PROPOSTA DE UM MODELO DE DADOS PARA SISTEMAS DE INFORMAÇÃO PARA ESCRITÓRIOS

Luz Corina del Pino Rodriguez

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

Aprovada por:

Prof. Jano Moreira de Souza
(Presidente)

Prof. Ana Regina Cavalcanti da Rocha

Prof. Amauri Marques da Cunha

RIO DE JANEIRO, RJ - BRASIL

Janeiro de 1991

DEL PINO RODRIGUEZ, LUZ CORINA

Proposta de um Modelo de Dados para Sistemas de Informação para Escritórios (Rio de Janeiro) 1991.

IX, 184 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1991)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Modelo de Dados

I. COPPE/UFRJ II. Título (série).

Dedico este trabalho aos meus pais, Moises e Gloria, esperando compensar, de alguma forma, as limitações sofridas a favor de sua realização.

AGRADECIMENTOS

Ao Prof. Jano Moreira, orientador deste trabalho, quero agradecer, pela sua participação eficiente, sua amizade e compreensão pelas mudanças e atrasos ocorridos no andamento do trabalho.

Ao Prof. Luiz Carlos Monte, que atendeu às solicitações efetuadas para a realização deste trabalho.

Ao colega Marco Vaz, pelo apoio na condução de diversos assuntos relacionados com este trabalho.

Ao pessoal do Laboratório de Engenharia de Sistemas e Computação, pelo apoio oferecido.

A minha amiga Morgana Diniz, pela boa convivência, amizade e estímulo constante em todos os momentos difíceis.

Aos meus pais agradeço a mais profunda contribuição, pois fizeram-me capaz de vencer os obstáculos.

Finalmente, agradeço a Deus acima de tudo.

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

PROPOSTA DE UM MODELO DE DADOS PARA SISTEMAS DE INFORMAÇÃO PARA ESCRITÓRIOS

Luz Corina del Pino Rodriguez

Janeiro, 1991

Orientador: Prof. Jano Moreira de Souza
Programa: Engenharia de Sistemas e Computação

Um escritório é um ambiente complexo e coloca demandas especiais sobre o modelo de dados usado. O ambiente de informação de escritório apresenta novos desafios de gerenciamento da informação.

Este trabalho apresenta as principais características dos Sistemas de Automação de Escritório, Metodologias e Modelos Conceituais para Escritórios, Modelos de Dados existentes para Sistemas de Informação para Escritórios (SIEs). Propõe-se um Modelo de Dados Orientado a Objetos, o qual fornece um conjunto de construções de modelagem que endereçam as questões básicas da gerência de informação para SIEs.

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

A PROPOSAL OF A DATA MODEL FOR OFFICE INFORMATION SYSTEMS

Luz Corina del Pino Rodriguez

Janeiro, 1991

Thesis Supervisor: Jano Moreira de Souza

Department: Computing and Systems Engineering

An office is complex environment and it sets especial requiriments upon the data model used. The environment of office information presents new challenges for information management.

This work shows the main features of Office Automation Systems, Methodologies and Office Conceptual Models and actual Data Models for Office Information Systems (OIS). It is proposed an Object Oriented Data Model, which provides a set of modelling blocks addressing the basic issues of information management in OIS.

Í N D I C E

CAPITULO I

INTRODUÇÃO

I.1. MOTIVAÇÃO

A demanda crescente de Automação de Escritórios para suportar ou substituir o trabalho feito nos escritórios, está sendo o motivo de muita pesquisa na ciência da computação. O enfoque de maior atenção reside nos sistemas e facilidades para apoiar o trabalho de escritório nos seus aspectos mais básicos.

Um Sistema de Informação para Escritório (SIE) automatizado tenta executar as funções do escritório comum por meio de um sistema de computador.

Na pesquisa de Sistemas de Informação para Escritórios, um SIE é definido como um sistema operacional distribuído com uma interface de usuário altamente refinada e uma facilidade de banco de dados.

Estando a tecnologia desses sistemas ainda em uma etapa de formação, não existindo uma teoria completa sobre eles e sendo esses sistemas bastante dinâmicos; podemos dizer que estes argumentos favorecem a análise teórica para SIEs e o uso de modelos de dados específicos para estes sistemas.

Atualmente existem poucos modelos de dados específicos para Automação de Escritórios (AE), embora projetar um modelo de dados especificamente para um sistema de escritório tenha as vantagens adicionais de estender e integrar uma ampla variedade de ferramentas fornecidas para esses sistemas; assim como a capacidade de apresentar uma imagem coerente dos relacionamentos que existem no

escritório aos usuários do sistema, pela formulação de uma descrição do escritório em termos da modelagem de dados. Além disso, um escritório é um ambiente complexo e coloca demandas especiais sobre o modelo de dados usado.

O ambiente de informação de escritório apresenta novos desafios de gerenciamento da informação. Em particular, os tipos de informação e os padrões de acesso à informação são bastante diferentes daqueles para os quais a tecnologia e sistemas de gerência de banco de dados convencional estão destinados.

