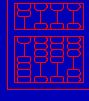




Normalização de BDs relacionais



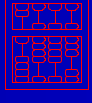
Motivação

Esses conceitos teóricos têm objetivos práticos:

- projetar BDs possivelmente sem inconsistências,
- com menos redundância de informação,
- determinar certos tipos de restrições sobre atributos de uma relação,
- determinar as possíveis chaves de uma relação.



Normalização de BDs relacionais



Exemplo de BD relacional *não normalizada*:

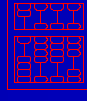
Fornecedores(numf, nome, end, tel, contato, CNPJ, num-mat, prazo, preço)

tem uma linha para cada possível material que o fornecedor fornece, o que leva a problemas:

- redundância: nome, endereço e CNPJ repetidos inúmeras vezes;
- atualização: endereço do fornecedor muda e alguma linha onde ele aparece não é atualizada;



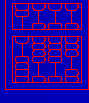
Normalização de BDs relacionais



- inserção de fornecedor sem materiais homologados: valores *nulos* teriam que ser inseridos; quando materiais forem homologados, a linha com esses nulos poderia ser esquecida na tabela;
- remoção sucessiva de materiais fornecidos: pode levar à perda completa dos dados do fornecedor.



Normalização de BDs relacionais



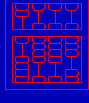
Problema básico: atributos de CEs distintos colocados na mesma tabela: Fornecimentos de materiais com atributos de Fornecedores.

Solução: eliminar informação redundante.

Custo: proliferação de tabelas (com menos colunas) e processamento para gerar relatórios com dados de diversas tabelas.



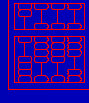
Normalização de BDs relacionais



Para que serve a teoria:

em projetos *top down*: projetar desde o início tabelas sem os problemas citados;

em projetos *bottom up*: corrigir maus projetos, *decompondo* de forma apropriada relações *mal projetadas*, isto é, *não normalizadas*.



O conceito de dependência funcional

Notação:

Dado o esquema $R(A_1, A_2, \dots, A_n)$,

letras do final do alfabeto, X, Y, Z, \dots denotarão subconjuntos de atributos de R ;

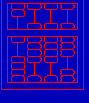
atributos compostos como $\{A, B\}$ e $\{B, A\}$ são idênticos e denotados por AB ou BA .

Definição: X determina funcionalmente Y

$(X \rightarrow Y)$ se dadas duas tuplas de R em que os valores de X são iguais, então necessariamente os valores de Y também serão iguais.



O conceito de dependência funcional



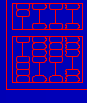
Dependência funcional é uma propriedade do projeto da base de dados, isto é, da semântica dos dados armazenados na relação.

Definição: A_i, A_j, \dots, A_k é uma *chave* de uma relação R se e somente se:

- (i) A_i, A_j, \dots, A_k determina funcionalmente todos os outros atributos de R;
- (ii) nenhum subconjunto de A_i, A_j, \dots, A_k determina funcionalmente os atributos restantes de R (minimalidade).



O conceito de dependência funcional



Definição: Uma *superchave* de R é um subconjunto dos atributos de R que contém uma chave.

Toda chave é uma superchave, mas nem toda superchave é uma chave!

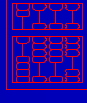
(superchaves podem violar a parte (ii) da definição de chave vista acima).

Definição: *Atributo não primo (ANP)* - não está contido em nenhuma chave de R.

Atributo primo é um atributo contido em alguma chave de R.



Regras de inferência de Armstrong



a1 Se $X \supseteq \underline{Y}$ então $X \rightarrow Y$ (regra da reflexividade)

a2 Se $X \rightarrow Y$ então para qualquer Z ,
 $XZ \rightarrow YZ$ (regra aumentativa)

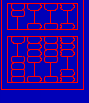
a3 Se $X \rightarrow Y$ e $Y \rightarrow Z$ então $X \rightarrow Z$
(regra da transitividade)

As regras acima são sólidas: só levam a deduções de dependências funcionais verdadeiras;

são completas: dado um conjunto de dfs F válidas para R , permitem obter todas as dfs que podem ser inferidas a partir de F e também válidas para R .



