Anais do XIII Seminário Integrado de Software e Hardware, Recife, 1986, 317 - 330

LBF: UMA LINGUAGEM PARA BANCOS DE DADOS FUNCIONAIS

José de Jesús Pérez Alcázar (1)
Alberto Henrique Frade Laender (2)
Roberto da Silva Bigonha (3)

SUMARIO

Este artigo apresenta uma descrição das características principais da linguagem LBF (Linguagem para Bancos de Dados Funcionais) que é uma linguagem auto-contida para a manipulação de bancos de dados funcionais. LBF é uma extensão da linguagem DAPLEX definida por Shipman. Estas extensões incluem comandos de entrada e saída, comandos condicionais, operadores aritméticos e lógicos e comandos para modificação do esquema.

ABSTRACT

This paper presents an overview of LBF, which is an special purpose self-contained language for manipulation of functional data bases. LBF is an improved version of Shipman's DAPLEX, which has been extended to include I/O and conditional statements, logical and arithmetic operations, and operations for schema modification.

(1) Engenheiro de Sistemas e Computação (Univ. de los Andes, Colômbia, 1983), Mestrando em Ciência da Computação (UFMG). Areas de Interesse: Banco de Dados, Linguagens e Compiladores.

(2) Engenheiro Eletricista (UFMG, 1974), Mestre em Ciência da Computação (UFMG 1979), Ph.D. em Ciência da Computação (Universidade de East Aglia, 1984). Areas de Interesse: Banco de Dados e Interfaces para Usuários Finais.

(3) Doutor em Ciência da Computação (UCLA, 1981). Professor Adjunto do DCC UFMG. Areas de Interesse: Linguagens e Compiladores.

Endereço dos autores: Departamento de Ciência da Computação, ICEX-UFMG, CP 702, 30161, Belo Horizonte, MG.

1. INTRODUÇÃO.

Nos últimos anos, vários modelos de dados têm sido propostos com o propósito de melhor capturar a semântica dos dados e remover várias das restrições impostas pelos chamados modelos clássicos (hierárquico, de rede e relacional). Estes modelos, denominados modelos semânticos [3,14], foram projetados para fornecer mecanismos de modelagem mais expressivos e ricos, permitindo descrever a estrutura de um banco de dados de forma mais clara e precisa [2,6,7,17,20].

Dentre os vários modelos semânticos descritos na literatura destaca-se, por sua simplicidade, o modelo funcional [5,8,11,18,25]. Das várias propostas para o modelo funcional, algumas incorporam linguagens de manipulação de dados usando funções e conjuntos de operadores. Destas, somente as propostas de Shipman [18] e Buneman & Frankel [5] integram operações de manipulação de dados com operações de propósito geral numa só linguagem. Este é um aspecto de grande importância já que muitos dos modelos de dados semânticos propostos na literatura se prestam mais à modelagem das propriedades estáticas das aplicações, dando pouca atenção aos aspectos dinâmicos (operações) [14].

Certamente, o modelo de Shipman [18] é o mais destacado dos modelos funcionais. Ele leva em consideração vários resultados recentes de pesquisa na área de modelagem de dados, o que faz de DAPLEX, sua linguagem de definição e manipulação de dados, uma linguagem bastante rica semanticamente.

O propósito deste artigo é descrever uma linguagem do tipo DAPLEX que está sendo desenvolvida no Departamento de Ciência da Computação da UFMG. Esta linguagem, denominada LBF (Linguagem para Bancos de Dados Funcionais), é uma extensão de DAPLEX, na qual foram incluídas algumas construções propostas por Kulkarni e utilizadas em EFDM [12], bem como facilidades para operações de E/S e comandos condicionais.

O restante do artigo se divide em quatro seções. Na Seção 2 descrevemos o modelo de dados funcional e os motivos de sua escolha para o desenvolvimento deste trabalho. Nesta Seção apresentamos também uma descrição das características gerais da proposta de Shipman. Na Seção 3 descrevemos a LBF, suas estruturas e operações, e apresentamos um pequeno exemplo de sua utilização. Finalmente, na Seção 4 descrevemos o estado atual do projeto e apresentamos algumas diretrizes para pesquisas futuras.

