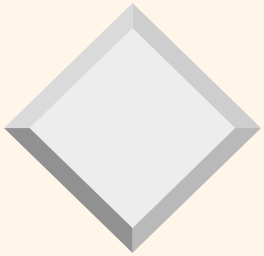
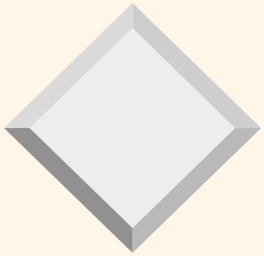


Refinamento de Esquemas e Formas Normais

Capítulo 19

Roteiro

- Problemas causados por Redundâncias.
- Refinamento de esquemas.
- O que são Dependências Funcionais.
- O que são Formas Normais e suas aplicações.
- Decomposições.
- Normalização.



Os Males da Redundância

- *Redundância* é a raiz de vários males associados com esquemas relacionais.
- Armazenamento: Mesma informação em vários lugares.
- Problemas: Atualização, Inserção e Remoção.

Exemplo de Redundância

- Considere a relação obtida de Hourly_Emps:
  - Hourly_Emps (*ssn*, *name*, *lot*, *rating*, *hrly_wages*, *hrs_worked*)

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

ssn é a chave : S → SNLRWH

rating determina *hrly_wages*: R → W

Exemplo de Redundância (Cont.)

- Considere a relação obtida de Hourly_Emps:
  - Hourly_Emps (*ssn, name, lot, rating, hrly_wages, hrs_worked*)
- *Notação*: Iremos denotar esta relação pela lista de atributos: SNLRWH
  - Na realidade, isto é um *conjunto* de atributos {S,N,L,R,W,H}.
  - Eventualmente, o nome da relação será usado para se referir a todos os atributos (e.g., Hourly_Emps para SNLRWH)
- Algumas Dependências Funcionais em Hourly_Emps:
  - *ssn* é a chave : S → SNLRWH
  - *rating* determina *hrly_wages*: R → W

Exemplo de Redundância (Cont.)

- Problemas de R → W :
  - *Anomalia de Atualização* : Podemos mudar W apenas na 1a tupla de SNLRWH ?
  - *Anomalia de Inserção*: E se nós quisermos inserir um empregado e não soubermos o valor da hora para o seu nível ?
  - *Anomalia de Remoção*: Se removermos todos os empregados de nível 5 perderemos a informação sobre o salário desse nível (R) ?

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Refinamento de Esquemas

- Restrições de integridade, em particular *dependência funcional* (R→W) , podem ser usadas para identificar esquemas com tais problemas e sugerir refinamentos.

- Principal técnica de refinamento : *decomposição*
  - ABCD vira AB e BCD, ou ACD e ABD.

- A decomposição deve ser cautelosa:
  - Existe uma razão para decompor a relação?
  - Que problemas a decomposição pode causar?

Exemplo de Decomposição

Hourly_Emps (*ssn, name, lot, rating, hrly_wages, hrs_worked*)

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Hourly_Emps2 (*ssn, name, lot, rating, hrs_worked*)

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

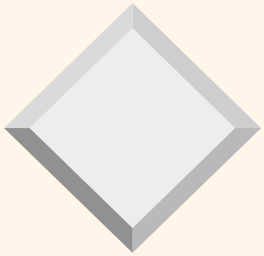
Wages (*rating, hrly_wages*)

R	W
8	10
5	7

Dependência Funcional (DF)

- Uma dependência funcional é um tipo de restrição que generaliza o conceito de chave.
- A dependência funcional $X \rightarrow Y$ vale sobre a relação R se, para cada instância possível *r* de R:
  - $t1 \in r$, $t2 \in r$, $\pi_X(t1) = \pi_X(t2)$ implica $\pi_Y(t1) = \pi_Y(t2)$
  - i.e., dadas 2 tuplas em *r*, se os valores de X são iguais , então os de Y também devem ser

    (X e Y são conjuntos de atributos).



Dependência Funcional (Cont.)

- Uma DF é válida sobre *todas* as relações possíveis.
  - Deve ser identificada baseada na semântica da aplicação.
  - Dada uma instância válida *r1* de R, pode-se verificar se uma DF *f* é violada, mas não se *f* é verdadeira em R!

- Se K é uma chave candidata de R
  - implica que $K \rightarrow R$
  - Entretanto, $K \rightarrow R$ não requer que K seja *mínima*!

