

M.A. CASANOVA*, L. TUCHERMAN**, A.L. FURTADO***

SUMÁRIO

Um método para projeto de banco de dados, baseado no conceito de módulos, é inicialmente descrito. Este método incorpora tanto uma estratégia para garantir restrições de integridade quanto uma tática para organizar grandes conjuntos de estruturas, restrições de integridade e operações de banco de dados. Em seguida, o protótipo de um sistema especialista que auxilia a criação de esquemas modulares de banco de dados é apresentado. Este protótipo, escrito em uma extensão de micro-PROLOG, é parte de uma ferramenta mais abrangente destinada a facilitar o desenvolvimento e manutenção de esquemas conceituais modulares.

ABSTRACT

A database design method, based on the concept of module, is first described. The method incorporates both a strategy for enforcing integrity constraints and a tactic for organizing large sets of database structures, integrity constraints and operations. A prototype expert system to help create modular database schemas is then presented. The prototype, written in an extension of micro-PROLOG, is part of a more comprehensive design tool to help develop and maintain modular conceptual schemas.

- * Doutor em Matemática Aplicada (Harvard Univ., 1979); teoria de bancos de dados, sistemas de gerência de bancos de dados distribuídos, projeto e análise de algoritmos concorrentes; Centro Científico de Brasília, IBM do Brasil - Caixa Postal 853 - Brasília, DF - CEP 70.000
- ** Mestre em Ciência da Computação (PUC/RJ, 1983); projeto de bancos de dados, sistemas de gerência de bancos de dados distribuídos, sistemas especialistas; Instituto Latino Americano de Pesquisa em Sistemas, IBM do Brasil - Caixa Postal 1830 - Rio de Janeiro, RJ - CEP 20.071
- *** Doutor em Ciência da Computação (Univ. Toronto, 1974); projeto de bancos de dados, tipos abstratos de dados, sistemas especialistas; Depto. de Informática, PUC/RJ - Rua Marquês de S. Vicente, 209 - Rio de Janeiro, RJ - CEP 22.453

INTRODUÇÃO

Discutiremos neste trabalho uma ferramenta de software que auxilia o administrador de banco de dados (ABD) a especificar e manter esquemas de banco de dados seguindo uma disciplina modular. Apenas a fase inicial da implementação da ferramenta, consistindo de um dicionário para armazenar a descrição de esquemas conceituais modulares e de facilidades para adicionar novos módulos a um esquema existente, será apresentada. A especificação completa da ferramenta é descrita em [TFC].

A ferramenta incorpora conhecimento sobre um método de projeto de banco de dados [TCF] que oferece descrições estruturadas das noções tradicionais de esquema conceitual e esquema externo. Esquemas relacionais, restrições de integridade e operações são agrupadas em módulos [Pa,LZ,ZLT] e apresentadas de maneira ordenada e estruturada, o que permite aumentar o entendimento sobre o banco de dados. O método também exige que as relações de um módulo M sejam atualizadas somente pelas operações definidas em M, o que corresponde à noção usual de encapsulamento [LZ]. Portanto, se as operações de cada módulo M preservarem consistência com relação às restrições de integridade de M, o método oferece uma maneira eficiente de garantir a consistência lógica do banco de dados. Por outro lado, neste método consultas não sofrem nenhuma restrição, da mesma forma que nos projetos tradicionais de banco de dados [TF]. Convém observar neste ponto que, embora o método esteja baseado no modelo relacional, esta não é uma opção essencial.

Projetar banco de dados modularmente não é uma estratégia nova, mas todas as referências conhecidas [DMW,EKW,LMWW,SFNC,SNF,We] tendem a explorar os princípios e a teoria de modularização. A contribuição maior deste trabalho está, portanto, na transformação desta estratégia em um método prático de projeto.

PROJETO MODULAR DE BANCOS DE DADOS

Módulos e Construtores

Um esquema de relação é uma sentença da forma $R[A_1, \dots, A_n]$, onde R é o nome do esquema e A_1, \dots, A_n são os atributos do esquema. Uma restrição de integridade é uma sentença da forma $n:Q$, onde n é o nome da restrição e Q é uma fórmula sobre os esquemas de relação em questão. Uma operação é um procedimento definido em alguma linguagem de programação adequada. Usaremos a notação $f(x_1, \dots, x_n)$ para indicar uma operação com n argumentos cujo nome é f.