Regras de inferência de Armstrong



As seguintes regras são simples de provar:

Regra da união: se $X \rightarrow Y$ e $X \rightarrow Z$ então
 $X \rightarrow YZ$

Prova: $X \rightarrow YX$ por a2,
e, como $X \rightarrow Z$, $XY \rightarrow ZY$ por a2,
logo $X \rightarrow ZY$ por a3 e como $ZY = YZ$,
segue o resultado

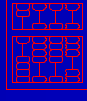
Regra da decomposição:

Se $X \rightarrow WZ$ então $X \rightarrow W$ e $X \rightarrow Z$

A prova segue diretamente de a1 ($WZ \rightarrow W$)
seguida de a3: $X \rightarrow W$. Idem para provar
 $X \rightarrow Z$.



Regras de inferência de Armstrong



As regras de união e decomposição nos permitem provar:

Lema 1: Regra geral da união/decomposição:

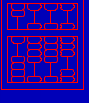
$$\begin{aligned} X &\rightarrow A_1, A_2, \dots, A_n \text{ se e somente se} \\ X &\rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n, \end{aligned}$$

o que nos permite simplificar certas provas, supondo que todas as dfs são do tipo

$X \rightarrow A_i$, onde A_i é um atributo simples de R ;
nesse caso dizemos que F está na *forma canônica*.



Regras de inferência de Armstrong



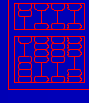
Uma df do tipo $X \rightarrow A_i$, onde X contém mais de um atributo simples, em geral não pode ser *decomposta* em dfs válidas com menos atributos à esquerda!

As regras de inferência de Armstrong são úteis em dois aspectos práticos:

1. dadas algumas dfs válidas para uma relação R , elas nos permitem descobrir outras dfs também válidas para R ;
2. elas podem nos ajudar a descobrir uma ou mais chaves de uma relação.



Regras de inferência de Armstrong

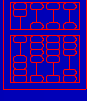


Definição: A df obtida pelo axioma $a1$ é também chamada de *dependência funcional trivial*

Chamaremos de *não trivial* toda df que não é trivial.



Formas normais (FN) de relações



São regras crescentemente restritivas para esquemas de relações *normalizadas*:

relações obedecendo à FN n devem obedecer à FN $n - 1$, mas não vice-versa.

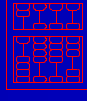
Cada forma normal *mais alta* elimina anomalias não cobertas pela forma normal *mais baixa*.

A 1ª FN é considerada como a própria definição do MR: nenhuma relação possui atributos multivalorados nem relações aninhadas.

Suporemos no que se segue, que todas as dfs dadas de uma relação são não triviais.



Segunda Forma Normal



Dependência funcional parcial:

um atributo A é dependente funcional parcial de um atributo X , se $X \rightarrow A$ e existe um subconjunto próprio Y de X (isto é, com menos atributos que X) tal que $Y \rightarrow A$

Def: uma relação R está na 2ª FN se nenhum ANP de R é dependente funcional parcial de qualquer chave de R .

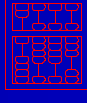
Parece complicado, mas não é! Exemplo:

$Fornitos(\underline{num_f}, num_mat, preço, endf)$, com

$F = \{num_f, num_mat \rightarrow preço, num_f \rightarrow endf\}$ não está na 2ª FN.



Terceira Forma Normal



Dadas: $X \rightarrow Y$, $Y \not\rightarrow X$, e $Y \rightarrow Z$
dizemos que “Z depende transitivamente” de X.

As aspas são intencionais: esse tipo de dependência transitiva é indesejável, porém o requisito:

$Y \not\rightarrow X$ (X não é dependente funcional de Y)

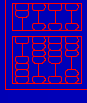
é importante: se X e Y forem chaves de R e Z for um outro atributo qualquer de R, então

$X \rightarrow Y$, $Y \rightarrow X$ e $Y \rightarrow Z$

pois, afinal, Y também é chave de R



Terceira Forma Normal



Def: R está na 3ª FN se nenhum ANP de R “depende transitivamente” de qualquer chave de R.

Exemplo:

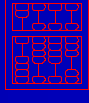
Funcionários(*numf*, *RG*, *nome*, *depto*, *nome_depto*)

numf é chave primária, *RG* é chave e *depto* → *nome_depto* é uma df válida,

R não está na 3ª FN, pois *nome_depto* é um ANP que “depende transitivamente” da chave *numf*.