2. O MODELO DE DADOS FUNCIONAL

O modelo funcional nasce da busca de novas formas de modelagem a

serem utilizadas como base numa futura descrição e comparação dos três modelos clássicos [18]. Este formalismo deveria fornecer uma base unificada para o projeto de esquemas conceituais relacionais quanto de rede, que fossem livres de anomalias de atualização.

O modelo funcional tem sua origem no conceito matemático de função e utiliza dois tipos de primitiva (estrutural) para a modelagem do mundo real [12]: os conjuntos de objetos e as funções que são aplicadas a estes objetos relacionando-os entre si. Os conjuntos de objetos estão divididos em dois tipos de entidades e conjuntos de valores. A principal diferença entre um conjunto de valores e um conjunto de entidades é que o primeiro nunca faz parte do domínio de uma função. Os conjuntos de valores são chamados por Kulkarni [12] conjuntos de entidades pré-definidas (ex: conjuntos de inteiros, reais, etc). As funções podem ser totais ou parciais, e a distinção corresponde a uma restrição [18] sobre o relacionamento existente do conjunto (domínio) que devem participar no relacionamento funcional.

A figura 1 apresenta a descrição gráfica, de acordo com o modelo funcional, de um esquema conceitual de um banco de dados de uma biblioteca pessoal de artigos publicados em periódicos. Este banco de dados será usado daqui para frente como base para os exemplos.

