (Z - B)^+$, então $Z := Z - B$
 - Substituir $X \rightarrow A$ por $Z \rightarrow A$

Algoritmo da Cobertura Mínima (cont.)

4. encontrar um menor subconjunto G de DF que é equivalente a F
 - o para cada $X \rightarrow A \in G$
 - Calcular X^+ a partir de $\{G - X \rightarrow A\}$
 - Se $A \in X^+$ então remover de G a DF $X \rightarrow A$

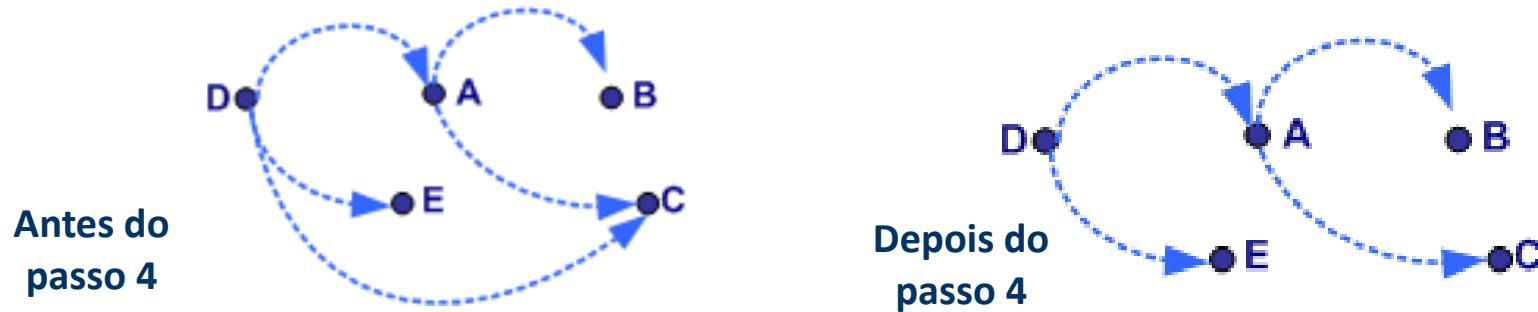
Aplicando o algoritmo descrito a um conjunto qualquer de F, obtêm-se outro conjunto de DF equivalente, menor ou igual ao de partida, sem DF redundantes!

Cobertura Mínima - Exemplo

- Determinar a Cobertura Mínima de
 $F = \{ D \rightarrow AC, A \rightarrow B, AB \rightarrow C, AD \rightarrow CE \}$?
- Aplicar o passo 2. – decompor todas as DF cuja parte direita tenha mais que um atributo
 - $G = \{ D \rightarrow A, D \rightarrow C, A \rightarrow B, AB \rightarrow C, AD \rightarrow C, AD \rightarrow E \}$
- Aplicar o passo 3. – Retirar da parte esquerda os atributos redundantes
 - $A^+ = \{ABC\}$, donde $AB \rightarrow C$ e $AD \rightarrow C$ são substituídos por $A \rightarrow C$
 - $D^+ = \{ABCDE\}$, donde $AD \rightarrow E$ é substituído por $D \rightarrow E$
- Aplicar o passo 4. – Encontrar o menor subconjunto de G
 - $G = \{D \rightarrow A, D \rightarrow C, A \rightarrow B, A \rightarrow C, D \rightarrow E\}$
 - Com $\{D \rightarrow A, D \rightarrow C, A \rightarrow C, A \rightarrow B, D \rightarrow E\}$ $D^+ = \{ABCDE\}$
 - Com $\{D \rightarrow A, A \rightarrow C, A \rightarrow B, D \rightarrow E\}$ $D^+ = \{ABCDE\}$
 - Ou seja, $D \rightarrow C$ é redundante

Cobertura Mínima – Exemplo (cont.)

- (cont.) $G=\{D \rightarrow A, D \rightarrow C, A \rightarrow B, A \rightarrow C, D \rightarrow E\}$
 - Com $\{D \rightarrow A, A \rightarrow C, D \rightarrow E\}$ $A+=\{AC\}$
 - Com $\{D \rightarrow A, A \rightarrow B, D \rightarrow E\}$ $A+=\{AB\}$
 - Com $\{D \rightarrow A, A \rightarrow C, A \rightarrow B\}$ $D+=\{ABCD\}$
 - Com $\{A \rightarrow C, A \rightarrow B, D \rightarrow E\}$ $D+=\{DE\}$
- Pelo que a cobertura mínima G é dada por
- $G=\{D \rightarrow A, A \rightarrow B, A \rightarrow C, D \rightarrow E\}$



Cobertura Mínima – 2º Exemplo

É possível existirem dois conjuntos G1 e G2, diferentes, sendo ambos cobertura mínima de F?

- Determinar a Cobertura Mínima de $F = \{ A \rightarrow BC, B \rightarrow C, C \rightarrow B \}$?
- Aplicar o passo 2. $G = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
- Aplicar o passo 3. $G = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
- Aplicar o passo 4. $G = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
 - com $\{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$, $A+ = \{ABC\}$
 - com $\{A \rightarrow C, B \rightarrow C, C \rightarrow B\}$, $A+ = \{ABC\}$
 - $A \rightarrow B$ é Redundante, ficando $G = \{A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
 - com $\{A \rightarrow B, A \rightarrow C, C \rightarrow B\}$, $B+ = \{B\}$
 - com $\{A \rightarrow B, A \rightarrow C, B \rightarrow C\}$, $C+ = \{C\}$

$G = \{A \rightarrow C, B \rightarrow C, C \rightarrow B\}$ é uma cobertura Mínima

Cobertura Mínima – 2º Exemplo (cont.)

