Modelagem de Esquemas em um Ambiente de Banco de Dados Orientado a Objetos

Leonardo Chaib Rossetti

Ana Maria de C. Moura

Instituto Militar de Engenharia
Seção de Matemática e Engenharia de Sistemas
Pça General Tiburcio 80 - Praia Vermelha
CEP: 22290 - Rio de Janeiro - RJ
Brasil

RESUMO

O paradigma de Orientação a Objetos (O-O) veio de encontro às exigências dos atuais Bancos de Dados, pois permite a definição de entidades que incorporam às suas estruturas o respectivo comportamento, viabilizando a construção de sistemas adaptáveis às necessidades do mundo real, conforme já demonstram na prática alguns sistemas gerenciadores de Banco de Dados Orientados a Objetos (SGBDOOs). Este artigo apresenta o desenvolvimento de um ambiente de modelagem de um Banco de Dados Orientado a Objetos (BDOO), a partir de um modelo de dados baseado em rede semântica estendido a objetos.

I - INTRODUÇÃO

A evolução experimentada pela área de Banco de Dados (B.D.) nas duas últimas décadas foi fortemente influenciada pelo afluxo gradual dos inúmeros modelos desenvolvidos para projetos de B.D., refinados a cada nova metodologia proposta. Primeiramente vieram os modelos clássicos de dados : hierárquico, redes e relacional (CODD70). Apesar de terem atendido a grande parte das necessidades dos usuários, estes modelos procuravam representar basicamente como os dados eram fisicamente modelados, deixando escapar boa

parte da semântica da aplicação, quase sempre embutida nos programas dos usuários. Os modelos semânticos surgiram a seguir, com o objetivo de preencher esta considerável lacuna entre o nível externo, visão do usuário, e o nível interno. Entre os mais conhecidos destacam-se o ER (CHEN76), SAM*(SU83), TAXIS (MYLO82), MORSE (BOUZ86), GALILEO (ALBA85) e muitos outros. O principal avanço introduzido pelos modelos semânticos foi a abstração estrutural, a exemplo dos conceitos de generalização, classificação e agregação. Porém, carecendo do aspecto comportamental relativo às entidades modeladas, deixam grande lacuna no que diz respeito a manutenção da base de dados.

Apesar do avanço possibilitado por estes modelos, muitas limitações permaneceram, principalmente devido à estrutura física baseada em registros, que obrigava a representação de dados homogênea tanto horizontal como vertical, nem sempre desejada. Estas limitações agravaram-se ainda mais com o surgimento das aplicações ditas não convencionais, tais como : sistemas CAD/CAM, Automação de Escritórios, BD Textuais e Pictóricos, Sistemas Cartográficos, e outros que, por envolverem o processamento de dados complexos, embutidos muitas veses em um grande número de transações longas, podem ser enquadrados nesta categoria.

Segundo Dittrich (DITT86), uma base de dados pode ser considerada O.O se for capaz de representar qualquer entidade do mundo real, independentemente de sua estrutura e complexidade, através de um objeto daquela base com as suas respectivas operações . A O.O. concentra seus esforços na busca de modularidade, simplicidade e na capacidade de evolução de código, a fim de possibilitar que novos requisitos possam ser incorporados ao sistema, sem levar em conta detalhes de implementação. Procura também diminuir o "gap" semântico entre o mundo real e as soluções projetadas, empregando-se níveis distintos de abstração, que vão

refinando-se sucessivamente, partindo sempre do mundo real para o das soluções e nunca ao contrário. Os SGBDOOs dentro da tendência mundial de integração de ambientes, transforma-se em componente importante no âmbito computacional, já que combinam as funcionalidades dos SGBDs com os conceitos de O.O., tornando-se o repositório ideal para as informações que são compartilhadas por múltiplos usuários, mútiplos produtos e em diferentes plataformas. Inúmeros SGBDOOs foram construídos nos últimos anos, com vistas a esses objetivos, tais como : ORION (BANE87), IRIS (WILK90), O_2 (BANC89a), GemStone (PENN87), POSTGRES (STON90), etc. Entretanto, não existe ainda um modelo de dados único em O.O., apresentando cada um desses sistemas o seu próprio modelo de dados.

Se a tarefa de projetar BDs já era difícil para os usuários nos modelos tradicionais, o problema vem se agravar ainda mais com esse novo direcionamento, haja visto o aumento de complexidade das novas aplicações e dos conceitos embutidos no paradigma da O.O.

Neste artigo é descrito um modelo de BDOO, bem como a implementação de uma ferramenta que possibilite, de forma amigável, a aquisição e a manutenção do esquema do BD que mapeie o modelo em questão. Este módulo é componente de um ambiente integrado do SGBDOO em desenvolvimento no IME. O presente artigo é estruturado da seguinte forma : em II são levantados os principais conceitos de O.O.; em III é feita uma descrição do modelo conceitual MORSO (Modelo em Rede Semântica para Objetos); em IV é apresentado o desenvolvimento da ferramenta de aquisição e manutenção de esquema, suas características e detalhes importantes de implementação; em V é feita uma conclusão geral sobre o andamento do projeto ora em estudo.

