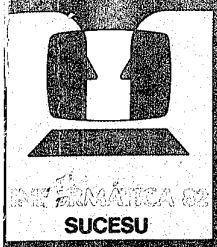


INFORMÁTICA 82

XV CONGRESSO NACIONAL DE INFORMÁTICA
II FEIRA INTERNACIONAL DE INFORMÁTICA

ANAIS

INFORMÁTICA 82

XV CONGRESSO NACIONAL DE INFORMÁTICA
II FEIRA INTERNACIONAL DE INFORMÁTICA

ANAIS

"DEPENDÊNCIAS FUNCIONAIS E DEPENDÊNCIAS DE INCLUSÃO: APLICAÇÕES AO PROJETO LÓGICO DE BANCO DE DADOS"

Severino Pompilho do Rego e Marco Antonio Casanova
Furnas Centrais Elétricas S. A.
Rua Real Grandeza, 219 – Rio de Janeiro, RJ.
Pontifícia Universidade Católica do Rio de Janeiro
22453 – Rio de Janeiro, RJ.

1. INTRODUÇÃO

O projeto físico dos bancos de dados tem como principal preocupação a maneira como os dados estão organizados na máquina (sua representação interna). Entretanto, a utilização dos dados armazenados é definida em uma fase anterior que costuma ser chamada de projeto lógico de banco de dados. É durante esta fase que se procura especificar a aplicação que será suportada pelo banco de dados. Sem dúvida alguma, podemos dizer que, se por um lado o custo do sistema a ser implementado é função da performance apresentada pelo banco de dados a ser manipulado, por outro lado, olhando do ponto de vista do usuário, o valor desse mesmo banco de dados será uma função de sua aderência a realidade do mundo no qual a aplicação está inserida, ou seja, antes de pensarmos na implementação de um banco de dados, é necessário que tenhamos uma descrição tão próxima quanto possível dos requisitos de informação da aplicação.

O presente trabalho trata desta fase preliminar, a qual estamos denominando de projeto lógico de banco de dados. Pelos motivos já apresentados acima, verificamos a necessidade de descrever o banco de dados de maneira clara, sem ambigüidades e principalmente correta. Isto não é, de modo algum, uma tarefa fácil, devido às dificuldades inerentes de se obter um entendimento correto da realidade a qual o banco de dados irá servir, pois a obtenção desses requisitos é feita via um processo informal através do qual o usuário expõe uma linguagem natural o que ele deseja. Cabe então ao projetista do banco de dados, tentar compreender aquilo que deve ser a realidade do mundo do usuário. Isto geralmente é feito através de metodologias informais que juntamente com a experiência do projetista permitem uma interpretação daquilo que o usuário deseja.

Admitindo que o projetista tenha compreendido a realidade na qual a aplicação está inserida, torna-se necessário que ele consiga descrever corretamente o banco de dados desejado. Para que isso seja conseguido com sucesso, é desejável a utilização de ferramentas formais que permitam uma descrição clara e precisa. Estas ferramentas podem ser encontradas em alguns ramos da matemática, e sua utilização tem sido cada vez mais difundida na área de banco de dados. Assim sendo, uma descrição de um banco de dados, chamada de esquema conceitual, pode ser considerada como um conjunto de estruturas de dados e um conjunto de restrições de integridade, as quais estabelecem quais as restrições a que estão sujeitos os valores dos dados a serem armazenados no banco de dados, de tal modo que, ao serem realizadas transações dos usuários, seja preservada a consistência do banco de dados.

Um subesquema descreve a visão que um grupo de usuários tem do banco de dados, e é também definido por um conjunto de estruturas de dados e um conjunto de restrições de integridade. Além disto, a definição do subesquema inclui também um mapeamento das suas estruturas sobre as do esquema conceitual.

No que diz respeito a estruturação dos dados, grande parte dos trabalhos acadêmicos em banco de dados tem feito uso do chamado "modelo relacional de dados", o qual adota tabelas planas, ou relações, como estrutura de dados básica. Várias classes de restrições de integridade, também chamadas dependências de dados, têm sido estudadas em conexão com o modelo relacional. Entre essas dependências de dados podemos citar dependências funcionais [Co1], dependências multivaloradas [Fa1, ZM], dependências de junção [ABU], dependências de inclusão [Fa5, CFP] e outras, muito embora todas essas classes sejam casos especiais de dependências implicacionais embebidas estendidas [Fa6].

