BANCO DE DADOS ESTATÍSTICOS ORIENTADO A OBJETOS

José Sant'Anna Bevilaqua

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DE SISTEMAS E COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Jano Moreira de Souza, Ph.D.
(Presidente)

Profa. Marta Lima de Queiroz Mattoso , DSc.

Dr. Antonio Cláudio de Carvalho Monteiro da Silva, Ph.D.

RIO DE JANEIRO - RJ

Março de 1994

BEVILAQUA, José Sant'Anna

Banco de Dados Estatísticos Orientado a Objetos [Rio de Janeiro] 1994viii, 199p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1994)

Tese: Universidade Federal do Rio de Janeiro, COPPE

1.Banco de Dados Estatísticos I. COPPE/UFRJ II. Título (Série)

Resumo da tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M.Sc.).

Banco de Dados Estatísticos Orientado a Objetos

José Sant'Anna Bevilaqua

Março de 1994

Orientador: Prof. Jano Moreira de Souza

Programa: Engenharia de Sistemas e Computação

Um ambiente para Banco de Dados Estatísticos, montado sobre um Sistema Gerenciador de Banco de Dados Orientado a Objetos, é apresentado.

Utilizando os conceitos de trabalho cooperativo apoiado por computador propõe-se um projeto de BDEOO onde grupos de usuários possam elaborar em conjunto uma pesquisa estatística, recebendo treinamento e orientação do tutor, um especialista no ambiente. A interface do usuário foi desenvolvida com técnicas de programação visual aplicadas a sistemas de computadores permitindo a consulta de metadados de objetos estatísticos e definindo critérios para seleção de informações através de componentes gráficos como listas e botões.

Um protótipo foi implementado sobre uma plataforma SUN/SPARC com os softwares UNIX, MOTIF e XWindows usando o SGBDOO O_2 para estudar a efetividade do modelo.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of requirements for the degree of Master of Science (M.Sc.)

Statistical Object Oriented Database

José Sant'Anna Bevilaqua

March, 1994

Thesis Supervisor: Prof. Jano Moreira de Souza

Department: System Engineering and Computer Science

A Statistical Database environment is presented built upon an Object Oriented Database System.

Using the concept of Computer Supported Cooperative Work, a SDBOO project is proposed where users may work together on statistical research and can be trained and oriented by an environmental expert tutor.

The user's interface developed using visual programming techniques applied to computer system allow the user to search statistical object metadata and defined criteria to select information on graphical components such as buttons and lists.

A prototype has been implemented on a SUN/SPARC platform with UNIX, MOTIF and XWindows using DBMSOO O_2 to study the effectiveness of the model.

ÍNDICE

1. INTRODUÇÃO

A análise estatística é a parte da estatística que se preocupa com a análise dos dados tomados numa determinada população para, com base nestes dados, tentar tirar conclusões, fazer ilações ou predições. Com a análise estatística o administrador, o político, o pesquisador, enfim, todos numa sociedade podem conhecer fatos com uma explicação sistemática e avaliarem objetivamente políticas e planos.

Análise estatística pode ser entendida como o conjunto de ferramentas de análise numérica de dados que permitem interpretar a complexidade das atividades rotineiras. Esta técnica se assenta sobre coleções de informações, preferencialmente grandes, para permitirem a estabilidade na inferência dos fenômenos. Com o incremento na capacidade de manuseio de dados em equipamentos de computação, a análise estatística experimentou grande desenvolvimento científico, acrescentando às ferramentas de análise descritiva sofisticados modelos matemáticos.

Os bancos de dados estatísticos são aqueles bancos de dados numéricos que contêm informações sobre classes de objetos, normalmente organizados com objetivo de assistir à análise estatística. Permitem manter os dados provenientes de grandes experimentos, em pesquisas, investigações ou censos de forma adequada a serem selecionados e utilizados pelas ferramentas de análise empregadas.

As novas tecnologias de banco de dados, como os sistemas de banco de dados orientados a objetos, trazem um enriquecimento semântico da informação que pode ser utilizado para superar os problemas de interpretação de conteúdo, freqüentes no manuseio de bancos de dados estatísticos. Dotados com mecanismos de herança, atributos virtuais, encapsulamento, entre outros, os SGBDOOs apresentam-se com mais recursos para o manuseio de bancos complexos como os estatísticos.

Outra tecnologia emergente, a do trabalho cooperativo apoiado por computador, permite superar o isolamento do indivíduo que trabalha com um computador para integrá-lo a grupos de trabalho, como os de natureza multidisciplinar muito comuns na análise social. Com esta tecnologia é viável a formação de grupos para trabalharem de forma síncrona ou assíncrona em locais geograficamente distantes ou para receberem treinamento apoiado por computador.

A programação visual aplicada a sistemas de computadores é outra técnica que vem resolvendo o problema da interação do homem com o computador. Os recursos para o uso intensivo de técnicas gráficas diminuem o esforço de programação e melhoram a qualidade das aplicações de computadores. O uso de interfaces gráficas facilita o uso dos sistemas, diminui os erros de operação, diminui o tempo de treinamento do usuário e aumenta sua percepção para as informações apresentadas o que resulta em aumento da produtividade do trabalho suportado em computadores.

Uma proposta de revisão do ambiente para Banco de Dados Estatístico utilizando um sistema gerenciador de banco de dados orientado a objetos, técnicas de trabalho cooperativo apoiado por computador e o emprego intensivo de uma interface gráfica, é apresentada nesta tese. A avaliação feita no projeto piloto procurou explorar estas novas tecnologias num modelo de dados que reuniu a informação numérica aos metadados necessários à indexação das informações. Com a junção do modelo de metadados ao do armazenamento numérico da informação, a busca de informações ficou mais simples e pode ser feita através de descritores, montados segundo um vocabulário hierarquizado e controlado. O uso de descritores permitiu ainda a substituição dos incômodos sistemas de códigos utilizados na classificação das informações, de difícil interpretação e utilização, por uma representação mais natural e próxima a da interpretação humana.

A tese está organizada em oito capítulos onde são apresentados: uma revisão literária dos assuntos que sustentam a proposta de revisão do ambiente do BDE; uma proposta de revisão do banco de dados; o detalhamento do projeto piloto de avaliação; e, ao final, uma avaliação sobre o trabalho com conclusões que ajudem o desenvolvimento de novos trabalhos.

No capítulo 2 é apresentada uma revisão da literatura sobre Bancos de Dados Estatísticos, apresentando-se sua conceituação. É discutido o problema de consultas em BDEs e são apresentadas as linguagens de consulta e as ferramentas para a análise estatística. Ao final, são apresentados alguns exemplos de sistemas de consulta sobre BDEs e os problemas existentes na área.

O paradigma da orientação a objetos é o assunto do capítulo 3. Os conceitos são vistos a partir dos dois principais documentos da área: o "Manifesto dos Sistemas Orientados a Objetos" e o "Manifesto dos Sistemas de Banco de Dados de 3ª Geração", também conhecido como "contra-manifesto". No final é apresentado um resumo

dos conceitos geralmente aceitos como característicos da orientação a objetos e a arquitetura dos sistemas de banco de dados orientados a objetos.

No capítulo 4 é revista a literatura sobre trabalho cooperativo apoiado por computador (CSCW) onde são apresentadas a taxionomia e as características dos projetos destes sistemas. Ao término do capítulo procura-se estabelecer o emprego desta técnica na área dos bancos de dados estatísticos.

No capítulo 5 é feita uma revisão dos conceitos de sistemas de programação visual aplicados a computadores. A dissertação é iniciada pela apresentação e arquitetura das interfaces do usuário. A seguir são apresentados os conceitos, os princípios para construção, a representação formal e discutido o problema da ambigüidade de ícones. Conclui-se apresentando os conceitos e mostrando-se exemplos de linguagens de programação visual.

No capítulo 6 é apresentada a proposta desta tese qual seja, uma concepção de ambiente para Banco de Dados Estatístico, de acordo com os princípios do paradigma da orientação a objetos e com a utilização de uma interface gráfica para acesso aos objetos estatísticos. A proposta inclui na construção do BDE as funções de um sistema de metadados com o propósito de expandir a capacidade de navegação nos descritores da informação.

O capítulo 7 versa sobre o projeto piloto do BDE que foi desenvolvido, apresentando a estrutura e a estrutura de aplicações implementada. No Anexo B consta a descrição das funções e as telas projetadas para a interface do usuário. São apresentados os recursos demandados pelo projeto piloto e, ao final, são apresentadas observações decorrentes da utilização do O_2, o SGBDOO de suporte do *software*.

A dissertação se encerra no capítulo 8 com as conclusões efetuadas sobre a proposta de um novo BDE. As sugestões reunidas durante o desenvolvimento do projeto são apresentadas visando novos trabalhos na área.

Em anexo, é apresentado um glossário de termos e abreviaturas utilizados e as especificações do projeto piloto implementado. Um manual do sistema com as instruções de uso, as especificações do projeto e a codificação dos programas é disponível a parte, como relatório do projeto TABA.

2. BANCO DE DADOS ESTATÍSTICOS

2.1. INTRODUÇÃO

2.1.1. Processamento Estatístico de Dados

Tradicionalmente, a produção de informações estatísticas, ou seja, o planejamento, a coleta, a computação e a edição de dados estatísticos, tinha como único objetivo a obtenção de tabelas descritivas do objeto da análise. Concluída a apresentação das tabelas, uma nova análise sobre os mesmos dados demandava um novo esforço de computação, quase que equivalente ao anterior. Muito pouco ou nada do trabalho anterior era reutilizado. Este objetivo e sistema de produção estatística frustravam o interesse de exploração da informação, restringindo-a aos limites de uma análise pré-concebida, como também impediam a análise conjunta de informações originadas em produtores independentes.

Graciano de Sá [GRAC80] ao analisar o problema constatou que os usuários de um Sistema Estatístico costumam pressionar os produtores para obter acesso aos dados e a serviços de análise. Entretanto, os grandes volumes normalmente envolvidos acarretam problemas complexos de seleção de dados e de observância dos requisitos de confidencialidade associados aos dados individuais. A solução adotada tem sido a manutenção dos dados coletados e das informações produzidas em bancos de dados, chamados de estatísticos.

2.1.2. Informação Estatística

Conforme Shoshani [SHOS82], as informações de uma base de dados estatísticos dividem-se em atributos de referência e dados mensurados. Os atributos de referência contêm as categorias de um sistema de classificação que identificam ou qualificam um objeto ou as condições em que uma informação foi obtida. Os dados mensurados são aqueles sobre os quais é feita a estatística propriamente dita, ou seja, representam o valor de um objeto, coletado segundo um determinado conceito de coleta.

Os atributos de um BDE também podem ser referidos como variáveis independentes e variáveis dependentes, segundo o critério de classificação e mensuração. Por exemplo, numa pesquisa social a variável mensurada *renda do informante* (variável dependente) pode ser associada com diversos atributos de referência (variáveis independentes) para classificar este valor segundo categorias de sexo, idade, grau de instrução, religião, ocupação e atividade, etc, enquanto, numa estatística de saúde, a variável dependente *frequência cardíaca* pode ser medida diversas vezes ao longo de um esforço físico do mesmo indivíduo (variável independente).

Normalmente, para fins de produção estatística é utilizada a primeira denominação, atributos de referência e dados mensurados, enquanto a segunda é mais utilizada em trabalhos de análise estatística. Para a segunda classificação, uma variável é dependente ou independente de acordo com a ordem adotada para a análise. Alterada esta ordem, uma variável pode mudar de classificação. Mudando o primeiro exemplo acima para uma pesquisa de distribuição dos indivíduos segundo renda, idade e sexo, a variável mensurada *renda do informante* deixa de ser dependente para se transformar em *categorias de renda*, uma variável independente.

As informações estatísticas podem ser armazenadas num Banco de Dados Estatístico (BDE) definido como um banco de dados numéricos que contêm informações sobre classes de objetos, normalmente organizado com objetivo de assistir à análise estatística. Apesar do emprego que o termo teve no passado quando era utilizado para identificar uma coleção de publicações ou um conjunto de tabelas impressas [SADO88], neste trabalho será empregado somente para os dados armazenados em meio legível por processamento eletrônico de dados, preferencialmente residente em computador.

Conforme observou Raffanelli [RAFA88], os BDE diferenciam-se dos banco de dados convencionais em três grandes pontos:

. *estrutura dos dados:* na maioria dos bancos de dados, os modelos são próprios para armazenar estruturas simples de dados como relações, enquanto os BDE necessitam suportar estruturas complexas de dados como tabelas (séries temporais, matrizes de dados ou tabelas de contingência) com grande quantidade de dados, de dados ausentes e com uma estrutura de dados esparsos;

. *manuseio dos dados:* as tradicionais operações de alteração e exclusão de dados e registros são raras ou proibidas nos BDE. O dado estatístico é estável

pois se refere a eventos já ocorridos. As novas informações são adicionadas fazendo com que o banco de dados cresça continuamente;

. *processamento dos dados:* no processamento de dados estatísticos são usadas tanto técnicas de imputação para dados ausentes, como de ajustamento aos limites de aceitação, em substituição aos tradicionais critérios de aceitação e rejeição de informações adotados nos sistemas convencionais de crítica de dados.

Os conceitos de BDE são aplicáveis à análise de experimentos científicos, a dados temáticos organizados sobre uma base geográfica, a dados sobre comércio mundial, a execução de testes de qualidade assistidos por computador sobre linhas de produção [GHOS86] a dados sociais, de saúde e educação, enfim, são úteis para todos os processos onde uma grande quantidade de dados necessite ser manuseada para gerar estatísticas que permitam a análise do processo.

2.1.3. Análise Estatística

Uma análise estatística sobre um BDE é um processo de cômputo de uma informação sobre os dados armazenados. Pode ser feita sobre todos os objetos de uma população, ou sobre uma partição desta população selecionada, segundo critérios definidos por atributos de referência.

Normalmente, é constituída de agregações sobre os dados mensurados, classificados segundo determinadas categorias existentes nos atributos de referência, ou seja, é um cálculo de estatísticas sobre valores controlados por parâmetros. Uma estatística pode ser uma simples verificação de existência da variável dependente no espaço de uma classificação, uma contagem ou uma média de elementos, uma distribuição de frequência ou outra medida descritiva. Uma estatística pode ser também o cômputo de complexos modelos multivariados sobre os dados do BDE.

A análise estatística feita sobre grandes volumes de dados requer: um sistema gerenciador de banco de dados para o manuseio das informações do BDE; uma função para a consulta e seleção de informações, ou seja, uma linguagem de consulta e extração de dados do BDE; e uma ferramenta para a execução da análise estatística propriamente dita.

2.2. CARACTERÍSTICAS

2.2.1. Atributos

Como apresentado anteriormente, os BDE possuem dois tipos de dados: dados mensurados, sobre os quais são feitas as estatísticas; e atributos de referências que descrevem os valores mensurados. Esta distinção é feita em termos lógicos, não implicando na forma física do seu armazenamento como atributos de registros, de tuplas, ou de outros modelos, de acordo com a estrutura utilizada.

Os atributos de referência também podem ser tratados como atributos categorizados quando a referência está contida em algum sistema de classificação. Por exemplo, as pesquisas sociais, ao investigar a renda do trabalho, costumam associá-la a características como a ocupação e o setor de atividade onde o informante exerce ocupação produtiva.

Os atributos de referência também podem ser utilizados para definir a referência espacial de um dado mensurado, associando à informação o local ou região de ocorrência. Além da utilidade para a análise estatística, esta informação é muito utilizada, isolada ou associada com outros atributos, no controle da coleta e edição de dados. Nas pesquisas municipais feitas pela Fundação Instituto Brasileiro de Geografia e Estatística - IBGE, como a série anual sobre agropecuária, a referência geográfica é a principal variável de controle da coleta dos dados, além de ser o conceito mais freqüente de agregação dos valores.

Quando os dados mensurados se referem aos valores coletados pela pesquisa são chamados de microdados. Os dados derivados de estatísticas sobre os microdados são chamados de atributos derivados, valores resumo ou valores agregados. O acesso aos dados mensurados e aos seus agregados, normalmente, é efetuado através dos atributos de referência, utilizados isoladamente ou em conjunto, como uma chave composta de acesso à informação.

2.2.2. Sistemas de Classificação

Classificações são partições de um universo em classes que agrupam seus elementos segundo propriedades comuns, ou seja, definem conjuntos de categorias que representam a dimensão da informação dentro de um determinado domínio. Uma classificação pode ser feita tanto a nível quantitativo quanto a nível qualitativo. As-

sim, podemos ter um atributo de referência como sexo usando duas categorias, homem e mulher, ou classificar ocupação profissional em cerca de mil categorias. Por outro lado, um atributo de valor contendo um dado mensurado, por exemplo, renda, poderá ser estratificado em 10 ou 15 intervalos de variação onde cada um será associado a uma categoria da classificação [MALV89].

Com os sistemas de classificação são feitas as taxionomias dos objetos, dos fatos ou dos atos manuseados numa estatística. O conhecimento dos métodos usados em sua construção é essencial para o manuseio das informações, seja para a análise do comportamento ou para comparabilidade com outras informações.

2.2.2.1. Hierarquias

Muitas vezes os sistemas de classificação adotam estruturas hierarquizadas para descreverem as categorias. As hierarquias, dependendo do conteúdo da categoria, podem utilizar muitos níveis de profundidade [SHOS82]. Desta maneira, as categorias se desdobram sucessivamente em subcategorias até atingirem o menor nível de detalhamento pretendido. Como exemplo temos a classificação adotada na Nomenclatura Brasileira de Mercadorias (NBM), que desce até o sétimo nível de profundidade.

Podemos considerar a distribuição espacial como um caso particular de classificação onde as categorias representam os ítens geográficos. A estrutura desta classificação geográfica utiliza uma estrutura desigual de profundidade: como por exemplo o conceito de região metropolitana que existe em alguns estados e o de distritos, que nem todos os municípios utilizam.

2.2.2.2. Códigos de categorias

Como a utilização de nomes oferece menos segurança e é mais cara do ponto de vista computacional, normalmente, um sistema de classificação é codificado. Um código, normalmente numérico, é utilizado para registrar a informação em substituição ao descritor da categoria. Posteriormente, dentro da função de recuperação de informações do BDE, é feita a decodificação com o uso de tabelas de associação de códigos às categorias.

Essas tabelas de associação muitas vezes são tratadas como arquivos de classificação para incorporarem a semântica necessária à interpretação dos conceitos e métodos usados no sistema de classificação.

2.2.2.3. Versões e conceitos

As tabelas de códigos representam as categorias de interesse em determinado momento de uma pesquisa estatística. Uma modificação no universo analisado, na forma de pesquisar a informação ou mesmo uma mutação conceitual, pode resultar na atualização da classificação e, conseqüentemente, na formação de versões das tabelas de categorias e códigos. A comparabilidade temporal efetuada entre diversas ocorrências de uma pesquisa é restringida ao conhecimento das versões destas tabelas e das sucessões de categorias estabelecidas.

Como todo processo qualitativo, a análise estatística de um objeto é condicionada à definição de padrões para comparação, por exemplo, resultados obtidos em pesquisas anteriores. Caso contrário, o processo será realizado como se fosse o da construção de uma nova informação, independente do acervo disponível, sem comparação com outros objetos da mesma família de dados.

2.2.3. Metadado da Informação Estatística

A necessidade semântica da informação estatística transcende a uma clara denominação dos atributos e interrelacionamentos. Quanto maior for um BDE mais complexa será a tarefa de identificação do conteúdo da Base de Dados e da terminologia adotada para seus atributos. Em sistemas convencionais o metadado chega a constituir um banco de dados separado do BDE, como o existente no IBGE [GUED89] [IBGE90] e no projeto SEEDIS do Lawrence Berkeley Laboratory [MCCA82].

2.2.3.1. Formação do metadado

O metadado, ou seja, a descrição do dado, representa um dos grandes desafios na administração de dados estatísticos. Localizar uma informação ou os dados necessários a uma agregação dentro de um BDE requer ferramentas de acesso e conhecimento complementar ao existente nos dicionários de dados normalmente disponíveis em sistemas gerenciadores de banco de dados. Um acervo de informações estatísticas, como o mantido na Fundação IBGE, descreve, para 70 pesquisas diferentes, 464 dici-

onários de dados que são usados por 779 modelos de arquivos onde existem 8.916 atributos de referência e a 70.878 atributos de mensuração de valor, muitos dos quais semelhantes. Existem, ainda, 114.229 diferentes códigos de categorias. Com essa dimensão, a própria tarefa de administrar o acervo de metadados é inviável sem um sistema específico e computadorizado.

O metadado estatístico é uma descrição sistemática sobre os dados existentes no BDE, de como foram obtidos, de como foram editados e de como eles podem ser recuperados, manuseados ou apresentados. As entidades a seguir descritas constituem seu principal conteúdo:

. *pesquisa* - através da qual o dado foi coletado. Uma pesquisa é descrita através de suas ocorrências, ou seja, das épocas e métodos de coleta e de produção da informações e pela descrição de seu conteúdo;

. *arquivo* - onde o dado encontra-se armazenado. Os arquivos são descritos por sua organização, método de acesso, restrições de utilização, identificação e pelo volume e frequência dos dados existentes;

. *dicionário* - que descreve o formato do arquivo onde o dado é armazenado. Um dicionário descreve os tipos de registros, a ordenação e as chaves de acesso, restrições de acesso e o layout dos dados;

. *atributos* - que caracterizam uma informação. Uma informação atômica ou elementar é descrita pelo formato da representação no arquivo pelo conceito de sua coleta ou formação, pela unidade de mensuração adotada, pelos limites do seu intervalo de variação, pela representação que assume na ausência de informação e pelos valores de categorias admissíveis. Uma informação complexa descreve os elementos que a compõem;

. *categorias* - que descrevem o domínio do valor do dado. As categorias são descritas por sua identificação, código, descrição conceitual e relacionadas com as classificações a que pertencem.

2.2.3.2. Indexação da informação

O mecanismo da indexação é utilizado para relacionar o metadado da informação com a informação existente no BDE. Viabiliza a localização dos atributos da

base e das categorias utilizadas nos sistemas de classificação, permitindo a busca de uma informação através do seu conteúdo semântico e, assim, elimina a complexidade da busca da informações em dicionários de dados, efetuada segundo o formato da organização de arquivos. O uso de índices hierarquizados facilita a seleção de variáveis para compor uma futura consulta e permite a administração de categorias e tabelas representativas das classificações.

No projeto de indexação da informação adotado no IBGE, cada elemento é indexado por uma tupla contendo as seguintes entidades:

. *descritor* - conceito inequívoco de um objeto real ou fato, não associado com dimensão, tempo ou local geográfico;

. *modificador* - conceito descritivo da natureza da investigação da informação;

. *complemento* - complementação da ação descrita no modificador.

Os descritores por sua vez pertencem a uma hierarquia cujo maior nível é um *tema* ou um assunto, detalhado por *grupos homogêneos* de informações e, quando necessário, por um *objeto complexo*, onde o mesmo descritor pode ser instância de muitas hierarquias. Desta maneira, usando a generalização de conceitos dos descritores é possível promover um agrupamento de atributos e categorias e reduzir a quantidade de ítens a serem pesquisados. Outra vantagem do processo de indexação é a obtenção dos métodos com que um descritor é pesquisado e assim, permitir a localização da informação dispersa entre pesquisas.

Com estes conceitos podemos indexar, por exemplo as seguintes tuplas:

{tomate, área plantada, cultivo simples}
{tomate, produção de concentrados};

Onde tomate é um descritor, área plantada e produção de concentrados são complementos e cultivo simples é um modificador. Convém observar que a primeira tupla se refere a uma informação produzida por uma estatística agrícola enquanto a segunda é industrial com o que, certamente, o mesmo descritor (tomate), será instância de, pelo menos, dois temas: agricultura e indústria alimentícia.

2.2.4. Organização dos Dados

A organização de um BDE é feita em diversas formas e tecnologias, segundo a dimensão do problema e do nível de recursos disponíveis. Em geral, visa-se minimizar o esforço considerado como típico de recuperação de dados. Esta característica decorre da natureza não volátil da informação de um BDE. O esforço de atualização, quando existe, consiste em agregar novas informações e na correção de erros.

Um problema freqüente na formação de arquivos estatísticos diz respeito à escolha do momento em que o processo de consistência é encerrado e os dados são considerados bons para o processo de agregação e inferência. Como o processo estatístico não é determinístico, a liberação de um arquivo não implica na afirmação de estar totalmente livre de inconsistências. Espera-se que o erro, caso exista, esteja contido abaixo de um nível de aceitação razoável. Assim, pode ocorrer que, após esta liberação, sejam detectadas evidências de inconsistências introduzidas por processos de correção, imputação, ou mesmo não verificadas na depuração efetuada, gerando um problema de manutenção de versões identificadas de arquivos liberados a fim de manter a consistência com os dados anteriormente divulgados.

2.2.4.1. Organização dos arquivos

As principais formas de organização dos arquivos estatísticos costumam ser o formato de arquivos planos ou de arquivos transpostos, conforme apresentado na figura II.1.

Arquivos planos ou tabelas são arquivos organizados seqüencialmente com as informações dispostas em registros representando linhas de uma tabela bidimensional onde cada coluna representa um tipo de informação estatística. Por sua simplicidade são os mais empregados e devem sua popularidade à maneira natural de representação, semelhante às tradicionais tabelas estatísticas. Os arquivos planos, apesar de exigirem algum tipo de registro dos valores das referências, diminuem a redundância no armazenamento dos dados. Os arquivos planos são aceitos por quase todas as ferramentas de análise estatística, mesmo por aquelas que dispõem de alguma forma mais elaborada de manuseio de dados.

Arquivos transpostos são coleções de arquivos utilizadas para separar o armazenamento dos atributos de uma relação [TURN79]. Num arquivo totalmente transposto, cada atributo armazenado na relação é transformado em arquivos de apenas

uma coluna e mesmo número de registros. Uma alternativa intermediária de estruturação é a chamada transposição agrupada, onde os atributos, cujo o uso em conjunto seja freqüente, são mantidos no mesmo arquivo. Esta técnica é pouco utilizada devido a dificuldade de se afirmar, a priori, quais as coleções de atributos que terão esta característica.

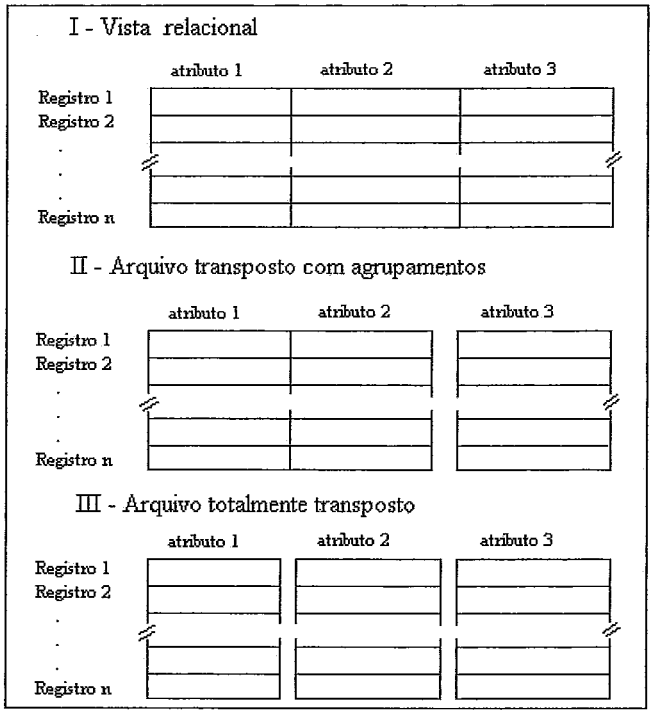


Figura II-1 - Transposição de arquivos

O uso de arquivos transpostos permite limitar o acesso às informações por fazê-lo apenas aos atributos necessários. Esta técnica propicia uma significativa economia de acessos físicos a disco nas consultas que manuseiam grandes quantidades de dados. Além do ganho em desempenho pela redução das operações de E/S, o enfileiramento de atributos de mesma natureza viabiliza o emprego de técnicas de compressão de dados que, por sua vez, poderão reduzir ainda mais o número destas operações. O RAPID, SGBD desenvolvido no Statistics Canada [TURN79], armazena os dados em arquivos transpostos e comprime a representação das informações através do conhecimento do domínio de sua variação. Por exemplo, se o domínio de um atributo compreende 12 valores distintos, utiliza 4 bits para representá-los. Entretanto, como as linguagens de programação costumam utilizar limites de bytes para representação de atributos, a compressão em bits costuma exigir rotinas especiais para acesso aos dados, nem sempre compatíveis com a linguagem utilizada.

2.2.4.2. Conteúdo dos arquivos

Os arquivos estatísticos também podem ser classificados quanto ao tipo dos dados que contém. A classificação é feita em arquivos de microdados, arquivos resumos e em arquivos de amostras.

Arquivos de microdados são os arquivos que reproduzem, fielmente, todas as informações coletadas numa pesquisa e consideradas boas para utilização. Os microdados são a fonte primária para todas as estatísticas de uma pesquisa e constituem a principal informação de um BDE. Pelo volume e por identificarem os dados individuais, em geral, tem utilização restringida aos produtores de estatísticas e não são liberados para o uso público. A exemplo dos demais arquivos estatísticos, para a utilização dos atributos de um arquivo de microdados é necessário um completo conhecimento dos conceitos empregados na coleta e das regras de validação utilizadas.

Arquivos resumos ou agregados são os arquivos obtidos com a aplicação de uma função estatística sobre os dados originais ou microdados. Esta função pode ser um critério de partição do arquivo segundo o conteúdo de determinados atributos de referência, ou uma agregação de valores executada sobre uma seleção de referências. Os arquivos resumo, também, são utilizados para viabilizar o manuseio de arquivos muito grandes, onde as estatísticas são calculadas sobre cada partição para, depois, serem juntadas e formarem um resultado final. Nos grandes BDE, a formação de arquivos resumo, fisicamente separados, costuma produzir problemas de administração de dados com a proliferação de termos e conceitos particulares utilizados nestas partições.

Amostras são partições do conjunto de microdados de um universo, onde os dados são desidentificados e selecionados segundo um plano de amostragem que assegure a representatividade deste novo conjunto. Para utilização destes arquivos, é obrigatória a existência de uma descrição completa dos procedimentos necessários à expansão da amostra. Os arquivos amostra podem ser considerados como um caso particular de arquivo resumo.

2.2.4.3. Dados esparsos

Muitas vezes, os arranjos de dados utilizados num BDE costumam produzir arquivos com grande quantidade de valores nulos, formando o fenômeno conhecido como *dados esparsos*. Ocorre, com frequência, na formação de arquivos planos obti-

dos com o cruzamento de muitas classificações, uma vez que os sistemas de classificação, ao tentarem cobrir o universo, descrevem categorias pouco freqüentes ou válidas em apenas poucos espaços. Ao ser realizado o cruzamento, essas categorias são expandidas para todo o modelo provocando regiões com ausência de informação. Também em arquivos transpostos é possível encontrar-se regiões onde um atributo seja nulo, decorrente da ausência de uma qualidade numa região ou época de pesquisa. Para estas situações, procura-se utilizar métodos de compressão adequados, como os do RAPID que eliminam brancos em cadeias de caracteres de tamanho variável.

2.3. CONSULTAS A UM BDE

Os produtores estatísticos administram três conflitos básicos: o primeiro é caracterizado na desproporção entre o volume de dados que coletam com a quantidade de informações que produzem; o segundo, com o intervalo de tempo decorrido entre a coleta e a edição dos resultados da pesquisa, geralmente em forma de tabelas impressas; e o terceiro surge da insatisfação dos usuários com a demanda não atendida por informações específicas, inexistentes nos planos tabulares, ao mesmo tempo em que razões de confidencialidade impõem restrições de acesso às bases de dados estatísticos.

O acesso irrestrito aos dados de um BDE mediante o emprego de uma linguagem de consulta e de uma ferramenta de análise estatística, apesar de parecer a priori como a solução ideal para esses problemas, não costuma ser disponível ao público. A limitação legal de acesso a dados individuais e às dificuldades de manuseio de dados devida à insuficiência semântica das informações restringem as soluções apresentadas pelos operadores de BDE's a sistemas de divulgação de agregados com pouca, ou nenhuma, liberdade de manuseio da informação.

Turner [TURN79] classificou o acesso à BDE's nas seguintes categorias:

. *informação*, quando a consulta requer o acesso a um ou a poucos registros, geralmente para a maioria dos atributos existentes;

. *operacional*, quando uma consulta requer um pequeno número de registros (no máximo 10% do total);

. *estatística*, quando uma consulta requer poucos atributos e um número grande de registros.

As consultas do primeiro tipo, informação, normalmente são bloqueadas num BDE uma vez que não se caracterizam como de natureza estatística.

G.A.Stephenson [STEP88] divide o problema de consulta a banco de dados estatísticos em quatro dimensões:

. o usuário desconhece os dados disponíveis no banco de dados, o que pode significar muito trabalho para localizar a existência de uma informação, quando se acessa um grande banco de dados estatístico;

. o usuário deve escolher uma fonte de informações sem dispor de documentação suficiente, problema típico de busca de uma mesma informação produzida por mais de uma pesquisa (p.ex.: população economicamente ativa);

. o usuário desconhece os detalhes técnicos que distinguem as fontes, como no caso das pesquisas que manuseiam fatos transitórios;

. o usuário necessita de informações adicionais para poder interpretar os dados, como no manuseio de amostras.

Considerando estas dimensões, Stephenson define seis modalidades o acesso à banco de dados conforme figura II-2.

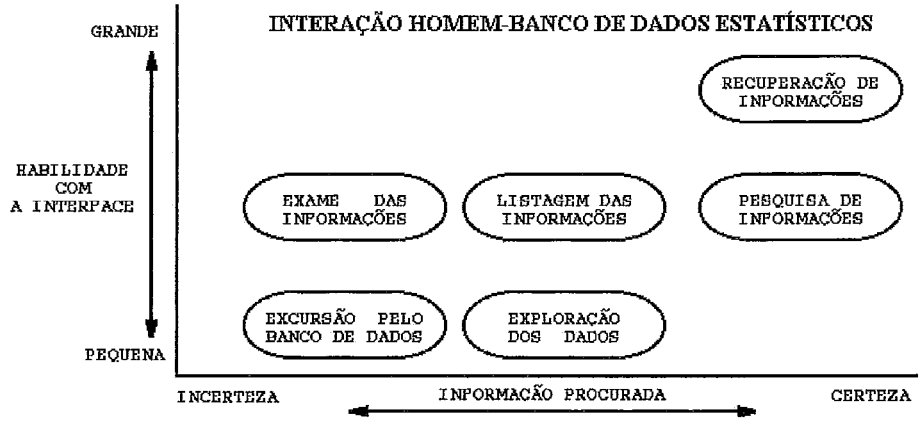


Figura II-2 - Formas de interação Homem-banco de dados estatístico

. *Recuperação de informações*: o usuário sabe o quer e como procurar no sistema e o sistema dispõe da informação procurada;

. *Pesquisa de informações*: o usuário sabe o que procurar, acredita que o sistema possa atendê-lo mas é obrigado a utilizar índices de acesso pois não sabe exatamente onde encontrar a informação procurada;

. *Listagem das informações*: o usuário não tem certeza sobre qual é a informação procurada e tem alguma dificuldade em manusear a interface. Geralmente lista os índices até encontrar indicações que o auxiliem na busca desejada;

. *Exame das informações*: o usuário não tem certeza sobre o assunto procurado e lista todas as informações e índices, mesmo aqueles que não dizem respeito ao assunto específico;

. *Exploração do sistema*: o usuário manuseia o sistema para tentar descobrir seus limites, restrições e possibilidades. Preocupa-se mais em entender a estrutura e o conteúdo do sistema do que as informações propriamente ditas;

. *Excursão pelo banco de dados*: o usuário desconhece o significado das informações e acessa o sistema de forma totalmente desestruturada.

Com a definição destas modalidades, podem ser estabelecidos quatro princípios para a construção de aplicações de consulta à bancos de dados estatísticos:

. a estrutura do domínio do conhecimento deve ser observável;

. o espaço de observação das informações deve ser de fácil utilização;

. o usuário não deve ser obrigado a tomar decisão que não entenda ou não queira;

. o sistema precisa assistir ao usuário, ajudando-o com listas e índices estruturados para que seja dispensável a memorização de códigos e comandos de acesso.

2.3.1. Sigilo e Confidencialidade da Informação

Muitos países, entre os quais o Brasil, dispõem de legislação regulando a atividade estatística onde todas as pessoas são obrigadas a prestar informações para produção de estatísticas, assegurando o sigilo sobre o informante. As informações indivi-

duais são consideradas sigilosas e indisponíveis para qualquer outra finalidade, inclusive a fiscal.

Como é possível obter-se informações individuais através do cálculo de funções estatísticas, como uma soma ou uma média, os mecanismos convencionais de bloqueio utilizados em bancos de dados são ineficazes. Até mesmo uma simples diferença entre dois conjuntos de informações permite recuperar qualquer informação individual. Este problema vem sendo estudado por diversos autores com a apresentação de soluções específicas, não generalizáveis. Entre as principais soluções sugeridas, podemos destacar as seguintes:

. limitar as consultas a um número mínimo de informações, conjugando com um mecanismo de registro das consultas anteriores que impeça a obtenção de informações por diferenças [SCHL80], [DENN79] e [DENN80].

. idem à anterior com limitação para a interseção de conjuntos [DOBK79] [CHIN78].

. fazer as consultas sobre amostras tomadas aleatoriamente [DENN80]

. utilizar modelos regressivos para identificar e bloquear as informações passíveis de perda de confidencialidade [PALL86] [PALL87].

. introduzir perturbações nos dados armazenados [BECK80].

Entretanto, a falta de generalidade desses processos aliada às dificuldades de limitação no acesso aos dados pelas linguagens de consulta aos BDE, fazem com que os operadores de bancos de dados estatísticos separem os microdados identificados em bancos com acesso restrito.

2.3.2. Insuficiência Semântica

A insuficiência semântica das informações nos BDE impede o perfeito manuseio do banco de dados. Dificulta a identificação das informações disponíveis, impede a visão dos relacionamentos existentes entre estas informações e dos procedimentos necessários para sua obtenção.

Sem uma boa explicitação semântica é impossível ao sistema identificar os requisitos de uma consulta e processá-la. Por exemplo, o levantamento da produção anual de um produto investigado em períodos mensais requer o conhecimento de que a produção anual equivale à soma das produções mensais. Ou então, o problema de definir a população atendida por serviços básicos de saneamento numa região passa pela compreensão do que é saneamento básico (instâncias de um descritor de informações), pelo conhecimento das categorias a serem consideradas (abastecimento de água e es[gotamento sanitário), pela definição precisa da região geográfica que será analisada, pela decisão sobre se não moradores presentes na data da pesquisa devem ser considerados na população (hotéis e pensões), pela data de referência do levantamento dos dados, entre outras considerações.

O manuseio de arquivos de dados amostrados requer o conhecimento dos pesos a serem aplicados aos valores das instâncias para expandir a amostra e o conhecimento do nível de confiabilidade dos dados, onde estão definidos o erro amostral e as condições de uso da amostra.

2.3.3. Integração de Ferramentas

Uma consulta típica a um BDE consiste no cálculo de uma função estatística para uma ou mais variáveis selecionadas entre os objetos do universo, ou seja, na resolução de uma função sobre uma coleção ordenada de tuplas obtidas através da seleção de valores existentes no banco de dados com o uso de fórmulas e operadores lógicos [MALV89]. Decompondo essa definição encontramos os três componentes que interagem numa consulta a um BDE:

. a identificação das variáveis sobre as quais será tomada a estatística;

. uma linguagem de consulta com seleção lógica de dados;

. uma ferramenta estatística.

Para identificação das variáveis existentes no acervo do IBGE foi desenvolvido um Banco de Metadados, independente do BDE, o que viabilizou o conhecimento semântico necessário ao acesso dos dados existentes, pelos analistas da instituição.

O problema de seleção de valores e cálculo de estatísticas depende dos recursos disponíveis. Em geral, os SGBD dispõem de bons recursos para executar eficientemente essa tarefa, mas não dispõem de recursos estatísticos mais complexos em suas linguagens de consulta, enquanto os pacotes estatísticos apresentam sérias limitações como gerenciadores de banco de dados. Mesmo tendo melhorado as funções de gerenciamento de dados e de aceitarem processar as principais estruturas específicas de dados estatísticos, as ferramentas estatísticas continuam especializadas nos seus objetivos e não apresentam as principais características exigidas de um SGBD.

O inverso, ou seja, incorporar uma biblioteca de funções estatísticas numa linguagem de consulta, apesar de viável, também apresenta o inconveniente da perda de padrão ao exigir sua modificação. As linguagens baseadas na álgebra relacional, do tipo SQL, hoje se constituem em padrão aceito por, praticamente, todos os SGBD comercializados.

A solução que é usada mais freqüentemente consiste na construção de interfaces entre as linguagens de consulta utilizadas pelo SGBD e a ferramenta estatística. No IBGE, foram desenvolvidas interfaces que permitem, através da linguagem implementada pelo pacote SAS, o acesso às bases de dados gerenciadas pelo SGBD RAPID, como as do Censo Demográfico e das pesquisas contínuas PNAD e PME. Para as bases de dados gerenciadas pelo DB2, como Banco de Microdados Agropecuário (BMA), utiliza-se a própria interface do SAS com aquele SGBD. As duas soluções permitem utilizar plenamente os recursos existentes nas linguagens de consulta dos SGBD e, após a construção de um arquivo de trabalho no formato SAS, utilizar todos os recursos da ferramenta estatística. Com uma vantagem adicional: o usuário não vê os dois ambientes distintos existentes, o do SGBD e o do SAS. Utiliza todos os recursos como se estivesse trabalhando em apenas um destes ambientes.

2.4. LINGUAGENS DE CONSULTA

As linguagens de consulta e manuseio de dados utilizados nos BDE diferenciam-se das demais linguagens para banco de dados por sua capacidade de manusear dados ausentes, dados esparsos e pela capacidade de efetuar operações estatísticas de agregação e amostragem. A álgebra relacional introduzida por Codd [CODD72] não incorpora uma função de agregação. O uso freqüente de tabelas exige dos BDE um suporte especial a este formato de exibição de dados.

Ozsoyoglu *et al* [OZSO85] ao estudarem linguagens de acesso a banco de dados, diferenciaram os BDE construídos sobre bancos de dados corporativos dos construídos especificamente para uso estatístico. Nos tópicos a seguir são apresentados os modelos identificados neste trabalho para as linguagens de consulta especificamente desenvolvidas para BDE.

2.4.1. Modelo Relacional

Os sistemas desta categoria usam relações, ferramentas de modelagem e funções para a agregação de dados. O RAPID [TURN79] utiliza a álgebra relacional para processar suas consultas. Cada relação do RAPID é um arquivo transposto contendo, além dos dados, sua descrição (metadado). Esta descrição é composta pelo nome do atributo, tipo de dado, tamanho, domínio, data da última atualização e outras informações. Existe, ainda, um dicionário onde o RAPID armazena o restante do metadado na forma de entidades e ítens. Uma entidade pode ser uma relação, um conjunto de códigos ou de valores ou pode ser um comentário. Os itens descrevem as informações das entidades. O RAPID é totalmente relacional e possui interface com os principais *softwares* de análise estatística, como o SAS e SPSS.

Outro exemplo é o GENISYS [MANE83], um BDE que dispõe de visões dos dados, do tipo relacional, onde uma relação corresponde a um arquivo de dados. Como permite que uma relação contenha muitos dados repetidos, o GENISYS é considerado como da primeira forma não normalizada. Emprega uma linguagem de consulta, a GQL, baseada na linguagem SQL. O GQL utiliza uma estrutura de ligações para especificar e executar junções de relações de uma forma eficiente. Permite ao usuário especificar agregações de valores sobre uma população sem que primeiro efetue o particionamento desejado para as classes de resultado.

A ABE [KLUG81] é uma linguagem orientada a telas , semelhante a *query-by-example* (QBE). A principal característica da ABE é a capacidade de usar consultas como parâmetros para uma expressão de agregação, da mesma maneira que as operações sobre grupos da SQL. Com a ABE pode-se expressar consultas relacionais do tipo conjuntivo, mas não se pode fazer união de conjuntos. Assim, a ABE não é totalmente compatível com uma linguagem relacional.

O sistema CANTOR [KARA83], projetado para manuseio de grandes conjuntos de dados estatísticos, usa um modelo de dados baseado em objetos e dispõe de uma linguagem de manipulação dos dados chamada de SAL baseada na álgebra rela-

cional. Os objetos podem ser tanto elementares (inteiros, literais, etc) como tuplas ou conjunto de objetos. Uma relação é um conjunto especial de objetos. O metadado mantém as informações sobre os dados armazenados em três tuplas. Cada objeto pode ter um modo "valor" (um objeto armazenado) e um modo "vista" (uma expressão que avaliada produz um valor). A SAL é uma linguagem de consulta para expressões algébricas não recursivas. Os operadores podem ser aritméticos ou lógicos.

2.4.2. Modelo de Rede ou Hierárquico

Existem poucos exemplos de BDE desenvolvidos com o modelo de dados em rede ou hierárquico. O TPL [TPLS80], usa uma forma própria para descrever os relacionamentos existentes entre conjuntos de entidades, sendo cada conjunto um arquivo. O processo se desenvolve em várias etapas. Inicialmente, o administrador do banco de dados constrói um grafo direcionado *G* de relacionamentos entre os conjuntos de entidades. Depois todas as possíveis árvores onde um nó tenha no máximo um relacionamento *1-n* para seus filhos são enumeradas num conjunto *S* de árvores . Um novo conjunto *V* é formado de *S* pela eliminação de todas as sub-árvores. Quando o usuário deseja fazer uma tabela no TPL ele escolhe uma entre as possíveis árvores identificadas previamente no conjunto *V* de árvores obtido a partir do grafo. O processo apresenta um inconveniente no tamanho do conjunto *V* de árvores, que pode assumir dimensões muito grandes e outro em obrigar o usuário a manusear uma árvore completa, mesmo que queira uma tabela com poucos atributos. Apesar destes inconvenientes, o TPL é geralmente empregado com pequeno número de entidades e é utilizado por agências oficiais de estatística em muitos países.

2.4.3. Modelo Relacional Estendido

Alguns sistemas foram desenvolvidos sobre modelos relacionais estendidos como o SSDL [BROW83]. O SSDL é uma linguagem de manipulação de dados com procedimentos de alto nível destinada ao manuseio de objetos do tipo conjunto, conjuntos ordenados, vetores, matrizes, séries temporais, texto e relações *G*, conhecidas como tipos de dados complexos. As relações G são utilizadas para representar o modelo de dados chamado de SAM* Semantic Association Data Model. Com os SAM* o mundo real é modelado em termos do conjunto associados aos seus interrelacionamentos. Como as relações *G* podem ser definidas recursivamente, um número variável de relações pode ser aninhado num simples atributo de uma tupla. Os atributos de uma relação *G* podem constituir categorias ou atributos de valor. Os operadores do SSDL incluem os operadores tradicionais da álgebra relacional para as relações *G*,

operadores da teoria dos conjunto para os mesmos, operadores da álgebra linear para vetores e matrizes, e manipulação de cadeias de caracteres para textos.