II - PRINCIPAIS CONCEITOS DA ORIENTAÇÃO A OBJETOS

Neste parágrafo descreveremos os principais conceitos da O.O, cujas características devem estar presentes em todo sistema

classificado segundo esse paradigma. São eles:

A - TIPO ABSTRATO DE DADOS (TAD): Abstração de dados é o mecanismo pelo qual modela-se ao mesmo tempo a estrutura de dados e as operações sobre ela. Assim, os objetos consistem de uma interface externa e uma implementação interna, que é encapsulada. A interface é constituída de um conjunto de operações que se aplicam sobre o objeto, formando a sua parte visível. A implementação possui a representação dos dados e das operações permitidas sobre eles, isto é, o código que implementa a operação

O objetivo de se esconder os dados é permitir uma certa independência entre o comportamento do objeto e a sua implementação. As principais vantagens do TAD são : permite uma melhor modelagem do mundo real, separa a implementação da especificação e permite que sistemas sejam extensíveis.

B - CLASSE : Classe é o construtor mais comumente usado para definir tipos abstratos de dados nas linguagens de programação orientada a objetos. A Classe incorpora a definição da estrutura e as operações permitidas sobre os elementos pertencentes àquela classe. Assim, através de uma classe define-se um TAD (KHOS90). Entretanto, a classe é mais que um TAD, pois ela incorpora dois aspectos importantes tais como: uma fábrica de objetos, ou seja, gera novas instâncias de objetos, e de um depósito de objetos, isto é, armazena todos os objetos de uma classe. A definição de uma classe é formada também por um conjunto de métodos, que definem o seu comportamento. Os métodos nada mais são do que um conjunto de funções que executam alguma operação sobre as variáveis de instância dos objetos. Para se ativar um método, deve-se enviar uma mensagem ao objeto.

C - SOBRECARREGAMENTO E "BINDING" DINÂMICO : Sobrecarregamento é a facilidade que permite definir operações de mesmo nome em classes diferentes, porém com semântica e implementação distintas. O

sobregarregamento está ligado ao "binding" dinâmico, significando que o sistema associará o seletor da mensagem ao método que o implementa em tempo de execução.

D - Herança : A idéia essencial da herança é fornecer um mecanismo simples e poderoso para definir novas classes, de forma que estas herdem as propriedades das classes existentes. Na maioria das linguagens orientadas a objetos, uma classe herda tanto os métodos, aspecto comportamental, como as variáveis de instância, aspecto estrutural. A herança fornece um bom mecanismo para a organização da informação, entretanto a sua mais importante contribuição é o compartilhamento ou reutilização de código. Se uma classe só possuir uma única superclasse imediata esta herança é denominada de herança simples , cuja hierarquia de classes forma uma árvore; no caso de mais de uma superclasse imediata, a herança é chamada de herança múltipla , e a hierarquia de classes é estruturada sob forma de um grafo acíclico direcionado (GAD).

Como as propriedades (variáveis de instância e métodos) de uma classe são formadas pela união das propriedades de todas as suas superclasses e das propriedades definidas na própria classe, pode ocorrer que classes antecessoras possuam propriedades com o mesmo nome, mas com semântica totalmente diversa. Quando a classe tenta herdar a mesma propriedade de superclasses distintas, surge o que é denominado de conflito de herança.O principal problema da herança múltipla são as estratégias para a resolução deste conflito, estudadas em (ROSS91).

E - IDENTIDADE DO OBJETO : Num sistema plenamente O.O., a cada objeto deve ser associado uma identidade a qual permanece como identificador, independentemente da sua estrutura ou do seu estado. É um conceito interno do objeto, cujo propósito é fornecer um meio de representar a sua individualidade, independentemente de como é acessado, do seu conteúdo e onde se encontra. A técnica

mais poderosa para suportar identidade é através de "surrogates", cujo identificador único e global é gerado automaticamente para cada objeto, de forma que este seja completamente independente do seu estado e endereço do objeto.

Bancilhon (BANC89b) propõe que os SGBDOOs devam satisfazer a dois critérios : incorporar os principais conceitos da O.O. e possuir as facilidades dos SGBDs "tradicionais", a exemplo de (KHOS90) : persistência, controle de transações e de concorrência, recuperação, linguagem de consulta, restrições de integridade, segurança, desempenho e versionamento. Os SGBDOOs buscam ainda a completude computacional, significando que qualquer função computável deve poder ser programada através do sistema.

