A ABORDAGEM FUNCIONAL PARA BANCOS DE DADOS E A LINGUAGEM LBF

AUTORES:
ALBERTO HENRIQUE FRADE LAENDER
JOSÉ DE JESÚS PÉREZ ALCÁZAR
ROBERTO DA SILVA BIGONHA

ENDEREÇO:
Departamento de Ciência da Computação
ICEx - UFMG
Caixa Postal 702
CEP 30161 - BELO HORIZONTE - MG
Tel.: (031) 443-4088

CURRICULUM VITAE:
ALBERTO HENRIQUE FRADE LAENDER é Ph.D. em Ciência da Computação (Universidade de East Anglia, Inglaterra, 1984). Professor Adjunto do DCC/UFMG. Áreas de interesse: Banco de Dados e Engenharia de Software.
JOSÉ DE JESÚS PÉREZ ALCÁZAR é Mestre em Ciência da Computação (UFMG, 1988). Áreas de interesse: Banco de Dados, Linguagens e Compiladores.
ROBERTO DA SILVA BIGONHA é Ph.D. em Ciência da Computação (Universidade da Califórnia, Los Angeles, 1981). Professor Adjunto do DCC/UFMG. Áreas de interesse: Linguagens e Compiladores.

RESUMO:
Este trabalho apresenta uma visão geral da abordagem funcional para bancos de dados, salientado as suas principais vantagens com relação às abordagens tradicionais baseadas nos chamados modelos de dados clássicos (hierárquico, de rede e relacional). Como ilustração da abordagem funcional é utilizada a linguagem LBF (Linguagem para Bancos de Dados Funcionais) desenvolvida no Departamento de Ciência da Computação da UFMG.

PALAVRAS-CHAVE:
Banco de Dados, Modelagem de Dados, Modelo de Dados Funcional, Linguagem de Manipulação de Dados, LBF.

INTRODUÇÃO

Nos últimos anos, tem sido intensa a procura por novas formas de modelagem e manipulação de bancos de dados. Vários modelos de dados têm sido propostos com o objetivo de melhor capturar a semântica dos dados de uma aplicação e remover várias das restrições impostas pelos chamados modelos clássicos (hierárquico, de rede e relacional). Estes novos modelos, denominados modelos semânticos [2, 8], foram projetados para fornecer mecanismos de modelagem semanticamente mais expressivos e ricos, que permitissem definir e manipular um banco de dados de forma mais clara e precisa [1, 4, 5, 15].

Dentre os vários modelos de dados semânticos descritos na literatura, destaca-se, por sua simplicidade, o modelo funcional [3, 6, 7, 13, 19]. O modelo de dados funcional tem suas bases no conceito matemático de função, oferecendo uma abordagem extremamente elegante para a manipulação de bancos de dados. Além disso, o modelo funcional possibilita a utilização de linguagens bastante poderosas e de fácil utilização por usuários não especializados.

Neste trabalho apresentaremos uma visão geral do modelo de dados funcional, salientando as suas principais vantagens com relação aos modelos clássicos. Como exemplo da abordagem funcional, utilizaremos a linguagem LBF (Linguagem para Bancos de Dados Funcionais) [10, 11, 12] desenvolvida no Departamento de Ciência da Computação da UFMG. A LBF é uma linguagem interativa e totalmente autocontida, que permite definir, manipular e atualizar um banco de dados de acordo com o modelo funcional. A especificação da LBF foi feita com base na linguagem DAPLEX, proposta por David Shipman na Computer Corporation of America [13], tendo sido, entretanto, incluídas várias extensões.

O MODELO DE DADOS FUNCIONAL

O modelo funcional nasceu da busca de novas formas de modelagem generalizadas que pudessem ser utilizadas como base comum para a descrição e comparação dos três modelos clássicos [14]. Este formalismo deveria fornecer uma base unificada para o projeto de esquemas, tanto de bancos de dados relacionais quanto de bancos de dados de rede, que não apresentassem anomalias de atualização.

