Universidade Federal da Fronteira Sul - UFFS
Campus Chapecó
Ciência da Computação
Banco de Dados I
Prof.: Denio Duarte

Notas de Aula - Projeto BD - Cap. 03 - A First Course in Database System

O projeto de um banco de dados deve ser feito de tal forma que os dados sejam armazenados de forma eficaz, ou seja, sem redundância, sem problemas de atualização, entre outros. Para resolver tais problemas, algumas teorias devem ser estudadas. Isso se faz necessário pois para uma determinada aplicação podemos ter vários esquemas de banco de dados diferentes e escolher qual deles é o melhor não é uma tarefa trivial.

Este texto tem como objetivo apresentar as seguintes teorias para um bom projeto de banco de dados: dependências funcionais e formas normais.

Dependências Funcionais

Uma dependência funcional (DF)[1] em uma relação R é descrita na forma *se duas tuplas em R têm os mesmos valores para os atributos A_1, A_2, ..., A_n, então os atributos B_1, B_2, ..., B_m também devem ter os mesmos valores.* Essa restrição (ou seja, dependência funcional) é escrita como $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$, e dizemos que:

A_1, A_2, ..., A_n determina funcionalmente B_1, B_2, ..., B_m

Se temos certeza que uma instância de R é verdadeira para uma dada DF, é dito que R satisfaz a DF. Uma FD f é uma restrição em R e deve ser verdadeira para todas as instâncias de R.

O lado direito de uma FD pode ter apenas um atributo, assim a FD $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ pode ser decomposta e reescrita como:
$A_1A_2 \ldots A_n \rightarrow B_1$, $A_1A_2 \ldots A_n \rightarrow B_2$, ..., e $A_1A_2 \ldots A_n \rightarrow B_m$

Dada um esquema de relação $Movie1(title, year, length, genre, studioName, starName)$ com a instância

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

Para fins didáticos, a relação $Movie1$ foi criada armazenando dados de três entidades distintas: filmes, estúdios e protagonistas de filmes. Assim, o esquema não representa um bom projeto de BD.

A identificação dos erros de projeto podem ser vista definindo, primeiro, as dependências funcionais. Percebe-se que a DF *title year* $\rightarrow$ *length genre* $studioName$ é respeitada. Essa FD diz que se duas tuplas têm o mesmo valor para *title* (título do filme) e para *year* (ano de lançamento do filme), essas tuplas devem ter o mesmo valor para *length* (duração), *genre* (gênero) e $studioName$ (nome do estúdio). Perceba se acrescentarmos $starName$ na DF anterior, a relação $Movie1$ não será válida para a mesma. A DF *title year* $\rightarrow$ $starName$ também não seria respeitada pois é aceitável que um filme tenha mais de um protagonista.

[1] Abreviada também como FD (Functional Dependency)

Baseado no conceito de dependência funcional, podemos revisitar o conceito de chave. Dizemos que um conjunto de um ou mais atributos $\{A_1, A_2, \ldots, A_n\}$ é uma chave de uma relação R se:

1. Esses atributos determinam funcionalmente todos os outros atributos. Ou seja, é impossível encontrar duas tuplas em R que possuam os mesmos valores para $A_1, A_2, \ldots, A_n$.
2. Nenhum subconjunto de $\{A_1, A_2, \ldots, A_n\}$ determina funcionalmente todos os outros atributos de R, *i. e.*, a chave deve ser mínima.

Por exemplo, os atributos {*title*, *year*, *starName*} forma a chave da relação $Movie1$. Isso é verdadeiro pois se tirarmos *starName* da chave sabemos que teremos valores diferentes desse atributo para a chave {*title*, *year*}. Também, não encontraremos um subconjunto menor que {*title*, *year*, *starName*} que determine funcionalmente os outros atributos, por exemplo, a DF *year starName* $\rightarrow$ *title* não é valida na instância da relação $Movie1$.