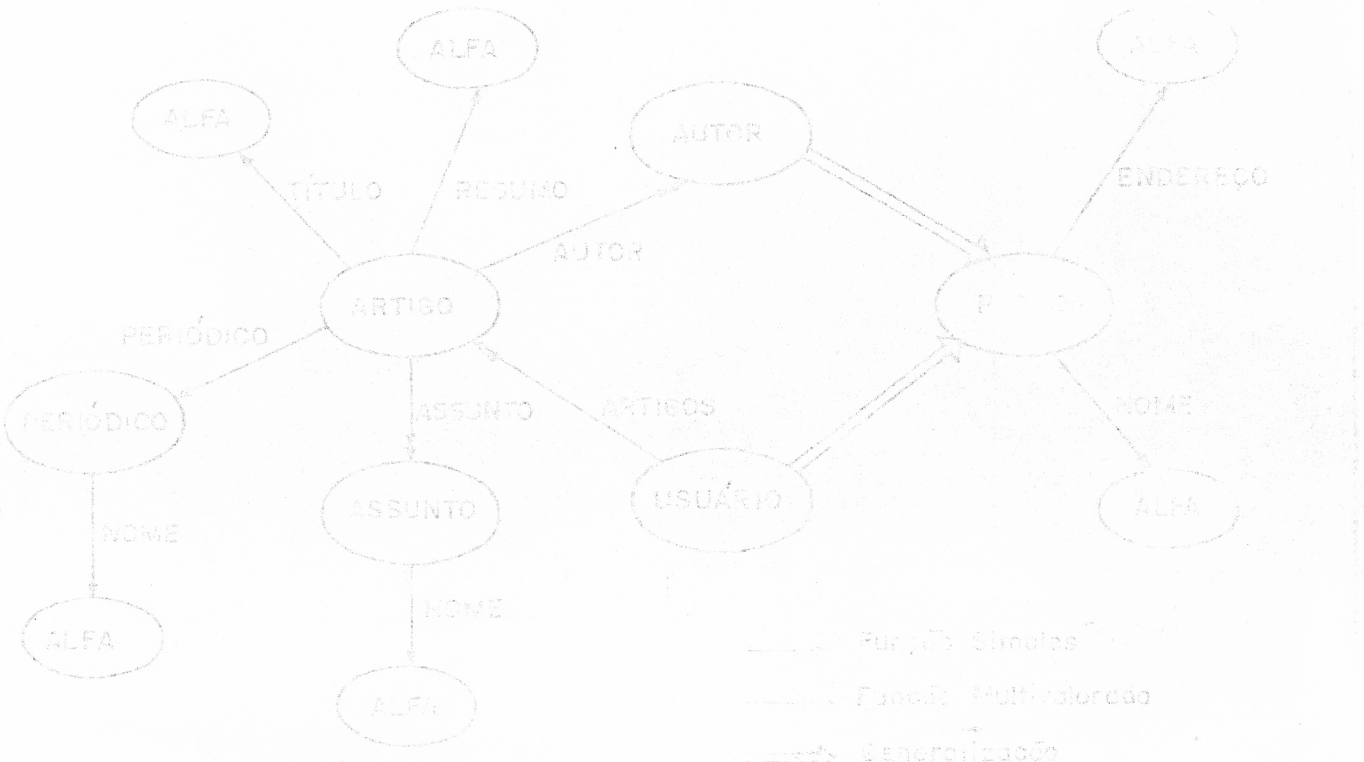

Figura 1

2.1. O Modelo de Dados Funcional de Shipman

Como já mencionamos, o modelo funcional mais destacado é o modelo proposto por Shipman [18], o qual está embutido na linguagem DAPLEX. Seu objetivo fundamental é fornecer uma linguagem conceitualmente natural que sirva de interface na utilização de um banco de dados. As construções básicas da linguagem DAPLEX são as entidades e as funções, que são usadas para modelar os objetos da aplicação e as suas propriedades respectivamente. O aspecto mais interessante da linguagem DAPLEX é a sua simplicidade aliada a construções de modelagem semanticamente poderosas.

Entre as caracteristicas principais da lingugem DAPLEX podemos enumerar as seguintes:

- Funções podem ter zero ou mais argumentos. As funções sem argumentos são utilizadas para definir atributos e relacionamentos entre entidades.
- Funções podem ser classificadas em: simples (retornam uma entidade) e multivaloradas (retornam um conjunto de entidades).
- Existe uma distinção entre uma entidade e sua identificação externa ou chave (autorepresentação de objetos).
- Entidades podem ser organizadas numa hierarquia de tipos [20,24], onde atributos e relacionamentos são automaticamente herdados dos supertipos por cada um dos subtipos.
- A partir de funções básicas é possível criar funções derivadas, isto é, o conceito de dados derivados [24] é suportado de forma simples e natural.

2.2. Uma Avaliação do Modelo Funcional

Mas porque o modelo funcional é uma ferramenta atraente para a modelagem conceitual? Kerschberg & Pacheco [11], Kulkarni [12] e Orman [15] enumeram algumas destas razões:

- Os modelos funcionais fornecem um ambiente de modelagem semanticamente rico. O fato de não distinguir entre dados e programas (dados derivados) facilita bastante a modelagem conceitual.
- As linguagens de consulta podem ser especificadas de uma forma mais natural. Além disso, facilidades de manipulação de dados podem ser integradas naturalmente com operações de propósito geral (FQL, a linguagem definida por Buneman & Frankel [5], é um exemplo).
- Existem algoritmos para mapear a estrutura de um banco de dados de rede (CODASYL) ou relacional para o modelo funcional [11]. Portanto, o modelo de dados funcional pode ser usado como mecanismo de integração de bancos de dados heterogêneos [21].
- O modelo funcional representa a informação através de fatos atômicos (funções) [15], eliminando alguns problemas de redundância e evitan

anomalias na atualização.

- A teoria matemática de funções pode ser explorada para fornecer uma base teórica sólida para bancos de dados.

- A função, segundo Orman [15], é uma estrutura mais primitiva e poderosa que outras comumente usadas pelos modelos já conhecidos, como relações, arquivos, conjuntos DBTG e arranjos. A partir de funções, consultas e restrições são naturalmente implementadas, já que consultas são funções aplicadas a valores de entrada e que retornam valores de saída, enquanto restrições são predicados que correspondem a funções cujo co-domínio é o conjunto de valores lógicos.

3. DESCRIÇÃO DA LINGUAGEM LBF

A LBF é uma linguagem interativa, com a qual o usuário pode criar, operar e atualizar seu banco de dados. Esta linguagem, como a linguagem EFDM [12], é auto-contida e portanto não precisa ser embutida numa linguagem de propósito geral como sugere Shipman [18] ao definir DAPLEX. LBF é basicamente uma extensão da linguagem DAPLEX, à qual foram adicionadas algumas das construções utilizadas em EFDM bem como comandos condicionais e de E/S. Uma descrição detalhada da sintaxe da LBF se encontra em [16].

De uma maneira geral, para a modelagem do mundo real são necessários dois tipos de mecanismo [24]: um mecanismo que permita a descrição das propriedades estáticas do mundo real, isto é, suas estruturas e as restrições sobre elas; e um mecanismo que permita a descrição das propriedades dinâmicas, isto é, as operações aplicadas às estruturas. A seguir descrevemos como estes mecanismos são implementados na LBF.