Decorrências de DFs

- Dadas DFs, usualmente inferimos novas DFs:
  - $ssn \rightarrow did$, $did \rightarrow lot$ implica $ssn \rightarrow lot$
- Clausura de F, denotada por F^+, é o conjunto de todas DFs que são implicadas por *F*.
- Axiomas de Armstrong (X, Y, Z são os conjuntos de atributos):
  - *Reflexibilidade*: Se $X \supseteq Y$, então $X \rightarrow Y$
  - *Expansibilidade*: Se $X \rightarrow Y$, então $XZ \rightarrow YZ$ p/ qq Z
  - *Transitividade*: Se $X \rightarrow Y$ e $Y \rightarrow Z$, então $X \rightarrow Z$
- Estas *regras de* inferência são *seguras* e *completas*.

Geram apenas DFs corretas e permitem criar clausura de F.

Decorrências de DFs (Cont.)

- 2 regras adicionais (conseqüência do AA):
  - *União*: Se X → Y e X → Z, então X → YZ
  - *Decomposição*: Se X → YZ, então X → Y e X → Z
- Exemplo: Contracts(*cid*,*sid*,*jid*,*did*,*pid*,*qty*,*value*), e:
  - C é a chave: C → CSJDPQV
  - Projeto compra cada parte com um contrato: JP → C
  - Dept compra no max. uma parte de cada forne.: SD → P
- JP → C, C → CSJDPQV implica JP → CSJDPQV
- SD → P implica SDJ → JP
- SDJ → JP, JP → CSJDPQV implica SDJ → CSJDPQV

Decorrências de DFs (Cont.)

C → contract (contrato)
S → supplier (fornecedor)
J → project (projeto)
D → departament (departamento)

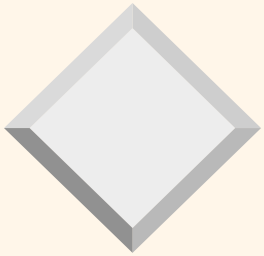
P → part (parte/produto)
Q → quantity (quantidade)
V → value (valor)

- Exemplo: Contracts(*cid*,*sid*,*jid*,*did*,*pid*,*qty*,*value*), e:
  - C é a chave: C → CSJDPQV
  - Projeto compra cada parte com um contrato: JP → C
  - Dept compra no max. uma parte de cada forne.: SD → P

- JP→C, C→CSJDPQV implica JP→CSJDPQV
- SD→P implica SDJ→JP
- SDJ→JP, JP→CSJDPQV implica SDJ→CSJDPQV

Decorrências de DFs (Cont.)

- 2 regras adicionais (conseqüência do AA):
  - *União*: Se X → Y e X → Z, então X → YZ
  - *Decomposição*: Se X → YZ, então X → Y e X → Z
- Exemplo: Contracts(*cid*,*sid*,*jid*,*did*,*pid*,*qty*,*value*), e:
  - C é a chave: C → CSJDPQV
  - Projeto compra cada parte com um contrato: JP → C
  - Dept compra no max. uma parte de cada forne.: SD → P
- JP → C, C → CSJDPQV implica JP → CSJDPQV
- SD → P implica SDJ → JP
- SDJ → JP, JP → CSJDPQV implica SDJ → CSJDPQV



Formas Normais

- Retornando à questão do refinamento de esquema, a primeira dúvida é: O refinamento é necessário ?
- Formas Normais: *guias* p/ o refinamento.
- Se a relação está em uma certa *forma normal* (BCNF, 3NF etc.), é sabido que certos tipos de problemas são evitados/minimizados. Isto pode ser usado para decidir se a decomposição será útil.
- Regras de DFs na detecção de redundância:
  - Considere a relação R com 3 atributos, ABC.
    - Sem DFs : Não existe redundância.
    - Dado A $\rightarrow$ B: Varias tuplas podem ter o mesmo valor em A e então terão o mesmo valor em B!

Formas Normais (Cont.)

- As formas normais baseadas em DFs são:
  - Primeira Forma Normal (1NF)
  - Segunda Forma Normal (2NF)
  - Terceira Forma Normal (3NF)
  - Forma Normal Boyce-Codd (BCNF)

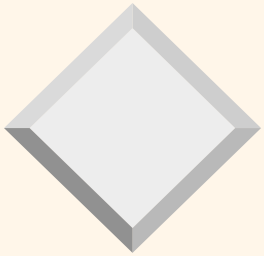
Forma Normal Boyce-Codd (BCNF)

- R com DFs *F* está na BCNF se, para todo $X \rightarrow A$ em F^+
  - $A \in X$ (DF *trivial*), ou
  - X contém uma chave para R (superchave).
- Em outras palavras, R está na BCNF se as únicas DFs não-triviais sobre R são restrições de chave.
  - Nenhuma dep. em R pode ser predita usando somente DFs.
  - Se nos são mostradas 2 tuplas com o valor X, nós não podemos inferir o valor de A de uma tupla usando a outra.
  - Se uma relação exemplo está na BCNF, as 2 tuplas devem ser idênticas (desde que X seja a chave).