O modelo funcional tem suas bases no conceito matemático de função e utiliza dois tipos de primitiva (estrutural) para a modelagem de uma aplicação do mundo real [17]: os conjuntos de objetos e as funções

que são aplicadas a estes para relacioná-los entre si. Os conjuntos de objetos estão divididos em conjuntos de entidades e conjunto de valores. A principal diferença entre um conjunto de entidades e um conjunto de valores é que este último nunca faz parte do domínio de uma função, isto é, eles são utilizados apenas para definir os atributos de um conjunto de entidades. Os conjuntos de valores são chamados em geral de conjuntos de entidades pré-definidas (por exemplo), conjuntos de inteiros, reais, etc. [7]. As funções podem ser totais ou parciais, e a distinção corresponde a uma restrição [19] sobre o número de elementos do conjunto (domínio) que devem participar do relacionamento funcional. Além disso, elas podem ser classificadas como simples (retornam como resultado uma única entidade) ou multivaloradas (retornam como resultado um conjunto de entidades).

As entidades podem ainda ser organizadas em hierarquias de tipos (generalizações [15, 18] através das quais atributos e relacionamentos são automaticamente herdados dos supertipos por cada um dos subtipos. Este mecanismo dá ao modelo uma grande expressividade semântica, uma vez possibilita modelar de forma simples situações comumente encontradas no mundo real onde uma mesma entidade desempenha diferentes papéis. Por exemplo, uma mesma pessoa pode ser ao mesmo tempo fornecedor e cliente de uma empresa.

A Figura 1 apresenta a descrição gráfica, de acordo com o modelo funcional, do esquema conceitual do banco de dados de uma biblioteca pessoal de artigos publicados em periódicos. De acordo com a notação, as elipses representam os conjuntos de objetos (entidades e valores) e as setas as funções que relacionam estes objetos entre si. As setas simples correspondem função simples e as setas duplas a funções multivaloradas. As generalizações são representadas por seta "largas" e correspondem a um tipo especial de função derivada do conceito de hierarquia "e-um" ("is-a") utilizado em inteligência artificial [18]. Este banco de dados será usado daqui para frente como base para todos os exemplos.

UMA AVALIAÇÃO DO MODELO FUNCIONAL

Várias versões do modelo funcional foram propostas na literatura [3, 6, 7, 13, 19]. A mais importante delas é, sem dúvida, a de Shipman cujos conceitos estão totalmente embutidos na linguagem DAPLEX [13].

Mas por que o modelo funcional é uma ferramenta atraente para a modelagem de bancos de dados? A principal característica do modelo funcional é a sua expressividade semântica [7, 12]. Ao contrário dos modelos clássicos (hierárquico, de rede e relacional) que são orientados a registros [2], o modelo funional é orientado a objetos; ou seja, no modelo funcional há uma distinção clara entre um objeto (entidade) e os nomes utilizados para representá-lo externamente (chaves de identificação). Isso permite a referência explícita a um objeto a partir de outro, tornado possível a definição de linguagens mais naturais, nas quais a semântica das operações de recuperação e manipulação é mais facilmente entendida [12].

Além dessa característica, podemos ainda enumerar as seguintes vantagens do modelo funcional [6, 7, 9, 12]:

- O modelo funcional representa a informação através de fatos atômicos (funções) [9], eliminando alguns problemaa de redundância e evitando anomalias de atualização.
- A função, segundo Orman [9], é uma estrututa mais primitiva e poderosa que outras comumente usadas pelos modelos já conhecidos, como arquivos, relações e conjuntos DBTG. 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 contra-domínio é o conjunto de valores lógicos.
- A partir de funções básicas é possível definir funções derivadas, isto é, o conceito de dados derivados [18] é suportado de forma simples e natural.
- Existem algoritmos para mapear a estrutura de um banco de dados de rede (CODASYL) ou relacional para o modelo funcional [6] e vice-versa. Portanto, o modelo de dados funcional pode ser usado como um mecanismo para a integração de bancos de dados heterôgenos [16].
- A teoria matemática de funções pode ser explorada para fornecer uma base teórica sólida para bancos de dados.
- Os modelos funcionais permitem a autodescrição de um banco de dados através de um simples metaesquema que também pode ser representado e manipulado com as mesmas construções do modelo [9].