Um módulo é uma tripla $M = (ER, RI, OP)$ onde

1. ER é um conjunto de esquemas de relação cujos nomes não coincidem.
2. RI é um conjunto de restrições de integridade sobre ER.
RI deverá conter, para cada esquema de relação $R[A_1, \dots, A_n]$, um axioma de esquema de relação da forma
$$\forall x_1 \dots \forall x_n (R(x_1, \dots, x_n) \Rightarrow A_1(x_1) \ \&\ \dots \ \&\ A_n(x_n)),$$
indicando que a interpretação de R deverá ser um subconjunto do produto cartesiano das interpretações de A_1, \dots, A_n .
3. OP é um conjunto de operações sobre LM.

Um módulo pode ser tanto primitivo, se for definido diretamente sem referência a outros módulos, quanto derivado, se for definido com o auxílio de outros módulos previamente definidos através de dois construtores de módulos, subjugamento e extensão.

Um módulo primitivo $M=(ER,RI,OP)$ é definido por uma declaração da forma:

(1) módulo M
 esquemas ER;
 restrições RI';
 operações OP;
 responsabilidades RE;
fim-do-módulo

onde RI' é RI sem os axiomas de esquemas de relação (como tais restrições de integridade são completamente fixadas por ER, podem ser omitidas de RI') e RE é um conjunto de cláusulas de responsabilidade da forma 'O garante I' onde O é o nome de uma operação e I é o nome de uma restrição de integridade de M.

O projetista do banco de dados deverá incluir uma cláusula de responsabilidade 'O garante I' sempre que a definição da operação O levar em consideração a restrição de integridade I, ou seja, sempre que alguma mudança na definição de I afetar a definição de O. Esta informação adicional explicita um relacionamento entre operações e restrições que seria extremamente difícil de se obter por uma análise mecânica da definição das restrições e operações.

No resto desta seção, sejam $M_i = (ER_i, RI_i, OP_i)$, $i=1, \dots, n$, módulos.

Considere agora o construtor de subjugamento. Intuitivamente, se o projetista define um módulo M como subjugando os módulos M_1, \dots, M_n , então M poderá conter novos esquemas de relação, novas restrições de integridade e novas operações. O módulo M sempre herdará todos os esquemas de relação e restrições de integridade de M_1, \dots, M_n . Além disto, M também herdará todas as operações de M_1, \dots, M_n , exceto aquelas que violarem alguma das novas restrições introduzidas em M. Tais operações não mais serão visíveis aos usuários, ou seja, não mais poderão ser chamadas diretamente pelos usuários. Os módulos M_1, \dots, M_n se tornarão então inacessíveis aos usuários e não mais poderão ser usados para definir outros módulos.

A seguinte declaração define um novo módulo M por subjugamento sobre M_1, \dots, M_n :

(2) módulo M subjuga M_1, \dots, M_n com
 esquemas ER_0 ;
 restrições RI_0 ;
 operações OP_0 ;
 responsabilidades RE;
 encobrimentos EN;
fim-de-módulo

onde:

1. ER_0 é um conjunto de esquemas de relação tal que nenhum nome em ER_0 ocorre em M_1, \dots, M_n , e nenhum dos esquemas em ER_0 tem o mesmo nome;
2. RI_0 é um conjunto de restrições de integridade sobre ER_0, ER_1, \dots, ER_n ;
3. OP_0 é um conjunto de operações podendo chamar como subrotinas apenas as operações no conjunto OP, onde OP é a união de OP_1, \dots, OP_n ;
4. EN é um conjunto, possivelmente vazio, de cláusulas de encobrimento da forma 'O pode violar C_1, \dots, C_k ', onde O é o nome de uma operação de M_i , para algum i em $[1, n]$, e C_j é o nome de uma restrição definida em RI_0 , para cada j em $[1, k]$. Dizemos neste caso que O foi encoberta em M.