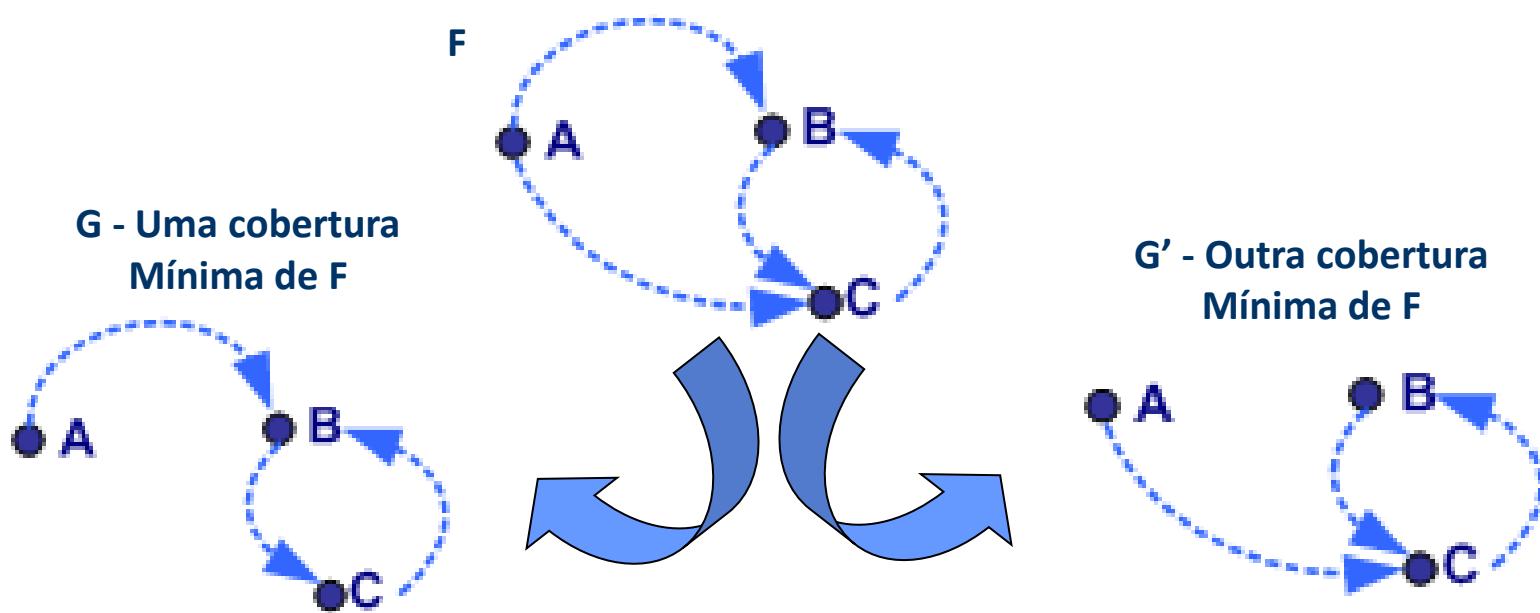
Refazendo, mas modificando o passo quatro

- Determinar a Cobertura Mínima de $F = \{ A \rightarrow BC, B \rightarrow C, C \rightarrow B \}$?
- Aplicar o passo 2. $G = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
- Aplicar o passo 3. $G = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
- Aplicar o passo 4. $G = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
 - com $\{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$, $A+ = \{ABC\}$
 - com $\{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$, $A+ = \{ABC\}$
 - $A \rightarrow C$ é redundante, ficando $G = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$
 - com $\{A \rightarrow B, A \rightarrow C, C \rightarrow B\}$, $B+ = \{B\}$
 - com $\{A \rightarrow B, A \rightarrow C, B \rightarrow C\}$, $C+ = \{C\}$

$G = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$ também é uma cobertura Mínima

Cobertura Mínima - 3º Exemplo

- $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$
- $G = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$ é Cobertura Mínima
- $G' = \{A \rightarrow C, B \rightarrow C, C \rightarrow B\}$ é Cobertura Mínima



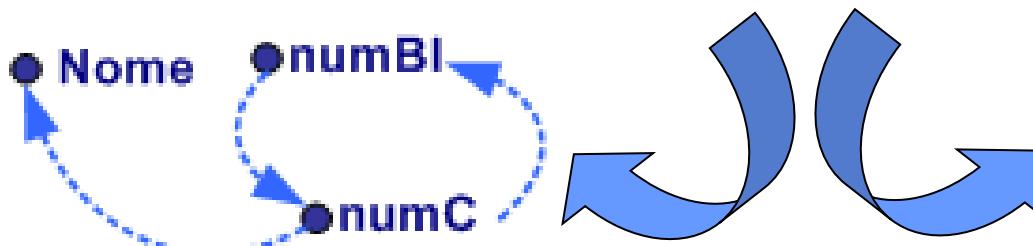
Cobertura Mínima – 4º exemplo

- $F = \{ \text{numBI} \rightarrow \text{nome}, \text{numBI} \rightarrow \text{numC}, \text{numC} \rightarrow \text{nome}, \text{numC} \rightarrow \text{numBI} \}$

G - Uma cobertura
Mínima de F



G' - Outra cobertura Mínima
de F



Chaves candidatas e superchave

- Seja um conjunto F contendo as seguintes DF:
 - $AB \rightarrow C$
 - $C \rightarrow D$
 - $D \rightarrow A$

Para uma dada Relação R(A,B,C,D), com as DF acima enumeradas, quais as chaves candidatas?

- Chaves candidatas: AB,BC,BD
- Superchaves: ABC,ABD,BCD

Chaves candidatas e superchave (cont.)

- Seja $R(A_1, A_2, \dots, A_n)$ e $X \not\rightarrow \{A_1, A_2, \dots, A_n\}$ e F um conjunto de DF
 - X é superchave de R sse:
 - $X \rightarrow A_1, A_2, \dots, A_n \in F^+$
 - X é chave candidata de R quando:
 - X é superchave
 - X é o conjunto mínimo de atributos que pode ser superchave em R
 - Formalmente:
 - $X \rightarrow A_1, A_2, \dots, A_n \in F^+$
 - $\neg \exists Y \subset X, Y \rightarrow A_1, A_2, \dots, A_n \in F^+$

Chaves candidatas e superchave (cont.)

- A Chave Candidata será no limite composta por todos os Atributos
 - $(A_1, A_2, \dots, A_n) \rightarrow A_1, A_2, \dots, A_n$
- Diz-se que um atributo A é primo se ele pertencer a alguma das chaves candidatas
- Qualquer Esquema de Relação com um determinado conjunto de DF tem pelo menos uma chave candidata
- Como já foi abordado anteriormente, a chave primária é eleita entre as chaves candidatas
- A determinação de todas as chaves candidatas passa por determinar todos os subconjuntos $W \subset \{A_1, A_2, \dots, A_n\}$ e escolher os menores tal que:
 - $W^+ = \{A_1, A_2, \dots, A_n\}$

Chaves candidatas

- Determinar todos os subconjuntos de um conjunto de atributos pode ser uma tarefa demorada e que à partida não garante a utilidade de alguns desses subconjuntos
- É necessário recorrer a algumas heurísticas para facilitar o trabalho:
 - Os atributos que só aparecem do lado esquerdo pertencem de certeza às chaves candidatas – Nenhum outro os determina
 - Os atributos que não pertencem nem ao lado esquerdo nem ao direito pertencem de certeza às chaves candidatas – Apenas são determinados por eles próprios
 - Os atributos que só aparecem do lado direito não pertencem de certeza às chaves candidatas – Não determinam nenhum outros
 - Para os restantes, é necessário efectuar uma análise caso a caso para se determinar se pertencem ou não a alguma das chaves candidatas