A LINGUAGEM LBF

A LBF é uma linguagem interativa através da qual é possível definir, manipular e atualizar um banco de dados funcional. A LBF foi definida a partir da linguagem DAPLEX, tendo sido incluídas várias extensões, entre elas algumas construções propostas por Kulkarni [7], comandos condicionais e de entrada/saída. A principal diferença entre LBF e a linguagem DAPLEX, entretanto, reside no fato de que a LBF é uma linguagem autocontida, ou seja, não precisa ser embutida em uma linguagem de propósito geral conforme sugerido por Shipman ao definir DAPLEX [13].

De uma maneira geral, para a modelagem de uma aplicação do mundo real, são necessários três tipos de mecanismo [18]: um mecanismo que permita descrever as propriedades estáticas do mundo real, isto é, as suas estruturas; um mecanismo que permita descrever as propriedades dinâmicas, isto é, as operações que podem ser processadas sobre as estruturas; e um mecanismo que permita restringir, entre as possíveis estruturas, aquelas que são válidas para a aplicação, isto é, as restrições de integridade. A seguir faremos uma descrição da LBF considerando estes três mecanismos. Uma descrição detalhada da sintaxe e da semântica da LBF pode ser encontrada em [12].

ESTRUTURAS

Como DAPLEX, a LBF modela as entidades do mundo real e as suas propriedades através de funções. A declaração destas funções é feita utilizando-se o co-

mando 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 os atributos das entidades e os relacionamentos entre entidades. Por exemplo, considerando o banco de dados da Figura 1,

DECLARE ARTIGO ( )->> *ENTIDADE*

declara ARTIGO como uma entidade multivalorada e

DECLARE TÍTULO (ARTIGO) -> ALFA (50)

DECLARE RESUMO (ARTIGO) -> ALFA (100)

declaram seus atributos TÍTULO com uma cadeia de 50 caracteres e RESUMO como uma cadeia de 100 caracteres.

A 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 e lógicos. Os conjuntos de entidades representam objetos do mundo real, mas não são identificadores, isto é, números ou nomes que identifiquem estes objetos externamente, não podendo, portanto, ser utilizados em comandos de entrada e saída. As funções são classificadas em simples e multivaloradas, conforme retornem como resultado uma única entidade ou um conjunto de entidades.

Uma outra característica da LBF herdada de DAPLEX é a possibilidade de se definir hierarquias de generalização [15, 18]. Assim, por exemplo,

DECLARE AUTOR ( )->>|PESSOA

declara AUTOR como um subtipo de PESSOA, o que indica que qualquer entidade do tipo AUTOR é também uma entidade do tipo PESSOA, e que todos os atributos e relacionamentos definidos para PESSOA são automaticamente herdados por AUTOR.

A LBF permite também a definição de funções derivadas 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 ARTIGOS (AUTOR)->>*INVERSA DE* AUTOR (ARTIGO)

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

Do ponto de vista estrutural, a diferença básica entre a LBF e linguagem DAPLEX, é o fato de que na LBF as funções simples são por definição consideradas parciais. Isto apresenta duas vantagens: (1) é possível introduzir no banco de dados objetos cujos dados são incompletos; (2) funções inversas podem ser definidas livremente.

A Figura 2 apresenta o esquema completo do banco de dados da biblioteca pessoal de acordo com a sintaxe da LBF. Note que, além das funções básicas, o esquema inclui também a definição de algumas funções derivadas.

