Manipulação de Bancos de Dados através de Formulários

Redação final da tese de Mestrado defendida e devidamente corrigida por Silvia Maria Fortuna Mendes de Souza e aprovada pela comissão julgadora.

Campinas, 14 de setembro de 1988

Orientadora Prof. Dra: Claudia Bauzer Medeiros

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

TESE

Manipulação de Banco de Dados através de Formulários

Agradecimentos

À Profa. Dra. Claudia Bauzer Medeiros, pela orientação e incentivo.

Ao Prof. Dr. Geovane Cayres Magalhães, pelo fornecimento de material bibliográfico e esclarecimentos.

Ao Prof. Dr. Mário Jino, pelas sugestões que permitiram melhorias do presente texto.

À Profa. Dra. Sonia Schechtman Sette, pelas críticas e sugestões de continuidade do presente trabalho.

Ao meu esposo Guilherme Alonso Mendes de Souza pelo apoio e estímulo.

À minha mãe, Waldenice Mascarenhas Fortuna, cujo auxílio muito contribuiu para a concretização deste trabalho.

SUMÁRIO

Formulários têm sido uma ferramenta básica na administração de escritórios, assegurando eficiência no armazenamento e recuperação das informações. Uma das vantagens dos sistemas baseados em formulários é que eles são simples e naturalmente aceitos pelos usuários finais. Também facilitam a definição de "interfaces padronizadas", a pré-definição de parâmetros e a integração de ferramentas.

Este trabalho versa sobre a utilização de formulários no manuseio de informações armazenadas em Bancos de Dados relacionais. Visa, em especial, definir uma interface geral que permita aos usuários criar seus formulários e utilizá-los para consultas e/ou atualizações às diversas relações que compõem o Banco de Dados.

ABSTRACT

Forms have been a basic tool in the management of offices, ensuring eficiency in the storage and retrieval of the information. One of the advantages of forms-based systems is that they are simple and well accepted by the end users. Also, they facilitate the definition of the "standard interfaces", the predefinition of parameters and the integration of tools.

This thesis concerns the utilization of forms in the handling of information in a Relational Database System. It is important to say that this work also defines a general interface that allows the users to design their own forms and to use them to query and/or update the various relations of a Database.

Manipulação de Bancos de Dados através de Formulários

índice

Assunto Página

Anexos

Índice de Figuras

Índice de Figuras

Anexos

Figura — Página

1 - Introdução e Conceitos

1.1 - Automação de Escritórios

Uma das áreas de desenvolvimento de aplicações que atualmente tem sofrido grandes avanços é a da automação de escritórios.

A automação de escritórios foi bastante estimulada pelos avanços na tecnologia de equipamentos de apoio, declínio dos custos de hardware e também pelo desejo de acréscimo da produtividade.

Uma das questões básicas é a necessidade de divulgação de conhecimentos no setor de automação de escritórios [TOM85]. Para muitas empresas, automação de escritórios restringe-se a processamento de texto, que ganhou aceitação como uma ferramenta essencial na administração comercial. Para empresas que já possuem maturidade em processamento de dados, o tratamento é integrar as atividades de escritório através de redes locais ou sistemas de grande porte interligados. Um dos aspectos a observar é que os trabalhadores desses escritórios precisam dos benefícios de um sistema que combine processamento de texto com as vantagens de um ambiente "on line", onde existirá uma fusão entre processamento de dados e textos [HER83]. Tudo isto se deve ao fato da grande aceitação do uso dos terminais de computadores em diversas áreas comerciais (vendas, entrada de dados, etc), que pode ser explicada por vários aspectos. Um deles diz respeito ao desejo de registrar e recuperar dados que são processados e armazenados no ponto de criação ou uso, sem as desvantagens de erros de transmissão além dos gastos nas instalações. Além disto, elimina-se o

tempo de impressão de várias cópias para serem examinadas pelos diversos pontos. Com isso obtém-se um melhor suporte no processamento, armazenamento, atualização, distribuição e impressão de todo tipo de informação e texto.

Enquanto no processamento de dados visamos tradicionalmente racionalizar a entrada e saída de dados do computador, no processamento de textos trata-se da racionalização do trabalho humano. Só recentemente alguns produtos começaram a integrar ambas as áreas em sistemas homogêneos. No processamento de dados, especialmente em aplicações comerciais, as informações manipuladas são formatadas e armazenadas em estruturas de Bancos de Dados, onde o tempo de resposta e a segurança são relativamente menos críticos e mais fáceis de controlar. Já no processamento de textos a informação não é formatada e a requisição de "performance" num terminal é bastante alta, balanceando o acesso a uma grande quantidade de informação com o tempo de resposta. Um outro problema na automatização de uma empresa é a multiplicidade de políticas e a diluição de responsabilidades dentro da empresa, podendo resultar na proliferação de terminais, equipamentos de comunicação e redes, daí a linha corrente de integração [HAM83]. Não é surpreendente que existam poucos sistemas interativos oferecendo bom suporte para aplicações que visem atender a esses requisitos tão variados. Nos últimos anos, muitos métodos e ferramentas se dirigem ao aumento de produtividade dos especialistas em processamento de dados e não do usuário final. A automatização não leva necessariamente à redução de custos, mas torna a empresa mais ágil e competitiva. Ao lado dos ganhos obtidos com a automação

de escritórios surgem problemas de natureza burocrática (problemas com reconhecimento de firma de assinaturas, por exemplo), e de natureza sócio-econômica (desemprego). No que diz respeito à preocupação com o conflito entre a modernização tecnológica e a geração de empregos, o que se conclui é que não necessariamente haverá desemprego, mas sim mudança no perfil dos profissionais da área, que deverão assumir certo nível de tomada de decisões, levando a um maior nível de capacitação de todo o pessoal da empresa. Para as secretárias que passaram das máquinas de escrever para os vídeo-teclados, a transição foi tranquila. No que diz respeito ao processamento de textos, os sistemas emulam as funções tradicionais, enquanto oferecem outras funções mais sofisticadas. Organizações que pretendem explorar o potencial de sistemas integrados necessitarão revisar muitos aspectos de sua estruturação interna e se a relação custos-benefícios justifica a reestruturação. Estruturas organizacionais comuns são baseadas na necessidade de reunir empregados de mesma função num mesmo espaço físico, de forma a dividir os recursos comuns e facilitar a comunicação. O novo conceito de estações de trabalho, com sistemas integrados, permite uma nova visão dessa exigência. Outras considerações sobre os impactos sociais da automação de escritórios em [MOW86].

Há alguns anos tornou-se evidente a necessidade de uma política que garanta a participação efetiva de empresas nacionais em segmentos até agora cativos de fornecedores estrangeiros. É evidente a distância entre o desenvolvimento tecnológico nesta área e a satisfação das necessidades dos usuários.

Dentro do universo de empresas brasileiras, os projetos de

automação de escritórios ainda são bastante restritos. Muitas são as razões dessa situação: limitações tecnológicas (ausência de alternativas tanto de hardware como de software), altos custos dos projetos (tanto de máquinas e programas como também de pessoal), limitações do mercado interno e também a influência dos fornecedores de produtos estrangeiros ou mesmo de empresas estrangeiras com filiais aqui no Brasil, que utilizam software da matriz estrangeira. Mesmo em estágio ainda incipiente, o mercado nesta área é muito promissor.

Como decorrência dos aspectos discutidos, surgiram várias linhas de pesquisa visando obter escritórios com nível crescente de automatização. Enquanto existem ferramentas (como processadores de textos, correio eletrônico, calendário, modelagem do fluxo de documentos, etc.) que visam atender atividades convencionais de escritórios, tais como preparação, distribuição, preenchimento e recuperação de documentos, confecção de cartas, memorandos, também existe outra corrente mais ambiciosa cuja meta é automatizar funções mais complexas indo além da mecanização de algumas tarefas. Por exemplo, o sistema denominado Alliance [CAS82] necessita de um conjunto de dispositivos associados a um telefone para o operador poder adicionar "anotações vocais" a documentos apresentados na tela. Usando técnicas de edição análogas às já existentes para o processamento de palavras, as mensagens podem ser modificadas antes do documento retornar ao disco. Obtemos assim uma definição de documento bastante incomum. Quando vozes são necessárias, elas são armazenadas em um disco como sinais digitais para serem adicionados aos documentos aos quais estiverem

relacionadas. Um arquivo que será utilizado e que possui mensagens associadas pode ter um símbolo na margem do texto. Ocorrendo neste ponto um chaveamento no mecanismo de reprodução, reproduz-se a informação armazenada sem ser necessário muito tempo de busca.

Para se obter um aumento de produtividade podemos concentrar os esforços em dois níveis: a nível individual, mecanizando tarefas ou a nível da organização, otimizando rotinas, reforçando cada componente de um determinado processo que visa atender uma função em particular [ELL80].

Funções mais complexas envolvem tanto atividades convencionais (manipulação de documentos), como atividades de processamento de dados (atualização de arquivos, união de várias fontes de dados, etc). Nas organizações, algumas dessas atividades podem ser computadorizadas, enquanto outras são manuais. Algumas atividades são fáceis de automatizar enquanto outras necessitam da intervenção do usuário e por isso são parcialmente automatizadas.

A automação de escritórios está sendo reconhecida como uma das mais avançadas técnicas de administração, com uma diversidade de componentes necessários como, por exemplo, redes de terminais, impressoras, discos, etc. A rota da automatização inevitavelmente envolve interfaceamento com equipamentos de várias fontes, o que exige uma padronização entre fornecedores de equipamentos e "softwares". Não existe nenhuma padronização no projeto de redes locais que possa ser base de um escritório automatizado [CAS82]. Um estudo sobre a evolução da automação de escritórios encontra-se em [UHL81].

Em ambientes de escritórios existem várias tarefas e procedimentos que podem ser identificados visando sua automatização. Dentre estas tarefas, uma atividade básica é o preparo (geração e preenchimento) de documentos, que por muitos anos consistiram em instrumentos essenciais na administração de escritórios. Esta atividade também implica na necessidade de arquivamento, recuperação e intercâmbio de documentos (correio).

1.2 Documentos de Escritórios

Documentos, ou, de uma forma mais geral, formulários, são a forma mais simples e usual de troca de informações. Os documentos podem ser formulários formatados (onde as informações são atributos desses formulários) ou formulários não formatados (textos, imagens, vozes ou combinação desses elementos). Em particular, preenchimento, distribuição e recuperação de documentos são operações básicas de todos os procedimentos de escritórios. Surge daí a importância de estudar ferramentas para representar documentos de escritórios de uma maneira formal, de modo a permitir um melhor manuseio através da automatização de procedimentos.

Com o advento da automação de escritórios surgiu o novo conceito de "documento eletrônico de escritório". Documentos eletrônicos são criados e manipulados em "estações de trabalho", transmitidos através de redes de comunicações de dados e arquivados, o que faz com que a sua recuperação neste ambiente dinâmico e distribuído se torne bastante difícil.

O problema de modelar tais documentos vem recebendo cada vez mais atenção nas pesquisas recentes. Uma das principais vantagens

dos sistemas baseados em formulários é que eles permitem uma transição natural dos sistemas manuais de escritórios para os automáticos. Os sistemas disponíveis atualmente enfocam aspectos distintos da manipulação de documentos criando definições implícitas de documentos.

Formulários possuem vários componentes, os quais podem ter vários tipos de estruturas e dados. De uma forma simplificada, podemos dizer que um formulário pode conter informações de natureza estética e instrutiva com locais pré-definidos para o preenchimento e difusão de informações.

O preenchimento deve respeitar a descrição física e as instruções, de forma a obter um documento que é na verdade uma mensagem escrita estruturada, que servirá para a comunicação entre vários tipos de entidades como por exemplo: pessoas, serviços e organizações.

1.3 - Trabalhos na área de Automação de Escritórios

A diversidade de problemas encontrados na área de Automação de Escritórios deu origem a várias propostas. Um exemplo é [SHU82], que concentrou seus esforços em desenvolver tecnologia para suportar a automatização de procedimentos comerciais. Como muitas dessas atividades comerciais envolvem processamento e/ou envio de formulários, a solução proposta é orientada por formulários, e resultou numa linguagem (compilada) para processamento de formulários.

Já [LAM84], teve por meta criar uma linguagem para desenvolvimento de aplicações de escritórios, englobando conceitos de de-

finição (composição, identificação de componentes, reconhecimento de tipos) de objetos de escritórios (cartas, memorandos, comunicação interna), envolvendo inclusive outros aspectos como o de comunicação e especificação de procedimentos. A modelagem de escritórios, no trabalho de [LAM84], visa prover ferramentas conceituais para descrever a semântica de sistemas de escritórios de forma a facilitar a programação de aplicações nesse ambiente. Sistemas de escritórios avançados podem ser vistos como organizações complexas de objetos, localizações, procedimentos, pessoas, etc, combinando estações de trabalho independentes que cooperam na execução de complicadas tarefas de escritórios. Em algumas soluções, procedimentos podem ser baseados em construtores de objetos que aceitam os diversos componentes e retornam objetos de escritórios de tipos compostos, como por exemplo: cartas, formulários, memorandos,etc.

Em [MAN84] descrevem-se as pesquisas dirigidas a projetos de Bancos de Dados através de formulários e também a projetos de visões globais. Neste trabalho pretende-se demonstrar a importância de formulários em ambientes de escritórios. Entre outros aspectos, fala-se da flexibilidade na representação de formulários através de diversos meios: papel, vídeo e até vozes. Fala-se também que formulários são modelos formais que suplantam a ambiguidade da linguagem natural, constituindo um modelo de dado, onde após análise tanto dos seus componentes como dos relacionamentos entre formulários, constatam-se várias dependências e mapeamentos. A primeira parte da pesquisa visa a definição de um modelo de formulário e a investigação intra e interformulários. Na aná-

lise intraformulário obtemos um diagrama baseado no modelo Entidade-Relacionamento. Já a análise interformulário envolve fluxo de documentos (resolvendo problemas de sinônimos, homônimos e também estabelecendo origens e destinos de campos). O projeto do esquema conceitual pode se desenvolver de forma incremental pela integração de esquemas de formulários e após estar completo define-se então o mapeamento. A análise interformulário visa ainda satisfazer o interesse em uniformizar o acesso a Bancos de Dados heterogêneos e distribuídos. Para isso, deve-se pesquisar melhores soluções para a obtenção da visão global, que é uma visão integrada de múltiplos Bancos de Dados. Este projeto pode ser dificultado por causa do número de objetos, do número de mapeamentos e da complexidade dos mapeamentos. Como resultado, desenvolveu-se uma metodologia para projeto de Visões Globais que requereu o desenvolvimento de algumas ferramentas auxiliares e demonstrou ser necessário maiores pesquisas em áreas tais como atualização de visões globais e objetos complexos.

[NIE85] concentrou-se em otimizar o projeto e a implementação de uma linguagem orientada para objetos para aplicações de escritórios. O ambiente integra conceitos de Sistemas Operacionais, Bancos de Dados e, em particular, trata de persistência de dados, transações atômicas, disparo de eventos, etc.

[BAR85] trata principalmente da recuperação de documentos por conteúdo. Introduz-se um nível conceitual de modelagem, resultando na definição das estruturas conceituais de documentos, pois a recuperação por conteúdo é obtida se podemos descrever as regras semânticas dos documentos. Entre sistemas orientados para

preenchimento e recuperação, a solução mais comum é criar extensões de Sistemas de Bancos de Dados, proporcionando a capacidade de manusear textos, imagens, etc, em adição aos dados formatados originalmente. Particular atenção foi dada aos Modelos Relacional e Entidade-Relacionamento (exemplo, Projeto BIG). O projeto TIGRE [COL83] visa a implementação de um Gerenciador de Bancos de Dados com capacidade de manusear dados gerais, onde documentos são considerados como objetos complexos. O TIGRE também é baseado no Modelo Entidade-Relacionamento. O conceito de tipo é imprescindível para entender a diferença entre a modelagem para sistemas de edição/formatação de documentos e sistemas de preenchimento/recuperação. No primeiro caso, especifica-se um esqueleto (pela sintaxe e formatação), no segundo especifica-se a estrutura e componentes comuns a todas as instâncias de uma mesma classe, facilitando o acesso e armazenamento. Um modelo bem projetado deve integrar as duas orientações para atender a esses objetivos conflitantes. Em outras palavras, deve ser tão flexível quanto possível, pois é impossível pré-determinar a estrutura de documentos de escritório. O trabalho salienta que é preciso fornecer o máximo possível de informações sobre a estrutura do documento para facilitar sua criação, preenchimento e recuperação. Um tipo neste projeto é a definição das propriedades comuns de um conjunto de documentos. Igual importância tem a definição de uma estrutura interna, envolvendo características semânticas (estrutura conceitual que reflete como usuários vêem classes de documentos similares), "layouts" e componentes sintáticos (estruturas lógicas como seções, tabelas, etc, envolvendo o mapeamento para os dispositivos

físicos). A estrutura lógica e o "layout" são mais fáceis de padronizar. Como resultado, introduziu-se um formalismo para a definição do modelo, utilizando-se diversas gramáticas (livres de contexto) para a definição dos diferentes níveis de modelagem.

[KAN85] introduz uma linguagem denominada D-OBE, que é uma extensão da OBE (Office-By-Example - que será descrita mais adiante). D-OBE é uma linguagem para sistemas distribuídos em automação de escritórios. Utiliza uma interface QBE. D-OBE é baseada na observação de que grandes escritórios dividem suas tarefas em departamentos. Cada departamento é responsável por suas tarefas e controles e irá administrar seu conjunto D-OBE, que é uma quantidade de estações de trabalho conectadas a um banco de dados, correio eletrônico, etc. Tal conjunto pode cooperar com outros conjuntos. Sistemas distribuídos para escritórios são sistemas que combinam manuseio de dados distribuídos com processamento de aplicações para escritório. Um dos problemas encontrados foi o desenvolvimento de uma linguagem simples embora eficiente ao ponto de atender à complexidade de um sistema desse tipo. Um sistema que implementa a linguagem D-OBE tem três classes de componentes: estações de trabalho, servidores (que fornecem serviços compartilhados) e uma rede que integra esses componentes. As exigências para as estações D-OBE se baseiam em computadores pessoais (PCs), pois eles são bastante populares mesmo em corporações com sistemas sofisticados. Essa preferência ocorre devido a certos aspectos: privacidade, disponibilidade e "interfaces" livres, que podem ser obtidos configurando-se as estações e projetando-se D-OBE de tal forma que cada estação possa ser usada tam-

bém como um sistema isolado. Cada conjunto pode ser atendido por uma rede local ou pela conexão das várias estações a um computador de grande porte onde os servidores serão implementados. Quanto ao aspecto de segurança, D-OBE identifica o responsável pelos diferentes objetos (dados) e componentes do sistema. Com atribuição de identificadores aos diversos servidores obtemos as "páginas amarelas" de uma rede de serviços. O administrador de cada escritório formula as regras para os diversos responsáveis seguirem, determinando inclusive "passwords" (ou seja, senhas) para controle de acesso.

O usuário numa estação de trabalho pode dispor de duas opções de armazenamento de dados: privada, com um banco de dados local e compartilhada, com um banco de dados acessado por mais de uma estação ou conjunto D-OBE. Como os conjuntos estão associados a departamentos com diferentes tarefas, a maioria das transações não envolve mais de um banco de dados compartilhado. O processamento de dados distribuído no sistema D-OBE pode ser considerado como uma generalização de um esquema para um Banco de Dados. Uma característica interessante no ambiente D-OBE é a vida longa das transações, principalmente porque envolvem interação com o usuário.

[WAR86] apresenta uma ferramenta reutilizável para software orientado para formulários, denominada Fillin, largamente usada em ambientes SPS (Software Productivity System), desenvolvido pela TRW Software Productivity Project, para melhorar a produtividade dos implementadores de software da TRW. Fillin oferece uma interface orientada por formulários para usuários e para as fer-

ramentas. é um pacote de propósito geral, para executar operações em formulários, cujo aspecto mais importante é a habilidade de integração (que está intimamente ligada à reutilização) e a preservação de consistência das "interfaces" dos usuários nas diversas ferramentas. A padronização de interfaces em sistemas integrados é mais facilmente obtida através do auxilio de ferramentas do que com políticas ou procedimentos. O pacote Fillin consiste de 5 partes: uma linguagem de definição de formulários (para descrever a informação e formato dos formulários), uma linguagem de descrição dos dados dos formulários (define o mapeamento entre dados e formulários), um conjunto de subrotinas para a interpretação dessas linguagens, um conjunto de subrotinas para manusear a "interface" do usuário e um nível de comandos para os dois conjuntos de subrotinas. A implementação do sistema já está na sua terceira versão, que foi bastante influenciada pelas opiniões dos diversos usuários que vinham utilizando o sistema.