X	Y	A
x	y1	a
x	y2	?

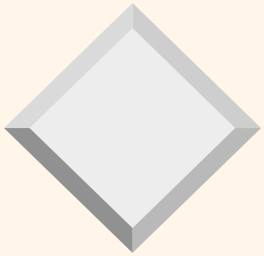
Terceira Forma Normal (3NF)

- R com DFs *F* está na 3NF se, para todo $X \rightarrow A$ em F^+
  - $A \in X$ (DF *trivial*), ou
  - X é uma superchave, ou
  - A é parte de uma chave em R.
- Chave em R é um conjunto de atributos mínimo que determina todos os outros atributos.
- Se R está na BCNF, obviamente está na 3NF.
- Se R está na 3NF, alguma redundância é possível. É um compromisso, usado quando BCNF não é atingível (i.e., sem decomp. "boa", ou considerações de desempenho).
  - *Decomposição 'Junção sem perda', preservando dependência de R em uma coleção de relações na 3NF é sempre possível.*



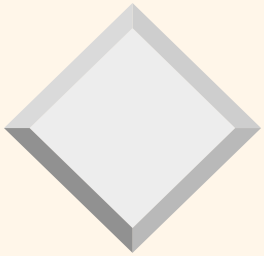
O que a 3NF Atinge?

- Se a 3NF é violada por (X→A), as causas são:
  - X é um subconjunto de alguma chave K *(dependência parcial)*.
    - Os pares (X, A) armazenados são redundantes.
  - X não é um subconjunto de alguma chave *(dependência transitiva)*.
    - Existe uma cadeia de DFs K → X → A, que significa que nós não conseguimos associar um valor de X com um valor de K a menos que também associemos um valor de A com um valor de X.
- Mas: mesmo usando a 3NF, estes problemas podem acontecer.
  - e.g., Reservas SBDC, S → C, C → S está na 3NF, mas para cada reserva do marujo S, o mesmo par (S, C) é armazenado.
- Então, 3NF é um compromisso relativo à BCNF.



Formas Normais

- Retornando à questão do refinamento de esquema, a primeira dúvida é: O refinamento é necessário ?
- Formas Normais: *guias* para o refinamento.
- Se a relação está em uma certa *forma normal* (BCNF, 3NF etc.), é sabido que certos tipos de problemas são evitados/minimizados. Isto pode ser usado para decidir se a decomposição será útil.
- Regras de DFs na detecção de redundância:
  - Considere a relação R com 3 atributos, ABC.
    - Sem DFs : Não existe redundância.
    - Dado A $\rightarrow$ B: Varias tuplas podem ter o mesmo valor em A e então terão o mesmo valor em B!



Decomposição

- Suponha que a relação R contenha atributos *A1 ... An*. Uma *decomposição* de R consiste e substituir R por duas ou mais relações tal que:
  - Cada novo esquema contém um subconjunto dos atributos de R (e nenhum atributo que não apareça em R), e
  - Cada atributo de R aparece como um dos atributos de uma das novas relações.
- Intuitivamente, *decompor* R significa que nós iremos armazenar instâncias dos esquemas produzidos pela decomposição, ao invés de instâncias de R.

Exemplo de Decomposição

Hourly_Emps (*ssn, name, lot, rating, hrly_wages, hrs_worked*)

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Hourly_Emps2 (*ssn, name, lot, rating, hrs_worked*)

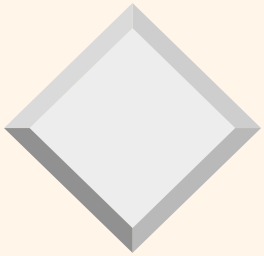
S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Wages (*rating, hrly_wages*)

R	W
8	10
5	7

Exemplo de Decomposição (Cont.)

- Decomposições devem ser usadas somente quando são necessárias:
  - SNLRWH tem DFs S→SNLRWH e R→W
  - A segunda DF causa uma violação na 3NF; valores W repetidamente se associam com valores R. A forma mais simples de resolver isto é criar uma relação RW para armazenar estas associações, e para remover W do esquema principal:
    - i.e., iremos decompor SNLRWH em SNLRH e RW
- A informação a ser armazenada consiste em tuplas SNLRWH. Se nós apenas armazenarmos as projeções das tuplas em SNLRH e RW, haverá qualquer problema potencial que devamos estar conscientes?