Chaves candidatas – Exemplo

- $R(\text{numSeq}, \text{numFact}, \text{ano}, \text{numLinha}, \text{codProd}, \text{quantidade})$
- $F=\{\text{numSeq} \rightarrow \text{numFact}, \text{numSeq} \rightarrow \text{ano},$
 $\quad (\text{ano}, \text{numFact}) \rightarrow \text{numSeq},$
 $\quad (\text{ano}, \text{numFact}, \text{numLinha}) \rightarrow \text{codProd},$
 $\quad (\text{ano}, \text{numFact}, \text{numLinha}) \rightarrow \text{quantidade}\}$
- Quais as chaves candidatas de R ?
 - numLinha só aparece do lado esquerdo das DF – Pertence às chaves candidatas
 - ano, numFact, numSeq aparecem em ambos os lados

Chaves candidatas – Exemplo (cont.)

- Como foi visto, o atributo numLinha tem de pertencer à chave
 - $\text{numLinha}^+ \neq \{\text{todos os atributos}\}$, logo não é, sozinho, Chave Candidata
 - Se fosse seria a única
- É necessário encontrar os menores subconjuntos que contêm numLinha e cujo fecho contenha todos os atributos de R
 - $\text{numLinha} \cup \{\text{ano, numFact, numSeq}\}$
- Tendo como objectivo encontrar o menor conjunto possível, calcula-se o fecho para cada um dos subconjuntos de $\{\text{numLinha, ano, numFact, numSeq}\}$

Chaves candidatas – Exemplo (cont.)

- $(\text{numLinha}, \text{ano})^+ \neq \{\text{atributos de R}\}$, logo não é Chave Candidata
- $(\text{numLinha}, \text{numSeq})^+ = \{\text{atributos de R}\}$, logo é uma Chave Candidata
- $(\text{numLinha}, \text{numFact})^+ \neq \{\text{atributos de R}\}$, logo não é Chave Candidata
- $(\text{numLinha}, \text{numFact}, \text{ano})^+ = \{\text{atributos de R}\}$, logo é uma Chave Candidata
- As Chaves Candidatas para a Relação R são
 - $(\text{numLinha}, \text{numSeq})$ e $(\text{numLinha}, \text{numFact}, \text{ano})$

Normalização - Objectivo

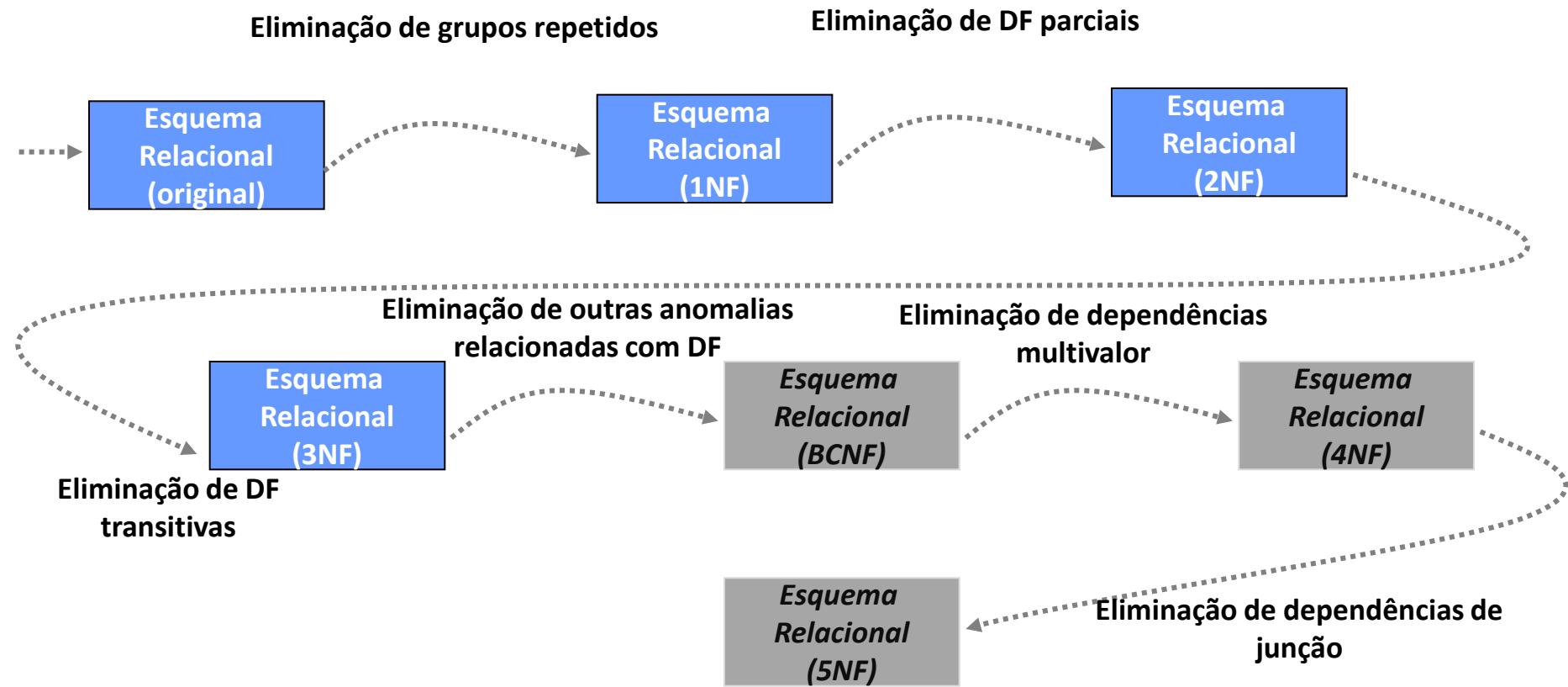
- Num projecto de Base de Dados, pretende-se construir o suporte de dados de um determinado problema
 - É necessário modelar o problema, primeiro num nível alto de abstracção – O Modelo Conceptual, livre de qualquer detalhe de implementação e que, tipicamente, é representado utilizando o modelo Entidade-Associação
 - Posteriormente a essa modelação é necessário passar para o Modelo Relacional (se for essa a infra-estrutura de suporte de dados), deferindo o Esquema Relacional para o problema
- A informação é representada nesse Esquema Relacional de uma forma natural, tal qual foi apresentada no problema.