5. RE é um conjunto de cláusulas de responsabilidade da forma 'O garante I' onde O é o nome de uma operação definida em M e I é um nome de uma restrição também definida em M.

Diz-se que cada objeto de M_i é importado por M.

Mais precisamente, a declaração em (2) define um módulo $M=(ER,RI,OP)$ onde

1. ER é a união de ER_0, \dots, ER_n e RI é a união de RI_0, \dots, RI_n
2. OP é a união de $OP_0, OP'_1, \dots, OP'_n$, onde OP'_1, \dots, OP'_n sem as operações encobertas em M_i , para cada $i=1, \dots, n$

Passemos agora à definição do construtor de extensão. Informalmente, um módulo M estende os módulos M_1, \dots, M_n se cada esquema de relação de M for uma visão sobre os esquemas de relação de M_1, \dots, M_n e cada restrição de integridade de M for uma consequência lógica das restrições de M_1, \dots, M_n . Além disto, para se evitar o chamado problema de atualização em visões [FC], a definição de M contém, para cada operação p em uma visão, uma implementação de p em termos das operações de M_1, \dots, M_n . Ao contrário do subjugamento, os módulos M_1, \dots, M_n permanecem acessíveis após a definição de M.

Um novo módulo M é definido por extensão sobre M_1, \dots, M_n através de uma declaração da forma:

(3) módulo M_0 estende M_1, \dots, M_n com

esquemas	ER_0 ;
restrições	RI_0 ;
operações	OP_0 ;
usando	
visões	VI;
sub-rogados	SR;
fim-do-módulo	

onde:

1. a tripla (ER_0, RI_0, OP_0) define o módulo M.
2. VI contém, para cada esquema $R[A_1, \dots, A_k]$ em ER_0 , uma definição de visão da forma $R[A_1, \dots, A_k]: Q$, onde Q é uma fórmula sobre ER_1, \dots, ER_n com k variáveis livres, ordenadas x_1, \dots, x_k . A fórmula $\forall x_1 \dots \forall x_k (R(x_1, \dots, x_k) \Leftrightarrow Q)$ é chamada de um axioma de definição de visão.
3. SR contém, para cada operação $f(y_1, \dots, y_m): r$ em OP_0 , um sub-rogado, que é uma operação da forma $f(y_1, \dots, y_m): s$ chamando apenas operações do conjunto $OP = OP_1 \cup \dots \cup OP_n$.

A declaração em (3) define então um novo módulo $M=(ER_0, RI_0, OP_0)$ e acopla M a M_1, \dots, M_n através do par (VI,SR). Uma sentença $R[A_1, \dots, A_k]: Q$ em VI indica que Q define R em termos dos esquemas de relação de M_1, \dots, M_n . Logo, uma consulta sobre R será traduzida em uma consulta sobre os esquemas de relação de M_1, \dots, M_n através de Q. Da mesma forma, um sub-rogado $f(y_1, \dots, y_m): s$ em SR descreve uma implementação de $f(y_1, \dots, y_m): r$ em termos das operações de M_1, \dots, M_n . Assim, uma chamada para f gera uma execução de s, e não de r.

Regras de Projeto para Esquemas Conceituais Modulares

Um esquema conceitual modular consiste de um conjunto de módulos que deve satisfazer a uma série de regras de projeto garantindo que, se o banco for atualizado apenas pelas operações visíveis aos usuários, o estado do banco sempre se manterá consistente. Mais precisamente, o conjunto dos esquemas conceituais modulares consistentes e o conjunto dos módulos ativos de cada esquema conceitual são definidos recursivamente como:

1. O conjunto vazio é um esquema conceitual modular consistente com um conjunto vazio de módulos ativos;
2. Seja D um esquema conceitual modular consistente e A o conjunto dos módulos ativos de D. Seja M um módulo tal que nenhum módulo em D tenha o mesmo nome que M. Então $D' = D \cup \{M\}$ é um esquema conceitual modular consistente se e somente se M satisfizer a uma das condições abaixo:
 - a. se M for um módulo primitivo, então M deverá satisfazer o Requisito 1 (a lista completa dos requisitos encontra-se na Figura 2.1 no final desta seção). O conjunto dos módulos ativos de D' será $A' = A \cup \{M\}$
 - b. se M for um módulo definido por extensão sobre M_1, \dots, M_n , então M deverá satisfazer os requisitos 2,3,4,5. O conjunto dos módulos ativos de D' será $A' = A \cup \{M\}$
 - c. Se M for um módulo definido por subjugamento sobre M_1, \dots, M_n , então:
 - 1) os nomes dos novos esquemas de relação definidos em M deverão ser diferentes dos nomes dos esquemas de relação de M_1, \dots, M_n .
 - 2) M deverá satisfazer os requisitos 6,7,8,9.
 - 3) O conjunto dos módulos ativos de D' será $A' = A \cup \{M\} - \{M_1, \dots, M_n\}$

Sempre que não qualificarmos um esquema conceitual D, assumiremos que D é consistente.

Seja D um esquema conceitual modular e A o conjunto de módulos ativos de D. O conjunto C de módulos conceituais de D é o subconjunto de A consistindo de todos os módulos primitivos e de todos os módulos ativos definidos por subjugamento; o conjunto E dos módulos externos de D é o conjunto de todos os módulos definidos por extensão em D. Uma operação p de D é ativa, conceitual ou externa se e somente se p for uma operação de um módulo ativo, conceitual ou externo de D, respectivamente.

Um usuário tem em princípio acesso a todos os módulos ativos de um esquema conceitual modular. Logo, verá todos os esquemas de relação e restrições de integridade definidos em todos os módulos, mas poderá atualizar o banco apenas através das operações ativas. Além disto, poderá consultar livremente qualquer esquema de relação. O ABD naturalmente poderá restringir os privilégios de um usuário permitindo o seu acesso a apenas um subconjunto dos módulos ativos, talvez mesmo a apenas alguns módulos definidos por extensão.

Quanto ao processo em si de projetar esquemas conceituais modulares, o método sugerido segue a definição formal. O projetista gradualmente adicionará novos módulos a um esquema modular inicialmente vazio, preocupando-se em como satisfazer as regras de projeto (veja [TFC]).

Para concluir esta seção, enunciamos um teorema que afirma ser a escolha dos requisitos suficiente para garantir preservação de consistência.

TEOREMA 2.1 [TCF]: Seja D um esquema conceitual modular. Suponha que D seja consistente, ou seja, que satisfaça aos requisitos 1 a 9. Então, toda operação ativa de D preserva consistência com respeito a todas as restrições de integridade definidas em módulos de D.

FIGURA 2.1: LISTA DE REQUISITOS

REQUISITO PARA MÓDULOS PRIMITIVOS

Requisito 1: cada operação definida em M não poderá violar nenhuma das restrições de integridade definidas em M.

REQUISITOS PARA MÓDULOS DEFINIDOS POR EXTENSÃO

Seja M um módulo definido por extensão sobre os módulos $M_i = (ER_i, RI_i, OP_i)$, $i=1, \dots, n$.

Sejam ER_0, RI_0, OP_0, VI e SR os novos esquemas de relação, restrições de integridade, operações, definições de visões e sub-rogados, respectivamente, definidos em M.

Requisito 2: se $f(y_1, \dots, y_m): s$ é o sub-rogado de $f(y_1, \dots, y_m): r$ definido em SR então s deve ser uma tradução fiel de r [FC].

Requisito 3: se $f(y_1, \dots, y_m): s$ é um sub-rogado definido em SR, então s só poderá modificar os valores dos esquemas de relação de M_1, \dots, M_n através de chamadas para as operações definidas em M_1, \dots, M_n .

Requisito 4: para cada restrição de integridade I em RI_0 , I' deve ser uma consequência lógica das restrições de integridade de M_1, \dots, M_n , onde I' é obtido de I substituindo-se cada fórmula atômica da forma $R(t_1, \dots, t_k)$ por $Q[t_1/x_1, \dots, t_k/x_k]$, onde $R[A_1, \dots, A_k]$: Q é a definição da visão R descrita em VI, e a lista de variáveis livres de Q é x_1, \dots, x_k .