Lembre-se que, às vezes, uma relação pode ter mais de uma chave. Nesse caso, é necessário escolher uma delas como chave primária. Nos SGBD comerciais a PK possui um papel importante na implementação das estratégias de armazenamento e acesso aos dados. Porém, na teoria das DF's não é dada nenhuma atenção especial a elas.

Como visto em outros encontros, uma chave é chamada de *superkey*, ou seja, toda a chave é uma superkey porém nem toda superkey é uma chave (mínima).

Regras para Dependências Funcionais

Às vezes, um conjunto de DFs é dado como atendido por uma certa relação. Baseado neste conjunto, é possível inferir outras DFs. A habilidade em descobrir estas outras DFs é importante para um bom projeto de banco de dados.

Suponha que a relação $R(A, B, C)$ satisfaz as DFs $A \rightarrow B$ e $B \rightarrow C$, é possível inferir que R satisfaça $A \rightarrow C$? Sim, pois se temos duas tuplas t_1 e t_2 em R com $t_1[A] = a$ e $t_2[A] = a$, e como a DF $A \rightarrow B$ é verdadeira em R, deveremos ter o valor de B igual nessas tuplas, digamos $t_1[B] = b$ e $t_2[B] = b$. Como a DF $B \rightarrow C$ também é válida em R, os valores do atributo C devem ser iguais, digamos $t_1[C] = c$ e $t_2[C] = c$. Neste caso, concluimos que as duas tuplas são $< a, b, c >$ e $< a, b, c >$ e assim idenficamos que para algum valor do atributo A teremos o mesmo valor para o atributo C, ou seja, $A \rightarrow C$ é verdadeiro.

Regra da combinação/decomposição: podemos decompor o lado direito de uma DF até termos apenas um atributo no lado direito. Assim, podemos decompor a DF $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ no conjunto $A_1A_2 \ldots A_n \rightarrow B_1$, $A_1A_2 \ldots A_n \rightarrow B_2$, ..., $A_1A_2 \ldots A_n \rightarrow B_m$. O contrário também é verdadeiro, ou seja, podemos combinar um conjunto de DFs com o mesmo lado esquerdo. Utilizando o exemplo anterior, se tivermos um conjunto de DFs $A_1A_2 \ldots A_n \rightarrow B_1$, $A_1A_2 \ldots A_n \rightarrow B_2$, ..., $A_1A_2 \ldots A_n \rightarrow B_m$, podemos combiná-lo em apenas uma DF $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$.

Devemos ter cuidado com a combinação/decomposição pois podem ser aplicadas apenas no lado direito das DFs. Por exemplo, baseado na relação $Movie1$, a DF *title year* $\rightarrow$ *length* é respeitada. Se decompormos o lado esquerdo para *title* $\rightarrow$ *length* e *year* $\rightarrow$ *length*, provavelmente essas DFs não serão respeitadas por alguma instância de $Movie1$ pois filmes com, por exemplo, anos de lançamentos iguais e duração diferentes podem ocorrer em $Movie1$.

Dependências Funcionais Triviais: uma restrição de qualquer tipo é dita trivial se é verdadeira para qualquer instância da relação, independente de outras restrições existentes. Quando tais restrições são DFs, é fácil indentificar a trivialidade das mesmas. As DFs na forma $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$ tal que $\{B_1B_2 \ldots B_m\} \subseteq \{A_1A_2 \ldots A_n\}$, são triviais pois o lado direito é um subconjunto do lado esquerdo.

Por exemplo, *title year* $\rightarrow$ *title* é trivial, tal como, *title* $\rightarrow$ *title*

Cálculo do Fechamento dos Atributos: às vezes é necessário identificar se uma determinada DF é pode ser inferida a partir de um conjunto de DFs. Para tal, é necessário calcular o fechamento para um conjunto de atributos. Por exemplo, se gostaríamos de saber se a DF $A_1A_2 \dots A_n \rightarrow B$ pode ser inferida a partir de um conjunto de DFs S, primeiro calculamos o fechamento de $\{A_1, A_2, \dots A_n\}$, denotado por $\{A_1, A_2, \dots A_n\}^+$. Se B pertencer a $\{A_1, A_2, \dots A_n\}^+$ então a DF em questão pode ser acrescentada em S.