Para Fillin, um formulário na percepção dos usuários é uma coleção linear de dados combinados com informações formatadas que determinam seu "layout". O propósito de Fillin é apresentar esta visão dos dados aos usuários junto com uma interface com comandos para criação, edição e apresentação de formulários. Para os construtores de ferramentas, Fillin oferece operadores (via tipos abstratos de dados) para manusear a estrutura interna dos formulários e a interface do usuário. Fillin é primordialmente um pacote de manipulação de formulários individuais, mas pode manusear coleções de formulários, embora não possua algoritmos sofisticados para armazenamento e recuperação de formulários. Fillin é

independente de qualquer sistema de gerenciamento de Bancos de Dados, embora possa ser usado para oferecer um mecanismo simples para Banco de Dados Relacional, pois as rotinas que interpretam os arquivos de formulários e de dados podem ser usadas independentemente de outras rotinas. Sistemas que utilizam Fillin podem acrescentar outras rotinas que incluam operações da álgebra relacional, tais como projeção ou seleção. Apesar do exposto, em muitas situações torna-se evidente a necessidade do poder de consulta de um gerenciador de Banco de Dados, embora em casos simples o Fillin seja satisfatório.

[KIN87] descreve um sistema gerenciador de formulários de escritórios denominado FREEFORM. Tanto as descrições dos formulários quanto os dados são armazenados num Banco de Dados orientado para objetos (CACTIS). Freeform permite edição de formulários com consequente alteração no esquema correspondente do banco de dados (quando necessário). Também permite a criação de versões de um mesmo formulário, surgindo daí o conceito de Família de Formulários (onde a integridade e consistência são mantidas automaticamente). é um sistema que nos níveis mais externos é orientado por 'menus' e é executado em uma estação de trabalho SUN, num ambiente de Sistema Operacional UNIX e foi implementado na linguagem C. Possui os seguintes módulos principais: Módulo de Edição, Módulo de Uso e Sistema de Correio. O interesse maior foi em sistemas que gerenciam formulários que são usados para criar, modificar e armazenar descrições de formulários e efetuar operações com dados tais como preenchimento e recuperação de formulários. Uma das principais características é a possibilidade do usuário criar

versões particulares de um dado formulário.

[RUL87] descreve uma proposta de linguagem de gerenciamento de documentos (estruturados hierarquicamente), que além de serem considerados como unidades de controle e gerenciamento, são também considerados como unidades de armazenamento em Bancos de Dados de Documentos. O produto desse trabalho corresponde a uma interface de consulta e manipulação, chamada DQL (Document Query Language), assim denominada para ressaltar a linguagem de acesso a Bancos de Dados utilizada: SQL (Structured Query Language). Conceitualmente, neste projeto, documentos são composições de objetos subordinados. Também são apresentadas definições gerais quanto a tipos de documentos e instâncias. Um tipo de documento (segundo a Álgebra de Documentos considerada no projeto) é expresso por meio de esquemas (mecanismos que suportam a construção e manutenção de instâncias de documentos). Esquemas podem ser representados graficamente por árvores, onde os nós representam os esquemas envolvidos no tipo do documento e os arcos representam relacionamentos entre os esquemas. Apresenta definições de diversos comandos, tais como: CREATE DOCTYPE (usado para criar tipos de documentos e definir os esquemas), SELECTDOC (recupera documentos por tipo), INSERTDOC (insere ocorrências de documentos). Através do uso de mecanismos básicos como os predicados de pesquisa de documentos (usados em diversos comandos para seleção condicional) e expressões de projeção (para obter subesquemas de documentos), DQL cobre criação e controle de documentos tão bem quanto atualização de documentos (inclusive de valores atômicos). Outras pesquisas sobre processamento de documentos

em [STO83].

Se analisarmos detalhadamente as diversas soluções dos trabalhos mencionados acima, sejam modelagens, linguagens ou ferramentas, observaremos um elo em comum, ou seja o manuseio de documentos, mesmo que de uma forma implícita. Considerando este fato e mais a evolução das necessidades de automatização dos serviços em escritórios e a ligação cada vez mais forte do usuário final com o computador surgiu um novo estilo de programação, que é a "programação orientada por formulários", visando o desenvolvimento de aplicações interativas para Bancos de Dados [ROW85]. Uma grande variedade de aplicações pode ser classificada como sistemas interativos que permitem a diversos usuários acessar e atualizar dados armazenados em Bancos de Dados. Essas aplicações envolvem uma interação significativa do usuário com a aplicação, mas em contrapartida não requerem muito tempo de processamento.

Uma interface típica é a apresentação de um formulário no vídeo de um terminal através do qual dados podem ser fornecidos ou apresentados. Neste ambiente, as aplicações são compostas por uma coleção de telas (que contêm formulários, onde os dados são aceitos ou apresentados) e "menus" com as operações permitidas. O usuário navega através de diversas telas, executando operações para obter as ações desejadas. O sistema oferece comandos de construção através do teclado e/ou dispositivos tais como o "mouse" para navegar na tela para efetuar operações. Existem produtos onde as aplicações podem ser definidas através do preenchimento de formulários, onde telas genéricas podem ser adaptadas para usos específicos. O FADS - Forms Application Development

System, da Universidade da Califórnia, associado ao Banco de Dados relacional INGRES e o ABF - Application-By-Forms, produto comercial baseado no FADS seguem esta filosofia de programação [ROW82] [ROW85].

A solução orientada por formulário contrasta bastante com outras que visam metas semelhantes, mas que foram baseadas em extensões de linguagens de programação convencionais ou linguagens de geração de relatórios ou ainda são projetos de novas linguagens. Nessas soluções, os programas são mais longos, exigem mais tempo para o desenvolvimento das aplicações e maiores custos de manutenção e expansão. Outro fator de complicação é a falta de suporte para ferramentas de alto nível. Além disso, existe dificuldade de utilização do produto por usuários finais. Esses sistemas não possuem comandos para especificação de funções de alto nível e consequentemente, para determinadas situações, o programador necessita especificar mais código.

A solução orientada por formulários é mais prática por várias razões. Uma característica importante é a existência de interfaces padrão. Outro fator é que suporta uma coleção de ferramentas de alto nível e isto facilita a especificação de aplicações, já que cada ferramenta é projetada para resolver um problema particular. Ainda outra característica é o fato de permitir o desenvolvimento de aplicações interativamente. Outro aspecto dos sistemas nesse novo ambiente diz respeito ao estabelecimento de condições "defaults", isto é, pré-definidas, o que agiliza o desenvolvimento de aplicações e exige pouca especialização na área. Existem soluções onde além das estruturas de telas

fornecidas pelos produtos podemos ter telas definidas pelo próprio usuário, que edita o formulário e usa uma linguagem de alto nível apropriada para a especificação das operações.

De certa forma, para a obtenção de flexibilidade e desenvolvimento de aplicações interativamente a "performance" é sacrificada. Isto é sentido principalmente se a aplicação for usada num ambiente de produção. Nesses ambientes, a performance é mais crítica porque o software será usado muitas vezes e de forma simultânea por vários usuários. Por isso os softwares que utilizam a filosofia de orientação por formulários devem permitir facilidades que aumentem a performance para ambientes de produção.

Alguns dos sistemas de manuseio de formulários disponíveis são:

-QBE - Office By Example [COL83] [ZLO82] [WAI87]

Trata-se de uma extensão do sistema relacional QBE [ZLO82] e integra as seguintes características: processamento de textos, dados e correio eletrônico; definição pelos usuários de seus próprios menus e procedimentos, bem como acesso ao banco de dados central; definição de formulários e relatórios; tratamento gráfico e de imagens; controle de concorrência. Para geração de formulários, define-se a estrutura gráfica e especificam-se os campos (com variáveis exemplo), de forma que tenham correspondência biunívoca com os atributos das relações base. Também é permitida a especificação de direitos de acesso (módulo de autorização) e restrições de integridade. Não é levada em consideração a semântica da aplicação do formulário. Atualmente, uma versão operacio-

nal para mono-usuário já está implementada.

-OFS - Office Filing System (para o MRS) [COM81] [COL83] [TSI83]

Atende aos seguintes aspectos: tratamento de textos, dados e correio eletrônico; comunicação oral e gráfica; acesso ao Banco de Dados; tratamento do aspecto de distribuição; definição de formulários. O formulário é uma abstração e é tratado como uma mensagem descrita por um conjunto de atributos e procedimentos. Cada tipo de formulário relaciona-se com uma relação base. Também são permitidas as definições de procedimentos automáticos que são executados sob certas condições.

-SQL*FORMS (para o ORACLE) [ZUS86]

É uma ferramenta que permite a definição de aplicações para manusear dados no SGBD relacional ORACLE. Atende aos seguintes requisitos: definição de formulários; manuseio de tela/teclado para permitir edição e seleção; acesso ao Banco de Dados ORACLE para inserir/alterar/eliminar/apresentar registros; validação, conversão e formatação de dados; definição de procedimentos automáticos para controlar a integridade das aplicações; definição de procedimentos do usuário para melhorar a eficiência no tratamento das informações ao atender a situações complexas. Um formulário "SQL*FORMS" é definido como um conjunto de blocos, onde cada bloco é associado a uma relação e pode apresentar uma só ocorrência a cada vez ou um conjunto de tuplas da relação. Sua estrutura pode ser "default", estabelecida pelo próprio SQL*FORMS ou pode ser definida pelo próprio projetista.

-PROFS - Professional Office Systems [IOM85]

é um projeto da IBM bastante difundido, possuindo até uma versão em português. Exige equipamento de grande porte, está apoiado no SGBD IMS e agrega um conjunto de sistemas para poder executar diversas funções de escritório. A comunicação com o usuário é feita através de "menus". Atende aos seguintes requisitos: preparação de documentos; correio eletrônico; pesquisa, recuperação e arquivamento de documentos; possibilidade do usuário desenvolver seus aplicativos e integrá-los ao sistema; acesso ao Banco de Dados central. A grande vantagem desse sistema é que aumenta a produtividade, melhorando a velocidade de comunicação entre as pessoas e reduzindo o tempo necessário para a preparação, pesquisa e recuperação de documentos. Já está implantado em grandes organizações bancárias (Bamerindus, Bradesco, Sulamerica), industriais (Philips, IBM) e comerciais (Grupo Silvio Santos).

-UNIFORMS [HUL86]

é uma ferramenta para automatização de escritórios que permite geração de formulários que poderão ser associados a programas de aplicação posteriormente. Integra as seguintes características: definição de formulários: projeto e manutenção; rotinas para permitir a apresentação de informações sobre os formulários; conjunto de funções (para serem usadas pelos programadores de aplicações em tempo de processamento para manusear todas as operações em formulários e para permitir que os dados sejam manipulados de uma forma simples e conveniente. Oferece não somente um editor para projeto (criação e manutenção) de formulários, mas

também um módulo para fornecer informações sobre os formulários que poderão ser impressas ou apresentadas através do terminal, contendo inclusive uma imagem do formulário e informações sobre todos os campos do mesmo. Oferece também uma série de funções que, quando associadas a programas de aplicação, permitem que os formulários previamente projetados sejam utilizados e os dados aceitos. Foi desenvolvido na linguagem de programação 'C' e implementado no DEC VAX-11/780, sob o VMS.

Como podemos observar, a maioria dos sistemas citados privilegia o aspecto de integração, que implica na combinação de diferentes tipos de informação, funções e dispositivos. O termo sistema integrado de escritório tem sido bastante usado nos últimos anos. Embora este conceito não tenha sido claramente definido, ele certamente envolve problemas de comunicação entre pessoas, diferentes sistemas e diferentes aplicações. Pode-se classificar sistemas integrados de escritórios em quatro categorias, baseadas nos níveis de integração [WAT87]: 1) inclui sistemas que permitem aos usuários processar diferentes aplicações em um mesmo ambiente. Não correspondem verdadeiramente a uma integração, são meramente coleções de programas (Exemplo: IBM TopView); 2) envolve sistemas que permitem integração pela troca de informações. Essa troca requer traduções de formatos de dados e possíveis interações com arquivos do sistema (Exemplos: PROFS, IBM-DISOSS); 3) correspondem a sistemas cuja integração ocorre com estruturas de dados comuns e um ambiente de operação comum. Informações podem ser trocadas entre diferentes aplicações mas sem necessidade de troca de contexto (Exemplos: Star da Xerox, Macintosh da

Apple, Lotus Symphony e Lotus 1-2-3); 4) são os sistemas da terceira categoria cuja integração se dá através de um Banco de Dados. Um sistema nessa categoria não é uma coleção de aplicações individuais. Ele provê um conjunto de primitivas através do qual muitas aplicações podem ser construídas (Exemplo: Office-By-Example - OBE).

Futuros sistemas dependerão cada vez mais dos seguintes aspectos: alta capacidade de armazenamento de documentos e vozes, uma boa integração vídeo-teclado que possa ser configurada por software, técnicas de software e hardware que tornem a operação dos sistemas eficiente e aceitável para um largo espectro de usuários.

Seguindo a tendência dos trabalhos mencionados neste capítulo, esta tese versa sobre a utilização de formulários no manuseio de informações armazenadas em Bancos de Dados relacionais, apoiando-se em conceitos pertinentes à manipulação de visões. Visa, em especial, definir uma interface geral que permita aos usuários criar seus formulários e utilizá-los para consultas e/ou atualizações a diversas relações. A organização deste trabalho é a seguinte: a seção 2 apresenta definições e fundamentações; a seção 3 propõe um sistema de edição de formulários a ser implementado, e descreve as operações pretendidas; a seção 4 contém uma descrição do protótipo; e a seção 5 apresenta conclusões e extensões.

2 - Definições e Fundamentações

Esta tese pretende explorar a manipulação de Banco de Dados relacionais através de formulários, com analogias ao problema de manipulação de visões. Para isto, esta seção estabelece as definições necessárias para o entendimento do trabalho.

2.1 - Visões, Objetos e Formulários

- Uma determinada aplicação usa os dados contidos em um Banco de Dados através do que se chama visão. O conceito de visão, de uma forma simplificada, nada mais é do que uma porção do Banco de Dados que pode ser manipulada por uma aplicação, permitindo assim um maior controle de acesso aos dados, visando a sua proteção e consequente integridade, bem como facilidade de manuseio [WIE86] [SET86].

Uma visão, no modelo relacional, pode ser definida como uma relação gerada com consultas sobre as relações base que compõem o Banco de Dados [KEL86], permitindo um mapeamento entre as operações do usuário e as operações no próprio Banco de Dados. A definição de uma visão pode ser feita através da função que a gera relacionada a um determinado esquema. Uma operação solicitada sobre uma visão é dita mapeada para a operação no Banco de Dados e seu resultado é refletido na visão através do mapeamento inverso.

Analisando a correlação entre manipulação de visões e manipulação de objetos, observamos que existe bastante semelhança nas duas situações. Assim como uma visão possui esquema (enumeração de atributos) e materialização (instanciação), objetos possuem uma definição esquemática e ocorrências.

O projeto de Banco de Dados orientado para objetos é uma área de pesquisa que vem sendo considerada de forma especial nos últimos anos [MEY87]. O projeto orientado para objetos pode ser definido como uma técnica que se baseia na decomposição modular de um sistema segundo as classes de objetos que ele manipula.

Soluções funcionais clássicas (como por exemplo o projeto "top-down") exigem que os projetistas inicialmente questionem o que o sistema faz. A solução "top-down" é bastante adequada para casos em que o programa resolve um problema fixo sempre, mas a situação muda quando temos um sistema que ao longo do tempo vai alterando suas funções. Os objetos que um sistema manipula são em geral "pré-definidos", pois pertencem a classes específicas (um sistema operacional por exemplo, manipula dispositivos, memórias, unidades de processamento, etc; um sistema de processamento de documentos manipula documentos, campos, páginas,etc.).

Os objetos permitem a manuseio dos dados em vários níveis de abstração, facilitando o acesso, pois podem ser estruturados internamente de forma bastante complexa. Como só são acessados por operações permitidas, o acesso é controlado e os dados são protegidos contra manipulações que prejudiquem a sua integridade. Outras semelhanças entre objetos e visões estão em [WIE86]. Embora o nível de semelhança seja grande existem algumas características dissonantes que necessitam ser contornadas, como por exemplo, o fato de que as visões são persistentes, mas os objetos não o são até serem explicitamente armazenados. Além disso objetos são manipulados tradicionalmente de forma procedimental enquanto as visões não o são.

Mesmo com linguagens não especializadas podemos desenvolver aplicações sobre Bancos de Dados convencionais usando o conceito de objetos, bastando para tanto prover o SGBD de interfaces específicas. Existem várias metodologias orientadas para objetos para as linguagens CLU, PASCAL, ADA e mesmo FORTRAN [JAC87].

Da fusão dos conceitos de formulário, visão e objeto surge um novo conceito, o de visão-formulário.

Sistemas que manipulam esta nova unidade, visão-formulário, devem possuir 3 itens:

I - Conjunto de relações-base;

II - Conjunto de geradores de visões-formulários, e

III - Conjunto de funções inversas que permitam remapear as visões-formulários nas relações-base.

Esta noção será usada para a definição de um sistema para projeto de formulários, onde um formulário é visto como um objeto definido em função de seu formato e campos. Uma visão-formulário utiliza arquivos (relações-base) que fornecem os dados para o formulário; seu "esquema" é dado pelo formato do formulário (aqui chamado esqueleto). As operações sobre esta visão são aquelas especificadas para o formulário. Para simplificar este texto, as visões-formulários serão de agora em diante chamadas de formulários, sempre que ficar claro pelo contexto que se trata deste tipo de objeto.

2.2 - Linguagens de acesso a Bancos de Dados

Como este trabalho envolve diálogo com sistemas de bancos de dados, vale analisar alguns pontos sobre a evolução e o estado atual de desenvolvimento de linguagens de acesso.

Ultimamente tem ocorrido uma revolução no que se refere a linguagens para computadores, por causa principalmente de dois aspectos: aumento no poder de computação (quantidade e velocidade) e escassez de programadores, implicando na necessidade de linguagens para usuários finais [CHO85].

As linguagens consideradas de Terceira geração, tais como COBOL, PL/I, PASCAL ou C, não atendem a esta situação, principalmente porque produzem sistemas com código fonte extenso, projetados por profissionais especializados e exigem alto custo de implementação, depuração e manutenção.

Linguagens de Quarta geração foram criadas em resposta a esses problemas e são às vezes referenciadas como "linguagens de alta produtividade". Para permitir o emprego de sequências de comandos, tais como encontrados nas linguagens de Terceira geração, emprega-se uma diversidade de mecanismos como, por exemplo, telas interativas (em geral orientadas por "menus") ou sistemas gráficos. Muitos não consideram esses mecanismos como "linguagens de Computação", embora sejam usados para construção de programas.

Muitas das linguagens de Quarta geração são dependentes de um determinado SGBD e seu dicionário de dados. O dicionário de dados em geral armazena, além dos dados sobre os elementos do banco de dados propriamente ditos, informações para sua manipula-

ção, como controles de segurança, tipo de autorizações para acesso, etc.

As linguagens de Quarta geração são bastante diversificadas quanto a sua construção e sintaxe, variando em poder e capacidade. Algumas são meramente linguagens de consulta, outras geradoras de relatórios, e ainda outras permitem programação de alto nível. Enquanto linguagens de Terceira geração podem ser usadas para uma enorme gama de aplicações, as de Quarta geração são projetadas para atender a classes específicas de problemas.

As linguagens de Quarta geração são ditas não procedimentais, contrariamente às de Terceira geração que são procedimentais. Em linguagens não procedimentais os comandos do usuário resultam em solicitações ao sistema que o software traduz para código executável. Linguagens não procedimentais trazem o perigo de tornar o processamento mais caro e demorado que o desejado.