III - MORSO - MODELO EM REDE SEMÂNTICA PARA OBJETOS

O MORSO (XAVI89) (XAMO90) é um modelo conceitual O.O, cujo objetivo é capturar os objetos do mundo real, utilizando-se de uma Rede Semântica, extensão do modelo MORSE (BOUZ86), para a representação do conhecimento. O MORSO consiste de uma dupla onde RS é a Rede Semântica que permite modelar conceitualmente a aplicação, e LC é a Linguagem Conceitual que permite manipular a rede, definindo e alterando o esquema conceitual do modelo. A Rede Semântica possui uma representação gráfica, através de um grafo acíclico orientado consistindo de nós e arcos, e uma representação interna, proposta para ser executada através de uma Base de Conhecimento (Base de Fatos e Regras), o que confere à rede um mecanismo de dedução. A modelagem dos objetos emprega os conceitos básicos de orientação a objetos. A figura (1) ilustra a representação gráfica de um objeto para o MORSO.

A rede MORSO, formalizada em (XAVI89), é definida pela quádrupla :**MORSO ::= < CN, CA, CR, CO >** onde :CN : Categoria de Nós; CA : Categoria de Arcos; CR : Categoria de Restrições de

Integridade; CO : Categoria de Operações em Tipos Básicos.

As categorias serão resumidamente descritas a seguir :

A) Categoria de Nos (CN) : os nós representam os objetos modelados pela rede. A rede admite duas categorias de nós : classe de objetos (cl), que representa a descrição das classes dos objetos, e instância de objeto (io), que representa a ocorrência de um determinado objeto.

B) Categoria de Arcos (CA) : os arcos são responsáveis pela interligação dos diversos nós, indicando por meio da categoria e do sentido, a classificação dos objetos que estão sendo ligados. Os principias arcos são : agregação (a), servindo como construtor de tuplas; generalização (g), indicando uma relacionamento "É-UM"; conjunto (s) e lista (l), utilizados para construir conjuntos e listas respectivamente.

C) Categoria de Restrições (CR) : as restrições representam uma informação adicional acerca da classe ou da instância do objeto. O MORSO classifica as restrições em : restrições associadas ao valor do objeto atômico e restrições nas associações entre objetos.

D) Categoria de Operação (CO) : determina as operações permitidas sobre os tipos básicos, tais como : projeção, seleção e junção para tuplas, união e interseção para conjuntos, etc.

Com o objetivo de fornecer maior flexibilidade à rede, no que diz respeito a sua implementação, algumas alterações foram acrescidas ao MORSO (ROSS91) :

A - CLASSES : o modelo é constituído de objetos. Todo objeto possue uma descrição de seu estado e das operações que ele pode realizar, denominada classe. Neste trabalho, a exemplo do ORION (KIM90a), consideramos a classe como sendo formada por uma agregação de variáveis locais, chamadas no modelo de variáveis de instância, associadas a domínios que tanto podem ser outras lasses como tipos primitivos definidos no sistema. A forma do

objeto passa a ser descrita pelas variáveis de instância e o comportamento, pelos métodos. As classes não são dependentes de outras classes, possuindo existência própria. Uma classe pode ser referenciada por qualquer número de classes. A ligação entre uma classe e suas classes componentes passa a ser feita através de suas variáveis de instância. Esta ligação está relacionada ao compartilhamento ou não de objetos que é dependente da semântica das classes componentes, e pode ser classificada em : ligação por referência e ligação por dependência.

1. **Ligação por referência** : o objeto referenciado através de uma ligação por referência não possui um vínculo de propriedade com o objeto que o referencia, isto é, este objeto pode ser referenciado por outros objetos (compartilhamento de objetos).

2. **Ligação por dependência** : esta ligação representa uma composição, isto é, o objeto é composto de uma agregação de outros objetos, existindo uma relação de propriedade exclusiva entre as partes do objeto. Este tipo de ligação está presente no ORION com o nome de "ligação composta", sendo utilizado para a criação dos objetos compostos. A ligação por referência e a ligação por dependência são a base para a construção dos objetos compostos. Existe, no entanto, a necessidade de se propiciar ao usuário um mecanismo que permita a especificação de "classes dependentes", isto é, classes cuja existência sejam dependentes da existência de outra classe. Adotou-se a solução da criação de valores complexos, semelhante ao encontrado no O_2. A vantagem desta solução está na simplicidade de definição, pois o usuário não precisa definir uma nova classe quando na verdade deseja definir apenas um valor composto, visto que não haverá qualquer tipo de compartilhamento, nem da sua definição nem das suas instâncias. A definição destes valores passa a fazer parte da definição da classe principal.