Normalização – Objectivo (cont.)

Será que o esquema Relacional encontrado é o melhor?
Será que não contém alguma redundância indesejada?

- Neste Ponto entra a Normalização.
- Com o objectivo claro de melhorar a qualidade do Esquema Relacional, este é avaliado de acordo com um conjunto de regras
- Se algumas destas regras não se verificarem, o Esquema Relacional é transformado, por forma a que o Esquema se torne menos redundante e mais estável

Normalização - Etapas



Normalização - Exemplo

- Pretende-se armazenar informação sobre clientes e as suas encomendas
- Para os clientes
 - Número de cliente (único), nome, morada
- Para os produtos
 - Código do produto (único), designação
- Para as encomendas
 - Número de encomenda (único), data de encomenda, dados do Cliente, dados dos produtos encomendados, quantidade de cada produto

Encomenda(numEnc, numcli, cliente, morada, dataEnc, codProd, produto, quantEnc)

Normalização – Exemplo (cont.)

numEnc	numCli	cliente	morada	dataEnc	codProd	produto	quantEnc
1	1	Antonio	Rua 1	10-03-2002	1	Batatas	2
1	1	Antonio	Rua 1	10-03-2002	2	Feijão	1
1	1	Antonio	Rua 1	10-03-2002	3	Couves	1
2	2	Maria	Rua 2	12-04-2002	1	Batatas	5
2	2	Maria	Rua 2	12-04-2002	2	Feijão	1
3	3	João	Rua 3	12-05-2001	4	Laranjas	4
3	3	João	Rua 3	12-05-2001	1	Batatas	2

- A tabela acima ilustra a informação que pode ser armazenada
- Da forma como é apresentada, colocam-se alguns problemas relacionados com a redundância
- Com a Normalização, esses problemas serão resolvidos

1ª Forma Normal - 1NF

Um Esquema de Relação **está** na 1NF se os valores para o seu domínio forem atómicos

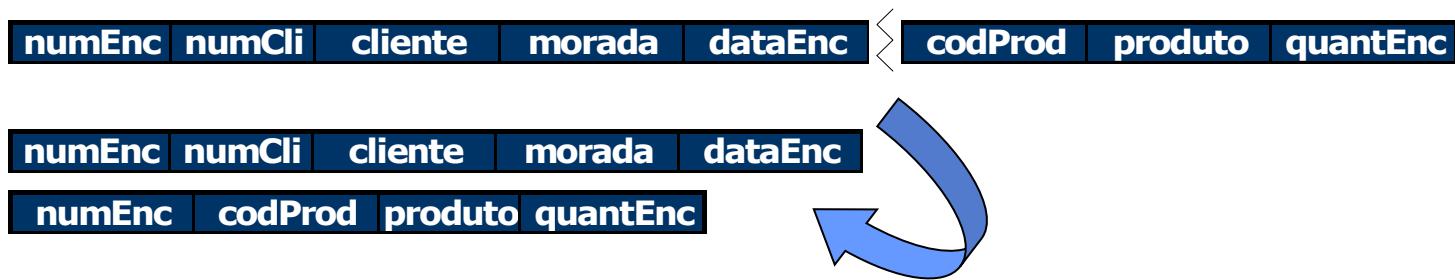
- Ou, por outras palavras, um Esquema de Relação está na 1NF se não existirem “Relações dentro de Relações”
- A Relação anterior pode ser visto como:

numEnc	numCli	cliente	morada	dataEnc	codProd	produto	quantEnc
1	1	Antonio	Rua 1	10-03-2002	1	Batatas	2
					2	Feijão	1
					3	Couves	1
2	2	Maria	Rua 2	12-04-2002	1	Batatas	5
					2	Feijão	1
3	3	João	Rua 3	12-05-2001	4	Laranjas	4
					1	Batatas	2

- Os atributos codProd, produto e quantEnc não são atómicos !
- Isto acontece pois cada encomenda tens vários produtos

1ª Forma Normal - 1NF (cont.)

- A forma de colocar o Esquema de Relação na 1NF é decompondo, até que todos os Esquemas de Relação recém-criados estejam na 1NF
- No exemplo, o Esquema de Relação tem de ser decomposto em dois



- Como ambos os novos Esquemas de Relação estão na 1NF, não é necessário continuar a decompor
- Isso não quer dizer, que não exista ainda redundância!

1ª Forma Normal - 1NF (cont.)

- Esquema de Relação Encomenda:
 - Encomenda(numEnc, numcli, cliente, morada, dataEnc)
 - Onde a chave candidata/chave primária é numEnc
- Esquema de Relação Linhas_Encomenda:
 - Linhas_Encomenda (numEnc, codProd, produto, quantEnc)
 - Onde a chave candidata/chave primária é composta por numEnc,codProd

numEnc	numCli	cliente	morada	dataEnc
1	1	Antonio	Rua 1	10-03-2002
2	2	Maria	Rua 2	12-04-2002
3	3	João	Rua 3	12-05-2001

Encomenda

numEnc	codProd	produto	quantEnc
1	1	Batatas	2
1	2	Feijão	1
1	3	Couves	1
2	1	Batatas	5
2	2	Feijão	1
3	4	Laranjas	4
3	1	Batatas	2

Linhas_Encomenda

1ª Forma Normal - 1NF (cont.)

Será que os problemas ficaram todos resolvidos?

- Anomalia de inserção
 - Pretende-se adicionar um novo produto para que possa estar disponível para novas encomendas
 - Mas como numEnc faz parte da chave da tabela, não é possível adicionar um novo produto sem existir uma encomenda para ele !
- Anomalia de remoção
 - Se se pretender remover informação sobre um determinado produto, perde-se informação desse produto nas encomenda
- Anomalia de alteração
 - Se se pretender alterar a nome do produto ‘Feijão’ para ‘Feijão Frade’, é necessário percorrer um conjunto de tuplos e proceder a essa alteração em cada um deles
- Estas anomalias serão resolvidas recorrendo ás 2NF e 3NF

2ª Forma Normal - 2NF

Quando um ou mais atributos não primos dependem parcialmente da chave, o Esquema de Relação não está na 2NF