3.1. Estruturas

Como DAPLEX, LBF modela as entidades do mundo real e as suas propriedades através de funções. A declaração destas funções é feita utilizando-se o comando DECLARE. Funções podem ter zero ou mais argumentos, sendo que funções sem argumentos são utilizadas para modelar entidades, enquanto que funções com argumentos são utilizadas para modelar propriedades e relacionamentos. Por exemplo, considerando o banco de dados da Figura 1,

```
DECLARE ARTIGO () ->> ENTIDADE
```

declara PERIODICO como uma entidade multivalorada e

```
DECLARE TITULO(ARTIGO) -> ALFA(2Ø)
DECLARE DETALHES(ARTIGO) -> ALFA(1ØØ)
```

declara seus atributos TITULO como uma cadeia de 2Ø caracteres e DETALHES como uma cadeia de 1ØØ caracteres.

LBF, como outras linguagens baseadas no modelo funcional, manipula conjuntos de entidades e conjuntos de valores. Estes últimos, também

chamados conjuntos de entidades pré-definidas, podem ser classificados em conjuntos de valores alfanuméricos, numéricos (os quais podem ter parte decimal) e lógicos. Os conjuntos de entidades representam objetos do mundo real, mas não são identificadores, isto é, números ou nomes que identificam estes objetos externamente, não podendo, por isso, ser utilizados em comandos de E/S. As funções são também classificadas em multivaloradas e simples, conforme retornem como resultado um conjunto de entidades, ou uma única entidade.

Outra característica da LBF herdada de DAPLEX é a utilização de hierarquias de generalização [20,24]. Assim, por exemplo,

DECLARE AUTOR() ->> PESSOA

declara um AUTOR como um subtipo de PESSOA, o que indica que qualquer entidade do tipo AUTOR é também uma entidade do tipo PESSOA, e que todas as propriedades e relacionamentos definidos para o tipo PESSOA são automáticamente herdados pelo tipo AUTOR.

LBF também permite a definição de funções derivadas [17] a partir das funções básicas declaradas através do comando DECLARE. As funções derivadas são definidas utilizando-se o comando DEFINA. Por exemplo, dada a função

AUTOR(ARTIGO) ->> AUTOR

podemos definir a função derivada (inversa)

DEFINA ARTIGO(AUTOR) ->> INVERSA DE AUTOR(ARTIGO)

que retorna como resultado a lista de artigos publicados por um determinado autor.

Do ponto de vista estrutural, LBF apresenta as seguintes diferenças com relação a DAPLEX:

- Ao contrário da proposta de Shipman, as funções simples são por definição consideradas parciais. Isto apresenta duas vantagens: é possível introduzir no banco de dados objetos cujos dados são incompletos e funções inversas podem ser definidas livremente.
- Shipman permite que argumentos de funções sejam expressões arbitrárias. A razão para isto parece ser o problema de manipular funções com múltiplos argumentos. Neste caso, se uma função utiliza tipos entidade como argumento, para algumas combinações destes argumentos a função pode não estar definida. Por exemplo, na declaração

DECLARE NUMARTIGOS(AUTOR, PERIODICO) -> NUMERICO(2)

podem existir pares autor-periódico para os quais a função não estaria definida, violando o fato que as funções devem ser totais. Como na LBF esta restrição não existe, todos os argumentos devem ser do tipo entidade, o que simplifica a sintaxe de definição das funções.

3.2. Operações

Do ponto de vista das operações, LBF apresenta sensíveis diferenças com relação a DAPLEX, que são abordadas a seguir.

3.2.1. Operações de Seleção e Recuperação de Dados

DAPLEX utiliza expressões e comandos como construções básicas para a manipulação de dados. Expressões aparecem sempre dentro de comandos e representam tanto um conjunto de entidades quanto uma só entidade, o que permite classificá-las como expressões de conjunto e expressões simples respectivamente. As expressões em geral têm três características: um valor, um papel e uma ordem. O valor da expressão é o conjunto de entidades (ou a entidade) retornado após a sua avaliação. O papel de uma expressão está relacionado com o tipo através do qual as entidades do conjunto são interpretadas. Ordem está associada com expressões de conjunto e representa a ordem entre as entidades do conjunto. LBF utiliza a sintaxe de DAPLEX para a especificação de uma ordem parcial de um conjunto (veja [16]).