Klug [KLUG82] propôs uma extensão da álgebra e cálculo relacionais para incorporar as funções de agregação e demonstrou que a linguagem obtida mantém a funcionalidade e potência do SQL. A extensão proposta consiste em acrescentar um novo operador, o "formador de agregação", que particiona uma relação (ou uma expressão algébrica utilizada para avaliação) num conjunto de atributos para então utilizar uma função agregação como a SUM em cada partição.

Ozsoyoglu et al [OZSO83] definem uma extensão da álgebra relacional e da linguagem de cálculo que utiliza funções de agregação e relações com atributos de referência. A linguagem obtida é equivalente à original e apresenta a mesma potência. A extensão consiste em acrescentar um novo operador, "agregação-por-modelo", e operações de compactação / descompactação às categorias existentes nas referências. O novo operador reagrupa as tuplas na relação de acordo com o modelo definido no operador para então proceder a agregação. ë mais conveniente do que o operador proposto por Klug por permitir a prévia definição do modelo de agregação.

2.4.4. Aplicações Desenvolvidas para Consultas

Os grandes produtores de informações estatísticas, coordenadores de sistemas oficiais de estatística e instituições internacionais, costumam disseminar a informação produzida por diversos meios como publicações, arquivos desidentificados, amostras, microfichas e por aplicações para acesso interativo à bancos de dados estatístico. Estas aplicações podem ser distribuídas junto com os arquivos de dados para serem operadas no ambiente do usuário ou são operadas pela próprio produtor que para tanto mantém um sistema de acesso remoto às informações. Os exemplos a seguir apresentam alguns destes sistemas:

2.4.4.1. SIDRA

O SIDRA é o principal sistema de disseminação de informações operado pelo IBGE e destina-se ao manuseio de dados regionais agregados. O SIDRA compreende uma base de dados agregados e uma interface de acesso que permite a exibição ou a gravação dos dados recuperados em arquivo do usuário. Com o SIDRA o usuário pode limitar uma área geográfica, o conjunto de informações e definir filtros de conteúdo da informação.

A organização da base é feita por pesquisas e o acesso pode ser efetuado pelo código das variáveis ou por um tema que compreende um conjunto pré-definido de variáveis.

2.4.4.2. BMA

O BMA, o Banco de Microdados Agropecuários é uma base de dados operada pelo IBGE, especializada em assuntos agropecuários. O BMA organiza os dados segundo um modelo relacional, registrando os microdados produzidos pelo IBGE na área de agropecuária. O modelo permite a comparabilidade nos aspectos espacial e temporal, incorporando conceitos de séries temporais de informação através da compatibilização dos sistemas de classificação adotados. O acesso às tabelas do BMA é efetuado através da linguagem de consulta do SGBD e por interface com a linguagem de análise estatística SAS.

2.4.4.3. REDATAM

O REDATAM é um sistema de banco de dados e uma linguagem de consulta distribuído pela ONU para análise e disseminação de dados em pequenas áreas. O REDATAM permite a análise de informações em áreas muito pequenas, como distritos, sendo uma ferramenta de apoio bastante interessante dentro de um sistema de disseminação de dados censitários.

O REDATAM manuseia arquivos totalmente transpostos e incorpora um pequeno dicionário de dados, hierarquizado, para apoio à busca de informações.

2.4.4.4. STARS

O STARS é o *software* desenvolvido pelo Banco Mundial para distribuição de três base de dados estatísticas, a chamada WTA ou seja as tabelas de indicadores mundiais, as tabelas sobre a dívida externa dos países e a dos indicadores sociais do desenvolvimento. O STARS utiliza um modelo relacional para armazenamento dos dados porém o acesso à base é restrito aos recursos do sistema, distribuído para ser operado no ambiente do usuário. O STARS permite que ao final de uma consulta os dados selecionados sejam editados para unidades mais convenientes à percepção e que sejam gravados em quatro formatos distintos para serem manuseados por *softwares* de análise estatística.

2.4.4.5. CRONOS

O CRONOS é o *software* utilizado pela Comunidade Européia (CE) e países membros para suporte a banco de dados estatísticos. As bases do CRONOS cobrem estatísticas sociais, macro-econômicas, industriais, energéticas, sobre o comércio externo, sobre agricultura e sobre países em desenvolvimento organizadas em modelo de séries temporais de informação. O CRONOS é operado pela CE para acesso *on-line* a suas séries de informações.

2.5. FERRAMENTAS DE ANÁLISE ESTATÍSTICA

Uma interessante coletânea dos principais *softwares* de análise estatística foi organizada por Sheryl Canter e outros e apresentada pela Revista PC Maganize [CANT93]. Além da apresentação dos produtos, a coletânea apresenta uma excelente análise comparativa das versões destinadas ao uso em plataformas PC compatíveis, detalhando seus principais componentes.

2.5.1. BMDP

O BMDP *Dynamic* produzido pela *BMDP Statistical Software* consiste num conjunto de 43 programas estatísticos e 3 programas utilitários muito apreciados pelos pesquisadores por sua facilidade de uso e potência dos recursos estatísticos. Apresenta uma boa capacidade de operar com dados ausentes e excelentes recursos para efetuar análises de grande complexidade estatística. No entanto, não dispõe de gráficos em três dimensões e não permite efetuar rotações sobre os eixos. Seus recursos de gerenciamento de dados são suficientes para um *software* de sua categoria, não operando sobre nenhum SGBD. O resultado de uma análise pode ser apresentado em arquivo texto para ser editado por outro *software* ou por um editor de texto.

2.5.2. Genstat

O Genstat é um programa bastante popular na Europa, principalmente pela sua capacidade de manuseio de experimentos estatísticos. Dispõe de excelentes recursos para a execução de análises de grande complexidade estatística. Os recursos gráficos permitem a construção tridimensional e permitem uma grande interferência do usuário. O principal destaque do Genstat é o manuseio de modelos lineares generalizados.

O Genstat utiliza uma estrutura própria para armazenamento dos dados e não dispõe de interfaces para dados existentes em SGBD.

2.5.3. SAS

O SAS do *SAS Institute* é o *software* de análise estatística mais utilizado em todo o mundo. Trata-se na realidade de uma linguagem de programação onde os recursos de análise estatística aparecem como rotinas prontas para o uso. A distribuição do *software* é feita em grupos de rotinas para análise descritiva, modelos lineares, séries temporais, controle de qualidade, entre outros. O SAS dispõe de bons recursos para construção de gráficos tridimensionais e mapas temáticos. Entre os *softwares* apresentados, é o que dispõe de melhores recursos para administração de dados, chegando a implementar um modelo de dados relacional, desenvolvido no próprio SAS. O SAS também dispõe de rotinas que permitem o acesso aos dados armazenados nos principais SGBD e planilhas eletrônicas.

2.5.4. SPSS

O SPSS, *Statistical Package for Social Sciences*, do *SPSS Inc.* permite processar praticamente qualquer análise estatística que seja concebida. Opera sobre qualquer plataforma conhecida de hardware e *software*. A portabilidade de uma aplicação é assegurada pelo uso da mesma sintaxe de programação em qualquer versão. Além de um excelente gerador de gráficos, o SPSS permite a construção de um *carrossel de imagens* com os gráficos a serem exibidos numa apresentação. O SPSS dispõe de bons recursos para a importação de dados armazenados em formatos de planilhas e de SGBD.

2.5.5. P-Stat

O P-Stat da *P-Stat Inc.* apresenta recursos razoáveis para a análise estatística. Apesar de não dispor de rotinas que o caracterize em alguma área específica da análise estatística, cobre quase que inteiramente todas as necessidades. Emprega uma linguagem de programação baseada no Fortran com poucos recursos de auxílio on-line. Seus recursos gráficos são limitados e de baixa resolução. O P-Stat dispõe de bons recursos para manuseio de arquivos com grande quantidade de dados.

2.5.6. Statgraphics

O Statgraphics da *Manugistics Inc.* é um programa para manuseio interativo de funções estatísticas. Permite uma boa integração de estatísticas, gráficos e dados. Apesar de não contar com rotinas de suporte para todo tipo de análise estatística, destaca-se na análise de séries temporais, desenho de experimentos e controle de qualidade. Tem na elaboração e utilização dos gráficos seu ponto forte. Possui uma boa capacidade de manuseio de dados restringida apenas na incapacidade de importar mais de um registro por ocorrência.

2.5.7. Statistica

O Statistica da StatSoft Inc. é um bom *software* para análises estatísticas de pouca complexidade feitas sobre pequenos conjuntos de dados. Tem seu ponto forte na produção de gráficos, apresentando uma excelente integração com estatísticas. As funções de manuseio de dados são restritas neste *software*. O Statistica permite importar dados em formato de registro e de planilhas mas não dispõe de interface para acesso ao SGBD.

2.5.8. Systat

O Systat da *Systat Inc.* é considerado um excelente *software* para análise estatística, inclusive por aquelas de grande complexidade. Cobre, praticamente, todas as necessidades de um análise estatística utilizando uma interface gráfica para a edição de programas numa linguagem tipo BASIC. A geração e manuseio de gráficos é destaque do *software*. Utiliza o conceito de WYSIWYG (conforme apresentado no ítem 4.3.5) para a introdução de comentários do usuário sobre um gráfico. Dispõe do recurso de efetuar rotação sobre qualquer dos eixos de um gráfico para melhor visualização dos dados. Apresenta alguma restrição no manuseio de dados e está limitado a 256 variáveis. O Systat não trabalha com SGBDs porém, permite importar dados armazenados no formato DBase, DIF e de algumas planilhas.

2.6. PROBLEMAS EXISTENTES

O uso das linguagens de consulta à BDE por usuários não especialistas em informática costuma apresentar algumas dificuldades, geralmente, no nível da interface do sistema. Estes problemas podem ser resumidos em 5 pontos:

. Dificuldade de interpretação do conteúdo do BDE, na sua estrutura, atributos, relações e interrelacionamentos;

. Insuficiência semântica dos modelos de dados utilizados;

. Dificuldade de expressar uma consulta utilizando linguagens formais;

. Impossibilidade de consultar o esquema em níveis de detalhamento;

. Insuficiência semântica da informação.

O modelo relacional, como o utilizado pelo RAPID, apesar de ser o de melhor desempenho operacional, apresenta os mesmos problemas de interface que as aplicações de BDE feitas sobre os outros modelos de dados.

Uma das sugestões feitas para superação destes problemas é mudar a técnica de acesso a informações em BDE para métodos que não exijam do usuário maiores conhecimentos do que os já possuídos no campo de sua competência. Outra, diz respeito a aumentar a semântica do BDE para permitir o perfeito conhecimento da estrutura de informações existentes, do modelo utilizado para seu armazenamento, do seu significado e do formato da representação de forma mais natural e compatível com a utilizada no mundo real.

Enfim, as propostas para superação dos problemas dizem respeito ao emprego de métodos e técnicas que liberem o usuário da obrigação de tornar-se um especialista nos sistemas de BDE para poder ser um especialista no campo de sua atuação.

3. ORIENTAÇÃO A OBJETOS

O termo "SGBDOO - Sistema Gerenciador de Banco de Dados Orientado a Objetos" vem sendo empregado em diversos sentidos. Muitos protótipos de novos sistemas gerenciadores de banco de dados utilizam esta denominação para modelos diferentes. Roderic G.G. Cattell [CATT91] propõe que sejam considerados como sistemas gerenciadores de banco de dados orientados a objetos, os SGBDs que satisfazem as necessidades de aplicações não convencionais, tais como CAD, engenharia de *software* (CASE) ou automação de escritório, e que satisfaçam alguns requisitos da orientação objeto. Cattell, utilizando a classificação proposta por J.Ullman, identifica três grandes categorias de sistemas gerenciadores de banco de dados:

. gerenciadores de dados - aqueles voltados ao manuseio de estruturas simples de dados e às operações normalmente requeridas nas tradicionais aplicações comerciais. Como exemplo, temos os SGBDs que implementam os modelos relacionais com sua estrutura de dados em tabelas de atributos simples de dados (inteiros, caracteres, etc.).

. gerenciadores de objetos - aqueles voltados para estruturas mais complexas de dados, tais como as requeridas na representação de partes de documentos, programas ou desenhos.

. gerenciadores de conhecimento - aqueles voltados à manutenção de bases complexas de regras utilizadas para definir a informação dentro de determinado domínio, através de um sistema de inferência.

Com esta definição simplificada, Cattell evita a discussão existente na conceituação do paradigma da orientação objeto, dividida em duas grandes vertentes: a do *Grupo de Altair* que publicou o "Manifesto dos Sistemas de Banco de Dados Orientados a Objetos" [ATKI89]; e o grupo denominado *Comitê para Funções Avançadas em SGBDs*, que publicou o "Manifesto dos Sistemas de Banco de Dados de Terceira Geração" [COMI90]. Como não poderia deixar de ser, existe uma forte interseção nas proposições, mesmo tendo sido formuladas de maneira distinta.

3.1. MANIFESTO DOS SISTEMAS ORIENTADOS A OBJETOS

O Grupo de Altair preocupou-se em definir os sistemas de banco de dados orientados a objetos como um novo modelo de dados, dotado de características próprias, evitando a adoção de um produto como um padrão *de facto*. O **"Manifesto"** considerou que a falta de conceituação da orientação objeto decorria de três problemas:

. falta de consenso sobre qual o modelo de banco de dados orientado a objetos existente a ser adotado;

. inexistência de uma teorização forte sobre o assunto;

. inexistência de experiências significativas na área.

Esse problemas levaram o Grupo de Altair a evitar a padronização de linguagens para poder aprimorar a terminologia adotada na orientação objeto e para produzir, assim, um conjunto básico de definições. As características foram divididas em três grandes grupos: mandatórias, aquelas obrigatórias para que um sistema possa ser reconhecido na categoria; opcionais, aquelas que são desejáveis nos sistemas porém, não são mandatórias; e as abertas, consistindo nas soluções individuais em uso.

3.1.1. Características Mandatórias

Um sistema de banco de dados orientado a objetos deve satisfazer dois critérios: ser um SGBD; e ser um sistema orientado a objetos, isto é, deve ser consistente com as características das linguagens orientadas a objetos.

Para ser um SGBD o sistema deve satisfazer cinco critérios: persistência, gerência de memória secundária, concorrência, recuperação e dispor de um recurso para consultas *ad hoc*. O segundo critério se traduz em oito componentes: objetos complexos, identidade de objetos, encapsulamento, tipos ou classes, herança, sobreposição combinada com montagem a posterior, capacidade de extensão e ambiente computacional completo.

3.1.1.1. Objetos complexos

De característica obrigatória, os objetos complexos são aqueles construídos a partir de objetos simples com a aplicação de construtores. Como objetos simples te-

mos os inteiros, os caracteres, cadeias de bytes de qualquer tamanho, informações binárias e de ponto flutuante. Como construtores podemos ter as tuplas, conjuntos, *bags*, listas e matrizes. Um banco de dados orientado a objetos deve dispor, pelo menos, de tuplas, conjuntos e listas.

Diferentemente dos modelos relacionais onde um conjunto só é aplicável às tuplas e estas a valores atômicos, os construtores do modelo orientado a objetos precisam ser ortogonais, no sentido de que qualquer construtor deve ser aplicável sobre qualquer objeto. As operações com objetos complexos precisam ser propagadas transversalmente a todos os componentes como nas operações de recuperação ou eliminação que devem atingir todo o objeto. Numa operação de cópia deve ser produzida uma cópia *deep* e não uma cópia *shallow*, na qual os componentes não são replicados.

3.1.1.2. Identidade do objeto

A idéia da identidade do objeto, obrigatória no modelo orientado a objetos, consiste em manter os objetos de forma independente de seu valor. Assim, dois objetos são idênticos quando tiverem o mesmo valor e a mesma identidade e serão iguais quando, apesar de terem o mesmo valor, tenham identidades diferentes.

3.1.1.3. Encapsulamento

O encapsulamento, mandatório no modelo orientado a objetos, permite implementar, na visão das linguagens de programação orientadas a objetos, tipos abstratos de dados. O objeto é dividido em duas partes: a interface e a implementação. A parte da interface consiste na especificação do conjunto de operações que podem ser realizadas com o objeto, sendo sua parte visível. A parte da implementação contém os dados e os procedimentos. Os dados representam o estado do objeto e os procedimentos descrevem, em termos de uma linguagem de programação, a implementação de cada operação. Na translação deste princípio para a visão de banco de dados, um objeto encapsula tanto os programas como os dados.

3.1.1.4. Tipos e Classes

Os sistemas orientados a objetos devem suportar tipos ou classes. Tipos representam o conjunto de recursos comuns a um grupo de objetos que tenha as mesmas características. O tipo corresponde a noção de um tipo abstrato de dados. Ele é dividido em duas partes: a interface, visível aos usuários, onde são definidas as possíveis

operações com suas respectivas assinaturas; e a implementação, visível apenas para o projetista do tipo, onde é descrita a estrutura interna dos dados do objeto e dos procedimentos utilizados para implementar as operações descritas na interface. A definição de tipos é utilizada nas linguagens de programação orientadas a objetos como ferramentas para a correção de programas.

O conceito de classe é diferente da noção de tipo. Apesar de conter a mesma especificação, classe é mais dirigida ao momento da execução, como uma fábrica que produz novos objetos através do uso da operação *new*. Resulta também na noção de repositório de objetos, uma vez que as instâncias de uma classe formam um conjunto de objetos.

3.1.1.5. Hierarquias de Classes ou Tipos

O mecanismo de herança é mandatório nos sistemas orientados a objetos, e consiste numa poderosa ferramenta de modelagem por fornecer uma descrição precisa e concisa do mundo e por ajudar no detalhamento de especificações comuns. O mecanismo de hierarquia tem como conseqüência a herança das propriedades de um tipo pelos seus subtipos. A herança pode ser efetuada de quatro maneiras:

. *substituição* - a herança é dita por substituição quando baseia-se no comportamento dos objetos, sendo indiferente aos seus valores. Quando um objeto herda de outro por substituição, pode utilizar livremente os métodos do primeiro;

. *inclusão* - a herança por inclusão baseia-se na estrutura dos objetos e não em suas operações. Corresponde a noção de classificação, ou seja, se o tipo t é um subtipo de t' então, todo objeto do tipo t é também um objeto do tipo t';

. *restrição* - a herança com restrição corresponde a uma subclasse da inclusão onde é definida uma restrição. Um tipo t é um subtipo do tipo t' então, todos os objetos do tipo t são também objetos do tipo t' e satisfazem determinada restrição;

. *especialização* - a herança por especialização corresponde a adicionar mais informações aos subtipos.

3.1.1.6. Sobreposição, sobrecarga e montagem a posterior

O mecanismo de sobreposição permite a redefinição de uma operação para ser utilizada de forma específica quanto ao tipo de objeto manuseado, ou seja, uma operação de *display* aplicada sobre um objeto apresentará, por exemplo, uma tupla com valores quando o tipo do objeto for uma tupla de atributos inteiros ou então, um desenho quando o tipo for uma imagem. Nos sistemas orientados a objetos a redefinição de operações é chamada de sobreposição (*overriding*), sendo feita de acordo com o tipo do objeto. Uma redefinição faz com que operações diferentes, apesar de serem vistas com o mesmo nome, carreguem (sobrecarga ou *overloading*) programas diferentes.

Para permitir essas características, os sistemas orientados a objetos não podem juntar as operações com programas no momento da compilação. A montagem deve ser feita a posterior, deixando para ser resolvido no momento da execução a translação de nomes para endereços de programas.

3.1.1.7. Ambiente computacional completo

Esta característica, mandatória nos sistemas orientados a objetos, é uma propriedade obrigatória nas linguagens de programação, ou seja, qualquer função computável deve poder ser expressa através da linguagem de manuseio de dados (DML). Entretanto, no ambiente de banco de dados convencional, isto não é verdade para as linguagens de consulta, como as SQL, que não são completas, no sentido que não podem executar qualquer função computável.

Para que um SGBD seja orientado a objetos, ele deve introduzir um ambiente computacionalmente completo, podendo utilizar-se de linguagens de programação já existentes.

3.1.1.8. Capacidade de extensão

Os SGBDs orientados a objetos devem permitir a extensão do conjunto pré-definido de tipos do sistema e devem tratar os tipos criados pelo usuário de forma idêntica aos do conjunto original, sem fazer qualquer distinção. A proposição se aplica somente aos tipos, não sendo mandatória a capacidade de extensão dos construtores de tipos (tuplas, conjuntos, listas, etc.).

3.1.1.9. Persistência

A capacidade de persistência, inerente aos sistemas de banco de dados, é mandatória nos SGBDs orientados a objetos. Entende-se por persistência de dados, a manutenção de um dado após a execução de um processo de forma a poder ser utilizado novamente por outro processo. A persistência deve ser ortogonal, isto é, cada objeto, independente do seu tipo, pode ser tornado persistente sem necessitar de translações explícitas.

3.1.1.10. Gerenciamento da memória secundária

A memória secundária, conjunto de mecanismos clássico dos SGBDs para melhorar o desempenho, é uma característica mandatória nos sistemas orientados a objetos. Esta característica implica na existência de um conjunto de mecanismos que gerenciem índices, segmentos de dados (*cluster*), *buffers* para dados, seleção do caminho de acesso aos dados e otimização de consultas. O gerenciamento da memória secundária deve ser também invisível para o usuário que deverá utilizá-la sem ter de escrever códigos de instruções para manter índices, alocar espaço em disco ou mover dados entre o disco e a memória. O conjunto destes mecanismos deve ser implementado com perfeita independência entre os níveis lógico e físico do sistema.

3.1.1.11. Concorrência

Os SGBDs orientados a objetos devem dispor, pelo menos, do mesmo nível de controle de concorrência existentes nos atuais SGBDs. Devem permitir a muitos usuários trabalharem simultaneamente sobre uma mesma base de dados.

3.1.1.12. Recuperação

Também, no aspecto de recuperação da base de dados, os SGBDs orientados a objetos devem dispor do mesmo nível de serviço oferecido pelos atuais SGBDs. Os sistemas devem poder restaurar a base de dados no estado anterior a falhas, tanto no caso das falhas do *software* como nas do hardware.

3.1.1.13. Recurso de consultas *ad hoc*

Os SGBDs orientados a objetos deverão dispor de linguagens que permitam ao usuário expressar consultas aos dados, observando os seguintes critérios:

. a linguagem deve ser de alto nível para que, de forma declarativa, enfatizando *o que* e não *o como*, com poucas palavras ou *clics*, possa ser elaborada uma consulta não trivial;

. a linguagem precisa ser eficiente, no sentido de contar com um otimizador de desempenho;

. a linguagem deve ser independente das aplicações, devendo ser executável em qualquer banco de dados.

3.1.1.14. Resumo

As características mandatórias diferenciam os SGBDs orientados a objetos dos modelos existentes. Os modelos relacionais não satisfazem as características descritas nos itens 3.1.1.1 a 3.1.1.8. Os modelos CODASYL não satisfazem as características descritas nos itens 3.1.1.3, 3.1.1.5, 3.1.1.6, 3.1.1.8 e 3.1.1.13 e satisfazem, apenas em parte, as dos itens 3.1.1.1 e 3.1.1.2, uma vez que os construtores não são ortogonais e a identificação do objeto não é tratada uniformemente nos relacionamentos restritos ao tipo *1:n*.

Na formulação das características mandatórias, o Grupo de Altair não chegou a um consenso sobre as quatro características de SGBDs descritas abaixo:

. definição de visões e dados derivados;

. recursos para administração do banco de dados

. restrições de integridade;

. recursos para evolução do esquema.

3.1.2. Características Opcionais

Como características opcionais foram definidas aquelas destinadas a melhorar os sistemas, mas que não são mandatórias para a classificação de um sistema na orientação a objetos. Algumas características de natureza orientada a objetos, como herança múltipla, foram classificadas como opcionais juntamente com outras clássicas em SGBDs, como o desenho do mecanismo de transações. Em geral, estas caracterís-

ticas são utilizadas pelas novas aplicações (CAD, CASE, etc), e apresentam-se mais como orientadas a aplicações do que orientadas a tecnologia.

3.1.2.1. Distribuição

A característica de distribuição de uma base de dados é considerada ortogonal a natureza dos sistemas orientados a objetos, que deverão permitir ou não a distribuição das bases.

3.1.2.2. Transações

Os SGBDs orientados a objetos devem suportar novos tipos de transações, como as transações muito longas exigidas nas aplicações de CAD, onde o critério de serialização de procedimentos não é adequado.

3.1.2.3. Versões

É desejável que os sistemas orientados a objetos suportem a construção de versões de um trabalho, como os requisitos dos processos de CAD e de CASE. Deve ser possível o manuseio simultâneo de qualquer das versões existentes de um trabalho.

3.1.3. Questões Abertas

Muitas questões são deixadas em aberto para serem utilizadas pelos sistemas orientados a objeto de forma livre. Essas questões diferem das características mandatórias por não serem essenciais para caracterizarem o sistema e das opcionais por não serem específicas à orientação a objetos.

3.1.3.1. Paradigma de programação

Não existem motivos para impor um paradigma de programação para os SGBDs orientados a objetos. Existem diversas alternativas que podem ser utilizadas como paradigmas de programação, como o da programação lógica, o da programação funcional ou da programação imperativa. Ou, até mesmo, suportar diversos paradigmas de programação em vez de optar por um.

3.1.3.2. Representação do sistema

A representação do sistema deve ser efetuada por um conjunto de tipos atômicos e por conjunto de construtores, para ser estendida em diferentes formas e maneiras.

3.1.3.3. Sistema de tipos

O sistema de formação de tipos é de livre utilização pelos sistemas orientados a objetos, sendo obrigatório apenas o mecanismo de encapsulamento.

3.1.3.4. Uniformidade

O tratamento de tipos, métodos e objetos deve ser visto em três níveis: no nível da *implementação*; no nível da *linguagem de programação*; e no nível da *interface*. No nível da implementação deve ser decidido que tipo de informação deve ser armazenado como objeto. No nível da linguagem de programação deve ser decidido se os tipos devem ser tratados como entidades especiais dentro da semântica da linguagem. E no nível da interface devem ser apresentados ao usuário com uniformidade os tipos, objetos e métodos, mesmo que não tenham sido tratados desta maneira nos demais níveis.

3.2. MANIFESTO DOS SGBDs DE 3ª GERAÇÃO

O grupo denominado "Comitê para Funções Avançadas em SGBDs" [COMI90] (Comitê) propõe que os modelos de dados sejam classificados por gerações. A primeira geração corresponde aos modelos em rede e aos modelos hierárquicos, primeiros sistemas a oferecerem, significativamente, as funções de SGBDs de forma unificada com a definição dos dados e com uma linguagem de manipulação dos registros de informações. O relatório da CODASYL sobre sistemas, de abril de 1971 [CODA71], define os modelos em rede baseados em registros com ligações do tipo pai-filhos (um-para-muitos). Diversas aplicações desenvolvidas com estes modelos são, até hoje, utilizadas graças ao excelente desempenho que apresentam.

A segunda geração corresponde aos modelos relacionais, muito empregados nos anos 80 nas aplicações comerciais. Apesar do sucesso que tiveram, estes modelos mostraram-se inadequados para um grande número de aplicações como as de CAD, CASE, hipertextos e informações geográficas. Para obter soluções complexas com o

uso dos SGBDs relacionais é necessário o desenvolvimento de funções específicas que manuseiem dados não convencionais como som e imagens.

A terceira geração corresponde aos objetivos desejados para manuseio de novos modelos, em especial, aos de natureza complexa. Para tanto, são estabelecidas três premissas que os modelos devem atender:

. os SGBDs de terceira geração devem suportar, além das estruturas manuseadas pelos sistemas de segunda geração, estruturas complexas de dados e regras;

. os SGBDs de terceira geração devem incluir todas as funções de segunda geração e, em especial, o acesso por linguagem de consulta sem procedimentos fixos e os requisitos de independência de dados.

. os SGBDs de terceira geração devem ser abertos para outros sistemas, devem dispor de uma linguagem de quarta geração, de ferramentas para suporte a tomada de decisões, permitir o fácil acesso a outros sistemas como o Lotus 1-2-3, devem dispor de interfaces com sistemas gráficos e devem permitir que uma aplicação rode em máquinas diferentes ou de forma distribuída.

Estas três premissas permitem o detalhamento de treze diferentes proposições que são apresentadas nos itens a seguir.

3.2.1. Objetos e Gerência de Regras

Os SGBDs não podem antecipar todas as necessidades de tipos de informação desejadas nas aplicações. Mesmo os tipos mais estáveis de informações recebem tratamento diferente de acordo com a aplicação. O dia, que no calendário normal se encerra às 24:00 horas, em algumas aplicações é encerrado no horário do término das operações, como no caso bancário. No calendário de dias para transações financeiras, não existem feriados nem fins de semana. O próximo dia é, na realidade, o primeiro dia subseqüente no qual existam novas transações. Assim, é desejável que os SGBDs saibam manusear estas variações de dados o que pode ser expresso nas seguintes proposições:

. proposição 1 - devem dispor de um completo sistema de tipos, incluindo um tipo abstrato que permita a construção de novos tipos, um construtor de tipos

manuseáveis em listas, um construtor de tipos manuseáveis em seqüências, um construtor de tipos sob a forma de registros, um construtor de tipos sob a forma de conjuntos, deve permitir o uso de funções como tipos, um construtor para união de tipos e a composição recursiva destes construtores;

. proposição 2 - devem permitir a utilização do mecanismo de herança;

. proposição 3 - devem permitir a utilização de funções, inclusive para procedimentos e métodos do banco de dados e encapsulamento de dados;

. proposição 4 - a utilização de identificadores de objetos deve ser utilizada somente quando não for definida uma chave primária pelo usuário.

. proposição 5 - as regras (gatilhos e restrições) deverão constituir mais um entre os grandes componentes dos SGBDs e não devem estar associadas a uma determinada função ou coleção de dados.

3.2.2. Funções dos SGBDs

Os SGBDs de 3ª geração devem incluir e aperfeiçoar as funções existentes nos sistemas atuais, com o que são enunciadas as seguintes proposições:

. proposição 6 - os acessos aos bancos de dados de 3ª geração deverão ser feitos com linguagens de alto nível e livres de procedimentos;

. proposição 7 - deve ser possível definir coleções de dados de, pelo menos, duas maneiras distintas;

. proposição 8 - as visões devem ser atualizáveis;

. proposição 9 - os índices de desempenho não devem fazer parte do modelo de dados.

3.2.3. Sistemas Abertos

As discussões sobre sistemas abertos em bancos de dados se confundem com a definição dos chamados "*API - Application Programming Interface*" (interfaces para

a programação de aplicações) através dos quais os programas de aplicações podem se comunicar com os SGBDs:

. proposição 10 - os SGBDs de 3ª geração devem ser acessados de múltiplas plataformas;

. proposição 11 - a persistência de dados deve ser suportada por SGBDs ao tempo de execução por extensões de compiladores e por linguagens de alto nível;

. proposição 12 - a linguagem SQL deve ser um padrão;

. proposição 13 - consultas e respostas devem ser o nível mais baixo de comunicação entre servidores e clientes

3.2.4. Resumo das Proposições

Existem muitos pontos em comum entre estas proposições com as do manifesto da orientação objeto (item 3.1) tais como tipos, funções, herança e encapsulamento. Porém, enquanto a proposição do Grupo de Altair [ATKI89] é voltada essencialmente para a gerência de objetos, o grupo do Comitê se preocupa com os aspectos de suporte aos dados, regras e com ferramentas que permitam a integração da nova geração de SGBDs de forma independente de suas plataformas.

A segunda grande diferença está no enfoque do grupo sobre o uso da linguagem de consulta considerado essencial aos SGBDs de 3ª geração, juntamente com um otimizador de consultas, com um sistema de regras, com um suporte cliente/servidor, com o suporte de visões e com o relaxamento do uso de identificadores de objetos.

3.3. ARQUITETURA DOS NOVOS SISTEMAS

A arquitetura dos novos sistemas de banco de dados, conforme apresentado nos itens abaixo, é feita de acordo com a natureza do recurso que a originou.

3.3.1. Linguagens de Programação

As linguagens de programação de banco de dados orientados a objetos originam uma arquitetura de mesmo nome. Suas aplicações são escritas em extensões de

funções implementadas por linguagens de programação existentes com o objetivo de incorporar a funcionalidade de um banco de dados. Como exemplo temos o O_2, ObjectStore, Objectivity/DB, ONTOS, VERSANT, GBase, GemStone, STATICE, ZEITGEIST, MCC ORION e ITASCA. A maioria dessas linguagens é integrada ou derivada de C++, enquanto algumas poucas são baseadas numa derivação de LISP.

3.3.2. Sistemas de Bancos de Dados Relacionais Estendidos

Na arquitetura de extensões sobre os sistemas de banco de dados relacionais, apesar de não constituírem modelos de dados orientados a objetos, tanto o modelo de dados quanto a linguagem de consulta foram expandidos para incluir procedimentos, identificação de objetos, tipos de hierarquias entre outros recursos. Como exemplos dessa arquitetura temos o POSTGRES, o Starburst e o ALGRES.

3.3.3. Sistemas de Bancos de Dados Semânticos e Funcionais Estendidos

Existem, ainda, as extensões em sistemas funcionais de banco de dados como o IRIS, PROBE e ADAPLEX, e em sistemas semânticos como o CACTIS e o SIM, efetuadas de forma semelhante as extensões dos funcionais.

3.3.4. Sistemas Geradores de Bancos de Dados

Outra abordagem possível para implementação de um SGBDOO é utilizar um gerador de um sistema de banco de dados, como o EXODUS ou o GENESIS, que permita ao usuário definir seu próprio modelo de dados nos níveis conceitual e interno. Apesar desses geradores não serem específicos para a orientação a objetos eles representam uma alternativa para o problema.

3.3.5. Gerência de Objetos

Os gerenciadores de objetos apresentam-se menos funcionais do que as outras alternativas apresentadas. Em geral, possuem a propriedade de tornarem persistentes objetos simples e de controlarem a concorrência de acesso, mas não dispõem de uma linguagem de consulta ou de programação. Como exemplos temos o POMS, Mneme, ObServer, sistema de arquivos iMAX-432, Camelot e o LOOM Smalltalk-80.

3.4. RESUMO DOS CONCEITOS DOS SGBDOOs

Juntando as características defendidas pelos autores dos manifestos com os produtos e protótipos desenvolvidos, é possível estabelecer uma lista de conceitos em uso pelos sistemas de banco de dados orientados a objetos. Os objetos possuem atributos e definem procedimentos com os quais constituem suas propriedades e formam tipos. Os objetos devem poder ser persistentes e os SGBDOOs devem controlar a concorrência de acesso e, caso seja necessário, sua recuperação.

3.4.1. Objetos

O termo objeto é utilizado pelos SGBDOOs para representar: uma entidade do mundo real, um "grupamento de objetos" e como "identidade de um objeto". Os objetos devem possuir uma identidade única e independente dos valores que possa conter.

3.4.1.1. Identificadores de objetos

Os identificadores de objetos, os *OID*, começaram a surgir após o desenvolvimento dos sistemas relacionais, para substituir o conceito de chaves na identificação de objetos, de forma a se obter uma identificação independente de valores.

3.4.1.2. Chaves de objetos

Alguns SGBDOOs permitem a definição de chaves de identificação de objetos para serem utilizadas da mesma maneira que as chaves primárias dos sistemas relacionais. Assim, simplifica-se a sintaxe usada para referenciar um objeto através de chaves e o SGBD pode verificar automaticamente a restrição de integridade de chave única.

3.4.1.3. Atributos simples

Atributos simples de um objeto são os valores armazenados num objeto, conjuntos de valores ou referências para outros objetos. Podem ser, inclusive, tipos definidos pelo usuário do SGBDOO.

3.4.1.4. Atributos referenciais

Atributos referenciais são atributos complexos nos quais as referências a outros objetos são utilizados para representar os interrelacionamentos, da mesma maneira que a chave estrangeira é utilizada nos sistemas relacionais. Difere, no entanto, desta chave na medida em que assegura a integridade referencial do objeto.

3.4.1.5. Atributos de coleção

Os atributos de coleção são aqueles utilizados para formar listas ou vetores de valores segundo algum critério de ordenação, ou para formar coleções de objetos, de atributos simples ou de referências.

3.4.1.6. Atributos derivados

Os atributos derivados ou virtuais são aqueles em que uma declaração de procedimentos substitui o armazenamento explícito do valor. Com os atributos derivados obtem-se a independência lógica dos dados.

3.4.2. Relacionamentos

Os relacionamentos entre objetos definem a forma com que dois ou mais tipos de objetos podem ocorrer no mundo real. Os relacionamentos também são utilizados em alguns SGBDOO para estabelecer restrições de integridade numa base de dados.

3.4.2.1. Relacionamentos binários

Relacionamentos entre objetos num modelo de dados orientado a objetos são descritos através de atributos complexos, do tipo atributo de referência ou de coleção de referências. Quando o relacionamento for do tipo binário *1:1*, a representação é feita com a inclusão de um atributo de referência em cada objeto. Para os relacionamentos do tipo *1:N* é definido um atributo de referência para o objeto com múltiplas ocorrências (*N*) e um atributo do tipo coleção no objeto unitário (*1*). Finalmente, para o caso *N:M* são acrescentados atributos de coleção nos dois objetos.

3.4.2.2. Atributos inversos

Atributos inversos são aqueles utilizados para fazer referência cruzada entre objetos. A descrição desses atributos é feita nos dois objetos que se referenciam. Para os relacionamentos *1:N* e *N:M* é acrescentado um atributo de referência aonde exista uma referência inversa.

3.4.2.3. Integridade referencial

Um modelo de dados orientado a objeto pode implementar diversos níveis de integridade referencial: desde não implementar nenhuma verificação de integridade sobre as referências até dispor de um completo teste de validade. Quando o SGBDOO não dispõe de testes de integridade, o usuário deve se prevenir para não fazer referência a um objeto errado ou inexistente. Caso existam testes de validade das referências, o sistema tanto pode eliminar automaticamente um objeto que tenha se tornado inacessível ao usuário, quanto requerer explicitamente sua eliminação. Com a utilização de atributos inversos, o SGBDOO pode assegurar a integridade dos relacionamentos de maneira semelhante às chaves primárias dos modelos relacionais. Outra forma de assegurar a integridade referencial nos SGBDOOs, consiste na utilização de regras semânticas declaradas com a definição de métodos associados aos objetos.

3.4.2.4. Relacionamentos não binários

Quando os relacionamentos forem de ordem superior à binária, o uso de atributos inversos não será suficiente para caracterizá-los. Assim, será necessário criar novos tipos de objetos que representem esses relacionamentos. Apesar do aumento de complexidade que esta técnica irá acarretar no esquema dos objetos, a solução é ainda mais simples do que a dos modelos relacionais por resolverem este problema com a construção de novas tabelas e relacionamentos.

3.4.2.5. Relacionamentos derivados

A exemplo dos atributos, os relacionamentos entre objetos também podem ser definidos através de procedimentos, em substituição a declarações explícitas. Como a implementação de relacionamentos é feita através dos atributos, basta utilizá-los de forma derivada para então obter relacionamentos também derivados. Para completar a definição, deve ser especificado um método de recuperação para cada atributo inverso envolvido no relacionamento.

3.4.3. Objetos Complexos

Nos sistemas de Banco de Dados, o termo agregação pode ser usado para definir o agrupamento de atributos na formação de um objeto, ou o agrupamento de objetos na formação de um objeto complexo. No conceito de agrupamento de objetos para formarem objetos complexos, o termo agregação refere-se à hierarquia estabelecida entre os objetos na sua forma estrutural e não ao conceito de generalização de tipos de objetos (classes) estabelecido pela hierarquia de tipos (herança).

3.4.3.1. Semântica na agregação

Um SGBD pode implementar recursos que tratem de forma automática as decorrências da abstração dos objetos complexos. Por exemplo, quando for feita uma referência para cópia ou eliminação de um objeto complexo, todos os objetos hierarquicamente subordinados serão copiados ou eliminados. E, caso alguns desses objetos também sejam complexos, o processo irá referenciando recursivamente toda a estrutura da hierarquia até chegar a sua base.

3.4.3.2. Relacionamentos na agregação

Inicialmente, para definir relacionamentos decorrentes do processo de agregação, foi utilizado o conceito de "parte de" derivado da representação do conhecimento de Inteligência Artificial. A limitação desta abordagem evoluiu para a definição de nomes e de novos relacionamentos definidos pelo próprio usuário, de acordo com os tipos de objetos (classes) que estiverem sendo agrupados.

3.4.4. Procedimentos

Diferentemente dos SGBDs convencionais, os SGBDOOs implementam linguagens de acesso a bancos de dados computacionalmente completas, ou seja, linguagens que podem executar as mesmas operações que uma linguagem comum executaria. Os SGBDOOs podem associar os procedimentos com objetos e armazená-los no banco de dados.

3.4.4.1. Procedimentos e encapsulamento

Métodos são os procedimentos utilizados para encapsular ou esconder os atributos de um objeto. Os métodos podem ser públicos ou privados. Os métodos públicos podem ser livremente utilizados. Os métodos privados somente podem ser invocados por outro método. Somente os métodos podem examinar ou atualizar os atributos privados de um objeto.

3.4.4.2. Variações no encapsulamento

Algumas vezes a definição de encapsulamento tem se mostrado restritiva nas linguagens de programação orientadas a objeto, uma vez que nem sempre é conveniente o encapsulamento de dados junto com métodos. Isto ocorre, por exemplo, quando os dados ou os métodos precisam ser tratados tanto como privados quanto como públicos em partes da definição dos objetos. Para tanto usa-se o conceito de encapsulamento externo permitindo a alteração de métodos privados sem afetar procedimentos externos ao objeto, ou a visibilidade de dados públicos por métodos associados a outros objetos. Assim, a definição tradicional de abstração de tipos de dados (classes) será tratada como encapsulamento de procedimentos. Existem ainda SGBDOOs, como as linguagens baseadas em C++, que permitem que o encapsulamento seja controlado permitindo a definição de "procedimentos livres", isto é, procedimentos independentes de um objeto e que podem manusear diversos tipos de objetos.

3.4.4.3. Dados ativos

A associação de procedimentos com objetos irá resultar em dados ativos, tais como dados derivados, regras e agentes:

. *dados derivados* são aqueles obtidos através de procedimentos sobre atributos ou sobre relacionamentos derivados;

. *regras* ou *gatilhos* são utilizados para codificar conhecimento, sendo executados como gatilhos disparados pelo sistema;

. *agentes*, assim como as regras, são iniciados espontaneamente e permitem, além da atualização do banco de dados, a execução de qualquer processo de uma aplicação.

3.4.5. Tipos e Heranças

As propriedades de um objeto, ou seja, seus atributos, relacionamentos e procedimentos são organizadas em estruturas chamadas *tipos*.

3.4.5.1. Tipos

Os tipos correspondem a um conjunto de propriedades de um objeto. Um tipo de objetos é utilizado para formar uma classe. Entretanto, o conceito de classe pode ser empregado de mais de uma maneira:

. como definição de um tipo de objeto ou intenção, abrangendo a estrutura (atributos e relacionamentos) e o comportamento (métodos) de objetos;

. como definição da extensão de um conjunto de objetos de um determinado tipo, ou, como muitas vezes é empregada, como uma classificação de objetos;

. como definição de uma representação de tipos de objetos, ou seja, como um meta-objeto.

3.4.5.2. Declarações e variáveis

Não existe uma distinção clara entre as linguagens de definição de dados (DDL) e de manuseio de dados (DML) nos SGBDOOs. Nos modelos tradicionais uma DML contém as linguagens de consulta e de programação, enquanto que com as DDL são definidos os tipos de objetos. Nos SGBDOOs uma definição de tipo pode conter procedimentos e variáveis junto aos tipos de objetos e, em alguns sistemas, pode inclusive alterar esta definição durante o tempo de execução. As variáveis num SGBDOO mesmo definidas sem persistência para seus nomes e valores, isto é, de forma transiente, podem ser utilizadas durante a execução de um programa.

3.4.5.3. Tipos de dados literais

Dados literais são constituídos de valores simples como inteiros, datas ou cadeias de caracteres e podem ser utilizados nas linguagens de programação da mesma maneira que uma entidade, ou seja, por um grupo de valores ou por um objeto identificado pelo seu OID. Nas linguagens de programação orientadas a objetos a distinção

entre literais e entidades é mínima devido a utilização da mesma sintaxe de declaração.

3.4.5.4. Hierarquia de tipos

O conceito de generalização nos SGBDOOs implementa a hierarquia entre tipos de objetos, podendo também ser chamado de herança, subtipagem ou subclasse. Como o conceito de classe pode ser usado como intenção, expansão ou representação de tipo de objeto, o conceito de generalização também aceita diversos significados, a saber:

. *especificação* - em muitas linguagens de programação a subtipagem pode ser definida como um tipo de predicado aplicável aos objetos;

. *classificação* - onde os subtipos são utilizados para classificar um conjunto de objetos segundo alguma propriedade comum;

. *especialização* - quando os subtipos definem atributos ou métodos adicionais à seu supertipo;

. *implementação* - para especificar o tipo de implementação de métodos definidos no supertipo, gerando procedimentos próprios.

Como esses conceitos não são mutuamente exclusivos, as linguagens de programação e os sistemas de banco de dados tratam da generalização de forma única, não fazendo nenhum tipo de separação. A combinação de subtipagem com herança permite obter bons resultados na abstração dos conceitos de generalização.

3.4.5.5. Hierarquias avançadas

Alguns SGBDOOs admitem que um tipo possa ser descendente de múltiplos supertipos ou que objetos podem estar associados a muitos tipos, o que confere para estes sistemas uma semântica mais sofisticada do que a dos demais. O conceito de herança múltipla é usado quando um tipo (e um objeto deste tipo) herda as propriedades de diversos supertipos. Assim, consegue-se uma redução no número de tipos e maior simplificação tanto no esquema dos dados, como nos programas de aplicação.

3.4.5.6. Definição dinâmica de tipos

Alguns SGBDOOs admitem definir um tipo em tempo de execução alterando o esquema da base de dados. Esta definição de tipos dinâmicos, que não são previamente declarados, aumenta a complexidade no manuseio dos objetos, uma vez que os programas existentes não conhecem a definição do novo esquema, exigindo que as interfaces de programação sejam compilados para serem utilizados.