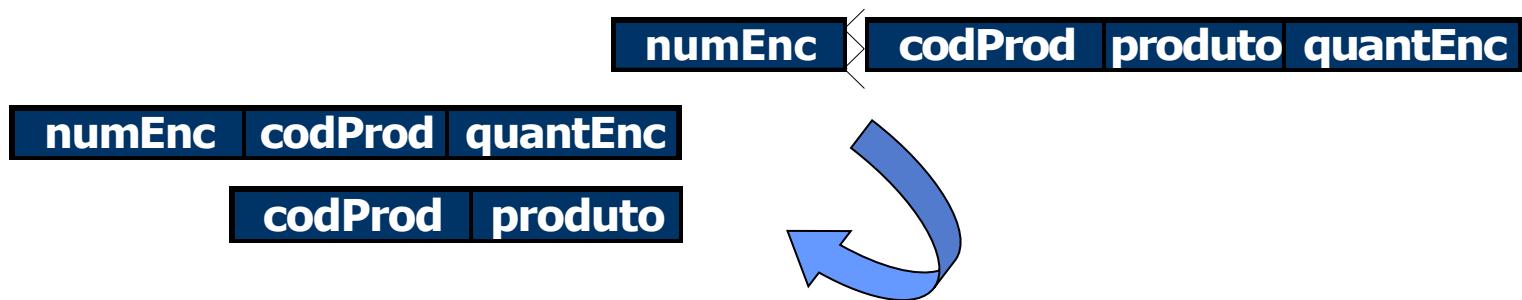
- Quando um Esquema de Relação não está na 2NF é porque existem DF que não são totais
- Uma DF $X \rightarrow Y$ diz-se total quando
 - $\neg \exists Z \subset X (X - \{Z\}) \rightarrow Y$
 - Ou seja, Y não é Funcionalmente dependente de nenhum subconjunto de X
- Pelo contrário uma DF $X \rightarrow Y$ diz-se parcial quando
 - $\exists Z \subset X (X - \{Z\}) \rightarrow Y$

2^a Forma Normal - 2NF (cont.)

- Formalmente:
- Seja $R(A_1, A_2, \dots, A_n)$
- R está na 2NF se
 - R está na 1NF
 - $\forall X \in \{\text{Chaves Candidatas}\} \text{ e } \forall A_i \notin \{\text{Chaves Candidatas}\}, \text{ a DF } X \rightarrow A_i \text{ é total}$
- Sempre que um Esquema de Relação está na 1NF e tem uma chave constituída por um único atributo esse Esquema de Relação está na 2NF
- No exemplo existem dependências parciais:
 - O atributo *produto*, no Esquema *linhas_encomenda* depende parcialmente da chave, uma vez que apenas o *codProd* o determina

2^a Forma Normal - 2NF (cont.)

- Mais uma vez, é necessário decompor os Esquemas de Relação para resolver as anomalias subjacentes às DF parciais
- O Esquema linhas_encomenda será decomposto



- Para o primeiro esquema entram a chave e os atributos que dela dependem totalmente
- No outro esquema ficam os atributos que dependem parcialmente da chave, mais essa parte da chave

2^a Forma Normal - 2NF (cont.)

- Esquema de Relação Linhas_Encomenda:
 - Linhas_Encomenda(numEnc, codProd, quantEnc)
 - Onde a chave candidata/chave primária é composta por numEnc,codProd
- Esquema de Relação Produto
 - Produto(codProd, produto)
 - Onde a chave candidata/chave primária é codProd

Linhos_Encomenda

numEnc	codProd	quantEnc
1	1	2
1	2	1
1	3	1
2	1	5
2	2	2
3	4	4
3	1	2

Produto

codProd	produto
1	Batatas
2	Feijão
3	Couves
4	Laranjas

2^a Forma Normal - 2NF (cont.)

Será que os problemas ficaram todos resolvidos?

- Anomalia de inserção
 - Pretende-se adicionar um novo Cliente. Tal não é possível até que seja criada uma nova encomenda. Mas não se pode criar uma encomenda para um cliente que não existe!
- Anomalia de remoção
 - Se se pretender remover informação sobre um determinado cliente, perde-se informação da encomenda
- Anomalia de alteração
 - Se se pretender alterar a morada de um cliente, é necessário percorrer um conjunto de tuplos e proceder a essa alteração em cada um deles
- Estas anomalias serão resolvidas recorrendo á 3NF

3ª Forma Normal - 3NF

Um Esquema de Relação está na 3NF se não existir nenhum atributo funcionalmente dependente de um atributo não primo

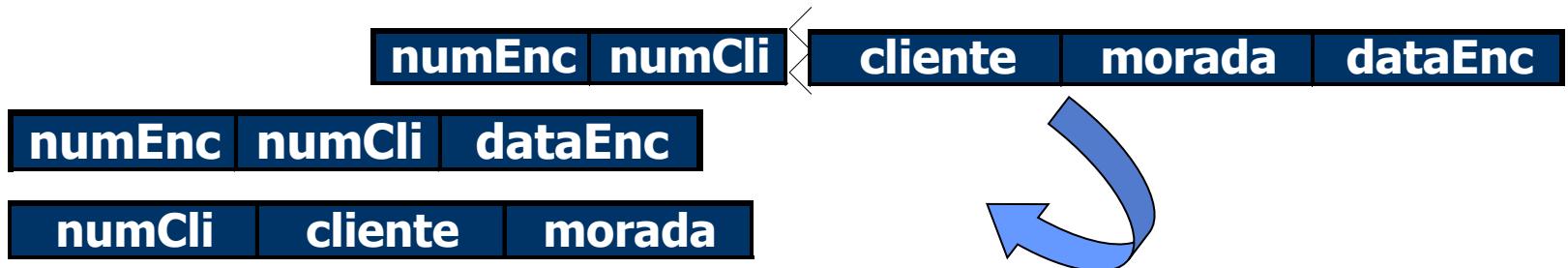
- Quando um Esquema de Relação está na 3NF é porque não existem DF transitivas
- Uma dependência $X \rightarrow A$ diz-se transitiva sse
 - $\exists Y \subseteq \{A_1, A_2, \dots, A_n\}$ tal que $X \rightarrow Y, Y \rightarrow A, Y \not\rightarrow X, A \notin X$
 - Ou seja não se consegue derivar $X \rightarrow A$ aplicando o 3º Axioma de Armstrong
- Uma DF parcial é um caso particular da DF transitiva
 - Se Y for um subconjunto de X ($Y \subset X$), tem-se que $X \rightarrow A$ é também uma DF parcial
- Então, um Esquema de Relação na 3NF está também na 2NF

3ª Forma Normal - 3NF (cont.)