Neste trabalho apresentamos algumas aplicações das dependências funcionais e das dependências de inclusão ao projeto lógico de banco de dados.

Juntamente com as DFs, as DIs formam a base do modelo estrutural de [WM]. Dentro do contexto relacional, as DIs foram usadas por [Co2, Fa6, SS]. Além disso, visto

sob outro aspecto, as DIs podem servir como uma saída para a controvérsia da suposição da relação universal. Entretanto, apesar de todas essas aplicações, quase não tem havido trabalhos que analisem as propriedades das DIs. Em um dos raros trabalhos incluindo DIs, Casanova, Fagin e Papadimitriou em [CFP] mostram que DIs tem uma axiomatização tão simples como as DFs. Contudo, se considerarmos DFs e DIs juntas, não é possível obter uma axiomática completa, em que cada regra é K-ária para algum K fixo. Isto é provado em [CFP]. Um sistema consistente e completo para dependências funcionais e dependências de inclusão sobre expressões relacionais é apresentado em [Re]. Este sistema escapa dos resultados limitativos citados em [CFP] baseando-se em uma linguagem formal que inclui, além das DFs e DIs, outras fórmulas do tipo a = b e c(a). Sistemas como este podem ser facilmente estendidos para outros tipos de dependências.

Na seção 2 descrevemos os conceitos básicos do modelo relacional e introduzimos as noções de dependências funcionais e de inclusão. Na seção 3 discutimos o projeto de esquemas no modelo relacional. Na seção 4 discutimos brevemente o problema de restrições de integridade em subesquemas. Finalmente na seção 5 apresentamos conclusões e sugestões para trabalho futuro.

2. Conceitos Básicos do Modelo Relacional

O modelo relacional para bancos de dados apoia-se no conceito matemático de **relação**, que é um subconjunto do produto cartesiano de uma lista de domínios. Um domínio é simplesmente um conjunto de valores. Os membros de uma relação são chamados **tuplas**. Se uma relação é um subconjunto do produto cartesiano de k domínios, então diz-se que ela tem **aridade** k. Uma tupla $(a_1, a_2, \ldots, a_k)$ tem k **componentes**. O i-ésimo componente dessa tupla é a_i. Uma relação pode ser visualizada como uma tabela de dados onde cada linha é uma tupla e cada coluna corresponde a um componente da tupla. As colunas freqüentemente recebem o nome de **atributos** de relação.

Uma descrição de banco de dados relacional é chamada de um **esquema**, o qual contém a descrição de cada relação do banco de dados, além de sentenças, chamadas **restrições de integridade**, capturando a semântica da aplicação. Um **esquema de relação** consiste de um nome de relação seguido da lista do nome de seus atributos. A aridade de um esquema de relação é igual ao número de seus atributos.

Um **estado** de um banco de dados é uma função que associa a cada esquema de relação do banco de dados uma relação da mesma aridade. Um estado que satisfaz a todas as restrições de integridade do esquema é chamado de **estado consistente**.

Dois casos importantes de restrições de integridade são as dependências funcionais e dependências de inclusão.

Intuitivamente, uma **dependência funcional** (abrev. DF) f:X → Y (lê-se, X determina Y) é válida em uma relação R com atributos X, Y se e somente se, não existem duas tuplas em R que coincidem no valor de X, mas diferem no valor de Y, ou seja, cada valor de X em R está associado a um único valor de Y.

As dependências funcionais não são deriváveis a partir dos dados, apesar de que o exame dos dados pode sugerir dependências em potencial. Elas são declaradas a partir do conhecimento que se tem sobre o mundo que está sendo modelado pelo banco de dados.

Na prática, muitas vezes encontramos situações em que é desejável estabelecer restrições entre relações. Seja o caso do esquema de relação apresentado a seguir.

