Teste de Preservação de Dependências

Para verificar se $\alpha \to \beta$ é preservada na decomposição R em $R_1, R_2, \ldots, R_n$ aplica-se o seguinte teste:

res := α
enquanto (houver alterações em res) **faz**
  para cada R_i na decomposição **faz**
    t := $(\texttt{res} \cap R_i)^+ \cap R_i$
    res := res $\cup$ t
  fimpara
fimenquanto

Se res contém todos os atributos em β, então $\alpha \to \beta$ é preservada.
Aplica-se este teste a todas as dependências de F, para verificar se a decomposição preserva as dependências.
Ao contrário do cálculo de F^+ ou de $(F_1 \cup F_2 \cup \cdots \cup F_n)^+$, que têm ambos complexidade exponencial, este procedimento tem complexidade polinomial.

Forma Normal de Boyce-Codd

Definição (Forma Normal de Boyce-Codd)

Um esquema R diz-se na Forma Normal de Boyce-Codd, BCNF, relativamente a um conjunto de dependências F, sse para toda a dependência em F^+ da forma $\alpha \to \beta$, onde $\alpha \subseteq R$ e $\beta \subseteq R$, pelo menos uma das seguintes condições é verdadeira:

- *$\alpha \to \beta$ é trivial, isto é, $\beta \subseteq \alpha$.*
- *α é super-chave de R, isto é, $\alpha \to R$.*

Evita redundâncias

Verificação de dependências funcionais definidas sobre atributos de R, limita-se à verificação de chaves.

BCNF — Exemplo/Exercício

- $R = (A, B, C)$, $F = \{A \to B, B \to C\}$.
- Chave, $\{A\}$.
- R não está em BCNF.
- Decomposição em $R_1 = (A, B)$, $R_2 = (B, C)$.
  - R_1 e R_2 estão na BCNF.
  - Decomposição sem perdas.
  - Preserva as dependências.

Teste para BCNF

- Para determinar se uma dependência não trivial $\alpha \to \beta \in F^+$ é causa de violação de BCNF:
  1. calcular α^+ (fecho de atributos em α);
  2. Verificar se inclui todos os atributos de R, i.e. é super-chave de R. Se incluir, então não viola a condição de BCNF; caso contrário há violação da BCNF, a relação não está na BCNF.
- Teste simplificado: Em vez de verificar para todas as dependências de F^+, verificar apenas para as dependências numa cobertura canónica.

  Se nenhuma das dependências da cobertura canónica for contra a BCNF, então nenhuma das dependências de F^+ vai contra a BCNF.
- É no entanto necessário verificar se as dependências são preservadas.

  Por exemplo: Seja $R(A, B, C, D)$, com $F = \{A \to B, B \to C\}$.
  - Decomposição de R em $R_1(A, B)$ e $R_2(A, C, D)$;
  - Nenhuma das dependências em F contém só atributos de (A, C, D), por isso, podemos (erradamente) pensar que R_2 satisfaz BCNF.
  - Mas a dependência $A \to C \in F^+$ mostra que R_2 não está na BCNF.

Decomposição BCNF

Dado o esquema

`(idCliente,nEmpréstimo,quantia)`

Verifica-se a dependência funcional `nEmpréstimo` $\rightarrow$ `quantia`.
No entanto `nEmpréstimo` não é uma super-chave.

Seja R um esquema que não está na BCNF, então existe pelo menos uma dependência funcional não trivial $\alpha \rightarrow \beta$ tal que α não é uma super-chave para R.
Substitui-se então R por dois esquemas:

- $(\alpha \cup \beta)$
- $(R - (\beta - \alpha))$

Ter-se-ia (para o exemplo acima)

- $(\alpha \cup \beta)$ = `(nEmpréstimo,quantia)`
- $(R - (\beta - \alpha))$ = `(idCliente,nEmpréstimo)`

Algoritmo para Decomposição BCNF

Algoritmo genérico para o cálculo de uma decomposição BCNF.

res := $\{R\}$;
fim := false;
calcular F^+;
enquanto $\neg$fim **faz**
  se há um esquema R_i em res que não está na BCNF **então**
    Seja $\alpha \rightarrow \beta$ uma dependência (não trivial) sobre R_i
      tal que $\alpha \rightarrow R_i \notin F^+$ e $\alpha \cap \beta = \emptyset$;
    res := $(res - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$;
  senão
    fim := true;
  fimse
fimenquanto

Nota: no fim do algoritmo cada R_i está na BCNF, e a decomposição é sem perdas.