- Por outro lado, se um Esquema de Relação está na 3NF:
 - Para $X \rightarrow A$, com $A \notin X$
 - Ou X é superchave
 - Ou A é um Atributo Primo
- No exemplo, para os Esquemas produto e linhas_encomenda, não se coloca este problema, uma vez que têm apenas por chave um atributo não primo
- No entanto existem dependências transitivas:
 - No Esquema encomenda podem tirar-se estas duas DF
 - $\text{numEncomenda} \rightarrow \text{numCliente}$
 - $\text{numCliente} \rightarrow \text{cliente,morada}$
 - A chave da tabela, como já foi visto, é numEncomenda , o que quer dizer que existem dependências transitivas da chave
 - Um atributo não primo, numcliente , determina atributos

3^a Forma Normal - 3NF (cont.)

- Mais uma vez, é necessário decompor os Esquemas de Relação para resolver as anomalias subjacentes às DF transitivas
- O Esquema encomenda será decomposto



- Para o primeiro esquema entram a chave e os atributos que dela dependem totalmente
- No outro esquema ficam os atributos que dependem transitivamente da chave mais o atributo que os determina

3^a Forma Normal - 3NF (cont.)

- Esquema de Relação Encomenda:
 - Encomenda(numEnc, numCli, dataEnc)
 - Onde a chave candidata/chave primária é numEnc
- Esquema de Relação Cliente
 - Cliente(numCli, cliente, morada)
 - Onde a chave candidata/chave primária é numCli

Encomenda

numEnc	numCli	dataEnc
1	1	10-03-2002
2	2	12-04-2002
3	3	12-05-2001

Cliente

numCli	cliente	morada
1	Antonio	Rua 1
2	Maria	Rua 2
3	João	Rua 3

3ª Forma Normal - 3NF (cont.)

- Existem alguns problemas neste “método intuitivo” de Decomposição
- Considerando $R(A, B, C, D, E)$ e $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$
 - Pretende-se colocar o Esquema R na 3NF
 - A chave de R será dada por {AB}
- Uma possível Decomposição de $R(A, B, C, D, E)$ na 3FN será:
 - $R1 (A, B, C)$
 - $R2 (D, E)$
- Mas será que com R1 e R2 ainda se têm todo o F ?
- Perdeu-se a DF $C \rightarrow D$!

Preservação da Dependências Funcionais

- Quando se normaliza um Esquema de Relação é importante que se mantenham todas as DF existentes em F
- Essa DF podem ser inferidas
 - Num dos Esquemas de Relação resultantes da decomposição
 - Ou a partir de alguma DF que apareça nesses Esquemas
- Quando isto acontece, diz-se que houve preservação das Dependências Funcionais
- É importante que se preservem as DF, pois elas representam restrições que os dados devem respeitar

Projecção de DF num Esquema de Relação

- Seja F o conjunto de DF de um Esquema de Relação R e uma decomposição D de R dada por $D=\{R_1, R_2, \dots, R_n\}$
- A projecção de F em R_i escreve-se $\pi_{R_i}(F)$ e consiste em
 - $X \rightarrow Y \in F^+, X \cup Y \subseteq R_i$
- Seja $R(A,B,C,D,E,F)$, $F=\{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$ e $D=\{R1(\underline{A},\underline{B},D), R2(\underline{C},E)\}$
- $C \rightarrow E$ pertence a $\pi_{R2}(F)$?
 - $C \rightarrow E \in F^+, C \cup E \subseteq R2$, pelo que é verdade
- $AB \rightarrow D$ pertence a $\pi_{R1}(F)$?
 - $AB \rightarrow D \in F^+, AB \cup D \subseteq R1$, pelo que é verdade
- $C \rightarrow D$ pertence a $\pi_{R1}(F)$?
 - $C \bowtie D \in F^+, C \cup D \not\subseteq R1, D \not\subseteq R2$, pelo que é falso

Preservação de DF num Esquema de Relação (cont.)

- Uma decomposição D de R diz-se que preserva DF relativamente a um conjunto F de DF em R se
 - a união das Projeções de F em cada R_i for equivalente a F, ou seja,
 - $(\pi_{R_1}(F) \cup \dots \cup \pi_{R_n}(F))^+ = F^+$
- Seja $R(A,B,C,D,E,F)$, $F=\{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$ e $D=\{R1(\underline{A},B,D), R2(\underline{C},E)\}$
- Analisando D:
 - $\pi_{R1}(F)=(AB \rightarrow D)^+$
 - $\pi_{R2}(F)=(C \rightarrow E)^+$
 - $C \rightarrow D \notin (\pi_{R1}(F) \cup \pi_{Rn}(F))^+$
 - $C \rightarrow D \in F$, pelo que a decomposição D não preserva dependências!

Preservação de informação (lossless join)

- Depois da decomposição, é importante assegurar que através de uma junção natural dos Esquemas de Relação obtidos, se consegue obter a informação do Esquema de Relação original
- Quando isso acontece diz-se que a decomposição preserva informação
- Uma decomposição onde isso acontece, tem a propriedade “sem perda por junção” (*lossless join*)

Preservação de informação (lossless join) – como verificar

- O algoritmo para a verificação do *lossless join*:
Entrada: F , $R(A_1, A_2, \dots, A_n)$ e um decomposição $D=\{R_1, R_2, \dots, R_n\}$
Saída: Verificação ou não da propriedade *lossless join*
1. Construir a Matriz $M(i,j)$ com linhas i para cada R_i e coluna j para cada A_j tal que
 - Se $A_j \in R_i$, $M(i,j) = a_j$
 - Se $A_j \notin R_i$, $M(i,j) = b_{ij}$
 2. Para cada $X \rightarrow Y$ em F , procurar linhas com o mesmo valor em todas as colunas de X e para essas colunas igualar os valores de Y do seguinte modo
 - se um dos valores for a_j , igualar os restantes a a_j
 - se os valores forem apenas b_{ij} , igualar a um qualquer dos b_{ij}
 3. Se existir uma linha i apenas com valores a_i , conclui-se que a Decomposição é *Lossless Join*

Preservação de informação (lossless join) – Exemplo

Considerando $R(A, B, C, D, E, F)$ e $F = \{A \rightarrow B, C \rightarrow DF, AC \rightarrow E\}$ e $D=\{R1(A,B), R2(C,D,F), R3(A,C,E)\}$, será que existe preservação de informação?