Quanto aos comandos, existem algumas diferenças entre LBF e DAPLEX. A mais importante é a utilização por parte da LBF de um comando condicional. O comando condicional foi definido para facilitar operações (consultas ou atualizações) dependentes de uma condição, evitando-se o desmembramento das mesmas. Por exemplo, considerando o banco de dados de uma empresa, a operação de atualização "Elevar em 25% o salário de todas as pessoas do Departamento de "Desenvolvimento" e em 15% o salário dos demais empregados" seria expressa na LBF da seguinte maneira:

```
PARA CADA EMPREGADO
    SE NOME (DEPART(EMPREGADO)) = "Desenvolvimento" ENTAO
        SEJA SALARIO(EMPREGADO) = 1.25 * SALARIO(EMPREGADO)
    SENAO
        SEJA SALARIO(EMPREGADO) = 1.15 * SALARIO(EMPREGADO)
    FIM
FIM
```

Outra forma de utilização do comando SE é na avaliação de expressões simples e de conjunto. Esta forma de utilização é bastante útil na definição de funções derivadas. Por exemplo,

```
DEFINA POTENCIA (X EM NUMERICO(5), J EM NUMERICO(1)) ->
                SE J = Ø ENTAO 1
                SENAO X * POTENCIA(X,J - 1)
                FIM
```

define a função POTENCIA que calcula o valor de X elevado à J-éssima

potência.

O primeiro tipo de comando SE sempre é utilizado dentro de um comando PARA. Portanto, nenhuma operação pode começar com o comando SE. Esta restrição foi imposta para facilitar a especificação das operações, não diminuindo em nada o poder da linguagem, já que qualquer operação que seria especificada a partir do comando SE pode igualmente ser definida utilizando-se o comando PARA. Uma descrição detalhada da sintaxe do comando condicional pode ser encontrada em [16].

Quanto aos operadores utilizados nas expressões de conjunto, LBF, ao contrário de DAPLEX, provê os operadores UNIAO, INTERSECAO e DIFERENCA. Em DAPLEX estes operadores são utilizados exclusivamente na definição de funções derivadas. Como operadores sobre expressões simples a LBF utiliza os operadores lógicos E, OU e NÃO, os operadores aritméticos "+", "-", "*", "/" e RES (módulo), e o operador "&" para concatenação de cadeias de caracteres.

3.2.2. Operações de Entrada e Saída de Dados

Para as operações de entrada de dados, LBF utiliza o conceito de ENTRADA que equivale a uma "entidade externa" cuja função é possibilitar a carga de grandes volumes de dados a partir de arquivos de entrada. Por exemplo, para alterar o endereço de alguns usuários da biblioteca pessoal (ver Figura 1), bastaria criar uma entidade externa INFORMACAO (com seus respectivos atributos) como a seguir,

```
DECLARE INFORMACAO() ->> ENTRADA
DECLARE NOME(INFORMACAO) -> CADEIA(2Ø)
DECLARE ENDERECO(INFORMACAO) -> CADEIA(3Ø)
```

e executar os comandos

```
PARA CADA INFORMACAO
    PARA O USUARIO TAL QUE
                NOME(USUARIO) = NOME(INFORMACAO)
        SEJA ENDERECO(USUARIO) = ENDERECO(INFORMACAO)
    FIM
FIM
```

Para a saída de dados, LBF utiliza o comando ESCREVA (IMPRIMA para emitir relatórios na impressora), que possibilita a geração de relatórios simples, em formato pré-definido, com cabeçalhos, quebra por diferentes campos e classificados em ordem ascendente ou descendente. Uma discussão mais detalhada dos comandos ESCREVA e IMPRIMA foge ao escopo deste artigo e pode ser encontrada em [16].