Problemas com Decomposições

- Há três problemas potenciais a considerar:
  1. Algumas *queries* se tornam muito caras.
     - ex., Quanto ganha o navegador Joe? (salário = W*H)
  2. Dadas as instâncias das relações decompostas, nós podemos não conseguir reconstruir a instância correspondente da relação original!
     - Por sorte, não é o caso do exemplo SNLRWH.
  3. A verificação de algumas dependências podem exigir a junção de instâncias das relações decompostas.
     - Por sorte, não é o caso do exemplo SNLRWH.
- *Compromisso*: Devemos considerar estas questões X redundância.

Decomposições 'Junção Sem Perda' (Lossless Join)

- Decomposição de R em X e Y é *'junção sem perda'* no que diz respeito a um conjunto de DFs F se, para cada instância *r* que satisfaz F:
  - $\pi_x(r) \bowtie \pi_y(r) = r$
- É sempre verdadeiro que $r \subseteq \pi_x(r) \bowtie \pi_y(r)$
  - Em geral, a outra direção não é assegurada! Se ela existe, a decomposição é 'junção sem perda'.
- A definição pode ser estendida em 3 ou mais relações justamente pelo mesmo caminho.
- *É essencial que todas as decomposições usadas para lidar com redundância sejam 'sem perda'!*

Decomp. 'Junções Sem Perda' (Cont.)

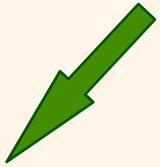
- A decomposição de R em X e Y é 'junção sem perda' em relação a F se e somente se a *clausura* de F contém:
  - $X \cap Y \rightarrow X$, ou
  - $X \cap Y \rightarrow Y$
- Em particular, a decomposição de R em UV e R - V é 'junção sem perda' se $U \rightarrow V$ é assegurado sobre R.

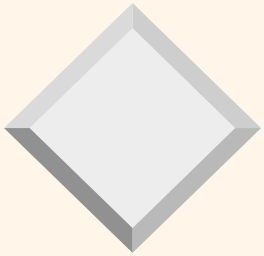
A	B	C
1	2	3
4	5	6
7	2	8

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3





Decomp. Preservando Dependência

- Considere a relação

  *Contracts(**C**ontractid, **S**upplierid, pro**J**ectid, **D**eptid, **P**artid, **Q**ty, **V**alue)*

  Contratos com os atributos CSJDPQV, C é chave, JP $\rightarrow$ C e SD $\rightarrow$ P.
  - Decomposição BCNF: CSJDQV e SDP
  - Problema: Verificar JP $\rightarrow$ C, requer uma junção!
- Decomposição preservando dependência (Intuitivo):
  - Se R é decomposto em X, Y e Z, e nós fazemos cumprir as DFs que são asseguradas em X, em Y e em Z, então todas as DFs que deveriam ser asseguradas em R também devem estar asseguradas.

Decomp. Preservando Dependência

- Considere a relação

  Contracts(Contractid, Supplierid, proJectid, Deptid, Partid, Qty, Value)

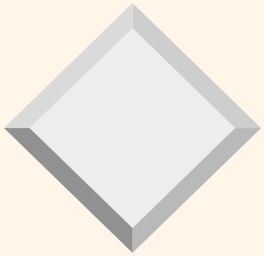
  Contratos com os atributos CSJDPQV, C é chave, JP → C e SD → P.

  - Decomposição BCNF: CSJDQV e SDP
  - Problema: Verificar JP → C requer uma junção!

C → contract (contrato)
S → supplier (fornecedor)
J → project (projeto)
D → departament (departamento)

P → part (parte/produto)
Q → quantity (quantidade)
V → value (valor)

estar asseguradas.



Decomp. Preservando Dependência

- Considere a relação

  *Contracts(**C**ontractid, **S**upplierid, pro**J**ectid, **D**eptid, **P**artid, **Q**ty, **V**alue)*

  Contratos com os atributos CSJDPQV, C é chave, JP $\rightarrow$ C e SD $\rightarrow$ P.
  - Decomposição BCNF: CSJDQV e SDP
  - Problema: Verificar JP $\rightarrow$ C requer uma junção!
- Decomposição preservando dependência (Intuitivo):
  - Se R é decomposto em X, Y e Z, e nós fazemos cumprir as DFs que são asseguradas em X, em Y e em Z, então todas as DFs que deveriam ser asseguradas em R também devem estar asseguradas.

Decomp. Preservando Dependência (Cont.)

- Decomposição de R em X e Y está *preservando dependência* se $(F_X \text{ união } F_Y)^+ = F^+$
  - i.e., se nós considerarmos somente dependências na *clausura* F^+ que podem ser verificadas em X sem considerar Y, e em Y sem considerar X, elas implicam em todas as dependências em F^+.
- Importância de considerar F^+, não F, nesta definição:
  - ABC, $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$, decomposto em AB e BC.
  - Está preservando dependência? $C \rightarrow A$ foi preservado?????
- Preservar dependência não implica em ser 'junção sem perda': ABC, $A \rightarrow B$, decomposto em AB e BC.