3.4.5.7. Montagem dinâmica

A montagem dinâmica, também chamada montagem a posterior, constitui-se na montagem em tempo de execução, das tabelas de símbolos utilizadas numa linguagem. Aplica-se a tabelas de nomes de tipos, de variáveis, de atributos, de métodos e de qualquer outro nome de símbolo utilizado numa linguagem.

3.4.6. Persistência, Concorrência e Recuperação

Os SGBDOOs introduzem novos problemas nos conceitos de concorrência e de recuperação devido aos requisitos dos novos grupos de aplicações. Os dados precisam ser mantidos durante longos períodos de tempo até poderem atualizar a base de dados. Os objetos podem possuir muitas versões que precisam ser identificadas e manuseadas.

3.4.6.1. Persistência

Os conceitos de dados persistentes e transientes nos SGBDOOs são utilizados no sentido da manutenção dos dados após o término da execução de um procedimento. Assim, tornam-se importantes os problemas de concorrência, de recuperação e de integridade lógica e física do banco de dados. Em geral, utiliza-se uma das regras abaixo para assegurar a persistência de dados nos SGBDOOs:

. por tipo - quando a definição do tipo especifica se os objetos instanciados são persistentes ou transientes;

. explicitamente - através de especificação feita na criação do objeto ou por um período de tempo;

. por referência - visando raízes persistentes na definição dos objetos.

3.4.6.2. Transações

Os SGBDOOs expandem o conceito de transações para os mecanismos de controle da concorrência e recuperação do banco de dados. Permitem, ainda, a definição de transações longas, que duram longos períodos de tempo, até mesmo dias, e de transações agrupadas que podem ser guardadas ou abortadas sem afetar a transação principal.

3.4.6.3. Versões de objetos

Os SGBDOOs podem coordenar múltiplas versões de um objeto, permitindo seu manuseio por diversos usuários em diferentes cópias. Sem esse recurso, uma aplicação como a de CASE necessitaria implementar a semântica das versões e contar com um controle explícito da seqüência de versões.

A história das versões pode tornar-se complexa quando é sucedida por uma ramificação que mais adiante poderá voltar a se unir, refazendo um ou mais ramos. Alguns sistemas classificam as versões como transiente, de trabalho ou liberada. Uma versão transiente é transformada em versão de trabalho ao final de uma sessão, ficando então visível aos demais usuários. As versões de trabalho são transformadas em versões liberadas quando forem congeladas, ou seja, quando não aceitarem novas modificações.

3.4.6.4. Configurações

Uma configuração é uma coleção de versões de objetos mutuamente consistentes. Os SGBDOOs implementam configurações através de:

. mecanismos para os usuários gerenciarem as configurações;

. controles automáticos para as configurações, conforme parâmetros definidos pelo usuário.

Para implementar o mecanismo de configurações o SGBDOO deve dispor de comandos para mover, copiar e cortar uma versão.

3.4.6.5. Semântica da concorrência baseada em objetos

Os possíveis conflitos de concorrência na atualização de objetos são reduzidos na medida que somente métodos podem manuseá-los. O controle da concorrência através de métodos específicos permite que os conflitos semânticos sejam tratados de forma mais flexível que os de um simples processo de "*read-write*".

3.4.6.6. Usuário e hora do dia

Outra maneira de controlar concorrência e versões é utilizar o código do usuário e a hora do dia relativos a última atualização (*user-time stamp)*, como mais um atributo do objeto.

3.4.6.7. Granularidade da concorrência

Os mecanismos de controle da concorrência podem operar em outras granularidades do objeto, desde o bloqueio de um objeto até uma granularidade física de páginas ou segmentos.

3.4.7. Outros Tópicos

3.4.7.1. Interfaces com usuário

Apesar de fazerem parte da aplicação e não do SGBDOO propriamente dito os sistemas baseados em SGBDOO geralmente dispõem de linguagem de quarta geração (4GL) ou de sistemas geradores de aplicação que ajudam o usuário a escrever aplicações para o usuário final. Em geral, as interfaces com usuário incluem uma da seguintes ferramentas:

. editores gráficos de esquemas, para manusearem diagramas de esquemas de dados;

. exibidores de objetos, onde podem ser vistos e modificados os objetos existentes.

O O_2 é um exemplo de SGBDOO que incorpora um sistema gerador de aplicações gráficas como ferramenta para o usuário final.

3.4.7.2. Distribuição

Bancos de dados distribuídos, processamento distribuído e acesso remoto a banco de dados são projetados para serem transparentes aos usuários e, assim, são poucos os pontos de modelagem de dados requeridos.

3.4.7.3. Proteção

Os controles de proteção e segurança visam impedir a destruição física dos dados através de programas, de linguagens de consulta ou de ferramentas de usuário final, como também o acesso não autorizado aos dados.

As soluções de proteção e segurança dependem da arquitetura utilizada na implementação do SGBDOO e, geralmente, irão interferir no desempenho do sistema.

Os controles tornam-se mais complexos nos SGBDOOs à medida que a proteção deve considerar objetos compostos e a herança no processo de hierarquia de tipos e subtipos.

3.4.7.4. Desempenho

O problema de desempenho de sistemas é, basicamente, de implementação. Como o tempo de atendimento a uma transação é importante, o desempenho constitui uma grande diferença funcional entre os sistemas.

4. TRABALHO COOPERATIVO APOIOADO POR COMPUTADOR

4.1. INTRODUÇÃO AO CSCW

Os sistemas tradicionalmente desenvolvidos com o apoio de computador visavam a transferência de funções realizadas manualmente para um suporte computadorizado, onde as operações fossem resolvidas com segurança e rapidez, os arquivos controlados e auditados e as informações combinadas de forma a aumentarem o conhecimento sobre os negócios. Com o aumento de potência computacional e a introdução de recursos de comunicação de dados, os sistemas aumentaram a interação com os usuários, melhorando ainda mais a velocidade e a qualidade das operações.

Entretanto, essa ótica na automação de operações isola o indivíduo na execução de suas tarefas, restringindo as interações entre grupos de indivíduos trabalhando de forma cooperativa, que é a forma mais comum de trabalho em qualquer ambiente.

As discussões do problema de trabalho cooperativo apoiado em recursos computacionais resultaram numa nova disciplina conhecida por sua sigla em inglês, *CSCW - Computer Supported Cooperative Work* (trabalho cooperativo apoiado por computador), que tem por objetivo o estudo dos projetos de sistemas cooperativos conjugando aspectos da informática, da ciência cognitiva, da psicologia, da sociologia, da antropologia, da administração e dos sistemas de informações gerenciais.

A visão de administração científica de Taylor, se por um lado permitiu a obtenção de grandes ganhos de produtividade com a especialização, por outro, anulou os benefícios que são obtidos com a sinergia de um grupo. Conhecer o comportamento das pessoas, a forma como pensam, agem e trabalham é essencial para o projeto de sistemas destinados a aumentar o volume e a qualidade dos trabalhos realizados coletivamente.

Em resumo, os trabalhos de CSCW visam preservar a forma com que as pessoas interagem, moldando as ferramentas a estas características, de forma a não desperdiçar o potencial de realização dos grupos.

4.2. TAXIONOMIA DOS SISTEMAS DE CSCW

Os sistemas de CSCW, também chamados *groupware* [OPPE88], costumam ser classificados segundo a interação dos usuários no tempo e no espaço, segundo a natureza da aplicação [ELLI91] e segundo a natureza da interação das pessoas no compartilhamento de mensagens [BONF91]:

4.2.1. Tempo e Espaço

Para a classificação "tempo e espaço", os participantes de um trabalho cooperativo podem estar reunidos no mesmo ambiente ou dispersos geograficamente. Podem interagir diretamente, em encontros do tipo face a face, ou de forma assíncrona, numa seqüência de ações no tempo. Assim, o produto destas dimensões, tempo e espaço, define quatro categorias, conforme apresenta a figura IV.1:

	mesmo momento	momento diferente
mesmo local	interação face-a-face	interação assíncrona
local diferente	interação síncrona distribuída	interação assíncrona distribuída

Figura IV.1- Interação tempo-espaço

. *mesmo local e momento* - característico dos sistemas de edição voltados a interação do tipo face a face como os editores de criação conjunta ou os de idéias, utilizados em ambientes do tipo sala de reuniões;

. *mesmo local em diferentes momentos* - típico dos sistemas de edição para interações assíncronas como os voltados a quadro de avisos;

. *locais diferentes no mesmo momento* - sistemas voltados para processos de interação síncrona distribuída, como os editores de documentos;

. *locais e momentos diferentes* - sistemas voltados para processos de interação assíncrona distribuída, como os editores para revisão de documentos.

4.2.2. Nível da Aplicação

Outro sistema de classificação para as aplicações de CSCW considera o nível de funcionalidade da aplicação. Como um sistema pode apresentar mais funções do que as utilizadas na definição das categorias, ele pode pertencer a várias categorias:

. *sistemas de mensagens* - como correio eletrônico, sistemas de teleconferência e BBS's;

. *editores multi-usuários*, subdivididos em: editores de idéias - aqueles voltados para a concepção de trabalhos, como o Cognoter [STEF87]; editores de textos - editores voltados a edição de documentos, como o Quilt [FISH88]; editores de gráficos - editores voltados a elaboração de desenhos e diagramas, como o Flecha [CAMA92]; editores de composição - editores especializados na produção de publicações, juntando texto com gráficos, quadros e desenhos, como o Shared Book [LEWI88];

. *sistemas para suporte a tomada de decisões em grupo*, aqueles voltados a salas de reuniões eletrônicas equipadas com workstations, projetores de telas, e outros produtos eletrônicos para apoio a discussões e decisões, como o PlexCenter [APPL86], do Laboratório de Planejamento e Suporte à Decisões da Universidade do Arizona.

4.2.3. Tipo de Interação

Bonfiglio *et al* [BONF91] apresenta a taxionomia elaborada por Garcia *et al*, que classifica os sistemas segundo a natureza da interação entre as pessoas ao compartilharem as informações:

. *compartilhamento de mensagens* - típico das aplicações voltadas a interação assíncrona, utilizadas para a revisão de documentos, como o Shared Book [LEWI88];

. *compartilhamento de arquivos* - característico dos sistemas assíncronos com uma estrutura do tipo cliente-servidor, como o Shared Book [LEWI88];

. *compartilhamento de processos* - que caracterizam os sistemas para processos de interação síncrona, em tempo real, normalmente utilizando arquitetura de replicação de processos como o DistEdit [KNIS90].

4.3. PROJETOS DE SISTEMAS CSCW

Apesar da pequena experiência existente no desenvolvimento e uso de aplicações multi-usuárias, algumas questões relativas a projetos começam a ser padronizadas na construção de protótipos e de produtos. Os tópicos a seguir resumem as principais observações pertinentes aos trabalhos publicados.

4.3.1. Arquitetura da Aplicação

Como qualquer aplicação multi-usuária, estes sistemas são baseados em recursos de rede de comunicação de dados, quer seja uma rede de computadores, local ou remota, ou de equipamentos terminais de um único computador. O modelo de rede empregado irá condicionar o tipo de arquitetura dos sistemas, onde normalmente são encontradas variações sobre os modelos básicos de replicação de processos ou do tipo cliente-servidor.

4.3.1.1. Replicação de processos

Na arquitetura de replicação de processos, cada usuário utiliza uma cópia completa do sistema como se fosse uma aplicação mono-usuária. A integração das informações entre os diversos usuários é feita através do acesso compartilhado aos mesmos arquivos ou pela replicação dos arquivos, gerando uma cópia completa para cada usuário. Nesta última situação, quando ocorre uma atualização, feita por qualquer um dos usuários, um sistema de mensagens comunica as alterações introduzidas de forma a manter a consistência entre todas as cópias de arquivos. Apesar dos problemas decorrentes da ineficácia ou inexistência do controle de integridade das informações, esta arquitetura diminui o esforço de reutilização dos sistemas originalmente do tipo mono-usuário para os ambientes de trabalho cooperativo e, com a disponibilidade local das informações, permite melhorar os tempos de resposta das transações.

4.3.1.2. Cliente-servidor

Neste modelo, o mais utilizados pelos sistemas multi-usuários, o servidor é o processo encarregado de gerenciar as funções comuns para todos os usuários sob de-

manda dos outros processos que são chamados de clientes. Em geral, o servidor gerencia, entre outras, a função de manuseio das informações persistentes, encarregando-se do armazenamento, do controle de concorrência e da recuperação das informações, enquanto a interface com o usuário e as funções específicas são resolvidas ao nível do cliente. A comunicação entre o servidor e os clientes é feita através de um sistema de mensagens elaboradas conforme protocolos específicos das aplicações.

Apesar da maioria das implementações ser feita com um único tipo de sistema, essa arquitetura também suporta a utilização de diferentes sistemas pelos usuários, desde que seja assegurada a compatibilidade do sistema de comunicação cliente-servidor. Essa compatibilização pode ser obtida através da adaptação dos sistemas existentes para um único modelo de comunicação ou através da utilização dos chamados *toolkits* para converterem as mensagens trocadas num formato padronizado.

4.3.2. Modelo de Dados

Não existe um modelo de dados típico dos sistemas multi-usuários. Os modelos de dados são definidos pelos tipos de estruturas de dados e das operações disponíveis sobre os dados. Podem variar desde os mais simples arquivos seqüenciais até os modelos de dados não convencionais, como os orientados a objetos.

4.3.2.1. Arquivos convencionais

Alguns sistemas implementam suas próprias estruturas de armazenamento dos dados. O uso de arquivos convencionais transfere as operações sobre os dados para o nível da aplicação. Isso, pode ser vantajoso quando se deseja a otimização de determinadas estruturas de armazenamento de dados requeridas na solução de problemas não convencionais ou quando as restrições do ambiente operacional demandam a construção de soluções específicas.

4.3.2.2. Banco de dados centralizado

A utilização de banco de dados centralizado para armazenamento dos dados simplifica o projeto do sistema, retirando do nível da aplicação os problemas de manuseio dos dados persistentes, de controle de concorrência, de recuperação de informações e os de integridade referencial.

Apesar da liberdade de escolha do modelo, a preferência dos projetistas de sistemas multi-usuário tem sido pelos modelos relacionais estendidos e pelos orientados a objeto devido a implementação de recursos para o manuseio de dados multimídia, como som e imagem, e a utilização de procedimentos para manusear dados encapsulados.

Os gerenciadores de banco de dados que implementam esses modelos, apesar de atenderem as necessidades dos sistemas multi-usuários, costumam ser criticados pelos retardos que introduzem nas transações, dificultando, ou mesmo impedindo, a obtenção de tempos de resposta nas aplicações compatíveis com a velocidade de interação de um grupo trabalhando em conjunto.

4.3.2.3. Banco de dados distribuído

Essa alternativa à utilização de banco de dados centralizado é pouco utilizada devido a dificuldade de ser estabelecido, a priori, um critério de distribuição dos dados entre os usuários, os quais, normalmente, são desconhecidos e apresentam muitas reações não previsíveis. Um modelo de banco de dados distribuído necessita que sejam definidos os critérios que irão otimizar a distribuição dos dados. Assim, poderá ser analisado se a solução supera as restrições operacionais dos arquivos convencionais sem perda de velocidade nas transações ou se melhora a velocidade de manuseio dos dados persistentes com a manutenção da visão do conjunto das informações.

4.3.3. Controle de Concorrência

O controle de concorrência visa resolver os conflitos de acesso simultâneo aos dados efetuados por diversos usuários. Este mecanismo isola do trabalho as porções que estão sendo modificadas até que estejam logicamente concluídas a fim de evitar superposições, ou seja, que a ação de um usuário destrua aquela executada por outro usuário. Este é o caso, por exemplo, de um editor de textos no qual dois usuários resolvem, ao mesmo tempo, um alterar um parágrafo e outro o eliminar. Se o primeiro comando processado for o de eliminação, o comando de alteração entrará em situação de erro por inexistência do texto a ser atualizado.

O controle de concorrência em sistemas de CSCW deve respeitar a natureza do trabalho cooperativo coordenando os diversos usuários que interagem numa mesma sessão. Assim, além de cuidar da integridade dos dados, deve trabalhar com uma granularidade que não inviabilize as ações dos usuários no trabalho conjunto.

4.3.3.1. Bloqueios

Normalmente, o controle de concorrência é feito através de um bloqueio. A porção do trabalho requerida por um usuário para ser modificada é bloqueada aos demais usuários, impedindo operações de gravação. Entretanto, o uso de bloqueios acarreta problemas na definição da granularidade do bloqueio, pelo tempo de sua duração e pela forma de notificá-los aos demais usuários.

As aplicações mono-operadas empregam a forma simplificada de um único bloqueio sobre todo o arquivo e, assim, apenas um usuário é capaz de introduzir novos dados ou de modificar os dados existentes. Os demais usuários ficam restritos a consulta do arquivo ou ao acompanhamento da evolução do trabalho. No outro extremo, temos as aplicações onde os usuários podem estabelecer, num mesmo momento, muitos bloqueios parciais sobre os dados de um mesmo arquivo. Nesta situação, devem ser tomados os cuidados e preocupações habituais para serem evitados os *deadlocks*, ou seja, todos os bloqueios requeridos por um usuário devem ser obtidos simultaneamente. A sobrecarga de processamento com as requisições e concessões de bloqueios acarreta retardos nos tempos de resposta, que podem inviabilizar a utilização deste controle.

Outro problema diz respeito a granularidade do bloqueio. Um bloqueio com maior granularidade pode trazer uma menor sobrecarga ao sistema porém, o tempo de sua duração pode inviabilizar um trabalho de grupo. Isto ocorre quando um usuário mantém uma parte significativa do dados bloqueados por um longo período e, assim, exige a interrupção dos demais trabalhos até sua conclusão. Por outro lado, a sobrecarga com o uso de granularidades muito finas, como a de palavras num texto, pode inviabilizar a interação de um grupo com a introdução de retardos insuportáveis nos tempos de resposta, originados pela excessiva fragmentação dos dados bloqueados.

O uso do mecanismo de bloqueios se aplica tanto aos modelos de dados centralizados como aos replicados ou distribuídos. Para os primeiros, a utilização é simplificada devido a existência de operações próprias nos gerenciadores de dados centralizados. Nos demais, a utilização de um modelo de bloqueios centralizados pode simplificar o controle da concorrência e minimizar o impacto do mecanismo sobre o desempenho da aplicação.

4.3.3.2. Transações

O controle de concorrência também pode ser implementado através do mecanismo de transações o qual terá a seu encargo a requisição dos bloqueios necessários a uma execução. Apesar de úteis em trabalhos assíncronos e de utilizarem plenamente as operações disponíveis nos gerenciadores de banco de dados, o mecanismo de transações não tem sido bem aceito para trabalhos multi-usuários em tempo real. As transações muito grandes são evitadas, pois seu resultado somente é colocado disponível para os demais usuários após ser completada e de seus dados terem sido armazenados de forma persistente, o que nega o princípio da visibilidade dos dados nos ambientes de trabalho cooperativo. Por outro lado, usar transações muito pequenas além de serem dispendiosas podem introduzir confusão na realização do trabalho.

Resta, ainda, observar que o uso de bloqueios pelo mecanismo de transações acrescenta aos acima relacionados os problemas já apresentados com os bloqueios.

4.3.3.3. Protocolos de autorização

Os protocolos para comunicação entre os membros de um grupo, podem ser utilizados como alternativa para a implementação de mecanismos de controle de concorrência. Como exemplo, temos o estabelecimento de um "direito à palavra", onde uma autorização circula pelo grupo de forma que um, e somente um, membro possa se expressar a cada momento. Assim, um usuário somente poderá introduzir modificações nos arquivos quando receber o direito à palavra.

Uma outra alternativa, quase informal, de controle de concorrência é o do "bloqueio tácito", ou seja, um usuário que deseja introduzir modificações consulta previamente o grupo através de mensagens, verbais ou escritas, e pede autorização para modificar determinada parte dos arquivos. A viabilidade deste método depende do tipo de aplicação. Com exceção das aplicações dedicadas a processos onde existe grande interação do grupo feita em tempo real, como os editores de idéias, uma boa organização do trabalho pode facilitar o uso desse mecanismo pois dificilmente dois usuários estarão trabalhando simultaneamente sobre a mesma parte do trabalho, o que torna mínima a possibilidade de conflito.

4.3.3.4. Detecção-dependência

O modelo de detecção-dependência pode resolver os problemas de concorrência sem utilizar o mecanismo de bloqueio. Cada vez que uma modificação é acrescentada ao arquivo, são acrescentados à mensagem a identificação do autor e o momento desta modificação. Quando uma parte dos dados for recuperada para fins de modificação, esta identificação servirá de controle para o momento da atualização. Se o conteúdo do arquivo já tiver sido modificado, ou seja, se a identificação já houver sido mudada, a mensagem de modificação é rejeitada.

O uso do mecanismo de identificação do tempo e do autor, conhecida como *user time-stamp*, permite melhorar o desempenho das aplicações com a eliminação dos retardos provocados pelos bloqueios. Como, normalmente, a introdução de modificações com este mecanismo acrescenta uma nova versão ao trabalho sem eliminar o conteúdo anterior, é possível a construção de mecanismos eficientes para a gerência do processo de edição de um arquivo, como a possibilidade de recuperação da história de todas as modificações de um trabalho ou da colaboração feita por um dos participantes, além de permitir restaurar uma determinada versão antecedente.

4.3.4. Controle de Acesso

O controle de acesso dos usuários aos dados faz parte dos mecanismos de segurança de uma aplicação multiusuária, sendo um importante controle da integridade de um trabalho. O acesso aos dados pode ser limitado para operações de leitura ou gravação, para determinadas operações da aplicação, pode ser válido para todo ou para partes dos arquivos, pode ter duração permanente ou definida de forma dinâmica.

O uso de visões simplifica a implementação do controle de acesso com a distribuição, entre os participantes, de autorizações, conforme o papel do usuário no trabalho. A autorização pode ser específica para um usuário ou para uma classe de usuários e pode ser válida até a ocorrência de um evento que a suspenda, como por exemplo, a conclusão do trabalho, um limite de tempo pré-estabelecido ou mesmo a simples revogação da autorização pelo supervisor da tarefa.

O controle de acesso deve gerenciar dinamicamente suas tabelas de controle para poder acompanhar as freqüentes mudanças de posição dos objetos exigidas, prin-

cipalmente, na fase de elaboração de um trabalho. A mudança de contexto de um objeto deve acarretar a definição de uma atualização na visão e nas suas autorizações.

Finalmente, resta lembrar que os modelos de controle de acesso das aplicações multi-usuárias também aumentam de complexidade por serem destinados aos usuários finais. É importante que o controle de acesso possa ser gerenciado através de uma interface que incorpore facilidades de manuseio, onde os comandos sejam simples e fáceis de serem entendidos, e que seja robusto para suportar os acidentes comuns nesses ambientes.

4.3.5. Interface do Usuário

O termo *WYSIWIG* (*What You See Is What You Get*) é geralmente usado para descrever as aplicações onde é exibida na tela uma imagem idêntica ao do resultado do trabalho, como os editores de texto nos quais o texto aparece tal qual será impresso. Analogamente, o termo *WYSIWIS* (*What You See Is What I See*) criado por Stefik [STEF87] refere-se à apresentação de imagens consistentes com a informação compartilhada por todos os participantes. Assim, uma ferramenta de reunião é estritamente *WYSIWIS* quando todos os participantes vêem exatamente o mesmo. *WYSIWIS* cria a impressão de que os membros do grupo estão interagindo com objetos tangíveis e compartilhados.

Apesar do *WYSIWIS* fornecer a mesma imagem para todos os usuários, pode ser útil fazer uma diferença entre janelas públicas, que podem ser acessadas por todo o grupo, e janelas privadas, com acesso limitado. Nas janelas públicas, as pessoas podem entrar livremente com novas contribuições, aumentando a quantidade de trabalho produzido pelo grupo. Entretanto, como o processo é feito sem que as idéias sejam discutidas ou negociadas, as janelas públicas podem acabar como uma fonte de dispersão quanto aos objetivos pretendidos para o trabalho. As janelas privadas por sua vez, apesar de violarem os conceitos específicos de *WYSIWIS* tais como os ponteiros da tela, permitem que um participante amadureça sua contribuição antes de apresentá-la ao grupo e evitam que o excesso de modificações de pouca importância apresentado numa janela seja mais uma fonte de distração.

Outro relaxamento conceitual admissível, é permitir a colocação de janelas em qualquer parte das telas. Isto permite aos participantes personalizarem seus ambientes de trabalho, melhorando a capacidade de interação com o grupo.

A técnica de WYSIWIS permite a utilização do telecontrole, ou seja, a janela de um usuário é sincronizada com a de outro, refletindo todas as operações exibidas. O telecontrole permite uma interação dos participantes remotos de um trabalho cooperativo compatível com a de um encontro direto, do tipo face a face, conseqüentemente estimulando a sinergia do grupo.

4.4. APLICAÇÕES À BANCO DE DADOS ESTATÍSTICOS

Como qualquer aplicação complexa, a utilização pelo usuário final da interface de acesso a um banco de dados estatístico é dificultada pela interpretação dos comandos da aplicação e pelo manuseio de linguagens formais de acesso, como também é restrita à interpretação do conteúdo pelo nível semântico da informação. Assim, a construção de aplicações multi-usuárias para BDE's permite utilizar o conhecimento de um usuário no apoio a outros, como num grupo de trabalho.

K.Neumann [NEUM90] lembra a necessidade de um alto grau de padronização no processamento estatístico, nas classes de banco de dados e nas interfaces de acesso como condição para se procurar cooperação em banco de dados estatísticos. Em sua visão, o emprego de técnicas de CSCW deve considerar que n pessoas podem trabalhar com m bancos de dados, onde tanto n como m podem variar de 1 até muitos.

4.4.1. Tutor do Sistema

O tutor de uma aplicação BDE, apoiado por CSCW, pode ser definido como o usuário que assume a função de treinamento e suporte aos demais usuários na utilização de um BDE. Como as aplicações CSCW necessitam de uma rede de computadores, ou de terminais de computadores, o conceito implica em facultar a dispersão geográfica dos usuários participantes de uma sessão, tutor e tutelados.

A construção da aplicação, baseada em compartilhamento de arquivos e no emprego de transações curtas, permite utilizar um mecanismo simplificado de bloqueio integral de arquivo por um único usuário operador da aplicação, acompanhado pelos demais participantes da sessão através de interfaces do tipo *WYSIWIS*. O papel de operador pode ser alternado entre os usuários participantes da sessão, permitindo que os treinandos desenvolvam um trabalho conjunto e que sejam assistidos em suas dificuldades pelo tutor.

Como elemento de suporte, o tutor pode analisar e intervir numa sessão estabelecida remotamente. O papel do tutor, neste momento, tanto pode ser no apoio aos problemas operacionais decorrentes de dificuldades na utilização da aplicação como, também, na interpretação semântica da informação. A existência de janelas de comunicação, onde as mensagens possam ser livremente trocadas, permite ao tutor interpretar a pretensão de consulta ao BDE para, a seguir, interagir com o usuário na definição de uma estratégia de busca das informações demandadas.

4.4.2. Equipe Multidisciplinar

A formação de uma equipe multidisciplinar para busca de informações em grandes BDE's permite a utilização do conhecimento específico de cada usuário participante de uma sessão na localização das informações de interesse. A equipe pode estar reunida numa mesma sala em encontros do tipo face a face, ou dispersa geograficamente. Através de um mecanismo de controle de autorizações, um usuário transfere a outro o direito de atualizar listas das variáveis do estudo.

Uma equipe multidisciplinar também pode utilizar uma aplicação CSCW para a indexação da informação contida num BDE. A definição de descritores atendendo a diversos sistemas de classificação, irá enriquecer a semântica da informação armazenada. Os requisitos para a construção de uma aplicação para esta finalidade são, em essência, os mesmo utilizados para o tutor.

5. PROGRAMAÇÃO VISUAL EM SISTEMAS DE COMPUTADORES

Programação Visual aplicada a sistemas de computadores é considerada como a chave que permitirá, a pessoas não familiarizadas com computadores, utilizá-los e mais, ainda, programá-los. Como o desenvolvimento de *software* será feito, não por especialistas em computação, mas sim pelos especialistas no domínio da aplicação, este movimento poderá resultar em ganhos de produtividade no trabalho e de qualidade no desenvolvimento de *software* de aplicação [ICHI90a].

Dos aspectos estudados em programação visual aplicada a sistemas de computadores, destacam-se os estudos em interfaces homem-máquina, utilização de ícones e de linguagens visuais para desenvolvimento de programas com a exploração de novos elementos, como cor, gráficos, vídeos e animação.

Como nas demais aplicações, também os bancos de dados estatísticos podem aperfeiçoar suas interfaces com os usuários utilizando os princípios e as ferramentas desenvolvidas com essa técnica. Os problemas descritos no capítulo 2 relativos à dificuldade de encontrar e extrair informações de um BDE podem ser simplificados, de forma que um usuário não especialista na aplicação possa manuseá-lo com proficiência.

5.1. INTERFACE DO USUÁRIO

As interfaces utilizados para a comunicação homem-máquina apresentam dificuldades de projeto de grande complexidade. As aplicações precisam ser executadas em diferentes locais, com línguas e costumes diferentes. Devem ser utilizadas por pessoas dos mais diferentes perfis e, freqüentemente, nos mais variados equipamentos, com telas grandes ou pequenas, de diferentes resoluções ou policromia, utilizando os mais variados tipos de teclados, *mouses* e "telas-de-toque".

Os componentes utilizados nos projetos da interface homem-máquina, ou como é normalmente denominado, interface do usuário, navegação em menus, operações sobre janelas, comandos textuais (comandos através da digitação de textos), teclas de funções, botões do *mouse*, ícones, etc, acabam sendo de complexa utilização.

Não raro, acabam acarretando muita decisão do usuário para sua utilização. Os métodos adotados para se obter uma informação, muitas vezes, são imprecisos e demandam muito tempo na interpretação e navegação do sistema.

O projeto da interface deve atender tanto as aplicações mono-usuário como as aplicações multi-usuário. A complexidade destes projetos resultou na criação de uma área voltada exclusivamente à pesquisa da interação do homem com o computador.

5.1.1. Evolução Histórica

J.Grudin apresenta em seus trabalhos [GRUD90a] [GRUD90b] [GRUD91] uma revisão histórica dos tipos das interfaces homem-computador, onde observa que, ironicamente, o termo "interface do usuário" é um termo centrado na tecnologia: o computador é assumido e o usuário é especificado. Na realidade, o uso do termo interface do usuário somente se explica numa distorção de origem: na ótica da engenharia do produto, um computador dispõe de muitas interfaces, entre elas, a do usuário.

As distorções nos termos utilizados ajudam a explicar as dificuldades impostas aos usuários de sistemas de computadores, que de especialistas no domínio da sua função acabam se tornando especialistas em sistemas de computação.

O próprio termo "usuário" é empregado na indústria da informática para os usuários de aplicações de computadores e não, por exemplo, aos engenheiros de sistemas que também trabalham com os equipamentos. Ao contrário das demais indústrias onde as instruções de uso de um equipamento são fornecidas em "manuais do proprietário", na de informática produzem-se "manuais do usuário".

Enquanto o termo "usuário casual" é empregado para denominar executivos ou técnicos com pouca habilidade em sistemas de computadores e especialistas em seus trabalhos, reserva-se o termo "usuário especialista" para aqueles que pouco conhecem além das fronteiras do computador. Outras terminologias utilizadas, como "usuário não-profissional" ou "usuário não-especialista" levam a indústria de *software* a tentar desenvolver "programas a prova de idiotas", numa referência explícita aos usuários especialistas em suas tarefas. Todas as referências são centradas no computador, em detrimento do ambiente onde ele se insere.

Nos primeiros computadores, os usuários eram os engenheiros e programadores, os únicos que tinham acesso e conseguiam operar com os equipamentos. A partir da década de 70, com o uso de equipamentos terminais nos escritórios dos usuários, começou a ser feita uma diferenciação, cunhando-se termos como "usuários não-profissionais" ou "usuários finais". Hoje, os usuários finais representam a maioria absoluta dos usuários diretos de computadores, não obstante a ótica da engenharia do produto ainda manter a centralização no equipamento.

5.1.2. A Interface

Define-se como interface do usuário o segmento de um programa que manuseia o diálogo com os usuários. A definição é feita no sentido da interface do computador com o usuário, abrangendo as operações com o teclado, com o *mouse*, com a tela do monitor e com outros dispositivos que requeiram intervenção do usuário, conforme apresentado na figura V.1.

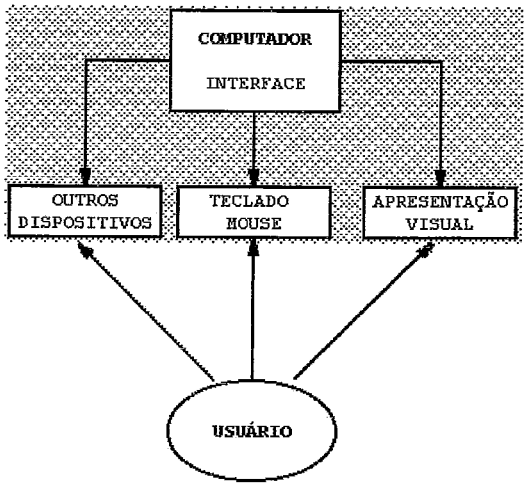


Figura V.1 - Interface do computador com o usuário

A mesma definição se tomada, no sentido do usuário com o computador, deveria abranger as interações do usuário com serviços de manutenção, com a documentação do sistema, com treinamento, com interações com grupos, enfim, com as interações rotineiras das pessoas na realização de seu trabalho conforme pode ser visto na figura V.2.

As novas interfaces começam a apresentar alguns avanços sobre a ótica convencional, implementando funções de documentação e auxílio on-line e tutores para o treinamento dos usuários. Melhorias no *software* e no hardware tendem a eliminar os

intermediários necessários nos sistemas anteriores, como os administradores de sistemas e consultores de informática.

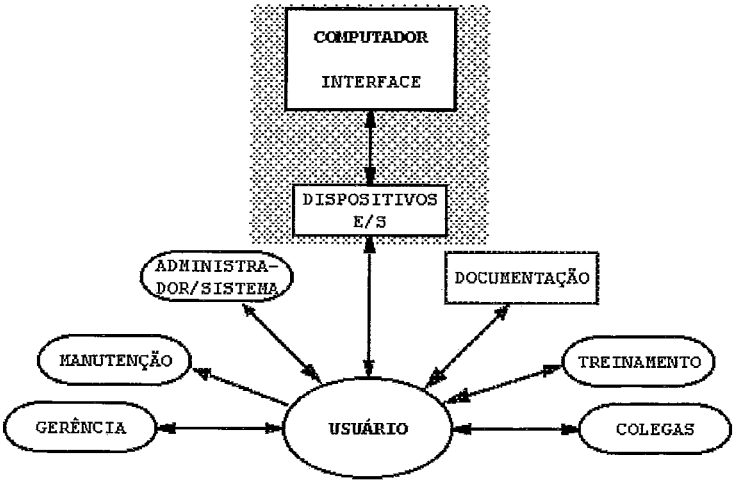


Figura V-2 - Interface do usuário com o computador

Assim pode-se imaginar que as novas interfaces passem a exibir a configuração exibida na figura V-3, onde a interação do usuário passe a ser com uma interface inteligente e com os colegas de trabalho.

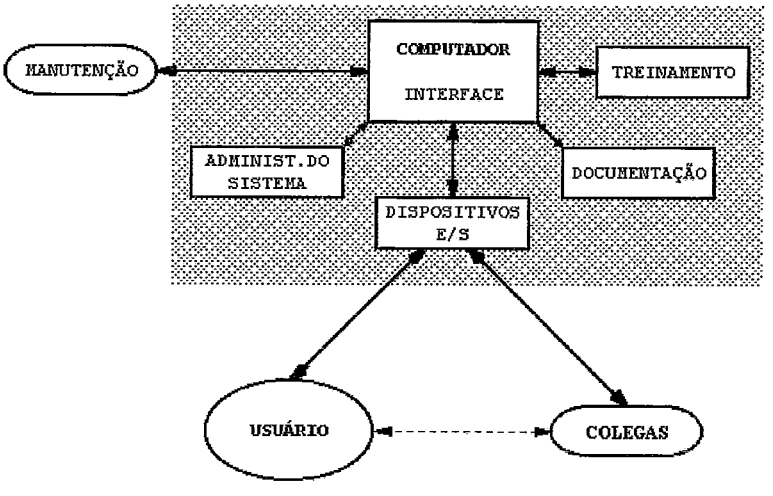


Figura V-3 - Interface do futuro

Entretanto, nem sempre a implementação de novas funções resulta em benefícios. Neerincx e Greef [NEER93] relatam a experiência no treinamento de alunos com o *software* de análise estatística SPSS/PC onde a turma foi divida em dois grupos, um desenvolvendo os exercícios no SPSS/PC com um sistema de auxílio on-line e o outro grupo sem esta função. Nas quatro avaliações semanais efetuadas, o grupo

que trabalhou sem a função de auxílio *on-line* apresentou resultados significativamente melhores do que os do outro grupo. Ao final, o grupo que utilizou o auxílio on-line havia desenvolvido menos 35,5% de comandos corretos do que o outro grupo. A conclusão dos autores foi no sentido que a função de auxílio on-line do SPSS/PC diminui a capacidade de aprendizado do *software* por novos usuários e justificam a assertiva nos erros estruturais que encontraram no projeto desta função.

5.1.3. As Novas Interfaces

O projeto de novas interfaces costuma ser restringido pelas interfaces existentes. A aceitação de uma interface pelo mercado faz com que a indústria de *software* procure explorá-la, evitando o risco da rejeição existente na introdução de novas tecnologias. Entretanto, o desenvolvimento das novas interfaces é inexorável e deve ser efetuado com progressiva incorporação de conhecimento, de forma cumulativa e com a absorção de novas tecnologias. Espera-se que o desenvolvimento de interfaces compostas de equipes multi-disciplinares, com especialistas em sistemas de programação visual, engenheiros de sistemas e especialistas no domínio das aplicações deva vir produzir interfaces mais "amigáveis" do ponto de vista do usuário.

O projeto FRIEND21 - "Future Personalized Information Environment Development" [NONO91] pesquisa a construção de máquinas para a informatização da sociedade no século 21. Os pontos considerados como base para o desenvolvimento de novas interfaces foram:

. a informação como base da comunicação, ou seja, todas as comunicações devem ser realizadas através dos meios de informação;

. estruturação da informação, de forma que a informação utilize as mais diversas estruturas, como hipermídia, para permitir seu uso em aplicações mais complexas;

. distribuição da informação, para que a informação seja mais utilizada para propósitos pessoais, inclusive nas redes de serviços on-line;

. informação para a comunicação entre entidades sociais, no lugar de informação para o suporte a sistemas de produção;

. abrangência universal, permitindo que as pessoas, independentemente do nível de familiaridade com equipamentos ou de inteligência, possam utilizar estas novas máquinas.

A partir destas bases, foram estabelecidas cinco premissas para o desenvolvimento de sistemas de informação para o século 21, a saber:

. premissa 1 - o ambiente de informações deve ser aberto, de forma a ser utilizado a qualquer momento e com qualquer recurso ou função do sistema;

. premissa 2 - a interface dos sistemas deve fornecer informações demandadas pelas pessoas no seu dia-a-dia;

. premissa 3 - o fluxo de operações precisa ser contínuo em todos os aspectos cognitivos, evitando a sensação de estagnação ou ruptura da operação;

. premissa 4 - as ferramentas precisam ser genéricas para serem utilizadas por qualquer tipo de informação;

. premissa 5 - os sistemas precisam ser adaptáveis ao ambiente pessoal do usuário.

Com estas premissas espera-se que a distância entre o homem e a operação dos sistemas seja reduzida tanto nos aspecto da compreensão semântica das interfaces como nos aspectos articulatórios demandados pelas características dos sistemas.

5.2. ARQUITETURA DAS INTERFACES

Duas arquiteturas dominam a construção de interfaces: Sistemas de Gerência de Interfaces do Usuário (SGIUs) e conjunto de ferramentas conhecidos como *toolkits*. Os SGIUs separam a aplicação da interface. Os procedimentos computacionais são segregados dos procedimentos e estilo do diálogo com o usuário. Os *toolkits* separam o estilo da aplicação. O controle do diálogo é feito pelo programa que utiliza uma biblioteca de códigos para implementar o estilo da interação [WIEC90].

5.2.1. Sistemas de Gerência da Interface do Usuário

Analogamente aos SGBDs, os SGIUs dividem uma aplicação em duas camadas. No nível da aplicação são implementados os procedimentos enquanto no nível da interface são implementados os detalhes de apresentação e interação com o usuário. Com a separação da aplicação em camadas fica mais simples sua reutilização em outros ambientes e sob outras interfaces. Inicialmente, esta técnica teve aceitação limitada devido a pouca clareza na separação das camadas, da inexistência de ferramentas de uso geral para o desenvolvimento de interfaces do usuário e do excessivo enfoque na construção dos diálogos em detrimento da definição de estilos de apresentação.

Sistemas mais recentes passaram a decompor a interface do usuário nos níveis dos objetos, comandos e controles. Em alguns, o estilo é gerado automaticamente devendo o projetista se preocupar apenas com o controle do diálogo. Os sistemas que permitem ao projetista controlar o estilo da apresentação fornecem ferramentas para a edição manual, no estilo WISIWYG, ou são editores baseados em restrições com geração automática através de regras de estilos. As restrições descrevem como os objetos são reposicionados quando uma janela muda de lugar ou de tamanho.

5.2.2. *Toolkits*

Os *toolkits* diferenciam-se dos SGIU por não separarem do nível da aplicação a interface. Utilizam uma biblioteca de códigos, com componentes necessários à construção da interface, como tipos de janelas, caixas prontas para diálogos, menus, botões, etc. Como nas aplicações convencionais, as aplicações que usam os *toolkits* controlam a seleção de técnicas e o fluxo da interação com o usuário. Para trocar uma técnica, por exemplo, substituir um comando textual por uma seleção dentro de um menu, o programador precisa alterar seu programa.

Os *toolkits* começaram a ser utilizados nos sistemas de janelas e hoje são empregados em, praticamente, todos ambientes os operacionais. Alguns conjuntos de ferramentas experimentam grande sucesso comercial por reduzirem a complexidade de programação no desenvolvimento de interfaces e por permitirem o controle do estilo de interface utilizada.

As críticas à utilização dos *toolkits* se concentram na obrigatoriedade do trabalho formal de programação para implementar ou modificar um componente da interface. Por exemplo, para criar um relógio analógico como componente de uma interfa-

ce, deverá ser codificado uma nova rotina no *toolkit* que contenha as primitivas necessárias para instanciar o novo objeto, para posicioná-lo na janela, para definir seu tamanho, etc. Faltam ferramentas para que um artista gráfico possa desenvolver o mesmo trabalho sem o auxílio do programador.

Outro problema com os *toolkit*, diz respeito à consistência com o estilo. Os *toolkits* pouco ou nada captam da interação do usuário sobre o estilo da apresentação. Somente os *toolkits* com estilos baseados em regras asseguram que as alterações da interface efetuadas pelo usuário mantenham a consistência entre os componentes.

5.2.3. Princípios para a Construção de Interfaces

Wu e Hsiao ao desenvolverem o projeto GLAD (Graphics LAnguage for Database) [WU89], uma interface visual para banco de dados, enumeram seis princípios básicos que devem ser observados na construção de interfaces:

. poder fornecer mais informações quando requerido. O princípio deve ser entendido além de um simples sistema de auxílio on-line. Deve abranger toda a sorte de detalhamento da informação e dos comandos de manuseio das funções, para uma melhor interpretação da semântica de uma informação e da operacionalidade de uma função;

. poder recuperar operações errôneas ou não intencionais. Isto significa ser capaz de recuperar o estado anterior ao da execução de um comando e se for impossível desfazer uma operação, esta somente deve ser executada após confirmação do usuário;

. poder realizar a mesma operação de mais de uma maneira, ou seja, a interface deve dispor de atalhos para que o usuário experiente no sistema possa executar diretamente uma função sem ter de percorrer um caminho longo;

. poder realizar operações logicamente equivalentes de maneira consistente, ou seja, em qualquer ponto da interface o uso das funções deve obedecer ao mesmo estilo de utilização;

. poder exibir muitas informações simultaneamente, ou seja, uma tela não deve ser restrita a exibição de um único assunto ou objeto. Deve poder exibir todas

as informações relacionadas, mesmo aquelas manuseadas em diferentes pontos do sistema;

. poder exibir muitas visões da mesma informação, ou seja, o formato de exibição de uma informação deve apresentar alternativas para ser melhor entendida pelo usuário.

K.T.Huang [HUAN90] lembra que o estabelecimento de padrões e diretrizes não são suficientes para o projeto de uma boa interface e que o aprendizado com outros sistemas existentes pode ser de grande valia. Aos princípios apresentados, Huang acrescenta mais algumas recomendações, resumidas na experiência de muitos pesquisadores:

. utilizar desenhos intuitivos. Os comandos visuais, tais como os ícones, podem ser poderosos se forem capazes de serem usados intuitivamente. Os sinais internacionais de tráfego são um bom exemplo de uma simbologia pobre por não ser intuitiva;

. evitar o uso de posições especiais no teclado, como o emprego de duas teclas simultaneamente. Quanto mais simples for a interação mais a vontade ficará o usuário frente ao sistema;

. antecipar-se ao usuário quando, obrigatoriamente, só exista um caminho a seguir. O usuário não deve ser forçado a responder quando não existam opções para serem selecionadas. Se o diálogo passa pela seleção de um item numa lista que, no momento, é unitária, então a seleção é feita pelo sistema sem a intervenção do usuário;

. a consistência de padrões deve ser sempre observada. Se <tecla> significa uma ação, então <shift> <tecla> deve significar a ação reversa. Se o botão da esquerda do *mouse* apaga uma marca, então o botão da direita refaz a marca.

5.2.4. Interfaces Gráficas

Os projetos de interface do usuário vem sendo dominados pelas interfaces gráficas, em substituição às interfaces de comandos textuais, muito utilizadas no passado.

As interfaces gráficas simplificam a implementação de um diálogo com o usuário. A complexidade do diálogo e o esforço de análise gramatical das interfaces controladas por comandos textuais resultam em maiores limitações de alternativas para as interfaces do usuário do que as aparentes restrições dos "*menus-driven*" das interfaces gráficas. Com as interfaces gráficas dirigidas por menus, podem ser construídos grafos orientados para seleção de funções ou obtenção de informações nos sistemas.

Mesmo o uso de teclas dedicadas a execução de funções, apesar da velocidade de interação que propiciam aos usuários experimentados no sistema, apresentam o grande inconveniente de não serem apropriadas à utilização por usuários "novatos" ou "casuais".