Exemplo 2.1:

COMP[ORDEM-DE-FORNECIMENTO, CONTRATO, FORNECEDOR]

(a) COMP: ORDEM-DE-FORNECIMENTO → CONTRATO
(b) COMP: CONTRATO → FORNECEDOR

Ao ser passado para a terceira forma normal, teremos outro esquema expresso por:

OF[ORDEM-FORNECIMENTO, CONTRATO]
CT[CONTRATO, FORNECEDOR]

(c) OF: ORDEM-DE-FORNECIMENTO → CONTRATO
(d) CT: CONTRATO → FORNECEDOR

Entretanto, embora não esteja representado, temos que dizer que em um determinado estado de banco de dados

OF[CONTRATO] = CT [CONTRATO]

isto é, as ocorrências de contrato em OF e CT devem ser as mesmas. Trata-se portanto, de uma restrição que envolve duas relações diferentes. Assim, no esquema normalizado proposto, como não estamos admitindo a suposição da relação universal [FMU, Kel], deveremos incluir a restrição mencionada, a qual pode ser subdividida nas duas seguintes restrições.

(e) OF[CONTRATO] ⊂ CT[CONTRATO]
(f) CT[CONTRATO] ⊂ OF[CONTRATO]

Por (e) queremos dizer que toda ocorrência de contrato na relação OF deve ser também uma ocorrência de contrato na relação CT. Este tipo de restrição é chamado de dependência de inclusão. De modo geral, uma **dependência de inclusão** (abrev. DI) nos diz que a projeção sobre m colunas dadas de uma relação R são um subconjunto da projeção de m colunas dadas da relação S. As dependências de inclusão são um caso especial das dependências implicacionais embebidas estendidas [Fa5]. A restrição (e) nos impede de incluirmos uma ordem de fornecimento na relação OF caso o contrato correspondente não exista na relação CT.

O caso da restrição (f) é análogo ao da restrição (e). Para incluirmos um contrato na relação CT, deveremos incluir pelo menos uma ordem de fornecimento relativa a este contrato, pois não poderemos ter um contrato que não tenha nenhuma ordem de fornecimento.

Generalizando, podemos ter restrições de integridade sobre expressões da álgebra relacional, ao invés de tê-las apenas sobre relações. Usaremos então a notação E:X → Y, onde E é uma expressão relacional, para indicar que Y é funcionalmente dependente de X na relação denotada por E; similarmente, usaremos a notação E ⊂ F, onde E e F são expressões relacionais denotando relações de mesma aridade, para indicar que a relação denotada por E é um subconjunto da relação denotada por F.

Exemplo 2.2:

Sejam os seguintes esquemas de relação:

(1) ALUNO[NOME-ALUNO, DEPARTAMENTO]
(2) DEPTO[DEPARTAMENTO, NOME-DIRETOR]

Com as seguintes restrições:

(3) ALUNO: NOME-ALUNO → DEPARTAMENTO
(4) DEPTO: DEPARTAMENTO → NOME-DIRETOR

Podemos então inferir que na relação denotada pela expressão relacional ALUNO *DEPT a seguinte restrição é válida

(5) ALUNO *DEPTO: NOME-ALUNO → NOME-DIRETOR

Além disto temos, por definição de junção natural que

(6) ALUNO *DEPTO [NOME-ALUNO, DEPARTAMENTO] ⊂ ALUNO[NOME-ALUNO, DEPARTAMENTO]

(7) ALUNO *DEPTO [DEPARTAMENTO, NOME-DIRETOR] ⊂ DEPTO [DEPARTAMENTO, NOME-DIRETOR]

3. Projeto de Esquemas no Modelo Relacional

Para permitir que os usuários possam ver representados os objetos existentes em sua visão de mundo, os proponentes de modelos de banco de dados, de um modo geral, utilizam conceitos como entidades, atributos, relacionamentos, etc.. Antes de abordarmos o projeto de esquemas no modelo relacional, apresentamos alguns desses conceitos, de maneira informal e portanto imprecisa em alguns detalhes [∪].

Uma **entidade** é um objeto concreto ou abstrato, o qual possui características que o tornam distinguível. É algo que tem existência por si mesmo. Por exemplo, cada fornecedor é uma entidade, cada aluno e cada curso também o são.