Decomp. Preservando Dependência (Cont.)

- Decomposição de R em X e Y está *preservando dependência* se $(F_X \text{ união } F_Y)^+ = F^+$
  - i.e., se nós considerarmos somente dependências na *clausura* F^+ que podem ser verificadas em X sem considerar Y, e em Y sem considerar X, elas implicam em todas as dependências em F^+.
- Importância de considerar F^+, não F, nesta definição:
  - ABC, $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$, decomposto em AB e BC.

Closure F^+:	
$A \rightarrow B$	$A \rightarrow C$
$B \rightarrow C$	$B \rightarrow A$
$C \rightarrow A$	$C \rightarrow B$

F_{AB} contém:
$A \rightarrow B$ e $B \rightarrow A$

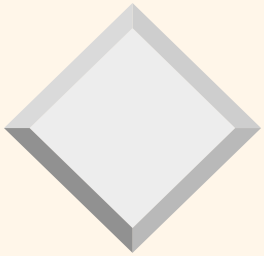
F_{BC} contém:
$B \rightarrow C$ e $C \rightarrow B$

$F_{AB} \cup F_{BC}$:
$A \rightarrow B$, $B \rightarrow A$, $B \rightarrow C$, $C \rightarrow B$

Clausura:
$C \rightarrow A$ e $A \rightarrow C$

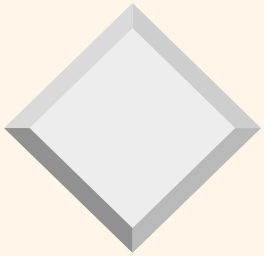
Decomp. Preservando Dependência (Cont.)

- Decomposição de R em X e Y está *preservando dependência* se $(F_X \text{ união } F_Y)^+ = F^+$
  - i.e., se nós considerarmos somente dependências no *closure* F^+ que podem ser verificadas em X sem considerar Y, e em Y sem considerar X, elas implicam em todas as dependências em F^+.
- Importância de considerar F^+, não F, nesta definição:
  - ABC, $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$, decomposto em AB e BC.
  - Está preservando dependência? $C \rightarrow A$ foi preservado ? N
- Preservar dependência não implica em ser 'junção sem perda': ABC, $A \rightarrow B$, decomposto em AB e BC.



Normalização

- Decompor relações em BCNF.
- Decompor relações em 3NF.



Decomposição em BCNF

- Considere a relação R com DFs F. Se $X \rightarrow Y$ viola a BCNF, decomponha R em R - Y e XY.
  - A aplicação repetida desta idéia nos dará uma coleção de relações que estão em BCNF; decomposições 'junção sem perda', e com garantia de término.
  - ex., CSJDPQV, chave C, $JP \rightarrow C$, $SD \rightarrow P$, $J \rightarrow S$
  - $SD \rightarrow P$, decomponha em SDP, CSJDQV.
  - $J \rightarrow S$, decomponha CSJDQV em JS e CJDQV.
- Em geral, muitas DFs podem causar violação da BCNF. A ordem na qual tratamos as DFs, pode resultar em diferentes conjuntos de relações.

Decomposição em BCNF (Cont.)

- Considere a relação R com DFs F. Se $X \rightarrow Y$ viola a BCNF, decomponha R em R - Y e XY.
  - A aplicação repetida desta idéia nos dará uma coleção de relações que estão em BCNF; decomposições 'junção sem perda', e com garantia de término.
  - ex., CSJDPQV, chave C, $JP \rightarrow C$, $SD \rightarrow P$, $J \rightarrow S$
  - $SD \rightarrow P$, decomponha em SDP, CSJDQV.