Uma interface gráfica utiliza os seguintes componentes:

. botões ou ícones - utilizados para ativar funções;

. botões de seleção única (*radio box*) utilizados para selecionar uma opção entre várias oferecidas pelo sistema;

. botões de opções (*check box*) utilizados para a escolha das opções desejadas pelo usuário;

. menus - para seleção de uma função entre várias;

. listas simples - para escolha de um item ou texto dentro de uma lista;

. listas múltiplas - para escolha de diversos ítens ou textos, dentro de um lista;

. lista editável - para o usuário escolher um item ou texto dentro de uma lista e/ou editar seu conteúdo;

Outra preocupação deve ser com a internacionalização das interfaces, ou seja, para facilitar o uso de sistemas em países com diferentes línguas e costumes, mantendo o respeito pelos principais elementos culturais. P.Russo e S.Boor [RUSS93], acham que devem ser avaliados os seguintes aspectos no projeto da interface do usuário:

. *textos* - de forma a tornar simples sua translação para outros idiomas. Devem ser evitados o uso de expressões idiomáticas e o alfabeto utilizado deve permitir a representação de caracteres utilizados em outros idiomas;

. *imagens* - devem permitir ser reconhecidas e aceitas por outras culturas. Como as palavras, nem sempre existe a possibilidade de translação de uma imagem para outra cultura o que irá condicionar a sua aceitação;

. *números, datas e horários* - devem poder ser expressos nos diversos formatos utilizados no mundo;

. *símbolos* - como nas imagens, a interpretação dos símbolos pode mudar através das culturas;

. *cores* - a interpretação do sentido das cores é outra característica cultural que varia através do mundo;

. *disposição* - a disposição com que textos e gráficos são percebidos varia através dos povos tanto no sentido horizontal como no sentido vertical;

. *funcionalidade* - a funcionalidade de um sistema aceita por uma cultura pode não ser por outra, com maior ou menor rigidez nas relações pessoais.

5.3. ÍCONES

A disponibilidade de novos equipamentos, de baixo custo, com alta qualidade de resolução gráfica tem permitido o emprego intensivo de imagens nas interfaces homem-máquina. A utilização de linguagens de comunicação visual tem permitido representar dados, operações e processos de forma mais natural, reduzindo a complexidade e o esforço mental de sua interpretação, com o emprego de ícones.

Os ícones são desenhos, ou figuras, utilizados para transmitir idéias ou ações, de forma não verbal. Os ícones, de acordo com o contexto, podem representar objetos ou processos. Os "ícones objeto" são utilizados como elementos de comunicação visual, representando atos ou fatos, enquanto os "ícones processo" são utilizados como elementos de uma linguagem de programação visual, representando funções ou processos. Um mesmo símbolo, como por exemplo o símbolo de parar utilizado na sina-

lização do trânsito, pode representar um objeto da interface de um programa, ou representar um processo de uma linguagem de programação visual.

Um sistema de ícones é um conjunto estruturado de ícones relacionados, composto por ícones simples ou complexos, onde os ícones complexos são aqueles compostos por dois ou mais ícones simples e que expressam um conceito visual derivado da composição efetuada.

5.3.1. Taxionomia dos Ícones

Lodding, segundo Chang, classificou os ícones segundo seu projeto ou sua função. Em sua classificação, existem três tipos de ícones:

. *ícones representacionais* - expressados por figuras que lembrem visualmente a idéia ou ação representada;

. *ícones abstratos* - expressados por símbolos expressados por analogia à idéia ou à ação representada;

. *ícones arbitrários* - expressados por sinais escolhidos em conjuntos compostos por imagens que arbitrariamente convenciona-se representar uma idéia ou ação.

5.3.2. Representação Formal de Ícones

Chang [CHAN90] define um ícone objeto, ou simplesmente ícone, com uma representação dual, representada por uma dupla (X_m, X_i) onde X_m representa a parte lógica, o significado, e X_i representa a parte física, a imagem. A correta interpretação de um ícone depende da boa visualização da imagem e da correta captação, efetuada dentro de um contexto.

Um sistema de ícones pode ser formalmente representado pela n-upla

G (VL, VP, S, x_0, R) onde:

VL é um conjunto de objetos lógicos (significados)
VP é um conjunto de objetos físicos (desenhos)
S é um conjunto finito, não vazio, de nomes de ícones

x_0 é o primeiro elemento de S
R é o mapeamento de S em $2^{VL \cup S}$ x VP, representando as regras dos ícones

Operadores de um sistema de ícones são símbolos para operações efetuadas sobre a representação dual dos ícones, ou seja tanto na parte lógica como na parte física. Um operador de ícones *OP* tem duas partes, OP_m para a parte lógica e OP_i para a física.

$$OP = (OP_m, OP_i)$$

Assim, uma operação com dois argumentos, X e Y, é representada:

$$OP\ (X, Y) = [OP_m(X, Y), OP_i(X, Y)]$$

5.3.3. Princípios para a construção de Ícones

Lodding, citado por Chang [CHAN90], sugere que a construção de um ícone seja feita em três etapas: na primeira é feita a escolha de um desenho para o ícone; na segunda são analisadas as possíveis interpretações do desenho; e, na terceira, é feito um teste com o ícone resultante.

Liu e Tai [LIU89], partem dos estudos de Chang para analisar o mais antigo dos sistemas de ícones em uso, constituído pelos caracteres chineses. Estes caracteres, chamados de pictogramas, vem evoluindo de formas antigas, de pequenas figuras, para as formas modernas, de ideogramas. Geralmente tem um pequeno significado quando vistos isoladamente porém, após serem combinados, passam a representar idéias completas, de amplo significado.

Observando-se as regras de formação dos caracteres chineses, pode-se inferir princípios para a construção de ícones, descritos nos ítens a seguir.

5.3.3.1. Aparência dos ícones

Os ícones podem se dividem quanto a sua aparência nos seguintes grupos:

. *seres e objetos naturais*, tais como são vistos na natureza. Quando não possuem um formato visível, utilizam um gráfico que os represente. Como exemplo temos os ícones representando montanhas, raio, a lua, etc.;

. *objetos artificiais* construídos pelo homem, como o ícone de um diskette, etc.;

. *objetos abstratos*, os símbolos definidos para a representação de um circuito elétrico ou qualquer outra convenção.

5.3.3.2. Ênfase nas características

Marcar uma determinada posição de uma imagem ou adicionar marcas aos ícones pode melhorar a interpretação do seu significado, como a colocação de setas, de arcos, de cores, ou de sublinhados.

5.3.3.3. Referências a similaridades ou a abstrações

Criar por similaridade uma parte do ícone que represente a classe de significado e acrescentar uma abstração que represente o sentido desejado.

5.3.3.4. Copiar outras representações

Outro princípio que pode ser usado consiste em copiar ícones cujo significado seja conhecido e utilizá-los em outros contextos. Uma solução clássica deste princípio consiste na utilização de uma taça para representar fragilidade.

5.3.3.5. Integração de imagens

A formação da imagem do ícone pode ser feita a partir da combinação de ícones existentes. A integração das imagens poderá produzir um novo conceito, composto pelo significado dos primitivos. Usando o símbolo de proibido conjugado com o de cigarro obtém-se um novo símbolo significando proibido fumar.

5.3.3.6. Combinação de gráficos com textos

A construção de um ícone pode ser completada com um texto para especificar um determinado conceito dentro do significado do ícone. Acrescentando-se o texto "mensagem" ao ícone de um radio obtem-se um novo ícone que significa envio de mensagens pelo rádio.

5.3.4. Ambigüidade de Ícones

A inexistência de uma padronização e de convenções universais para os conjuntos de ícones sujeita a interpretação das imagens a problemas de ambigüidade. O mesmo ícone pode permitir diversos significados diferentes. Alguns ícones são inerentemente ambíguos enquanto outros somente são interpretáveis em seu correto contexto.

A ambigüidade também pode ser originada na falta de um comportamento dinâmico do ícone, ou seja, pela ausência de informações adicionais que a elimine. Estas informações podem ser gráficas ou podem ser relativas a clareza do contexto. Apesar dos problemas de ambigüidade, a comunicação através de ícones é mais eficiente e melhor recebida pelos usuários.

5.3.5. Animação

O uso de ícones estáticos, muitas vezes é suficiente para a correta interpretação do seu significado, mesmo por usuários "casuais". Outros, mesmo para usuários experimentados no sistema, causam confusão exigindo a disponibilidade de textos de apoio a sua interpretação. Um borracha pode trazer, intuitivamente, a noção de uma ferramenta do sistema para eliminação de dados. Porém, o usuário deverá obter informações adicionais para seu manuseio. O uso de ícones animados, permite aumentar a percepção do significado e da forma de utilização de uma função.

Baecker et al [BAEC91] observam que cada detalhe da animação deve ser claro para evitar más interpretações. Nas experiências realizadas, verificaram uma grande dificuldade em criar animações que fossem suficientemente genéricas para que evitassem confusões. As expectativas e o contexto do usuário precisam ser considerados no projeto das animações para minimizar o impacto que tem sobre a interpretação.

A utilização de som no processo de animação também ajuda a melhorar a interpretação do usuário. O simples barulho do clic de um *mouse* durante uma animação ajuda o usuário a prestar mais atenção as fases de um desenho.

Finalmente, deve ser observado que, nas experiências realizadas, os usuários apresentaram grande hesitação no momento do uso de ferramentas apresentadas com processos de animação.

5.4. LINGUAGENS DE PROGRAMAÇÃO VISUAL

Uma "linguagem de programação visual" ou, mais simplesmente, uma "linguagem visual" é um conjunto de sentenças de ícones, construídas segundo uma sintaxe e uma semântica conhecidas. Uma "sentença de ícones" ou "sentença de ações" ou "sentença visual" é um arranjo espacial de ícones, tomados de um sistema de ícones [CHAN90].

A análise sintática de uma linguagem visual corresponde à análise dos arranjos espaciais dos ícones, segundo sua. A análise semântica corresponde à interpretação espacial, ou seja, à interpretação de uma sentença de ícones determina um sentido, uma idéia ou uma ação. No exemplo da figura V.4, temos um sistema de ícones com os símbolos para avião, tripulação e horários.

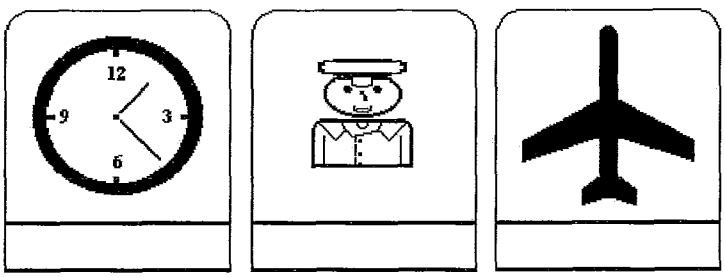


Figura V-4 - Dicionário de ícones

Com estes elementos podemos, então, formular sentenças como "Qual a tripulação do vôo 463 da Delta?" (figura V.5a).

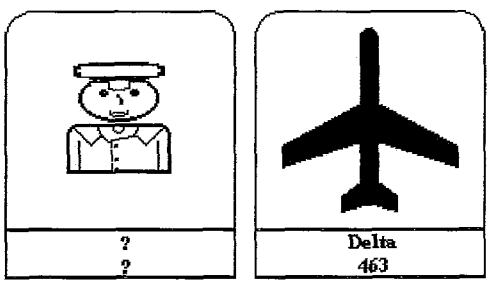


Figura V.5a - Sentença visual

Ou então, "A que horas chega o vôo 463 da Delta?" (figura V.5b).

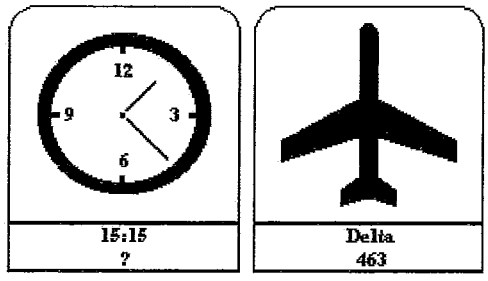


Figura V.5b - Sentença visual

A propriedade mais importante de uma linguagem visual é a capacidade de expressar um conceito utilizando apenas imagens, como na engenharia de *software* onde a representação da idéia é exposta em diagramas de fluxo de dados, de entidades-relacionamentos, em fluxogramas de programas e em diagramas de módulos, etc., todos utilizando formas gráficas para expor idéias.

As experiências a seguir apresentadas descrevem algumas das experiências realizadas na construção de linguagens de programação visual.

5.4.1. HI-VISUAL

Hirakawa, Ichikawa e Tanaka apresentam um sistema de programação por ícones [HIRA90] [ICHI90b] denominado HI-VISUAL. A linguagem foi proposta, inicialmente, em 1986, como uma linguagem de programação com ícones com suporte à programação de interações visuais. Em seguida, em 1987, HI-VISUAL foi expandida para um ambiente de programação com muitos recursos para a programação por ícones.

Como outros sistemas de programação por ícones, HI-VISUAL é programado sobre uma espaço bidimensional, onde os ícones são posicionados e relacionados através de conexões num processo de alta interação homem-máquina. A efetividade do processo é condicionada pelo conjunto de ícones disponíveis. Em geral, os ícones para representar dados são mais simples de serem obtidos do que os destinados a representação de funções. Para a solução deste problema, HI-VISUAL implementa um espaço de gerência de ícones onde só existem símbolos para os objetos reais. Para as funções são utilizadas combinações de dois ou mais ícones, cujo sentido é determinado dinamicamente pelo contexto onde ocorrem.

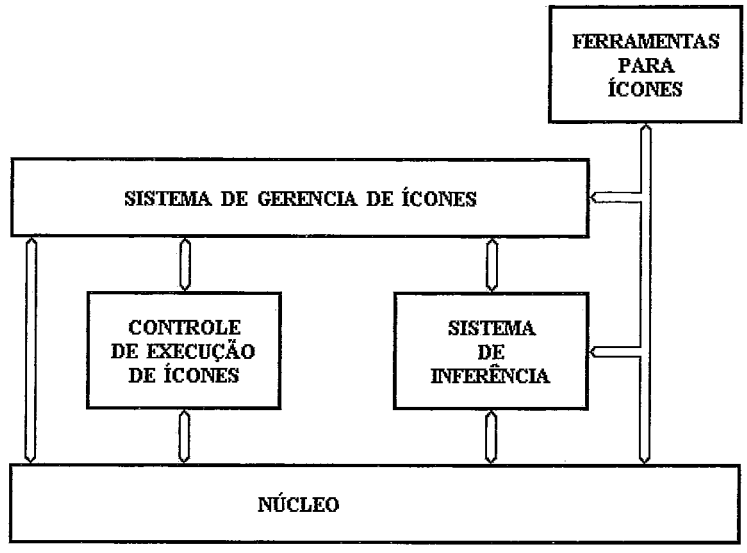


Figura V-6 - Estrutura do sistema HI-VISUAL

A versão atual da HI-VISUAL incorpora ao espaço de programação o uso de regras destinadas a dar mais flexibilidade na interpretação dos ícones. Uma simples adição ou modificação das regras pode mudar o comportamento do sistema frente ao mesmo par de ícones. A estrutura do HI-VISUAL pode ser vista na figura V-6, onde aparecem em destaque seus 5 grandes módulos, a saber:

. *Núcleo (Kernel)* - suporte das primitivas do sistema tais como controle de processos, manuseio de dispositivos, manuseio da tela e das janelas (apoiado pelo X-Windows) e gerência do banco de dados;

. *Sistema de Inferência (IFE - Inference Engine)* - suporte na interpretação de regras em conexão com o SGBD do Kernel;

. *Controle de Execução de Ícones (IEM - Icon Execution Manager)* - que gerencia a interpretação e a execução de programas com ícones;

. *Sistema de Gerência de Ícones (ISM - Icon System Manager)* - que supervisiona os outros quatro componentes;

. *Ferramentas para Ícones (ICT - Icon Tools)* - que fornece as ferramentas utilizadas para manusear e programar com ícones.

HI-VISUAL foi implementado como protótipo em workstations SONY NEWS sob o sistema operacional UNIX. A implementação foi feita em C e foi utilizado o sistema X-Windows.

5.4.2. QBD*

O sistema QBD* - Query by Diagrams, implementa um modelo de dados, uma linguagem de consulta e uma interface gráfica para o usuário [ANGE90], sendo destinado para usuários "não especialistas" entenderem o conteúdo e recuperarem informações de um banco de dados.

QBD* foi desenvolvido a partir de quatro idéias básicas:

. representar a parte intencional de um banco de dados através de um modelo conceitual (modelo E-R expandido com a introdução da generalização de abstrações);

. utilizar intensivamente uma interface gráfica amigável;

. estudar as propriedades formais de uma linguagem gráfica a partir da definição de sua isomórfica textual;

. estender os construtores gráficos da linguagem de forma a expressar facilmente a classe de consultas recursivas.

A arquitetura do sistema é composta por três módulos principais, conforme apresentado na figura V.7, a saber: uma interface gráfica, um tradutor de consultas e uma interface com o SGBD.

A interface gráfica permite a realização de consultas ao banco de dados. A consulta é expressa por um esquema E-R através de comandos gráficos da interface e pode se estender pelas quatro biblioteca do sistema: biblioteca de esquemas E-R; biblioteca de esquemas E-R do usuário; biblioteca de esquemas E-R top-down; biblioteca de consultas E-R do usuário.

O tradutor analisa as consultas e, usando a biblioteca de esquemas lógicos do banco de dados, traduz as consultas não recursivas para expressões da álgebra relacional e as recursivas em programas.

A interface com o SGBD traduz as expressões algébricas em consultas. Caso o SGBD não aceite instruções interativas, o sistema constrói programas na linguagem hospedeira do banco.

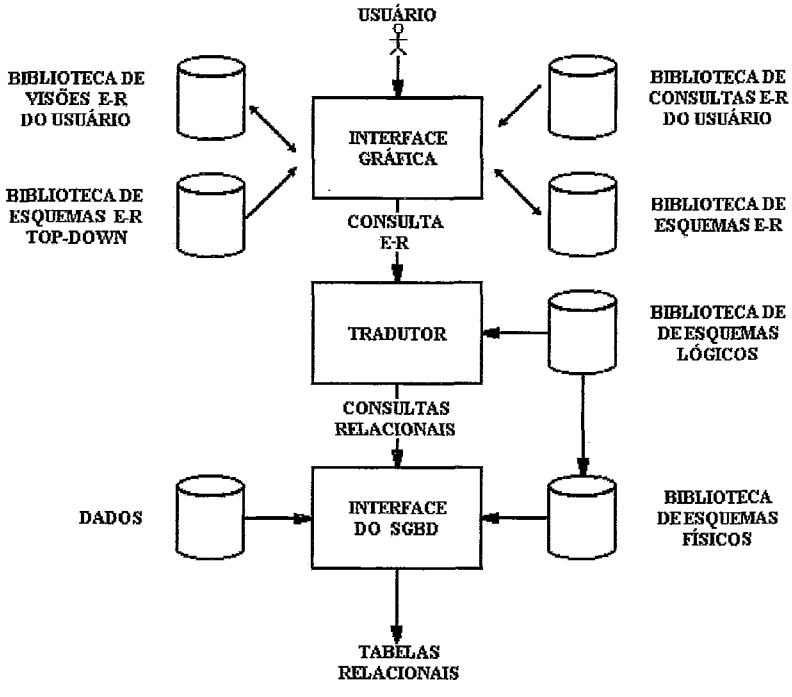


Figura V-7 - Estrutura do QBD*

Assim, um usuário processa uma consulta em três passos:

. o usuário, utilizando a interface gráfica, constrói uma consulta sobre o esquema E-R do banco;

. a consulta E-R é traduzida numa expressão relacional;

. a interface do SGBD formula a consulta nos termos da linguagem adequada ao SGBD utilizado.

Para a formulação da consulta na interface gráfica, o usuário procede da seguinte maneira:

. seleciona o ícone da função "navegação";

. escolhe o principal conceito da consulta com a ajuda do *mouse*;

. com a ajuda de uma lista auxiliar, define as condições para os atributos a serem recuperados (filtros);

. define o caminho para início da consulta relacionando a cadeia de filtros ou outras entidades do esquema;

. seleciona a entidade desejada;

. define as condições "cíclicas" de recuperação (filtros de conteúdo);

. especifica as condições de formação do conjunto resultado.

O protótipo do QBD* foi desenvolvido em ambiente MS-DOS, utilizando a linguagem de programação C e para o desenvolvimento da interface gráfica foi utilizado o HALLO. A próxima versão do QBD* deverá ter como objetivo o manuseio de um banco de dados estatístico.

5.5. CONCLUSÃO

As interfaces do usuário tendem para um processo de especialização, onde o emprego apenas de técnicas de informática não é suficiente para seu desenvolvimento. Os projetos de interfaces devem ser vistos como projetos multidisciplinares, com suas responsabilidades distribuídas entre especialistas de áreas tais como comunicação visual, psicologia perceptual, cognitiva e social, profissionais em lingüística, em vídeo e em som, enfim com especialistas nos mais diversos campos abrangidos.

O emprego das técnicas de programação visual para a construção da interface do usuário de um BDE permite resolver os problemas de acesso ao banco descritos no item 2.6. Um bom exemplo é o QBD*: com sua navegação por ícones e listas, este sistema exibe de uma forma simples os objetos, atributos e relacionamentos existentes no banco.

6. AMBIENTE PARA BANCOS DE DADOS ESTATÍSTICOS

Neste capítulo apresentamos a proposta de um ambiente de banco de dados estatísticos orientado a objetos. A proposta compreende a substituição do modelo de dados relacional por um orientado a objetos, visando incorporar às ferramentas de administração e manuseio do banco os recursos deste novo modelo de SGBD como, também, propiciar a utilização intensiva de recursos de programação visual através do uso de interfaces gráficas homem-máquina. A proposta compreende, ainda, a incorporação de técnicas de trabalho cooperativo apoiado por computador (CSCW), tanto em regime de operação síncrona como em regime assíncrono distribuído, com o objetivo de implementar a figura do tutor do sistema e de permitir sua operação conjunta por uma equipe multidisciplinar.

No capítulo 7 é apresentado o protótipo desenvolvido com os requisitos especificados. Com exceção das funções de trabalho cooperativo que necessitavam de uma versão multi-usuária do SGBDOO, versão esta não disponível durante o desenvolvimento do projeto piloto, todas as demais foram implementadas e testadas. No anexo b encontra-se a descrição detalhada do projeto piloto. O manual do sistema contendo as especificações de uso e a programação desenvolvida encontra-se em publicação no Projeto TABA.

6.1. CONCEPÇÃO

Figura VI-1 - Ambiente do BDE

A concepção do ambiente para banco de dados estatísticos, proposta nesta tese, está divida em cinco grandes funções, conforme o diagrama da figura VI-1. Uma função gerencia os recursos de dados do ambiente, formados pela base de objetos estatísticos e de metadados, outra, da manutenção do banco e as demais funções tratam da interação com o usuário. A implementação destas funções pressupõe um conjunto de recursos operacionais composto por *workstations*

operando em rede, o sistema operacional multi-tarefa *Unix*, o gerenciador gráfico *Motif* para a interface do usuário e o sistema gerenciador de banco de dados orientado a objetos O_2 [BANC88] [O2-91].

A proposta prevê a utilização do modelo de dados orientado a objetos tanto para as informações produzidas por pesquisas estatísticas como para os metadados que as descrevem. Prevê, também, a utilização de uma interface gráfica para as interações homem-máquina e a realização de consultas e extração de dados de forma cooperativa, onde vários usuários podem participar de uma mesma sessão, simultaneamente ou não.

Com o ambiente proposto, o usuário pode consultar estruturadamente os descritores das informações, os conceitos utilizados na investigação estatística e os sistemas de classificação em uso. Pode, também, acessar os objetos estatísticos utilizando uma lista das propriedades das pesquisas (o dicionário da pesquisa) ou o índice de assuntos organizado pelo sistema de metadados.

A seleção das informações utiliza as dimensões espacial, temporal e a categoria de classificação da informação (assunto). Pode-se, também, utilizar filtros que restrinjam os objetos segundo o valor de um atributo comparando-o com valores fixos ou com o conteúdo de outros objetos.

Os resultados de uma consulta podem ser exibidos na tela das *workstations*, a escala de representação dos valores pode ser alterada e os dados resultantes podem formar uma nova base de dados, a base do usuário, onde, através da linguagem de consulta do SGBDOO O_2 (SQL) os dados podem ser analisados em estrutura formal, sem o emprego de atributos referenciais que tornem a navegação mais complexa. E da onde os dados podem ser reformatados em arquivos convencionais e importados por *softwares* de análise estatística.

As ferramentas para administração do banco de dados apoiam o responsável pelas tarefas de manutenção do esquema, de manutenção dos recursos genéricos do sistema e de carga dos dados.

O sistema de indexação utiliza os descritores das informações existentes na base de metadados, associando-os com os atributos da base de objetos estatísticos. Esta operação é registrada no dicionário da pesquisa, construído como uma tupla na classe *Ocorrencia da pesquisa*. A manutenção dos descritores obedece a um vocabu-

lário de termos controlados, isentos das dimensões temporal e espacial e sem ser relacionado com uma determinada grandeza, de forma a especializar as categorias e classificações no aspecto conceitual. A idéia é manter as descrições da informação de uma forma natural, sem a preocupação com a forma com que foi registrada no banco de dados.

6.2. ARQUITETURA DO AMBIENTE

A arquitetura do ambiente compreende os módulos apresentados na figura VI-2 que implementam as funções, agrupando-as em três aplicações. O conceito utilizado para *aplicação* nesta tese é o de um grupo de programas conforme definido pelo SGBDOO O_2. As aplicações Apoio e Metadados destinam-se ao administrador do sistema e podem atualizar a base de dados, representada em duas partes distintas: a base de objetos estatísticos e a base de metadados. A aplicação BDE utiliza estas bases para a realização de consultas e pode gravar ao final uma nova base de dados, a base do usuário, composta com objetos definidos pelos atributos selecionados pelo usuário.

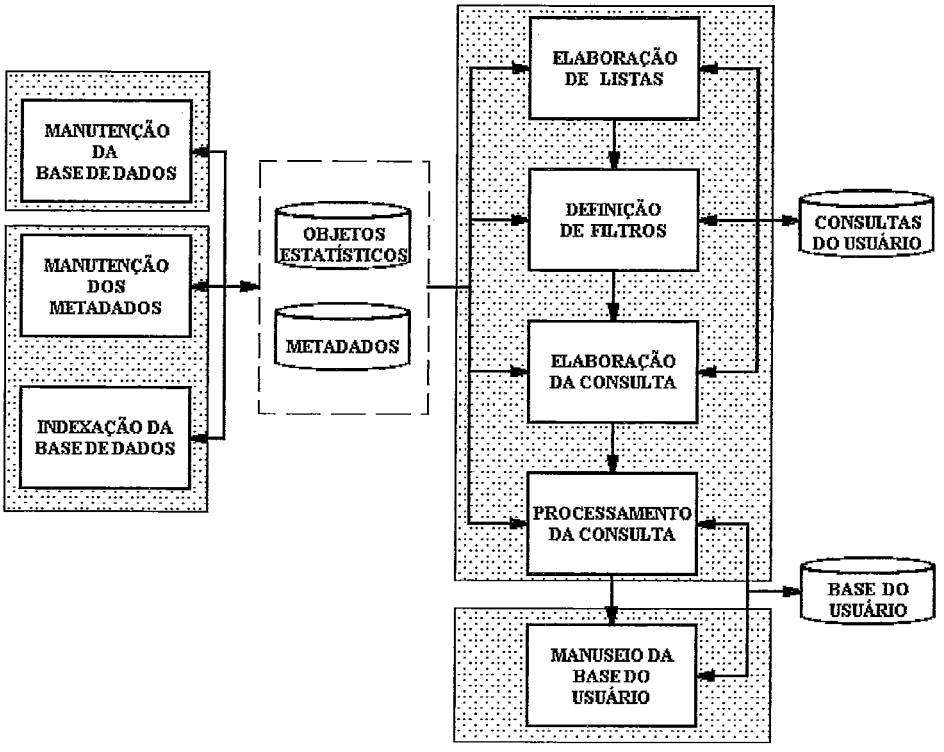


Figura VI-2 - Arquitetura do ambiente

A base de consultas do usuário é utilizada para o registro de listas contendo os critérios para a seleção de informações. O registro destas listas em objetos persistentes permite sua re-utilização em diversas consultas ou sessões. Numa sessão de consultas com a participação de diversos usuários, a base de consultas é visível para todos mas é atualizável por apenas um dentre eles. Existe um critério para definir o usuário que, num determinado momento, está com o papel de operador do sistema e, por tanto, com o privilégio de poder atualizar essa base. Estas operações são feitas sob a observação dos demais participantes da sessão que podem opinar através de um sistema de conversação.

Uma consulta é elaborada após o usuário especificar nas listas os critérios para a execução de um corte espacial, temporal na base e para a seleção dos objetos referentes a determinadas categorias de informações. O usuário, também, pode definir filtros para restringir o conjunto das informações selecionadas. Estes filtros utilizam uma definição algébrica e podem ser encadeados com operadores lógicos.

No ambiente proposto, a elaboração de uma consulta consiste na tradução dos critérios utilizados para as definições das listas de consulta em relações de objetos com seus atributos, expressos na linguagem de programação do SGBDOO. Neste programa são acrescentadas as sentenças de programação relativas a execução dos filtros. O processamento da consulta irá extrair valores das bases de objetos estatísticos e metadados para formar a do usuário. É criada uma nova classe no esquema, vinculada ao usuário, onde o tipo do objeto é composto pela relação dos atributos selecionados, acrescidos dos atributos de decodificação dos códigos e das referências à base de dados.

No módulo de manuseio da base do usuário é permitido o uso da linguagem de consulta do SGBDOO para a análise dos objetos. O usuário, também, pode exibir os objetos na tela e alterar a escala dos valores e pode, ainda, gravar arquivos convencionais para integração com outras ferramentas, como as de análise estatística.

6.3. MODELO LÓGICO DE DADOS

O modelo de dados adotado para as aplicações do ambiente do BDE, do tipo orientado a objetos, conforme diagrama da figura VI-3 é constituído por três grupos de informações: a base de dados estatísticos, propriamente dita; a base do usuário, formada por valores extraídos da base de dados principal; e a base de consultas, com

as listas de opções do usuário. Na figura as classes de objetos pertencentes aos metadados estão marcadas pela área hachurada.

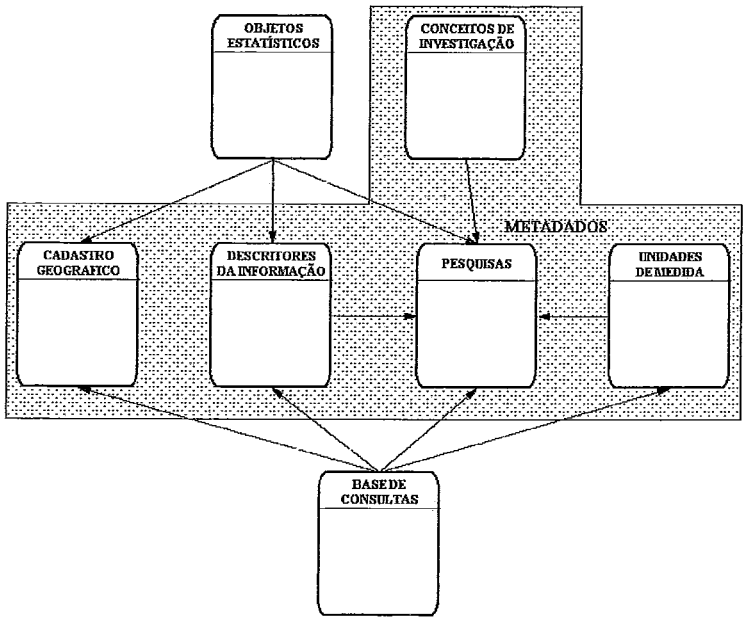


Figura VI-3 - Esquema de classes do BDE

A definição do modelo lógico de dados é feita conforme o diagrama de entidades relacionamentos (E-R) exibido na figura VI-4 que destaca os relacionamentos dos objetos estatísticos com os metadados.

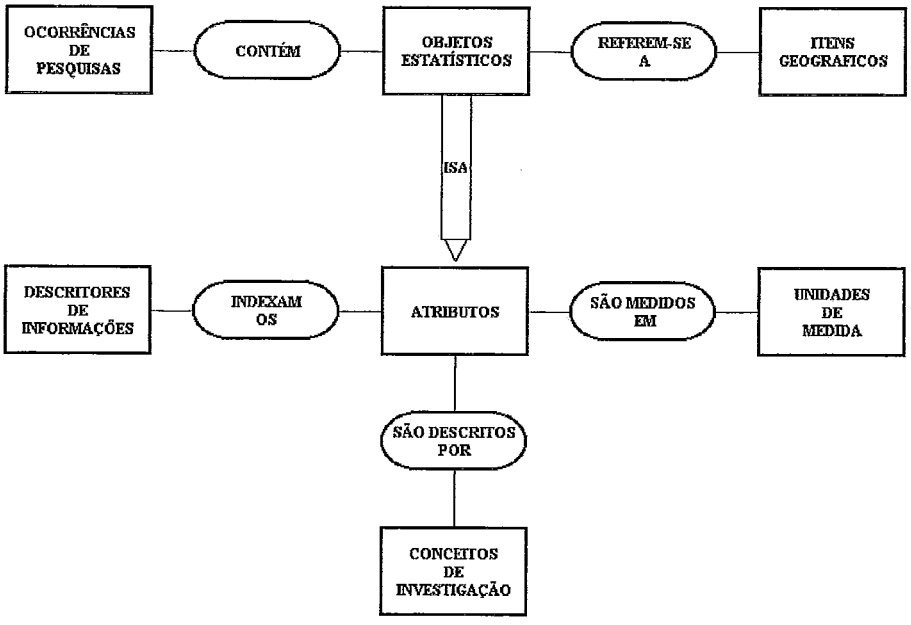


Figura VI-4 - Modelo de entidades-relacionamentos da base

Na figura VI-5 é apresentado o diagrama E-R do dicionário da pesquisa. O destaque nesta figura é feito para os relacionamentos do dicionário da pesquisa, definido como propriedade da classe *Pesquisas,* com os demais metadados e com os objetos estatísticos.

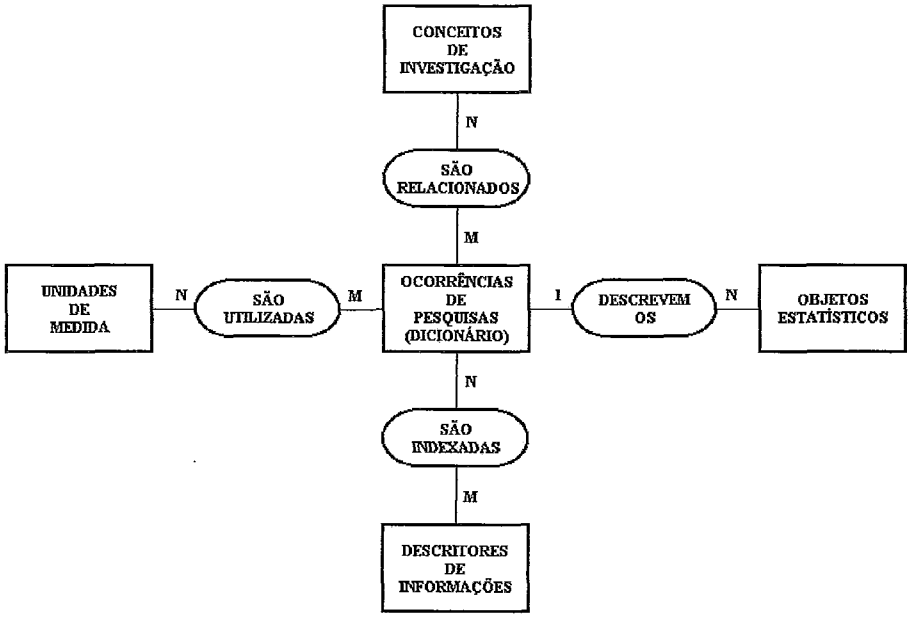


Figura VI-5 - Modelo E-R do dicionário da pesquisa

6.3.1. Base de Dados Estatísticos

A base de dados estatísticos é constituída, logicamente, por dois conjuntos de objetos. Os objetos estatísticos contém os valores produzidos pelas pesquisas estatísticas e são instanciados como objetos da classe *Dados* (figura VI-6). Os metadados descrevem e indexam os objetos estatísticos através das instância armazenadas nas classes *Descritor de Informação, Conceitos de investigação, Unidade de medida, Pesquisa e Cadastro Geográfico* (figuras VI-7 e VI-8).

A modelagem de hierarquias nesta base é feita como num sistema convencional utilizando atributos referenciais ou atributos de coleções de referências. Esta técnica, apesar de exigir a programação de um mecanismo para manusear a hierarquia e para manter sua integridade, é de fácil implementação e não exige o uso de recursos especiais de um determinado SGBDOO. Com o uso do mesmo tipo, auto-referenciado, é possível ser definida uma hierarquia de qualquer profundidade sem a utilização do recurso de herança.

Entretanto, a orientação a objetos permite uma outra técnica de manuseio de hierarquias utilizando o mecanismo de herança, com uma subtipagem por especialização. Com esta modelagem seria utilizado o próprio mecanismo de controle de classes do sistema para manter a hierarquia. Assim, a modelagem para um sistema de classificações poderia ser feita com a definição de uma classe para cada grupo de objetos que herdam a definição da superclasse. O processo de definição de novas classes continuaria até o nível mais baixo da hierarquia onde estariam as instâncias dos objetos descritivos das categorias da classificação. Os atributos descritivos da raiz e dos níveis intermediários seriam implementados por instâncias únicas de suas subclasses.

Esta alternativa é interessante mas exige um recurso não especificado no paradigma dos sistemas orientados a objetos. Na versão 4.2 do O_2 foram implementados os comandos para acesso ao esquema, com o nome de comandos de acesso ao meta-esquema.

6.3.1.1. Objetos estatísticos

Os objetos estatísticos estão definidos como uma hierarquia de objetos da classe *Dados*, conforme mostrado na figura VI-6. Na superclasse são definidos os atributos comuns a todas as pesquisas estatísticas para a identificação das informações, a saber:

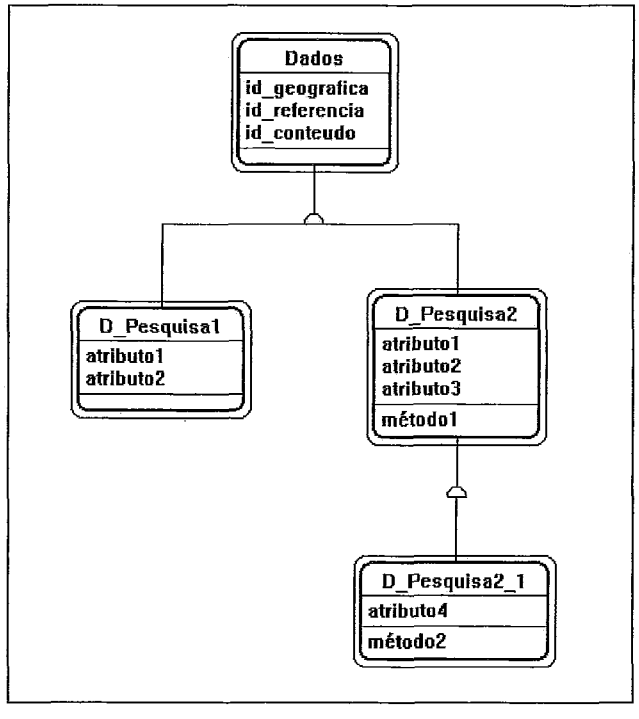


Figura VI-6 - Diagrama da classe *Dados*

. identificação do espaço geográfico, onde o fato é relacionado a um item da estrutura político-administrativa formal ou região metropolitana;

. caracterização da época da ocorrência, onde o fato é relacionado com uma data ou período de ocorrência;

. categoria do fato dentro de um sistema de classificação, onde o fato é associado a um elemento de qualificação do assunto.

Para cada pesquisa é definida uma subclasse da classe *Dados* com os atributos específicos da pesquisa. A hierarquia continua com as revisões no modelo de investigação das ocorrências das pesquisas.

6.3.1.2. Metadados

Os metadados das informações estatísticas estão instanciados nas cinco classes apresentadas nos diagramas das figuras VI.7 e VI.8. Utilizando atributos de referência aos metadados, os objetos estatísticos compõem os atributos de identificação geral, existentes na superclasse *Dados.*

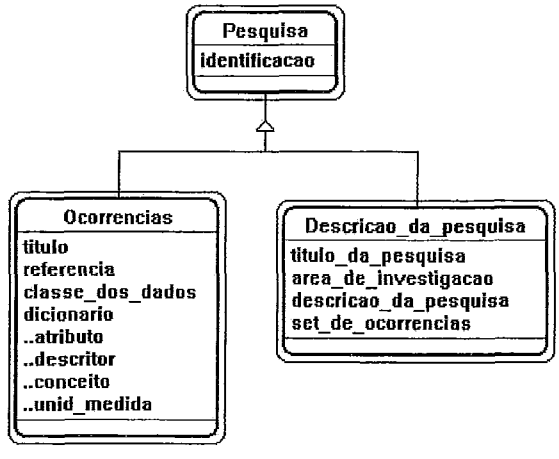


Figura VI-7 - Superclasse *Pesquisa*

A superclasse *Pesquisa,* figura VI.7, é destinada ao registro das características das pesquisas. A descrição da pesquisa é feita de forma textual numa instância da subclasse *Descricao_da_pesquisa.* Na subclasse *Ocorrencias* é criada uma instância para cada ocorrência da pesquisa com o detalhamento e a indexação dos objetos estatísticos. Este detalhamento, uma lista das propriedades do objeto estatístico, é referido no projeto como dicionário da pesquisa. Neste dicionário fica registrada a indexação da

base, composta pelas referências dos atributos da classe de objetos estatísticos da pesquisa aos metadados. As classes *Descritor* e *Conceito* que são utilizadas para a indexação mantém referências inversas aos atributos relacionados no dicionário.

A classe *Cadastro_geográfico* tem como finalidade o registro dos itens da estrutura político-administrativa, inclusive regiões metropolitanas, com suas hierarquias de itens geográficos. Todos os itens geográficos, desde o nível de município são instanciados como objetos desta classe e as hierarquias são descritas através de atributos de coleção de itens subordinados e de referência a cada item superior da hierarquia.

A classe *Descritor das informações* tem como finalidade o registro dos sistemas de classificação. As categorias e seus sistemas de classificação são instanciadas como objetos de hierarquias, inclusive múltiplas, evitando a redundância de descritores no banco de dados.

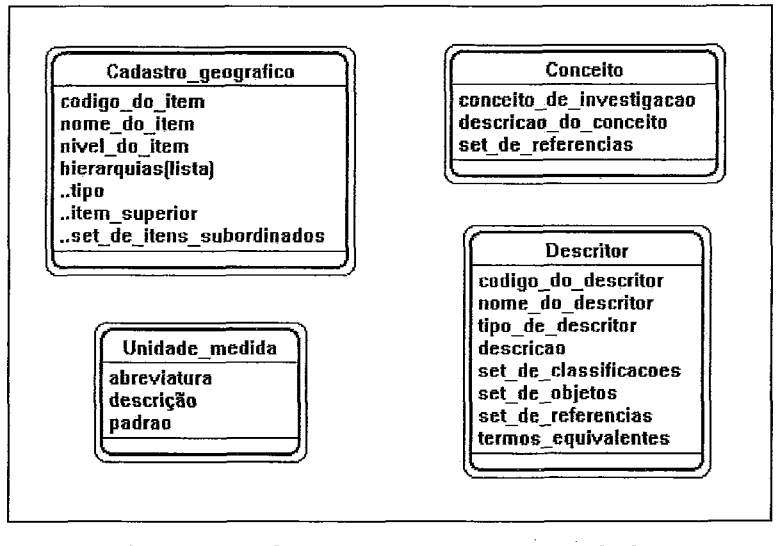


Figura VI-8 - Classes dos metadados

A classe *Conceito* tem como finalidade o registrar dos conceitos de investigação estatística, detalhando de forma conceitual a natureza do item pesquisado.

A classe *Unidades de medida* tem com a finalidade manter um sistema de registro da escala de grandezas empregadas de forma a permitir ao usuário ajustar os valores às unidades de medida desejadas.

6.3.2. Base do Usuário

A base do usuário é definida dinamicamente durante uma consulta. O usuário especifica o tipo de informação que deverá constar dos objetos desta base selecionando-os entre os atributos dos objetos estatísticos e dos metadados. O usuário pode es-

pecificar a inclusão de códigos ou de suas descrições no objeto da nova base. A idéia é formar uma base de dados que seja transportável para outros sistemas, ou equipamentos, onde o usuário possa continuar o seu trabalho de pesquisa fazendo, por exemplo, uma análise estatística. Assim, a base do usuário, por princípio, não deverá contar com atributos referenciais nem com códigos que dificultem ou mesmo impeçam a continuidade do trabalho. Numa primeira abordagem, pode-se comparar a base do usuário com a estrutura de um arquivo plano, onde todos os atributos são dispostos numa mesma linha ou registro.

A efetividade dessa função é obtida pela capacidade de manuseio do esquema pelo O_2. O O_2 dispõe de uma função que executa todos os comandos de sua linguagem de manuseio de esquemas através de sua linguagem de programação, a O_2C.

6.3.3. Consultas do Usuário

A base de consultas do usuário, figura VI-9, corresponde ao conjunto de instâncias da superclasse *Usuario* onde estão registrados os usuários e a especificação dos critérios para consultas e extração de dados. Cada usuário pode instanciar quantos objetos de definição desejar objetivando a composição de consultas.