B - **OBJETO COMPOSTO** : o paradigma da O.O., na sua forma

tradicional, é suficiente para representar uma coleção de objetos relacionados. A hierarquia de classes captura o relacionamento "É-UM" entre uma classe e suas subclasses. Entretanto, a hierarquia de classes não captura o relacionamento "É-PARTE-DE" entre um objeto e os objetos por ele referenciados. Este relacionamento é representado pelo conceito de objeto composto. Objeto composto é uma coleção de instâncias relacionadas, formando uma estrutura hierárquica de estrutura bidimensional (KIM90b). O objeto composto introduz uma restrição de integridade no modelo O.O. através da noção de objeto dependente. Um objeto dependente é um objeto que depende da existência de um outro objeto. Com isto, a instanciação de um objeto composto deve ser realizada de cima para baixo, ou seja, o objeto raiz de uma hierarquia de objeto composto deve ser criado primeiro.

O MORSO apresenta uma definição para a semântica de evolução do esquema. Esta evolução é composta de uma conjunto de propriedades do esquema denominadas de "invariantes", e de um conjunto de regras que guiam a seleção do melhor caminho para a preservação dessas mesmas propriedades, além das restrições de implementação denominadas de "diretivas" de implementação (ROSS91). No MORSO as mudanças estão divididas em três categorias:

1. Alteração do conteúdo de uma classe : correspondem basicamente às operações de adição e exclusão das variáveis de instância, métodos e restrições das classes.
2. Alteração de um arco : corresponde à inclusão e exclusão de um arco de generalização entre duas classes.
3. Alteração de uma classe : operação que permite a exclusão e inclusão de classes no esquema.

IV - MAME : MÓDULO DE AQUISIÇÃO E MANIPULAÇÃO DE ESQUEMA

Inicialmente o modelo MORSO foi desenvolvido com planos para

ser construido em PROLOG, razão pela qual cada um dos seus conceitos (arcos,nós,restrições, etc) foi estruturado sob forma de claúsulas de Horn. Ao se dar início a fase de desenvolvimento do módulo, verificamos que uma série de pré-requisitos considerados de relevada importância para o projeto (ROSS91) não seriam satisfatórios em PROLOG, fazendo-nos direcionar a implementação do sistema para a linguagem C. Com esta nova abordagem, as representações das estruturas básicas do modelo anteriormente previstas sofreram alterações, sendo apresentadas a seguir:

A - OBJETO : Um objeto na rede MORSO é definido como tendo a seguinte forma :

ioC"Identificador", "Versão", "Nome_Classe", "Valor").

A representação proposta neste trabalho é mostrada na figura (2.a) onde :

1. Identificador : gerado pelo sistema, é baseado no conceito de "surrogates", a partir da data e hora de criação da classe .

2. Tamanho : tamanho em bytes do objeto. Como os objetos possuem tamanho variável, este campo é utilizado pelo gerenciador de objetos para a sua manipulação.

3. Identificador da Classe : como todo objeto é uma instância de uma classe, o sistema deve possuir um modo de identificar a que classe ele pertence.

4. Versão : apesar deste trabalho não ter abordado o versionamento de objetos, ele já está previsto no modelo, possibilitando o seu desenvolvimento posterior.

5. Valores : Corresponde aos valores assumidos pelos diversos componentes do objeto. A determinação exata se os valores reais dos componentes dos objetos ou seu identificador serão armazenados é uma decisão a ser tomada pelo módulo responsável pelo armazenamento dos objetos. Nesse módulo serão utilizadas técnicas de agrupamento entre objetos ("cluster") relativas as suas

superclasses e componentes. Este trabalho, bem como os métodos de acesso a esses objetos, está em desenvolvimento (FREI91).

B - CLASSE : Uma classe foi definida no MORSO como tendo a seguinte forma geral :

"cl("nome_classe","nome_tipo",["lista de operadores"]).

Considere o exemplo:

cl(aluno, tupla, [matricula, dar_nota, formar]).

O conceito inicial propunha que a classe possuisse apenas o seu nome e tipo. Logo de inicio surgiu o seguinte problema : considere no exemplo anterior a definição da classe "aluno". Analisando-se somente a definição, não encontramos os campos correspondentes às tuplas de aluno. Esta informação, pelo modelo, está armazenada nas instâncias de aluno. Esta modelagem exige que, ao se analisar uma classe busquemos pelo menos uma de suas instâncias e, portanto, na definição de uma classe deve-se incluir pelo menos uma instância desta classe.

Outro conceito introduzido pelo MORSO é o de classe componente, isto é, uma classe é formada pela agregação de classes componentes. Neste trabalho foi proposto que uma classe seja formada pela agregação de variáveis de instância associadas a domínios, que tanto podem ser outras classes como tipos primitivos definidos no sistema. A forma geral para a definição de uma classe é ilustrada na figura (2.b), onde :

1. Identificador : identificador gerado pelo sistema, de forma análoga ao gerado para o objeto.

2. Versão : como já mencionado anteriormente, não foi implementado na presente versão.

3. Lista de Superclasses : o modelo proposto, ao contrário do Smalltalk, aceita herança múltipla necessitando para isto, de uma lista ordenada, sem repetição, das superclasses. A implementação proposta é feita através de uma lista encadeada.

