Normalização do Esquema Relacional

Os Males da Redundância

- Redundância é a causa de vários problemas com esquemas relacionais:
  - armazenamento redundante, anomalias de inserção, de exclusão e de atualização.
- Restrições de integridade, particularmente dependências funcionais, podem ser usadas para identificar esquemas com esses problemas e para sugerir refinamentos.
- Principal técnica de refinamento: a decomposição de um esquema em sub-esquemas.
- A decomposição deve ser usada judiciosamente:
  - Há motivos para se decompor uma relação?
  - A decomposição pode causar problemas?

Os Males da Redundância (cont.)

- Considere o esquema:

  Pacientes(Id, Nome, Endereço, Telefone, Sexo, Data_nascimento, Sigla_convênio, Nome_convênio, Endereço_convênio, Telefone_convênio)

- Esse é um exemplo de mau projeto!
  - Os dados de pacientes e os de convênios não deveriam estar na mesma tabela.
- Por que?
  - Os dados de um convênio (nome, endereço e telefone do convênio) são repetidos para cada paciente associado a esse convênio. Por exemplo, os dados da AMIL serão repetidos para cada um de seus associados.

Os Males da Redundância (cont.)

- Anomalia de inserção:
  - Quando se inserir um paciente é preciso inserir também os dados do convênio, mesmo que já estejam cadastrados.
  - Não é possível inserir um convênio sem inserir também um paciente.
- Anomalia de exclusão:
  - Ao se excluir um paciente, se este for o único associado de um convênio então os dados do convênio serão perdidos.
- Anomalia de modificação:
  - Para se modificar os dados de um convênio, é preciso atualizar os mesmos dados em todas as tuplas de pacientes que estejam associados àquele convênio.

Dependências Funcionais

- Dependências funcionais (DFs) são restrições de integridade mais gerais que as restrições de chave.
- Exemplo de dependência funcional:

{Sigla_convênio} → {Nome_convênio, Endereço_convênio, Telefone_convênio}

  - Leia-se: Sigla_convênio determina funcionalmente Nome_convênio, Endereço_convênio e Telefone_convênio.
  - Significado: "Se duas linhas da tabela Pacientes tiverem o mesmo valor de Sigla_convênio, então elas tem de ter o mesmo valor de Nome_convênio, de Endereço_convênio e de Telefone_convênio."
  - Em outras palavras: "Não é válida uma tabela que tenha duas linhas que coincidem na(s) coluna(s) listadas antes da seta (→), mas são diferentes em alguma coluna listada depois da seta."

Dependências Funcionais (cont.)

- Como toda restrição de integridade, DFs são baseadas na semântica da aplicação.
  - Podemos checar uma instância de tabela e ver se uma DF é violada ou não. Mas examinando uma instância NUNCA podemos concluir uma DF deve ser imposta ou não.
  - Uma DF diz respeito a *todas as possíveis* instâncias!
- Uma restrição de chave é um caso especial de DF: a chave (ou superchave) determina funcionalmente todos os outros atributos da tabela.
  - Como Id é chave da tabela Pacientes, temos que

{Id} → {Nome, Endereço, Telefone, Sexo, Data_nascimento, Sigla_convênio, Nome_convênio, Endereço_convênio, Telefone_convênio}

Inferindo DFs

- Dadas algumas DFs, podemos geralmente inferir outras.
  - Dadas as DFs

    {Id} → {Sigla_convênio}
    {Sigla_convênio} → {Endereço_convênio}

    podemos inferir que {Id} → {Endereço_convênio}
- Regras de inferência (X, Y e Z são conjuntos de atributos):
  - Reflexividade: Se $Y \subseteq X$ então $X \rightarrow Y$.
  - Aumentação: Se $X \rightarrow Y$, então $XZ \rightarrow YZ$.
  - Transitividade: Se $X \rightarrow Y$ e $Y \rightarrow Z$ então $X \rightarrow Z$.
  - União: Se $X \rightarrow Y$ e $X \rightarrow Z$ então $X \rightarrow YZ$.
  - Decomposição: Se $X \rightarrow YZ$ então $X \rightarrow Y$ e $X \rightarrow Z$.

Fecho de um Conjunto de DFs

- O fecho (*closure*) de um conjunto de dependências funcionais F é o conjunto de todas as DFs que podem ser inferidas a partir de F.
- Não é preciso listar *todas* as DFs impostas sobre um esquema. Normalmente especificarmos um subconjunto dessas DFs e considerarmos o fecho desse subconjunto.
  - Para o esquema Pacientes, é suficiente especificarmos as DFs:

    {Id} → {Nome, Endereço, Telefone, Sexo, Data_nascimento, Sigla_convênio}
    {Sigla_convênio} → {Nome_convênio, Endereço_convênio, Telefone_convênio}
- O fecho inclui dependências funcionais triviais (aquelas que são satisfeitas por todas as instâncias de tabela possíveis).
  - As dependências triviais tem a forma $X \rightarrow Y$, onde $Y \subseteq X$.
  - Exemplos: {Nome, Endereço} → {Nome}
    {Nome} → {Nome}

Formas Normais

- Certas DFs causam redundância!
  - Para cada associado de um convênio, os dados do convênio são repetidos na tabela Pacientes. A causa desse problema é a DF

    {Sigla_convênio} → {Nome_convênio, Endereço_convênio, Telefone_convênio}

- Se uma relação estiver numa certa forma normal (FNBC ou 3FN), tais problemas são evitados ou minimizados.
  - Essas formas normais são definidas em termos de dependências funcionais.
  - Elas nos fornecem critérios para decidir o esquema de uma relação deve ser decomposto em sub-esquemas ou não.
  - Existem várias formas normais: 2FN, 3FN, FNBC, 4FN.
    - Estudaremos as que têm importância prática (FNBC e 3FN).