Um grupo de entidades do mesmo tipo forma uma **classe ou conjunto de entidades**. Exemplos de classes de entidades: todos os fornecedores, todos os alunos, todos os departamentos, todos os empregados.

As entidades possuem propriedades. Estas propriedades. Estas propriedades recebem a denominação de **atributos**. A cada atributo de uma entidade é associado um **domínio** de valores. Esses domínios podem ser um conjunto de números inteiros, números reais, cadeias de caracteres ou qualquer outro tipo de valores. Por exemplo, as entidades na classe de entidade fornecedor podem ser atributos como razão social (uma cadeia de caracteres), capital social (um número real), etc. . . .

Um atributo ou conjunto de atributos cujos valores identifiquem de maneira única cada entidade em uma classe de entidade é chamado uma **chave** para aquela classe de entidade. Como uma entidade deve ser distingüível, toda entidade deverá possuir uma chave. Por exemplo, a classe de entidade aluno, poderia usar como chave o número de matrícula do aluno. A classe de entidade órgão de uma empresa, poderia usar como chave, a sigla do órgão.

Um **relacionamento** entre classes de entidades é simplesmente uma lista ordenada de classes de entidades. Uma determinada classe de entidades pode aparecer mais de uma vez na lista. O caso mais comum é o relacionamento entre duas entidades, porém, existem casos de três, quatro e até mais classes de entidades interelacionadas.

Passamos agora a descrever um exemplo em que, a partir de um modelo entidade-relacionamento [Ch], chegamos a um modelo relacional que pode não estar normalizado, e fazendo uso de DFs e DIs, chegamos a um modelo relacional normalizado.

A escolha do modelo entidade-relacionamento para ponto de partida deve-se ao fato de este modelo ser de fácil utilização além de expressar bem a semântica da aplicação. Constitui-se num modelo informal que faz a modelagem de situações do mundo real. Possui uma representação gráfica para exibir entidades e relacionamentos, chamada diagrama de entidade-relacionamento, o qual tem a forma da figura 3.1. Nesta figura, poderão ser encontradas várias características deste tipo de diagrama, sobre a qual fizemos algumas adaptações em relação ao tipo de diagrama originalmente proposto por Chen em [Ch].

(A) Diagrama Entidade-Relacionamento

1. Os retângulos representam classes de entidades. Assim ITEM-DE-MATERIAL é uma classe de entidade.

2. Os losangos representam classes de relacionamentos. Uma classe de relacionamento pode ser definida sobre uma, duas ou mais classes de entidades. Por exemplo, a classe de relacionamento FORNECE é definida sobre três classes de entidades: FORNECEDOR DE MATERIAL, OBRA e ITEM-DE-MATERIAL.

3. O diagrama também indica a funcionalidade. Ela informa quantas entidades de uma classe de entidade podem ser associadas com quantas entidades de uma outra classe de entidade. A forma de relacionamento mais simples é 1:1. Um exemplo desse caso é o relacionamento entre FORNECEDOR e FORNECEDOR-DE-MATERIAL que é um relacionamento **é-um**. Outro exemplo é o caso da classe de relacionamento CASAMENTO na classe de entidade PESSOAS. Um relacionamento 1:1 também pode ocorrer entre a classe de entidade GERENTE e a classe de entidade DEPARTAMENTO em uma empresa. A classificação do relacionamento quanto a funcionalidade (1:1, 1:n ou m:n) é uma suposição que o projetista faz a partir de seu conhecimento do mundo real. A classe de relacionamento RECEBE entre FORNECEDOR DE MATERIAL e ENCOMENDA é do tipo 1: n, isto é, um fornecedor de material pode receber n (n = 0, 1, 2,. . .) encomendas e cada encomenda pertence a um único fornecedor de material.