4. Lista de Subclasses : a exemplo da lista de superclasses, esta lista contem as subclasses imediatas da classe. O objetivo desta lista é facilitar o caminhamento da consulta.

5.Lista de Restrições : o MORSO propõe uma série de restrições para uma classe, denominadas de "restrições associadas aos arcos de generalização e especialização". Esta restrição associa uma determinada condição a uma variável de instância herdada a partir de uma classe mais genérica.

6. Lista de Campos : como já definido, uma classe é composta de uma coleção de variáveis de instância. A lista de campos é uma coleção de variáveis de instância no seguinte formato :

nome	tipo	tamanho	característica	lista restrições

onde :

1) nome : nome da variável de instância, semelhante ao nome do atributo no modelo relacional. Possui escopo local a classe, isto é, mais de uma classe pode possuir o mesmo valor para nome.

2) tipo : corresponde a identificação do tipo de domínio ao qual esta variável é associada. No modelo proposto pode-se ter os seguintes casos :

- tipo primitivo: inteiro, real, data, longo, etc ;
- valor composto: formado a partir dos construtores Tupla,Lista ou Conjunto;
- classe definida pelo usuário.

Com o objetivo de permitir a manipulação de objetos "multimídia", é previsto o tipo Longo, podendo suportar Texto, Som e Imagem.

3) tamanho : define o tamanho do campo em bytes.

4) característica : este campo informa duas características da variável de instância : unicidade e valores nulos para valores atômicos. Este campo também é usado para o sistema identificar se a ligação é por dependência ou por referência.

5) lista de restrições: estas restrições referem-se as propostas para os atributos atômicos no MORSO : restrições de domínio (valor e intervalo) e restrições de estado.

C - **RESTRIÇÕES** : As restrições no MORSO não são especificadas separadamente de sua estrutura, fazendo parte integrante da definição do objeto. As restrições reproduzem informações adicionais a respeito das classes ou das suas instâncias. Nos modelos de BDOO, não é comum encontrá-las explicitamente definidas, pois o paradigma de O.O. possibilita a inclusão das restrições através dos métodos. O Morso define uma série de restrições, sendo as seguintes implementadas no protótipo atual :

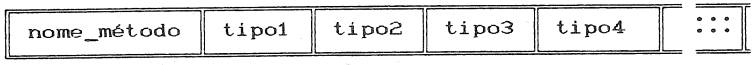
1. Restrições de "domínio" : as restrições de domínio referem-se somente aos objetos atômicos definidos no modelo. As restrições de domínio implementadas são : restrição de valor e restrição de estado;

2. Restrições nas associações entre objetos : são definidas as restrições associadas ao arco de agregação (restrição de unicidade e de existência), a restrição de cardinalidade e a de participação de uma instância numa classe.

As restrições foram implementadas como uma agregação de dois campos: tipo e valor. O campo tipo é constituido de um código numérico indicando qual o tipo de restrição escolhida e o campo valor foi implementado como uma sequência de caracteres.

D - **MÉTODOS** : no paradigma de O.O., os objetos são encapsulados, isto é, os seus valores internos só podem ser acessados através de rotinas específicas definidas para este fim, denominadas de métodos. Os objetos no MORSO são passivos, segundo a classificação de Takahashi (TAKA89), isto é, necessitam que uma mensagem lhes seja enviada para que se tornem ativos. As mensagens ativam os métodos que executam as operações sobre os objetos. O usuário, à semelhança do O_2 , deve especificar uma "assinatura" para os

métodos. Esta "assinatura" ou protocolo é utilizada para a verificação de tipos no momento da ativação do método. A representação do protocolo de um método é dada por :



onde : **tipo1** : tipo do retorno do método;
tipo2, tipo3, etc : tipo dos parâmetros.

Assim como no Smalltalk, o sistema foi desenvolvido supondo-se que o esquema do banco de objetos, isto é, as descrições das classes definidas pelo usuário, estejam na memória "RAM" do micro. Esta obrigatoriedade está diretamente relacionada à eficiência do sistema. A descrição de uma classe está ligada diretamente às suas superclasses. Portanto para se acessar uma variável de instância ou um método é necessário, às vezes, acessar toda a hierarquia desta classe. Para facilitar este acesso às descrições das classes, elas foram organizadas num vetor ("array"), alocado dinamicamente durante o início da carga do sistema, e denominado Diretório, cuja estrutura é ilustrada na figura (2.c), onde :

1. Nome_Classe: nome fornecido pelo usuário no momento de criação da classe. Na criação de um valor composto, o sistema preenche automaticamente com o nome do tipo do construtor escolhido pelo usuário (TUPLA,LISTA ou CONJUNTO).