Em comparação com as linguagens de Terceira geração, as de Quarta geração são mais fáceis de usar: por exemplo, existem aspectos pré-definidos (de forma inteligente) que simplificam o trabalho de programação. O fato de operarem sob a forma de diálogo com o usuário permite uma diminuição no tempo necessário ao projeto e depuração. Em contraposição, as linguagens de Quarta geração são mais restritas em opções, perdendo assim em flexibilidade, além de serem de difícil otimização de processamento. Até o presente momento não há uma linguagem de Quarta geração padrão.

Como consequência desta situação, uma solução para a implementação de uma interface com o SGBD seria o uso de uma linguagem de consulta bastante divulgada. Uma das metas no projeto de um

sistema de manuseio de Bancos de Dados através de formulários é a sua portabilidade e conseqüente obtenção de um certo grau de independência entre o código do módulo Manuseador e as características do SGBD hospedeiro.

Uma das linguagens mais aceitas e difundidas do ponto de vista de acesso a Banco de Dados relacionais, com sérias tendências de padronização, é a SQL (Structured Query Language). SQL é uma linguagem de consulta e atualização, que possui várias implementações. Diversos sistemas de Bancos de Dados relacionais oferecem a SQL como uma opção para a recuperação e manipulação de dados armazenados. Considerando estes aspectos, nosso projeto foi orientado para linguagens de acesso com as características de implementação da SQL.

A linguagem SQL pode aparecer disponível em dois tipos de "interfaces": interativa e embutida. No primeiro caso, as respostas às solicitações digitadas pelo usuário através do teclado do terminal são apresentadas através do vídeo. Já na segunda opção, os comandos da linguagem (que deverão estar embutidos num programa de aplicação, escrito em uma linguagem de Terceira geração denominada então de linguagem hospedeira) só serão executados quando o programa for executado e os resultados serão retornados não diretamente para o terminal, mas sim para variáveis do programa. Para a construção de aplicações mais complexas pode-se usar a SQL em conjunto com linguagens de Terceira geração [MAR84] [DAT84].

Podemos encontrar também a opção denominada de SQL dinâmica, que corresponde a um conjunto de facilidades da SQL embutida, que

possibilita a geração de comandos SQL dinamicamente.

Nas aplicações "on-line" típicas, os comandos são construídos considerando-se os seguintes passos:

1- Aceitação de comandos através do terminal.

2- Análise de comandos.

3- Emissão de comandos SQL apropriados para o Banco de Dados.

4- Retorno de mensagens e/ou resultados para o terminal.

Se o conjunto de comandos que o programa pode aceitar é razoavelmente pequeno, o conjunto de comandos SQL possível também será pequeno e poderá estar pré-definido no programa. Neste caso, os passos 2 e 3 consistem simplesmente de lógica que examina os comandos de entrada e desvia para a parte do programa que emite os comandos SQL pré-definidos. Se, por outro lado, existe uma grande variedade de entradas, pode não ser prático pré-definir comandos SQL. Portanto , em geral, é muito mais conveniente construir comandos SQL dinamicamente para serem executados também dinamicamente. Inicialmente o comando é construído no formato fonte sendo depois convertido (compilado e gerado código de máquina) por um comando SQL especial, obtendo-se assim o formato objeto que será executado por outro comando SQL apropriado. A maioria dos comandos SQL pode participar deste processo com exceção do comando SELECT, que se diferencia dos demais pois retorna dados, enquanto os demais retornam somente informações que sinalizam determinados estados (como por exemplo: tupla não encontrada, etc).

A seguir faremos uma breve explanação sobre o conjunto básico de comandos SQL seguida pela descrição da interação da SQL embutida em programas com o SGBD.

A SQL usa estruturas de dados relacionais, onde os dados estão logicamente armazenados na forma de tabelas (isto é, relações). Toda a navegação de dados é feita pela SQL, sendo transparente para o usuário.

As funções da SQL são:

. inserção, eliminação ou alteração de dados.

. recuperação (consulta) de dados. Relações, tuplas, ou partes de tuplas (projeção de atributos) podem ser recuperadas.

Os comandos básicos utilizados são:

. comandos de manipulação de dados - INSERT, UPDATE, DELETE

. comandos de consulta - SELECT

COMANDO INSERT - permite a inserção de tuplas em tabelas. Sua forma básica é:

INSERT

INTO nome de tabela [(atributo [,atributo]...)]

VALUES (constante [, constante]...),

COMANDO UPDATE - pode ser usado para calcular valores e modificar dados de uma relação. Sua forma básica é:

UPDATE nome de tabela SET atributo = expressão [, atributo = expressão]...

WHERE condição

Todas as tuplas que satisfazem a condição são atualizadas.

COMANDO DELETE - pode ser usado para eliminar tuplas de uma relação. Sua forma básica é:

DELETE FROM nome de tabela WHERE condição

Este comando elimina todas as tuplas da relação especificada que satisfazem a condição.

COMANDO SELECT - É o mais importante e é utilizado em consultas e recuperação de dados. Sua forma básica é:

SELECT	nomes dos atributos cujos valores se deseja
INTO	nome de variáveis que conterão os valores desejados
FROM	nome(s) de tabela(s)
WHERE	condições a serem satisfeitas

O comando SELECT embutido requer a cláusula INTO, especificando as variáveis hospedeiras nas quais os valores recuperados do Banco de Dados serão colocados. Variáveis hospedeiras devem ser compatíveis em tipo com as do Banco de Dados.

Outras cláusulas, tal como ORDER BY (para uma classificação, por exemplo), podem ser incluídas. Nem sempre a cláusula WHERE é necessária (quando há recuperação de todas as linhas da tabela, por exemplo). Atributos podem ser recuperados numa ordem diferente daquela em que estão armazenados. Também as variáveis que conterão os valores recuperados serão alocadas dinamicamente, exigindo com isso que a linguagem hospedeira tenha essa característica.

Comandos embutidos geralmente possuem indicação de início e fim para serem diferenciados dos comandos da linguagem hospedeira. Comandos SQL executáveis podem aparecer onde qualquer comando da linguagem hospedeira aparece.

Em algumas implementações, os comandos fontes da SQL dinâmica não devem fazer referências a variáveis hospedeiras, que só serão permitidas no momento da execução. A passagem de dados entre o

Banco de Dados (acessado pela SQL embutida) e o programa hospedeiro é feita através de áreas intermediárias denominadas de "áreas de comunicação". Após qualquer comando SQL ter sido executado, informações são retornadas ao programa através destas áreas de comunicação. Além disto, há indicadores que sinalizam o sucesso ou insucesso na execução do comando. Em princípio todo comando SQL deve ser seguido de um teste sobre estes indicadores.

Dentre os comandos da liguagem SQL utilizados num ambiente "embutido", o comando SELECT é o que exige um tratamento mais complexo. O problema é que a linguagem SQL é na verdade orientada para conjuntos e a maioria das linguagens hospedeiras está programada para manipular um registro de cada vez. O mecanismo que compatibiliza essa dissonância é o CURSOR. Esse mecanismo consiste basicamente em uma espécie de ponteiro que pode ser utilizado para percorrer um conjunto de registros, apontando para um registro por vez. Em outras palavras, a consulta SQL retorna como resultado um conjunto de tuplas na área de comunicação, onde as tuplas podem ser manipuladas individualmente com auxílio do CURSOR. O CURSOR é definido utilizando-se um comando declarativo que possui o seguinte formato básico:

```
DECLARE nome-cursor CURSOR FOR
        SUBQUERY [UNION SUBQUERY].
        [FOR UPDATE OF nome de campos...
          [ORDER BY CLAUSE],
```

Ao declarar um cursor, aquele nome e aquela solicitação são associados permanentemente a ele. Se o cursor for utilizado em alguma condição de atualização, então a cláusula "FOR UPDATE"

deve ser incluída, especificando quais os campos que serão atualizados. Um programa pode ter várias declarações de cursores. A cláusula "ORDER BY" controla a sequência na qual as tuplas serão recuperadas da área via comando "FETCH". Para ambientes onde é utilizada a SQL dinâmica, este comando tem a sua sintaxe modificada para fazer referência ao comando SELECT objeto. A requisição de dados não é executada neste ponto. Esse comando é apenas declarativo. A requisição é executada quando o CURSOR é aberto na parte procedural do programa, via comando OPEN:

OPEN nome do cursor;

Este comando ativa o cursor especificado (ainda não aberto). O conjunto de registros correspondente ao conjunto recuperado pela consulta é identificado e torna-se o conjunto ativo do cursor. Conjuntos ativos são sempre considerados como tendo uma ordenação, daí a importância do conceito de posição. A ordem é definida pela cláusula "ORDER BY" ou é determinada pelo sistema, na ausência da definição do usuário.

O comando FETCH é usado para recuperar registros do conjunto. O formato básico do comando é o seguinte:

FETCH nome do cursor INTO variáveis hospedeiras,

No formato básico os valores recuperados são colocados nas variáveis hospedeiras indicadas pela cláusula "INTO". Quando estamos usando a opção da SQL dinâmica a sintaxe é alterada para compatibilizar com os novos mecanismos de referência às variáveis que armazenarão os valores recuperados.

Devido à necessidade de recuperar um registro por vez o comando FETCH aparecerá normalmente em algum "loop" do programa.

Após processada a consulta, o cursor deve ser fechado via um comando "CLOSE":

CLOSE nome do cursor;

Este comando desativa o cursor especificado que não possuirá mais um conjunto ativo. No caso do cursor ser aberto novamente, será obtido um novo conjunto ativo, provavelmente distinto do anterior, especialmente se os valores referenciados na cláusula de condição do comando SELECT associado ao CURSOR tenham sido alterados entre uma operação de consulta e outra.

Se o cursor X está correntemente posicionado num registro particular do banco de dados, então é possível atualizar ou eliminar o "corrente de X", isto é, o registro no qual X está posicionado com os comandos adequados (UPDATE, DELETE).

Além de permitir operações sobre os dados, muitas interfaces SQL também possibilitam consultas aos esquemas de banco de dados. Para isto, o esquema é armazenado na mesma forma tabular que as relações [DAT84].

Como mencionado neste capítulo, o sistema proposto nesta tese é baseado em manipulação de objetos do tipo visão-formulário. Sua interface com um SGBD aproveita noções da filosofia de funcionamento da SQL, com utilização de cursores e áreas de comunicação.

A seguir faremos uma descrição deste sistema, quanto à sua arquitetura e operações pretendidas. Poderemos observar que é utilizada a idéia de áreas intermediárias como interface entre o sistema e o Banco de Dados, sendo utilizados elementos que permitem a navegação nestas áreas.

3 - Descrição de um Sistema de Edição de Formulários

Com base no já exposto, propomos a seguinte arquitetura para um sistema de edição de formulários (SEF), baseado no novo conceito de visão-formulário (Figura 3.1).

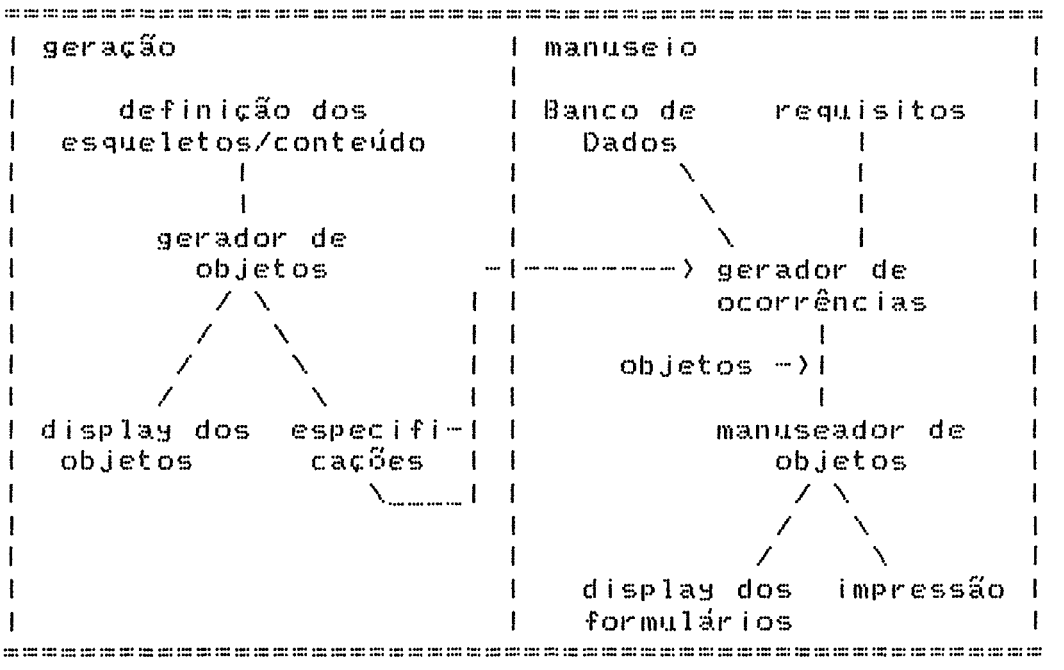


Figura 3.1- Arquitetura proposta do Sistema de Edição de Formulários.

Descrição geral:

O Sistema SEF será composto principalmente de 2 grandes módulos:

- Gerador de esqueleto
- Manuseador de formulários

Características gerais:

- Gerador de esqueletos:
  - flexibilidade na definição do tamanho da página do documento;
  - utilização de "layouts" já cadastrados;
  - digitação livre : carta
    tabela
    rótulos de campos
    títulos
    traços;
  - definição de campos de dados;
  - controle da tela como uma janela que se desloca para cima e para baixo, ao longo do formulário, apresentando 20 linhas por vez;
  - "refresh" da tela quando necessário (por exemplo, quando mensagens do sistema ou do operador prejudicam os dados apresentados na tela).
- Manuseador de formulários:
  - escolha livre dos arquivos de dados a serem manuseados através de um formulário;
  - controle da tela como uma janela que se desloca para cima e para baixo, ao longo do formulário, apresentando 20 linhas por vez;
  - "refresh" da tela quando necessário;
  - consultas e atualizações, através de formulários, a arquivos de dados.

O sistema **SEF - Sistema de Edição de Formulários** é uma ferramenta para automação de escritórios que separa a especificação do formulário do seu manuseio. Isto faz com que este sistema se torne bastante versátil e facilite a sua portabilidade, permitindo que as rotinas de manipulação possam ser desenvolvidas baseadas em modelos distintos do modelo de geração. Um formulário passa assim a ser um objeto, cujas características são dadas pelo conjunto de operações de geração e manipulação. Se o sistema fosse implantado em uma empresa que possui suas informações registradas num Banco de Dados distribuído poderíamos ter uma estação geradora de esqueletos de formulários que seriam manipulados pelas demais estações, o que facilitaria o trabalho dos analistas de O&M e traria uma padronização para a documentação da empresa, independente tanto do modelo de dados usado em cada estação, como da forma de armazenamento. De agora em diante, portanto, entende-se um formulário como um objeto complexo, que fornece ao usuário uma visão formatada de dados armazenados. Ao "esquema" desta visão-formulário denominamos Esqueleto.

O projeto de formulários "on-line" também elimina todo o processo convencional de desenho de formulários já que será assistido por rotinas gráficas simplificadas, que possibilitam o manuseio da tela com a máxima flexibilidade até a obtenção da "arte-final".

A presente proposta se concentrará nas fases de geração e manipulação de visões-formulário, bem como descrição da interface com o banco de dados, não levando em conta a utilização de rotinas gráficas especiais, que podem ser consideradas numa etapa

posterior.

3.1 - Geração do esqueleto

3.1.1 - Composição

Por esqueleto de um formulário entende-se a disposição física da informação. No sentido comum, um esqueleto corresponde a um formulário em branco (o layout). Seguindo a analogia com visões, um esqueleto corresponde ao esquema de uma visão.

Da mesma forma que um formulário comum pode exigir uma ou mais páginas para representar o seu conteúdo, teremos uma ou mais telas para representar um esqueleto.

Existem basicamente dois tipos de esqueletos:

1 - Para formulário simples:

Neste caso o esqueleto do formulário conterá dados de uma única relação, podendo ainda ser classificado em 2 tipos:

a - Formulário-tupla -

Apresenta dados de uma única tupla de uma relação.

b - Formulário-tabela -

Apresenta dados de várias tuplas de uma relação ao mesmo tempo.

2 - Para formulário composto:

Neste caso o esqueleto do formulário será constituído de áreas, onde cada área conterá dados de uma única relação, possuindo as mesmas características de um esqueleto para formulário simples. Também podemos encontrar áreas que não se refiram a nenhuma relação, isto é, que só contenham textos.

Para proporcionar maior flexibilidade na especificação do esqueleto de um formulário, podemos associar a uma determinada área um esqueleto de um formulário já definido, bastando para isto indicar a identificação do esqueleto desejado e o seu posicionamento dentro do novo formulário. O esqueleto assim composto é um objeto complexo que incorpora outros objetos (esqueletos) e suas características.

Dentro de cada área ou esqueleto de um formulário simples podemos encontrar os seguintes elementos:

. Campos de dados - são os locais reservados para entrada ou apresentação dos dados. Cada campo de dados possui basicamente as seguintes características:

- . identificação : nome do campo
- . tipo de dado : inteiro, caractere
- . tamanho
- . máscara
- . posição dentro do formulário

. Linhas - são especificadas como texto comum e podem ser de qualquer tamanho. Podem ser:

- . horizontais
- . verticais

. Texto - correspondem ao título do formulário, aos rótulos dos campos de dados, instruções, etc.

A Figura 3.2 representa o esqueleto de um formulário composto, contendo três áreas, correspondendo a um formulário utilizado no setor de pessoal de uma empresa, projetado para apresentar a relação de funcionários de uma das lojas da empresa.

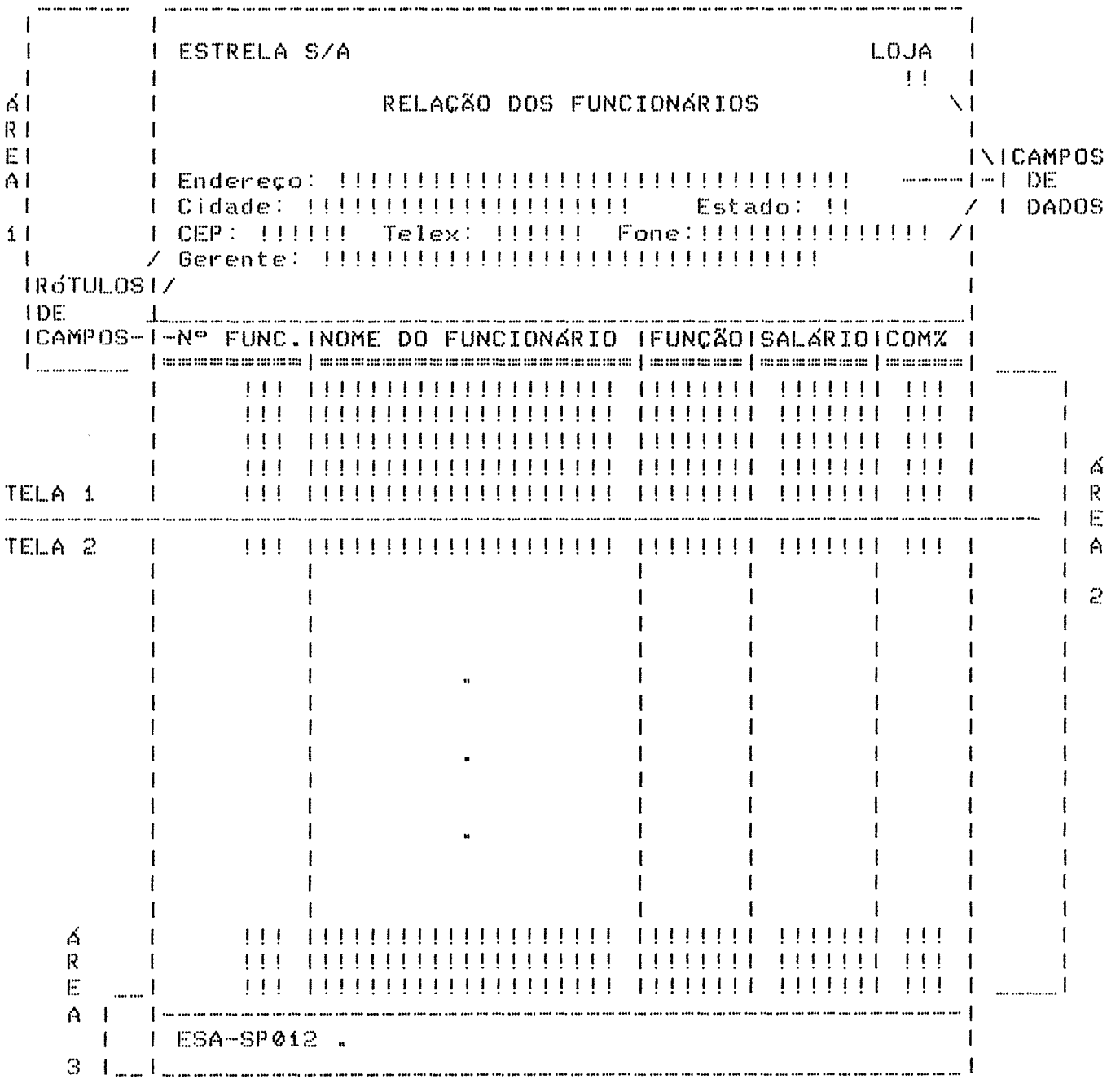


Figura 3.2 - Esqueleto de um formulário.