Exemplo de Decomposição BCNF

- Esquema:
  Amigos = (nome, telefone, codPostal, localidade)
  $F = \{$nome $\rightarrow$ telefone, codPostal; codPostal $\rightarrow$ localidade$\}$
- Decomposição:
  res = $\{$Amigos$\}$
  codPostal $\rightarrow$ localidade $\in F^+$ e codPostal não é chave
  res = $\{$(nome, telefone, codPostal),
  (codPostal, localidade)$\}$
- Pode-se verificar que os esquemas resultantes estão na BCNF.

Outro exemplo de Decomposição BCNF

- Esquema:
  R = (balcao, localidade, activos, cliente, numEmprestimo, valor)
  $F = \{$balcao $\rightarrow$ activos, localidade; numEmprestimo $\rightarrow$ valor, balcao $\}$
  Chave = $\{$numEmprestimo, cliente$\}$
- Decomposição:
  res = $\{R\}$
  balcao $\rightarrow$ activos, localidade $\in F^+$ e balcao não é chave em R
  R_1 = (balcao, activos, localidade)
  R_2 = (balcao, cliente, numEmprestimo, valor)
  res = $\{R_1, R_2\}$
  numEmprestimo $\rightarrow$ valor, balcao $\in F^+$ e nunEmprestimo não é chave em R_2
  R_3 = (numEmprestimo, valor, balcao)
  R_4 = (cliente, numEmprestimo)
  res = $\{R_1, R_3, R_4\}$
- Já está na BCNF.

Teste de Decomposição BCNF

Para verificar se R_i numa decomposição de R está na BCNF:

- Ou se testa R_i relativamente à restrição de F a R_i, isto é todas as dependências em F^+ que só contêm atributos de R_i;
- Ou se usa o conjunto original de dependências sobre R mas com o teste seguinte:

  Para todo $\alpha \subseteq R_i$, verificar se α^+ não contém nenhum atributo de $R_i - \alpha$, ou então α^+ contém todos os atributos de R_i.

  - Se a condição for violada por alguma $\alpha \rightarrow \beta$ em F, a dependência $\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$ é verdadeira em R_i, e R_i viola BCNF.
  - Usa-se essa dependência para decompor R_i

BCNF e preservação de dependências

Nem sempre é possível obter uma decomposição BCNF que preserve as dependências.

- $R = (J, K, L)$, $F = \{JK \rightarrow L, L \rightarrow K\}$, existem duas chaves candidatas JK e JL.
- R não está na BCNF.
- Nenhuma decomposição de R preserva $JK \rightarrow L$.

Exemplo

Considere o esquema `Gestores = (balcao, cliente, gestorConta)`, com as dependências:

1. `gestorConta` $\longrightarrow$ `balcao`
2. `cliente, balcao` $\longrightarrow$ `gestorConta`

Não está na BCNF (a dependência 1 não é trivial e `gestorConta` não é super-chave, não implica `cliente`).

Decompor em `GestoresBal = (balcao, gestorConta)` e `Clientes = (cliente, gestorConta)`

Agora já está na BCNF.

Mas não se preservam as dependências!!! A dependência 2 não se pode verificar numa só relação.

Motivação para a 3ª Forma Normal

Há situações em que:

- a BCNF não preserva as dependências;
- a eficiência na verificação de integridade aquando de alterações é importante

Solução: definir uma forma normal mais fraca – 3ª Forma Normal:

- admite alguma redundância (o que pode trazer problemas, como veremos à frente);
- as dependências podem ser verificadas sem recorrer a junções;
- é sempre possível fazer uma decomposição sem perdas para a 3NF, que preserva as dependências.

Exemplo Motivador

O esquema `Gestores = (balcao, cliente, gestorConta)`, com as dependências:

1. `gestorConta` $\longrightarrow$ `balcao`
2. `cliente, balcao` $\longrightarrow$ `gestorConta`

não está na BCNF por causa da primeira dependência.

A única chave candidata de Gestores é {`cliente, balcao`}. Mas: Ao decompor Gestores com base na 1ª dependência, `balcao` vai ficar numa relação diferente daquela onde fica `cliente`.
Logo deixa de ser possível verificar a chave candidata de Gestores numa só relação!

Solução: Para se continuar a poder verificar a chave candidata da relação original, não decompor um esquema com base numa dependência que à direita contenha apenas alguns dos atributos duma chave candidata.