O algoritmos que constrói o fechamento de um conjunto de atributos é simples:

Entrada: Um conjunto de atributos $\{A_1, A_2, \dots A_n\}$ e um conjunto de DFs S

Saída: O fechamento $\{A_1, A_2, \dots A_n\}^+$

1. Se necessário, decomponha todas as DFs em S para que tenham apenas um atributo no lado direito
2. X será o conjunto de atributos que se tornarão o fechamento. X deve ser inicializado com $\{A_1, A_2, \dots A_n\}$.
3. Repetidamente procure por alguma FD $B_1B_2 \dots B_m \rightarrow C$, tal que $B_1B_2 \dots B_m$ esteja em X e C não. Acrescente C a X e continue com a repetição. Quando X parar de crescer (receber novos atributos), finalize esse passo.
4. O conjunto X, após o passo anterior, é a saída do algoritmo, ou seja, $\{A_1, A_2, \dots A_n\}^+$

Considere o esquema de relação $R = (A, B, C, D, E, F)$ e o conjunto S de DFs $AB \rightarrow C$, $BC \rightarrow AD$, $D \rightarrow E$ e $CF \rightarrow B$ que é verdeiro para as instâncias de R. Suponha que queremos saber se $AB \rightarrow D$ pode ser inferida a partir de S. Nesse caso, temos que calcular o fechamento de $\{A, B\}$, ou seja, $\{A, B\}^+$ e verificar se $D \in \{A, B\}^+$. Executando o algoritmo, devemos fazer os seguintes passos: construir um novo conjunto S, chamaremos de S', com lados direitos com apenas um atributo. $S' = \{AB \rightarrow C,\ BC \rightarrow A,\ BC \rightarrow D,\ D \rightarrow E$ e $CF \rightarrow B\}$. Em seguida, inicializamos X com o conjunto de entrada, ou seja, $X = \{A,\ B\}$. Agora, executamos o passo 3 do algoritmo: o lado esquerdo de $AB \rightarrow C$ está em X, assim, acrescentamos C a X, temos então $X = \{A,\ B,\ C\}$. Agora podemos testar com as DFS $BC \rightarrow A$ e $BC \rightarrow D$, como A já está em X, inserimos apenas D. A próxima DF que possui o lado esquerdo em X é $D \rightarrow E$, assim E é acrescentado em X. A DF $CF \rightarrow B$ não pode ser utilizada pois F não pertence a X. Então, o algoritmo para e $X = \{A, B, C, D, E\}$. Como D é um membro de $\{A, B\}^+$, concluímos que $AB \rightarrow D$ pode ser inferida a partir de S.

Como exercício, verifique se poderíamos inferir $D \rightarrow A$ a partir do mesmo cenário acima.

Axiomas de Armstrong: pode-se encontrar todas as DFs a partir de um conjunto de DFs S. O novo conjunto será chamado de fechamento de S, denotado por S^+. Os axiomas são:

1. Reflexidade: se $\beta \subseteq \alpha$ então $\alpha \rightarrow \beta$ (trivial)
2. Aumento: se $\alpha \rightarrow \beta$ então $\gamma\alpha \rightarrow \gamma\beta$.
3. Transitividade: se $\alpha \rightarrow \beta$ e $\beta \rightarrow \gamma$ então $\alpha \rightarrow \gamma$

Para complementar o fechamento de um conjunto de depedências funcionais, utilizamos mais duas regras:

1. União: se $\alpha \rightarrow \beta$ e $\alpha \rightarrow \gamma$ então $\alpha \rightarrow \beta\gamma$ (combinação)
2. Decomposição: se $\alpha \rightarrow \gamma\beta$ então $\alpha \rightarrow \beta$ e $\alpha \rightarrow \gamma$.
3. Pseudo-transitividade: se $\alpha \rightarrow \beta$ e $\beta\gamma \rightarrow \delta$ então $\alpha\gamma \rightarrow \delta$

Aplicando as regras/axiomas acima sobre um conjunto de DFs S pode-se aumentar este conjunto obtendo um conjunto com todas as DFs possíveis, chamado conjunto de fechamento S^+. S^+ pode ter DFs redundantes. Quando temos o conjunto de fechamento das DFs sem DFs redundantes é dito que temos a *cobertura canônica* de dependências funcionais, ou seja, o menor conjunto completo das DFs de uma relação. Por exemplo, se o conjunto fechamento de DFs é $\{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C\}$, a cobertura canônica seria $\{A \rightarrow B,\ B \rightarrow C\}$ pois $A \rightarrow C$ é redundante.

Normalização

Decomposição de esquemas para evitar as anomalias de exclusão, atualização e inserção, e geração de tuplas ilegítimas (vistas em aula anterior) é chamada de normalização. A normalização é um mecanismo formal para analisar esquemas de BD baseado nas chaves e dependências funcionais.

Os esquemas normalizados resultantes devem possuir características que podem ser analisadas por ordem de complexidade. Essa análise pode ser feita utilizando as formas normais.

Formais Normais

As formas normais são utilizadas para desenvolver esquemas de banco de dados minimizados das anomalias citadas. As formas normais são orientação que o projetista do banco de dados deve considerar para fazer bons projetos. Existem seis formas normais e a hierarquia é apresentada abaixo (esta hierarquia indica que para atender uma forma normal qualquer, o esquema deve antender as formas normais anteriores a que se pretende atender):

1. Primeira forma normal (1NF)[2]
2. Segunda forma normal (2NF)
3. Terceira forma normal (3NF)
4. Forma normal de Boyce-Codd (BCNF)
5. Quarta forma normal (4NF)
6. Quinta forma normal (5NF)

No curso e neste texto, iremos tratar até a BCNF.

Primeira Forma Normal: é a forma normal mais básica e diz que todos os atributos de uma relação devem ser atômicos, ou seja, indivisíveis. Esta forma evite que uma relação tenha atributos compostos ou multivalorados. Não entraremos em detalhes com a 1NF pois é a mais simples das formas normais.

Segunda Forma Normal: um esquema relacional está na 2FN se estiver na 1NF e respeitar a seguinte restrição: todos os atributos não chave dependem funcionalmente da chave primária completa. Uma relação que tem apenas um atributo como chave primária respeita essa regra.

Suponha um esquema de relação $Encomenda(\underline{codpro}, \underline{codcli}, qtde, desc, nompro)$. Observando as PK da relação *Encomenda* conclui-se que a DF $CODPRO\ CODCLI \rightarrow QTDE\ DESC\ NOMPRO$ é respeitada pela relação. Para *Encomenda* respeitar a 2FN, todos os atributos da direita da DF devem depender de toda a chave, ou seja, dos atributos $CODPRO$ e

[2]NF é a abreviação de Normal Form.

$CODCLI$. Dado que $CODPRO$ representa o código do produto encomendado, $CODCLI$ representa o código do cliente que fez a encomenda, $QTDE$ é a quantidade encomendada, $DESC$ é o desconto recebido e $NOMPRO$ é o nome do produto encomendado, verifica-se que a dependência funcional é verdadeira para todas as instâncias de *Encomenda* pois não poderá existir duas ocorrências dos atributos $CODPRO$ e $CODCLI$ com os mesmos valores. Semanticamente, isso quer dizer que um cliente só poderá encomendar um produto apenas uma vez. Obervando com mais detalhes a relação e suas dependências funcionais, observa-se que a decomposição $CODPRO\ CODCLI \rightarrow NOMPRO$ é verdadeira mas poderia ser simplificada pois o código de um produto deve ter sempre o mesmo nome, não precisando do código do cliente para caracterizá-lo. Assim, a DF $CODPRO \rightarrow NOMPRO$ também é verdadeira em *Encomenda*. Nesse caso, a relação *Encomenda* não respeita a 2FN pois existem atributos não chave que não dependem da chave primária completa.