Terceira Forma Normal



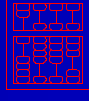
Definição equivalente de 3ª FN porém mais fácil de verificar:

Uma relação R está na 3ª FN se para toda df não trivial $X \rightarrow Y$, válida para R , então X é uma superchave de R ou Y é um atributo primo.

Teorema: toda relação que está na 3ª FN também está na 2ª FN.



Forma Normal de Boyce Codd



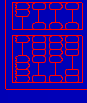
Def: uma relação R está na FNBC se para toda d não trivial, $X \rightarrow Y$, válida para R , então X é uma superchave R .

numa relação na FNBC, as únicas dependências funcionais não triviais são derivadas de chaves da relação.

Projeto *top down* de relações normalizadas segundo a FNBC: todas as dfs da relação devem ter uma chave do lado esquerdo!



3ª FN ou FNBC?



Excepcionalmente não se pode ter todas as relações na FNBC. Exemplo clássico:

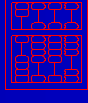
Endereços(Cidade, Rua, CEP),

com dfs: *CidadeRua* \rightarrow *CEP* e
CEP \rightarrow *Cidade*

É comum num sistema postal:

- o par *Cidade Rua* “determina” o *CEP*, mas nenhum deles determina o *CEP*,
- dado um *CEP*, a *Cidade* fica determinada, (mas *CEP* não determina *Rua*).

3ª FN ou Forma Normal de Boyce Codd?



Como (*Cidade Rua*) é chave da relação Endereços, ela não está na FNBC, pois CEP não é chave ou superchave.

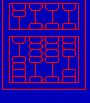
É fácil demonstrar que ela não tem outras chaves,

como Cidade não é um ANP,

a df *CEP* → *Cidade* não viola a 3ª FN.



Quarta forma normal



Origem: o mapeamento na mesma relação de dois ou mais atributos multivalorados independentes entre si gera redundâncias indesejáveis. Exemplo:

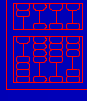
Livros(num_tombo, autores, assuntos),

onde um dado livro pode ter vários autores e tratar de vários assuntos.

Esta propriedade é refletida na chave primária da relação.



Quarta forma normal



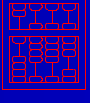
Por exemplo, poderíamos ter na tabela Livros seis linhas para o livro 31416:

num_tombo	autores	assuntos
31416	Aho	algoritmos
31416	Aho	complexidade
31416	Aho	busca
31416	Ullman	algoritmos
31416	Ullman	complexidade
31416	Ullman	busca

Livros está em FNBC, mas, existe grande redundância: um livro com k autores e m assuntos, poderia ter $k \cdot m$ linhas nessa relação e inconsistências podem ocorrer.



Quarta forma normal



Solução óbvia: decompor a tabela Livros em duas tabelas *Livros1* e *Livros2*:

Livros1 (num_tombo, autores) e,

Livros2 (num_tombo, assuntos).

Apenas cinco linhas vão aparecer nas duas tabelas para o livro 31416.

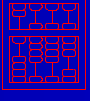
Caso geral de um livro com k autores e m assuntos:

ele apareceria $k + m$ vezes nas duas tabelas em vez de $k \cdot m$ vezes

Eliminamos as redundâncias: um autor aparece uma só vez (para um dado livro), assim como um assunto.



Quarta forma normal



A 4ª FN utiliza o conceito de *dependência (funcional) multivalorada*.

Dependências multivaloradas explicitam a independência entre os atributos e possuem regras de inferência não óbvias.

Toda dependência funcional é também uma dependência multivalorada, mas não vice-versa.