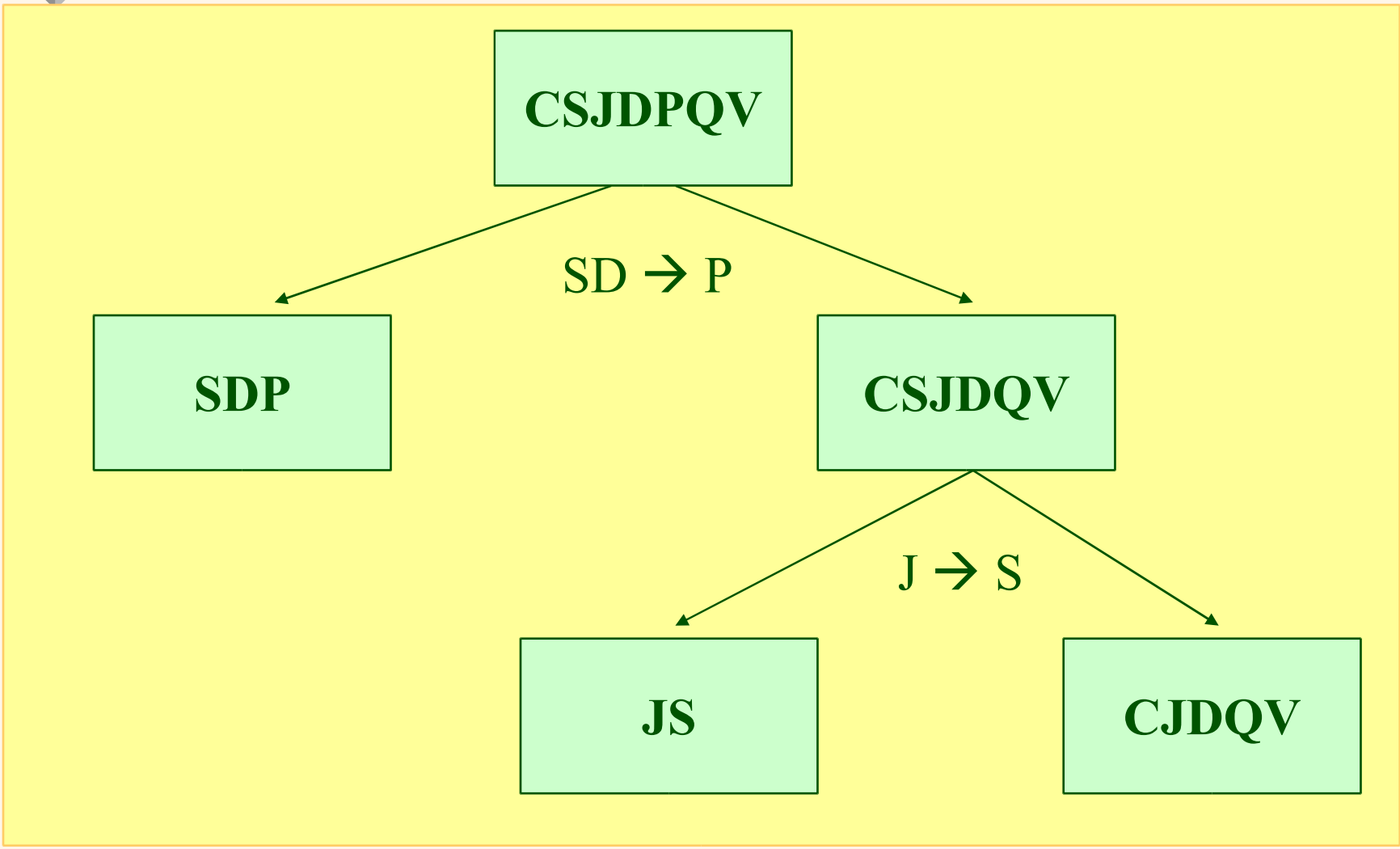
C → contract (contrato)
S → supplier (fornecedor)
J → project (projeto)
D → departament (departamento)

P → part (parte)
Q → quantity (quantidade)
V → value (valor)