Requisito 5: M_1, \dots, M_n devem ser módulos ativos de D.

REQUISITOS PARA MÓDULOS DEFINIDOS POR SUBJUGAMENTO

Seja M um módulo definido por subjugamento sobre os módulos $M_i = (ER_i, RI_i, OP_i)$, $i=1, \dots, n$.

Sejam ER_0, RI_0, OP_0, EN os novos esquemas de relação, restrições de integridade, operações e encobrimentos, respectivamente, definidos em M. Seja RI a união de RI_0, \dots, RI_n e OP a união de OP_0, OP_1, \dots, OP_n , onde OP_i é o conjunto OP_i , excetuando-se aquelas operações que foram encobertas em M, para $i=1, \dots, n$.

Requisito 6: cada operação em OP preserva consistência com respeito às restrições de integridade em RI_0 .

Requisito 7: cada operação em OP_0 só pode modificar os valores de esquemas de relação em M_1, \dots, M_n através de chamadas para as operações definidas em M_1, \dots, M_n .

Requisito 8: D não pode conter um módulo definido por extensão usando M_i , para algum i em $[1, n]$.

Requisito 9: M_1, \dots, M_n devem ser módulos conceituais de D (ou seja, módulos ativos de D que não foram definidos por extensão).

DEFINIÇÃO DO DICIONÁRIO

Nesta seção descreveremos um dicionário contendo os objetos - módulos, esquemas, restrições e operações - definidos em um esquema conceitual modular bem como os relacionamentos entre estes objetos. A descrição do dicionário captura vários conceitos introduzidos na Seção 2 e provê a base para a ferramenta de projeto descrita na Seção 4.

O esquema conceitual do dicionário será descrito em termos do modelo de entidades e relacionamentos. Embora não seja essencial, consideraremos que o dicionário contém apenas as entidades e relacionamentos derivadas de um único esquema conceitual modular D. É importante observar ainda que todas as entidades e relacionamentos e seus atributos podem ser identificados observando apenas a sintaxe dos módulos que compõem D.

O esquema conceitual é o seguinte (usaremos $B(A_1, \dots, A_n)$ para indicar tanto um conjunto de entidades com nome B e conjunto de atributos A_1, \dots, A_n , quanto para indicar um relacionamento, sem atributos e cujo nome é B, sobre os conjuntos de entidades cujos nomes são A_1, \dots, A_n):

CONJUNTOS DE ENTIDADES

é-primitivo(nome), é-sub(nome) e é-externo(nome)

a cada módulo primitivo, definido por subjugamento ou definido por extensão do esquema conceitual modular em questão, D, corresponde uma entidade no conjunto é-primitivo, é-sub ou é-externo, respectivamente. O único atributo é o nome do módulo.

módulo(nome)

generalização dos três conjuntos anteriores. O único atributo é o nome do módulo.

esquema(nome, lista, def)

a cada esquema de relação definido em um módulo de D corresponde uma entidade neste conjunto. Os atributos são o nome e a lista de atributos do esquema de relação, bem como o mapeamento que o define, se o esquema pertencer a um módulo definido por extensão, ou um valor indefinido, nil, em caso contrário.

restrição(nome, def)

a cada restrição de integridade definida em um módulo de D corresponde uma entidade neste conjunto. Os atributos são o nome e a fórmula que a define.

operação(nome, def, sub-rogado)

a cada operação definida em um módulo de D corresponde uma entidade neste conjunto. Os atributos são o nome e a rotina que define a operação, bem como o sub-rogado que a define, se a operação pertencer a um módulo definido por extensão, ou um valor indefinido, nil, em caso contrário.

RELACIONAMENTOS

subjuga(módulo, módulo) e estende(módulo, módulo)

o par (M, M') estará no relacionamento subjuga ou estende se e somente se M e M' representam dois módulos tais que M é definido por subjugamento ou por extensão, respectivamente, sobre M'.

é-esquema-definido-em(esquema, módulo)

o par (A, M) estará no relacionamento é-esquema-definido-em se e somente se A representa um esquema definido no módulo representado por M.