Forma Normal de Boyce-Codd

- Uma relação está na FNBC (*BCNF*) se todas as suas dependências funcionais não triviais forem da forma

  superchave → *conjunto-de-atributos*

  - Em outras palavras: além das dependências funcionais triviais, só podemos ter restrições de chave.
  - Uma relação na FNBC está livre de redundâncias que podem ser detetadas usando DFs.
- O esquema Pacientes não está na FNBC: a dependência funcional

  {Sigla_convênio} → {Nome_convênio, Endereço_convênio, Telefone_convênio}

  é uma violação da FNBC, pois Sigla_convênio não é chave.

Decomposição na FNBC

- Decompomos Pacientes nas duas relações abaixo.

  Pacientes1(Id, Nome, Endereço, Telefone, Sexo, Data_nascimento, Sigla_convênio)
  Convênio(Sigla_convênio, Nome_convênio, Endereço_convênio, Telefone_convênio)

  – Essas relações estão na FNBC com respeito ao conjunto de DFs

  $\{Id\} \rightarrow \{Nome, Endereço, Telefone, Sexo, Data\_nascimento, Sigla\_convênio\}$
  $\{Sigla\_convênio\} \rightarrow \{Nome\_convênio, Endereço\_convênio, Telefone\_convênio\}$

  – Agora a segunda dependência funcional não viola a FNBC, pois Sigla_convênio é chave de Convênios.

- Essa decomposição elimina a redundância do esquema original Pacientes.

Problemas com Decomposições

Três problemas potenciais devem ser considerados:

1. Certas consultas requerem mais processamento (junções).
   – Exemplo: Qual o telefone do convênio do paciente José da Silva?
2. Perda de informação: Dadas instâncias das tabelas decompostas, pode não ser possível reconstruirmos a instância correspondente da tabela original!
   – Isso é inaceitável e não acontece no caso de nosso exemplo.
3. Custo de verificação de dependências: Para verificar se uma dependência é satisfeita pode ser preciso efetuar junções das tabelas decompostas (a verificação dessa dependência é custosa).
   – Geralmente inaceitável, não acontece no caso de nosso exemplo.

Decomposição Sem Perdas

- Uma decomposição é sem perdas se for sempre possível reconstruir a instância da tabela original efetuando a junção das instâncias correspondentes das tabelas decompostas.
  - Exemplo de decomposição com perdas:

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

perda de informação
(ganho de tuplas)

Decomposição Sem Perdas: Critério

A decomposição de um esquema R em sub-esquemas X e Y é sem perdas se e somente se pelo menos uma das duas DFs abaixo valer:

$X \cap Y \rightarrow X$,
ou $X \cap Y \rightarrow Y$.

Neste exemplo nenhuma delas vale!

- » Caso especial: se valer a dependência $U \rightarrow V$, então a decomposição de R em UV e R – V é sem perdas.
  - ☞ Verifique que a decomposição de Pacientes satisfaz esta condição!

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

perda de informação
(ganho de tuplas)

Algoritmo de Decomposição na FNBC

- Considere uma relação R e as DFs associadas a ela.
  Se X → Y violar a FNBC, decomponha R em XY e R – Y.
  - Aplicando esta idéia repetidamente, obteremos uma decomposição sem perdas de R em uma coleção de relações na FNBC.
  - Em geral, mais de uma DF pode violar a FNBC. Dependendo da ordem com que usarmos as dependências violadoras, podemos obter decomposições diferentes (e mesmo assim corretas).
- Exemplo: o esquema

  Empréstimos1(Nome_agência, Ativos, Cidade_agência, Num_empréstimo, Nome_cliente, Quantia)

  e as DFs

  ❶ {Nome_agência} → {Ativos, Cidade_agência}
  ❷ {Num_empréstimo} → {Quantia, Nome_agência}

  - Note que {Num_empréstimo, Nome_cliente} é a única chave.

Algoritmo de Decomposição na FNBC (cont.)

Empréstimos1(Nome_agência, Ativos, Cidade_agência, Num_empréstimo, Nome_cliente, Quantia)

❶ {Nome_agência} → {Ativos, Cidade_agência}
❷ {Num_empréstimo} → {Quantia, Nome_agência}

- A DF ❶ viola a FNBC, pois Nome_agência não é superchave de Empréstimos1. Portanto decompomos essa tabela em

  Agências(Nome_agência, Ativos, Cidade_agência)
  Empréstimos2(Nome_agência, Num_empréstimo, Nome_cliente, Quantia)

- A DF ❷ viola a FNBC, pois Num_empréstimo não é superchave de Empréstimos2. Portanto decompomos essa tabela em

  Empréstimos(Num_empréstimo, Nome_agência, Quantia)
  Tomadores(Num_empréstimo, Nome_cliente)

- Resultado: 3 tabelas (Agências, Empréstimos e Tomadores).