2. Identificador: o mesmo que é fornecido para o descritor da classe. Esta duplicação facilita o acesso, pois as rotinas internas acessam o diretório via identificador.

3. Número_Arquivo: campo deixado para ser preenchido pelo módulo de armazenamento de objeto (FREI91).

4. Lista_SuperClasse :lista de todas as superclasses de uma determinada classe. Esta lista é utilizada para a instanciação dos objetos (FREI91).

5. Lista_Atributos: visando facilitar as rotinas de

atualização, criou-se uma lista onde ficam preservados os pares: nome da variável de instância e nome da classe de origem de todas as variáveis de instância herdadas por uma classe. Com esta lista, o sistema possui todas as informações necessárias a propagação e resolução dos conflitos de herança que possam surgir. Evita-se a construção de rotinas recursivas, pois só será necessário acessar as superclasses imediatas de cada classe, no caso de conflito de herança.

6. Lista_Métodos : uma lista de métodos herdados à semelhança da Lista_Atributos.

7. Descritor_Classe: é um ponteiro para a estrutura da classe, descrita anteriormente.

A figura 3 ilustra um exemplo da representação das classes Funcionário e Departamento no MAME.

V - CONCLUSÃO

O paradigma de orientação a objetos pode ser definido através da seguinte expressão (KHOS90) :

Orientação Objeto =

Tipo Abstrato de Dados + Herança + Identidade do Objeto

Da necessidade de combinar as propriedades do paradigma de Orientação a Objetos com as funcionalidades dos SGBDs surgiram os SGBDOOs, como sendo (KHOS90):

SGBDOO = Orientação a Objetos + Funcionalidades dos SGBDs

O MAME é parte integrante de um SGBDOO ora em desenvolvimento no IME. Apresenta um interface amigável para aquisição de objetos, atendendo as expectativas iniciais previstas pelo projeto. Deverá ser integrado brevemente ao gerenciador de objetos, responsável pelo acesso, armazenamento e manipulação dos mesmos, e ao módulo de consulta, cuja linguagem encontra-se em fase de especificação.

Os estudos sobre Bancos de Dados Orientado a Objetos

encontram-se ainda em fase de padronização. Os anos 90 despontam como a década em que os SBGBDOOs serão exaustivamente pesquisados em busca de padronização e consolidação desta nova tecnologia.

Referências Bibliográficas

(ALBANO85) -ALBANO, A.; CARDELLI, L; ORSINI, R. "**Galileo: A Strongly-Typed Interactive Conceptual Language**". ACM Transactions on Database Systems, 10(2), June, 1985.

(BANC89a) - BANCILHON,F. et alii. "**A Query Language for the O_2 Object-Oriented Database**", in Proc. Second Workshop Database Programming Languages, 1989.

(BANC89b) - BANCILHON, F. . "**Query Languages for Object-Oriented Database Systems : Analysis and a Proposal**". 4º Simpósio Brasileiro de Banco de Dados, Campinas - S.P., abril de 1989.

(BANE87)- BANERJEE,Jay et alii. "**Data Model Issues for Object-Oriented Aplications**".ACM Transaction on Office Information Systems, Vol. 5, No 1, Janeiro 1987.

(BOUZ86) - BOUZEGHOUB, B. . "**Un Systeme Expert en Conception de Systemes D'Informations**". Dissertação de Doutorado. L'Université de Paris VI, 1986.

(CHEN76) - CHEN, P. . "**The Entity Relationship Model - Toward a Unified View of Data**". ACM TODS, Vol. 1, Num. 1, 1976.

(CODD70) - CODD, E. . "**Relational Model of Data for Large Shared Data Bank**". ACM TODS, Vol 13, Num. 6, 1970.

(DITT86) - DITTRICH, K.R. ."**Object-Oriented Database Systems**". Proc. 5th ER Conference 1986, North Holland, 1987.

(FREI91) - FREITAS, L. O. ."**Esquema Físico para Armazenamento de Objetos Complexos**". Tese de Mestrado, IME, (em desenvolvimento).

(KHOS90) - KHOSHAFIAN, S & ABNOUS, R. "**Object Orientation Concepts, Languages, Databases, User Interfaces**". John Wiley & Sons,Inc.,1990.

(KIM90a) - KIM, W. et alii. "Architecture of the ORION

Next-Generation Database System". IEEE Transactions on Knowledge and Data Engineering, Vol. 2, Nº 1, Março 1990.

(KIM90b) - KIM, WON. "**Object-Oriented Databases: Definition and Research Directions**". IEEE Transactions on Knowledge and Data Engineering, Vol. 2, Nº 3, Setembro, 1990.