Operações

Do ponto de vista das operações, a LBF apresenta sensíveis diferenças com relação a DAPLEX. Uma discussão detalhada dessas diferenças foge ao escopo deste trabalho e pode ser encontrada em [12].

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

O comando PARA é o comando básico da LBF e possibilita a iteração sobre um conjunto de entidades. Assim, considerando o banco de dados descrito na Figura 1, para se obter o título de todos os artigos e o nome do periódico em que foram publicados, a seguinte consulta deve ser apreciada:

```
PARA CADA ARTIGO
    ESCREVA (TÍTULO (ARTIGO), NOME (PERIÓDICO (ARTIGO))
FIM
```

Note que no exemplo anterior o nome do periódico é obtido pela composição das funções NOME (PERIÓDICO) e PERIÓDICO (ARTIGO).

Um outro exemplo, no qual é imposta uma condição sobre o conjunto de entidades retornado, é a consulta "Liste título de todos os artigos publicados pelo autor de nome Shipman":

```
PARA CADA ARTIGO TAL QUE
    ALGUM NOME (AUTOR (ARTIGO) = "Shipman"
        ESCREVA TÍTULO (ARTIGO))
FIM
```

Note que neste exemplo a função NOME (AUTOR) é automaticamente herdada da entidade PESSOA. Além disso, devido ao fato de a função AUTOR (ARTIGO) ser multivalorada, foi utilizada a claúsula ALGUM que corresponde a um quantificador existencial em termos do cálculo de predicados.

Com relação aos comandos, a principal diferença entre a LBF e a linguagem DAPLEX 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:

```
PARA CADA PESSOA
    SE SEXO (PESSOA) = "M" ENTÃO
        ESCREVA (NOME (PESSOA))
    SENÃO
        ESCREVA (NOME (PESSOA), ENDEREÇO (PESSOA))
    FIM
FIM
```

Quanto aos operadores utilizados nas expressões de conjunto, a LBF, ao contrário de DAPLEX, provê os operadores UNIÃO, INTERSEÇÃO e DIFERENÇA. 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.

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

```
DECLARE INFORMAÇÃO ( ) ->> ENTRADA
DECLARE NOME (INFORMAÇÃO) -> ALFA (30)
DECLARE ENDEREÇO (INFORMAÇÃO) -> ALFA (30)
```

e executar os comandos

```
PARA CADA INFORMAÇÃO
    PARA O USUÁRIO TAL QUE NOME (USUÁRIO) = NOME (INFORMAÇÃO)
        SEJA ENDEREÇO (USUÁRIO) = ENDEREÇO (INFORMAÇÃO)
    FIM
FIM
```

Uma entidade externa é criada a partir de um texto digitado pelo usuário que contém os dados da entidade. Um utilitário se encarrega de converter os dados do formato texto para um arquivo do banco de dados que representa a entidade externa [12].

Para a saída de dados, a 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 de acordo com determinados campos. Uma discussão detalhada dos comandos ESCREVA e IMPRIMA também foge ao escopo deste trabalho e pode ser encontrada em [12].

Quanto às operações de atualização, a 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. Entretanto, para facilitar a identificação de uma entidade, pelo menos uma de suas funções (ou um grupo delas) deve ser inicializada no momento de sua criação. Por exemplo, uma ocorrência da entidade PESSOA pode ser criada atribuindo-se um valor inicial apenas para função NOME (PESSOA):

```
PARA UMA NOVA PESSOA
    SEJA NOME (PESSOA) = "Shipman"
FIM
```

- Na LBF é possível incluir uma entidade já 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 contra-domínio de uma função multivalorada. Por exemplo, o comando
  INCLUA USUÁRIO = (*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, o comando
  EXCLUA USUÁRIO = (O AUTOR *TAL QUE* NOME (AUTOR) = "Shipman") exclui o autor de nome "Shipman" do conjunto de usuários da biblioteca pes-
- 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.