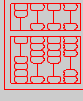
Na relação Livros acima teríamos as dependências multivaloradas:

num_tombo $\rightarrow\rightarrow$ *autores*

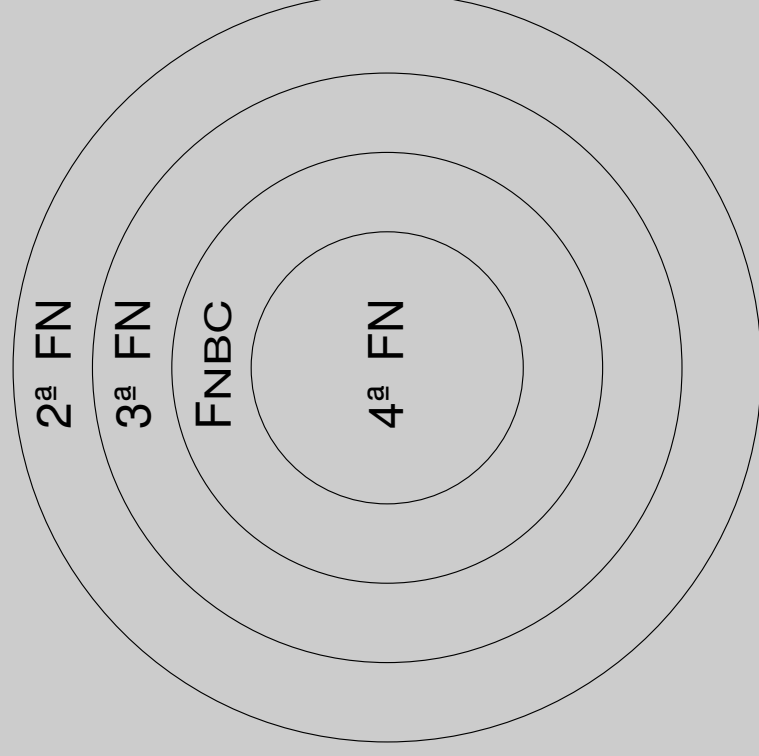
e *num_tombo* $\rightarrow\rightarrow$ *assuntos*.



Continência de formas normais

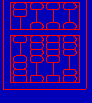


Relações na 4ª FN, também estão na FNBC; por sua vez, relações na FNBC estão em 3ª FN e estas, por sua vez, estão na 2ª FN; estas restrições sucessivas podem ser visualizadas assim:





Decomposição de relações



Origem do problema: BD convertida sem reprojeto de modelo não relacional para um SGBD relacional;

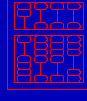
conveniente *normalizar* as relações da BD; o processo corresponde a um projeto *bottom up* ou de *análise* da BD;

implica em *decompor* relações não normalizadas em relações menores na FNBC; não trivial: tabelas estão preenchidas de dados; *perda de informação* poderia ocorrer.

Na verdade, o que pode ocorrer é a introdução de informação espúria;



Decomposição de relações



O que significa *decompor* uma relação?

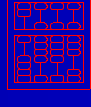
Exemplo: dada uma relação $R(A, B, C)$ com uma dependência funcional $A \rightarrow B$:

queremos decompor R em duas relações com atributos (A, B) e (B, C) ;

vamos fazer duas projeções de R : uma nas colunas A e B e outra nas colunas B e C ;

dada a instância de R a seguir:

$$R' = \begin{pmatrix} a1 & b1 & c1 \\ a2 & b1 & c2 \end{pmatrix}$$



Decomposição de relações

a decomposição leva a duas relações R_1 e R_2 :

$$R_1 = \Pi_{A,B}(R') = \begin{pmatrix} a1 & b1 \\ a2 & b1 \end{pmatrix} \text{ e,}$$
$$R_2 = \Pi_{B,C}(R') = \begin{pmatrix} b1 & c1 \\ b1 & c2 \end{pmatrix}$$

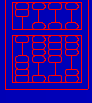
esperamos agora que R' seja recuperada pela junção $R_1 \bowtie R_2$; no entanto:

$$R_1 \bowtie R_2 = \begin{pmatrix} a1 & b1 & c1 \\ a1 & b1 & c2 \\ a2 & b1 & c1 \\ a2 & b1 & c2 \end{pmatrix}$$

nova tupla!
nova tupla!



Decomposição de relações



O resultado é inaceitável, o que nos leva ao conceito de decomposições *não aditivas* ou *sem perda* (em inglês *lossless*):

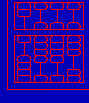
elas não adicionam novas tuplas quando se faz a junção das relações decompostas.

Uma decomposição válida não deve jogar fora atributos da relação original e, em geral terá atributos comuns nas relações decompostas;

ela deve também garantir que as dfs originais sejam *preservadas* isto é, possam todas ser inferidas a partir das dfs presentes nas relações decompostas.