A Área 1 (Figura 3.2) corresponde ao esqueleto de um formulário-tupla, onde são apresentados apenas dados de uma única tupla por vez (possivelmente referente a uma relação que contém dados das lojas de uma empresa). Área2 corresponde ao esqueleto de um formulário-tabela onde temos os dados de várias tuplas ao mesmo tempo, extraídos de outra relação com dados de funcionários. A Área 3 é do tipo texto, contendo a identificação do formulário, dado imprescindível ao setor de documentação de qualquer empresa.

Outras características podem ainda ser especificadas para a definição de um campo de dados, como por exemplo:

- Indicação de que o valor do campo depende de um dado externo, caso em que uma das seguintes alternativas pode ocorrer:
  - é um valor que segue uma sequência
  - não deve ultrapassar o valor externo
  - Valor do campo é calculado por uma função
- Indicação de critérios de consistência pelos quais o valor do campo deve passar (por exemplo, deve respeitar domínios específicos)
- Indicação para o preenchimento do campo:
  - obrigatório
  - opcional
- Indicação de que possui ou não valores "defaults".
- Definição de rotinas associadas, chamadas de procedimentos automáticos, com indicação de quando devem ser executadas: por exemplo, antes ou depois de cada consulta/atualização ou de qualquer outro evento, à semelhança de "triggers"

(procedimentos cuja execução depende da ocorrência de eventos).

3.1.2 - Geração

O Gerador de esqueletos atende à atividade de projeto de formulários e facilita o processo de "desenho". Durante a Geração do esqueleto teremos a discriminação de todo o "layout", isto é, nome do formulário e seus componentes, como por exemplo: molduras, cabeçalhos, campos, rótulos de campos, disposição gráfica e dimensões, através de uma linguagem de especificação. O esqueleto do formulário é formatado interativamente.

A Geração terá duas saídas: a primeira é a tela formatada segundo o formulário e a segunda são arquivos contendo as especificações do formulário, que serão utilizadas pelo gerador de ocorrências, quando da manipulação dos formulários (preenchimento do esqueleto).

Ao final de uma sessão de trabalho, já com todo o objeto definido, pode-se incluir ou não a especificação do esqueleto nos arquivos de "layouts" de formulários. A responsabilidade quanto à coerência e apresentação do formulário é inteiramente do usuário.

A linguagem de geração utilizada pelo SEF foi dividida em 2 tipos de comandos:

1 - Para geração dos formulários;

2 - Para apresentação dos formulários na tela.

As operações permitidas para geração do esqueleto são:

1. Definição de textos.

1.1- Instruções, observações, etc;

1.2- Linhas horizontais e verticais, molduras;

1.3- Definição de cabeçalhos do formulário e rótulos dos campos em qualquer posição do formulário e com qualquer tamanho.

2- Utilização de 'layout' já implantado para ser inserido como parte de um outro esqueleto, bastando para isso indicar o local no novo formulário em que ele ficará e o nome do 'layout' desejado (utilização de objetos já definidos).

3- Definição de campos que corresponderão aos atributos das relações que serão manuseadas através deste formulário. Esta definição será usada quando da materialização de ocorrências do objeto.

Na geração do esqueleto também deve ser especificado quais tipos de operação podem ser aplicados através daquele formulário, durante o manuseio, se consulta e/ou atualização, para atender a exigências de integridade. Por exemplo, quando do manuseio temos o caso de desejarmos inserir uma tupla numa determinada relação e não existirem no formulário todos os campos obrigatórios para a geração da mesma. Caso se deseje proibir esta operação o formulário deverá ser gerado sem a indicação de que são permitidas operações de inserção.

3.2 - Manuseio do formulário (visão-formulário)

Tendo em vista que o SEF permite ver dados através de formulários, isto nada mais é que uma nova maneira de apresentar visões relacionais ao usuário.

O conceito de visão corresponde ao resultado de uma consulta, apresentada sob forma de tabela. Como visto no capítulo 2, visão-formulário é uma visão relacional que ao invés de ser apresentada em forma de tabela, é formatada em um formulário pré-definido.

Para falarmos sobre o manuseio de dados segundo formulários, é preciso analisar, principalmente, os seguintes aspectos:

a - dados a serem manuseados;

b - interação entre usuário, esqueleto e dados, e

c - operações permitidas.

As informações a serem manuseadas através dos esqueletos estarão em relações, as chamadas relações-base. Existem várias maneiras de se selecionar essas informações, ou seja, através do uso dos conceitos de:

. instantâneos ("snapshots")

. arquivos auxiliares

. visões

. diretamente a partir das relações

Cada uma dessas soluções tem suas características com suas facilidades e limitações. Esta tese optou pela última solução, onde o acesso às relações é intermediado por um Gerenciador de "buffer" (vide 3.2.3).

Ao passo que a maioria dos sistemas existentes fixa formulá-

rios a relações, esta tese considera que existe liberdade de escolha na requisição do usuário quanto ao esqueleto utilizado e a relação a ser manuseada. Esta associação é feita na hora do manuseio, e não quando da geração do esqueleto. A ligação entre campos de um formulário e atributos de uma relação é feita através dos nomes dos campos definidos durante a geração do esqueleto. Em outras palavras, se um esqueleto tem o seu terceiro campo situado à linha 12, coluna 3, especificado como estando relacionado ao atributo NOME-EMP, então, qualquer atributo chamado NOME-EMP poderá ser "visto" através deste campo, obedecidas as restrições a seguir.

Por exemplo, seja F1 o esqueleto solicitado (correspondendo a um formulário simples) contendo os campos denominados A1, A2, A3 e A4, e seja R a relação solicitada para preenchimento do formulário. Teremos os seguintes casos a considerar (onde atende indica que a formatação será atendida e rejeita que será rejeitada).

1 - Os atributos do esquema R correspondem exatamente aos campos do esqueleto F1, idênticos em nome, ou seja, R = (A1, A2, A3, A4), então teremos:

1.1 - Especificações são idênticas (tamanho e tipo): atende

1.2 - Especificações diferem em:

1.2.1 - Tipo dos dados (p. ex., inteiro)

a - Tipo do formulário mais abrangente que o da relação, isto é, engloba todas as características (p. ex., o tipo caractere em relação ao numérico): atende

b - Tipo do formulário mais restrito ou incompatível: rejeita

1.2.2 - Tamanho

Se o tipo admite a diferença existente, isto é, se houver necessidade de acrescentar brancos ou zeros ou então ocorrer truncamento sem alteração do significado do campo: atende

Caso contrário: rejeita

2 - O número de atributos do esquema R é o mesmo que o de campos do esqueleto F1, mas com um ou mais nomes diferentes: rejeita

3 - O número de atributos do esquema R é menor do que o de campos do esqueleto F1: rejeita

4 - O número de atributos do esquema R é maior do que o de campos do esqueleto F1, mas existe correspondência biunívoca entre todos os campos de F1 e atributos de mesmo nome do esquema R: atende

Esta característica de independência entre esqueleto e dados é garantida através de uma rotina que faz a consistência entre campos do esqueleto e atributos dos arquivos solicitados.

Esta flexibilidade de associação entre esqueleto e relação no manuseio é exigida em muitas situações práticas, como por exemplo:

a - **Sistemas distribuídos** - onde encontramos vários nós contendo arquivos com definições idênticas, mas com diferentes identificações que, entretanto, precisam ser manuseados pelo mesmo grupo de formulários para atender, princi-

palmente, à padronização da empresa.

b - **Projeto auxiliado por computador** - neste caso existe um aspecto bastante peculiar que se refere ao problema de manuseio de versões e do processamento histórico.
Considerando que, em projetos que requerem ciclos, informações antigas não podem desaparecer, então surge o problema de fornecer a mesma visão para diferentes versões de um mesmo arquivo.

c - **Atividades comerciais ou científicas** - nas situações onde existem arquivos de acompanhamento (fiscalização), de segurança (backups) ou mesmo arquivo morto, onde há a necessidade de recuperação das informações ao longo do tempo. Existem também casos característicos de aplicações que exigem a coexistência de vários arquivos idênticos na sua especificação, mas que representam uma situação dinâmica. Este fato é comum na área comercial, nos sistemas de contabilidade, com seus arquivos de movimentos diários ou mensais, ou então nas aplicações científicas de coletas de dados para controle (sistemas de controle de processos), e outros.

Também para obtermos uma maior flexibilidade no manuseio dos dados, o ideal seria que a cada formulário pudéssemos associar mais de uma relação-base, mas esta situação não ocorre na maioria dos sistemas atuais.

Como nossa solução permite o uso de áreas (com cada uma associada a uma relação), como componentes de um formulário, obtendo assim um formulário composto, parte deste objetivo é atingido. Em

outras palavras, permite-se o uso de várias relações-base para uma única visão-formulário.

Apesar do sistema permitir a existência de formulários compostos (contendo mais de uma área), o manuseio é feito a nível de cada área. Desta forma, é como se cada área correspondesse a uma visão sobre uma relação.

Quanto às macro-operações que podem ser aplicadas aos formulários, temos:

. Consultas

. Atualizações (Inclusões, Eliminações e Modificações)

Todas estas operações são processadas com intermediação de um Módulo Gerenciador de buffer (vide seção 3.2.3).

3.2.1 - Consultas

Correspondem ao resultado do preenchimento de um esqueleto com as informações das relações (ou seja, casamento do esqueleto com o conteúdo das relações).

No processamento de consultas temos várias operações permitidas no formulário preenchido, como por exemplo:

- tupla seguinte/anterior;
- grupo seguinte/anterior (para o formulário tabela).

3.2.2 - Atualizações

Embora a contribuição da tese esteja na maleabilidade de manuseio de dados através de uma interface de formulários, os problemas existentes na área de atualização de visões escapam ao âmbito deste trabalho. Por este motivo, decidiu-se limitar o manu-

seio da visão-formulário a visões que correspondem a uma única relação-base (ambiente da área em uso). Suprimem-se desta forma vários problemas de mapeamento de atualização em visões, que aparecem quando uma visão é formada a partir de mais de uma relação-base [KEL86].

As atualizações serão feitas sobre visões-formulários. Eliminação de tuplas e modificações de valores (atributos) serão feitos sobre visões-formulários previamente materializadas através de uma consulta. A inserção de tuplas vai exigir a apresentação de esqueletos na tela para serem preenchidos pelo usuário.

Em função do exposto, e devido às simplificações feitas com relação à interação entre relações e esqueletos, não serão discutidos problemas conceituais adicionais da atualização através de visões-formulário (como por exemplo, de semântica de atualização ou de interferência entre visões formadas a partir de várias relações). Trata-se apenas de, a partir de dados fornecidos, atualizar relações em Bancos de Dados. A única preocupação da interface do SEF será a de formatar as tuplas a serem atualizadas para em seguida emitir o pedido de atualização ao SGBD. Na versão implementada não há preocupação com a interpretação da semântica da atualização. A validação da execução de uma atualização (se não foi proibida quando da definição do esqueleto) é determinada pelo SGBD hospedeiro, bem como seu mapeamento para as relações-base.

Para a atualização teremos 3 situações a considerar:

. Inclusão -

O usuário solicita o formulário (esqueleto) adequado e preenche os campos necessários.

. Eliminação -

O usuário solicita o formulário (esqueleto) adequado e indica a chave das tuplas a eliminar.

. Modificação -

O usuário solicita o formulário (esqueleto) adequado e após proceder a uma consulta, altera os atributos desejados.

Em qualquer dos três casos o usuário no final confirma a alteração ou desiste de toda a transação. Em caso de confirmação, as atualizações são solicitadas ao Banco de Dados.

No Anexo A apresentamos uma breve orientação sobre o modo de utilização do protótipo.

3.2.3 - Manipulação do buffer

A interação entre o sistema e o SGBD ocorre com o auxílio de uma área intermediária. Esta área intermediária funciona como um grande "buffer", com a finalidade de otimizar o acesso ao SGBD, facilitando as consultas e atualizações, sem prejudicar ou perder as características necessárias inerentes ao uso de um Banco de Dados. Entende-se de agora em diante por "buffer" a área alocada para manipular requisições relativas a uma relação. Assim, embora várias relações possam ser manipuladas através de uma única área intermediária alocada, considera-se que esta área é dividida em tantos buffers quanto o número máximo de relações que possam ser acessadas em uma dada consulta pelo SEF.

O mecanismo do buffer também permite fazer recuperações parciais de informações, bem como atender rapidamente consultas do

tipo 'próxima tupla' ou 'próximas n tuplas', podendo ser entendido como uma simulação de um mecanismo semelhante à materialização de visões. Com este mecanismo, o usuário pode efetuar várias operações sobre um mesmo conjunto de tuplas sem que para isso seja necessário efetuar vários pedidos ao SGBD. Por exemplo uma consulta (visão-formulário) que só utilize 3 atributos de uma relação contendo 20 atributos (projeção), necessitará armazenar no buffer apenas o resultado da projeção, em cima do qual o usuário trabalhará. Para o usuário, no entanto, tudo funciona como se trabalhasse diretamente com o Banco de Dados.

Com base no já exposto obtemos a seguinte definição esquemática para o manuseio dos dados (Figura 3.3):

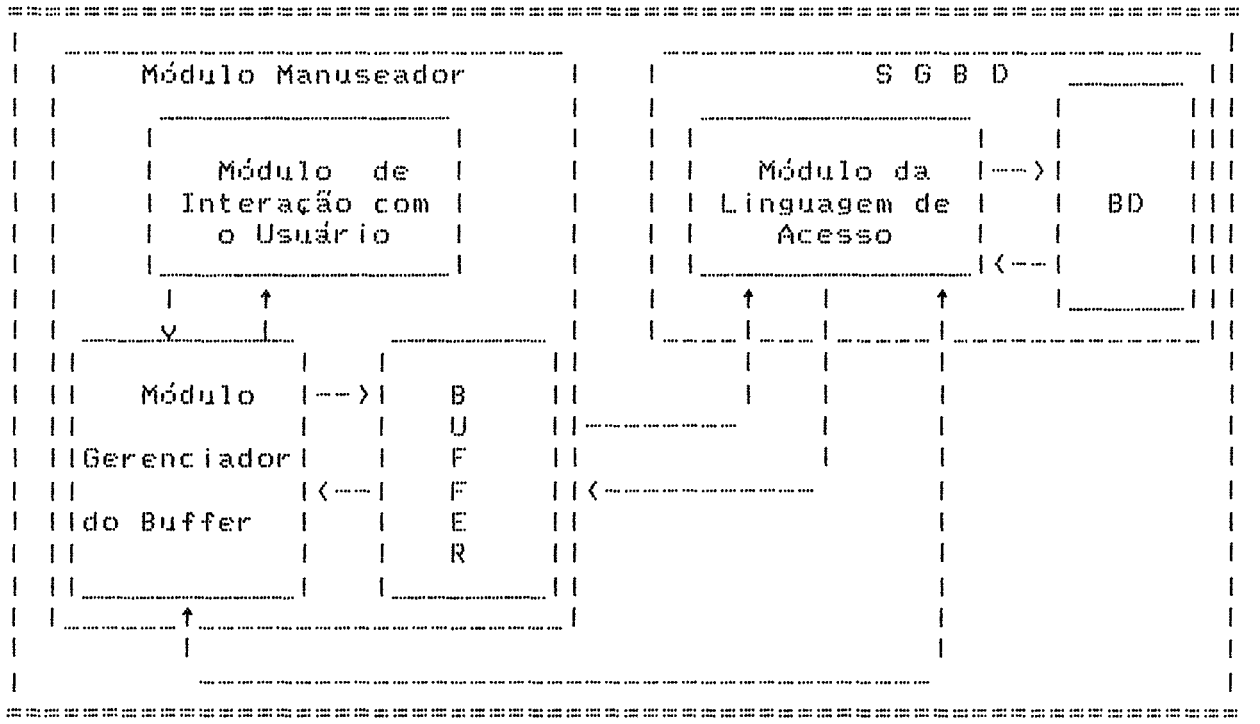


Figura 3.3 - Diagrama do manuseio das relações.

A seguir descrevemos a dinâmica dos relacionamentos entre os

módulos indicados na Figura 3.3.

As operações de eliminação e modificação de tuplas são sempre precedidas por consulta. Por este motivo o pré-processamento destas operações será discutido no âmbito de "consulta". Operações de inserção não são necessariamente precedidas de consulta.

Suponha que o usuário vai proceder a uma operação de consulta (por exemplo, se deseja acessar a relação R_{α} através de um formulário que especifica os atributos B, C, D). Sua solicitação é analisada pelo Módulo de Interação (para ver, por exemplo, se a área permite a operação). Se a operação for válida para a área, a solicitação é passada para o Módulo Gerenciador do Buffer que formaliza a requisição ($\pi_{BCD} R_{\alpha}$) para o SGBD, e executa as ações necessárias ao preenchimento do buffer com o resultado da consulta e conseqüente indicação para o usuário de seu sucesso ou insucesso. A operação de inserção é processada pelo Módulo de Interação juntamente com o Módulo Gerenciador do buffer,que a transmite para o módulo de acesso do SGBD.

As ações executadas pelo Módulo Gerenciador do buffer envolvem definição e manipulação de cursores, controle da sequência e periodicidade da solicitação de "fetches" para o Módulo da Linguagem de Acesso. Além disso, o Módulo cuida da emissão de comandos para garantir a efetivação das atualizações no Banco de Dados (COMMIT) durante as operações de alteração de dados, e da comunicação entre os Módulos de Interação e de Linguagem de Acesso através de códigos de controle.

O resultado de qualquer solicitação ao Banco de Dados é acompanhado de códigos de controle, que são analisados pelo Gerencia-

dor do buffer e transmitidos ao usuário, quando cabível. Estes códigos incluem: código de erro, que indica o sucesso/insucesso de operações; código de processamento de atualização, que fornece informação sobre a viabilidade de proceder a uma atualização (ou consulta para posterior atualização), por exemplo quando uma tupla está trancada por outra transação (isto é, inacessível até ser liberada por quem solicitou a tranca ou "lock"). Os pedidos de atualização exigem que a interface com o SGBD solicite tranca dos registros envolvidos, para prevenir colisões e ocorrências prejudiciais ao Banco de Dados.

Vários problemas podem ocorrer na manipulação de informações através de formulários. Alguns deles são citados a seguir:

- Inserção de tupla através de formulário que não contém todos os atributos obrigatórios. Neste caso a operação poderá ser rejeitada pelo SGBD.
- Mapeamento de consultas, de forma a permitir flexibilidade à solicitação do usuário.
- Mapeamento ambíguo de atualizações para o SGBD. Isto ocorre sempre que um pedido de atualização pode levar a uma situação ambígua na determinação das tuplas a serem atualizadas durante o remapeamento. Neste caso, uma solução é que o responsável pela definição do esqueleto deve recusar a atualização (por exemplo, modificando a projeção dos atributos para o esqueleto de modo que se elimine a situação ambígua). O Módulo Manuseador especificado não prevê a solução desse tipo de problema.

. Determinadas pesquisas de registros do tipo anterior/próximo podem exigir muitas operações de entrada/saída no buffer, caso o buffer não comporte todas as tuplas referentes a uma consulta de uma única vez.