4. O diagrama expressa ainda a **dependência de existência** de uma entidade em relação a outra. Dois tipos de dependências de existência podem ocorrer. No primeiro caso, um atributo de uma entidade é a chave da entidade da qual ele depende. Este é o caso de uma entidade ENCOMENDA, a qual, para existir, ou para ser incluída depende da existência da entidade FORNECEDOR-DE-MATERIAL correspondente. Expressamos esta situação no diagrama, colocando um asterisco na ponta do arco que liga FORNECEDOR-DE-MATERIAL a ENCOMENDA. O segundo caso de dependência de existência aparece quando temos uma **classe de entidade fraca**, que definiremos como sendo

aquela que não possui atributos que sejam capazes de identificá-la. Sua identificação é feita através de seu relacionamento com outra entidade. No diagrama identificamos uma entidade fraca com um retângulo de linha dupla. No caso de uma entidade da classe ITEM-DE-ENCOMENDA, além de sua existência estar condicionada a existência da entidade ENCOMENDA correspondente, isto é, se não existe encomenda, não há como haver a existência de seus ítens, para identificar ITEM-DE-ENCOMENDA precisamos de seu relacionamento CONTEM com ENCOMENDA. Esta situação é expressa no diagrama colocando-se no arco que liga as duas entidades, um círculo na ponta do arco mais próxima da entidade fraca.

5. No caso de um relacionamento **é-um**, para evitar ambiguidades, usamos uma seta para indicar a direção do relacionamento.

(B) Mapeamento do Modelo Entidade-Relacionamento para o Modelo Relacional

O mapeamento do Modelo ER (entidade-relacionamento) para o modelo relacional vai resultar em esquemas de relações, dependências funcionais e dependências de inclusão. Assim podemos considerar os seguintes casos.

1. Uma classe de entidade pode ser representada por uma relação cujo esquema de relação consiste de todos os atributos da classe de entidade, conforme a figura 3.2. Sua chave implicará na existência de uma dependência funcional. Seus relacionamentos com outras entidades darão origem a dependências de inclusão.

2. Um relacionamento entre classes de entidades pode também dar origem a uma relação cujo esquema de relação é composto dos atributos do relacionamento representado.

Exemplo 3.1:

Vamos estudar o banco de dados apresentado pelo diagrama entidade-relacionamento da figura 3.1. Trata-se de um banco de dados que registra informações referentes a compra de materiais necessários a obras em uma empresa que administra projetos de engenharia. Partiremos do "mundo real" representado pelo diagrama e chegaremos a um esquema de banco de dados relacional onde teremos apenas relações e restrições de integridade do tipo de DF e DI. Embora o diagrama não esteja apresentando os atributos das entidades e dos relacionamentos, apresentamos a seguir uma ilustração de como se processa o mapeamento das entidades e relacionamentos para o modelo relacional.

1. A classe de entidade ENCOMENDA possui os seguintes atributos:

NUM-ENCO / número da encomenda /
DATA-EMISSÃO / data de emissão da encomenda /
MOEDA / em que moeda a encomenda será paga /
TIPO-ENTREGA / por exemplo, CIF, FOB, FOT, etc./
COD-FORN / código do fornecedor /

Esta classe de entidade possui dois relacionamentos. O relacionamento RECEBE pelo qual a classe de entidade ENCOMENDA possui uma dependência de existência com o FORNECEDOR-DE-MATERIAL, ou seja, toda encomenda, para existir, deve ter um fornecedor-de-material correspondente. Por outro lado, pelo relacionamento CONTEM, a classe de entidade ENCOMENDA tem também uma dependência de existência em relação a classe de entidade ITEM-DE-ENCOMENDA, o que quer dizer que, para que uma encomenda possa existir, ela deverá ter pelo menos uma ocorrência da classe de entidade ITEM-DE-ENCOMENDA correspondente. Assim, ao mapearmos a classe de entidade ENCOMENDA para o modelo relacional, obteremos o seguinte esquema de relação:

ENCO [NUM-ENCO, DATA-EMISSÃO, MOEDA, TIPO-ENTREGA, COD-FORN]

com as seguintes restrições de integridade

ENCO : NUM-ENCO → DATA-EMISSÃO, MOEDA, TIPO-ENTREGA, COD-FORN
ENCO [NUM-ENCO] ⊂ ITEMENCO[NUM-ENCO]
ENCO [COD-FORN] ⊂ FORNMAT [COD-FORN]