(MYLO82) - MYLOPOULOS, J.; BERNSTEIN, P.; WONG,H.A. . "**Language Facility for Designing Database Aplication**". ACM TODS, Vol. 5, Num. 2, 1982.

(PENN87) - PENNEY, J. & Stein J. ."**Class Modification in the GemStone Object-Oriented DBMS**". OOPSLA, 1987.

(ROSS91) - ROSSETTI, L.C. "**Linguagem de Definição e Manipulação de Esquema num Banco de Dados Orientado a Objetos**". Tese de Mestrado, IME , Rio de Janeiro, BR, fevereiro,1991.

(STON90) - STONEBRAKER, M. et alii. "**The Implementation of POSTGRES**". IEEE Transactions on Knowledge and Data Engineering. Vol.2, Nº. 1, March 1990.

(SU83) - SU,S. Y. ."**SAM*** **: A Semantic Association Model For Corporate**". Inf. Science, Num. 29, 1983.

(TAKA89) - TAKAHASHI, Tadao. "**O Paradigma de Objetos: Introdução e Tendencias**".VIII Jornada de Atualização em Informática, Urberlândia, 1989.

(XAMO90) - XAVIER, C.M.S. & MOURA, A.M.C."**Modelagem Conceitual Orientada a Objetos para Aplicações Não Convencionais em Banco de Dados**". 5º Simposio Brasileiro de Banco de Dados, Rio de Janeiro - RJ, abril,1990

(XAVI89) - XAVIER, Carlos Magno da Silva. "**Modelagem Conceitual Orientada a Objetos para Banco de Dados**". Tese de Mestrado, IME, dezembro,1989.

(WILK90) - WILKINSON, K. et alii. "**The Iris Architecture and Implementation**". IEEE Transactions on Knowledge and Data Engineering, Vol.2, No.1, March 1990.

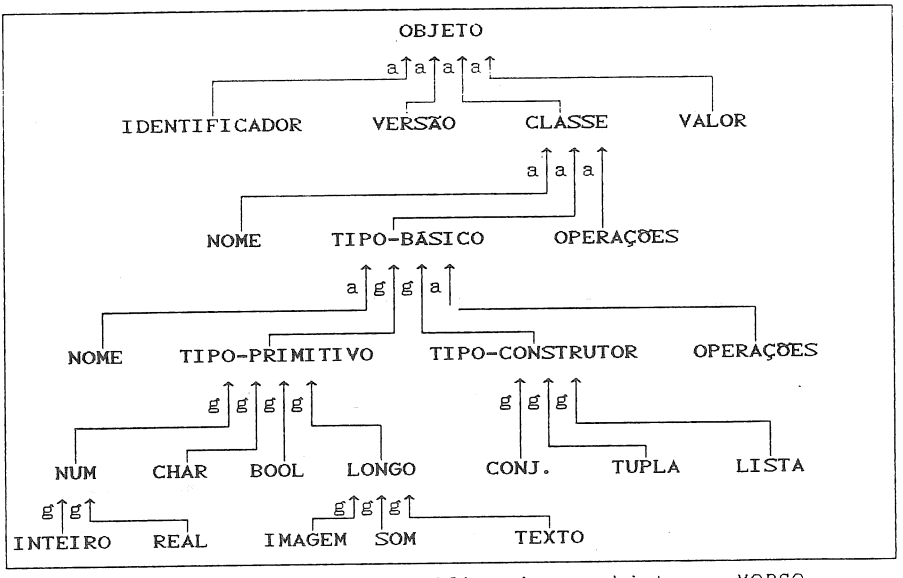


FIGURA 1: Representação gráfica de um objeto no MORSO.