De acordo com a proposta de Shipman, sobre um esquema DAPLEX é permitida apenas a operação de adição de novas funções. A LBF aproveita as idéias de Kulkarni [7] e fornece um novo comando ELIMINE, para a eliminação de funções que o usuário já não mais interessa em manter no banco de dados. Este comando possibilita, portanto, a eliminação de:

- Atributos de uma entidade:
  ELIMINE ENDEREÇO (PESSOA)
- Relacionamentos de uma entidade:
  ELIMINE ARTIGOS (AUTOR)
- Entidades:
  ELIMINE USUÁRIO

Na linguagem LBF, as consultas podem ainda ser encapsuladas, através da cláusula CONSULTA, e identificada 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 ELIMINE. 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 analisado toda vez que a consulta é executada.

Restrições

A especificação de restrições na linguagem DAPLEX inclui o uso de duas construções CONSTRAINT e TRIGGER [13]. Estas construções, por não serem suficientemente completas, não foram incluídas na LBF. Um mecanismo mais geral para a especificação de restrições deverá ainda ser definido e adicionado posteriormente à linguagem. Entretanto, a 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 a cardinalidade: As cláusulas MÁXIMO e MÍNIMO são utilizadas para restringir o número de ocorrências das entidades e das funções multivaloradas. Por exemplo,
  DECLARE ARTIGOS (USUÁRIO) ->> ARTIGO *MÁXIMO* 3
  indica que um usuário da biblioteca pessoal pode tomar emprestado no máximo três artigos.
- Restrições sobre valores das funções: Certas funções podem ser restringidas de modo a não ter seus valores alterados. Por exemplo,

DECLARE TÍTULO (ARTIGO) –> *ALFA* (50) *FIXO*
indica que após a inclusão do título de um artigo este não pode ser mais alterado.

– Restrições sobre a identificação de entidades: A cláusula ÚNICO permite associar valores únicos a determinadas funções, permitindo a definição de identificadores externos através dos quais os usuários podem distinguir as ocorrências das entidades e se referir a elas de forma não ambígua. Por exemplo,
DECLARE NOME (PESSOA) –> *ALFA* (30) *ÚNICO*
indica que duas pessoas não podem ter o mesmo nome.

CONCLUSÕES

O objetivo deste trabalho foi apresentar uma visão geral da abordagem funcional para a modelagem e a manipulação de bancos de dados. Nesta apresentação procuramos mostrar as características gerais do modelo de dados funcional, enfatizando as suas principais vantagens em relação aos modelos clássicos.

Como ilustração da abordagem funcional, utilizamos a linguagem LBF desenvolvida no Departamento de Ciência da Computação da UFMG. A LBF foi implementada para microcomputadores compatíveis com a linha IBM PC, utilizando como suporte para o gerenciamento de arquivos o sistema Btrieve [17]. O objetivo principal desta implementação foi mostrar a viabilidade desse tipo de linguagem como ferramenta para a manipulação de bancos de dados.

A versão atual da LBF, entretanto, apresenta alguns problemas de eficiência que estão sendo, aos poucos, minimizados. A principal razão para estes problemas é a utilização do sistema Btrieve como suporte para o gerenciamento de arquivos [12]. Como a maioria dos comandos da LBF envolve operações com conjuntos (expressões de conjunto), estas se tornam pouco eficientes devido à estrutura em árvore-B manipulada pelo sistema Btrieve. Uma solução definitiva para este problema requer, no entanto, novas pesquisas sobre estruturas de armazenamento de dados, uma vez que as utilizadas atualmente para implementações de linguagens e sistemas baseados nos modelos de dados clássicos não se prestam aos modelos semânticos. Algumas extensões para a linguagem também estão sendo estudadas visando, principalmente, melhorar as facilidades de entrada/saída e de tratamento numérico, e introduzir novos mecanismos para a especificação de restrições de integridade [12].