O Módulo de Interação com o usuário manipula os dados do buffer, através do Módulo Gerenciador, atualizando-os se necessário. Quando o usuário se dá por satisfeito, o Módulo de Interação emite comando do tipo COMMIT para efetivar as atualizações. É também possível que o Módulo Gerenciador indique essa necessidade independentemente do Módulo de Interação, principalmente se já foi excedida a capacidade do buffer e novas informações são necessárias. O Módulo Gerenciador do Buffer envia uma atualização para o SGBD que retornará código de controle, indicando sucesso ou insucesso, que será analisado e apresentado ao usuário. As efetivações podem ser processadas em bloco ou individualmente dependendo do tipo de interface com o SGBD. A princípio para cada tupla atualizada no buffer é colocado um código, indicando que a tupla foi alterada/eliminada/inserida, que será utilizado quando da efetivação da operação.

Este capítulo descreveu os Módulos que compõem o sistema de Formulários e a interação entre eles. O capítulo que se segue discute as estruturas de dados utilizadas para atender as operações especificadas e a implementação do protótipo.

4 - Descrição do Protótipo

Este capítulo descreve as estruturas de dados que foram utilizadas para a implementação do protótipo do SEF. O SEF foi implementado em ambiente PC, em linguagem Pascal e contém 2800 linhas de código fonte com 40 kb de código executável. O protótipo implementado consiste tanto do Gerador quanto do Manuseador de Formulários, sendo a interface com o Banco de Dados simulada através da colocação de dados em arquivos. Os dados foram lidos de arquivos previamente definidos e manipulados com o auxílio do Gerenciador de "buffers".

4.1 - Descrição das áreas de trabalho e arquivos utilizados na geração do esqueleto.

Durante a fase de edição de um formulário (geração do esqueleto) são usadas as seguintes estruturas:

4.1.1 - Estrutura que representa o "layout" do formulário que está sendo editado (mapa do formulário):

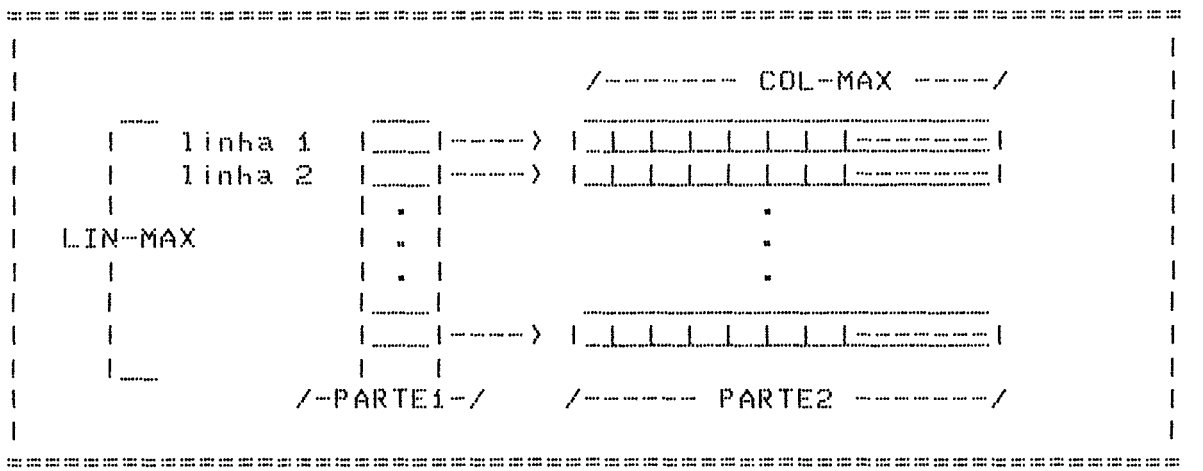


Figura 4.1 - Esquema do mapa do formulário.

onde a PARTE1 controla a sequência das linhas do texto e a PARTE2 contém o texto propriamente dito, sendo LIN-MAX o número máximo de linhas que um formulário pode ter e COL-MAX, o número máximo de colunas.

Ao lado dessa estrutura encontramos diversas variáveis de trabalho que auxiliam o controle da mesma e da sua relação com a parte visível através da tela, como por exemplo:

. LIN-FINAL

  - aponta para a última linha já criada no formulário.

. N-LIN, N-COL

  - indicam qual o tamanho da página do formulário em linhas e colunas, respectivamente.

. LIN-INICIO-TELA

  - primeira linha do formulário que aparece na tela.

. LIN-CURSOR, COL-CURSOR

  - posição do cursor dentro da área de trabalho da tela.

. LIN-CURSOR-MAPA, COL-CURSOR-MAPA

  - posição (linha/coluna) no formulário correspondente ao local onde se encontra o cursor na tela, e podemos definir as seguintes relações:

  LIN-CURSOR-MAPA := LIN-INICIO-TELA + LIN-CURSOR -1

  COL-CURSOR-MAPA := COL-CURSOR

Observações:

Esta estrutura se torna bastante adequada para o manuseio do

formulário principalmente quando existe inserção, troca de posição ou eliminação de linhas, que implica apenas no manuseio da "PARTE1" da estrutura, relocando os apontadores que indicam as linhas.

4.1.2 - Estrutura que controla o manuseio das áreas e campos de dados:

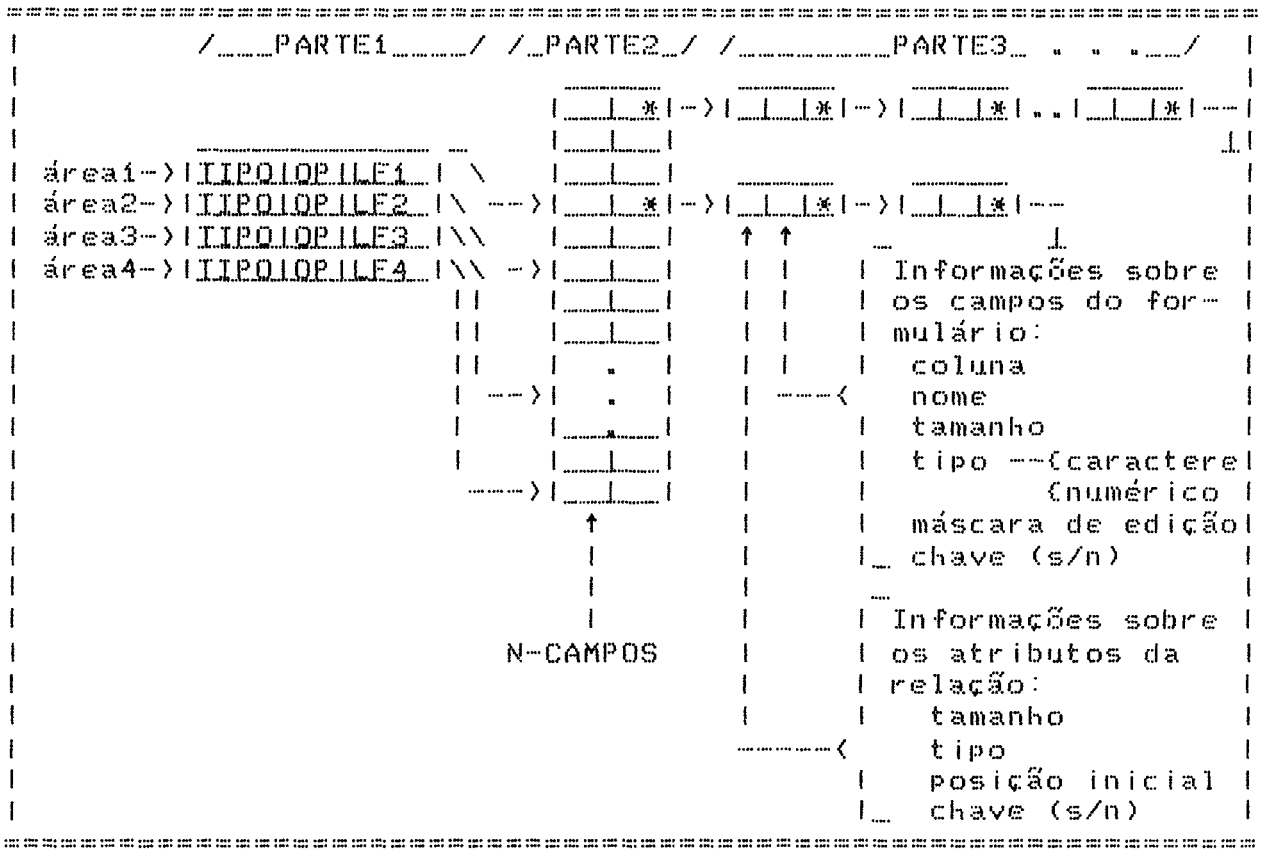


Figura 4.2 - Estrutura de controle do manuseio de áreas e campos.

PARTE1 controla as informações sobre as áreas (para o protótipo teremos no máximo quatro), PARTE2 controla a localização dos campos nas diversas linhas do formulário e PARTE3 contém informações sobre os campos, sendo que a lista é mantida na sequência em que os campos de dados se encontram na linha. As informações so-

bre os atributos das relações só serão preenchidas quando do manuseio do formulário. Ao ser requisitado o acesso às relações, as informações sobre o esquema (isto é, informações sobre os atributos da relação) são solicitadas ao SGBD, e a partir destas informações o SEF se encontra pronto para processar qualquer operação sobre os dados a serem recuperados.

TIPO - indica qual o tipo de área, isto é, se é texto, tupla ou tabela.

OP - indica as operações que poderão ser aplicadas na área (consulta, inclusão/alteração/eliminação).

LF1, LF2, LF3 e LF4 - indicam as últimas linhas do que seria a área1, área2, área3 e área4, respectivamente, no formulário.

N-CAMPOS - indica quantos campos de dados estão definidos para aquela linha.

Além de haver uma íntima ligação entre o controle dessa estrutura e da anterior, também temos algumas variáveis auxiliares:

N-AREAS - número de áreas já definidas.

AREA-USO - indica qual a área que está em uso.

AREA-ANT - indica qual a última área manipulada.

4.1.3 - Arquivos

Após a conclusão da edição de um formulário, serão gerados/atualizados dois arquivos:

4.1.3.1 - Arquivo do 'layout' do esqueleto

- Denominado IDENT.ESQ, onde IDENT é a identificação do formulário e ESQ é uma extensão padrão. Contém informações da 1a.

estrutura de dados (Figura 4.1). Corresponde a um arquivo texto representando o "layout" no que se refere à parte visual (Figura 4.3). Serão desprezadas linhas em branco, tornando-se necessária a inserção de controles (salto de página, etc) no início de cada linha ou registro gravado, que serão utilizados quando da impressão dos formulários.

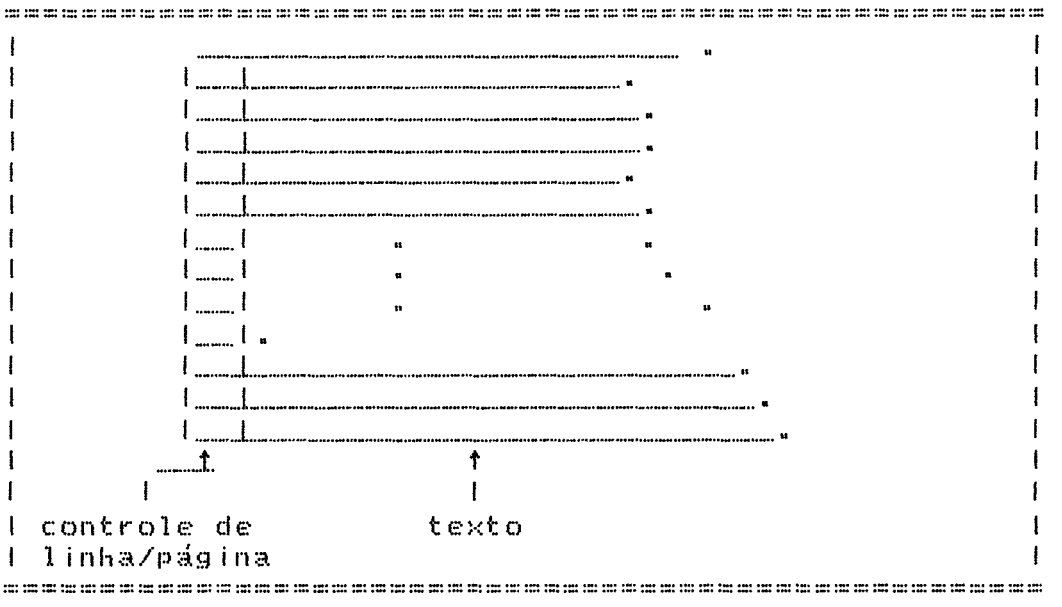


Figura 4.3 - Esquema do arquivo do "layout" do esqueleto

4.1.3.2 - Arquivo de identificação do esqueleto

- Denominado IDENT.IES, onde IDENT é a identificação do formulário e IES é uma extensão padrão. Contém informações da segunda estrutura de dados (Figura 4.2). Possui um registro de controle (Figura 4.4) com todas as informações gerais sobre o formulário, e uma série de registros sobre todos os campos de dados (Figura 4.5), onde a primeira posição indica qual a área a que ele pertence.

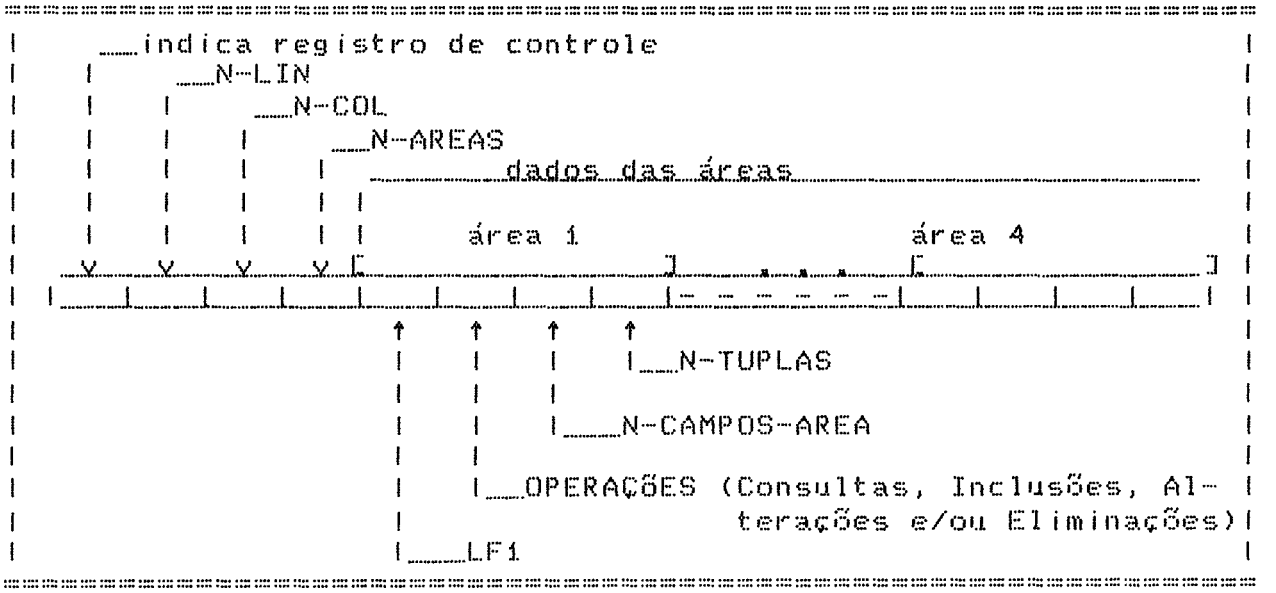


Figura 4.4 - Esquema do registro de controle

N-TUPLAS - indica o número de tuplas a apresentar:

0 => área texto;

1 => área tupla;

N => área tabela.

N-CAMPOS-AREA - nº. de campos associados àquela área.

OPERAÇÕES - tipos de operações permitidas nesta área.

LF1 - última linha da área no formulário.

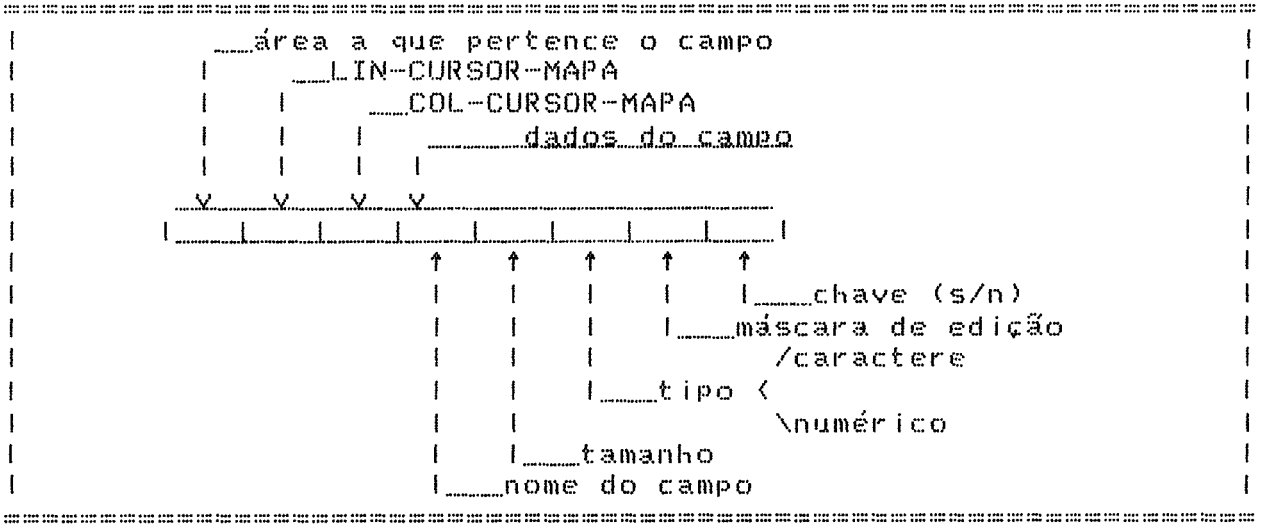


Figura 4.5 - Esquema dos registros dos campos de dados.

Observação:

Na área tabela só existe a definição do registro correspondente à primeira linha da tabela.

4.2 - Descrição das áreas de trabalho utilizadas na rotina de consistência entre a solicitação do formulário e as relações a serem manuseadas.

Para maior clareza descreveremos as estruturas através do seu manuseio.

Para cada área associada a uma relação teremos:

1º Passo

- Armazena todas as informações sobre os campos do formulário associados à área, classificando pelo nome do campo, conforme estrutura definida na Figura 4.6.

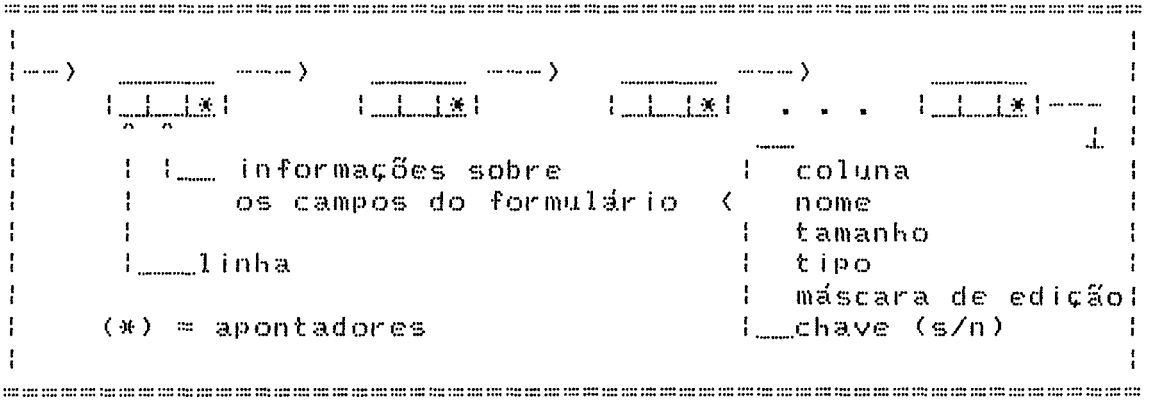


Figura 4.6 - Esquema da estrutura do 1º. Passo.