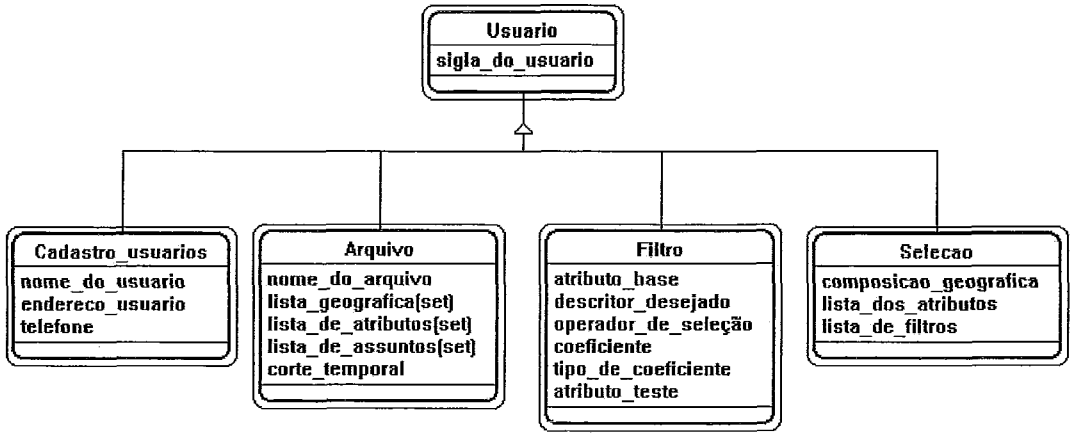


Figura VI-9 - Diagrama da classe *Usuario*

A classe *Usuario*, a exemplo da classe *Dados*, contém somente o atributo de identificação dos objetos. Nesta classe, a identificação utilizada é a sigla de *logon* do usuário no sistema, ao estabelecer uma sessão de consultas. As subclasses herdam este atributo, especializando o conteúdo pela finalidade do objeto. Com a classe *Cadastro de usuário* é feito o registro dos dados do usuário, em forma de cadastro. Na classe *Arquivo* são descritas as listas elaboradas pelo usuário para a execução de consultas, compreendendo uma lista de itens geográficos, uma de atributos da base, uma

de assuntos (pesquisas e descritores e/ou conceitos) e uma definição do corte temporal. A classe *Filtro* é utilizada para a descrição dos critérios de restrição adotados numa seleção de informações e a classe *Seleção* é utilizada para o registro da lista de atributos utilizada numa consulta, pelo módulo de definição do programa de consulta extração de dados.

6.4. INTERFACE DO USUÁRIO

A interface homem-máquina foi construída de acordo com os padrões de interfaces gráficas. Todos os comandos são emitidos por botões de opções ou por botões de seleção de funções. A preferência do usuário por assuntos é efetuada por seleção em listas compostas com textos editados. A interface não implementa comandos textuais nem a seleção é feita de forma dirigida.

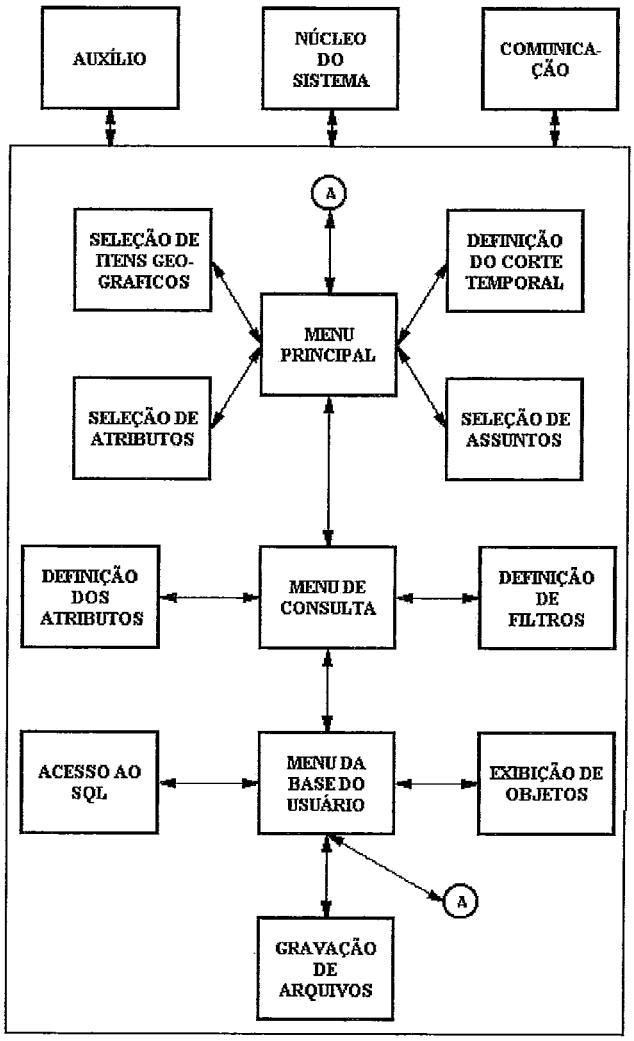


Figura VI-10 - Grafo de transição das telas da aplicação BDE

Quando a seleção é feita sobre hierarquias de assuntos, a interface permite ao usuário definir os níveis de composição das listas permitindo a exibição em conjunto dos seus componentes, "pais" e "filhos", ou de apenas um nível. Também é permitido ao usuário definir as listas de seleção apenas com aqueles itens nos quais os "pais" foram selecionados.

A interface é baseada num conceito relaxado da metáfora WYSIWIS (*What You See Is What I See*). Todas as telas manuseadas pela interface do usuário são mapeadas em posições definidas da tela do equipamento porém, os participantes não operadores somente recebem uma imagem atualizada depois que o usuário operador pressiona algum botão de processamento. Outro relaxamento é a liberdade de rolar a barra de uma lista de forma individual, sem a participação dos demais usuários, como forma de melhorar a compreensão individual em determinado assunto.

Em todas as telas de comunicação são definidos botões para ativar um auxílio dentro do contexto ou para utilizar a janela de comunicação dos participantes de uma sessão.

A figura VI-10 apresenta o grafo de transições da interface da aplicação BDE. Durante uma sessão o usuário ficará interagindo com um dos três menus principais da onde pode selecionar uma função ou ativar as telas de auxílio, de comunicação ou de controle do sistema. O modelo das telas utilizadas encontra-se no Anexo B, ítem 4.2.

6.5. PRINCIPAIS FUNÇÕES

6.5.1. Núcleo do Sistema - *NS*

O núcleo do sistema (NS) consiste na função de controle de uma sessão e de suas comunicações. O NS viabiliza o processamento cooperativo isolando os participantes do usuário operador da sessão, ou seja, aquele com o poder de executar as funções consulta, extração de dados e manuseio da base do usuário. O NS é implementado pelo núcleo propriamente dito e por dois módulos: controle e comunicações.

O módulo de controle registra as sessões e seus participantes. O primeiro usuário de uma sessão assume o papel de operador e, durante uma sessão, todos os registros serão feitos em seu nome, independente de haver transferido a operação para outro participante. As autorizações para um novo usuário participar de uma ses-

são ou para operar o sistema são controladas por este módulo que, através de uma interface com os demais módulos, autoriza a execução de um comando ou dissemina uma apresentação para as telas dos equipamentos dos demais participantes.

O módulo de comunicações é o encarregado de gerenciar a troca de informações entre os participantes. A qualquer momento, um participante pode dirigir aos demais uma mensagem. A granularidade utilizada para o *WYSIWIS* é a do texto enviado, que somente é disseminado após o botão de enviar ser pressionado. O sistema de comunicações também é utilizado pelo módulo de controle para, através de mensagens padronizadas, consultar o usuário operador sobre o acesso de novos participantes e sobre a passagem do direito à operação.

6.5.2. Administração da Base de Dados

A função de administração da base de dados contém as ferramentas com as quais o administrador do sistema pode atualizar o esquema da base, carregar novas ocorrências de pesquisas, manter o sistema de auxílio e intervir em consultas de usuários. Esta função com as subfunções decorrentes de seu detalhamento constituem uma aplicação de uso restrito no sistema, chamada de aplicação Apoio, que é implementada pelos seguintes módulos: módulo de atualização do esquema, módulo de carga de pesquisas e módulo de auxílio.

No módulo de **atualização do esquema** é feita a definição de uma nova classe de dados com a manutenção dos padrões utilizados para a descrição de atributos e métodos. O módulo verifica, também, a possível existência de outra classe com o mesmo nome antes de incluí-la na hierarquia da superclasse de dados. Para as operações de alteração ou exclusão de classes ou propriedades do esquema, o administrador pode utilizar a linguagem de definição de dados do SGBDOO.

Como padrão observado temos a diferenciação dos métodos herdados do objeto raiz do SGBDOO daqueles implementados como atributos virtuais numa classe de objetos de uma pesquisa. Outro padrão do sistema consiste no uso de uma nomenclatura controlada para subtipagem de ocorrências de uma pesquisa.

Os módulos de **carga de pesquisas** facilitam ao administrador do sistema a carga de novos dados das pesquisas contínuas utilizando programas utilitários que permitem rapidez de execução. Com a ajuda de um menu para seleção da pesquisa e

de sua ocorrência, é ativado um dos módulos de carga o qual faz a seleção entre as classes do sistema e cria as novas instâncias de dados.

Com o **módulo de auxílio** o administrador do sistema faz a manutenção do sistema de auxílio ao usuário, contando com um sistema de edição dos objetos de auxílio e dos textos utilizados. As telas de auxílio são montadas segundo a estrutura das aplicações visando o apoio dentro do contexto em que são chamadas.

6.5.3. Administração dos Metadados

A função de administração dos metadados trata da manutenção dos metadados e do sistema de indexação das variáveis da base de dados A manutenção dos metadados abrange os objetos que descrevem as informações estatísticas ou seja abrange os descritores das informações, os conceitos de investigação, as unidades de medida e a descrição das pesquisas. Faz parte também desta função o sistema de indexação destinado a associar as propriedades dos objetos estatísticos com as suas descrições. Esta função está implementada pelos módulos *manutenção dos descritores*, *manutenção dos conceitos*, *manutenção das unidades de medida*, *manutenção das pesquisas* e com o *indexação*.

O módulo de **manutenção dos descritores** faz o registro ou correção dos objetos instanciados na classe *Descritores*. Permite a formação ou alteração das classificações através do relacionamento do descritor com seus "pais" e com seus "filhos", ou seja, com os descritores nos quais seja uma categoria da classificação e com as categorias para as quais ele próprio seja uma classificação.

O módulo de **manutenção dos conceitos** permite ao administrador do sistema corrigir e complementar os conceitos de investigação utilizados no processo de indexação e armazenados nos objetos da classe *Conceito*. O módulo de **manutenção das unidades de medida** é semelhante ao de conceitos porém voltado aos objetos da classe de *Unidades de medida*.

Com o módulo de **manutenção das pesquisas** o administrador do sistema pode definir, alterar ou excluir novas pesquisas e suas ocorrências, utilizando os objetos da superclasse *Pesquisas*. Quando uma ocorrência de pesquisa é definida, o programa constrói a base do dicionário de dados utilizado no processo de indexação, usando para tanto a definição do meta-esquema das propriedades dos objetos estatísti-

cos. Quando é comandada uma exclusão de pesquisa ou ocorrência pela aplicação Apoio, este módulo retira todas referências existentes aos objetos excluídos.

O módulo da **indexação** fará a associação dos atributos de uma classe de objetos estatísticos utilizada para o registro de determinada ocorrência de uma pesquisa com um descritor de informações, com um conceito de investigação e com uma unidade de medida.

6.5.4. Consulta e extração de dados

A função de consulta e extração de dados é a encarregada da montagem de listas com as definições desejadas para a busca de dados, da tradução destas listas para a relação de atributos dos objetos estatísticos e para a definição das restrições de seleção de informações. Esta função é implementada pelos módulos de *elaboração de lista de ítens geográficos,* de *definição de corte temporal,* de *definição de atributos,* de *seleção assuntos*, de *elaboração uma consulta*, de *definição de filtros* e do *processamento da consulta.*

O módulo de **elaboração da lista de ítens geográficos** apresenta os itens geográficos para o usuário selecionar aqueles que irão compor uma lista. Na montagem da lista de seleção o usuário poderá optar entre utilizar a hierarquia formal político-administrativa ou a estrutura de regiões metropolitanas. Os itens podem ser exibidos em ordem alfabética ou ordenados pelo código de identificação. A qualquer momento o usuário pode registrar ou atualizar a lista num objeto da classe *Arquivo*, superclasse *Usuario.*

O módulo de **definição de corte temporal** apresenta ao usuário a alternativa de escolher os valores da base entre os dados mais recentes, mais antigos, todos os dados ou com os dados de determinado período. A opção temporal pode ser registrada num objeto já instanciado na classe *Arquivo* ou pode originar uma nova instância desta classe para o registro da opção.

O módulo de **definição de atributos** permite a seleção dos atributos entre os atributos de uma pesquisa ou de uma determinada ocorrência de pesquisa. A lista formada por duplas "ocorrência de pesquisa - atributo" pode ser registrada num novo objeto ou em algum já existente como instância da classe *Arquivo.*

O módulo de **seleção de assuntos** permite definir o interesse em valores dos objetos estatísticos através dos descritores da informação. Na montagem da lista de seleção de assuntos, o usuário pode especificar a opção de exibir as classificações com ou sem as categorias subordinadas. Também por opção do usuário, o sistema pode exibir os conceitos de investigação utilizados pelos descritores para alguma pesquisa. Como nas demais listas, esta também pode ser registrada num objeto existente ou num novo objeto da classe *Arquivo*.

Com o módulo de **elaboração de consulta** o usuário pode selecionar as listas que irão compor a busca de informações. O processo é feito com a conversão das listas de assuntos numa relação de propriedades de classes de objetos estatísticos. As propriedades obtidas pela conversão dos assuntos e acrescidas daquelas provenientes de listas de seleção atributos são submetidas à definição do corte temporal com a ajuda dos dicionários das pesquisas. A relação final dos atributos é guardada na classe *Selecao* e é utilizada para criar uma classe de objetos específica para o usuário.

Com o módulo de **definição de filtros** é feita a especificação das restrições para a seleção dos atributos dos objetos estatísticos registrados na lista de consulta instanciada como objeto da classe *Selecao.* A definição do filtro é feita através de um teste de comparação do atributo contra um valor. A comparação também pode ser efetuada contra outro atributo existente na lista da classe *Selecao*, modificado ou não por uma constante numérica. Os filtros assim definidos podem ser combinados em sequências lógicas do tipo "ou" e os estes novos conjuntos de filtros são então executados como seqüências de operações "e".

O módulo de **processamento da consulta** inclui no esquema uma classe para a extração dos dados da base e efetua a codificação e execução de um programa na linguagem de programação do SGBDOO. Este programa é destinado a instanciar os objetos do usuário com os valores das bases de objetos estatísticos e de metadados, conforme a definição de atributos existente no objeto da classe *Selecao*. Após a criação de uma instância o programa faz os testes definidos pelos filtros, mantendo apenas aquelas cujo resultado das sentenças seja verdade.

6.5.5. Base do Usuário

A função base do usuário permite o manuseio do conjunto de informações extraído do BDE e armazenadas em objetos desta base. Esta função compreende os módulos *linguagem de consulta*, *exibição dos dados* e de *intercâmbio de arquivos*. O

módulo linguagem de consulta é uma interface para a linguagem de consulta do SGBDOO, com o objetivo de facilitar o manuseio de uma SQL dentro do ambiente da aplicação. Com o módulo de exibição dos dados o usuário pode exibir os dados de sua base e alterar a escala de exibição dos valores. O módulo de intercâmbio de arquivos permite a gravação dos dados da base do usuário em arquivos convencionais destinados ao intercâmbio dos dados com outros *softwares*.

7. UM PROTÓTIPO DE BDEOO

As especificações do projeto foram testadas através da implementação de um protótipo. A plataforma utilizada foi a de *workstations SUN/SPARC* operando com o sistema operacional *UNIX-V*, versão 4.2, gerenciador gráfico *MOTIF* e com o *XWindows*. O protótipo foi desenvolvido com a linguagem de programação do SGBDOO O_2, a O_2C, versão 4.2. A versão utilizada do O_2, apesar de ser do tipo cliente-servidor, era restrita a um cliente por servidor o que limitou o protótipo a somente manusear operações mono-usuárias.

A arquitetura do projeto piloto foi baseada no projeto do ambiente para o BDEOO. Entretanto, algumas limitações na plataforma utilizada restringiram a implementação e não permitiram testar as técnicas de CSCW projetadas para o sistema. Assim, o projeto piloto foi dividido em três grandes aplicações, a saber:

. *BDE* - aplicação para manuseio dos objetos estatísticos propriamente dito, voltada ao usuário final do projeto;

. *METADADO* - aplicação destinada à manutenção das pesquisas estatísticas e de seus metadados no ambiente do projeto, voltada ao administrador do banco de dados;

. *APOIO* - destinada a implementar recursos gerais para as duas aplicações anteriores, como a manutenção dos textos de ajuda utilizados, programas utilitários para a carga de dados, etc.

No Anexo B estão detalhadas as aplicações implementadas, descrevendo-se para cada uma a estrutura de módulos utilizada, os tipos de objetos manuseados e as telas construídas para a interface do usuário.

No detalhamento dos módulos implementados, foram considerados como programas as principais funções da aplicação. Neles são descritos os objetivos, os dados, os parâmetros para processamento, o algorítimo de processamento e as saídas produzidas (telas). Ao final de cada programa são relacionadas as funções (rotinas de apoio à programação) utilizadas com uma descrição sucinta de sua finalidade. Foram acres-

centadas na descrição dos módulos da aplicação *APOIO* as funções de apoio à programação desenvolvidas com objetivo de expandir os recursos de programação do O_2C no manuseio de tipos ou para permitir a reutilização de rotinas por mais de um programa.

Os programas que foram desdobrados em diversas funções foram descritos logo após o programa principal com a utilização da mesma estrutura de detalhamento dos programas ou módulos principais.

7.1. PROJETO PILOTO

O banco de dados do projeto piloto do BDEOO foi carregado com todos os componentes da estrutura político-administrativa brasileira existentes na época do Censo Demográfico de 1991, além das 10 regiões metropolitanas. Foram também carregados os dados e metadados de duas pesquisas estatísticas do IBGE: a *PAM - Pesquisa Agrícola Municipal* e as *estimativas populacionais produzidas para 1990.*

A **PAM** é uma pesquisa anual realizada desde 1937 em todos os municípios do Brasil. Investiga as principais culturas da lavoura permanente e da lavoura temporária nos conceitos de área plantada, produção obtida, rendimento da cultura e preço médio obtido pelo produtor. A partir de 1988 a PAM passou também a investigar a área colhida . Na base do projeto piloto foram carregados os dados relativos às ocorrências da PAM na região Norte entre 1986 e 1990. Os dados municipais foram agregados dentro da estrutura político-administrativa formal e para a região metropolitana de Belém.

As **estimativas populacionais** são realizadas nos anos em que não são levantados censos demográficos visando estabelecer a expectativa de população nos municípios brasileiros. Das estimativas feitas em 1990, foi carregada a variável que estima a população total para todos os municípios brasileiros. Os dados municipais foram consolidados tanto na estrutura político-administrativa formal como para as 10 regiões metropolitanas.

7.2. CLASSES DE OBJETOS

Conforme o modelo lógico de dados elaborado para o projeto e apresentado no capítulo 6, foram implementadas na aplicação BDE duas superclasses de objetos: uma para a classe *Dados*, destinada ao manuseio dos objetos estatísticos propriamente

ditos e outra para classe *Usuario*, destinada ao registro do usuário e de suas listas de trabalho. No teste piloto foi definido um "named set" por superclasse, destinado à persistência de suas instâncias e as das classes subordinadas. Para o O_2, um "named set" é um tipo de conjunto onde é possível a persistência das instâncias de uma classe.

A implementação das aplicações METADADOS e APOIO também foi feita de acordo com o projetado e foram definidos "named sets" para dar persistência aos atributos descritivos das pesquisas e para as demais classes de metadados da informação estatística.

A superclasse ***Dados*** foi implementada com as subclasses *D_PAM, D_PAM_1* e *D_POPULACAO* destinadas, respectivamente, aos dados da PAM relativos aos anos de 1986 e 1987, aos relativos a 1988 à 1990 e a estimativa populacional de 1990.

As instâncias da superclasse ***Usuario***, implementada com as subclasses *Cadastro_usuarios, Arquivo, Selecao* e *Filtro,* foram armazenadas no *named set ARQUIVO.*

A superclasse ***Pesquisa*** foi implementada com as subclasses *Descricao_da_pesquisa* e *Ocorrencias*, sendo suas instâncias mantidas no *named set PESQUISA.*

As classes ***Descritor***, ***Conceito, Unidade_medida, Cadastro_geografico***, correspondentes aos metadados tiveram sua persistência feita com os *named set DESCRITOR, CONCEITOS, UNIDADES, CADGEO,* respectivamente, e a classe *Ajuda,* de textos de auxílio, teve seus objetos mantidos no *named set AJUDA.*

7.3. APLICAÇÃO BDE

Com a limitação do SGBDOO O_2 para aplicações mono-usuárias, não foram implementadas as funções de processamento cooperativo de *tutor do sistema* e de *consulta cooperativa multidisciplinar.* A aplicação foi desenvolvida com vistas a testar a funcionalidade do metadado na busca de informações e no uso da interface gráfica para consultas.

7.3.1. Estrutura Implementada

A aplicação BDE foi implementada utilizando o que é denominado de transações e funções no O_2C. Além dos módulos previstos no projeto, as características do SGBDOO O_2 e os problemas existentes no ambiente operacional demandaram algumas modificações, conforme apresentado na figura VII.1. Foram acrescentados os módulos de *inicialização* e *término* para utilizar a característica do software de executar automaticamente estas funções e o módulo de *restart* foi acrescentado para contornar o problema encontrado com a gestão de memória.

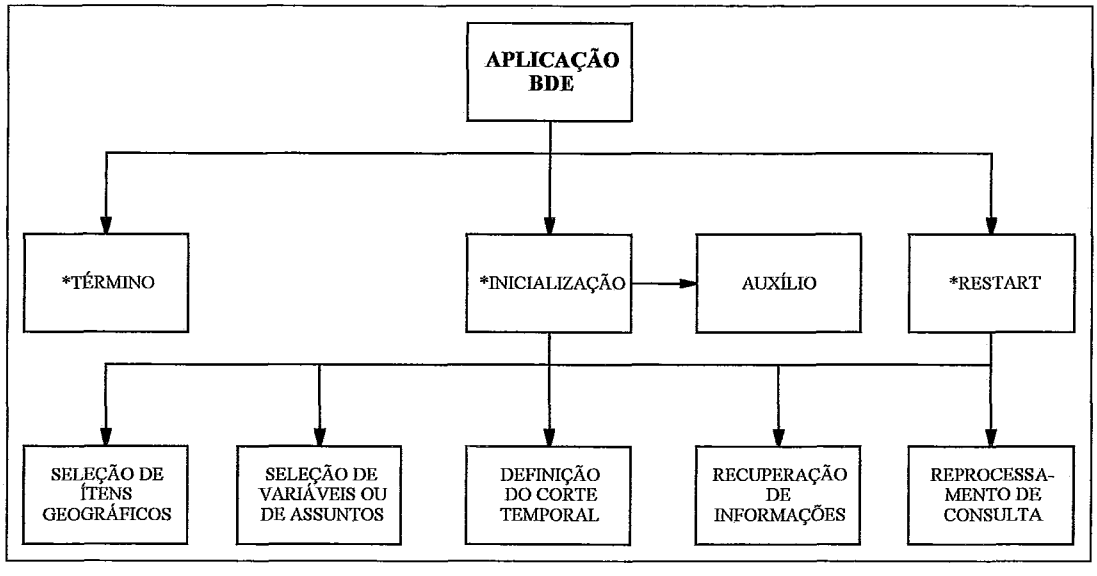


Figura VII-1 - Diagrama da aplicação BDE

7.3.2. Detalhamento dos Módulos

Todos os programas da aplicação BDE foram implementados com uma interface comum para transferência troca informações e parâmetros. O padrão consiste na transferência do atributo de identificação do objeto da base de consultas em uso e do *pid* (presentation identifier) da última tela empilhada pelo programa chamador. Ao término da execução, os programas devolvem a identificação do objeto corrente. Caso a execução seja cancelada através do botão *CANCELA*, o programa chamado retorna um objeto *nil*. Devido a impossibilidade de devolver um parâmetro para o programa chamador em O_2C, os programas foram codificados como funções, única forma capaz de aceitar este padrão.

O programa de **inicialização** é executado automaticamente no início da aplicação. Com ele é feito o controle da execução dos programas da aplicação até que um *commit* seja emitido e o programa *restart* assuma o controle do processamento.

Os programas de **seleção de itens geográficos, de variáveis ou assuntos e de definição corte temporal** editam as listas para a definição dos critérios de consulta ao banco de dados.

Com os módulos do programa de **recuperação de informações** é feita a tradução das listas de critérios para sentenças da linguagem de programação do SGBDOO, são definidos os filtros de seleção, é criada a classe para a base do usuário e é efetuada a extração dos dados.

O programa de **reprocessamento de consultas** permite ao usuário reprocessar a última consulta efetuada remontando sua base, enquanto o programa **restart** cuida das interrupções causadas pelos commits emitidos durante o processamento da preparação e consulta à base de dados.

7.4. APLICAÇÃO METADADO

A aplicação METADADO compreende as ferramentas de manutenção dos metadados do BDE distribuídas em três grandes funções: manutenção de pesquisas, manutenção de descritores e indexação da base. Com as duas primeiras são mantidos os metadados necessários ao processo de indexação que é executado pela terceira destas funções.

7.4.1. Estrutura Implementada

Os programas que implementam as funções da aplicação METADADO são independentes entre si e executáveis através de menu criado automaticamente pelo O_2 no momento da definição de uma aplicação. A estrutura utilizada para implementar a aplicação metadado pode ser visualizada nos diagramas de módulos constantes da figura VII-2.

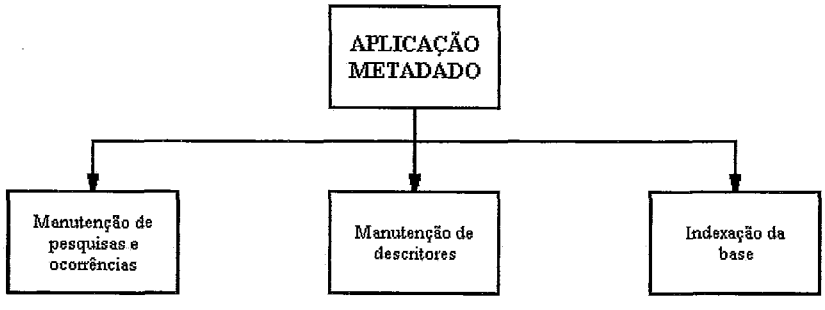


Figura VII-2 - Diagrama da aplicação METADADO

O programa **manutenção de pesquisa** tem como finalidade a manutenção de características das pesquisas e de suas ocorrências. O programa **manutenção dos descritores** destina-se à manutenção dos descritores da informação e dos sistemas de classificação. O programa **indexação da base** tem por objetivo associar os metadados com os objetos da base de dados.

7.5. APLICAÇÃO APOIO

Todos os programas da aplicação APOIO tem como característica serem executáveis de forma independente como programas utilitários do sistema. Somente o programa de manutenção dos textos de auxílio opera com menus para o encadeamento das telas de edição de textos. A aplicação APOIO dispõe dos programas de edição dos textos de auxílio, de carga de dados e para carga dos metadados.

7.6. UTILIZAÇÃO DE UM SGBDOO (O_2)

A implementação do projeto piloto foi realizada com o Sistema Gerenciador de Banco de Dados Orientado a Objetos - O_2. Este SGBD foi inicialmente desenvolvido como um protótipo e depois foi transformado em produto comercial pelo GIP Altair em Le Chesnay, França. O O_2 segue os ítens considerados obrigatórios no paradigma da orientação objeto, conforme o "Manifesto" [ATKI89]. Implementa objetos complexos, identidade para os objetos, encapsulamento de propriedades, tipos e classes de objetos, hierarquia de classes, montagem a posterior de objetos (late binding), persistência de dados, listas, conjuntos, conjuntos únicos e uma linguagem de consulta. O O_2 implementa também alguns dos ítens considerados desejáveis para SGBDOO que são a herança múltipla e um mecanismo de transações.

O O_2 foi desenvolvido em C a partir do WiSS (Wisconsin Storage System) [CHOU86], um gerenciador de objetos desenvolvido por H.Chou, D.DeWitt, R.Katz e A.Klug, em 1986 [BANC89].

O O_2 também pode ser utilizado através de uma linguagem de programação estendida do C, a O_2C, e através de interface com o C++. O mecanismo de montagem a posterior dos objetos é complementado com o armazenamento no banco de dados dos procedimentos e métodos programados. Assim, durante a execução de um programa ou de uma consulta, os procedimentos e métodos demandados são dinamicamente carregados de acordo com o tipo de objeto manuseado.

A linguagem de consulta do O_2 é do tipo SQL, provida de filtros para a seleção de dados. A O_2SQL dispõe de um conjunto de operadores para construir ou decompor listas, conjuntos e tuplas de dados. O resultado de uma consulta pode ser uma lista, um conjunto ou uma tupla de qualquer tipo de objeto. A linguagem de consulta também pode ser utilizada dentro de programas escritos em O_2C.

O mecanismo de transações do O_2 bloqueia as páginas lidas no servidor. As páginas que forem alteradas não são atualizadas no servidor até que um comando commit seja emitido para encerrar a transação. Qualquer problema ocorrido durante o período de processamento da transação provoca o seu cancelamento e o abandono das páginas alteradas que não são gravadas.

Durante a implementação do teste foram verificados problemas nos seguintes componentes do O_2: ferramentas gráficas, que não satisfizeram plenamente os requisitos de interfaces do usuário; manuseio de propriedades de subclasses dentro de um *named set* definido na superclasse; acesso à objetos, que não dispõe de acesso por índices; mecanismo de transações; e com o uso do alfabeto português.

7.6.1. Ferramentas Gráficas

O O_2 dispõe de ferramentas para a construção de interfaces gráficos [BORR92]. Permite a manipulação interativa de objetos do banco de dados através de funções que exibem um objeto de forma gráfica permitindo modificá-lo, inclusive através de operações de "cut/copy/paste", e atualizar o novo estado deste objeto no banco de dados.

Para a construção das interfaces gráficas desenhadas pelo usuário, o O_2 fornece um conjunto de ferramentas chamadas de O_2Kit. Estas ferramentas são recursos elaborados sob o O_2Look e permitem a rápida construção de janelas simples ou complexas.

Apesar do bom resultado visual da interface gráfica do O_2, a funcionalidade do O_2Look/O_2Kit não é completa. Faltam recursos do tipo "pop-up" e botões que permitam a execução de opções dentro do contexto da janela. O O_2Look implementa apenas dois botões de processamento em cada janela, botões *OK/Cancela*, *SIM/NÃO* ou *SAVE/ERASE* que liberam o estado de "wait" de um programa. Quando são desejadas diversas opções de processamento em um menu ou até mesmo quando se deseja

dotar a interface com um botão para auxílio inserido no contexto, o projeto da janela pode torna-se complexo e de difícil manuseio pelo usuário.

Visando simplificar a operação da interface gráfica, no sentido que o manuseio de um único componente seja suficiente para executar uma função, foi desenvolvido um novo objeto para compor o O_2Kit, implementado no esquema do banco de dados como uma subclasse da superclasse "Component". Esta classe, denominada "Botao", foi definida, para fins de utilização dos recursos gráficos do O_2Look, como sendo um objeto e foram acrescentados dois métodos aos herdados da classe "Component": método "processa" e "ajuda". Os recursos gráficos da aplicação foram modificados para que o menu dos botões passasse a exibir apenas estes dois métodos com os nomes alterados. Esta solução resolveu o problema permitindo o uso de diversos botões numa única janela para execução direta de funções.

7.6.2. Consultas à Propriedades de Subclasses

O teste do projeto foi efetuado utilizando um único "named set" para as instâncias de cada superclasse. A solução pretendia simplificar o esforço de programação na recuperação de objetos utilizando os métodos do O_2 para localização das suas instâncias. Mesmo considerando que a identificação do tipo e da classe de instância pode ser efetuada durante a execução de um programa ou de uma consulta de forma trivial, falta às linguagens de programação e consulta um operador específico para esta separação.

7.6.3. Acesso aos Objetos

O acesso às instâncias de uma classe é efetuado pelo O_2C através do comando "for" onde é fornecido um objeto para ser atribuído a todas as instâncias que satisfizerem as condições definidas no comando (cláusula where). Desta maneira, o O_2 funciona como um servidor de objetos e não de servidor de páginas. Antes de fornecer os objetos pretendidos, o O_2 compila uma a lista de acesso ao set nomeado e, concluída a montagem da lista, começa a liberar as instâncias encontradas. Com o emprego deste método, o O_2 otimiza o acesso à um set de objetos de forma não visível ao usuário. O usuário também é impedido de intervir neste método de acesso.

A primeira conseqüência desse método de acesso é a obtenção de tempos de processamento grandes em sets com muitas instâncias. A função de construção da lista de ítens geográficos apresenta uma demora média de 8 segundos para exibir a

primeira janela, onde procura 6 instâncias num set com cerca de 5.000. A tentativa de reduzir o tempo de espera com o emprego de índices não apresentou nenhum ganho perceptível. Como as instâncias dessa classe são hierarquizadas, um acesso do tipo sequencial-indexado possivelmente apresentaria melhores resultados.

Outro problema constatado no teste do projeto foi a não liberação da memória utilizada ao término do acesso a um set. O O_2 mantém a memória utilizada para a construção de uma lista, mesmo após o término de uma interação, até que seja emitido um commit. Um programa de teste que utilizou três vezes o "for", em três pontos diferentes, para acesso a um "named set" com cerca de 30.000 instâncias não pode ser processado, sendo cancelado pelo sistema devido a insuficiência de memória. O programa, para cada condição de processamento, buscava as instância de determinadas subclasses, uma de cada vez. Após ser alterada sua estrutura para primeiro selecionar as instâncias de interesse e depois operar com as condições estabelecidas o programa foi executado sem problemas.

7.6.4. Mecanismo de Transações

O problema de uso da memória é crítico no O_2. O usuário não dispõe de recursos para intervir, ficando limitado ao uso dos comandos de commit e validate. O validate permite descarregar para a base de dados as transações já concluídas que estejam em memória, não desalocando as áreas de memória solicitadas em algum ponto anterior. O empilhamento de janelas e a construção de janelas com muitos atributos são, além da construção de listas descritas no item anterior, fatores de grande demanda de memória. A solução encontrada para poder trabalhar com transações maiores ou para editar janelas com muitos atributos foi dividir a transação em transações menores e utilizar o commit para liberar a memória necessária.

Esse uso do commit, desvirtuando o conceito de transação, acarreta na construção de um procedimento para "restart" pois o término de uma transação é também entendido pelo O_2 como final de processamento. A solução adotada consistiu na utilização de uma variável global para o programa que emitia um commit informar ao programa de restart qual seria o próximo a ser executado.

7.6.5. Alfabeto Português

Outra restrição observada na utilização do O_2 diz respeito ao uso do alfabeto português. Apesar do manual do O_2C apresentar um exemplo de uso de caracteres

acentuados (em francês, no exemplo de programação anexado), isso se dá apenas no arquivo de recursos gráficos uma vez que a versão em uso do compilador CC não aceita nem a introdução de vogais acentuadas nem do "ç". Foi experimentada tanto a definição dos caracteres através de seu valor numérico como por composição no teclado. Nas duas tentativas o compilador acusou a existência de erros graves que impediram a geração do código utilizando textos em português.

Na nota onde se comenta o uso da vogal acentuada, o manual alerta para possíveis incompatibilidades no uso destes caracteres, justificando esta restrição na limitação de fontes das interfaces gráficas e responsabilizando o programador para uma correta seleção.

8. CONCLUSÕES

Apresentamos neste trabalho uma revisão da literatura sobre bancos de dados estatísticos e sobre as técnicas de Orientação a Objetos, sobre Trabalho Cooperativo Apoiado por Computador e sobre Programação Visual aplicada a sistemas de computadores, não utilizadas nos bancos convencionais, que podem contribuir para solucionar problemas existentes e aprimorar as interações dos usuários com estas categorias de sistemas.

O uso de um SGBDOO, não obstante as facilidades que apresentou para o desenvolvimento do projeto de teste, deve ser objeto de análise de desempenho neste ambiente. Em alguns momentos da experimentação do projeto, os tempos de resposta ultrapassaram períodos considerados como toleráveis, mesmo naqueles em que o equipamento estava ocupado somente com o processamento do teste. Para o problema da formação de listas de acesso à conjuntos de informação (vide item 7.6.3) a perturbação foi minimizada com a alteração do projeto. Porém, em outras contenções a causa não foi ainda determinada exigindo monitoração do projeto para averiguar entre consequências de problemas estruturais do O_2, inadequações dos sistemas orientados a objetos para bancos de dados estatísticos ou problemas de projeto da aplicação.

Os problemas constatados com o desempenho não invalidam os demais resultados obtidos no teste. A extensão feita no *toolkit* do O_2, adicionando botões para disparar diversos processos (vide item 7.6.1), permitiu a construção de uma interface gráfica com todas as recomendações da literatura revista e apresentada no capítulo 5. Mesmo não dispondo do mais completo conjunto de ferramentas gráficas, o *toolkit* do O_2 permite elaborar interfaces simples devido a boa variedade de ferramentas oferecidas.

O protótipo deixou de examinar um dos principais itens propostos, o do emprego de processamento cooperativo apoiado por computador, devido às restrições da versão disponível do O_2. O trabalho limitou-se à revisão da literatura do assunto e a especulação sobre a estrutura e requisitos funcionais de CSCW em bancos de dados estatísticos.

A modelagem utilizando a orientação a objetos permitiu implementar um ambiente onde o uso de códigos de identificação das informações foi efetuado mais no aspecto conservador do projeto do que por exigência de manipulação das informações. O uso de atributos referenciais no modelo tornou, sob o ponto de vista do manuseio de informações no ambiente do projeto, totalmente desnecessário o uso de sistemas de codificação. Como as fontes que forneceram os dados utilizados no projeto de teste produzem as informações utilizando códigos, os mesmos foram acrescentados para facilitar o intercâmbio e o teste dos resultados. A introdução de códigos no passado foi produto da incapacidade de manuseio dos dados pelos sistemas de suporte existentes.

O uso de nomes e descrições representa a forma natural do homem para lidar com as informações de hierarquias e sistemas de classificação. A habilidade dos modelos orientados a objetos em facultarem a substituição dos sistemas de codificação pela origem da informação deve ser considerada como um fator decisivo para a utilização de SGBDOO em bancos de dados de dados estatísticos.

Cabe, ainda, considerar a rapidez de implementação propiciada pela linguagem de programação orientada a objetos O_2C. A medida que os conceitos de utilização foram melhor assimilados e que aumentou a experiência de programação, a tarefa de desenvolvimento passou a ser concentrada na concepção da lógica dos programas visto que a codificação passou a ser executada rapidamente. O código necessário à implementação de um programa em O_2C é pequeno e poderoso. Vale ainda acrescentar o conforto e a confiabilidade obtidos com a versão 4.2 do O_2. As versões que foram utilizadas antes, a 3.6.3 e a 4.0, tinham muitos erros que acarretavam a descontinuidade de uma sessão, até mesmo por erros na codificação que não sabiam ser tratados pelo compilador O_2C.

Finalmente, o desenvolvimento do protótipo encoraja o prosseguimentos de novos trabalhos na área como o desenvolvimento de um Banco Cooperativo de Dados Estatísticos, um BDEOO com as técnicas de CSCW e atendendo à consultas coletivas multidisciplinares e ao tutor do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

[ANGE90] Angelaccio,M., Catarci,T., Santucci,G., QBD*: A Graphical Query Language with Recursion, IEEE Transactions on Software Engineering, Vol.16, No.10, Out 1990, pp.1150-1163.

[APPL86] Applegat, L.M., Konsynski, B.R., Nunamaker, J.F., A group decision support system for idea generation and issue analysis in organization planning, in Proceedings of the First Conference on Computer-Supported Cooperative Work, Austin, Texas, Dez 1986, pp 16-34.

[ATKI89] Atkinson,M., Bancilhon,F., DeWitt,D., Dittrich,K., Maier,D., Zdonik,S., The Object-Oriented Database System Manifesto, Relatório Técnico ALTAIR 30-89, ago 1989, França, p.18

[BAEC91] Baecker,R., Small,I., Mander,R., Bringing Icons to Life,Conference on Computer Human Interface 91 - Proceedings, New Orleans, USA, Abr 1991, pp.1-6.

[BANC89] Bancilhon,F., *et al*, The Design and Implementation of O_2, an Object-Oriented Database System, Second International Workshop on Object-Oriented Database System, K.R.Dittrich ed., Bad Münster, Alemanha, Set 1988.

[BECK80] Beck,L., A Security Mechanism for Statistical Databases, ACM Transactions on Database Systems, Vol.5,No.3, Set 1980, pp.316-338.

[BONF91] Bonfiglio,A., Malatesta,G., Tisato,F., Conference Toolkit: A Framework for Real-Time Conferencing, in Studies in Computer Supported Cooperative Work, ed. J.M.Bowers & S.D.Benford, North-Holland Publ., 1991.

[BORR92] Borras,P., Mamou,J.C., Plateau,B., Poyet,B., Tallot,D., Building user interfaces for database applications: The O2 experience, SIGMOD RECORD, Vol.21, No.1, Mar 1992, pp.32-38.

[BROW83] Brown,W.A., Navathe,S.B., Su,S.Y.W., Complex data types and a data manipulation language for scientific and statistical databases, 2nd Int Workshopp Statistical Database Management, Los Altos, CA, Set 1983.

[CAMA92] Camargo,C.S., FLECHA: Um editor gráfico cooperativo para o ambiente do Geotaba, dissertação de MsC, Coppe/UFRJ, 1992.

[CANT93] Canter,S., ed., Statiscal Analysis, Stat of the Art, PC Magazine, Vol.12, No.9, Mai 1993, pp.227-287.

[CATT91] Cattell,R.G.G., Object Data Management, Addison-Wesley Publishing, 1991, p-318.

[CHAN90] Chang, S.K., ed., Principles of Visual Programming Systems, Prentice-Hall, 1990.

[CHEN77] Chen,P.P.S., The Entity-Relantionship Model: A Basis for the Enterprise View of Data, Proceedings of 1977 National Computer Conference, Dallas, Texas, AFIPS Conference Proceedings, Vol.46, 1977, pp. 77-84.

[CHIN78] Chin,F.Y., Security in Statistical Databases for Queries with Small Counts, ACM Transactions on Database Systems , Vol.3, No.1, Mar 1978, pp.92-104

[CHOU86] Chou,H., DeWitt,D., Klug,A., Design and Implementation of the Wisconsin Storage System, Software Practice and Experience, Vol.15, No.10, Out 1985.

[CODA71] CODASYL - Programming Language Committee Report, ANSI Data Base Task Group, Abr 1971.

[CODD72] Codd, E.F., Relational completeness of database sublanguages, Database Systems, Computer Science Symposia Series, Vol. 6, Englewood Cliffs, NJ: Prentice-Hall, 1972.

[COMI90] Stonebraker,M., Rowe,L.A., Lindsay,B., Gray,J., Carey,M., Brodie,M., Bernstein,P., Beech,D. (Committee for Advanced DBMS Function), Third-Generation Data Base System Manifesto, relatório técnico, 1990, p-30.

[DENN79] Denning,D.E., Denning,P.J., Schwartz,M.D., The Tracker: A Threat to Statistical Database Security, ACM Transactions on Database Systems, Vol.4, No.1, Mar 1979, pp.76-96.

[DENN80] Denning,D.E., Secure Statistical Databases with Random Sample Queries, ACM Transactions on Database Systems ,Vol.5, No.3, Set 1980, pp.291-315.

[DOBK79] Dobkin,D., Jones,A.K., Lipton,R.J., Secure Databases:Protection Against User Influence, ACM Transactions on Database Systems, Vol.4, No.1, Mar 1979, pp.97-106.

[ELLI91] Ellis,C.A., Gibbs,S.J., Rein,G.L., GROUPWARE: Some Issues and Experiences, Communications of the ACM, Vol.34, No.1, Jan 1991, pp.38-58.

[FISH88] Fish,R.S., Kraut,R.E., Leland,M.D.P., Quilt: a collaborative tool for cooperative writing, ACM SIGOIS Bulletin, Vol.9, No.2&3, Mar 1988, pp.30-37.

[GHOS86] Ghosh, S.P., Statistical Data Reduction for Manufacturing Testing, IEEE, 1986, pp.58-66.

[GRAC80] Graciano Sá, Cadastro, Tabelas e Padrões: Um exame do sistema estatístico nacional, Revista de Administração de Empresas, 20(2), Abr/Jun. 1980, pp.25-34.

[GRUD90a] Grudin,J., The Computer Reaches Out: The Historical Continuity of Interface Design, Conference on Computer Human Interface 90 - Proceedings, Abr 1990, pp.261-268.

[GRUD90b] Grudin,J., Interface, Conference on CSCW 90 Proceedings, Out 1990, pp.269-278.

[GRUD91] Grudin,J., CSCW: The Convergence of Two Development Contexts, Conference on Computer Human Interface 91 -Proceedings, New Orleans, USA, Abr 1991, pp.269-278.

[GUED89] Guedes,A.P., Banco de Metadados, Seminario Bases de Datos y Difusión Computacional, ONU/CEPAL, 27 Nov-1 Dez, 1989, Cuernavaca, Mexico.

[HIRA90] Hirakawa,M., Tanaka,M., Ichikawa,T., An Iconic Programming System, HI-VISUAL, IEEE Transactions on Software Engineering, Vol.16, No.10, Out 1990, pp.1178-1184.

[HUAN90] Huang, K.T., Visual Interface Design System, em Visual Programming System, ed. Shi-Kuo Chang, Prentice-Hall, 1990, pp.60-143.

[IBGE91] Banco de Metadados - Apresentação do Sistema, Banco de Metadados - Manual de Consulta, Banco de Metadados - Manual de Atualização, manuais publicados pelo DI/DEBAD - Departamento de Administração da Base de Dados, Fundação IBGE, Rio de Janeiro, 1991.

[ICHI90a] Ichikawa,T., Chang,S.K., ed., Introduction Visual Programming, IEEE Transactions on Software Engineering, Vol.16, No.10, Out 1990, pp.1105

[ICHI90b] Ichikawa,T., Hirakawa,M., Iconic Programming: Where to Go?, IEEE Software, Nov 1990, pp.63-68.

[JONE91] Jones,R., User Interface Architectures for Information Sharing, ACM / SIGOIS Bulletin, Vol.13, No.1, Abr 1992, pp.13-14.

[KARA83] Karasolo,I., Svensson,P., An overview of CANTOR - A new system for data analysis, 2nd Int Workshopp Statistical Database Management, Los Altos, CA, Set 1983.

[KIM89] Kim,W., A Model of Queries for Object-Oriented Databases, Proceedings of the Fifteenth International Conference on Very Large Databases, Amsterdam, Out 1989, pp.423-432

[KLUG82] Klug,A.., Equivalence of relational algebra and relational calculus query languages having aggregate functions, Journal of the ACM, Vol. 29, No. 3, 1982.

[KLUG83] Klug,A., ABE - A query language for constructing aggregates by example, 2nd Int Workshopp Statistical Database Management, Los Altos, CA, Set 1983.

[KNIS90] Knister,M.J., Prakash,A., DistEdit: ADistributed Toolkit for Supporting Multiple Group Editors, Proceedings of 3-CSCW, Out 1990, pp.343-355.