3ª Forma Normal

Definição (3ª Forma Normal)
Um esquema R está na 3ª Forma Normal, 3NF, sse para toda $\alpha \rightarrow \beta$ pertencente a F^+, pelo menos uma das seguintes condições é verdadeira:

- *$\alpha \rightarrow \beta$ é trivial, isto é, $\beta \subseteq \alpha$.*
- *α é super-chave de R, isto é, $\alpha \rightarrow R$.*
- *Todo atributo A em $(\beta - \alpha)$ está contido numa chave candidata de R, não necessariamente a mesma.*

Se R está na BCNF então está também na 3NF.

A 3ª condição relaxa a BCNF para garantir a preservação de dependências

3ª Forma Normal - Exemplo

$$R = (J, K, L)$$
$$F = \{JK \rightarrow L, L \rightarrow K\}$$

Duas chaves candidatas: JK e JL.
R está na 3NF.

- $JK \rightarrow L$, JK é super-chave
- $L \rightarrow K$, K está contido numa chave candidata

A decomposição BCNF dá origem a $R_1 = (JL)$ e $R_2 = (LK)$.
Para esta decomposição o testar de $JK \rightarrow L$ obriga a uma junção.
Pode haver redundância em R

J	L	K
1	a	x
2	a	x
3	a	x
4	b	y

dado a dependência $L \rightarrow K$, os dados da coluna K são, sempre que haja repetição, redundantes.

Teste para 3NF

- Optimização: Basta verificar para uma cobertura canónica de F (não é necessário verificar para toda a dependência em F^+).
- Usar o fecho de atributos para verificar, em toda a dependência $\alpha \rightarrow \beta$, se α é super-chave.
- Se α não for super-chave, há que verificar se todo a atributo em β pertence a alguma chave candidata de R

- Este teste é bastante ineficiente, pois envolve o cálculo de chaves candidatas.
- Pode demonstrar-se que verificar se um conjunto de esquemas está na 3NF é um problema intratável ("NP-hard").
- Existe uma decomposição para a 3NF (descrita mais à frente) com complexidade polinomial.

Algoritmo de Decomposição para 3NF

Seja F_c uma cobertura canónica de F;
i := 0;
para todo $\alpha \to \beta \in F_c$ **faz**
  se nenhum dos esquemas $R_j, 1 \leq j \leq i$ contém $\alpha\beta$ **então**
    i := i + 1;
    R_i := $\alpha\beta$;
  fimse
fimpara
se nenhum dos esquemas $R_j, 1 \leq j \leq i$ contém uma chave candidata de R
então
  i := i + 1;
  R_i := uma (qualquer) chave candidata de R;
fimse

A decomposição resultante é $(R_1, R_2, \ldots, R_i)$.

O algoritmo descrito garante que:

- Todo o esquema R_i está na 3NF;
- A decomposição preserva as dependências e é sem perdas.

Exemplo

- Considere o esquema:
  `InfoGestores = (balcao, cliente, gestorConta, gabinete)`
- As dependências definidas sobre este esquema são:
  $$\begin{aligned} \texttt{gestorConta} &\to \texttt{balcao, gabinete} \\ \texttt{cliente, balcao} &\to \texttt{gestorConta} \end{aligned}$$
- A chave candidata é: {`cliente, balcao`}
- O ciclo **para todo** do algoritmo, leva à introdução dos seguintes esquemas na decomposição:
  `GestoresGab = (gestorConta, balcao, gabinete)`
  `Gestores = (cliente, balcao, gestorConta)`
- Como `Gestores` contém uma chave candidata de `InfoGestores`, o processo de decomposição termina.

BCNF versus 3NF

- É sempre possível decompor um esquema, num conjunto de esquemas na 3NF em que:
  - a decomposição é sem perdas;
  - as dependências são preservadas.

  Mas pode haver alguma redundância!!!
- É sempre possível decompor um esquema, num conjunto de esquemas na BCNF em que:
  - a decomposição é sem perdas;
  - não há redundância.

  Mas nem sempre se podem preservar as dependências!!!

BCNF versus 3NF (cont.)

Exemplo de problemas causados pela redundância na 3NF

- Seja R a seguinte relação:

$$\begin{aligned} R &= (J, K, L) \\ F &= \{JK \to, L \to K\} \end{aligned}$$

com a seguinte instância

J	L	K
j_1	l_1	k_1
j_2	l_1	k_1
j_3	l_1	k_1
`null`	l_2	k_2

- A relação R, que está na 3NF mas não na BCNF, tem problemas de:
  - Repetição de informação (relação $L \to K$);
  - Necessita usar valores `null`, por exemplo para representar a dependência entre l_2, k_2 sem que haja valores correspondente em J).