2º Passo

- Armazena todas as informações sobre os atributos da relação, classificando pelo nome do atributo, numa estrutura segundo o esquema da Figura 4.7. Estas informações são obtidas como consequência de consultas ao dicionário do banco de dados.

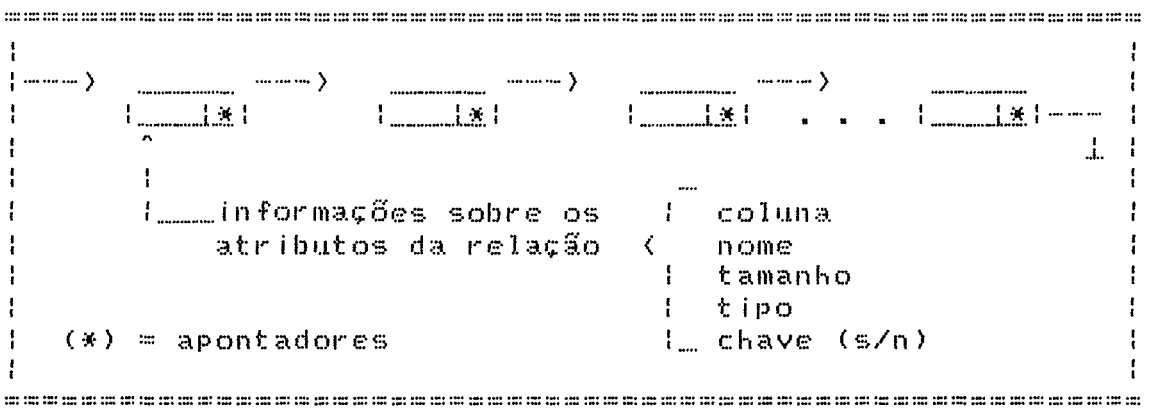


Figura 4.7 - Esquema da estrutura do 2º. Passo.

3º Passo

- Critica a aridade/compatibilidade de características entre os campos do formulário e atributos da relação, eliminando da estrutura do 2º Passo (Figura 4.7) 'nós' desnecessários. Em caso de erro, emite mensagem e cancela rotina. Caso contrário, ao término da crítica de cada área, da estrutura do 1º Passo (Figura 4.6) obtemos a linha e a coluna onde se encontra cada campo no formulário. Podemos então transferir as informações sobre os atributos da relação que se encontram na estrutura do 2º passo (Figura 4.7), para a estrutura da memória que controla os dados de áreas e campos. A partir deste momento, o sistema passa a processar as várias funções de consulta à relação. No protótipo implementado, a função de consulta foi simplificada de forma a prever apenas as seguintes situações: se o número de atributos for menor que o do esquema, trata-se de uma projeção. Senão, especifica-se uma seleção.

Essas duas estruturas são utilizadas apenas durante o teste de compatibilidade. Durante as operações de consultas/inclusões/eliminações e alterações só serão utilizadas as duas estruturas

principais, isto é, a que contém o texto propriamente dito e a que controla áreas/campos, em interação com o Módulo Gerenciador do Buffer.

Na verdade essa solução foi projetada considerando-se Sistemas Gerenciadores de Bancos de Dados bastante simples, devido ao ambiente de desenvolvimento do protótipo ser em microcomputador. Em sistemas mais evoluidos, com as facilidades encontradas na linguagem de acesso quanto à obtenção de informações sobre o dicionário de dados, a segunda estrutura (Figura 4.7) seria desnecessária.

4.3 - Descrição das áreas de trabalho e arquivos utilizados no manuseio do formulário.

As principais estruturas/arquivos a serem usados no manuseio do formulário são:

- arquivos relacionados ao esqueleto:
  . IDENT.ESQ
  . IDENT.IES (vide descrição na geração do esqueleto).
- estrutura relacionada ao "layout" do formulário, mapa do formulário, (vide descrição na geração do esqueleto).
- estrutura relacionada às áreas e campos do formulário, (vide descrição na geração do esqueleto).
- interface com o SGBD, área de "buffer" definida para facilitar o diálogo entre o sistema e o banco de dados (vide seção 3.2.3).

4.4 - Exemplos da utilização das estruturas de dados.

Nesta seção será demonstrado o relacionamento entre o projeto e o uso de um formulário com as estruturas de armazenamento utilizadas no sistema. Será considerado o mesmo exemplo utilizado na descrição do SEF (Figura 3.2), ou seja, o de um formulário contendo uma relação dos funcionários de uma determinada loja pertencente a uma rede de lojas, aqui denominada Estrela S/A.

Como já foi dito, este formulário é composto de tres áreas: a primeira é uma área tipo tupla (da 1ª linha à 15ª), contendo texto e dados da Relação Lojas; a segunda é do tipo tabela (da 16ª linha à 38ª), que apresenta informações da Relação Funcionários; já a terceira corresponde a uma área texto (da 39ª linha à 41ª), correspondendo à identificação do formulário.

Durante a geração, o aspecto visual obtido através da tela será algo similar ao apresentado na figura seguinte (Figura 4.8), onde destacamos alguns elementos importantes.

```
=================================================================================================
¦                  ¦MACROOPERAÇÃO                ¦IDENTIFICAÇÃO DO                           ¦
¦                  ¦                             ¦FORMULÁRIO QUE   ¦POSIÇÃO DO               ¦
¦                  ¦                             ¦ESTÁ SENDO       ¦CURSOR NO          ¦TOTAL DE ¦
¦                  ¦                             ¦EDITADO          ¦FORMULÁRIO         ¦LINHAS   ¦
¦                  ¦                             ¦                 ¦                   ¦GERADAS  ¦
¦                  V                             V                 V                   V         ¦
¦-----------------------------------------------------------------------------------------------¦
¦LINHA DE INFORMAÇÃO  .EDITAR.  Nome Formulário:  STAR  Posição: 20 x 1  Ultima Linha: 20        ¦
¦                 /   ¦                                                               ¦ \        ¦
¦                 ¦   ¦ ESTRELA S/A                                              LOJA ¦ ¦        ¦
¦                 ¦   ¦                                                           !!  ¦ ¦        ¦
¦                 ¦   ¦                     RELAÇÃO DE FUNCIONÁRIOS                   ¦ ¦        ¦
¦                 ¦   ¦                                                               ¦ ¦        ¦
¦                 ¦   ¦                                                               ¦ ¦        ¦
¦                 ¦   ¦ Endereço: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!            ¦ ¦        ¦
¦                 ¦   ¦ Cidade: !!!!!!!!!!!!!!!!!!!!          Estado: !!              ¦ ÁREA 1   ¦
¦                 ¦   ¦ CEP: !!!!!!!!!  Telex: !!!!!!  Fone: !!!!!!!!!!!              ¦ ¦        ¦
¦                 ¦   ¦ Gerente: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!                       ¦ ¦        ¦
¦                 ¦   ¦                                                               ¦ ¦        ¦
¦      ÁREA DE  /     ¦_______________________________________________________________¦ ¦        ¦
¦      TRABALHO \     ¦ Nº FUNC. ¦        NOME DO FUNCIONÁRIO        ¦ FUNÇÃO ¦ SALÁRIO ¦COM %¦ ¦        ¦
¦                 ¦   ¦===============================================================¦ /        ¦
¦                 ¦   ¦    !!!  ¦ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ¦ !!!!!!! ¦ !!!!!!! ¦   ¦ \        ¦
¦                 ¦   ¦    !!!  ¦ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ¦ !!!!!!! ¦ !!!!!!! ¦   ¦ INÍCIO   ¦
¦                 ¦   ¦    !!!  ¦ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ¦ !!!!!!! ¦ !!!!!!! ¦   ¦ DA       ¦
¦                 ¦   ¦    !!!  ¦ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ¦ !!!!!!! ¦ !!!!!!! ¦   ¦ ÁREA 2   ¦
¦                 \   ¦    !!!  ¦ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ¦ !!!!!!! ¦ !!!!!!! ¦   ¦ /        ¦
¦LINHA GABARITO       .........1.........2.........3.........4.........5.........6.........7.........8 ¦
¦LINHA DE CONVERSAÇÃO  NOME:[COMISS] TAM:[33] TP:[ ] MASC:[              ] CHV:[ ] CONFIRMA:[ ]   ¦
¦LINHA PARA MENSAGENS ***** TAMANHO DE CAMPO ILEGAL *****                                        ¦
-------------------------------------------------------------------------------------------------
=================================================================================================
```

Figura 4.8 - Tela-Exemplo de Geração de um Esqueleto

O projeto deste formulário se refletirá nas estruturas internas da seguinte maneira:

Mapa de Memória;

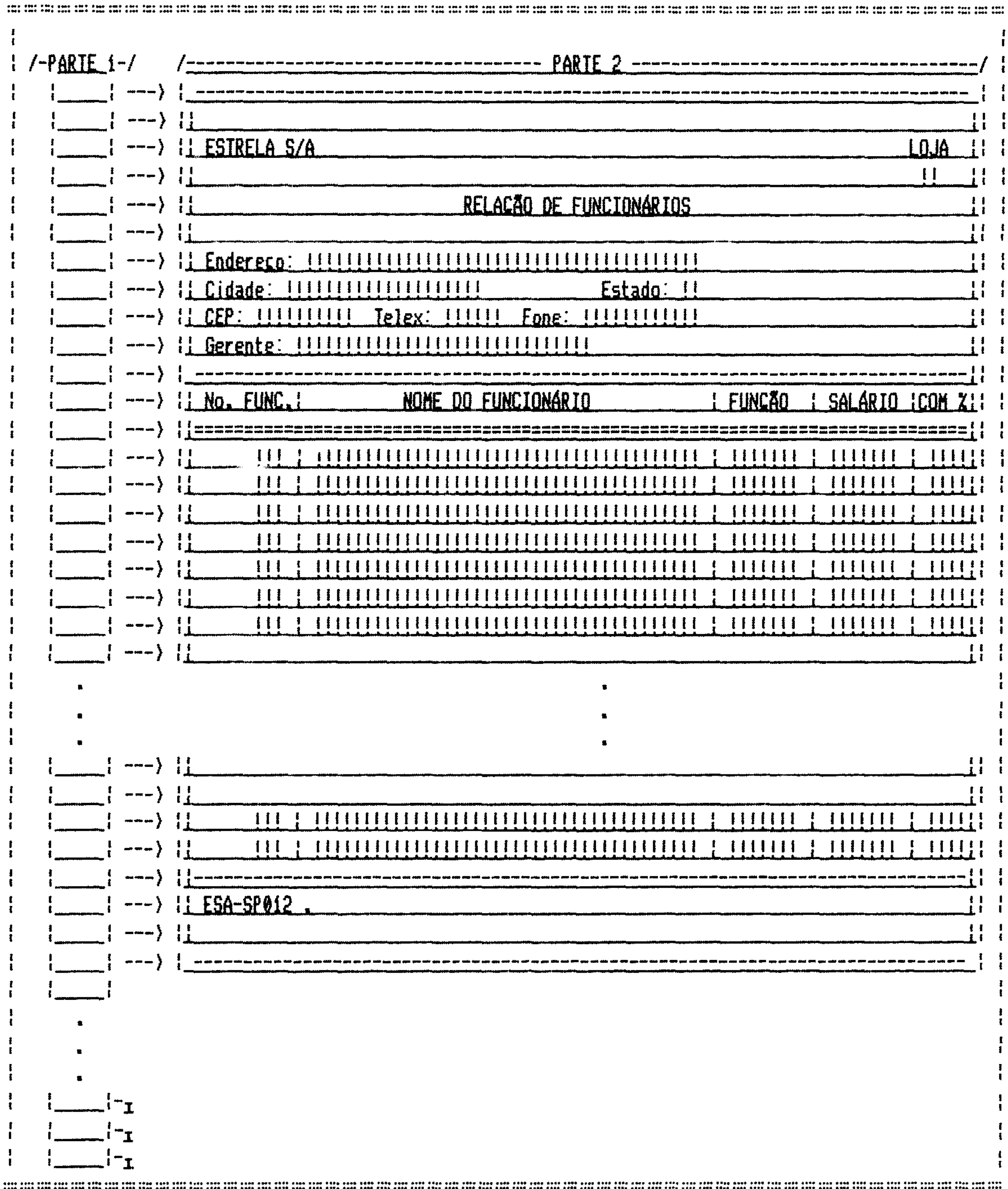


Figura 4.9 - Estrutura do Mapa de Memória

Como vemos, as estruturas de armazenagem da Figura 4.9 contêm todo o "layout" do formulário projetado.

As estruturas que controlam o manuseio das áreas e campos de dados contêm os dados da figura a seguir.

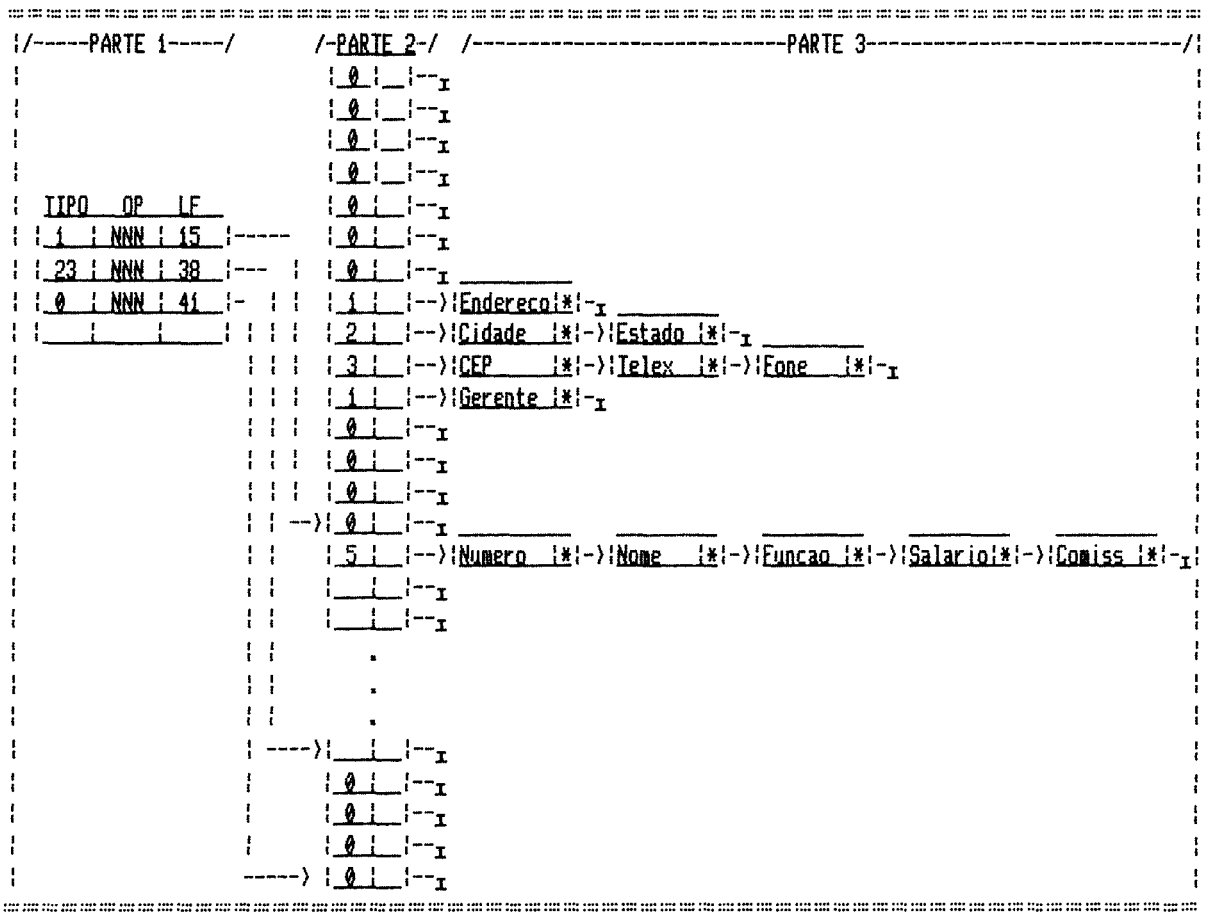


Figura 4.10 - Estruturas que controlam o manuseio de áreas e campos de dados.

Além das estruturas apresentadas, cabe lembrar a existência de estruturas para tratar o casamento (de tipos) da solicitação.

Como vemos, com essas estruturas é possível controlar todo o "layout" de um formulário, tanto a disposição gráfica quanto as especificações dos campos e características do formulário.

Para o manuseio, considerando ainda o mesmo exemplo, observaremos o mesmo formulário, preenchido com os dados das relações associadas (Figuras 4.11 e 4.12 a seguir).

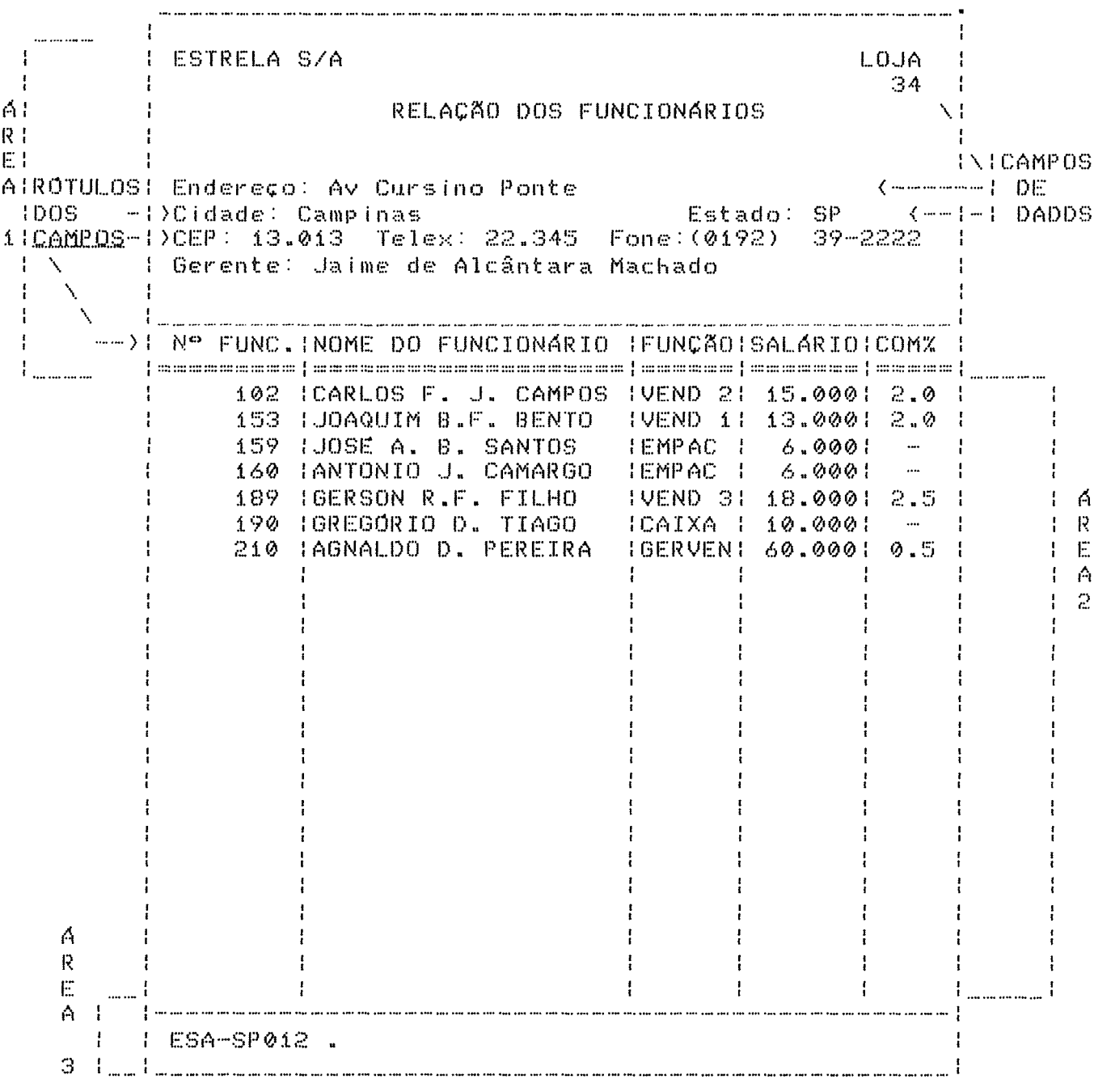


Figura 4.11 - Esquema do Formulário preenchido com os dados das Relações.

relação: LOJA

NUMERO	ENDERECO	CIDADE	ESTADO	CEP	TELEX	FONE	GERENTE
34	Av Cur...	Camp...	SP	13...	223...	01...	Jai...

relação: FUNCIONÁRIO

NÚMERO	NOME	FUNÇÃO	SALÁRIO	COMISSÃO	NDEPTO
102	CAR...	ven...	15000	2.0	4
153	JOA...	ven...	13000	2.0	4
159	JOS...	eme...	6000	0	4
160	ANT...	eme...	6000	0	4
182	GER...	ven...	18000	2.5	4
190	GRE...	cai...	10000	0	4
210	AGN...	ger...	60000	0.5	4

Figura 4.12 - Esquema das Relações

4.5 - Considerações sobre a Implementação do Protótipo.