[LAKI88] Lakin,F., A Performing Medium for Working Group Graphics, in Computer-supported cooperative work: A book of readings, I.Greif ed.,Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

[LEWI88] Lewis,B.T., Hodges,J.D., Shared Books: Collaborative Publication Management for an Office Information System, ACM SIGOIS Bulletin, Vol.9, No.2&3, Mar 1988, pp. 197-204.

[LIU89] Liu,F.S., Tai,J.W., Some Principles of Icon Construction, Visual Database Systems, T.L.Kunii, ed., Elsevier Science Publishers, IFIP, 1989, pp.89-104.

[MALV89] Malvestuto,F.M., Moscarini,M., Aggregate Evaluability in Statistical Databases, Proceedings of the Fifteenth International Conference on Very Large Databases, 1989, Amsterdam, pp.279-286.

[MANE83] Maness,A.T., Dintelman,S.A., The GENESYS data definition facilities, 2nd Int Workshopp Statistical Database Management, Los Altos, CA, Set 1983.

[MCCA82] McCarthy,J.L., Metadata Management for Large Statistical Database, Proceedings of the Eighth International Conference on Very Large Databases, 8, 1982, Mexico City, Set 1982, pp.234-243.

[NEER93] Neercincx,M., Greef,P., How to Aid Non-Experts, INTERCHI'93 - Conference Proceedings, Amsterdam, Abr 1993, pp.165-171.

[NEUM90] Neumann,K., Co-Operative Processing - A Challenge for Statistics and Database Management, COMPSTAT 90 Proceedings , Set 1990, pp.225-236.

[NONO91] Nonogaki,H., Ueda,H., FRIEND21 Project: A Construction of 21st Century Human Interface, Conference on Computer Human Interface 91 - Proceedings, New Orleans, USA, Abr 1991, pp.407-414.

[OPPE88] Opper,S., A Groupware Toolbox, BYTE, Vol.13, No.13, Dez 1988, pp.275-282.

[OZSO83] Ozsoyoglu,G., Ozsoyoglu,Z.M., An extension of relational algebra for summary tables, 2nd Int Workshopp Statistical Database Management, Los Altos, CA, Set 1983.

[O2-91] O_2 Technology, The O_2 User's Manual, version 2.3.1, Versailles, France, Dez 1991.

[OZSO85] Ozsoyoglu,G., Ozsoyoglu,Z.M., Statistical Database Query Languages, IEE Transactions on Software Engineering, Vol.SE-11, No. 10, Out 1985, pp.1071-1081.

[PALL86] Palley,M.A., Security of Statistical Databases: Compromise Through Attribute Correlational Modeling, Proceedings of the 2nd International Conference on Data Engineering, IEEE, 1986, pp.67-74.

[PALL87] Palley,M.A.,Simonoff,J.S., The Use of Regression Methodology for the Compromise of Confidential Information in Statistical Databases, ACM Transactions on Database Systems, Vol.12, No.4, Dez 1987, pp.593-608.

[RAFA88] Rafanelli,M., Research Topics in Statistical and Scientific Database Management, Proceedings of the IV Statistical and Scientific Database Management, Roma, Italia, Jun 1988, pp.1-18.

[RUSS93] Russo,P., Boor,S., How Fluent is Your Interface? Designing for International Users, INTERCHI'93 - Conference Proceedings, Amsterdam, Abr 1993, pp.342-347.

[SADO88] Sadowsky,G., Statistical Data Processing in Development Countries: Problems and Prospects, Interregional Workshop on Statiscal Data Processing and Data Bases in the Developing Countries, Geneve, Switzerland, 30 Mai-3 Jun,1988.

[SCHL80] Schlörer,J., Disclosure from Statiscal Databases: Quantitative Aspects of Trackers, ACM Transactions on Database Systems, Vol.5, No.4, Dez 1980, pp.467-492.

[SHOS82] Shoshami,A., STATISTICAL DATABASES: Characteristics, Problems, and some solutions, Proceedings of the Eighth International Conference on Very Large Databases, Mexico City, Set 1982, pp.208-222.

[STEF87] Stefik,M., Foster,G., Bobrow,D.G., Kahn,K., Lanning,S., Suchman,L., Beyond the Chalkboard: ComputerSupport for Collaboration and Problem Solving in Meetings, Communications of the ACM, Vol.30, No.1, Jan 1987, pp.32-47.

[STEP88] Stephenson,G.A., Knowledge Browsing - Front Ends to Statistical Databases, Proceedings of the IV Statistical and Scientific Database Management, Roma, Italia, Jun 1988, pp.325-337.

[TPLS80] Table Producing Language System, version 5, Bureau of Labor Statistics, Washington, Jul 1980.

[TURN79] Turner,M.J., Hammond,R., Cotton,P., A DBMS for a Large Statistical Databases, Proceedings of the fifth International Conference on Very Large Databases, Mai 1979, Rio de Janeiro, pp.319-327.

[WIEC90] Wiecha,C., Boies,S., Generatig Users Interfaces: Principles and Use of its Styles Rules, ACM, Abr 1990, pp.21-30.

[WU89] Wu,C.T., Hsiao,D.K., Implementation of Visual Database Interface using Object-Oriented Language, Visual Database Systems, T.L.Kunii, ed., Elsevier Science Publishers, IFIP, 1989, pp.105-125.

ANEXO A
GLOSSÁRIO DE TERMOS E ABREVIATURAS UTILIZADOS

- BDE: Banco de Dados Estatísticos

- CSCW: Computer Supported Cooperative Work (trabalho cooperativo apoiado por computador)

- DER: Diagrama Entidades-Relacionamentos

- E-R: Entidades-Relacionamentos

- OID: identificador de objetos

- SGBD: Sistema Gerenciador de Banco de Dados

- SGBDOO: Sistema Gerenciador de Banco de Dados Orientado a Objetos

- SGIU: Sistema Gerenciador de Interfaces do Usuário

- SQL: Structured Query Language (linguagem de consulta ao SGBD)

- TOOLKITS: Conjunto de funções para implementar uma interface gráfica

- 4GL: Forth Generation Language (linguagem de quarta geração)

ANEXO B
ESPECIFICAÇÕES DO PROTÓTIPO

1. INTRODUÇÃO

Conforme apresentado no capítulo 7, as especificações do projeto foram testadas através da implementação de um protótipo. A plataforma utilizada foi a de *workstations SUN/SPARC* operando com o sistema operacional *UNIX-V*, versão 4.2, gerenciador gráfico *MOTIF* e com o *XWindows*. O protótipo foi desenvolvido em O_2C, que é a linguagem de programação do sistema gerenciador de banco de dados orientado a objetos O_2, em sua versão 4.2. A versão utilizada do O_2, apesar de ser do tipo cliente-servidor, era restrita a um cliente por servidor o que limitou o protótipo a somente manusear operações mono-usuárias.

A arquitetura do projeto piloto foi dividida em três grandes aplicações, a saber:

. *BDE* - aplicação para manuseio dos objetos estatísticos propriamente dito, voltada ao usuário final do projeto;

. *METADADO* - aplicação destinada à manutenção das pesquisas estatísticas e de seus metadados no ambiente do projeto, voltada ao administrador do banco de dados;

. *APOIO* - destinada a implementar recursos gerais para as duas aplicações anteriores, como a manutenção dos textos de ajuda utilizados, programas utilitários para a carga de dados, etc.

Neste anexo são detalhadas as aplicações implementadas, descrevendo-se os módulos de suas estruturas, os tipos de objetos manuseados e as telas construídas para a interface do usuário.

Para detalhamento dos módulos implementados, foram considerados como programas as principais funções da aplicação. Neles são descritos os objetivos, os dados, os parâmetros para processamento, o algorítimo de processamento e as saídas produzidas (telas). Ao final de cada programa são relacionadas as funções (rotinas de apoio à programação) utilizadas com uma descrição sucinta de sua finalidade. Foram

acrescentadas na descrição dos módulos da aplicação *APOIO* as funções de apoio à programação desenvolvidas com objetivo de expandir os recursos de programação do O_2C no manuseio de tipos ou para permitir a reutilização de rotinas por mais de um programa.

Os programas que foram desdobrados em diversas funções foram descritos logo após o programa principal com a utilização da mesma estrutura de detalhamento dos programas ou módulos principais.

2. APLICAÇÃO PILOTO

O banco de dados estatístico foi carregado com todos os componentes da estrutura político-administrativa brasileira existentes na época do Censo Demográfico de 1991, além das 10 regiões metropolitanas. Foram também carregados os dados e metadados de duas pesquisas estatísticas do IBGE: a *PAM - Pesquisa Agrícola Municipal* e as *estimativas populacionais produzidas para 1990.*

3. CLASSES DE OBJETOS

Conforme o modelo lógico de dados elaborado para o projeto, foram implementadas na aplicação BDE duas superclasses de objetos, uma para a classe *Dados*, destinada ao manuseio dos objetos estatísticos propriamente ditos, e outra para classe *Usuario*, destinada ao registro do usuário e de suas listas de trabalho. No teste piloto foi definido um "named set" por superclasse, destinado à persistência de suas instâncias e as das classes subordinadas.

A implementação das aplicações METADADOS e APOIO também foi feita de acordo com o projetado e foram definidos "named sets" para dar persistência aos atributos descritivos das pesquisas e para as demais classes de metadados da informação estatística.

3.1. Superclasse *Dados*

A superclasse *Dados* tem por finalidade o registro da identificação comum das informações estatísticas: a distribuição espacial, temporal e de conteúdo. A superclasse *Dados* compreende as classes *D_PAM* e *D_POPULACAO* que herdam as suas propriedades, conforme figura 1.

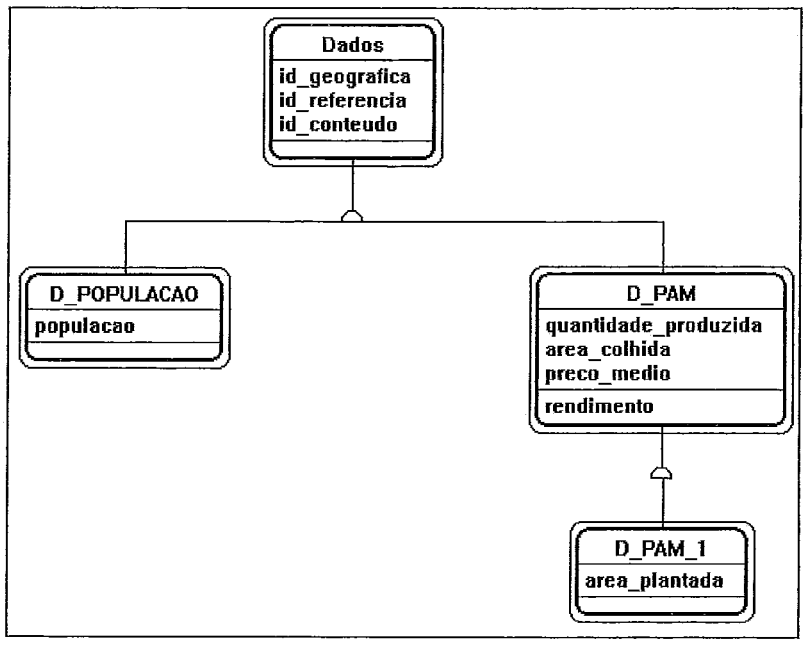


Figura 1 - Diagrama da superclasse *Dados*

A superclasse *Dados* contém os seguintes atributos:

id_geografica: *string* - identificação de instância da classe *Cadastro_geografico* que identifica o item geográfico do local onde os dados foram levantados;

id_referencia: *Ocorrencias* - referência ao objeto da classe *Ocorrencias* que descreve a ocorrência da pesquisa no levantamento dos dados;

id_conteudo: *Descritor* - referência à objeto da classe *Descritor* que identifica a instância do descritor utilizado na indexação da informação.

A persistência das instâncias da superclasse *Dados* e das classes subordinadas é dado pelo *named set DADOS*.

3.2. Classe *D_PAM*

A classe *D_PAM* tem por finalidade registrar as informações da Pesquisa Agrícola Municipal levantadas até 1987. A classe *D_PAM* herda da classe *Dados*, na forma de especialização, os atributos de identificação e acrescenta as seguintes propriedades como atributos:

quantidade_produzida: *real* - quantidade total produzida em unidades de contagem de acordo com o tipo de produto;

area_colhida: *real* - área total colhida em hectares;

preco_medio: *real* - preço médio dos produtos em unidades monetárias da época do levantamento.

A classe *D_PAM* acrescenta, também, o método *rEndimento*, do tipo *real*, destinado a definir, como atributo virtual, o rendimento médio das culturas, ou seja a quantidade produzida dividida pela área colhida.

3.3. Classe *D_PAM_1*

A classe *D_PAM_1* tem por finalidade registrar as informações da Pesquisa Agrícola Municipal levantadas a partir de 1988, herdando da classe *D_PAM*, na forma de especialização, todos os atributos superclasse e acrescentando o atributo *area_plantada*, do tipo *real*, para registrar a área plantada pela cultura em hectares.

3.4. Classe *D_POPULACAO*

A classe *D_POPULACAO* tem por finalidade registrar as estimativas populacionais de 1990. Esta classe herda da classe *Dados*, na forma de especialização, os atributos de identificação e acrescenta o atributo *populacao*, do tipo *real*, destinado ao registro da estimativa da população total em 1990.

3.5. Superclasse *Usuario*

A superclasse *Usuario* destina-se ao registro das informações do usuário e das listas de trabalho utilizadas para fazer uma consulta e extração de dados. A superclasse é especializada em quatro diferentes classes que herdam o atributo de identificação, conforme diagrama da figura 2. A classe *Usuario* contém o atributo de identificação *sigla_do_usuario*, do tipo *string*, onde o usuário é identificado com a sigla de sua sessão no sistema (*unix*). As instâncias da superclasse e das classes subordinadas são mantidas no *named set ARQUIVO*.

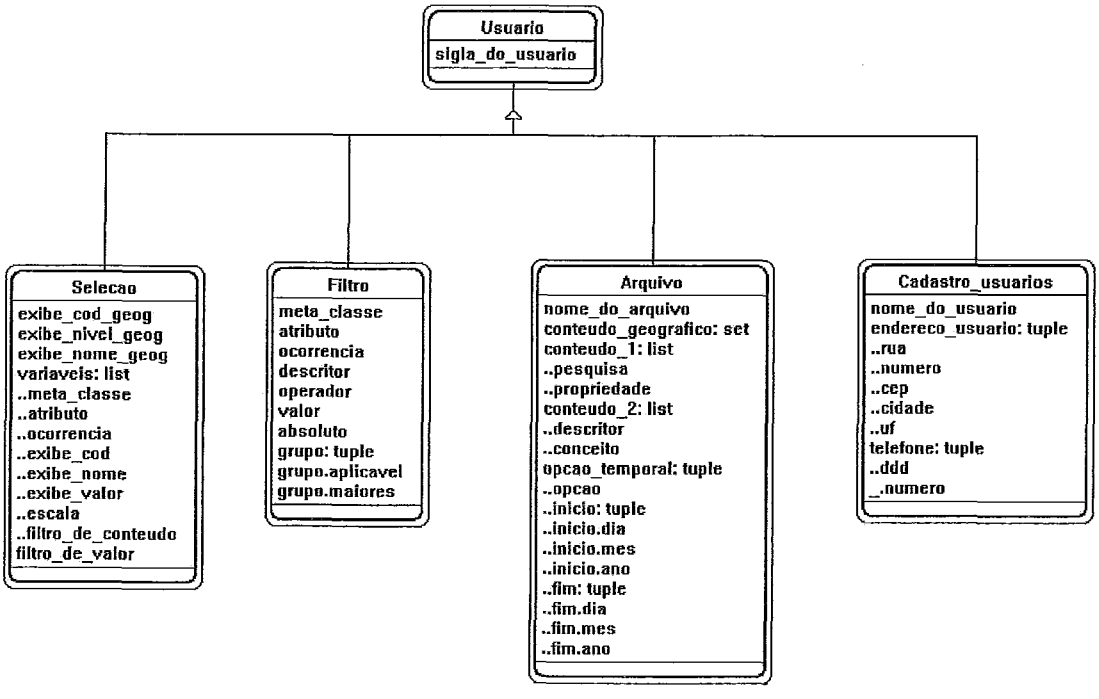


Figura 2 - Diagrama da classe *Usuario*

3.6. Classe *Cadastro_usuarios*

A classe *Cadastro_usuarios* destina-se aos registro cadastral do usuário, herdando, por especialização, o atributo de identificação da superclasse *Usuario*. Esta classe acrescenta os seguintes atributos ao herdado:

nome_do_usuario: *string* - nome do usuário;

endereco_usuario: *tuple* - tupla com o endereço completo do usuário, composta com:

rua: *string* - nome do local (rua, avenida, etc);

numero: *string* - número e complemento do endereço;

cep: *string* - código de endereçamento postal;

cidade: string - nome da cidade;

uf: *string* - sigla da unidade da federação;

telefone: *tuple* - tupla para registro do telefone do usuário, composta com:

ddd: *string* - código de área do telefone;

numero: *string* - número do telefone.

3.7. Classe *Arquivo*

A classe *Arquivo* tem por finalidade registrar as listas preparadas pelo usuário para executar uma consulta na base de dados. Esta classe acrescenta os seguintes atributos ao herdado na superclasse:

nome_do_arquivo: string - nome dado pelo usuário para identificação do conjunto de listas. O sistema utiliza o nome "_BDE_WORK" para o arquivo de trabalho (genérico) de uma sessão do usuário;

conteudo_geografico: set(*Cadastro_geografico*) - lista de itens geográficos representada por um conjunto de instâncias da classe *Cadastro_geografico*;

conteudo_1: *list(tuple)* - lista de variáveis selecionadas na base de dados representada por tuplas compostas de:

pesquisa: *string* - código de identificação da pesquisa;

propriedade: *string* - nome do atributo ou do método;

conteudo_2: *list(tuple)* - lista de assuntos (descritores e conceitos de pesquisa) representada por tuplas compostas de:

descritor: *Descritor* - referência a um objeto da classe Descritor;

conceito: *unique set(Conceito)* - lista dos conceitos de investigação válidos para o descritor selecionado representada pelo conjunto de objetos sem repetição da classe Conceito;

opcao_temporal: *tuple* - tupla descritiva do período de tempo para seleção dos dados composta de:

opcao: *integer* - código numérico representativo do tipo de opção temporal desejada onde: "0" significa obter os dados mais recentes entre os existentes

na base; "1" para os dados mais antigos; "2" para obter todos os dados; e "3" para os dados de determinado período;

inicio: tuple - tupla para a data de início de um período de busca (opção 3) composta de:

ano: *integer* - ano de início (com quatro dígitos);

mes: *integer* - mes de início;

dia: *integer* - dia inicial do período;

fim: tuple - tupla para a data de término de um período de busca (opção 3) composta de:

ano: *integer* - ano de término (com quatro dígitos);

mes: *integer* - mes de término;

dia: *integer* - dia final do período.

3.8. Classe *Selecao*

A classe *Selecao* destina-se a receber a tradução do conteúdo das listas elaboradas na classe *Arquivo* para o formato de busca de informações utilizado na função de elaboração de programas de busca de informações. Ao atributo herdado da superclasse *Usuario*, esta classe se especializa com:

exibe_cod_geog: boolean - valor lógico destinado a instruir quanto a exibição do código do item geográfico no resultado final, onde "verdadeiro" significa exibi-lo;

exibe_nivel_geog: boolean - valor lógico destinado a instruir quanto a exibição do nível do item geográfico no resultado final, onde "verdadeiro" significa exibi-lo;

exibe_nome_geog: boolean - valor lógico destinado a instruir quanto a exibição do nome do item geográfico no resultado final, onde "verdadeiro" significa exibi-lo;

variaveis: list(tuple): lista com as tuplas descritivas dos atributos dos objetos a serem recuperados para composição da base do usuário ou para uso como filtro de informações, contendo ainda instruções sobre a forma de manuseio e exibição dos valores no resultado final;

atributo: string - nome da propriedade (atributo ou método) na classe;

ocorrencia: Ocorrencias - objeto da classe Ocorrencias relativo a época do levantamento de dados da pesquisa;

exibe_cod: boolean - valor lógico destinado a instruir quanto a exibição do código do descritor da informação no resultado final, onde "verdadeiro" significa exibi-lo;

exibe_nome: boolean - valor lógico destinado a instruir quanto a exibição do nome do descritor da informação no resultado final, onde "verdadeiro" significa exibi-lo;

exibe_valor: boolean - valor lógico destinado a instruir quanto a exibição do valor da informação no resultado final, onde "verdadeiro" significa exibi-lo;

escala: integer - coeficiente utilizado para mudar a escala de valores das informações, onde o padrão é a unidade;

filtro_de_conteudo: unique set (Filtro) - conjunto sem repetição de referências à filtros de conteúdo aplicáveis sobre a variável;

filtro_de_valor: unique set (Filtro) - conjunto sem repetição de referências à filtros de valor aplicáveis ao processo de seleção de valores (não implementado);

3.9. Classe *Filtro*

A classe *Filtro,* subclasse da classe *Usuario,* destina-se ao registro das condições impostas para a recuperação de informações (filtros de conteúdo e de valor). Ao atributo herdado, a classe Filtro especializa a definição acrescentando:

atributo: string - nome da propriedade (atributo ou método) que contém a variável a ser utilizada na comparação;

ocorrencia: Ocorrencias - objeto da classe *Ocorrencias* que caracteriza o atributo selecionado em relação a época do levantamento de dados da pesquisa;

descritor: Descritor - objeto da classe *Descritor* que detalha o conteúdo para utilização da variável como filtro;

operador: integer - código numérico descritivo do operador do filtro, onde: "0" significa igual ou pertence; "1" maior; "2" menor; "3" maior ou igual; "4" menor ou igual; "5" diferente; e "9" não é aplicável à filtro de conteúdo;

valor: real - valor para ser utilizado na comparação.

absoluto: boolean - valor lógico informando se o valor a ser utilizado na comparação é um valor absoluto ou um percentual;

grupo: tuple - tupla contendo a descrição do tipo de grupo para ser utilizado na comparação dos filtros de valor (não implementados), composto de:

aplicavel: boolean - valor lógico para considerar o valor de comparação aplicável ao valor total do grupo de comparação;

maiores: boolean - valor lógico para utilizar como grupo de comparação o grupo dos maiores, conforme definição percentual do atributo valor;

3.10. Classe *Pesquisa*

A superclasse *Pesquisa* destina-se ao registro das pesquisas, das suas características e das suas ocorrências, onde procura aumentar a descrição semântica dos ob-

jetos estatísticos. A superclasse *Pesquisa* se especializa em duas classes, *Descricao_da_pesquisa* e *Ocorrencias,* conforme diagrama da figura 3.

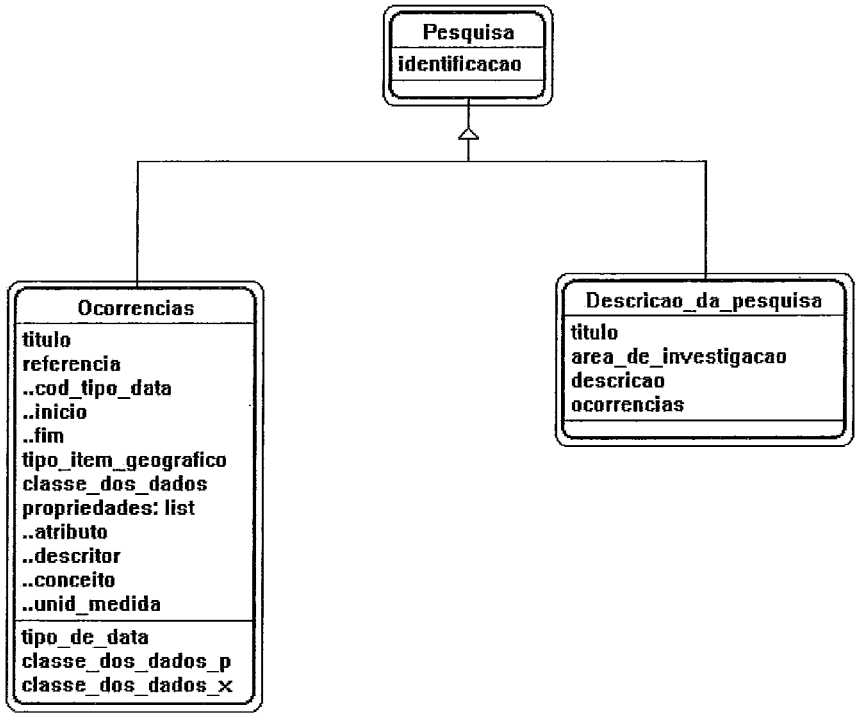


Figura 3 - Diagrama da classe *Pesquisa*

A classe *Pesquisa* define o atributo de identificação das pesquisas *identificacao*, do tipo *string*, para registrar como código de identificação. Todos os objetos da superclasse *Pesquisa* e classes subordinadas são mantidos no *named set PESQUISA*.

3.11. Classe *Descricao_da_pesquisa*

A classe *Descricao_da_pesquisa* tem por finalidade registrar os dados característicos de uma pesquisa e manter o controle sobre suas ocorrências. Esta classe acrescenta ao atributo de identificação herdado da superclasse as seguintes propriedades:

titulo: string - texto livre contento a designação utilizada para a pesquisa.

area_de_investigacao: string - texto livre informando a área de investigação da pesquisa;

descricao: Text - objeto do tipo *Text* destinado a um texto descritivo das características das pesquisa;

ocorrencias: set(Ocorrencias) - conjunto de ocorrências registradas na base de dados para a pesquisa.

3.12. Classe *Ocorrencias*

A classe *Ocorrencias* destina-se a registrar a época da realização dos levantamentos de dados das pesquisas e a relacioná-los com uma classe de dados. Através do dicionário de propriedades da classe são mantidas as referências utilizadas no processo de indexação da base. A classe acrescenta os seguintes atributos ao atributo herdado da superclasse:

titulo: string - texto livre descritivo da designação utilizada para a ocorrência da pesquisa;

referencia: tuple - tupla descritiva da época de referência das informações, contendo:

cod_tipo_data: integer - código descritivo do tipo de período de tempo coberto na realização da ocorrência, onde: "0" significa realização eventual; "1" diária; "2" semanal; "3" decendial; "4" quinzenal; "5" mensal; "6" bimestral; "7" trimestral; "8" quadrimestral; "9" semestral; "10" anual; "11" bianual; "12" trianual; "13" qüinqüenal; e "14" decenal;

inicio: tuple - tupla contendo a data de início do período de referência no levantamento de dados, contendo:

ano: *integer* - ano de início (com quatro dígitos);

mes: *integer* - mes de início;

dia: *integer* - dia inicial do período;

fim: tuple - tupla contendo a data de término do período de referência, composta de:

ano: *integer* - ano de término (com quatro dígitos);

mes: *integer* - mes de término;

dia: *integer* - dia final do período.

tipo_item_geografico: unique set - conjunto de códigos dos níveis geográficos dos dados existentes na base de dados (não implementado);

classe_dos_dados: Meta_class - metaclasse utilizada para o armazenamento dos dados.

propriedades: list(tuple) - lista das propriedades da classe dos dados relacionadas com os descritores da informação, com os conceitos de investigação e com a unidade de medida utilizada. A tupla é composta com:

atributo: string - nome da propriedade (atributo ou método) dos objetos da classe;

descritor: Descritor - objeto da classe *Descritor* que descreve os dados;

conceito: Conceito - objeto da classe *Conceito* que descreve o tipo de investigação estatística utilizado no levantamento da informação;

unid_medida: Unidade_medida - objeto da classe *Unidade_medida* que descreve a unidade utilizada no registro do valor da informação;

A classe *Ocorrencias* também acrescenta os seguintes métodos:

tipo_de_data: string - para a interpretação literal do código de tipo de data;

classe_dos_dados_p: (Meta_class) - destinado a manutenção do atributo privado *classe_dos_dados*;

classe_dos_dados_x: Meta_class - destinado a leitura do atributo privado *classe_dos_dados*;

3.13. Classe *Descritor*

A classe *Descritor*, integrante do Sistema de Metadados, destina-se a descrição dos indexadores das informações estatísticas e do seu emprego em pesquisas. Esta classe, visualizada no diagrama da figura 4, apresenta os seguintes atributos:

descritor: string - denominação do descritor da informação;

codigo: string - texto livre para o registro de um código utilizado pelo descritor;

tipo: integer - código descritivo do tipo de descritor, onde: "1" significa tema; "2" área temática; de "3" a "8" classificações; "9" objetos simples;

descricao: string - texto livre para a descrição conceitual do descritor;

classificacao: set(Descritor) - conjunto das classificações que compreendem diretamente o descritor;

Figura 4

objetos: set(Descritor) - conjunto de descritores diretamente associados ao descritor.

ocorrencias: set(Ocorrencias) - conjunto de ocorrências de pesquisas onde existam variáveis indexadas pelo descritor;

termos_equivalentes: set(Descritor) - conjunto de descritores equivalentes (não implementado);

Os objetos da classe *Descritor* são mantidos no named set *DESCRITOR.*

3.14. Classe *Conceito*

A classe Conceito, parte do Sistema de Metadados, destina-se ao registro dos conceitos de investigação estatística utilizados na produção de informações (figura 5).

Fazem parte da classe Conceito os seguintes atributos:

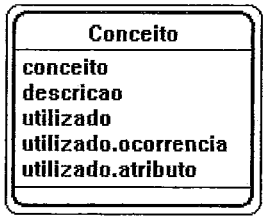


Figura 5

conceito: string - texto resumido do conceito de investigação estatístico;

descricao: string - texto livre descritivo do conceito;

utilizado: list(tuple) - lista de tuplas contendo a relação das ocorrências e de seus atributos onde o conceito foi empregado;

ocorrencia: Ocorrencias - referência à uma instância da classe *Ocorrencias*;

atributo: string - nome da propriedade (atributo ou método) para a qual se aplica o conceito de investigação

A persistência da classe é dada pelo *named set CONCEITOS.*

3.15. Classe *Unidade_medida*

Esta classe, parte do Sistema de Metadados, destina-se ao registro das unidades de medida em uso visando, além do aumento semântico, o manuseio de valores pelo sistema de recuperação (figura 6). Fazem parte desta classe os seguintes atributos:

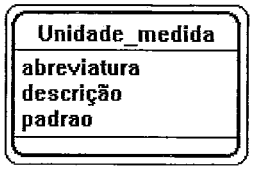


Figura 6

abreviatura: string - abreviatura utilizada para a unidade de medida;

descricao: string - texto livre descritivo da unidade de medida;

padrao: integer - coeficiente de conversão da unidade para a unidade padrão;

A persistência dos objetos desta classe é obtida com o *named set UNIDADES*.

3.16. Classe *Cadastro_geografico*

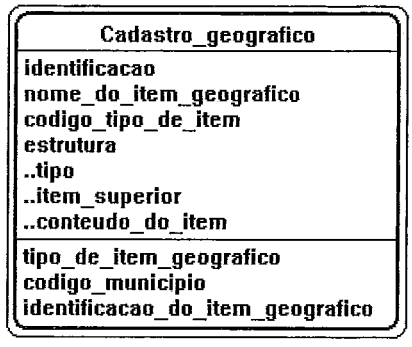


Figura 7

A classe *Cadastro_geografico* destina-se ao registro dos itens componentes da estrutura político-administrativa e regiões metropolitanas. Fazem parte desta classe os seguintes atributos:

identificacao: string - código composto por 12 (ou 16) caracteres usado para a identificação dos itens componentes da estrutura político-administrativa do Brasil e das regiões metropolitanas, de acordo com a estrutura:

```
Brasil                - 0
Grandes Regiões       - x
Unidades da federação- xu  (u de zero a nove)
Mesorregiões          - xuee
Microrregiões         - xueeiii
Região Metropolitana - xu007rr
Município             - xueeiiimmmmm
Distrito              - xueeiiimmmmmdddd
```

nome_do_item_geografico: string - designação completa do item geográfico;

codigo_tipo_de_item: integer - código descritivo do tipo de item geográfico, onde: "0" é usado para o nível Brasil; "1" para grandes regiões; "2" para unidades da federação; "3" para mesorregiões; "4" para microrregiões; "5" para regiões metropolitanas; "98" para municípios; e "99" para distritos(nível não implementado);

estrutura: list(tuple) - conjunto de atributos destinados a caracterizar e situar o item geográfico na estrutura:

tipo: integer - código identificador do tipo de estrutura geográfica, onde "0" representa a estrutura formal e "1" a de regiões metropolitanas;

item_superior: Cadastro_geografico - atributo referencial à objeto da classe de hierarquia imediatamente superior;

conteudo_do_item: set(Cadastro_geografico) - referência ao conjunto de objetos da classe diretamente subordinados;

A classe *Cadastro_geografico* também contém os seguintes métodos:

tipo_de_item_geografico: string - método destinado a decodificar o tipo de item geográfico;

codigo_municipio: string - método para formar o código simplificado de município, composto pelo código da unidade da federação (2 caracteres) seguido dos 4 caracteres iniciais do código do município;

identificacao_do_item_geografico: string - método para editar o código dos itens geográficos, separando os diversos componentes por pontos.

Os objetos da classe *Cadastro_geografico* são mantidos no *named set CADGEO*.

3.17. Classe *Ajuda*

A classe Ajuda destina-se a manutenção dos textos de auxílio utilizados na aplicação BDE. Esta classe é composta por dois atributos:

Figura 8

referencia: string - código para acesso ao texto de auxílio ao usuário;

texto: Text - referência à objeto da classe Text utilizado para a edição dos textos de auxílio;

Os objetos desta classe são mantidos no *named set AJUDA*.

4. APLICAÇÃO BDE

A estrutura utilizada para implementar a aplicação BDE pode ser visualizada nos diagramas de módulos constantes das figuras 9 à 14.

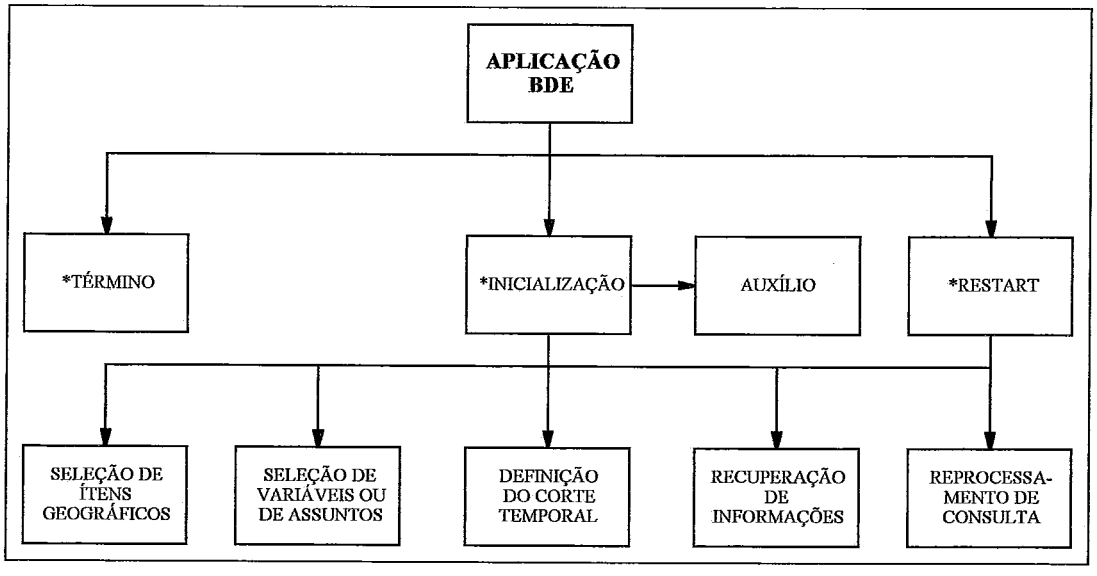


Figura 9 - Diagrama da aplicação BDE

Observação: "*" programa privado invisível aos usuários da aplicação

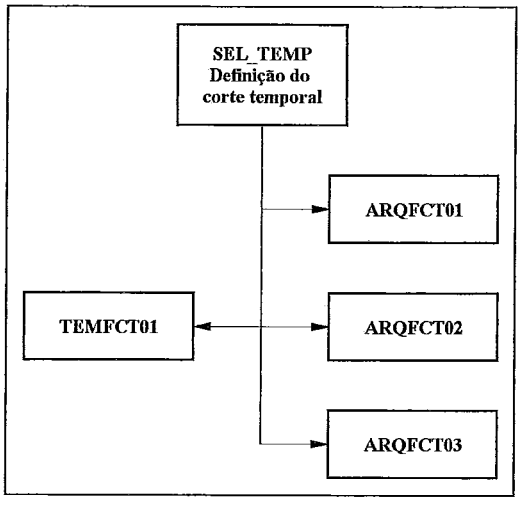


Figura 10 - Diagrama da função de definição do corte temporal

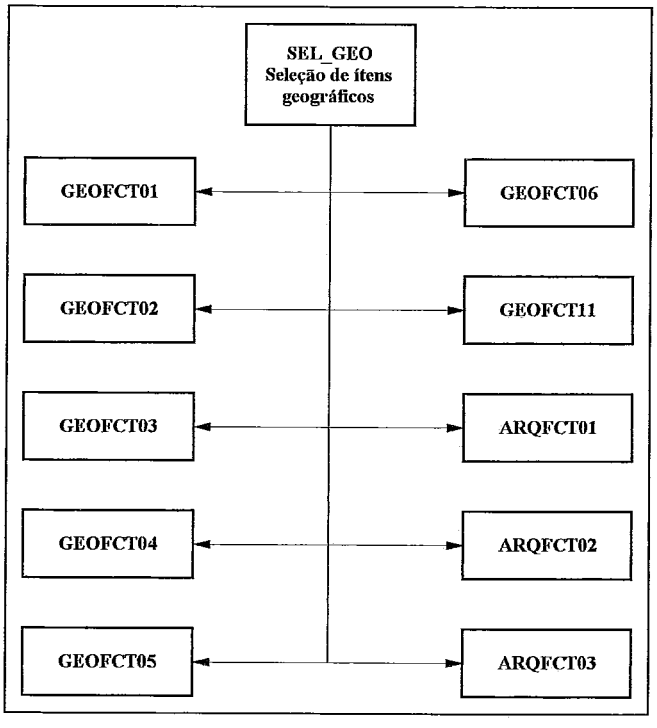


Figura 11 - módulos da função de seleção de ítens geográficos

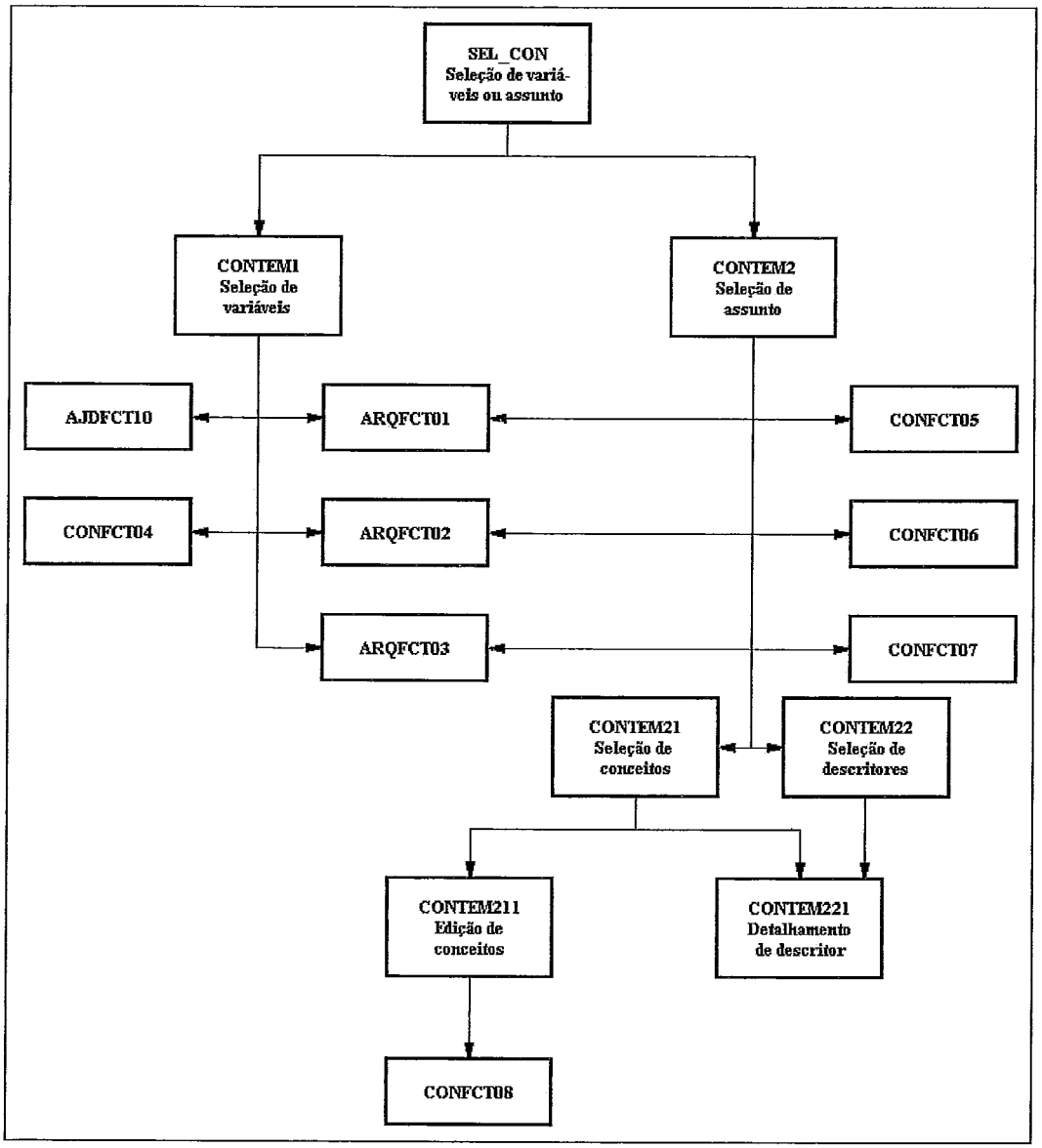


Figura 12 - módulos da função de seleção de variáveis ou de assuntos

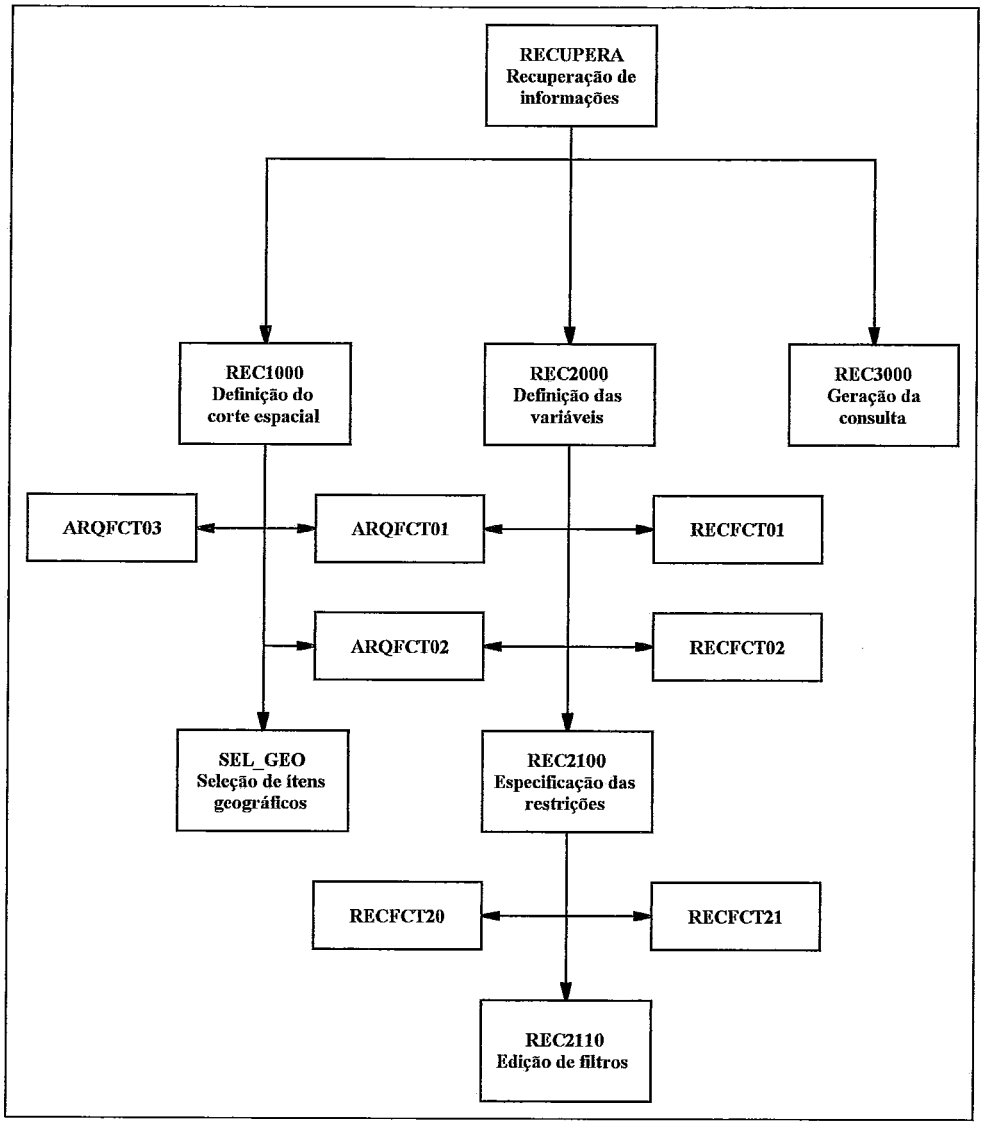


Figura 13 - módulos das funções de recuperação de informações

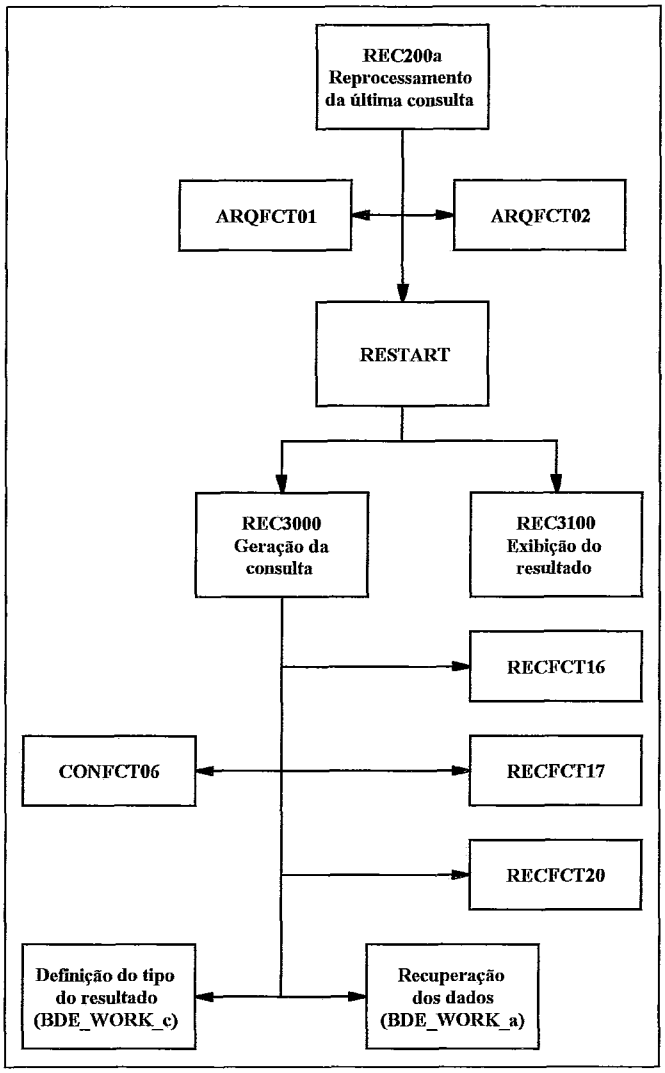


Figura 14 - módulos de consulta e processamento

4.1. Detalhamento dos Módulos

Todos os programas da aplicação BDE utilizam uma interface comum para transferência de informações através de parâmetros. O padrão consiste na transferência do nome do arquivo em uso e o *pid* (presentation identifier) da última tela empilhada pelo programa chamador. Ao término da execução, os programas devolvem para o chamador o nome do arquivo do usuário em uso. Caso a execução seja cancelada através do uso do botão *CANCELA*, o programa chamado retorna um objeto *nil*. Devido a impossibilidade de devolver um parâmetro para o programa chamador em O_2C, os programas foram codificados como funções, única forma capaz de aceitar este padrão.