é-restrição.definida-em(restrição, módulo)

(mesmo, quando A é restrição definida em M).

é-operação-definida-em(operação, módulo)

(mesmo, quando A é operação definida em M). O par (O, M) estará no relacionamento.

é-visão-sobre(esquema, esquema)

o par (V,S) estará no relacionamento é-visão-sobre se e somente se V representa uma visão cujo mapeamento de definição menciona o esquema representado por S.

é-restrição-sobre(restrição, esquema)

o par (I,S) estará no relacionamento é-restrição-sobre sobre se e somente se I representa uma restrição cuja definição menciona o esquema representado por S.

é-operação-sobre(operação, esquema)

o par (O,S) estará no relacionamento é-operação-sobre se e somente se O representa uma operação cuja definição ou cujo sub-rogado (se O for uma operação definida em um módulo introduzido por extensão) mencionam o esquema representado por S.

garante(operação, restrição)

o par (O,I) estará no relacionamento garante se e somente se a definição da operação O levar em consideração a restrição representada por I.

pode-violar(operação, restrição)

O par (O,I) estará no relacionamento pode-violar se e somente se O representa uma operação cuja execução poderá violar a restrição I.

chama(operação, operação)

o par (O,O') estará no relacionamento chama se e somente se O representa uma operação cuja definição ou cujo sub-rogado (se O for uma operação definida em um módulo introduzido por extensão) mencionam a operação representada por O'.

UM SISTEMA ESPECIALISTA PARA PROJETO DE BANCO DE DADOS

Nesta seção descreveremos resumidamente o protótipo de uma ferramenta de software que auxilia o Administrador de Banco de Dados (ABD) adicionar, interativamente, módulos a um esquema de banco de dados. Este protótipo implementa parcialmente o dicionário descrito na Seção 3.

O protótipo é um exemplo de um assistente especialista cujo conceito foi apresentado em [FM] para designar ferramentas inteligentes e relativamente pequenas que auxiliam no projeto, uso e manutenção de grandes sistemas convencionais. A versão corrente desta ferramenta roda em computador da família do IBM PC e foi escrita usando o apes [HS] que é uma extensão do micro-PROLOG [CM].

Representaremos parte do dicionário através de uma série de axiomas, que serão literais positivos sobre um único símbolo predicativo binário, 'tab'. Cada axioma corresponde ou a um relacionamento ou a uma entidade designando módulos de um esquema de banco de dados. Um axioma correspondente a um relacionamento consiste, intuitivamente, de duas listas: a primeira denota o tipo de relacionamento, expresso pelos tipos de entidades envolvidas, e a segunda indica o relacionamento em si, expresso pelos nomes das entidades. O formato geral deste tipo de axioma é (usando notação infixa para 'tab'):

(<tipo> <tipo>) tab (<nome> <nome> <versão>)

onde <versão> é um inteiro positivo (sempre 1, quando o dicionário é criado).

Um axioma representando um módulo tem o seguinte formato:

(mod) tab (<nome> <tipo> <versão>)

onde <tipo> pertence ao conjunto primitivo, subjugamento, extensão).

Na tabela 4.1 apresentamos a correspondência entre as entradas do dicionário e suas representações como axiomas, conforme foi implementado pela ferramenta.

Tabela 4.1 - Representação do Dicionário

Entrada no Dicionário	Formato do Axioma
<u>é-primitivo</u> (M)	(mod) tab (M 'primitivo' n)
<u>é-sub</u> (M)	(mod) tab (M 'subjugamento' n)
<u>é-externo</u> (M)	(mod) tab (M 'extensão' n)
<u>esquema</u> (S,L,Q)	não implementado
<u>restrição</u> (I,Q)	não implementado
<u>operação</u> (O,P,P')	não implementado
M <u>subjuga</u> M'	(mod mod) tab (M M' n)
M <u>estende</u> M'	(mod mod) tab (M M' n)
S <u>é-esquema-definido-em</u> M	(sch mod) tab (S M n)
I <u>é-restrição-definida-em</u> M	(con mod) tab (I M n)
O <u>é-operação-definida-em</u> M	(ope mod) tab (O M n)
V <u>é-visão-sobre</u> S	(sch sch) tab (V S n)
I <u>é-restrição-sobre</u> S	(con sch) tab (I S n)
O <u>é-operação-sobre</u> S	(ope sch) tab (O S n)
O <u>garante</u> I	(ope con) tab (O I n)
O <u>pode-violar</u> I	(ope con) tab (hid O) I n)
O <u>chama</u> O'	(ope ope) tab (O O' n)