Durante a fase de implementação do protótipo surgiram vários pontos com uma maior necessidade de análise de opções e decisões.

O primeiro aspecto a decidir se referiu à interação homem e sistema, isto é, se seria um sistema "on-line" onde cada ação do usuário teria uma imediata consequência visível através da tela do terminal, ou então se seria um sistema com definição de formatos padrões de formulários (tipo carta, tipo ofício,etc.), onde o usuário responderia a "questionários" para a especificação do formulário e só numa fase posterior poderia observar o aspecto visual do formulário.

Linguagens de definição de formulários os descrevem em termos de código de alto nível e toda validação ocorre durante a definição do mesmo. Várias linguagens já foram propostas, mas não obtiveram muita eficiência pois possuem muitas limitações. A principal limitação diz respeito ao tipo de usuário, pois esta solução exige alguma especialização em computação. Outra consequência é a necessidade de criação de "tipos de formulários", que apesar de considerarem alguns aspectos semânticos, não proporcionavam a flexibilidade desejada. Por outro lado, a criação de formulários interativamente na tela de um terminal, facilita a criação e manutenção de formulários [HUL86]. Sistemas que oferecem ao usuário uma tela formatada que permite a ele manipular porções do documento e observar o efeito imediatamente são usualmente chamados de: "What you see is what you get" ou seja "WYSIWYG" [CHA87].

Optou-se então, pela linha de implementação "on-line", pois só assim conseguimos simular realmente o processo convencional de

projetos de formulários e também acompanhar essa tendência demonstrada nos sistemas atuais.

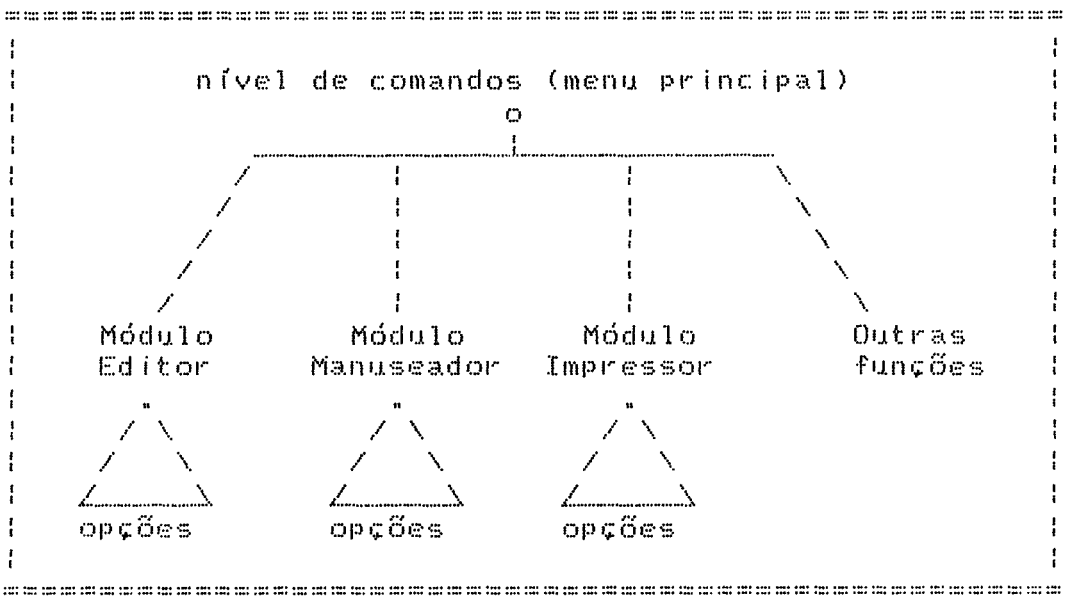
Outro aspecto analisado foi o modo de trabalho do editor, isto é, se seria orientado por linha ou se deveria considerar duas direções, vertical e horizontal. Optou-se pelo manuseio orientado por linha, principalmente pela sua simplicidade e facilidade de implementação.

Após definido o modo de trabalho, surgiu a necessidade da definição das estruturas de dados que fossem adequadas a uma situação tão dinâmica quanto a que encontramos num editor e que também permitíssem facilidades durante a fase do manuseio.

O projeto do buffer da tela, por exemplo, pode ser uma tarefa complexa se visamos otimizar o uso da memória e/ou o tempo de atualização (inserção/eliminação) [DUF83]. Otimizar a manipulação do buffer da tela implica em minimizar a quantidade de dados que deve ser fisicamente movida. A melhor forma de fazer isto é armazenar as linhas em posições fixas do buffer e colocar seus endereços em outro conjunto de variáveis. Quando a ordem das linhas é alterada, as variáveis contendo os endereços são meramente atualizadas para refletir a nova ordem. Este procedimento é mais eficiente do que mover os dados.

Optou-se então pelo uso da linguagem Pascal e por um maior uso das variáveis dinâmicas que permitem uma grande flexibilidade no uso do espaço disponível e também no manuseio das informações, principalmente pelas facilidades de manuseio de endereços encontradas na linguagem que, além de tudo, é uma linguagem simples, bastante divulgada e, por outro lado, bem abrangente.

Observando-se o fator humano devemos elaborar uma interface fácil de aprender e utilizar. De forma a fazer a navegação dentro da interface simples e prática, esta foi organizada como uma árvore semelhante ao esquema abaixo:



O usuário inicia no nível de comandos e escolhe uma das opções disponíveis. O usuário passa a interagir com o nível abaixo, dentro da subárvore apropriada. Ao terminar, o usuário volta ao nível anterior. Por facilidade de uso e implementação a árvore não tem mais que três níveis. Também não é permitido movimento horizontal, tal como passar do Módulo Editor para o Módulo Manuseador. Para facilitar a navegação deve-se assegurar o término de uma tarefa para iniciar outra. Para se obter maior consistência a interação é feita através de "menus", quando possível. Já as opções de cada Módulo esperam entradas do usuário, pois algumas tarefas são impossíveis de executar através de menus.

Um aspecto importante também foi a escolha do elenco de comandos, tanto para o Editor como para o Manuseador. Tentou-se de-

finir um conjunto que não fosse extenso ou mesmo complexo e que possibilitasse uma boa cobertura das necessidades de um sistema desse tipo.

No Módulo Editor houve um maior cuidado na definição do manuseio das diversas áreas, permitindo que o usuário crie, elimine ou atualize qualquer área de uma forma simples e similar ao uso de qualquer dos outros comandos, sem necessidade de "menus" ou outros mecanismos mais complexos.

As funções oferecidas no Módulo Gerador/Editor se encontram entre as comumente exigidas em editores [DUF83], como por exemplo: comandos para inserção/eliminação de caracteres, quatro movimentos de cursor, "scroll" de página, indicadores da coluna e linha da posição corrente do cursor, capacidade de abandonar a edição (com verificação), capacidade de impressão durante a edição, "refresh" da tela, uso de "layouts" pré-definidos, etc.

Quanto ao conjunto de consultas implementado pelo Manuseador, ele foi bastante restrito, limitando-se a projeção ou seleção de tuplas. Não foram consideradas situações em que campos de formulários possam ser derivados de atributos de relações ou obtidos através de processamento de rotinas auxiliares. Além disso, não se chegou a implementar acionamento de gatilhos como parte do processamento de um formulário.

No que diz respeito ao funcionamento do protótipo em ambientes distribuídos, temos que considerar dois aspectos: o primeiro se refere aos dados cujo acesso concorrente dependerá do SGBD utilizado, já o segundo, corresponde aos "layouts" dos formulários que logo após serem transferidos para as estruturas internas

do SEF, serão imediatamente liberados para outros usuários.

Também houve preocupação quanto a se obter um sistema modular que permitisse fácil manutenção e expansão, que são responsáveis pelos maiores custos no ciclo de vida de um programa. Por isso os módulos são pequenos e seguem os dogmas da programação estruturada (sequência, decisão e iteração).

Para efeito do protótipo implementado por esta tese, o usuário poderá definir formulários com as dimensões de no máximo 132 linhas por 80 colunas, o que atende às dimensões comumente requeridas correspondentes a 2 páginas de 66 linhas por 80 colunas, embora o tamanho da página possa ser definido livremente. Cada formulário pode ter no máximo quatro áreas.

5 - Conclusões e extensões

Esta tese apresentou um protótipo de um sistema de interface para Bancos de Dados relacionais baseado nas características de formulários tratados como visões formatadas. O protótipo permite definição e armazenamento de "layouts" independente dos dados que serão vistos através deles. O sistema é composto de 2 Módulos principais: Módulo Gerador/Editor (de esqueletos) e Módulo Manuseador (que materializa consultas ao Banco de Dados através dos "layouts" previamente definidos). Também foi necessário um Módulo Impressor que possibilita a apresentação dos formulários seja através da tela de um terminal ou via uma impressora.

Devido à sua modularidade, o sistema permite que "layouts" já definidos possam ser incorporados a outros. Além disto, um "layout" pode ser modificado a qualquer momento sem com isto exi-

gir interação com as relações que podem potencialmente ser vistas através dele.

A arquitetura proposta também propicia que novos módulos possam ser integrados ao sistema, como por exemplo um que permitisse a utilização dos "layouts" já cadastrados em programas de aplicação, de modo similar ao sistema UNIFORMS [HUL86].

Em qualquer momento o usuário pode solicitar informações gerais sobre o formulário ou seus componentes (áreas, campos, etc).

Constatamos que atualmente um conjunto de sistemas privilegia o aspecto de integração de utilitários necessários ao manuseio de aplicações de escritório. O conceito de formulários é largamente utilizado para a especificação dessas aplicações (modelagem, tratamento e interface).

Como, de uma forma geral, um formulário informatizado corresponde à imagem eletrônica de um formulário de papel, representado através do que chamamos de esqueleto, na tela do terminal, devemos prover uma aplicação baseada em formulários de soluções que se aproximem o mais possível do uso comum de um documento como por exemplo:

- projeto independente do manuseio;
- facilidade de preenchimento;
- instruções de uso simplificadas.

Algumas áreas onde deveriam ser envidados maiores esforços, para completar a ferramenta especificada são: procedimentos de manuseio em que se leve em conta a semântica do formulário (por exemplo, com a permissão de criação de tipos de formulários através de tipos abstratos de dados e a consequente possibilidade

de composição, identificação de componentes e reconhecimento de tipos), obtenção de um maior número de procedimentos automáticos (por exemplo, com a definição de "triggers"). Além dessas sugestões, outras extensões são também desejáveis, como por exemplo aumentar o universo de consultas permitidas, permitir acesso a consultas que envolvam mais de uma relação, permitir especificar o mapeamento da aplicação quando há mais de uma solução (evitar ambiguidade), criar novos tipos de campos no formulário, não necessáriamente associados a um Banco de Dados, que serão usados para entrada ou saída associada ao processamento do formulário (possibilitando inclusive encadeamento de formulários), bem como considerar relacionamentos entre campos de um formulário ou mesmo entre campos de mais de um formulário.

Devido à modularidade da arquitetura proposta, visando uma maior independência do Manuseador com relação ao Banco de Dados hospedeiro, é possível sem muito esforço de codificação expandir o protótipo com vários tradutores de solicitações para SGBDs específicos. Esses tradutores deveriam incorporar a otimização das solicitações. Com estas características estaríamos cada vez mais próximos de atingir a interface de Bancos de Dados adequada, principalmente no que diz respeito à portabilidade.

Bibliografia

[BAR85] - Barbic, F. e Rabitti, F.; The Type Concept in Office Document Retrieval. *Proceedings of the VLDB*, (1985), pp 34-48.

[CAS82] - Casey, D.; Developments in The Electronic Office. *Data Processing*, Vol 24, nº 4, (Abril/Maio 1982). pp 8-20.

[CHA87] - Chamberlin, D. D.; Document Convergence in an Interactive Formatting System. *IBM J. Res. Develop.*, Vol 31, nº 1, (Janeiro 1987), pp 58-72.

[CHO85] - Chorafas. D. N.; Fourth and Fifth Generation Programming Languages. *McGraw Hill - Book Company*, Vol 2, (1985).

[COL83] - Collet, C.; Caracterization des Formularies pour Une Base de Données generalisées. *R. R. Tigre Nº 9*, (Novembro 1983).

[COM81] - Computer System Research Group, University of Toronto; Technical Report CSRG - 127, Form Procedures. *Edited by Tsichritzis, D.* (Março 1981).

[DAT84] - Date, C. J.; A Guide to DB2. *Addison-Wesley Publishing Company* ; (1984).

[DUF83] - Duff, C. B.; Graphics Editor for IBM PC. *Byte Publications Inc*, (Novembro 1983), pp 211- 230.

[ELL80] - Ellis, C. A. e Nutt, G. J.; Office Information Systems and Computer Science. *Computing Surveys*,. Vol 12, nº 1, (Março 1980).

[HAM83] - Hamper, R.; Integrating WP and DP . *Data Processing*, Vol 25, nº 1, (Janeiro/Fevereiro 1983), pp 17-18.

[HER83] - Herrmann, K. H.; Caught between two Stools. *Data Processing*, Vol 25, nº 1, (Janeiro/Fevereiro 1983), pp 11- 18.

[HUL86] - Hull, M. E. C., Wlison, T. P.; Uniforms: an Automatic Forms Facility. *Data Processing*, Vol 28, nº 5, (Junho 1986), pp 258- 264.

[JAC87] - Jacky, J. P., e Kalet, I. J.; An Object - Oriented Programming Discipline for Standard Pascal. *Communications of the ACM*,. Vol 30, nº 9, (Setembro 1987), pp 772- 780.

[KAN85] - Kantrowitz, E., Maryanski, F., Shasha, D.; Distributed Office By Example (D-OBE). *RC 11632 (#50614) Computer Science*, (junho 1985).

[KEL86] - Keller, A. M.; The Role of Semantics in Translating View Updates. *Data Enginering Computer* , (Janeiro 1986), pp 63 - 73.

[KIN87] - King, R., e Novak, M.; A User - Adaptable Form Management System. *Proceedings of the 13th VLDB Conference*, (1987), pp 331-338.

[LAM84] - Lamersdorf, W., e Muller, G., e Schmidt, J. W. ; Language Support for Office Modelling. *Proceedings of VLDB*, (1984), pp 280- 288.

[MAN84] - Mannino, M. V. e Chovbineh, J.; Research on Form Driven Database Design and Global View Design. *IEEE Database Engineering*, Vol 3, (1984), pp 239- 244.

[MAR84] - Martin, J.; Fourth Generation Languages. *McGraw Hill-Book Company*, (1984).

[MEY87] - Meyer, B.; Reusability: The Case for Object-Oriented Design. *IEEE Software*, (Março 1987), pp 50- 64.

[MOW86] - Mowshowitz, A.; Social Dimensions of Office Automation. (1986), pp 337- 399.

[NIE85] - Nierstrasz, O. M., e Tsichritzis, D. C.; An Object-Oriented Environment for OIS Applications. *Proceedings of VLDB*, (1985), pp 335- 345.

[ROW82] - Rowe, L. A. e Shoens, K. A.; A Form Application Development System. *ACM-SIGMOD Conference Proceedings, ACM-SIGMOD International Conference on Management of Data*; (Junho 1982), pp 295- 311.

[ROW85] - Rowe, L. A.; Fill-in-the-Form "Programming".*Proceedings of VLDB*, (1985), pp 394- 404.

[RUL87] - Ruland, D. e Kratzer, K. P.; DQL - a Query Language for Hierarchically Structured Documents. *Research Report, Computer Science*, (Abril 1987), pp 1- 44.

[SET86] - Setzer, V. W.; Projeto Lógico e Projeto Físico de Banco de Dados. *V Escola de Computação - Belo Horizonte - UFMG* . (1986).

[SHU82] - Shu, N. C.,e Lum, V. Y., e Tung, F. C., e Chang, C. L.; Specification of Forms Processing and Business Procedures of Office Automation. *IEEE Transactions on Software Engineering*, Vol SE-8, nº 5, (Setembro 1982), pp 499- 512.

[STO83] - Stonebraker, M., e Stettner, H., e Lynn, N., e Kalash, J., e Guttman, A.; Document Processing in a Relational Database System; *ACM Transaction on Office Information Systems*, Vol 1, nº 2, (Abril 1983), pp 357- 375.

[TOM85] - Toma, T.; A Automação de Escritório Avança no País. *Dados & Idéias*, (Janeiro 1985), pp 6- 13.

[TSI83] - Tsichritzis, D.,e Christodoulakis, S., e Economopoulos, P.,e Faloutsos, C., e Lee, A., e Lee, D. e Vandenbroek, J., e Wod, C.; A Multimedia Office Filing System. *Proceedings of VLDB*, (1983), pp 2-7.

[UHL81] - Uhlig, R. P., e Farber, D. J.,e Bair. J. H.; The Office of the Future (Monograph Series of the International Council for Computer Communications), *North Holland Publishing Company*, Vol 1, (1981).

[WAR86] - Wartik, S. P., e Penedo, Maria H.; Fillin: A Reusable Tool for Form-Oriented Software; *IEEE Software*, (Março 1986), pp 61-68.

[WAT87] - Whang, K-Y. e outros, IBM Thomas J. Watson Research Center; Office-By-Example: An Integrated Office System and Database Manager. *ACM Transactions on Office Information Systems*, Vol. 5, nº 4, (Outubro 1987), pp 393- 427.

[WIE86] - Wiederhold, G.; Views, Objects and Databases. *Computer*, Vol. 19, nº 12, (Dezembro 1986), pp 37- 44.

[ZLO82] - Zloof, M. M.; Office By Example: A Business Language That Unifies Data and Word Processing and Electronic Mail. *IBM System Journal*, Vol. 21, nº 3 , (1982).

[ZUS86] - Zussman, J. U., e Sachs, J., e Gomez, G.; SQL*FORMS Designer's Reference. *ORACLE Part number: 3304 - V2.0*, (1986).

ANEXO A

Utilização do Protótipo - execução do SEF

O Sistema proposto compreende os seguintes módulos:

. Gerador de Esqueleto

. Manuseador de Formulários

. Impressor de Formulários

Inicialmente o usuário deverá escolher entre esses três módulos principais através de um Menu (Figura A.1).

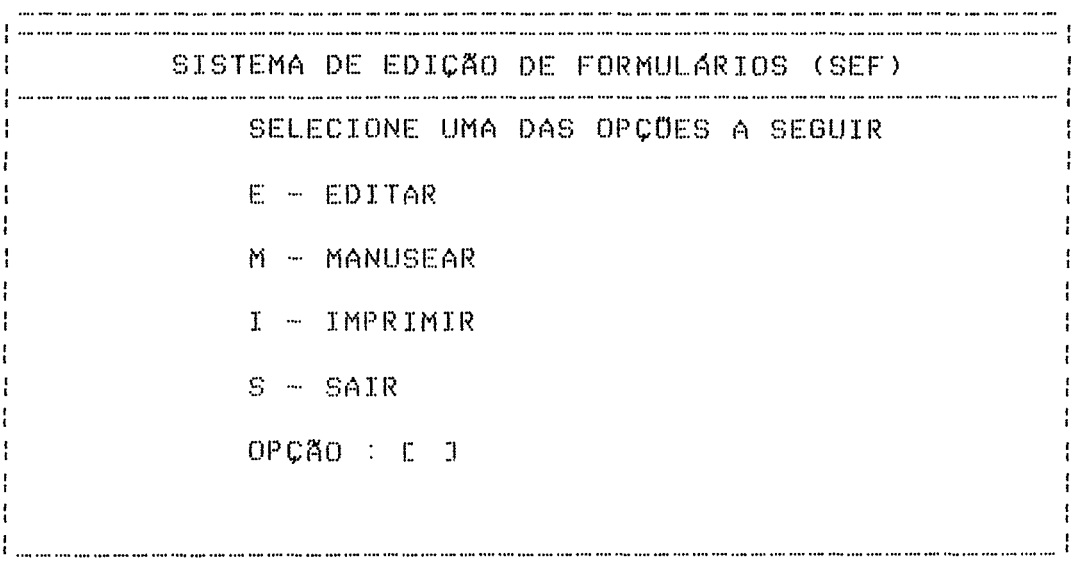


Figura A.1 - MENU PRINCIPAL

Opção E - permite ao usuário criar novos esqueletos de formulários ou alterar os já existentes (Módulo Gerador).