Decomposição de relações

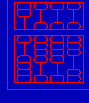


A teoria é relativamente complexa [Ullman88]:
apresentaremos um resultado matemático que
pode ser usado de forma simples na prática:
o teorema a seguir diz que uma decomposição
aceitável é sempre possível:

Teorema 3: “Toda relação R obedecendo a um
conjunto de dfs F e que não esteja na 3ª FN,
possui uma decomposição em relações
obedecendo à 3ª FN, não aditiva e que
preserva dfs.”



Decomposição de relações

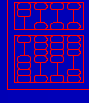


Algoritmo que resulta do Teorema 3:

1. *Obtenha uma cobertura mínima F_{min} de F .*
2. *Se F_{min} contém uma df onde aparecem todos os atributos de R , então R já está em 3ª FN (fim do algoritmo).*
3. *Crie uma decomposição de R contendo uma relação para cada df $X \rightarrow A$ de F_{min} contendo os atributos X e A (porém, se $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n$, simplifique, criando uma única relação S com esquema $S(X, A_1, A_2, \dots, A_n)$.*



Decomposição de relações



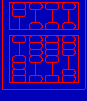
4. Se a decomposição obtida não contém nenhuma relação com uma chave de R , então acrescenta-se uma relação contendo uma chave de R .

Na maioria dos casos, as relações obtidas por este algoritmo também estarão na FNBC.

Vejamos a seguir um exemplo:



Decomposição de relações



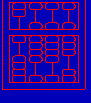
Considere o esquema a seguir de uma BD de Livros, onde um livro pode ter diversos autores:

L(num_tombo, título, autor, anoed, editora, end_editora, pres_editora, end_pres_editora)

F = {num_tombo → título, num_tombo → anoed, num_tombo → editora, editora → end_editora, editora → pres_editora, pres_editora → end_pres_editora}



Decomposição de relações



num_tombo → todos os atributos de L exceto autor;

então o par *num_tombo*, *autor* é chave de L;
fácil provar que F é um conjunto minimal de dfs;

logo, podemos usar F para decompor L em:

L1(num_tombo, título, anoed, editora),

L2(editora, end_editora, pres_editora),

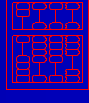
L3(pres_editora, end_pres_editora).

Como a chave de L não aparece na decomposição, acrescentamos a relação:

L4(num_tombo, autor)



Considerações práticas



Normalização até o fim não é uma regra invariável.

Casos em que não é apropriada:

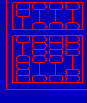
- se a tabela Funcionários possui n atributos, poderíamos decompô-la em $n-1$ tabelas, cada uma com dois atributos: a chave primária e um outro atributo;

o projeto resultante está em FNBC e na 4ª FN, mas é um péssimo projeto:

possui $n-1$ tabelas desnecessárias, tornando muito ineficientes consultas e geração de relatórios envolvendo vários atributos de um funcionário.



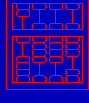
Considerações práticas



- em certos casos pode-se ter uma relação violando a 3ª FN; a tabela Funcionários serve para ilustrar o ponto:
o endereço do funcionário inclui rua, número, bairro, cidade e CEP;
CEP é determinado pelos atributos anteriores: logo, não deveria ser atributo da relação;
porém, qualquer consulta / relatório envolvendo o endereço geralmente requer o CEP; se a tabela estivesse normalizada seria preciso uma junção cara com uma tabela (muito grande) de CEPs!



Considerações práticas finais

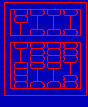


Cuidados no projeto *top down*:

- especifique as relações *mestre da sua BD*:
para cada uma delas verifique se a chave primária determina funcionalmente todos os outros atributos,
e que há apenas dependências funcionais envolvendo chaves;
- certifique-se de que todos os atributos de uma relação são realmente dela,
procurando dfs entre atributos que não são chaves;



Considerações práticas finais



- mapeie os relacionamentos N : N em relações, especificando sua chave primária e seus atributos;
- quando todas as suas relações estiverem na FNBC (excepcionalmente na 3ª FN), verifique se não há violação da 4ª FN: se houver, decomponha as relações infratoras de forma apropriada, deixando apenas um atributo multivalorado em cada relação.

O projeto *top down* é obviamente preferível ao projeto *bottom up*.