3.2.3. Operações de Atualização do Banco de Dados

Quanto às operações de atualização, LBF possui as seguintes características:

- Já que as funções são por definição parciais, uma função simples não precisa ser inicializada no momento de sua criação. Não obstante, para facilitar a identificação de uma entidade, pelo menos uma de suas funções (ou um grupo) deve ser inicializada no momento de sua criação.
- Na LBF é possível incluir uma entidade existente no banco de dados no conjunto de entidades de um determinado tipo. Para isso, é utilizada a mesma sintaxe usada para a inclusão de entidades no conjunto correspondente ao co-domínio de uma função multivalorada. Por exemplo,

```
INCLUA USUARIO = (O AUTOR TAL QUE
                  NOME(AUTOR) = "Shipman")
```

inclui o autor de nome "Shipman" também como usuário da biblioteca pessoal.
- Da mesma forma é possível excluir uma entidade do conjunto correspondente a um determinado tipo, o que resulta na exclusão de suas referências dos conjuntos correspondentes a todos os seus subtipos. Por exemplo,

```
EXCLUA USUARIO = (O AUTOR TAL QUE
                  NOME(AUTOR) = "Shipman")
```

exclui o autor de nome "Shipman" do conjunto de usuários da biblioteca pessoal, permanecendo porém, como autor.
- Entidades podem ser removidas do banco de dados através do comando REMOVA. Assim,

```
REMOVA O AUTOR TAL QUE NOME(AUTOR) = "Shipman"
```

resulta na completa remoção do banco de dados do autor de nome "Shipman" e de suas referências nos conjuntos correspondentes a todos os seus subtipos e supertipos.

3.2.4. Operações sobre o Esquema

De acordo com a proposta de Shipman, sobre um esquema DAPLEX somente é permitida a operação de adição de novas funções. LBF aproveita as idéias de Kulkarni [12] e fornece um novo operador, REMOVA, para a eliminação de funções que o usuário já não mais interessa em manter no banco de dados. Por exemplo,

```
REMOVA ENDERECO(PESSOA)
```

causa a remoção da função ENDERECO e de todas as funções definidas a partir dela.

3.3. Armazenamento e Especificação de Consultas

Na linguagem LBF, as consultas podem ser encapsuladas, através da cláusula CONSULTA, e identificadas por um nome para posterior referência. Por exemplo, a consulta

```
CONSULTA MULHERES:
  PARA CADA PESSOA TAL QUE SEXO(PESSOA) = "F"
      ESCREVA(NOME(PESSOA))
  FIM
FIM
```

pode ser referenciada e executada a qualquer momento através da invocação de seu nome. Além disso, ela pode também ser removida utilizando-se o comando REMOVA.

O encapsulamento é muito útil no caso de consultas muito usadas e tem como vantagem a velocidade de execução, já que o código gerado pode ser armazenado, evitando ser compilado toda vez que a consulta é executada.

3.4. Restrições

A proposta de Shipman para a especificação de restrições em DAPLEX inclui o uso de duas construções CONSTRAINT e TRIGGER [17]. Estas construções, entretanto, não são completas. LBF não suporta estes mecanismos de restrição. Um trabalho complementar deverá introduzir a definição de um mecanismo mais geral para a especificação de restrições, podendo ser adicionado posteriormente à linguagem. Não obstante, LBF permite a especificação de restrições básicas, tais como:

Restrições de totalidade. São utilizadas no caso em que todo objeto pertencente a um tipo de entidade estiver sempre associado a outro objeto no banco de dados. Por exemplo,

```
DECLARE AUTOR(ARTIGO) ->> AUTOR TOTAL
```

indica que um artigo deve ter sempre um autor.

Restrições sobre cardinalidade. As palavras MAXIMO e MINIMO são utilizadas para restringir o número de possíveis valores para a população de tipos entidade e funções multivaloradas. Por exemplo,

```
DECLARE ARTIGOS(USUARIO) ->> ARTIGO MAXIMO 3
```

indica que o número máximo de artigos emprestados a um usuário da biblioteca pessoal é 3.

Restrições sobre valores das funções. Certas funções podem ser restringidas de modo a não ter seus valores alterados. Por exemplo,

```
DECLARE TITULO(ARTIGO) -> ALFA(30) FIXO
```

indica que após a inclusão do título do artigo este não pode ser mais alterado.

Restrições sobre a identificação de entidades. Num banco de dados os

usuários podem estar interessados em distinguir entidades individuais de modo que possam se referir a elas de forma não ambígua. Por exemplo,

DECLARE NOME(PESSOA) -> ALFA(2Ø) UNICO

indica que duas pessoas não podem ter o mesmo nome.