Finalmente, cabe ressaltar, que a experiência com a LBF demonstra que é viável o desenvolvimento de projetos de ponta no país, bastando para isso que se dê aos nossos grupos de pesquisa recursos e condições de trabalho, para que possamos chegar a soluções próprias sem a necessidade de importar tecnologia.

AGRADECIMENTOS

Este trabalho contou com apoio financeiro do CNPq (processos 402170/85-CC e 402171/85-CC) e da CAPES.

REFERÊNCIAS BIBLIOGRÁFICAS

1. 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, pp. 41-91.
2. 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 Inteligence, Databases, and Programming Languages, New York, Springer Verlag, 1984, pp. 21-47.
3. BUNEMAN, P. & FRANKEL, R.E. FQL – A functional query language. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, Boston, Mass., 1979, pp. 52-58.
4. CODD, E.F. Extending the relational model of data to capture more meaning. ACM Transactions on Database Systems, 4(4):397-434, Dec. 1979.
5. HAMMER, M. & McLEOD, D. Database description with SDM: a modelling mechanism for database aplications. ACM Transactions on Database Systems, 6(3):351-386, Sept. 1981.
6. KERSHBERG, L. & PACHECO, J.E.S. A Functional Data Base Model, Rio de Janeiro, Pontifícia Universidade Católica, 1976. (Monograph in Computer Science 2/76).
7. KULKARNI, G. K. Evaluation of Functional Data Models for Database Design and Use. Edinburgh, University of Edinburgh, 1983. (Ph. D. Thesis).
8. 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 noticies, 16(1):19-25, 1981.
9. ORMAN, L. Functions in informations systems. Data base, 16(3):10-13, 1985.
10. PÉREZ-ALCÁZAR, J.J., LAENDER, A.H.F. & BIGONHA, R.S. LBF: Uma linguagem para bancos de dados funcionais. In: Anais do VI Congresso da Sociedade Brasileira de Computadores, Recife, 1986, pp. 331-341.
11. PÉREZ-ALCÁZAR, J.J., LAENDER, A.H.F. & BIGONHA, R.S. Diseño de un lenguaje para bases de datos funcionales. In: Anales de la XII Conferencia Latinoamericana de Informatica, Montevideo, Uruguay, 1986, pp. 97-116.
12. PÉREZ-ALCÁZAR, J.J. Projeto e Implementação de uma Linguagem para a Manipulação de Bancos de Dados Funcionais, Belo Horizonte, Departamento de Ciência da Computação, UFMG, 1988. (Dissertação de Mestrado).
13. SHIPMAN, D.W. The functional data model and the data language DAPLEX. ACM Transactions on Database Systems, 6(1): 140-173, Mar. 1981.
14. SIBLEY, E.H. & KERSCHBERG, L. Data architecture and data model considerations. In: AFIPS Conference Proceedings, National Computer Conference, v. 46, Montvale, N.J., 1977, pp. 85-96.
15. SMITH, J.M. & SMITH, D.C.P. Database abstractions: aggregation and generalization. ACM Transactions on Database Systems, 2(2):105-133, Jun. 1977.
16. SMITH, J.M. et al. MULTIBASE – Integrating heterogeneous distributed database systems. In:

AFIPS Conference Proceedings, National Computer Conference, V. 50. Montvale, N.J., AFIPS Press, 1981, pp. 487-499.
17. SOFTCRAFT INC. Btrieve Version 3.0 User's Guide, Austin, Texas, 1984.
18. TSCICHRITZIS, D. & LOCHOVSKY, F.H. Data Models, Englewood Cliffs, N.J., Prentice-Hall, 1982.
19. 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, 1980, pp. 344-383.