1. Construção da Matriz $M(i,j)$

$M(i,j)$	A	B	C	D	E	F
R1(A,B)	a1	a2	b13	b14	b15	b16
R2(C,D,F)	b21	b22	a3	a4	b25	a6
R3(A,C,E)	a1	b32	a3	b34	a5	b36

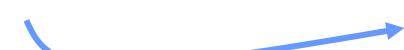
2. Verificação para cada DF de F

- $A \rightarrow B$



	A	B	C	D	E	F
R1(A,B)	a1	a2	b13	b14	b15	b16
R2(C,D,F)	b21	b22	a3	a4	b25	a6
R3(A,C,E)	a1	a2	a3	b34	a5	b36

- $C \rightarrow DF$



	A	B	C	D	E	F
R1(A,B)	a1	a2	b13	b14	b15	b16
R2(C,D,F)	b21	b22	a3	a4	b25	a6
R3(A,C,E)	a1	a2	a3	a4	a5	a6

A Decomposição é lossless join !! (última linha da matriz toda ai)

Algoritmo de passagem para a 3NF

- Pretende-se que na passagem para a 3NF não se perca nenhuma DF e que as decomposições gozem da propriedade *lossless join*
- **Entrada:** Um Esquema de Relação $R(A_1, \dots, A_n)$ e um conjunto de DF F
- **Saída:** Decomposição $D=\{R_1, \dots, R_N\}$ na 3FN, com preservação de DF e *lossless join*
- Os passos do algoritmo são:
 1. Partir do menor conjunto de DF (G) que é equivalente a F (cobertura mínima)
 2. A partir de G , construir iterativamente uma decomposição D que preserva as DF de F
 3. Garantir que essa decomposição é *lossless join*

Algoritmo de passagem para a 3NF (cont.)

1. Encontrar a cobertura mínima de F, designando-a de G
2. Para cada DF funcional $X \rightarrow A$ existente em G
 - No inicio D não tem nenhum Esquema de Relação
 - Se não existe em D nenhum Esquema de Relação que contenha X e A, adiciona-se uma nova decomposição às existentes dada por $R_i(X,A)$
3. Verificar se em D existe algum esquema que inclua todos os atributos pertencentes a uma Chave Candidata de R
 - Se tal não acontece, adiciona-se a D um novo Esquema de Relação com os atributos da Chave Candidata de R

Algoritmo de passagem para a 3NF – Exemplo 1

Enunciado:

- Seja $R(A, B, C, D, E)$ e $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E, CD \rightarrow E\}$
- Pretende-se encontrar uma decomposição que esteja na 3NF

Resolução:

1. Determinar G de F (a Cobertura Mínima)
 - $G = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$
2. Decompor com Base em G
 - $D = \{R_1(A, B, C), R_2(C, D), R_3(D, E)\}$
3. Garantir que D é *lossless join*
 - Chave Candidata de R é (AB) , pois $(AB)^+ = \{ABCDE\}$
 - Verificar se essa chave aparece em algum dos Esquemas de Relação presentes em D
 - R_1 contém A e B pelo que verifica

Esta decomposição está na 3NF, preserva DF e é lossless join !

Algoritmo de passagem para a 3NF – Exemplo 2

Enunciado

- Seja $R(A, B, C, D, E)$ e $F = \{A \rightarrow BCD, CD \rightarrow E\}$

Resolução:

- Pelo processo intuitivo
 - $D=\{R_1(A,B,C,D), R_2(C,D,E)\}$
- Pelo algoritmo 3NF
 - $D=\{R_1(A,B), R_2(A,C), R_3(A,D), R_4(C,D,E)\}$
- Em ambos os casos
 - São preservadas DF
 - D é *lossless join*
- Neste exemplo o processo intuitivo chegou a uma solução que tem menos Esquemas de Relação.



No entanto não é garantido que pelo método intuitivo se cheguem sempre a decomposições na 3NF onde se preserva DF e *lossless join* !

Forma Normal de Boyce-Cood - Motivação

- Considere-se o seguinte cenário:
 - Um estudante pode frequentar, simultaneamente, vários anos
 - Cada estudante, para cada ano que frequenta, está afecto a um professor responsável
 - A cada ano estão afectos vários professores responsáveis
 - No entanto, cada professor é responsável apenas por um ano
- Dependências funcionais
 - $(\text{numAluno}, \text{anoEscolar}) \rightarrow \text{professorResponsavel}$
 - $\text{professorResponsavel} \rightarrow \text{anoEscolar}$

Forma Normal de Boyce-Cood – Motivação (Cont.)

- Chaves candidatas
 - $(\text{numAluno}, \text{anoEscolar})^+ = \{\text{todos os atributos}\}$
 - $(\text{numAluno}, \text{professorResponsavel})^+ = \{\text{todos os atributos}\}$
- Esquema de Relação na 3NF
 - Responsavel(numAluno,anoEscolar,professorResponsavel)

numAluno	anoEscolar	professorResponsavel
111111	1	Joaquim
222222	3	João
111111	2	Luis
333333	2	Hugo
333333	3	João
333333	4	Manuel

Forma Normal de Boyce-Cood – Motivação (Cont.)

- Embora na 3NF, o Esquema de Relação anterior tem alguns problemas, nomeadamente:
 - Anomalia de inserção
 - Se se pretender acrescentar mais um professor responsável a um ano, por ex. Prof. Carlos responsável pelo 1º ano, só se consegue inferir essa informação quando lhe for atribuído pelo menos um aluno
 - Anomalia de remoção
 - Se para um determinado ano, um professor for responsável por um único aluno e esse aluno for removido, perde-se a informação relativa ao professor
 - Anomalia de actualização
 - Se um determinado aluno alterar a sua inscrição de um ano para o outro, é necessário garantir que é também alterado o professor responsável por esse aluno, para um dos responsáveis por esse ano

Estes inconvenientes serão resolvidos na BCNF

Forma Normal de Boyce-Codd

Um Esquema de Relação está na BCNF se todos os atributos são funcionalmente dependentes da chave e apenas da chave

- Sempre que $X \rightarrow A$, com $A \notin X$
 - X é superchave
- Entre a 3NF e a BCNF existe apenas uma ligeira diferença
- A 3NF permitia que atributos primos fossem funcionalmente dependentes de atributos não primos, não sendo isso possível na BCNF
- Considerando o exemplo anterior
 - Responsavel(numAluno, anoEscolar, professorResponsavel)
 - $F=\{(numAluno,anoEscolar) \rightarrow professorResponsavel,$
 $professorResponsavel \rightarrow anoEscolar\}$
- Verifica-se que o Esquema de Relação não está na BCNF !

Forma Normal de Boyce-Cood (cont.)

- Mais uma vez, é necessário decompor os Esquemas de Relação para passar para a BCNF
- Neste caso, o Esquema **Responsavel** será decomposto:



- Para o primeiro esquema entram os atributos da DF que está a violar a BCNF
- No outro esquema ficam os restantes atributos, excepto a parte direita da DF utilizada acima

Forma Normal de Boyce-Cood (cont.)