4.1.1. Programa *init*

a) Objetivo:

O programa *init*, declarado como do tipo privado para a aplicação BDE, é executado automaticamente no início da aplicação (característica das aplicações no O_2). Com ele é feito o controle da execução dos programas da aplicação até que um *commit* seja emitido e o programa *restart* assuma o controle do processamento. Este programa é o responsável pelo manuseio do menu de opções da aplicação BDE chamando o programa corresponde à opção feita pelo usuário e, ao término deste, recebe de volta o controle da aplicação.

b) Dados de entrada:

O programa manuseia a tela do menu de opções após ser acionado o botão *OK*.

c) Descrição do processamento:

Ao ser iniciado, o programa verifica a sigla do usuário para buscar o objeto da classe arquivo associado cujo nome seja BDE_WORK. Caso não exista, o programa constrói uma nova instância na classe Arquivo para o trabalho do usuário. O programa mantém o menu de opções da aplicação para seleção única pelo usuário. Quando o botão *OK* for acionado, ele ativa o programa correspondente. Ao receber de volta o controle, o programa *init* verifica se o programa chamado devolveu um nome de arquivo válido: caso afirmativo, substitui o nome do arquivo em uso por este novo arquivo; caso o arquivo enviado seja *nil*, o programa reposicionará como nome do arquivo em uso o do arquivo de trabalho.

O programa *init* faz as seguintes associações entre os *radio buttons* e programas:

"Itens geograficos" - *sel_geo*
"Selecao Temporal" - *sel_temp*
"Selecao de variaveis" - *sel_con*
"Busca de informacoes" - *recupera*
"Reprocessar ultima consulta" - *rec200a*
"Auxílio" - não implementado

d) Saídas produzidas pelo módulo:

O programa mantém a tela "4.2.1. Menu de funções da aplicação BDE" permanentemente em exibição

e) Funções de apoio ao programa:
Não utiliza.

4.1.2. Programa *sel_geo*

a) Objetivo:
Editar uma lista de itens geográficos para serem utilizados na seleção de informações do banco de dados estatístico.

b) Dados de entrada:
O programa prepara uma lista de itens com base numa lista existente e com informações selecionadas pelo usuário sobre a tela de trabalho.

c) Descrição do processamento:
O programa manuseia a lista de itens geográficos considerando as opções selecionadas pelo usuário sobre o tipo de ordenação na exibição das lista (alfabética pelo nome ou numérica pelo código dos itens) e sobre a estrutura hierárquica desejada: formal ou de regiões metropolitanas. Com a ajuda de botões interativos, o usuário seleciona a função desejada, a saber:

. *DESCE*: forma a lista de seleção múltipla com os objetos hierarquicamente inferiores;

. *DESCE**: forma a lista de seleção múltipla com os objetos hierarquicamente inferiores aos que estejam marcados;

. *DESCE*: preenche a lista de seleção múltipla com os itens do nível hierárquico imediatamente inferior aos do nível em exibição;

. *SOBE*: inverso do botão "desce";

. *SOBE**: inverso do botão "desce*";

. *ABRE ARQUIVO:* busca uma lista previamente armazenada nas instâncias da classe Arquivo associadas à sigla do usuário;

. *SALVA*: grava a lista de itens geográficos existente numa instância da classe Arquivo cujo nome é editado sobre uma lista de arquivos existentes;

. *MARCAR*: inclui todos os itens em exibição na lista de itens geográficos;

. *DESMARCAR*: desmarca todos os itens que estejam em exibição da lista de trabalho;

. *NOVO*: recomeça a formação de uma lista de trabalho a partir de uma lista vazia.

d) Saídas produzidas pelo módulo:

O programa atualiza as instâncias selecionadas da classe Arquivo com listas de itens geográficos e exibe a tela "4.2.2. Seleção de ítens geográficos".

e) Funções de apoio ao programa:

. *geofct01* - monta uma lista de itens de determinado nível geográfico;

. *geofct02* - edita uma lista de itens geográficos para serem exibidos na janela de seleção, montando strings com o código e com o nome dos itens, na ordenação desejada;

. *geofct03* - monta a lista de índices para pré-seleção de uma lista de seleção a partir dos itens existentes na lista de trabalho;

. *geofct04* - atualiza a lista de ítens geográficos com os ítens marcados na lista de seleção;

. *geofct05* - define qual o próximo item de descida a ser utilizado na hierarquia dos itens geográficos;

. *geofct06* - define qual o item superior na hierarquia dos itens geográficos.

. *geofct11* - equivalente à função geofct01 com restrição de manusear apenas os ítens contidos nos já selecionados.

. *arqfct01* - testa a existência de instância de Arquivo para o usuário;

. *arqfct02* - seleção de um nome de Arquivo do usuário;

. *arqfct03* - edição de um nome de arquivo para armazenar uma instância de Arquivo.

4.1.3. Programa *sel_con*

a) Objetivo:

Editar listas de variáveis ou de assuntos para serem utilizados na função de recuperação de informações.

b) Dados de entrada:

O programa manuseia as informações selecionadas pelo usuário sobre a tela de trabalho.

c) Descrição do processamento:

O programa exibe o menu para seleção de variáveis ou de assuntos. Em função da opção do usuário, ativa o programa correspondente. Ao ativar um programa ou receber o controle pelo seu término, o programa *sel_con* utiliza a mesma interface de comunicação do programa *init*. Caso seja utilizado o botão *CANCELA* o programa volta ao programa chamador (*init*) devolvendo um objeto *nil*. O programa efetua a seguinte associação entre botões de seleção e programas:

"variáveis da base de dados" - *contem1*
"descritores da informação" - *contem2*

d) Saídas produzidas pelo módulo:

O programa exibe a tela "4.2.3. Menu para seleção de variáveis ou assunto".

e) Funções de apoio ao programa:

Não utiliza.

4.1.3.1. Módulo *contem1*

a) Objetivo:

Este módulo, desdobramento do programa *sel_con*, tem por finalidade a edição de listas de variáveis para recuperação de informações na base de dados.

b) Dados de entrada:

O programa prepara a lista das variáveis com base, opcionalmente, numa lista existente e com as informações selecionadas pelo usuário sobre a tela de trabalho.

c) Descrição do processamento:

O programa apresenta a lista de variáveis para seleção do usuário composta com as opções marcadas na tela. O usuário pode selecionar pesquisas e atributos de pesquisas para compor a lista de seleção utilizando os dois botões de opções múltiplas da tela: pesquisas e variáveis. Com a ajuda de botões interativos, o usuário seleciona a função desejada, a saber:

. *ABRE ARQUIVO*: busca uma lista previamente armazenada nas instâncias da classe Arquivo associadas à sigla do usuário;

. *SALVA*: grava a lista de variáveis existente numa instância da classe Arquivo cujo nome é editado sobre uma lista de arquivos existentes;

. *MARCAR*: inclui todos os itens em exibição na lista de itens geográficos;

. *DESMARCAR*: desmarca todos os itens que estejam em exibição da lista de trabalho;

. *NOVO*: recomeça a formação de uma lista de trabalho a partir de uma lista vazia;

. *AUXILIO*: exibe a tela de auxílio do contexto.

d) Saídas produzidas pelo módulo:

O programa atualiza instâncias selecionadas da classe Arquivo com as listas de variáveis e exibe a tela "4.2.4. Tela para seleção de variáveis".

e) Funções de apoio ao programa:

. *ajdfct10* - edita uma lista de pesquisas e de suas propriedades (atributos e métodos);

. *confct04* - atualiza a lista de variáveis;

. *arqfct01* - testa a existência de instância de Arquivo para o usuário;

. *arqfct02* - seleção de um nome de Arquivo do usuário;

. *arqfct03* - edição de um nome de arquivo para armazenar uma instância de Arquivo.

4.1.3.2. Módulo *contem2*

a) Objetivo:

Este outro módulo, também desdobrado do programa *sel_con*, tem por finalidade a edição de listas de assuntos, ou seja de temas , classificações e descritores de informações para recuperação de variáveis na base de dados.

b) Dados de entrada:

O programa prepara a lista de assuntos com base numa lista existente e com as informações selecionadas pelo usuário sobre a tela de trabalho.

c) Descrição do processamento:

O programa apresenta a lista de variáveis para seleção do usuário composta de acordo com as opções marcadas na tela. O usuário pode selecionar temas, classificações e objetos para compor a lista de seleção utilizando os três botões de opções múltiplas da tela de mesma designação. Com a ajuda de botões interativos, o usuário seleciona a função desejada, a saber:

. *DESCE*: forma a lista de seleção múltipla com os objetos hierarquicamente inferiores;

. *DESCE**: forma a lista de seleção múltipla com os objetos hierarquicamente inferiores aos que estejam marcados;

. *SOBE*: inverso do botão "desce";

. *SOBE**: inverso do botão "desce*";

. *ABRE ARQUIVO*: busca uma lista previamente armazenada nas instâncias da classe Arquivo associadas à sigla do usuário;

. *SALVA*: grava a lista de variáveis existente numa instância da classe Arquivo cujo nome é editado sobre uma lista de arquivos existentes;

. *MARCAR*: para incluir na lista de itens geográficos todos os itens em exibição;

. *DESMARCAR*: desmarcar da lista de trabalho todos os itens que estejam em exibição;

. *NOVO*: recomeça a formação de uma lista de trabalho a partir de uma lista vazia.

. *AUXILIO*: exibe a tela de auxílio do contexto;

. *DESCRITORES*: ativa o programa *contem21* para seleção de conceitos;

. *LISTA*: ativa o programa *contem22* para exibir todos os detalhes dos descritores marcados.

d) Saídas produzidas pelo módulo:

O programa atualiza instâncias selecionadas da classe Arquivo com as listas de variáveis e exibe a tela "4.2.5. Tela para seleção de assuntos por temas".

e) Funções de apoio ao programa:

. *confct05* - atualiza a lista de assuntos;

. *confct06* - edita uma lista com todos os objetos contidos numa classificação;

. *confct07* - edita uma lista das classificações de um conjunto de descritores;

. *arqfct01* - testa a existência de instância de Arquivo para o usuário;

. *arqfct02* - seleção de um nome de Arquivo do usuário;

. *arqfct03* - edição de um nome de arquivo para armazenar uma instância de Arquivo.

4.1.3.3. Módulo *contem21*

a) Objetivo:

Este módulo detalha o programa *contem2* e tem por finalidade a edição de listas de descritores existente na lista de assuntos em manuseio.

b) Dados de entrada:

O programa prepara a lista de descritores com base nas informações existentes na lista de assuntos existentes na instância da classe Arquivo em uso.

c) Descrição do processamento:

O programa apresenta a lista de descritores para seleção do usuário. Com a ajuda de um botão interativo, o usuário seleciona um destes cuja seleção de conceitos será efetuada pelo programa *contem211*, para quem será passado o controle de execução.

d) Saídas produzidas pelo módulo:

O programa exibe a tela "4.2.6. Tela para seleção do descritor".

e) Funções de apoio ao programa:

Não utiliza.

4.1.3.4. Módulo *contem211*

a) Objetivo:

Este programa tem por finalidade a edição de listas de conceitos de pesquisa associados a um descritor.

b) Dados de entrada:

O programa prepara a lista de conceitos de investigação de um descritor com base nas instâncias das classes *Descritor* e *Ocorrencias*.

c) Descrição do processamento:

O programa prepara para o descritor informado a lista de todos os conceitos de pesquisa utilizados na indexação das variáveis, conforme as ocorrências das pesquisas associadas ao descritor. O usuário, com a ajuda de botões interativos, seleciona a função desejada, a saber:

. *MARCAR*: marca todos os conceitos em exibição;

. *DESMARCAR*: desmarca todos os conceitos associados ao descritor;

. *AUXILIO*: exibe a tela de auxílio do contexto;

. *LISTA*: ativa o programa *contem22* para exibir todos os detalhes dos descritores marcados.

d) Saídas produzidas pelo módulo:

O programa atualiza na instância selecionada da classe Arquivo a lista de assuntos e exibe a tela "4.2.7. Seleção dos conceitos de pesquisa".

e) Funções de apoio ao programa:

. *confct08* - edita uma lista com todos os conceitos associados a descritor;

4.1.3.5. Módulo *contem22*

a) Objetivo:

Exibir a lista de descritores para detalhamento.

b) Dados de entrada:

O programa utiliza a lista de descritores existente no arquivo em uso.

c) Descrição do processamento:

Com base no arquivo em uso o programa exibe a lista dos descritores selecionados. Com a ajuda de um botão interativo, o programa ativa o programa *contem221*.

d) Saídas produzidas pelo módulo:

O programa manuseia a tela "4.2.6. Tela para seleção do descritor".

e) Funções de apoio ao programa:

Não utiliza.

4.1.3.6. Módulo *contem221*

a) Objetivo:

Exibir o detalhamento de um descritor na base de dados.

b) Dados de entrada:

Recebe do programa chamador o descritor a ser detalhado.

c) Descrição do processamento:

Com base no descritor informado pelo programa chamador, o programa elabora a relação de classificações as quais o descritor esteja subordinado, a relação dos objetos compreendidos em sua descrição, a relação dos termos equivalentes registrados na classe Descritor, a relação das variáveis que indexa e dos conceitos de investigação a que está associado. Com a ajuda de um botão interativo, o programa exibe a tela de texto com o conceito definido para o descritor.

d) Saídas produzidas pelo módulo:

O programa apresenta a tela "4.2.8. Tela para detalhamento de um descritor" e uma segunda tela do padrão do O2 para exibição de textos (modelo "4.2.18. Tela para auxílio").

e) Funções de apoio ao programa:

Não utiliza.

4.1.4. Programa *sel_temp*

a) Objetivo:

Editar a definição do corte temporal a ser utilizado na seleção de informações do banco de dados estatístico.

b) Dados de entrada:

O programa prepara a definição do corte temporal com base numa lista existente e com as informações selecionadas pelo usuário sobre as telas de trabalho.

c) Descrição do processamento:

O programa registra no arquivo em uso a seleção do usuário da opção de época apresentada por radios buttons na tela. Caso escolha a última opção, dados de determinado período, deve informar as datas de início e término pretendidas. Estas

datas poderão ser informadas de forma abreviada, informando apenas o ano. O programa , com a ajuda de botões interativos, pode executar ainda as funções:

. *ABRE ARQUIVO*: busca uma lista previamente armazenada nas instâncias da classe Arquivo associadas à sigla do usuário;

. *SALVA*: grava a lista de itens geográficos existente numa instância da classe Arquivo cujo nome é editado sobre uma lista de arquivos existentes;

. *AUXILIO*: exibe a tela de auxílio do contexto;

d) Saídas produzidas pelo módulo:

O programa atualiza a instância selecionada da classe Arquivo com a definição da época pretendida para os dados (opção de corte temporal) e exibe a tela "4.2.9. Tela para definição do corte temporal".

e) Funções de apoio ao programa:

. *temfct01* - edição de data para o formato de tupla numérica;

. *arqfct01* - testa a existência de instância de Arquivo para o usuário;

. *arqfct02* - seleção de um nome de Arquivo do usuário;

. *arqfct03* - edição de um nome de arquivo para armazenar uma instância de Arquivo.

4.1.5. Programa *recupera*

a) Objetivo:

Este programa tem por objetivo efetuar o controle da função de preparação da recuperação de informações.

b) Dados de entrada:

O programa utiliza a opção de uma tela de menu para definir o programa a ser ativado.

c) Descrição do processamento:

Conforme a opção do usuário, o programa ativará a função de definição do corte espacial (geográfico) ou a de seleção de variáveis na base de dados. O programa prepara um objeto na classe *Selecao* para a receber a indicação dos atributos referenciados nas listas. O programa utiliza também três botões interativos para as funções:

. *ABRE ARQUIVO*: busca uma instância da classe Arquivo associada à sigla do usuário;

. *NOVO*: recomeça a formação de uma lista de trabalho a partir de uma nova instância, vazia;

. *AUXILIO*: exibe a tela de auxílio do contexto;

d) Saídas produzidas pelo módulo:

O programa atualiza uma instância na classe Seleção e exibe a tela "4.2.10. Menu da função de recuperação de informações".

e) Funções de apoio ao programa:

. arqfct01 - testa a existência de instância de Arquivo para o usuário;

. arqfct02 - seleção de um nome de Arquivo do usuário;

4.1.5.1. Módulo *rec1000*

a) Objetivo:

Este programa tem por objetivo implementar uma alternativa de definição de itens geográficos, de forma rápida ou através da função *sel_geo*, de forma detalhada.

b) Dados de entrada:

O programa utiliza as opções da tela e o conteúdo do cadastro geográfico.

c) Descrição do processamento:

Conforme a opção do usuário, o programa pode construir a lista de ítens geográficos a ser utilizada na seleção com uma lista previamente elaborada e armazenada no arquivo em uso ou por uma nova lista composta com os ítens dos níveis hierárqui-

cos selecionados nas opções do programa exibidas numa segunda tela. O programa utiliza diversos botões interativos para as seguintes funções:

. *ABRE ARQUIVO:* busca uma instância da classe Arquivo associada à sigla do usuário;

. *SALVA*: grava a lista de itens geográficos existente numa instância da classe Arquivo cujo nome é editado sobre uma lista de arquivos existentes;

. *OPCOES*: exibe uma segunda tela com as opções de níveis hierárquicos da estrutura político-administrativa;

. *ITENS*: ativa a função *sel_geo* para a definição detalhada de uma lista de itens geográficos;

. *AUXILIO*: exibe a tela de auxílio do contexto;

d) Saídas produzidas pelo módulo:

O programa atualiza a classe Seleção e exibe as telas "4.2.11. Tela para definição de itens geográficos" e "4.2.12. Tela para definição de níveis geográficos".

e) Funções de apoio ao programa:

. sel_geo - função de edição de listas de itens geográficos;

. *arqfct01* - testa a existência de instância de *Arquivo* para o usuário;

. *arqfct02* - seleção de um nome de *Arquivo* do usuário;

. *arqfct03* - edição de um nome de arquivo para armazenar uma instância de *Arquivo*.

4.1.5.2. Módulo *rec2000*

a) Objetivo:

Este programa tem por objetivo converter as listas de variáveis e assuntos em lista de propriedades dos objetos estatísticos para proceder a extração dos dados da base.

b) Dados de entrada:

O programa utiliza as listas de variáveis e de assuntos existentes nas instâncias da classe Arquivo selecionadas pelo usuário.

c) Descrição do processamento:

O programa irá converter as variáveis e os assuntos existentes nas listas dos arquivos do usuário em propriedades dos objetos da base de dados através da identificação das propriedades correspondentes às definições dadas nas listas que atendam aos requisitos temporais correspondentes. Quando um arquivo é aberto, sua lista é convertida e acrescentada na lista final de trabalho mantida em objeto da classe *Selecao*. Com o auxílio de botões interativos, o usuário pode executar as seguintes funções:

. *ABRE ARQUIVO*: busca uma instância da classe Arquivo associada à sigla do usuário;

. *VARIAVEL*: para definição das características das variáveis no tipo resultante da seleção;

. *FILTRO*: para visualização ou definição de um filtro de conteúdo;

. *AUXILIO*: exibe a tela de auxílio do contexto;

. *RECUPERACAO DE INFORMACOES*: executa a recuperação de informações propriamente dita.

d) Saídas produzidas pelo módulo:

O programa atualiza uma instância na classe Seleção e exibe a tela "4.2.13. Definição de variáveis para a recuperação de valores".

e) Funções de apoio ao programa:

. *recfct01* - converte uma lista de variáveis para propriedades de objetos;

. *recfct02* - preparação do objeto *Selecao* para uma nova consulta;

. *arqfct01* - testa a existência de instância de Arquivo para o usuário;

. *arqfct02* - seleção de um nome de Arquivo do usuário;

4.1.5.3. Módulo *rec2100*

a) objetivo:
Edição da relação de filtros de conteúdo.

b) Dados de entrada:
O programa utiliza as instâncias das classes *Filtro* e *Selecao*.

c) Descrição do processamento:
O programa exibe a lista de filtros de conteúdo para o usuário, através de botões interativos, optando entre acrescentar um novo ou listar um existente. Quando é pressionado o botão *OK*, o programa ativa o *rec2110* para a definição de novos filtros. Os botões permitem as seguintes funções:

. *LISTAR*: mostrar toda a definição de um filtro;

. *AUXILIO*: exibe a tela de auxílio do contexto.

d) Saídas produzidas pelo módulo:
O programa exibe a tela "4.2.14. Tela para definição de filtros".

e) Funções de apoio ao programa:
Não utiliza.

4.1.5.4. Módulo *rec2110*

a) objetivo:
Edição de filtros de conteúdo.

b) Dados de entrada:
O programa utiliza as instâncias das classes *Filtro* e *Selecao*.

c) Descrição do processamento:
O programa exibe uma tela para especificação de um filtro. Sobre esta tela o usuário marca ou acrescenta as opções para a construção do filtro e, caso deseje um

filtro composto, do tipo "ou nova condição", uma nova tela é aberta para a continuação.

d) Saídas produzidas pelo módulo:

O programa atualiza a classe *Filtro* e exibe a tela "4.2.15. Tela para inclusão de novo filtro".

e) Funções de apoio ao programa:

Não utiliza.

4.1.5.5. Módulo *rec3000*

a) objetivo:

Edição do tipo e do programa em O_2C para a construção do resultado.

b) Dados de entrada:

O programa utiliza as instâncias das classes *Filtro*, *Selecao* e *Ocorrencias*.

c) Descrição do processamento:

O programa, com base nos objetos das classes *Filtro* e *Selecao*, edita dois arquivos *unix*: um com os comandos O_2 para a criação de uma nova classe, a classe *R_codigo_do_usuario*, destinada a receber o resultado do processo de busca de informações; e outro para o programa em O_2C que irá preencher as instâncias desta nova classe com os valores dos objetos estatísticos. Quando os novos programas são concluídos, o programa comanda a atualização do esquema e a compilação do novo programa, o "programa do usuário" emitindo então um commit para liberar a memória e permitir a execução deste novo programa.

d) Saídas produzidas pelo módulo:

O programa atualiza a classe *Filtro* e exibe a tela "4.2.15. Tela para inclusão de novo filtro".

e) Funções de apoio ao programa:

Não utiliza.

4.1.6. Programa *rec200a*

a) objetivo:
Reprocessar uma consulta.

b) Dados de entrada:
O programa utiliza a infância da classe *Selecao*.

c) Descrição do processamento:
O programa mantém o último objeto da classe *Selecao* construído para o usuário para chamar o programa *rec3000* de recuperação de informações. Caso não exista um objeto da classe *Selecao* associado ao usuário o controle retorna ao programa *init*.

d) Saídas produzidas pelo módulo:
O programa não produz saídas.

e) Funções de apoio ao programa:
Não utiliza.

4.1.7. Programa do usuário

a) objetivo:
Recuperar os valores solicitados pelo usuário na base de dados.

b) Dados de entrada:
O programa utiliza os objetos estatísticos, da classe Dados.

c) Descrição do processamento:
O programa, construído segundo as propriedades e filtros especificados pelo usuário, inicia montando um "*named set*" com uma instância da classe do usuário para cada item geográfico da lista do arquivo de trabalho. Depois, seleciona em *DADOS* as instâncias que pertencem a uma das classes selecionadas. Para cada instância é verificado se o código geográfico pertence à lista de itens em uso. Caso não seja, a instância é abandonada. A seguir, é verificada a classe da instância para adicionar aos objetos do usuário os valores dos atributos recuperados. Ao final, as instâncias do *set* do usuário são submetidas aos filtros com a eliminação das que não atenderem aos seus requisitos e é emitido um commit para ativar o programa *restart*.

d) Saídas produzidas pelo módulo:

O programa cria o "*named set*" para as instâncias selecionadas dos objetos do usuário.

e) Funções de apoio ao programa:

Não utiliza.

4.1.8. Programa rec3100

a) objetivo:

Exibir os objetos resultantes de uma consulta.

b) Dados de entrada:

O programa utiliza o *set* do usuário.

c) Descrição do processamento:

O programa utiliza o O_2SQL para exibir tuplas formadas pelo código ou nome do item geográfico, de acordo com a opção registrada em *Selecao*, e pelo identificador do objeto do usuário. Ao término, o programa devolve o controle ao programa *init*.

d) Saídas produzidas pelo módulo:

O programa exibe a tela "4.2.16. Menu dos objetos recuperados na base de dados" e, se o usuário utilizar o *edit* ou *display* de um objeto, é exibida a tela "4.2.17. Tela para apresentação de um objeto recuperado".

e) Funções de apoio ao programa:

Não utiliza.

4.1.9. Programa exit

a) Objetivo:

Finalizar uma sessão da aplicação BDE.

b) Dados de entrada:

Não utiliza.

c) Descrição do processamento:
O programa encerra a aplicação.

d) Saídas produzidas pelo módulo:
Não aplicável.

e) Funções de apoio ao programa:
Não utiliza.

4.1.10. Programa *restart*

a) Objetivo:
Controlar a emissão de commit's.

b) Dados de entrada:
O programa recebe através de uma variável global a identificação do programa que emitiu o commit.

c) Descrição do processamento:
O programa recebe através de uma variável global o código daquele que emitiu o commit e ativa o programa seguinte do fluxo, na seguinte ordem: *rec3000* - recdados - rec3100.

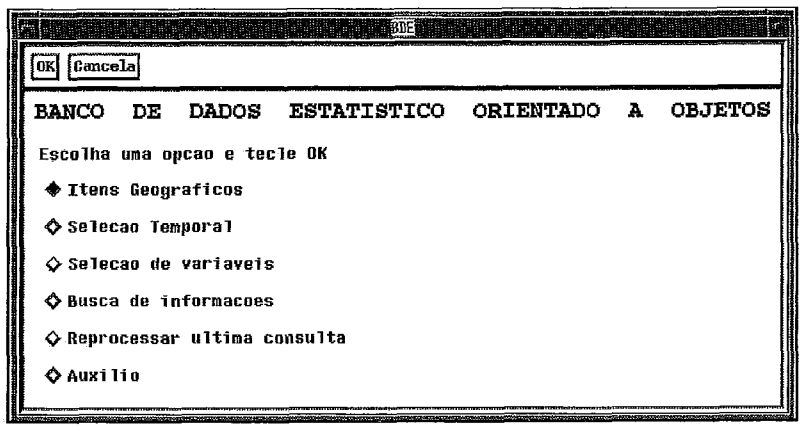
d) Saídas produzidas pelo módulo:
Não aplicável.

e) Funções de apoio ao programa:
Não utiliza.

4.2. Definição das Telas

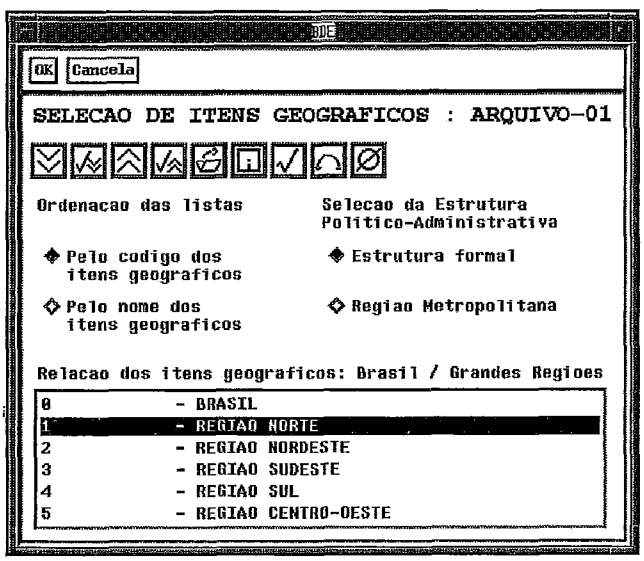
4.2.1. Menu de funções da aplicação BDE

. Programa: *init*



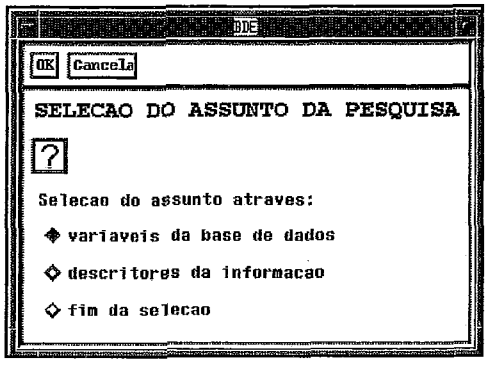
4.2.2. Seleção de ítens geográficos

. Programa: *sel_geo*



4.2.3. Menu para seleção de variáveis ou assunto

. Programa: *sel_con*

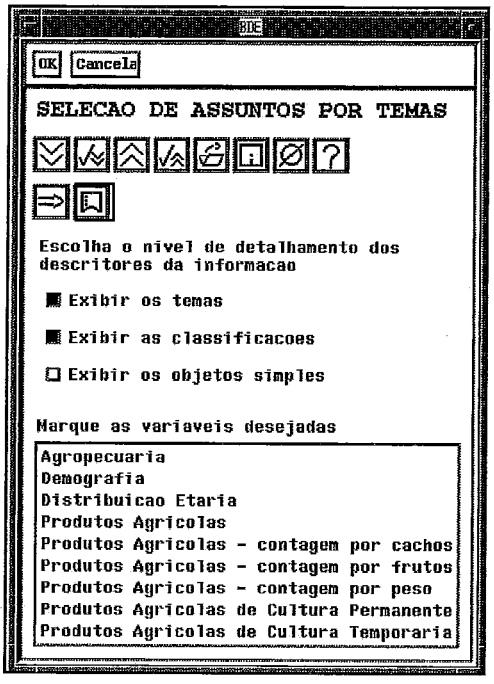


4.2.4. Tela para seleção de variáveis

. Programa: *contem1*

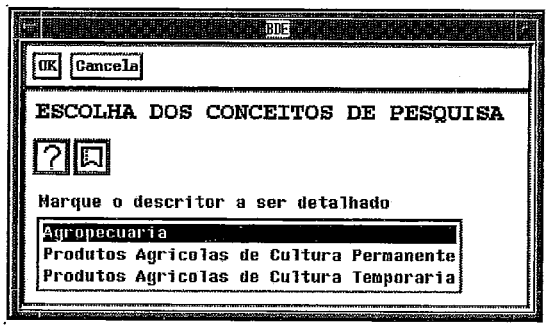
4.2.5. Tela para seleção de assuntos por temas

. Programa: *contem2*



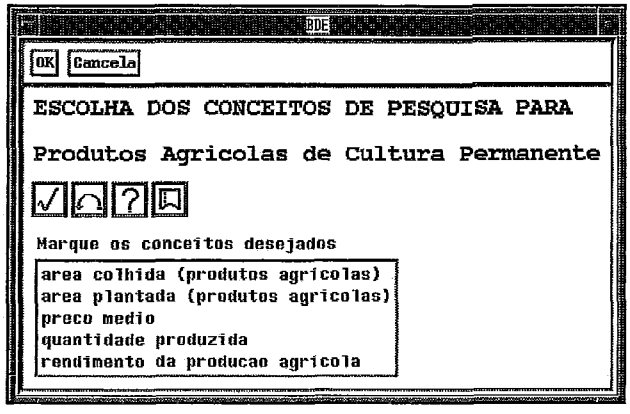
4.2.6. Tela para seleção do descritor

. Programas: *contem21* e *contem22*



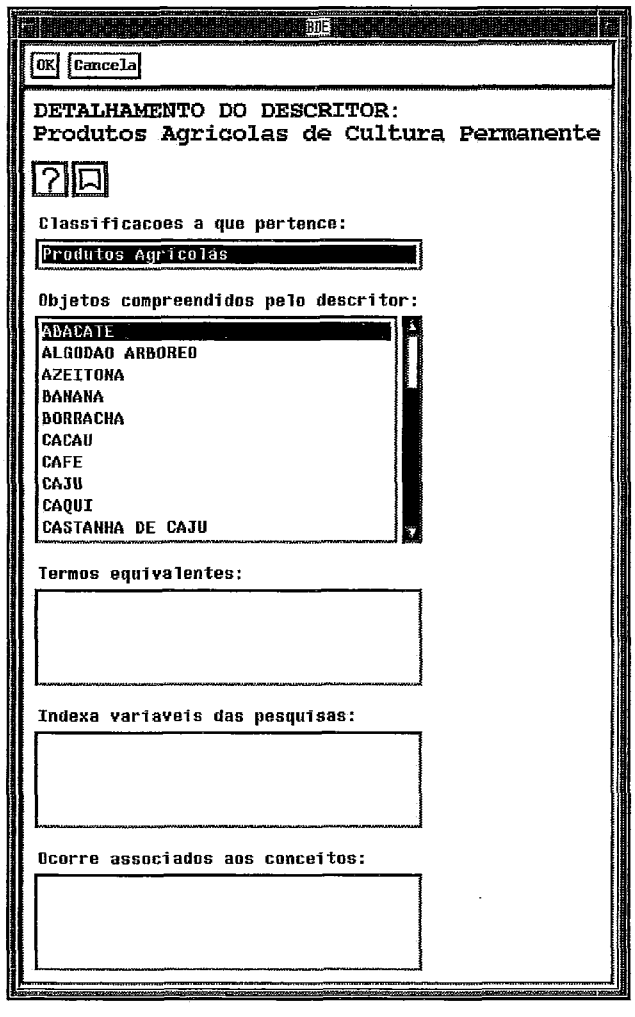
4.2.7. Seleção dos conceitos de pesquisa

. Programa: *contem211*



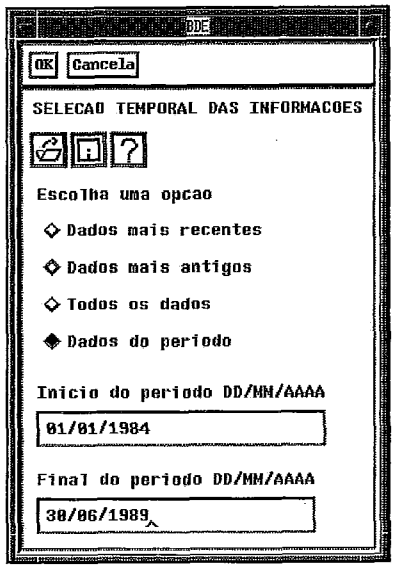
4.2.8. Tela para detalhamento de um descritor

. Programa: *contem221*



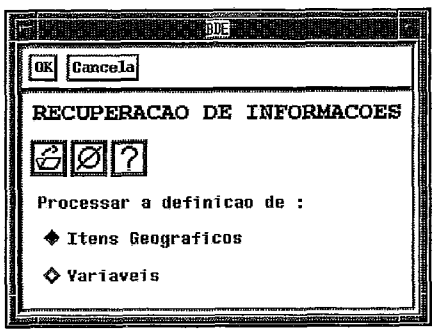
4.2.9. Tela para definição do corte temporal

. Programa: *sel_temp*



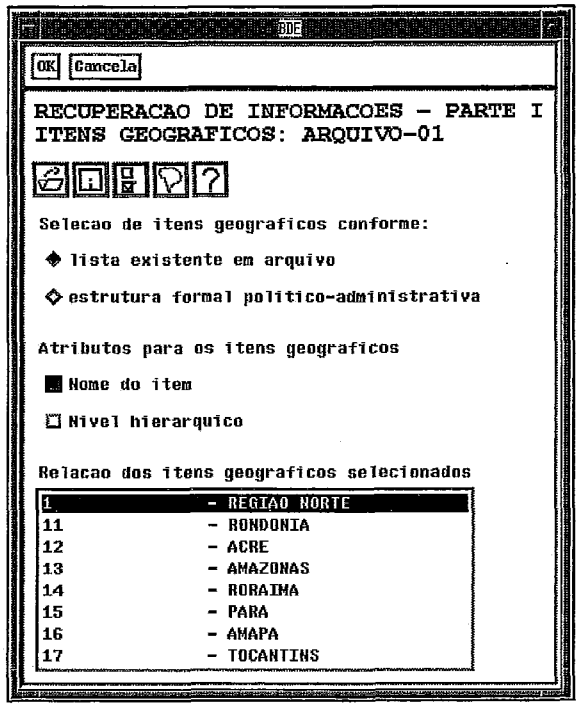
4.2.10. Menu da função de recuperação de informações

. Programa: *recupera*



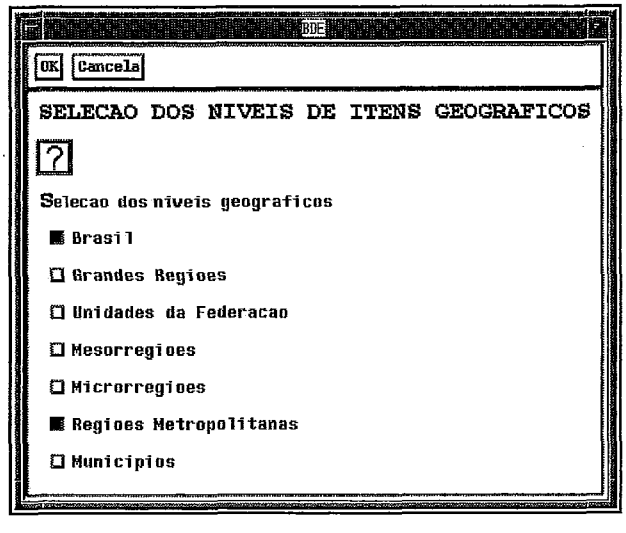
4.2.11. Tela para definição de itens geográficos

. Programa: *rec1000*



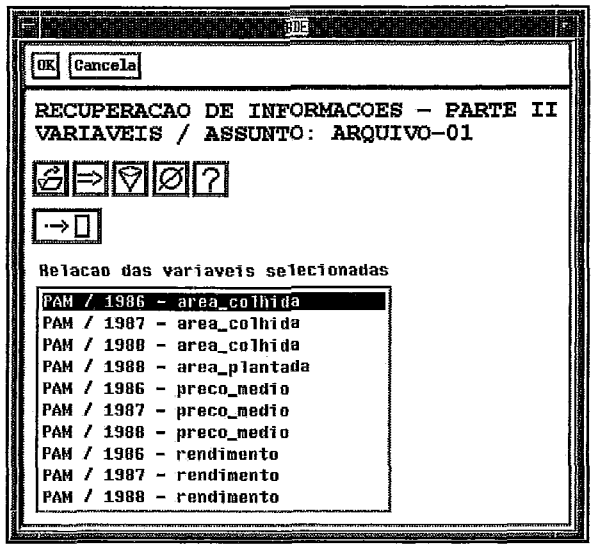
4.2.12. Tela para definição de níveis geográficos

. Programa: *rec1000*



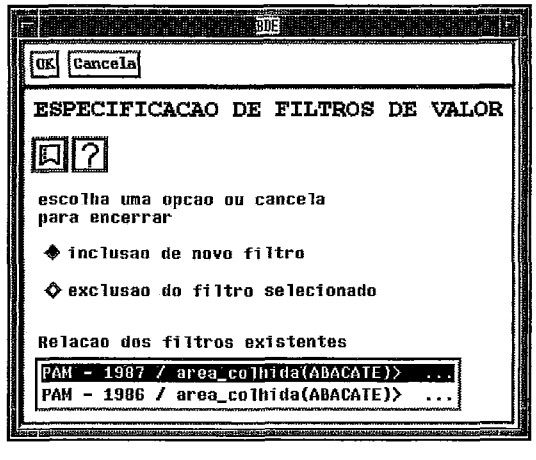
4.2.13. Definição de variáveis para a recuperação de valores

. Programa: *rec2000*



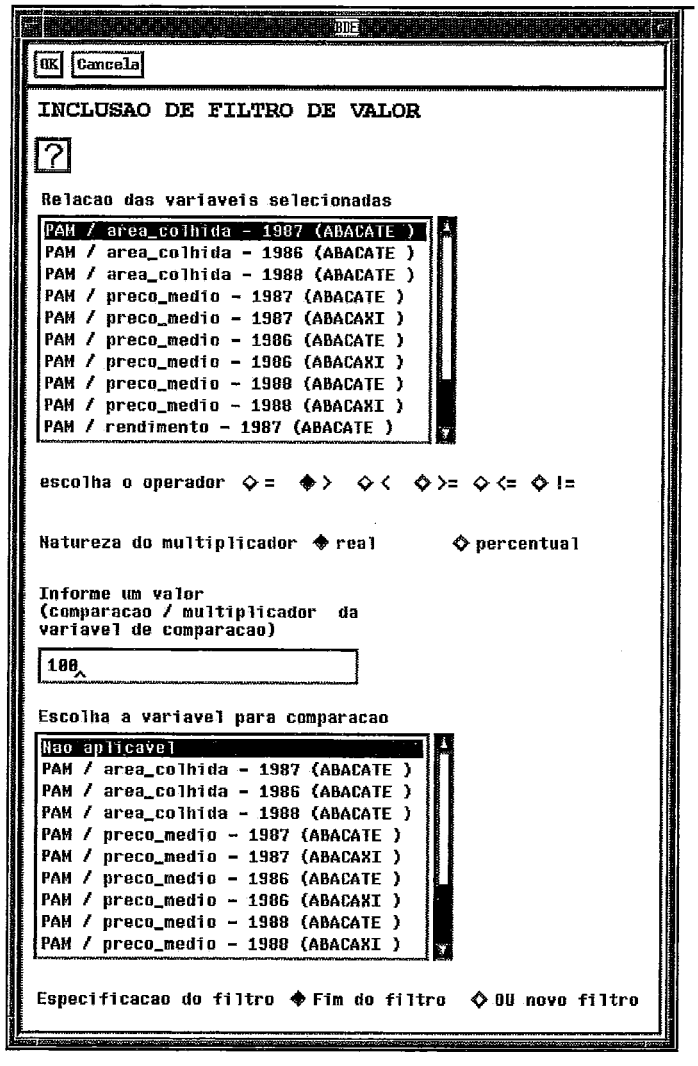
4.2.14. Tela para definição de filtros

. Programa: *rec2100*



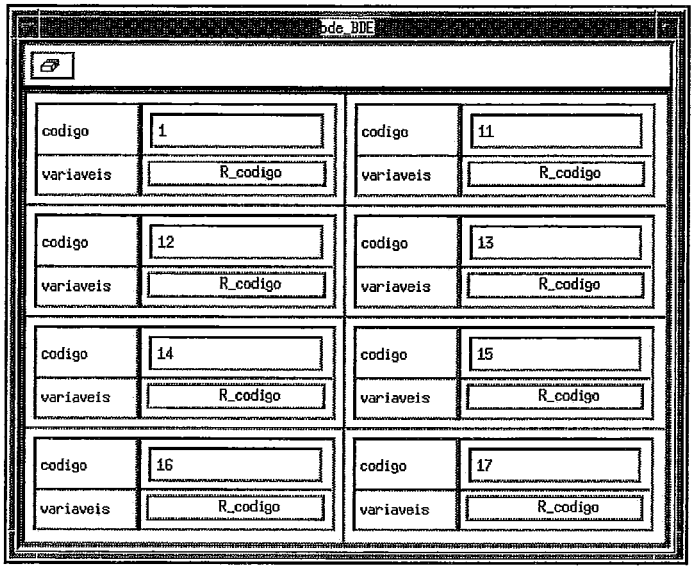
4.2.15. Tela para inclusão de novo filtro

. Programa: *rec2110*



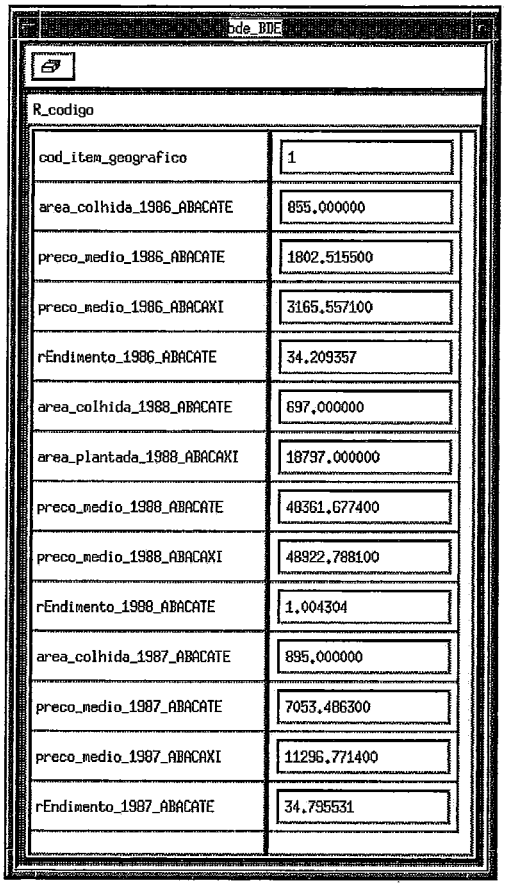
4.2.16. Menu dos objetos recuperados na base de dados

. Programa: *rec3100*



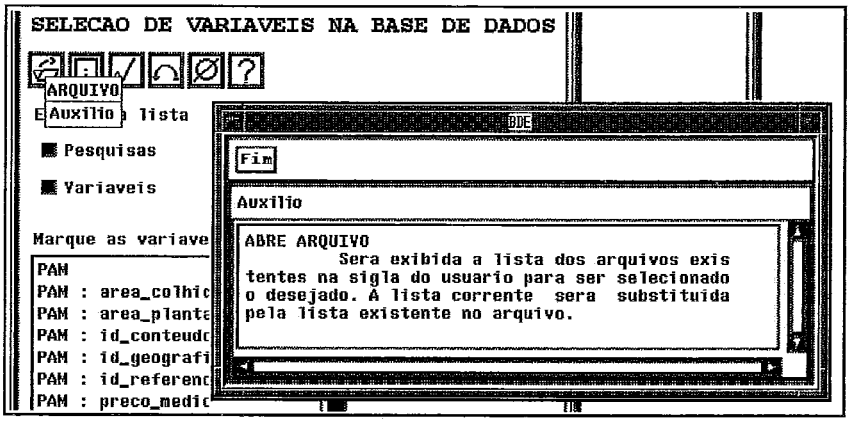
4.2.17. Tela para apresentação de um objeto recuperado

. Programa: *rec3100*



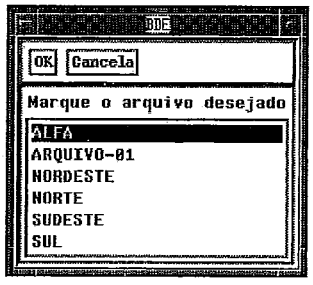
4.2.18. Tela para auxílio

. Programa: método *ajuda* da classe *Botao*



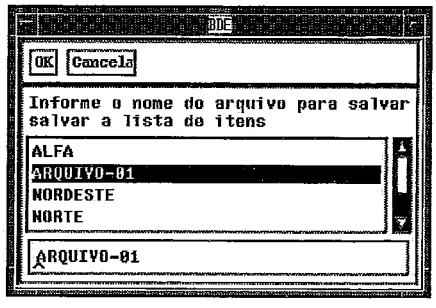
4.2.19. Tela para abrir um arquivo de listas

. Programa: *arqfct02*



4.2.20. Tela para salvar uma lista num arquivo

. Programa: *arqfct03*



5. APLICAÇÃO METADADO

A aplicação METADADO compreende as ferramentas de manutenção dos metadados do BDE distribuídas em três grandes funções: manutenção de pesquisas, manutenção de descritores e indexação da base. Com as duas primeiras são mantidos os metadados necessários ao processo de indexação que é executado pela terceira destas funções.