Como resolveríamos a situação acima? Basta tirar o atributo $NOMPRO$ da tabela *Encomenda* e criar uma nova tabela para armazenar os códigos e nomes de produtos. O resultado seria $Encomenda(\underline{codpro}, \underline{codcli}, qtde, desc)$ e $Produto(\underline{codpro}, nompro)$. A tabela *Produto* respeita a 2NF pois respeita a 1NF e sua chave primária contém apenas um atributo.

Terceira Forma Normal: esta orientação existe para que não exista, entre outros, uma dependência transitiva entre a chave e seus atributos. Por exemplo, suponha que tenhamos a dependência $A \rightarrow B$ e A é a chave da relação. Suponha também, que nesta mesma relação exista a DF $B \rightarrow C$. Neste caso, a relação em questão não respeitaria a 3FN.

Mais formalmente, Dados um esquema R, um conjunto de DFs S válidos sobre R, X um subconjunto de atributos de R e A um atributo de R. R está na 3FN se para toda DF $X \rightarrow A$ em S, uma das seguintes declarações é verdadeira:

1. $A \in X$, isto é, uma dependência funcional trivial, ou
2. X é uma superchave, ou
3. A faz parte de alguma chave de R.

A terceira condição é a mais importante para se entender: uma chave é um conjunto mínimo de atributos que determina exclusivamente todos os outros atributos. A deve fazer parte de qualquer chave de R (possivelmente, apenas uma), não é suficiente que A faça parte de uma super chave pois essa condição é satisfeita por todo atributo da relação.

Suponha que uma DF $X \rightarrow A$ provoque a violação da 3FN porém a relação respeite a 2FN. Teremos a seguinte situação: X não é subconjunto de nehuma chave e essa dependência é chamada de transitiva, pois significa que temos um encadeamento de dependências $K \rightarrow X \rightarrow A$

Por exemplo, dada a relação $Carro(\underline{placa}, modelo, cor, montad)$, com tuplas ($< XX1010$, Gol, $Azul$, $VW >$, $< YY2321$, Ka, $Branco$, $Ford >$, $< MM4544$, Fox, $Prata$, $VW >$, $< RR1000$, $Bora$, $Branco$, $VW >$, $< XX1000$, Ka, $Preto$, $Ford >$) e a chave indica que as DFs $PLACA \rightarrow MODELO$, $PLACA \rightarrow COR$ e $PLACA \rightarrow MONTAD$ são respeitadas em *Carro*. Porém, pode-se observar que a DF $MODELO \rightarrow MONTAD$ também sempre será respeitada em *Carro*. Assim, temos uma transitividade $PLACA \rightarrow MODELO$ e $MODELO \rightarrow MONTAD$. Ou seja, *Carro* não respeita a 3FN. Para acertar essa situação, seria necessário decompor a tabela *Carro*, movendo o atributo $MONTAD$ para uma nova tabela adicionada do atributo $MODELO$.

Outro exemplo: suponha a relação $Campeao(\underline{torneio}, \underline{ano}, clube, fundclube)$ que armazena os campeões de torneios realizados. As tuplas de *Campeao* são ($< CopaBrasil$,2011, $Vasco$, 1898 $>$, $< CopaBrasil$, 2009, $Corinthians$, 1910 $>$, $< Brasileirao$, 2000, $Vasco$, 1898 $>$, $< Catarinense$, 2011, $Chapecoense$, 1973 $>$). Pela chave primária temos as DFs *torneio ano* $\rightarrow$ *clube* e *torneio ano* $\rightarrow fundclube$. Porém, percebe-se também que a DF *clube* $\rightarrow fundclube$

também é válida e temos, então, uma transitividade: $torneio\ ano \rightarrow clube$ e $clube \rightarrow fundclube$, assim a relação $Campeao$ não respeita a 3FN.