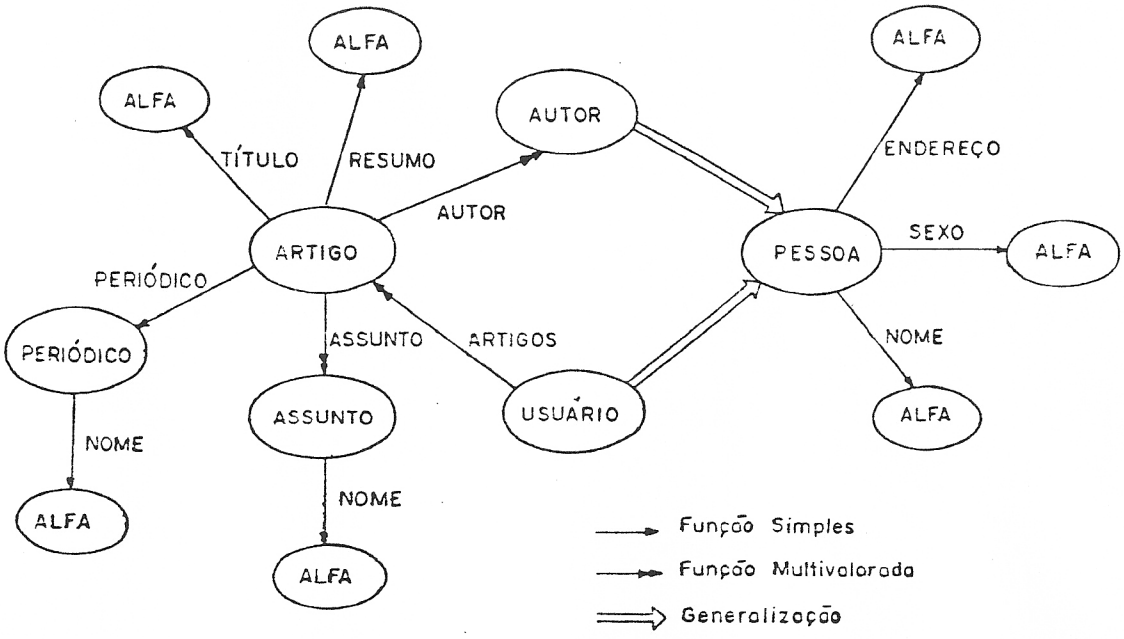

Figura 1

```
DECLARE ARTIGO () ->> ENTIDADE
DECLARE PESSOA () ->> ENTIDADE
DECLARE USUARIO () ->> PESSOA
DECLARE AUTOR () ->> PESSOA
DECLARE ASSUNTO () ->> ENTIDADE
DECLARE PERIODICO () ->> ENTIDADE
DECLARE TITULO (ARTIGO) -> ALFA(50)
DECLARE RESUMO (ARTIGO) -> ALFA(100)
DECLARE PERIODICO (ARTIGO) ->> PERIODICO
DECLARE ASSUNTO (ARTIGO) ->> ASSUNTO
DECLARE AUTOR (ARTIGO) ->> AUTOR
DECLARE NOME (PESSOA) -> ALFA(30)
DECLARE ENDERECO (PESSOA) -> ALFA(30)
DECLARE SEXO (PESSOA) -> ALFA
DECLARE ARTIGOS (USUARIO) ->> ARTIGO
DECLARE NOME (ASSUNTO) -> ALFA(20)
DECLARE NOME (PERIODICO) -> ALFA(30)
DEFINA ARTIGOS(AUTOR) ->> INVERSA DE AUTOR(ARTIGO)
DEFINA ASSUNTO(AUTOR) ->> ASSUNTO(ARTIGOS(AUTOR))
DEFINA ARTIGOS(ASSUNTO) ->> INVERSA DE ASSUNTO(ARTIGO)
DEFINA ARTIGOS(PERIODICO) ->> INVERSA DE PERIODICO(ARTIGO)
DEFINA ASSUNTO(PERIODICO) ->> ASSUNTO(ARTIGOS(PERIODICO))
DEFINA USUARIO(ARTIGO) ->> INVERSA DE ARTIGOS(USUARIO)
```

Figura 2