2. A classe de relacionamento FORNECE possui os atributos chaves de OBRA, FORNECEDOR-DE-MATERIAL e ITEM-DE-MATERIAL e ainda o atributo QUANT-FORN que indica qual a quantidade de um determinado item de material fornecido para uma obra por um fornecedor de material. Evidentemente, teremos a dependência de existência para este relacionamento expressa por três dependências de inclusão. A chave de FORNECE é composta de COD-FORN, COD-OBRA e COD-MAT. Assim, ao mapearmos a classe de relacionamento FORNECE para o modelo relacional, teremos o seguinte esquema de relação:

FORNECE [COD-FORN, COD-OBRA, COD-MAT, QUANT-FORN]

e as seguintes restrição de integridade

FORNECE : COD-FORN, COD-OBRA, COD-MAT → QUANT-FORN

FORNECE [COD-FORN] ⊂ FORNMAT[COD-FORN]

FORNECE [COD-MAT] ⊂ OBRA [COD-OBRA]

FORNECE [COD-MAT] ⊂ ITEMAT[COD-MAT].

Para o caso das outras entidades e relacionamentos, o procedimento é análogo aos já vistos. Desta forma, obtemos o esquema de banco de dados relacional para esta aplicação, o qual é composto dos seguintes esquemas de relações:

(1) FORN [COD-FORN, NOME-REF, RAZÃO-SOCIAL, CGC, ENDER]
(2) FORNMAT [COD-FORN]
(3) ITEMAT [COD-MAT, DESC-RES, DESC-COMP]
(4) OBRA [COD-OBRA, NOME-OBRA, CHEFE-OBRA]
(5) ENCO [NUM-ENCO, DATA-EMISSÃO, MOEDA, TIPO-ENTREGA, COD-FORN]

(6) ITEMENCO [NUM-ENCO, NUM-ITEM, DESC-ITEM, UF, PREÇO, COD-MAT, QUANT-ITEM]
(7) FORNECE [COD-FORN, COD-OBRA, COD-MAT, QUANT-FORN]

O esquema possui ainda as seguintes restrições de integridade:

(8) FORN : COD-FORN → NOME-REF, RAZÃO-SOCIAL, CGC, ENDER
(9) ITEMAT : COD-MAT → DESC-RES, DESC-COMP
(10) OBRA : COD-OBRA → NOME-OBRA, CHEFE-OBRA
(11) ENCO : NUM-ENCO → DATA-EMISSÃO, MOEDA, TIPO-ENTREGA, COD-FORN
(12) ITEMENCO : NUM-ENCO, NUM-ITEM → DESC-ITEM, UF, PREÇO, COD-MAT, QUANT-ITEM
(13) FORNECE : COD-FORN, COD-OBRA, COD-MAT → QUANT-FORN
(14) FORNMAT [COD-FORN] ⊂FORN [COD-FORN]
(15) ENCO [COD-FORN] ⊂FORNMAT [COD-FORN]
(16) ENCO [NUM-ENCO] = ITEMENCO [NUM-ENCO]
(17) ITEMENCO [COD-MAT] ⊂ITEMAT [COD-MAT]
(18) FORNECE [COD-FORN] ⊂FORNMAT [COD-FORN]
(19) FORNECE [COD-OBRA] ⊂OBRA [COD-OBRA]
(20) FORNECE [COD-MAT] ⊂ITEMAT [COD-MAT]

O símbolo ' = ' foi usado em (16) para substituir duas dependências de inclusão, ENCO [NUM-ENCO] ⊂ ITEMENCO [NUM-ENCO] e ITEMENCO [NUM-ENCO] ⊂ ENCO [NUM-ENCO] por uma única, ou seja ENCO [NUM-ENCO] = ITEMENCO [NUM-ENCO].

4. Restrições de Integridade em Subesquemas

O principal problema que ocorre no projeto de subesquemas é o chamado **problema das restrições de integridade em subesquemas**. Este problema consiste em, dado um esquema básico e um subesquema verificar se as restrições de integridade do subesquema serão sempre satisfeitas, supondo que as restrições do esquema básico sempre serão satisfeitas.