Nota: n é o número da versão

Graças ao uso de apes, o protótipo é altamente interativo. Esboçaremos a seguir como ele pode ser usado pelo ABD para adicionar um novo módulo a um esquema de banco de dados.

Para iniciar a definição de um módulo, o ABD tecla module <nome>. A partir deste ponto, o protótipo induz o ABD a fornecer todas as informações necessárias à definição dos esquemas, das restrições e das operações do módulo. O "programa" consiste de um predicado chamado "module" que por sua vez chama outros predicados para criarem os vários componentes do módulo. Um módulo particular pode ou não ter esquemas, restrições ou operações. Entretanto:

- o se o módulo M não é primitivo, o ABD deve listar os módulos que M subjuga ou estende;
- o se o módulo M é definido por extensão, cada esquema S é uma visão. Assim, o ABD deve definir o mapeamento de S sobre os esquemas dos módulos estendidos por M;
- o para cada restrição ou operação O, o ABD deve listar todos os esquemas que O referencia;
- o somente operações de módulos não-primitivos podem chamar outras operações; além disso, todas as operações dos módulos criados por extensão são sub-rogados e devem, portanto, incluir tais chamadas. O ABD deve então informar os relacionamentos do tipo chama.

Assim, a presença de certos relacionamentos (indicados pela adição das entradas correspondentes em tab) é compulsório, e o predicado "module" falhará se o ABD declarar que eles não existem (teclando "end" em resposta a uma pergunta apresentada pelo protótipo).

O protótipo determina, de forma procedural, a seqüência a ser seguida pelo ABD na criação dos vários relacionamentos e a natureza compulsória ou opcional destes. Por outro lado, usando as facilidades de 'unique-answer' e 'valid-answer' do apes, o protótipo define separadamente, de forma declarativa, que critérios decidem se os valores fornecidos pelo ABD, como respostas, são aceitáveis.

Apresentamos abaixo, por tipo de relacionamento criado, os critérios que são atualmente garantidos.

(mod) tab (x y 1)	y ∈ {primitivo, subjugamento, extensão}
(mod mod) tab (x y 1)	y é um módulo ativo, que não deve ter sido criado por extensão nem estendido se x está sendo criado por subjugamento
(sch sch) tab (x y 1)	o esquema y é acessível para algum módulo usado na definição do módulo no qual a visão x está sendo definida
(con sch) tab (x y 1)	o esquema y é acessível para algum módulo no qual a restrição x está sendo definida
(ope ope) tab (x y 1)	a operação y é acessível para algum módulo usado na definição do módulo no qual a operação x está sendo definida; se esta última é definida por extensão, y é relacionado a algum esquema que suporta sua visão
(ope sch) tab (x y 1)	o esquema y é acessível para o módulo onde a operação x está sendo definida
(ope con) tab (x y 1)	a operação x e a restrição y tem algum esquema em comum
O mesmo, para x= (hid z)	a operação z é chamada por uma operação da qual a restrição y depende.

O protótipo formula as perguntas relevantes ao ABD através de sentenças em linguagem natural, e adota menus estáticos e dinâmicos para restringir as suas respostas. O protótipo assegura que os nomes no dicionário são únicos. Para estes propósitos são usadas algumas facilidades adicionais do apes ('which-template', 'is-menu', 'is-template').

Se olharmos para a Figura 2.1 no final da Seção 2, podemos comparar os critérios implementados com os requisitos necessários para um projeto modular correto. Os Requisitos 1, 2, 4, 6 e 7 não são garantidos por requererem uma descrição detalhada dos componentes. Os Requisitos 5, 8 e 9 são explicitamente garantidos pelos critérios implementados. O Requisito 3, que se refere a módulos criados por extensão, é garantido através de restrições sobre operações e visões do módulo em relação a operações e esquemas dos módulos que foram estendidos.