4. ESTADO ATUAL DO TRABALHO E DIRETRIZES PARA PESQUISAS FUTURAS

O objetivo deste artigo foi apresentar uma descrição geral das características da LBF, mostrando suas principais diferenças em relação às linguagens DAPLEX e EFDM. LBF está neste momento sendo implementada para microcomputadores compatíveis com o IBM PC, tendo como suporte para o gerenciamento de arquivos o sistema Btrieve [22].

Após esta implementação inicial, nosso trabalho consistirá em adicionar alguns mecanismos que até o momento não foram considerados, tais como:

Definição de visões. A proposta de Shipman relacionada com a definição e atualização de visões é incompleta já que nela somente é tratado o caso em que uma atualização sobre uma visão ocasiona uma única atualização sobre o banco de dados. EFDM [12], por outro lado, fornece um mecanismo diferente para definir visões que não permite visões atualizáveis. O nosso propósito é estender posteriormente a linguagem LBF, de maneira a permitir visões atualizáveis considerando o caso de múltiplas atualizações sobre o banco de dados.

O uso de tipos de dados nas linguagens de bancos de dados. Um conceito bastante útil para a modelagem do mundo real é o de tipos de dados. Por exemplo, em PASCAL [9] e EUCLID [13] podemos definir os seguintes dois tipos de dados

TYPE ALTURA = 1...1ØØ

TYPE PESO = 1...1ØØ

Como consequência, operações sem significado podem ser efetuadas sobre estes tipos (por exemplo, somar pesos com alturas). Portanto, é necessário que a linguagem faça uma melhor verificação de tipos para obter uma maior integridade semântica. Trabalhos a este respeito têm sido desenvolvidos no contexto do modelo relacional [1,2Ø]. Seria interessante estender LBF para que permitisse diferenciar entre dois tipos de entidade semanticamente diferentes.

O uso de tipos abstratos de dados. Tipos abstratos de dados têm sido estudados extensivamente no contexto de linguagens de programação. Ultimamente, bastante atenção tem sido dada no sentido de utilizar abstração de dados no contexto de modelagem semântica de dados [4]. Em certas áreas de aplicação, como a utilização de bancos de dados para suporte a

PAC (Projeto Assistido por Computador), existem sérios problemas de representação e operação de objetos. Tipos abstratos de dados (TAD's) podem ser uma alternativa para a criação e manipulação de objetos como pontos, linhas e polígonos. Recentemente, algumas implementações de TAD's têm sido desenvolvidas, entre elas podemos citar o trabalho de Stonebraker no contexto do sistema INGRES [23]. Neste trabalho, foram definidos TAD's sobre colunas de uma relação, isto é, TAD's simples. Um trabalho interessante seria estender LBF não só para suportar TAD's simples como proposto por Stonebraker [23], mas também para suportar outros tipos como aqueles necessários para modelar e manipular objetos complexos (por exemplo, em aplicações de PAC).

5. BIBLIOGRAFIA

[1] BRODIE, M.L. "The Aplication of Data Types to Database Semantic Integrity". Information Systems, Pergamon Press , 5(4):287-296, 1980.

[2] BRODIE, M.L. & SILVA, A. "Active and Passive Component Modelling". In: OLLE, T. W. et al (eds.). "Information Systems Design Methodologies: A Comparative Review". North-Holland, Amsterdam, 1982, p. 41-91.

[3] BRODIE, M.L. "On the Development of Data Models". In: BRODIE, M.L., MYLOPOULOS, J. & SCHMIDT, J.W. (eds.). "On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and Programming Languages". New York, Springer Verlag, 1984, p.21-47.

[4] BRODIE, M.L. & RIDJANOVIC, D. "On the Design and Specification of Database Transactions". In: BRODIE, M.L., MYLOPOULOS, J. & SCHMIDT, J.W. (eds.). "On Conceptual Modelling. Perspectives from Artificial Inteligence, Databases, and Programming Languages". New York, Springer Verlag, 1984, p.277-306.

[5] BUNEMAN, P. & FRANKEL, R.E. "FQL - A Functional Query Language". In: PROCEEDING OF ACM SIGMOD CONF. ON MANAGEMENT OF DATA, Boston, Mass., 1979, p. 52-58.