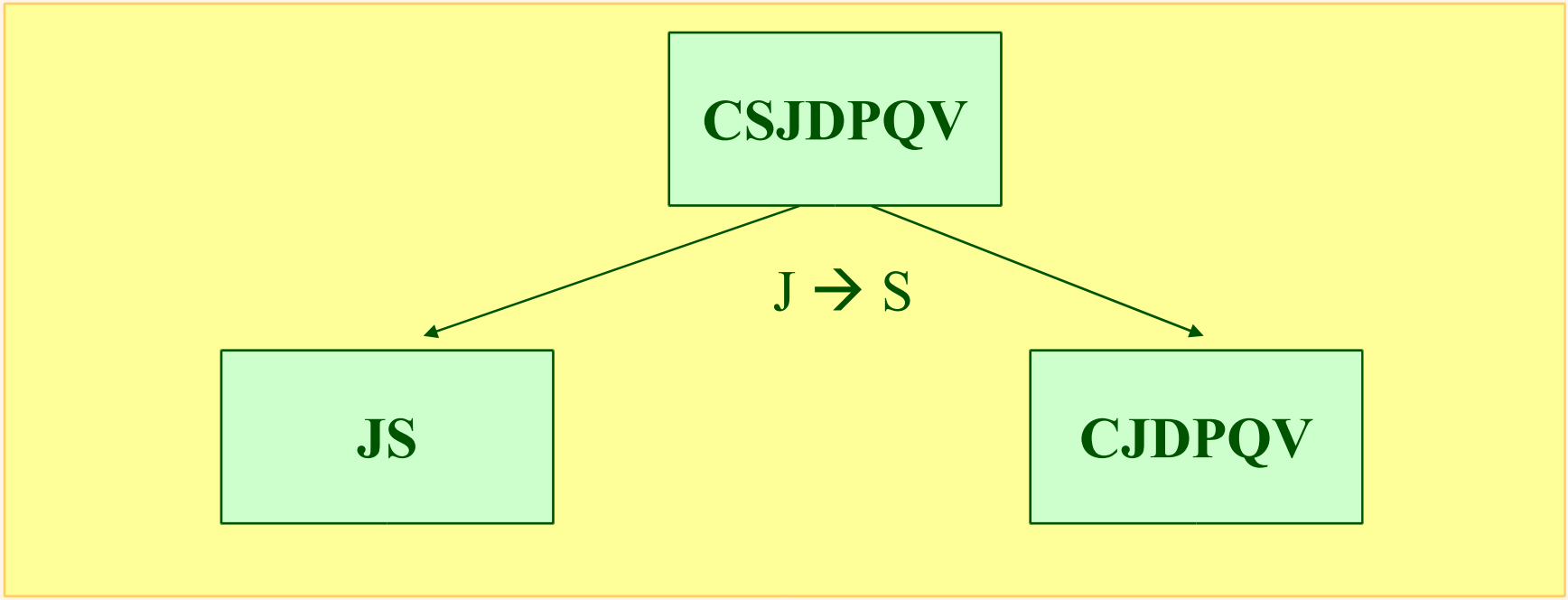
Decomposição em BCNF (Cont.)

- Considere a relação R com DFs F. Se $X \rightarrow Y$ viola a BCNF, decomponha R em R - Y e XY.
  - A aplicação repetida desta idéia nos dará uma coleção de relações que estão em BCNF; decomposições 'junção sem perda', e com garantia de término.
  - ex., CSJDPQV, chave C, $JP \rightarrow C$, $SD \rightarrow P$, $J \rightarrow S$
  - $SD \rightarrow P$, decomponha em SDP, CSJDQV.
  - $J \rightarrow S$, decomponha CSJDQV em JS e CJDQV.
- Em geral, muitas DFs podem causar violação da BCNF. A ordem na qual tratamos as DFs, pode resultar em diferentes conjuntos de relações.

Decomposição em BCNF (Cont.)



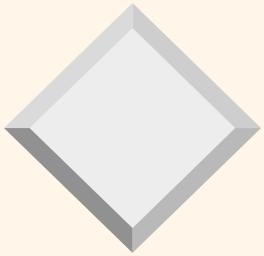
Decomposição em BCNF (alternativas)



Depois da primeira decomposição, já não há mais a DF: SD → P evitando a segunda decomposição.

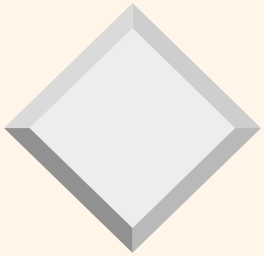
BCNF e Preservação de Dependência

- Em geral, pode não haver decomposições preservando dependência em BCNF.
  - ex., CSZ, CS $\rightarrow$ Z, Z $\rightarrow$ C
  - Não se pode decompor preservando a 1ª DF; *não está na BCNF*.
- Similarmente, a decomposição de CSJDQV em SDP, JS and CJDQV não está preservando a dependência (no que se refere as DFs JP $\rightarrow$ C, SD $\rightarrow$ P e J $\rightarrow$ S).
  - Mas é uma decomposição 'junção sem perda'.
  - Neste caso, adicionar JPC à coleção de relações nos dá uma decomposição que preserva a dependência.
    - Tuplas JPC armazenadas somente para verificar a DF! (*Redundância!*)