Exemplo 4.1:

Considere o banco de dados de fornecedores-peças de [Da, cap. 4 e 9]. Pode-se descrevê-lo por um esquema com DFs e DIs, cujos parâmetros incluem dois nomes de relação ternário FORNEC e FP, um nome de relação binária CS e um nome de relação 5-ária PEÇA. Assim, teremos os seguintes esquemas de relação:

(1) FORNEC [FN, FNOME, CIDADE]
(2) CS [CIDADE, STATUS]
(3) PEÇA [PN, PNOME, COR, PESO, CIDADE]
(4) FP [FN, PN, QUANT]

O conjunto de restrições deste esquema é o seguinte:

(5) FORNEC : FN → FNOME, CIDADE
(6) CS : CIDADE → STATUS
(7) PEÇA : PN → PNOME, COR, PESO, CIDADE
(8) FP : FN, PN → QUANT
(9) FP [FN] ⊂FORNEC [FN]
10) FP [PN] ⊂PEÇA [PN]
11) FORNEC [CIDADE] ⊂CS [CIDADE]

As últimas três restrições de integridade não estão incluídas no exemplo original. Entretanto, elas refletem restrições naturais que devem estar presentes na descrição do banco de dados. Por exemplo, sem a restrição (11), FORNEC e CS poderiam ser atualizadas independentemente, permitindo a inserção de um novo fornecedor localizado em uma cidade cujo status não seja dado por CS.

Seja um subesquema que contém apenas um nome de relação quaternária R. Os valores atribuídos a R relacionam FN, FNOME, CIDADE e STATUS. Isto é indicado pelo seguinte esquema de relação:

(12) R [FN, FNOME, CIDADE, STATUS]

O mapeamento que define R é o seguinte:

(13) R = FORNEC *CS

O conjunto de restrições de integridade desse subesquema consiste da única restrição abaixo:

(14) R : FN → FNOME, CIDADE, STATUS

O problema básico neste caso seria mostrar que para qualquer estado consistente J do esquema, se I é o estado do subesquema gerado a partir de J através do mapeamento em (13), então I satisfaz à restrição em (14). Mais precisamente, suponha que J seja um estado do esquema que satisfaz às restrições (5) e (11). Construa uma relação r fazendo a junção natural das relações associadas a FORNEC e CS por J (pois R = FORNEC * CS). Então, na relação r, FN deve determinar funcionalmente FNOME, CIDADE, STATUS para que a restrição em (14) seja satisfeita.

Isto pode ser dito de forma mais concisa, dizendo que todo estado do esquema que satisfaz às restrições (5) a (11), deve satisfazer também à seguinte restrição:

(15) FORNEC * CS : FN → FNOME, CIDADE, STATUS.

Observe que a restrição em (15) é uma DF sobre uma expressão relacional, e não simplesmente uma DF definida sobre uma relação.

Para uma discussão mais precisa destes conceitos, o leitor deve consultar [Re].

5. Conclusão

Vimos que dependências funcionais e dependências de inclusão formam um conjunto rico e natural de restrições de integridade. As dependências de inclusão diferem das outras dependências estudadas em banco de dados principalmente porque podem ser interrelacionais, enquanto que as outras tratam com uma única relação de cada vez. As DIs são relevantes para o processo de projeto de esquemas pois quando uma relação R é dividida em duas, R_1 e R_2 (como, por exemplo, no processo de normalização), as DIs são capazes de expressar quais os dados de R_1 que devem ser duplicados em R_2, se necessário. Além disso, vimos que é possível expressar restrições de integridade sobre expressões relacionais e não apenas sobre relações. Isto nos permite estudar formalmente problemas como o das restrições de integridade em subesquemas.

6. Bibliografia

[ABU] A. V. Aho, C. Beeri, J. D. Ullman. "The Theory of Joins in Relational Databases", ACM TODS 4, 3 (Sept. 1979)

[Ca] M. A. Casanova – "A Theory of Data Dependencies over Relational Expressions". PUC-Technical Report DB 108104 (Oct. 1981).