Opção M - permite ao usuário manusear relações de Bancos de Dados Relacionais através de um formulário compatível (Módulo Manuseador).

Opção I - permite imprimir um ou mais formulários ou então

apresentá-los na tela do terminal (Módulo Impressor).

Opção S - retorna o controle ao Sistema Operacional.

Ao optar pelo Gerador, o usuário deverá indicar o nome do esqueleto do formulário a ser especificado ou alterado. Numa geração inicial (Figura A.2), o usuário especifica as características gerais do novo esqueleto, como por exemplo o tamanho da página. Precauções devem ser tomadas quando da especificação da dimensão da página de um formulário, pois sua coerência é responsabilidade do usuário (para não acontecer por exemplo, que dados definidos para o rodapé do formulário, últimas linhas de uma página, passem para o topo da próxima página).

```
+-----------------------------------------------------------+
|        SISTEMA DE EDIÇÃO DE FORMULÁRIOS (SEF)             |
+-----------------------------------------------------------+
|                                                           |
|        ROTINA DE EDIÇÃO                                   |
|                                                           |
|        NOME DO FORMULÁRIO : [XXXXXXXX]                    |
|                                                           |
|        DIMENSÃO (LINHA x COLUNA) : [XXX]x[XX]             |
|                                                           |
|                                                           |
|        CONFIRMA INFORMAÇÕES ? (S/N) : [X]                 |
|                                                           |
+-----------------------------------------------------------+
```

Figura A.2 - GERAÇÃO INICIAL

Já na alteração de um esqueleto (Figura A.3), após o sistema confirmar a intenção de modificação, tornará disponível ao usuário a situação atual do "layout".

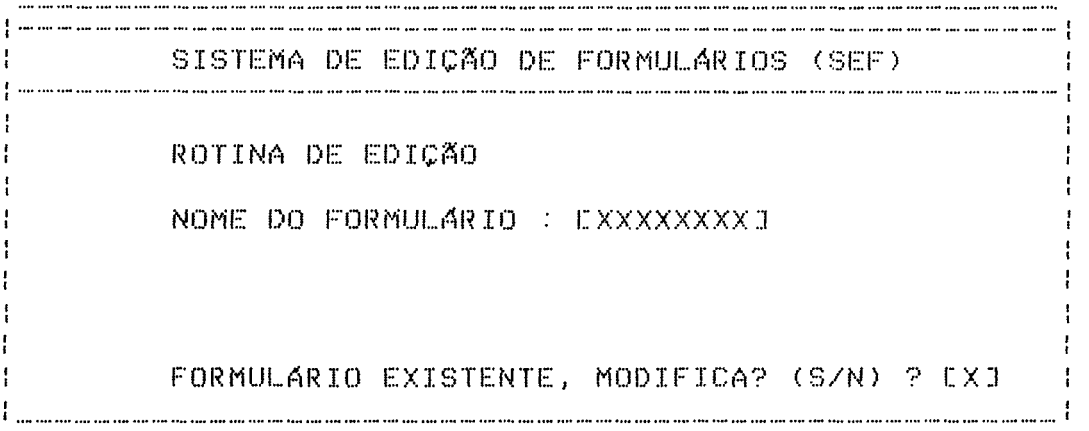


Figura A.3 -- ALTERAÇÃO

Em qualquer caso, o usuário é inicialmente posicionado na primeira posição da primeira área do esqueleto e poderá iniciar (ou continuar) o projeto deste esqueleto. Os comandos de edição têm validade dentro de cada área e a cada nova área criada (Figura A.4), o usuário é obrigado a especificar o tipo da área (0 = texto, 1 = tupla, N = tabela, onde N é um número inteiro positivo que especifica o número de linhas da tabela), e das operações que poderão ser aplicadas quando do manuseio do formulário (Figura A.5), naquela área (consulta e/ou alteração e/ou eliminação e/ou inclusão).

A primeira linha da tela é ocupada por informações do SEF contendo o nome do formulário , tipo da operação, linha e coluna que a pessoa está manuseando no formulário, e o total de linhas já criadas.

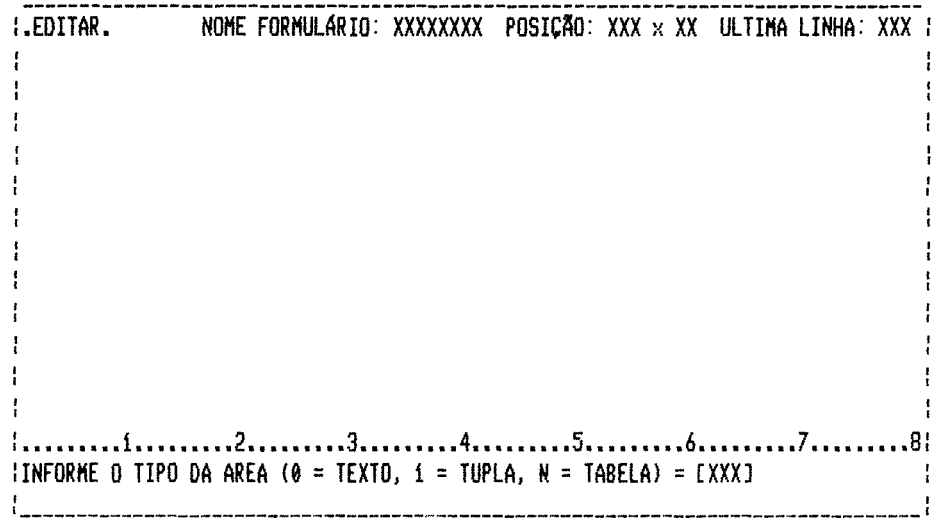


Figura A.4 - SOLICITAÇÃO DO TIPO DA ÁREA NUMA GERAÇÃO.

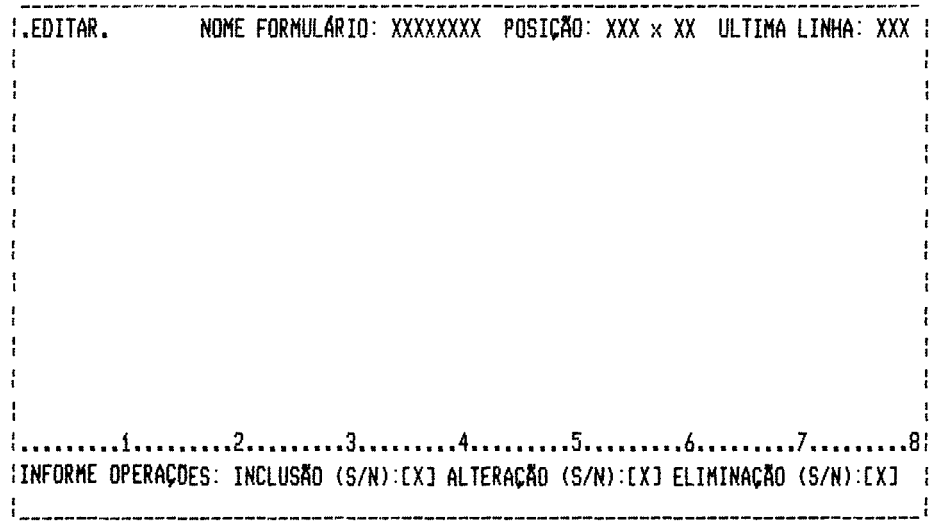


Figura A.5 - SOLICITAÇÃO DAS OPERAÇÕES PERMITIDAS NA ÁREA NUMA GERAÇÃO

A linguagem para utilização do Gerador pode ser dividida em

duas classes de comandos (vide tabela de comandos).

a - Para a apresentação dos dados na tela, referente a comandos que não causam nenhum efeito nas estruturas de dados internas, servindo apenas para dar o efeito de janela desejado.

Exemplos: ^S - move o cursor à esquerda;

^D - move o cursor à direita;

^E - move o cursor para a linha anterior; etc.

b - Para a Geração dos esqueletos dos formulários referente a comandos que acarretam alterações nas estruturas de dados internas e consequentemente alterações na tela.

Exemplos: ^T - ativa a rotina de definição de campo de dados;

^N - inclui nova linha no formulário;

^Y - elimina a linha corrente; etc.

Ao digitar caracteres alfabéticos, especiais e numéricos, em conjunto com os comandos do tipo 'b', o usuário cria ou altera o esqueleto do formulário, definindo então títulos, rótulos de campos, linhas, campos, etc. Durante a edição, na parte que diz respeito ao texto, os caracteres teclados vão sendo armazenados e apresentados na tela, mas ao teclar o comando que solicita a criação de campo, o programa muda de ambiente e o usuário fica restrito a determinadas respostas e assim que termina fica novamente liberado para a criação ou atualização do texto, até indicar o fechamento da área.

Após concluir a edição, o usuário poderá receber alguns avisos alertando sobre alguns problemas, tais como, número de linhas do formulário incompatível com o tamanho da página, etc. A seguir, o usuário poderá ou não gravar a nova situação decorrente

daquela sessão de trabalho, seja gerando um novo esqueleto ou alterando um já existente (quando então serão criados arquivos de "backup" com a situação anterior). A seguir o "menu" principal estará novamente disponível ao usuário.

Se a opção inicial foi para o manuseio (Figura A.6), o usuário deverá indicar qual esqueleto será utilizado. O formulário será então criticado para verificar sua viabilidade de uso pelo módulo Manuseador. Logo após deve-se especificar a relação associada a cada área. O sistema executa então a rotina de "casamento" entre o formulário solicitado e relações a serem manuseadas. Para cada área, associada a uma relação, verifica-se a compatibilidade entre a aridade e as características dos campos do formulário e dos atributos da relação.

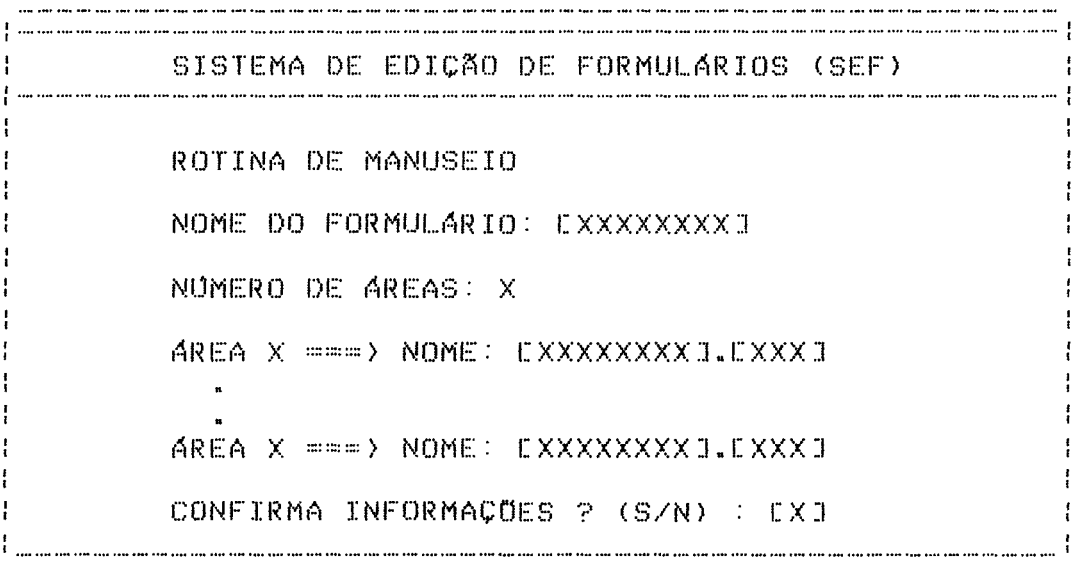


Figura A.6 - MANUSEIO

Após ser confirmada a requisição (formulário versus relações associadas), o esqueleto é colocado disponível para o usuário. Também neste caso os comandos serão válidos dentro de cada área,

cabendo ao usuário a responsabilidade pela coerência das requisições.

Diversos comandos utilizados durante a edição também serão válidos durante o manuseio, principalmente os que controlam a apresentação dos dados na tela.

Os principais comandos característicos do manuseio são os que disparam as rotinas que permitem o acesso às relações: consulta, inclusão, alteração ou eliminação (vide tabela de comandos). Qualquer uma dessas operações é orientada pelos campos definidos como chaves durante a geração.

Na consulta (Figura A.7), podemos optar pelo modo 'default' (consultas segundo a ordem das tuplas nas relações) ou com condições (especificadas pelo usuário).

```
.MANUSEAR.     NOME FORMULÁRIO:     STAR  POSIÇÃO:  16 x  1  ULTIMA LINHA:  41
 ESTRELA S/A                                                          LOJA
                                                                       34
                        RELAÇÃO DOS FUNCIONÁRIOS
 Endereço: Av Cursino Ponte
 Cidade: Campinas                      Estado: SP
 CEP:     13013  Telex: 22.345  Fone: 0192-39-2222
 Gerente: Jaime de Alcântara Machado
 No. FUNC.|        NOME DO FUNCIONÁRIO           | FUNÇÃO  | SALÁRIO |COM %
      102 | CARLOS F. J. CAMPOS                  | VEND 2  |  15.000 | 2.0
      153 | JOAQUIM B. F. BENTO                  | VEND 1  |  13.000 | 2.0
      159 | JOSÉ A. B. SANTOS                    | EMPAC   |   6.000 |  -
      160 | ANTONIO J. CAMARGO                   | EMPAC   |   6.000 |  -
      189 | GERSON R. F. FILHO                   | VEND 3  |  18.000 | 2.5
.....1.....2.....3.....4.....5.....6.....7.....8
```

Figura A.7 - CONSULTA

No modo "default" a área é preenchida com os dados da primeira tupla (se estiver numa área tupla) ou das primeiras tuplas (se estiver numa área tabela). No ambiente de uma consulta há diversos comandos válidos, exemplo: próxima tupla (para áreas tipo tupla), próximo grupo (para áreas tipo tabela).

No processamento de operações de atualização, o mapeamento para o SGBD associado é feito por intermédio do gerenciador de "buffer" com chamadas que retornam um argumento indicativo de sucesso ou falha da solicitação.

Tanto na edição quanto no manuseio estarão disponíveis rotinas que possibilitarão a impressão da tela corrente ou de todo o formulário. Em qualquer momento o usuário poderá obter informações gerais sobre o formulário agrupadas em 3 classes: do formulário propriamente dito (dimensão da página, total de linhas gravadas, total de áreas definidas, número de páginas), das áreas (tipo, operações permitidas, nomes dos campos pertencentes àquela área), e dos campos (nome, tamanho, tipo, máscara e se é chave ou não).

Se a opção for para o módulo Impressor, o usuário inicialmente deverá escolher qual o dispositivo de saída, se a tela ou a impressora. Feita esta opção, deverá indicar sucessivamente quais os nomes de arquivos que contêm os formulários que deverão ser impressos ou apresentados na tela (estes arquivos são resultado de uma operação de Gravação de formulário, comando ^K^G, possível tanto na edição quanto no manuseio).

ANEXO B

TABELA DE COMANDOS

	MOVIMENTO DO CURSOR
↑D	Move o cursor uma posição para a direita. No modo de Sobre Escrito que é o "default" a posição limite é a ultima posição da área corrente. No modo de Inserção é, a última posição da linha.
↑E	Move o cursor para a linha anterior à corrente, limitando-se ao cursor estar posicionado na primeira linha da área em uso.
↑L	Move o cursor para o fim da linha.
↑Q	Move o cursor para o inicio da linha.
↑S	Move o cursor uma posição para a esquerda. No modo de Sobre Escrito que é o "default". a posição limite é a primeira posição da área corrente. No modo de Inserção é a primeira posição da linha.
↑X	Move o cursor para a linha seguinte à corrente, limitando-se ao cursor estar posicionado na última linha da área em uso.
↑W	Desloca a tela (como uma janela), de uma linha para cima, limitando-se ao cursor estar posicionado na primeira linha da área em uso.
↑Z	Desloca a tela (como uma janela), de uma linha para baixo limitando-se ao cursor estar posicionado na última linha da área em uso.
	GERAÇÃO DO ESQUELETO
↑A	Encerra a área atual e posiciona o cursor na primeira posição da área anterior.

↑B	Insere uma linha em branco abaixo da linha corrente.
↑G	Elimina o caractere sobre o qual está o cursor.
↑H	Elimina o campo sob o qual o cursor está posicionado.
↑I	Muda para o modo de Inserção e viceversa. No modo de Inserção são válidos os seguintes comandos: ↑S, ↑D, ↑Q, ↑K, ↑G, ↑R,↑L. A Inserção é limitada ao ambiente de uma linha.
↑J	Altera tamanho da página do formulário linha x coluna (o tamanho da coluna não pode ser reduzido).
↑N	Insere uma linha em branco acima da linha corrente.
↑O	Altera tipo de operação da área corrente: (Inclusão (S/N), Alteração (S/N), Eliminação (S/N)).
↑P	Encerra a área atual e posiciona o cursor na primeira posição da área seguinte.
↑T	Cria um campo a partir da posição atual do cursor, possibilitando a especificação das características do campo: nome, tamanho, tipo, máscara e chave (S/N).
↑U	Elimina a área corrente, devendo o usuário confirmar ou desistir da eliminação.
↑V	Insere Subformulário indicado pelo usuário na posição adequada (o subformulário só deve conter uma única área).
↑Y	Elimina a linha na qual está posicionado o cursor.
	GERAÇÃO E MANUSEIO
↑K↑A	Apresenta as características principais de uma área: Tipo (texto = 0, tupla = 1 ou tabela = n>1)

	Operações (consulta, inclusão, alteração e/ou eliminação) e uma lista com os nomes de todos os campos associados à área (completa ou até o ponto que o usuário desejar).
↑K↑D	Apresenta as características principais de um campo sobre o qual está posicionado o cursor: - Nome, Tamanho, Tipo (inteiro, caracter), Máscara, e se é chave ou não.
↑K↑F	Apresenta as características principais de um formulário: - dimensão da página (número de linhas x número de colunas), número de linhas total (considerando todas as páginas até a última linha gravada), número de áreas definidas e o número de páginas do formulário.
↑K↑G	Grava o conteúdo corrente do formulário num arquivo indicado pelo usuário, que poderá ser um arquivo novo ou que contenha formulários anteriores (arquivo cumulativo). Esses arquivos poderão ser impressos com a opção IMPRIMIR (no menu principal).
↑K↑I	Descarrega o conteúdo da tela na impressora.
↑R	Executa o 'refresh' da tela.
↑F	Fim de serviço, no modo de Geração ou de Manuseio
	MANUSEIO
↑A	Encerra a área atual e posiciona o cursor na primeira posição da área anterior.
↑B	Posiciona o cursor no campo anterior.
P	Solicita próxima tupla (numa área tupla) ou próximo grupo (numa área tabela)
A	Solicita tupla anterior (numa área tupla) ou grupo anterior (numa área tabela)

↑H	Possibilita a Eliminação de tuplas em relações através de um determinado formulário.
↑I	Possibilita a Inclusão de tuplas em relações através de um determinado formulário.
↑J	Possibilita a Alteração de tuplas em relações através de um determinado formulário.
↑N	Posiciona o cursor no campo seguinte.
↑O	Possibilita o processamento de consultas em dois modos: "default" (sequencial) ou com condições (especificadas pelo usuário).
↑P	Encerra a área atual e posiciona o cursor na primeira posição da área seguinte.