Normalização

- Decompor relações em BCNF.
- **Decompor relações em 3NF.**



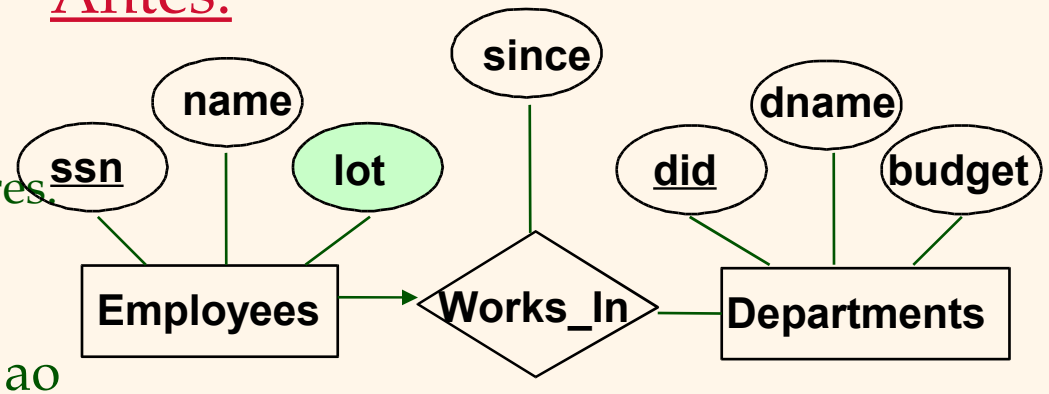
Decomposição na 3NF

- Obviamente, o algoritmo para decomposição 'junção sem perda' em BCNF pode ser usado para obter uma decomposição 'junção sem perda' na 3NF (tipicamente, pode parar antes).
- Para garantir a preservação de dependência, uma idéia:
  - Se $X \rightarrow Y$ não foi preservado, adicione a relação XY.
  - O problema é que XY pode violar a 3NF! ex., considere a adição de CJP para `preservar' $JP \rightarrow C$. E se nós também tivermos $J \rightarrow C$?

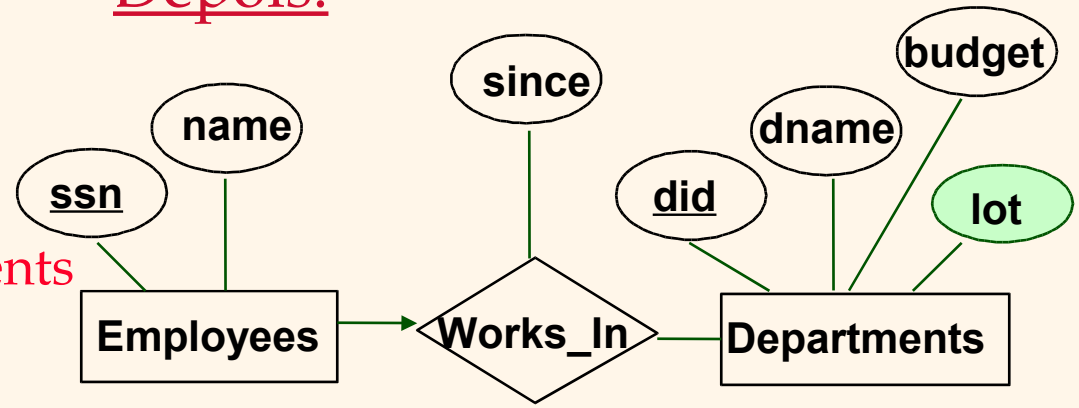
Refinando um diagrama ER

- 1o diagrama traduzido:
  Workers(S,N,L,D,S)
  Departments(D,M,B)
  - Lots associado com trabalhadores.
- Suponha que todos os trabalhadores de um departamento sejam associados ao mesmo lot: D → L
- Redundância; corrigida por:
  Workers2(S,N,D,S) Dept_Lots (D,L)
  Departments(D,M,B)
- Ajuste fino:
- Workers2(S,N,D,S) Departments (D,M,B,L)

Antes:

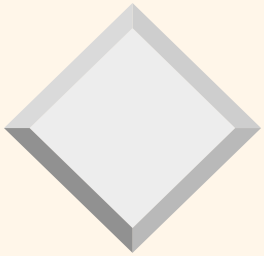


Depois:



Referências

- E. F. Codd, “Normalized Data Base Structure: A Brief Tutorial.”, Proc. 1971 ACM SIGFIDET Workshop on Data Description , Access and Control.
- E. F. Codd, “Recent Investigations into Relational Data Base Systems.”, Proc. IFIP 1974. (BCNF).
- R. Fagin, “Normal Forms and Relational Database Operators.”, Proc. 1979 ACM SIGMOD Int. Conference on Management of Data. (forma normal projeção/junção).
- W. W. Armstrong, “Dependency Structures of Data Base Relationships”, Proc. 1974 IFIP Congress. (formalizou a teoria dos FDs).



Sumário

- Se uma relação está na BCNF, ela está livre de redundâncias que podem ser detectadas usando DFs. Portanto, tentar garantir que todas as relações estejam em BCNF é uma boa heurística.
- Se uma relação não está na BCNF, nós podemos tentar a decompor em uma coleção de relações BCNF.
  - Devemos considerar se todas as DFs são preservadas. Se uma decomposição 'junção sem perda' e preservando a dependência em BCNF não é possível (ou inconveniente, dadas as *consultas* típicas), devemos considerar a decomposição na 3NF.
  - Decomposições pode ser realizadas e/ou reexaminadas quando tivermos requisitos de performance em mente.