Preservação de Dependências

- Considere o esquema e as DFs:

  Gerentes(Nome_agência, Nome_cliente, Nome_gerente)

  ❶ {Nome_gerente} → {Nome_agência}
  ❷ {Nome_cliente, Nome_agência} → {Nome_gerente}

  (Cada cliente de uma agência tem um "gerente pessoal" na agência.)

- Uma decomposição FNBC é:

  Gerentes_agências(Nome_gerente, Nome_agência)
  Gerentes_clientes(Nome_cliente, Nome_gerente)

- Problema: É necessário fazer uma junção para verificar se a segunda DF é violada ou não.
  - Inaceitável! Teríamos que fazer uma junção após cada atualização dessas tabelas, apenas para assegurarmos que a DF ❷ é cumprida.

Preservação de Dependências (cont.)

- Decomposição com preservação de dependências:
  - Considere uma relação R e as DFs associadas a ela, bem como uma decomposição de R em R1, R2, …, Rn. Se, para cada Ri, assegurarmos que as DFs aplicáveis a Ri (aquelas que envolvem somente atributos de Ri) estão satisfeitas, então podemos ter certeza que todas as DFs associadas a R estão satisfeitas.
- Nossa decomposição FNBC da tabela Gerentes não tem esta propriedade.
  - A dependência {Nome_cliente, Nome_agência} → {Nome_gerente} foi "perdida".
- É importante considerar o fecho do conjunto de DFs!
  - Exemplo: R(A,B,C), com DFs {A} → {B}, {B} → {C} e {C} → {A}.
    - A decomposição de R em (A,B) e (B,C) é com preservação de dependências? A terceira DF é preservada ou não?

Terceira Forma Normal (3FN)

- Uma relação R está na 3FN (*3NF*) se cada uma de suas dependências funcionais da forma X → A cair num dos casos abaixo:
  ① A ∈ X (isto é, a DF é trivial), ou
  ② X é uma superchave de R, ou
  ③ A faz parte de alguma chave de R (A é um atributo primo).

  Aqui X é um conjunto de atributos e A é um atributo simples.
- Em outras palavras: além das DFs triviais, só podemos ter restrições de chave ou dependências da forma

  conjunto-de-atributos → *atributo-primo*

  – A 3FN é uma forma normal mais fraca (menos restritiva) que a FNBC. Toda relação que estiver na FNBC estará também na 3FN, mas a recíproca não é verdadeira.

Exemplo de relação na 3FN mas não na FNBC

- Considere novamente o exemplo do "gerente pessoal":

  Gerentes(Nome_agência, Nome_cliente, Nome_gerente)

  ❶ {Nome_gerente} → {Nome_agência}
  ❷ {Nome_cliente, Nome_agência} → {Nome_gerente}

  – A tabela tem as seguintes chaves candidatas:
    {Nome_cliente, Nome_agência}
    {Nome_cliente, Nome_gerente}

    Todos os atributos são primos!
  – Essa tabela não está na FNBC, mas está na 3FN!
    - A DF ❶ é uma violação da FNBC, pois Nome_gerente não é chave.
    - Ela não viola a 3FN, pois Nome_agência é um atributo primo.
  – Note que a tabela Gerentes não está livre de problemas de redundância.
    - A associação entre um gerente e sua agência é repetida para cada cliente desse gerente.

FNBC ou 3FN?

- A FNBC nos assegura que uma relação está livre de redundâncias detectáveis através de DFs.
- Com a 3FN, alguma redundância ainda é possível.
  - A 3FN é uma solução de compromisso!
- Dada uma relação que não está na FNBC, nem sempre é possível conseguirmos uma decomposição FNBC que seja *sem perdas* e *com preservação de dependências*.
  - Se insistirmos numa decomposição FNBC, poderemos ter que abrir mão da preservação de dependências.
  - Mas é sempre possível uma decomposição 3FN que seja sem perdas e com preservação de dependências.
- Nos casos (raros) em que for necessário optar entre a FNBC e a preservação de dependências, ficamos com a preservação de dependências e com uma forma normal mais fraca, a 3FN.

Conclusões

- Como a FNBC nos garante que não temos redundância detectável através de DFs, devemos considerar essa forma normal como um objetivo desejável do projeto lógico.
- Se uma relação não estiver na FNBC, podemos tentar decompô-la num coleção de relações na FNBC.
  - A decomposição deve ser sem perdas e com preservação de dependências.
  - Se isso não for possível, considerar decomposição na 3FN.
- Lembrar que a decomposição tem um custo (junções).
  - Não assuma a postura rígida, dogmática, de normalizar a todo custo!
  - Decomposições devem ser efetuadas e reexaminadas tendo em mente os requisitos de desempenho da aplicação.