O atributo a esquerda da DF que causa o problema na 3NF deve ser um atributo não chave. Por exemplo, a relação $aluno(\underline{cpf}, mat, nome, ender)$ está na 3FN apesar da DF $mat \rightarrow nome$ ser válida e existir a DF $cpf \rightarrow mat$. Nesse caso, não existe transitividade pois mat é também uma chave.

Forma Normal de Boyce-Codd: o objetivo de decompor uma relação em outras é evitar as anomolias de projeto. A BCNF é a forma normal que garante que as anomalias de atualização não ocorrem nas relações decompostas. Uma relação R está na BCNF se R está na 3FN e obedece a regra:

- Todas as vezes que existir um DF não trivial $A_1A_2 \ldots A_n \rightarrow B_1B_2 \ldots B_m$, os atributos $A_1A_2 \ldots A_n$ formam um super chave em R.

Ou seja, o lado esquerdo de qualquer DF de uma relação deve ser uma super chave. Pode ser entendido: o lado esquerdo de qualquer DF deve conter uma chave. Lembrando: uma super chave é um conjunto de atributos (talvez apenas 1) que identifica unicamente uma tupla de uma relação. A chave é uma super chave que, ao se retirar um atributo, deixa se ser super chave.
Exemplo: dada a relação $ItemNFiscal(\underline{numnota}, \underline{codprod}, codcli, codven, qtd, prc)$, as super chaves são:
$numnota, codprod, codcli, codven, qtd$, podemos tirar o atributo prc que ainda temos a super chave $numnota, codprod, codcli, codven, qtd$, retirando qtd temos $numnota, codprod, codcli, codven$ que ainda é uma super chave. Retirando $codven$ temos $numnota, codprod, codcli$ que ainda é uma super chave, o mesmo vale para a retirada do atributo $codcli$. Quando termos $numnota, codprod$ não é mais possível retirar atributos da super chave, assim, temos a chave (ou super chave mínima). Caso, utilizarmos a chave para identificar unicamente a relação, esta chave é chamada de chave primária. Uma relação pode ter mais de uma chave. As chaves não escolhidas como chave primária são chamadas de chaves candidatas.

Voltando a BCNF, todas as dependências funcionais de uma relação devem ter, no seu lado esquerdo, atributos pertencentes a uma super chave.

A relação $Movie1$ não respeita a BCNF. Inicialmente, devemos identificar os atributos que são chaves. Os atributos $title\ year\ starName$ formam a chave de $Movie1$ e, assim, qualquer super conjunto dessa chave será uma super chave. Se considerarmos a DF apresentada anteriormente $title\ year \rightarrow length\ genre\ studioName$ que é respeitada em $Movie1$. O lado esquerdo dessa DF (*i.e.*, $title\ year$) não é uma super chave, assim, essa DF viola a BCNF e, portanto, $Movie1$ não está na BCNF.

Dada a relação $Movie2(title, year, length, genre, studioName)$ com a instância

title	year	length	genre	studioName
Star Wars	1977	124	SciFi	Fox
Star Wars	1977	124	SciFi	Fox
Star Wars	1977	124	SciFi	Fox
Gone With the Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount
Wayne's World	1992	95	comedy	Paramount

podemos verificar que $Movie2$ respeita a BCNF pois a única chave é $title\ year$, ou seja, $title\ year \rightarrow length\ genre\ studioName$ é respeitada por $Movie2$ e nem $year$ nem $title$ determinam funcionalmente qualquer outro atributo. Qualquer outra super chave será um super conjunto de $title\ year$.