[CFP] M. A. Casanova, R. Fagin and C. Papadimitriou, "Inclusion Dependencies and their Interaction with Functional Dependencies". Conf. Principles of Database Systems, Los Angeles, California (March 1982)

[Ch] P. P. S. Chen. "The Entity-Relationchip Model Toward e Unified View of Data". ACM TODS 1,1 (March 1976)

[Co1] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". Comm. ACM 13,6 (June 1970)

[Co2] E. F. Codd. "Extending the Database Relational Model to Capture More Meaning" ACM TODS 4,4 (Dec. 1979)

[Da] C. J. Date. "An Introduction to Database Systems" (2nd ed.), Addison-Wesley Pub. Co. (1977)

[Fa1] R. Fagin. "Multivalued Dependencies and a New Normal Form for Relational Databases" ACM TODS 2,3 (Sept. 1977)

[Fa5] R. Fagin. "A Normal Form for Relational Databases That is Based on Domain and Keys". ACM TODS 6,3 (Sept. 1981)

[Fa6] F. Fagin. "Horn Clauses and Database Dependencies", Proc. ACM-SIGACT Symp. Theory of Computing, 1980.

[FMU] R. Fagin, A. O. Mendelzon, J. D. Ullman - "A Simplified Universal Relation Assumption and its Properties". IBM Res. Res. RJ 2900 (Nov. 1980)

[Kel] W. Kent. "Consequences of Assuming a Universal Relation". ACM TODS 6,4 (Dec. 1981)

[Re] S. P. Rego. "Uma Teoria de Dependências Funcionais e Dependências de Inclusão sobre Expressões Relacionais". Dissertação de Mestrado - PUC/RJ (Abril 1982)

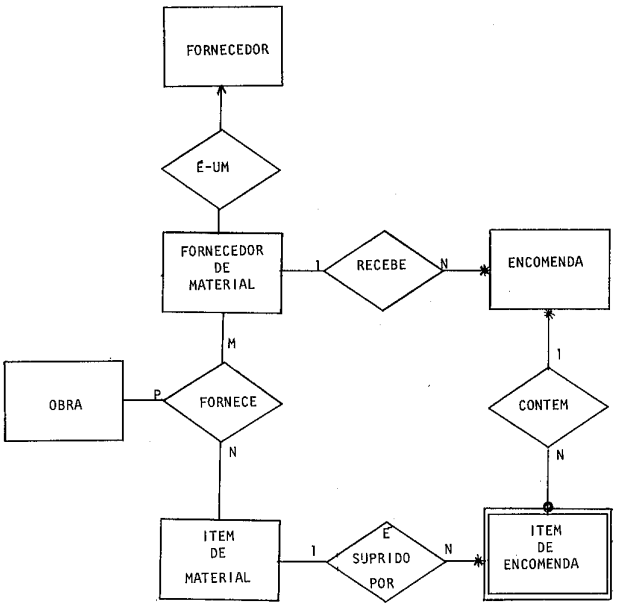
[SS] J. M. Smith, D. C. P. Smith. "Database Abstracions: Agregation and Generalization". ACM TODS 2,2 (June 1977)

[WM] G. Wiederhold, R. E1 - Masri. "A Structural Model for Database Systems". TR STAN-CS-79-722, Stanford University (Feb. 1979)

[U1] J. M. Ullman. "Principles of Database Systems". Computer Science Press (1980)

[ZM] C. Zaniolo, M. Melkanoff. "On the Design of Relational Database Schemata". ACM TODS 6,1 (March 1981).

FIGURA 3.1



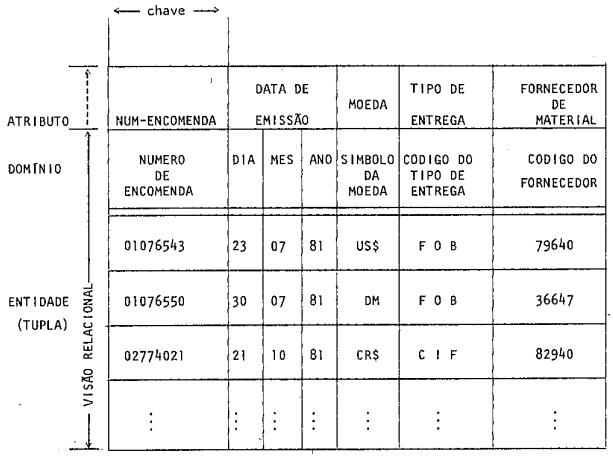


FIGURA 3.2