[6] CODD, E.F. "Extending the Relational Model of Data to Capture more Meaning". ACM Tranactions on Database Systems, New York, 4(4):397-434, 1979.

[7] HAMMER, M. & MCLEOD, D. "Database Description with SDM: a Modelling Mechanism for Database Aplications". ACM Transactions on Database Systems, New York, 6(3):351-386, 1981.

[8] HOUSEL, B.C. , WADDLE, V. & YAO, S.B. "The Functional Dependency Model for Logical Database Design". In: PROCEEDINGS OF INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 5. Rio de Janeiro, Brazil, Out.

1979, p.194-208.

[9] JENSEN, K. & WIRTH, N. "PASCAL User Manual and Report", 2nd Ed., Lectures notes in Computer Science, Berlim, Springer-Verlag, 1974, p.18.

[10] KATZ, R.H. & WONG, E. "Resolving Conflicts in Global Storage Design through Replication". ACM Transactions on Database Systems, New York, 8(1):110-135, 1983.

[11] KERSHBERG, L. & PACHECO, J.E.S. "A Functional Data Base Model". Rio de Janeiro, Brazil, Pontificia Universidade Católica, Fev. 1976. (Monograph in Computer Science 2/76).

[12] KULKARNI, G. K. "Evaluation of functional data models for database design and use". Edinburgh, University of Edinburgh, 1983. 153p. (Tese de Ph.D.).

[13] LAMPSON, B.W. & et al. "Report on the Programming Language EUCLID", SIGPLAN Notices, 12(2), 1977.

[14] McLEOD, D. & SMITH, J.M. "Abstractions in Databases". In: BRODIE, M.L. & ZILLES, S.N. (eds.) PROC. WORKSHOP ON DATA ABSTRACTION, DATABASES AND CONCEPTUAL MODELLING. SIGPLAN Notices, New York, 16(1):19-25, 1981.

[15] ORMAN, L. "Functions in Information Systems". Data Base, 16(3):10-13, 1985.

[16] PEREZ, J.J.,LAENDER, A.H.F. & BIGONHA, R.S. "Especificação da Linguagem para Banco de dados Funcionais (LBF)". (Relatório técnico em elaboração).

[17] SCHIEL, U. "An Abstract Introduction to the Temporal Hierarchic Data Model (THM)", In: PROCEEDINGS OF INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASE, 9, Florence, Italy, 1983.

[18] SHIPMAN, D. W. "The Functional Data Model and the Data Language DAPLEX". ACM Transactions on Database Systems, New York, 6(1): 140-173, 1981.

[19] SIBLEY, E.H. & KERSCHBERG, L. "Data Architecture and Data Model Considerations". In: AFIPS CONFERENCE PROCEEDINGS, NATIONAL COMPUTER CONFERENCE, Dallas, Texas, Montvale, N. J., AFIPS Press, 1977, V.46, pp.85-96.

[20] SMITH, J. M. & SMITH D.C.P. "Database Abstractions: Aggregation and Generalization". ACM Transactions on Database Systems, New York, 2(2):105-133, 1977.

[21] SMITH, J.M. et al. "Multibase - Integrating Heterogeneous Distribute Database Systems". In: AFIPS CONFERENCE PROCEEDINGS, NATIONAL COMPUTER CONFERENCE, Montvale, N.J., AFIPS Press, 1981, V.50, p.487-

499.

[22] SOFTCRAFT, Inc. "Btrieve version 3.Ø user's guide". Austin, Texas, 1984.

[23] STONEBRAKER, M. "Adding Semantic Knowledge to a Relational Database System". In: BRODIE, M.L., MYLOPOULOS, J. & SCHMIDT, J.W. (eds.). "On Conceptual Modelling. Perspectives from Artificial Inteligence, Databases, and Programming Languages". New York, Springer Verlag, 1984, p.333-353.

[24] TSICHRITZIS, D. & LOCHOVSKY, F.H. "Data Models". Prentice-Hall, Englewood Cliffs, N.J., 1982, 381p.

[25] WONG, E. & KATZ, R.H., "Logical Design and Schema Conversion for Relational and DBTG Databases". In : CHEN P. P.(ed.) "Entity-Relationship Approach to Systems Analysis and Design". Amsterdam, North-Holland, 198Ø, p.344-383.