- Esquema de Relação Responsavel:
 - Professor(professorResponsavel,anoEscolar)
 - Onde a chave candidata/chave primária é professorResponsavel
- Esquema de Relação Disciplina
 - Disciplina(numAluno,professorResponsavel)
 - Onde a chave candidata/chave primária é composta por numAluno,professorResponsavel

Disciplina	
professorResponsavel	numAluno
Joaquim	111111
João	222222
Luis	111111
Hugo	333333
João	333333
Manuel	333333

Professor	
professorResponsavel	anoEscolar
Joaquim	1
João	3
Luis	2
Hugo	2
Manuel	4

Durante a passagem para a BCNF perdeu-se a DF
(numAluno,anoEscolar) → professorResponsável

Algoritmo de passagem para BCNF

- **Entrada:** Um Esquema de Relação $R(A_1, \dots, A_n)$ e um conjunto de DF, F
- **Saída:** Decomposição $D = \{R_1, \dots, R_N\}$ na BCNF
 - É *lossless join*
 - No entanto pode não preservar DF
- Os passos do algoritmo são:
 1. Considerar D igual a R
 2. Para cada $X \rightarrow Y$, com $Y \not\subset X$ e X não é superchave
 - Decompor D em Esquemas de Relação $R_1(XY)$ e $R_2(R-Y)$
 - As Chaves Candidatas de R_2 são todas Chaves Candidatas de R excepto aquelas que incluem o atributo Y
 3. Aplicar recursivamente este passo
 - $D = D \cup \{\text{aplicação recursiva do passo 1 a } D\}$

Algoritmo de passagem para BCNF - Exemplo

- Considere a seguinte descrição de um problema:

“Um hospital está funcionalmente dividido em serviços correspondentes às especialidades médicas vulgares (Cirurgia, Cardiologia, etc.). Em cada serviço existem vários médicos especialistas naquela área, atendendo os pacientes que se apresentam na consulta. No sentido de prestar um melhor atendimento aos seus pacientes, neste hospital existe uma regra de funcionamento em que cada paciente, num dado serviço, é sempre atendido pelo mesmo médico”

- Da descrição do problema tira-se a DF (paciente,servico) → medico, uma vez que para cada paciente e para um determinado serviço, o médico é sempre o mesmo
- Mas também se tira que medico → servico, uma vez que um médico pertence a um dado serviço
- Ficando com o Esquema de Relação ServicoMedico(paciente,servico,medico)

Algoritmo de passagem para BCNF – Exemplo (cont.)

- O Esquema de Relação, embora na 3NF, apresenta anomalias
 - Só é possível registar um médico quando existir um paciente para ele
- Assim, para resolver este problema é necessário passar para a BCNF
- Aplicar o algoritmo de passagem para a BCNF
 1. $D=\{ServicoMedico(paciente,servico,medico) \}$
 2. Chaves candidatas
 - $(Paciente,servico)$, $(paciente,medico)$
 - $X \rightarrow A$, $medico \rightarrow servico$
 - $R1(medico,servico)$
 - $R2(paciente,medico)$, que tem os atributos de $ServicoMedico$ excepto o atributo A
- $D=\{R1(medico,servico),R2(paciente,medico)\}$
- D está na BCNF, mas perdeu-se a DF $(paciente,servico) \rightarrow medico$

Algumas estratégias para a Normalização

- Quando se normaliza, raramente se percorrem todas as Formas Normais
- Quando da análise do problema, o analista, frequentemente coloca implicitamente os Esquemas de Relação extraídos na 3NF, devido à sua experiência
- Pode a normalização, no entanto, ser levada a cabo de um modo sistemático
 - Partindo de um Esquema de Relação universal que engloba todos os atributos do problema
 - Com base nas DF e nos algoritmos de passagem para as Formas Normais, procede-se à Normalização dos Esquemas de Relação
- Mais comumente usada é a passagem para a Forma Normal de um conjunto de Esquemas de Relação, resultantes de uma análise prévia do problema, descrita numa linguagem de modelação de alto nível (EA) e de onde se parte já de um conjunto de Esquemas de Relação, tirados directamente da descrição do problema

Normalização – Alguns Problemas

- A Normalização surge como resolução dos problemas causados pela redundância de informação
- No entanto, a redução dessa redundância passa pela **criação de cada vez mais Esquemas de Relação**. Para se conseguir chegar à informação inicial é necessário cruzar informação que se encontra dispersa pelas Relações criadas. Esse cruzamento é feito à custa de juncções entre as Relações, traduzindo-se isso numa **degradação do desempenho** do sistema
- O objectivo é produzir um esquema de dados flexível, coerente, suportando facilmente alterações, em que os dados são armazenados tendo em conta as restrições próprias do problema. **Há então um meio termo entre Normalização vs Desempenho**
- Por essa razão, levar a Normalização até às ultimas Formas Normais pode ser contraproducente. Normalmente opta-se por ficar pela **3NF** ou BCNF

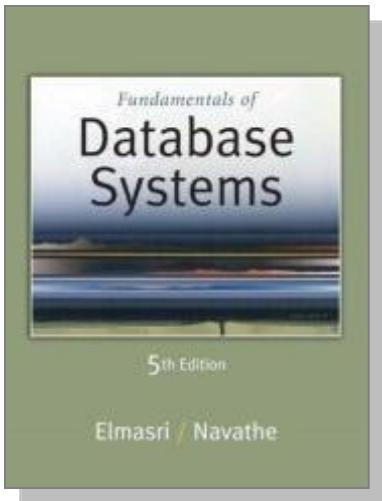
Desnormalização

- Por vezes, depois dos Esquemas de Relação estarem normalizados, não se consegue pontualmente, obter os níveis de desempenho desejados
- Neste casos é necessário proceder à desnormalização dos Esquemas de Relação
- Essa desnormalização é feita pontualmente, em locais bem específicos, com consciênciade que a redundância inserida irá traduzir-se em desempenho mas que deverá ser manipulada com cuidado
- Essa desnormalização será feita depois de conhecidos os padrões de acesso aos dados, nomeadamente nas interrogações mais frequentes, pois serão essas que provavelmente estão a degradar o desempenho global do sistema
- O esquema desnormalizado, assim como a sua justificação, devem constar de forma clara na documentação do sistema.

Bibliografia

Folhas da unidade curricular de Introdução aos Sistemas de Informação

Prof. Walter Vieira, ISEL;



Fundamentals of Database System (5th Edition)

R. Elmasri, Shamkant B. Navathe

Addison Wesley, 2003