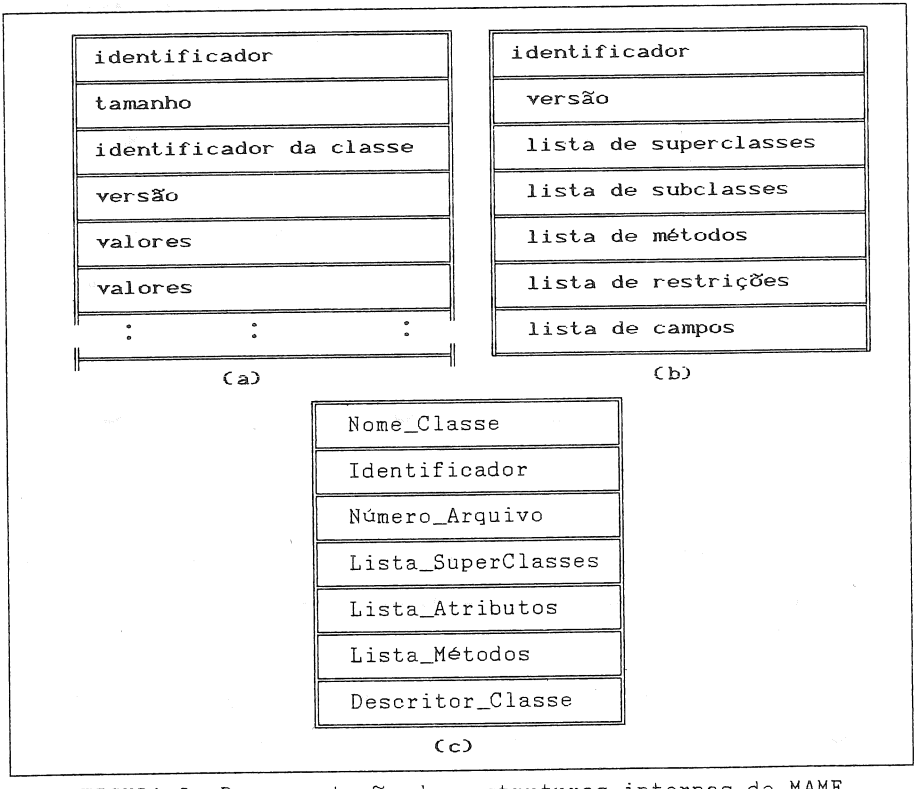


FIGURA 2: Representação das estruturas internas do MAME

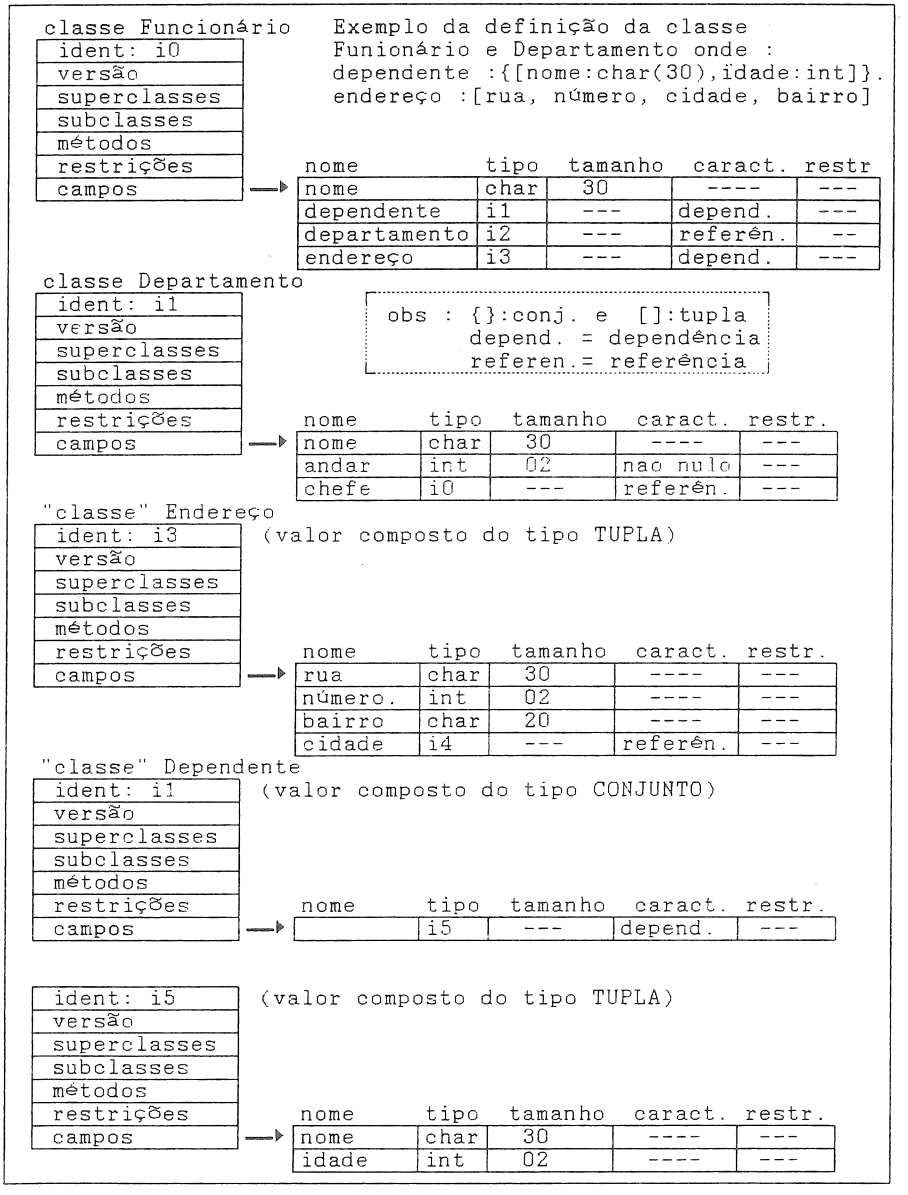


FIGURA 3: Exemplo de representação de classes no MAME