A estrutura utilizada para implementar a aplicação metadado pode ser visualizada nos diagramas de módulos constantes da figura 15.

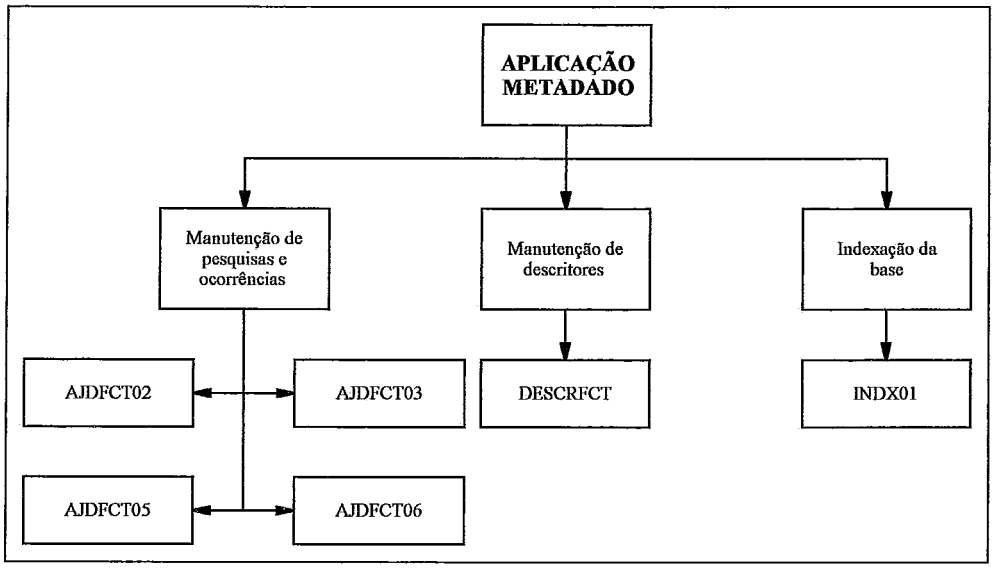


Figura 15 - Diagrama da aplicação METADADO

5.1. Detalhamento dos Módulos

5.1.1. Manutenção de pesquisas (*pesquisa*)

a) Objetivo:

O programa *pesquisa* tem como finalidade a manutenção de características das pesquisas e de suas ocorrências.

b) Dados de entrada:

O programa dispõe de diversas telas para a edição de informações.

c) Descrição do processamento:

O programa dispõe de um único menu do tipo *radio button* para as funções de inclusão, alteração e exclusão de pesquisas e ocorrências, além de um sétimo botão para a função de auxílio.

c.1) Pesquisas

Quando uma pesquisa é incluída, são criadas instâncias: uma na classe *Pesquisa* e outra na subclasse *Descricao_da_pesquisa*. A função de alteração apresenta uma lista para auxiliar a seleção da pesquisa desejada antes de exibir os atributos públicos da classe *Descricao_da_pesquisa* para serem editados. A função de exclusão também apresenta a lista de seleção para, a seguir, pedir ao usuário a confirmação de intenção e, além de excluir as instâncias das classes *Pesquisa* e *Descri-*

cao_da_pesquisa, exclui todas as referências existentes nas classes *Ocorrencias*, *Descritor* e *Conceito*.

c.2) Ocorrências de pesquisas

As funções de manutenção das ocorrências iniciam com a apresentação de uma lista com as pesquisas existentes. Para a função de inclusão é apresentada a lista das subclasses da classe Dados definidas para a pesquisa de forma a ser informado o tipo do novo objeto. A inclusão é feita com a criação de uma instância na classe *Ocorrencias* onde é incluída a lista de propriedades da subclasse de *Dados*, complementada os atributos de caracterização da nova ocorrência. A alteração é feita naquela que for selecionada entre as apresentadas em uma lista de seleção de ocorrências com a edição de uma tela com todos os atributos públicos do objeto. A exclusão usa uma lista semelhante a da alteração e exige que o usuário confirme sua intenção. Como na exclusão de pesquisas, a exclusão de ocorrência implica na eliminação das referências existentes nas classes *Descricao_da_pesquisa*, *Descritor* e *Conceito*.

d) Saídas produzidas pelo módulo:

O programa atualiza os *named sets PESQUISAS, DESCRITOR* e *CONCEITOS* e utiliza as seguintes telas:

5.2.2. Menu de manutenção de pesquisas / ocorrências
5.2.3. Tela para inclusão de pesquisas
5.2.4. Tela para inclusão /alteração de pesquisas
5.2.5. Relação de pesquisas
5.2.6. Manutenção de textos de pesquisas
5.2.7. Confirmação de exclusão de pesquisas
5.2.8. Relação de classes de uma pesquisa
5.2.9. Detalhamento temporal de pesquisas
5.2.10. Manutenção de dados de uma pesquisa
5.2.11. Relação de ocorrências de uma pesquisa
5.2.12. Confirmação de exclusão de ocorrência

e) Funções utilizadas pelo programa:

. *ajdfct02* - seleção de uma pesquisa;

. *ajdfct03* - seleção de uma subclasse de *Dados* para a pesquisa;

. *ajdfct05* - seleção de uma ocorrência de pesquisa;

. *ajdfct06* - definição de um período de referência para uma ocorrência.

5.1.2. Manutenção de descritores (*descritor*)

a) Objetivo:

O programa *descritor* destina-se à manutenção dos descritores da informação e de suas classificações.

b) Dados de entrada:

O programa dispõe de diversas telas para a edição de informações.

c) Descrição do processamento:

O programa dispõe de um menu do tipo *radio button* para as funções de inclusão, alteração e exclusão de descritores e para a exclusão de uma classificação que é apresentado junto com uma lista editável dos descritores existentes. A inclusão é feita com a criação de uma nova instância da classe *Descritor,* cuja identificação foi escolhida no menu, editada numa tela onde a lista dos descritores existentes é apresentada duas vezes na modalidade de seleção múltipla: uma para definir as classificações nas quais o descritor participa e outra para os objetos que compreende como classificador. A alteração permite incluir e excluir o descritor de classificações existentes e efetuar a manutenção do conjunto dos objetos que classifica. A exclusão é confirmada antes de ser executada e elimina toda a referência ao objeto nas suas classificações e objetos.

d) Saídas produzidas pelo módulo:

O programa atualiza o *named set* DESCRITOR e utiliza as seguintes telas:

5.2.13. Menu para manutenção de descritores
5.2.14. Inclusão de descritores
5.2.15. Alteração de descritores
5.2.16. Confirmação de exclusão de descritor

e) Funções utilizadas pelo programa:

O programa utiliza a função *descrfct* que compõe, recursivamente, a lista dos descritores compreendidos numa classificação.

5.1.3. Indexação da base (*indexa*)

a) Objetivo:

O programa *indexa* tem por objetivo associar os metadados com os objetos da base de dados.

b) Dados de entrada:

O programa dispõe de diversas telas para a edição de informações.

c) Descrição do processamento:

O programa apresenta um menu para acesso à função de indexação oferecendo as alternativas de indexar todos as propriedades de uma ocorrência de pesquisa, de limitar o processo à aquelas propriedades ainda não indexadas ou de copiar a indexação efetuada na ocorrência de data imediatamente à anterior, nas propriedades de mesmo nome. A indexação será efetuada relacionando cada propriedade da ocorrência existente no objeto de sua classe selecionado com um objeto da classe *Descritor*, da classe *Conceito* e da classe *Unidade_medida*. Uma nova instância é incluída quando o conceito de investigação ou a unidade de medida não existam nos seus *named sets*.

d) Saídas produzidas pelo módulo:

O programa atualiza os *named sets PESQUISAS*, DESCRITOR, CONCEITOS e UNIDADES. O programa também apresenta as seguintes telas:

5.2.17. Menu de opções para indexação de informações

5.2.18. Relação das variáveis para indexação

5.2.19. Indexação da informação

e) Funções utilizadas pelo programa:

O programa utiliza a função *indx01* que edita a lista de todos os descritores compreendidos numa classificação.

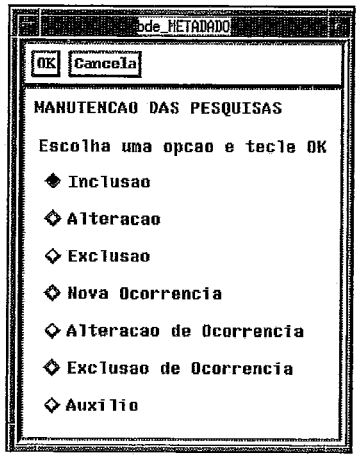
5.2. Definição das Telas

5.2.1. Tela da aplicação METADADO

. Programa: Tela gerada pelo comando *run* do O_2

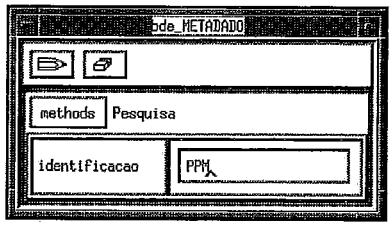
5.2.2. Menu de manutenção de pesquisas / ocorrências

. Programa: *pesquisa*



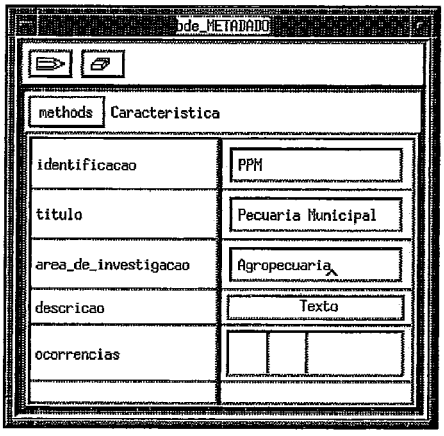
5.2.3. Tela para inclusão de pesquisas

. Programa: *pesquisa*



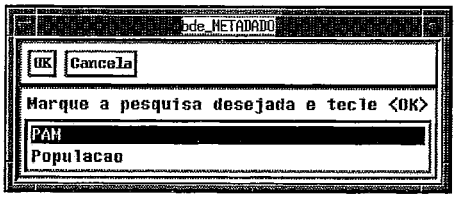
5.2.4. Tela para inclusão /alteração de pesquisas

. Programa: *pesquisa*



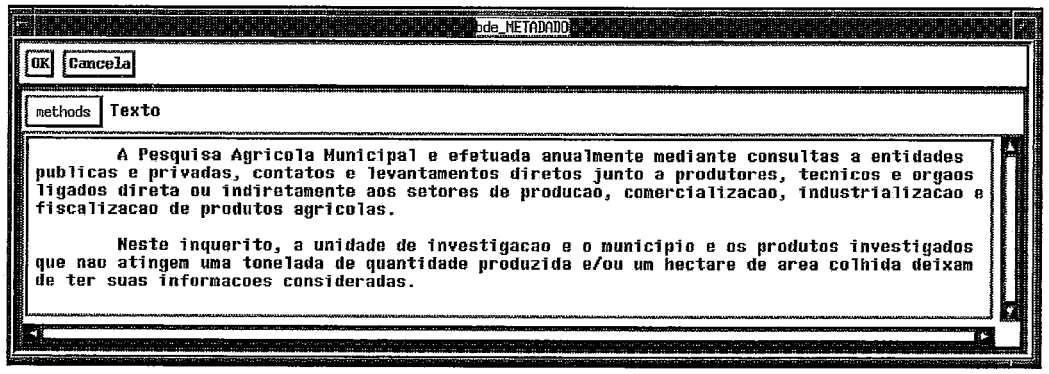
5.2.5. Relação de pesquisas

. Programa: *pesquisa*



5.2.6. Manutenção de textos de pesquisas

. Programa: *pesquisa*

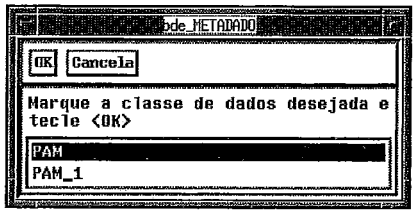


5.2.7. Confirmação de exclusão de pesquisas

. Programa: *pesquisa*

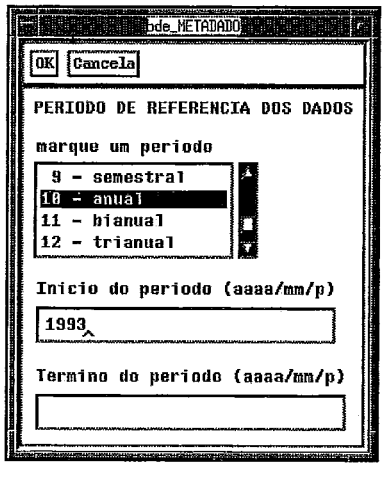
5.2.8. Relação de classes de uma pesquisa

. Programa: *pesquisa*



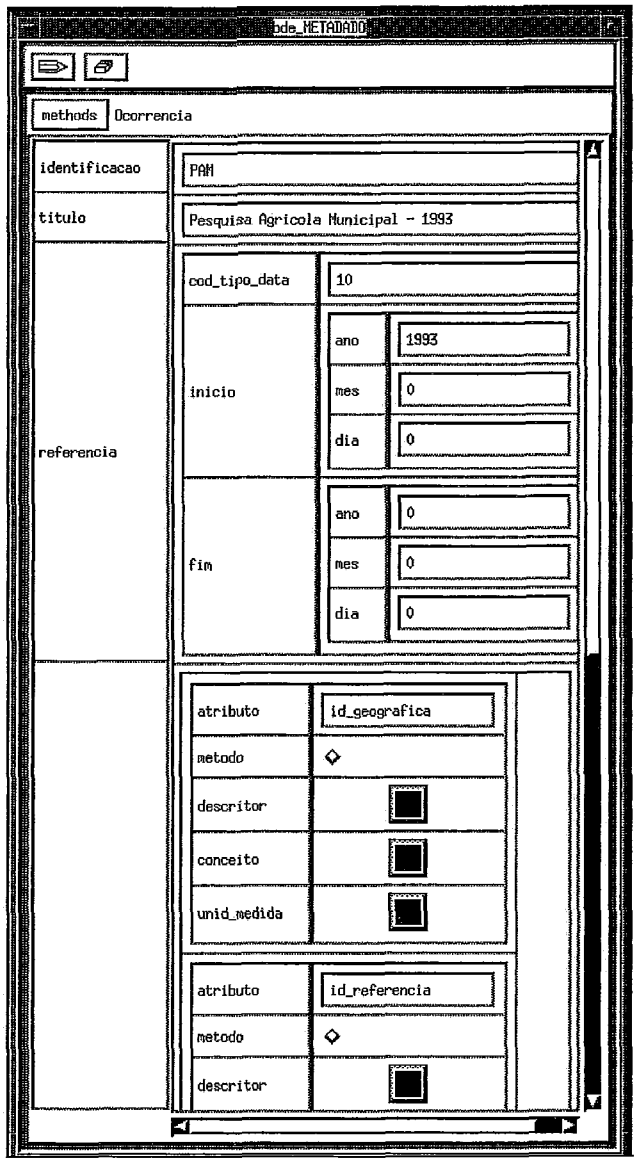
5.2.9. Detalhamento temporal de pesquisas

. Programa: *pesquisa*



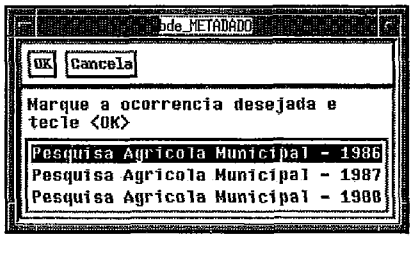
5.2.10. Manutenção de dados de uma pesquisa

. Programa: *pesquisa*



5.2.11. Relação de ocorrências de uma pesquisa

. Programa: *pesquisa*

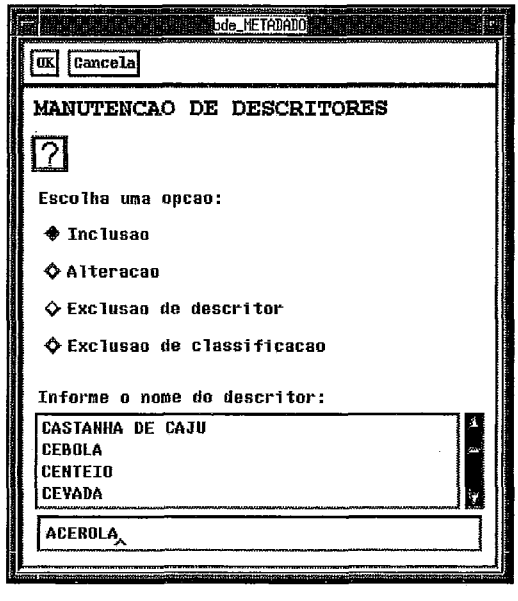


5.2.12. Confirmação de exclusão de ocorrência

. Programa: *pesquisa*

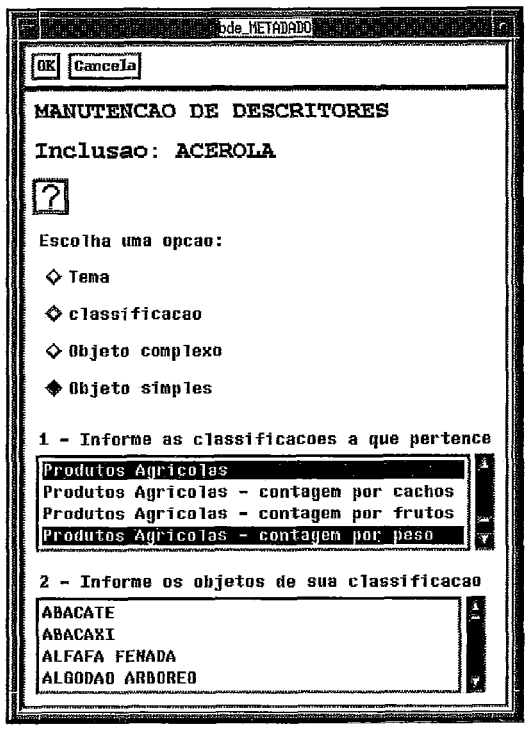
5.2.13. Menu para manutenção de descritores

. Programa: *descritor*



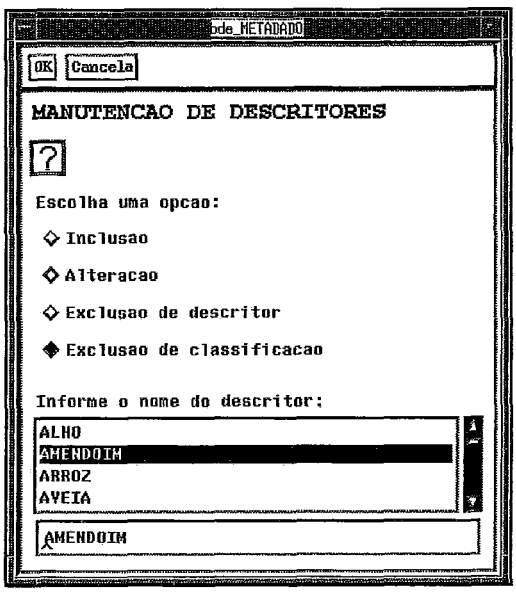
5.2.14. Inclusão de descritores

. Programa: *descritor*



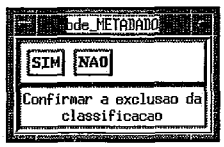
5.2.15. Alteração de descritores

. Programa: *descritor*



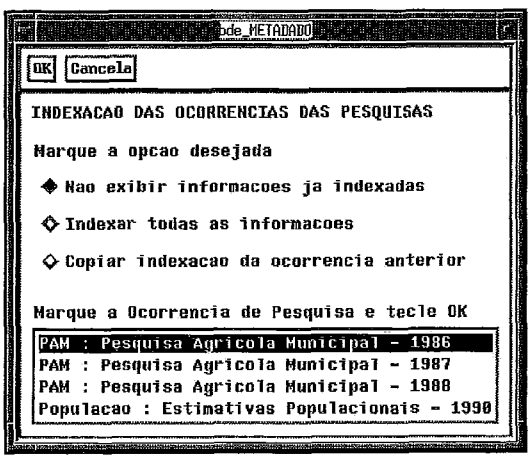
5.2.16. Confirmação de exclusão de descritor

. Programa: *descritor*



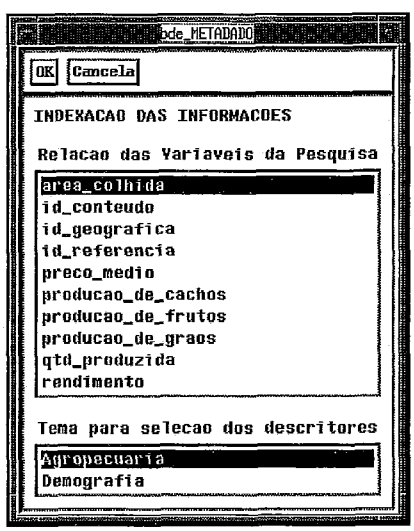
5.2.17. Menu de opções para indexação de informações

. Programa: *indexa*



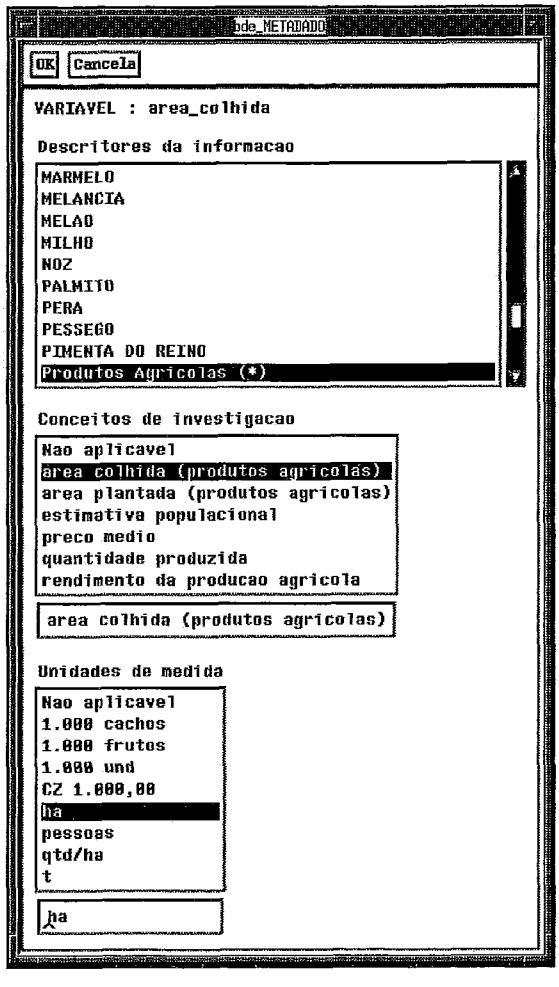
5.2.18. Relação das variáveis para indexação

. Programa: *indexa*



5.2.19. Indexação da informação

. Programa: *indexa*



6. APLICAÇÃO APOIO

A estrutura utilizada para implementar a aplicação APOIO pode ser visualizada nos diagramas de módulos constantes da figura 16, onde a diagramação é feita apenas para uma melhor visualização pois não existem ligações entre eles.

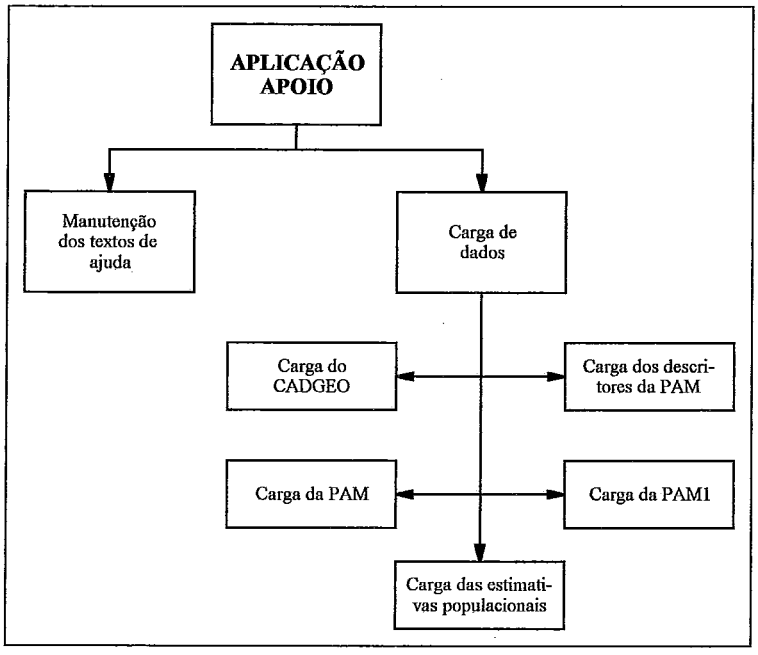


Figura 16 - Diagrama da aplicação APOIO

6.1. Detalhamento dos Módulos

Os programas da aplicação APOIO são do tipo utilitários, independentes entre si, destinados à execução das tarefas de manutenção do banco de dados.

6.1.1. Programa *auxilio*

a) Objetivo:

Efetuar a manutenção dos textos de auxílio do sistema.

b) Dados de entrada:

O programa utiliza telas para edição de textos, conforme os tipos apresentados no item seguinte, e os objetos da classe *Ajuda* mantidos no *named set* AJUDA.

c) Descrição do processamento:

O programa apresenta um menu para seleção de uma opção entre inclusão, alteração ou exclusão de um texto de auxílio. A seleção do texto é feita segundo o atributo de identificação, *referência*, que é usado como chave de acesso. Para as operações de alteração e exclusão é apresentado um menu com as referências dos textos existentes para a seleção. Para a inclusão, o processo é iniciado com a apresentação de um objeto vazio para ser preenchido.

d) Saídas produzidas pelo módulo:

O programa atualizada o *named set* AJUDA e exibe as telas 6.2.2 à 6.2.5.

e) Funções utilizadas pelo programa:

O programa utiliza a função *ajdfct01* para a montagem e seleção da lista de referências.

6.1.2. Programa *carga_pam*

a) Objetivo:

Carregar os dados das ocorrências da PAM - Pesquisa Agrícola Municipal produzidos até 1987 na base de dados.

b) Dados de entrada:

O programa lê arquivos *unix* formatados com os dados da PAM no modelo utilizado até 1987. O programa também consulta os *named sets* PESQUISA e DESCRITOR.

c) Descrição do processamento:

O programa recebe como parâmetro o ano de referência dos dados para selecionar o objeto da classe *Ocorrencia*. Para cada registro lido é criada uma instância da classe *D_PAM* no *named set* DADOS onde o código de produto é convertido para a identificação do objeto da classe *Descritor* e é acrescentada a referência ao objeto da classe *Ocorrencia* selecionado.

d) Saídas produzidas pelo módulo:

O programa inclui instâncias no *named set* DADOS.

6.1.3. Programa *carga_pam1*

a) Objetivo:

Carregar os dados das ocorrências da PAM - Pesquisa Agrícola Municipal produzidos a partir de 1988 na base de dados.

b) Dados de entrada:

O programa lê arquivos *unix* formatados com os dados da PAM no modelo utilizado a partir de 1988. O programa também consulta os *named sets* PESQUISA e DESCRITOR.

c) Descrição do processamento:

O programa é idêntico ao *carga_pam*, a menos dos objetos incluídos no banco de dados que são da classe *D_PAM_1* e do formato do arquivo de entrada que acrescenta um atributo.

d) Saídas produzidas pelo módulo:

O programa inclui instâncias no *named set* DADOS.

6.1.4. Programa de carga das estimativas populacionais

a) Objetivo:

Carregar os dados das estimativas populacionais na base de dados.

b) Dados de entrada:

O programa lê arquivos *unix* formatados com os dados das estimativas populacionais. O programa também consulta os *named sets* PESQUISA e DESCRITOR.

c) Descrição do processamento:

O programa seleciona no *named set* PESQUISAS o objeto correspondente as estimativas populacionais de 1990 para identificar as instâncias que são criadas em DADOS para cada registro lido no arquivo de entrada.

d) Saídas produzidas pelo módulo:

O programa inclui instâncias no *named set* DADOS.

6.1.5. Programa de carga dos descritores da PAM

a) Objetivo:

Carregar os descritores utilizados na indexação da PAM - Pesquisa Agrícola Municipal.

b) Dados de entrada:

O programa lê um arquivo *unix* com os descritores da PAM.

c) Descrição do processamento:

O programa cria, para cada registro lido, uma instância da classe Descritor no *named set* DESCRITOR, formando as referências dos atributos *classificacao* e *objetos*, através da estrutura de codificação utilizada.

d) Saídas produzidas pelo módulo:

O programa inclui instâncias no *named set* DESCRITOR.

6.1.6. Programa de carga do CADGEO

a) Objetivo:

Carregar os itens geográficos da estrutura formal político-administrativa e da estrutura legal de regiões metropolitanas no banco de dados.

b) Dados de entrada:

O programa lê os arquivos *unix* do cadastro geográfico e de regiões metropolitanas.

c) Descrição do processamento:

O programa cria, para cada registro lido, uma instância da classe *Cadastro_geografico*, compondo as referências existentes na tupla *estrutura* através da estrutura do código utilizado. Para as regiões metropolitanas, é usado o tipo de código 2 e a estrutura de objetos é definida por tabela de municípios componentes.

d) Saídas produzidas pelo módulo:

O programa inclui instâncias no *named set* CADGEO.

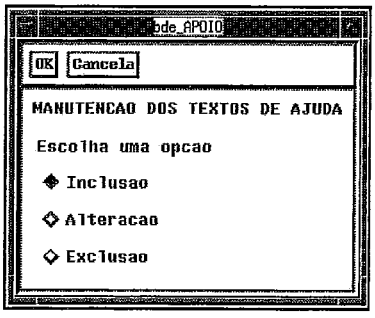
6.2. Definição das Telas

6.2.1. Tela da aplicação APOIO

. Programa: Tela gerada pelo comando *run* do O_2

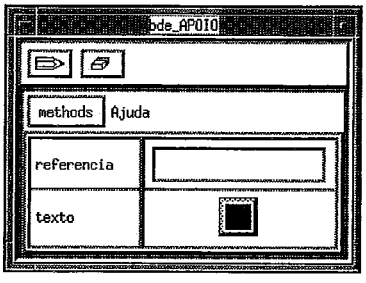
6.2.2. Menu para manutenção de textos de auxílio

. Programa: *auxilio*



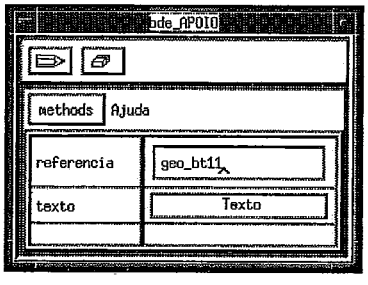
6.2.3. Tela para inclusão de novo auxílio

. Programa: *auxilio*



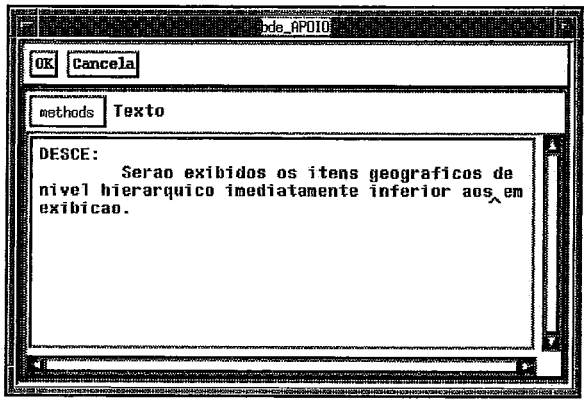
6.2.4. Tela para controle da inclusão do auxílio

. Programa: *auxilio*



6.2.5. Tela para edição de texto de auxílio

. Programa: *auxilio*



6.2.6. Lista para manutenção de auxílios

. Programa: *auxilio*

7. CONSIDERAÇÕES SOBRE O O_2

Neste item estão detalhadas as observações feitas no capítulo 7 da tese a respeito do SGBDOO O_2, para as quais foram desenvolvidas soluções de contorno aos problemas encontrados.

7.1. Ferramentas gráficas

O O_2Look é um conjunto de ferramentas do O_2 para a auxiliar a construção de uma interface gráfica utilizando o gerenciador gráfico Motif. O O_2Look permite o manuseio de todos os tipos de objetos, inclusive os complexos. Dado um comando, um display ou edit, e um objeto o O_2Look verifica na base de dados qual o tipo e o

estado deste objeto para então, automaticamente, preparar e exibir uma tela. Quando esta tela for alterada, o O_2Look irá atualizar a base de dados se a alteração for aceitável. O gerador de interfaces O_2Look é um conjunto de ferramentas do Motif para manuseio de objetos de uma base de dados O_2. Um comando do O_2Look pode envolver milhares de linhas de comando Motif.

Para a construção das interfaces gráficas desenhadas pelo usuário, o O_2 fornece um conjunto de ferramentas chamadas de O_2Kit. Estas ferramentas são recursos elaborados sob o O_2Look e permitem a rápida construção de janelas simples ou complexas.

As janelas simples manuseiam um único componente de uma interface gráfica. Permitem um diálogo do tipo envio de mensagem, perguntas do tipo sim/não, edição de um texto, escolha única ou múltipla, ou edição de uma alternativa escolhida numa lista de ítens. Os métodos disponíveis na classe "Box" implementam inteiramente os recursos gráficos e devolvem ao programador a opção feita pelo usuário após a execução.

As janelas complexas manuseiam muitos componentes, do mesmo tipo ou não, numa única janela. O programador define a lista dos componentes que irão implementar os recursos gráficos utilizando: botões (Button) de seleção compostos por botões para seleção única dentre várias alternativas (radio buttons) ou de botões para seleção múltipla (options buttons); componente para a definição de título (label); componente para receber um texto do usuário (prompts); componente para superposição de desenhos em formato pixmap ou bitmap (picture); e componentes para o manuseio de lista de ítens permitindo optar por escolha unitária, múltipla ou editada. Para cada componente o O_2Kit dispõe de um método de edição (get_mask) que irá definir a forma do componente na janela, e um método para inicialização do objeto (init) que irá construir a instância do componente com os dados desejados. Os componentes gráficos são editados num objeto da classe "Dialog_box" que dispõe dos métodos para criar uma apresentação (create_presentation) e para informar as opções tomadas pelo usuário sobre a janela (answer). Após ser criada uma apresentação, o O_2Look dispõe de métodos para exibir ou retirar a janela de uma tela (map/unmap), para parar o processamento até que o usuário interaja com o botão "ok/cancela" da janela (wait/grab), para atualizar o objeto com as opções do usuário (lk_consult) para liberar uma janela (lk_free) ou para acabar com a apresentação (lk_delete).

Apesar do bom resultado visual da interface gráfica do O_2 a funcionalidade do O_2Look/O_2Kit não é completa. Faltam recursos do tipo "pop-up" e botões que permitam a execução de opções dentro do contexto da janela. O O_2Look implementa apenas dois botões de processamento em cada janela, botões *OK/Cancela*, *SIM/NÃO* ou *SAVE/ERASE* que liberam o estado de "wait" de um programa. Quando são desejadas diversas opções de processamento em um menu, ou, até mesmo, quando se deseja dotar a interface com um botão para auxílio inserido no contexto, o projeto da janela pode torna-se complexo e de difícil manuseio pelo usuário.

Visando simplificar a operação da interface gráfica, no sentido que o manuseio de um único componente seja suficiente para executar uma função, foi desenvolvido um novo objeto para compor o O_2Kit, implementado no esquema do banco de dados como uma subclasse da superclasse "Component". Esta classe, denominada "Botao", foi definida, para fins de utilização dos recursos gráficos do O_2Look, como sendo um objeto e foram acrescentados dois métodos aos herdados da classe "Component": método "processa" e "ajuda". Os recursos gráficos da aplicação foram modificados para que o menu passasse a exibir apenas estes dois métodos com os nomes alterados em cada botão, de forma que o método processa fosse exibido com o nome da função. Assim, ao ser acionado um "botão" o usuário pode confirmar o significado do ícone através do seu nome ou escolher a função de auxílio que provoca a exibição de uma janela com o texto de ajuda restrito ao contexto.

Os ícones utilizados foram construídos com o uso do "bitmap editor", em preto e branco, no formato de 24x24 pontos. O O_2Look utiliza o atributo "ed_name", herdado da superclasse "Component", para associá-lo ao nome do bitmap informado no arquivo de recursos em uso pela aplicação conforme o esquema mostrado a seguir:

a) definição da classe "Botao":

```
class Botao inherit Component
  public type tuple(
              pid : integer)
  method public init (ed_name: string),
              public get_mask : integer,
              public processa,
              public ajuda
  end;
  method body init (ed_name: string) in class Botao {
      self->ed_name = ed_name;
```

```
return;
};
method body get_mask : integer in class Botao {
    return(lk_object(self->ed_name,0,0,0)); };
method body processa in class Botao {
    lk_free(self->pid, atoi(self->ed_name[6:7]));
    return; };
method body ajuda  in class Botao {
    o2 Ajuda auxilio;
    o2 Box dialogo = new Box;
    for (auxilio in AJUDA where
      (auxilio->referencia == self->ed_name)) {
      auxilio->texto->display;
      return; }
dialogo ->message("\nAuxilio nao\ndisponivel\n","auxilio");
return; };
```

b) no arquivo *.LKdefaults*, criado no diretório principal (*$HOME*) do usuário:

```
*geo_bt11.menu: {{"e1" "DESCER" processa "Auxilio" ajuda}}
*geo_bt11.bitmap:  /oca/BD/bevi/bitmaps/botao11.xbm
```

O atributo "pid" da classe "Botao" foi definido para conter o ponteiro da apresentação da janela. Os programas que utilizam estes botões definem tantas instâncias desta classe quanto forem os botões desejados e, após a criação da apresentação, atribuem o conteúdo do ponteiro da janela ao atributo. O método "processa", quando executado, libera o processamento do programa executando a função "lk_free" do O_2Look e envia pela interface de comunicação das funções um código numérico que identifica o botão utilizado. Para simplificar esta identificação foi utilizada uma padronização no conteúdo do atributo "ed_name", de forma a que o código numérico fosse sempre tomado das posições 7 e 8, como no exemplo abaixo:

a) na definição de objetos do programa:

```
o2 Botao bt1 = new Botao("geo_bt11"),
     bt2 = new Botao("geo_bt12"),
      ...
     bt9 = new Botao("geo_bt19");
  o2 list (Component)
    lista_radio2 = list(radio2),
```

```
   ...
  lista_de_botoes = list(bt1, bt2, ... , bt9);
Presentation geo_pid;
o2 Dialog_box geo_box;
```

b) na construção da apresentação:

```
geo_box = new Dialog_box("geo_box", list(titulo,
      lista_de_botoes,... , lista_radio2));
geo_pid = geo_box->create_presentation;
bt1->pid = geo_pid;
...
bt9->pid = geo_pid;
```

c) na exibição da janela:

```
lk_map(geo_pid, STACK, FREE, pid_ant, 10, 10);
n = lk_wait(geo_pid);
lk_consult(geo_pid, geo_box);
opcao_radio2 = radio2->answer;
...
switch (n) {
   case 1:          /* botao SAVE */
   ...
   break;
   ...
   case 16:        /* botao SALVA lista num arquivo */
   ...
```

A solução escolhida permitiu o uso de diversos botões numa única janela dotando as aplicações desenvolvidas com o tipo de requisito desejado nas interfaces gráficas.

7.2. Consultas à propriedades de subclasses

O teste do projeto foi efetuado utilizando um único "named set" para as instâncias de cada superclasse. A solução pretendia simplificar o esforço de programação na recuperação de objetos utilizando os métodos do O_2 para localização das suas instâncias. Mesmo considerando que a identificação do tipo e da classe de instância pode ser efetuada durante a execução de um programa ou de uma consulta de forma trivial,

falta às linguagens de programação e consulta um operador específico para esta separação.

A alternativa consiste em utilizar o método genérico "title", herdado da classe "object" (definida pelo O_2 como a raiz da hierarquia de classes), para fazer a comparação com o título da classe desejada:

```
if (objeto->title op "Classe_x") ...
```

Ou, então, a partir de um objeto conhecido pode ser feita a comparação do tipo ou da classe, com o novo objeto:

```
...
o2 Classe_x objeto_1 = new Classe_x;
if (objeto->class_of op objeto_1->class_of) ...
if (objeto->type_of op objeto_1->type_of)...
```

Uma consulta também pode identificar a classe de uma instância utilizando os mesmos métodos, como no exemplo a seguir que constrói um set de objetos da classe "Classe_x", entre as instâncias existentes no "named set" chamado de "SET":

```
select x from x in SET where x.title = "Classe_x"
```

O uso do mecanismo de herança exige certos cuidados com a linguagem de programação para ao manuseio das instâncias de subclasses. A linguagem de programação O_2C não compila para um objeto da superclasse as propriedades definidas nas subclasses. É preciso fazer explicitamente um "cast" do objeto para poder utilizá-las. A solução mais cômoda, do ponto de vista do programador, consiste em trabalhar com a definição de um objeto para cada subclasse. A navegação dentro do "named set" é realizada pela classe que o definiu, normalmente a superclasse, e assim que a classe de uma instância seja identificada, deve ser feita sua atribuição ao objeto correspondente. Este procedimento evitará que tenha de ser acrescentado aos comandos da linguagem que manuseiam propriedades específicas da subclasses a definição da classe desejada, como no exemplo abaixo:

```
... if ((o2 Classe_x) objeto->propriedade op ...
```

Outra restrição importante existe na linguagem de consulta. A O_2SQL somente reconhece as propriedades da classe que define o "named set". Para se obter

uma propriedade específica de uma subclasse, deve-se definir uma pseudo propriedade (um método vazio) com o mesmo nome na superclasse. Com o uso de redefinição de propriedades, o mecanismo de montagem a posterior irá utilizar a propriedade correta, ou seja, aquela existente na classe da instância, conforme o exemplo:

```
create class Super ...
  method propriedade ...
  end;
  method body propriedade in class Super { return;};

create class Classe_x inherit Super ..
  propriedade: string, ...
  end;
```

Esta restrição é uma limitação de linguagem específica do O_2. O ORION, por exemplo, pode manusear as propriedades definidas numa classe ou nas classes de uma hierarquia de classes, em sua linguagem de consulta [KIM89]. Por outro lado, apresenta a restrição de não permitir o manuseio de classes diferentes, o que impede operações como as de junção dos modelos relacionais.

7.3. Mecanismo de transações

O problema de uso da memória é crítico no O_2. O usuário não dispõe de recursos para intervir, ficando limitado ao uso dos comandos de commit e validate. O validate permite descarregar para a base de dados as transações já concluídas que estejam em memória, não desalocando as áreas de memória solicitadas em algum ponto anterior. O empilhamento de janelas e a construção de janelas com muitos atributos são, além da construção de listas descritas no item anterior, fatores de grande demanda de memória. A solução encontrada para poder trabalhar com transações maiores ou para editar janelas com muitos atributos foi dividir a transação em transações menores e utilizar o commit para liberar a memória necessária.

Esse uso do commit, desvirtuando o conceito de transação, acarreta na construção de um procedimento para "restart" pois o término de uma transação é também entendido pelo O_2 como final de processamento. A solução adotada consistiu na utilização de uma variável global (e única pois o O_2 limita o uso a apenas uma variável global) para o programa que emitia um commit informar ao programa de restart qual seria o próximo a ser executado. O O_2, após executar o commit, carrega o programa

de restart para que este analise o motivo da interrupção do processamento e intervenha no processo, no caso, chamando o próximo programa da transação. No exemplo a seguir, encontram-se os comandos e definições necessários a este manuseio:

. programa restart:

```
program restart (why: integer) in application BDE
program body restart (why: integer) in application BDE {
#include "o2_event.h"
...
o2 string from_prog, nome;
o2 Arquivo arq;
typedef struct
        { char current_prog[20];
          char nome_do_arquivo[30];
          int current;
        } context;
extern context *o2_global;
...
strcpy (from_prog, o2_global->current_prog);
strcpy (nome, o2_global->nome_do_arquivo);
switch (why) {
...
case O2_COMMIT:
   if (from_prog == "rec3000"){
      recdados(arq); };
...
   if (from_prog == "recdados"){
     rec3100(arq); };
...
```

programas de aplicação:

```
typedef struct
        { char current_prog[20];
          char nome_do_arquivo[30];
          int current;
        } context;
extern context *o2_global;
...
```

```
      o2_global = (context *) malloc (sizeof(context));
      strcpy (o2_global->current_prog,"rec3100");
      strcpy                              (o2_global->nome_do_arquivo,
arq_sessao->nome_do_arquivo);
      commit; };  /* fim do programa */
```