De uma forma geral, podemos certamente fazer mais em termos de verificações na consistência do projeto modular usando as informações que são extraídas do ABD. Entretanto, o que verificamos agora é suficiente para demonstrar a utilidade deste tipo de assistente especialista.

CONCLUSÕES E DIREÇÕES FUTURAS DE PESQUISAS

Descrevemos neste trabalho uma ferramenta de software orientada para o método de projeto modular de banco de dados apresentada primeiramente em [TCF]. O método em si foi aumentado pela incorporação das cláusulas de encobrimento e responsabilidade, e pelo refinamento de algumas regras de projeto.

A ferramenta de software está implementada até o ponto de auxiliar o administrador de banco de dados a adicionar novos módulos a um esquema conceitual existente. O processo de reprojeto, embora não implementado, está especificado em detalhe em [TFC].

Os planos futuros incluem transformar a ferramenta em um sistema de dicionário completo, incorporando o máximo possível de conhecimento sobre a metodologia de projeto.

REFERÊNCIAS

- [CM] K.L. Clark e F.G. McCabe. "micro-PROLOG: programming in logic". Prentice-Hall (1984)
- [DMW] W. Dosch, G. Mascari, M. Wirsing. "On the Algebraic Specification of Databases". Proc. 8th Int'L Conf. on Very Large Data Bases (1982)
- [EKW] H. Ehrig, H.-J. Kreowski, H. Weber. "Algebraic Specification Schemes for Data Base Systems". Proc. 4th Int'L Conf. on Very Large Data Bases (1978)
- [FC] A.L. Furtado e M.A. Casanova. "Updating Relational Views", in "Query Processing in Database Systems", Springer Verlag (no prelo)
- [FM] A.L. Furtado and C.M.O. Moura. "Expert helpers to data-based information systems". Proc. of the First International Workshop on Expert Database Systems (1984), 298-313
- [HS] P. Hammond and M. Sergot. "apes: augmented PROLOG for expert systems - reference manual". Logic Based Systems Ltd. (1984)
- [LMW] P.C. Lockemann, H.C. Mayr, W.H. Weil, W.H. Wohlleber. "Data Abstractions for Data Base Systems". ACM Transactions on Database Systems 4:1 (1979)
- [LZ] B. Liskov, S. Zilles. "Specification Techniques for Data Abstractions". IEEE Transactions on Software Engineering SE-1 (1975)
- [Pa] D. Parnas. "On the Criteria to be Used in Decomposing Systems into Modules". Comm. of the ACM 15:12 (1972)
- [SFNC] U. Schiel, A.L. Furtado, E.J. Neuhold, M.A. Casanova. "Towards Multi-level and Modular Conceptual Schema Specifications". Inform. Systems 9:1 (1984), 43-57.
- [SNF] C.S. dos Santos, E.J. Neuhold, A.L. Furtado. "A Data Type Approach to the Entity-Relationship Model". Int'L. Conf. of the Entity-Relationship Approach to Systems Analysis and Design (1980)
- [TCF] L. Tucherman, M.A. Casanova e A.L. Furtado, "A Pragmatic Approach to Modular Database Design", Proc. of the 9th Int'L Conf. on Very Large Data Bases, Florence, Italy (1983), 219-231.
- [TFC] L. Tucherman, A.L. Furtado e M.A. Casanova, "A Tool for Modular Database Design", Relatório Técnico do CCB (Centro Científico de Brasília) a ser publicado.
- [TF] T.J. Teorey, J.P. Fry. "Design of Database Structures", Prentice-Hall, Inc. (1982)
- [We] H. Weber. "Modularity in Data Base Systems Design". Proc. Joint IBM/Univ. Newcastle upon Tyne Seminar (1979)
- [ZLT] S.N. Zilles, P. Lucas, J.W. Thatcher. "A Look at Algebraic Specifications". Research Rep. RJ3568 IBM Thomas J. Watson Research Center (1982)