Uma outra motivação para esta pesquisa é o projeto Engenharia de Dados existente na COPPE, que visa estudar as caracteristicas de interface entre classes específicas de usuários e/ou aplicacões em um Sistema de Gerência de Banco de Dados (SGBD). Estas interfaces específicas usuário/SGBD são: CAD/CAM, automação de escritório, computação gráfica e inteligência artificial; e dentro desses componentes/interfaces têm-se identificado os modelos de dados específicos para eles.

O objetivo da tese é fazer um estudo dos modelos de dados para SIEs existentes e propor um modelo adequado às necessidades do projeto Engenharia de Dados.

I.2. ORGANIZAÇÃO DA TESE

A presente tese foi assim organizada:

O Capítulo I apresenta a motivação para a realização do trabalho e como está organizado.

O Capítulo II explora os Sistemas de Automação de Escritórios (AE), apresentando seus conceitos básicos tais

como: Funções e Serviços, Recursos Tecnológicos, Sistemas Integrados, Ambiente de Escritórios, Estação de Trabalho, Procedimentos de Escritórios e Sistemas de Informação para Escritórios (SIEs) e, ainda, analisando os seus impactos.

No Capítulo III, são apresentadas as Metodologias e Modelos propostos para o Projeto Conceitual dos Sistemas de Informação para Escritórios (SIEs) e duas classificações de Modelos, uma segundo a visão organizacional do escritório e, a outra, segundo a visão técnica do escritório.

No Capítulo IV, são apresentados os Modelos de Dados existentes para SIEs tais como: os modelos Relacionais "QBE/OBE", "SEDL", "NT", "NFQL"; o modelo de Rede "Entidade-Documento"; o modelo Entidade-Relacionamento "ODB"; o modelo de dados Recursivo; os modelos Semânticos "ADABTPL", "OMS", "Minstrel-ODM"; e os modelos de dados orientados a objetos (O-O) "Son of Oz", "OPAL", "IRIS" e o Modelo de Gibbs.

O Capítulo V apresenta os requisitos para um Modelo de Dados para SIEs, as características dos modelos de dados Orientados a Objetos (O-O) e propõe um Modelo de Dados para SIEs, o qual é devidamente analisado.

Finalmente, no Capítulo VI são apresentadas as conclusões do trabalho.

CAPITULO II

CONCEITOS DE SISTEMAS DE AUTOMACAO DE ESCRITÓRIOS

II.1. INTRODUÇÃO

Uma pesquisa realizada por Booz, Allen & Hamilton (Wolfsdorf,1988) nos Estados Unidos revelou que "os gerentes gastam de 66 a 80% de seu tempo em comunicações verbais, que são menos eficazes que as escritas e consomem mais tempo; os empregados gastam 40% de seu tempo em processamento de correspondência e chamadas telefônicas; as secretárias gastam em média apenas 20% do tempo de trabalho em datilografia, sendo o restante consumido em telefonemas, arquivos e pesquisa de informações.

Os gerentes gastam, por absoluta falta de recursos, somente 13% de seu tempo em atividades criativas tais como planejamento, pesquisa de novas fontes de mercado, etc."

A partir da década de 70 tenta-se resolver estes problemas através da Automação de Escritórios (AE), tornando o trabalho dos funcionários mais produtivo.

A produtividade no escritório é resultado do valor adicional dado ao "enriquecimento" do trabalho, que implica, não se fazer aquilo que não precisa ser feito.

A crescente demanda de informação fez com que surgissem nas empresas, novas expectativas gerenciais no tocante à automação tais como:

- incremento do acesso à informação;
- disponibilidade de meios simples e rápidos para distribuição da informação;
- incremento e facilidade de comunicação para fora da empresa;

- redução do impacto geográfico, isto é, possibilidade da empresa trabalhar como se suas instalações estivessem fisicamente num mesmo local;
- otimização do tempo dedicado ao gerenciamento;
- maior produtividade;
- ajuda na tomada de decisões. (Wolfsdorf, 1988).

Além destas razões, outros fatores que favorecem ao uso de AE são: o barateamento dos terminais inteligentes e o desenvolvimento de software transparente ao usuário; o ambiente competitivo no qual as empresas operam e à própria disponibilidade da informação que ainda vai procurar melhorar a qualidade de nossas vidas. (Chorafas,1982), (Carvalho,1986).

Em um escritório automatizado as atividades serão executadas mais eficientemente e o conceito de trabalho de escritório será mudado. (Sassone,1986).

Coloca-se no entanto a questão do poder dentro do escritório e, segundo Olson (Olson,1982) o maior potencial da AE está na capacidade dos gerentes de obter um controle crescente sobre as suas operações.

II.2. DEFINIÇÃO DE AUTOMAÇÃO DE ESCRITÓRIO

Segundo Hammer, Automação de Escritórios (AE) é um fenômeno, uma corrente, um conceito e um dos temas mais comentados na atualidade. Há mais de uma visão ou conceito dele: 1) É visto como uma extensão do processamento de dados para aplicações tradicionais;

2) Vai na direção de um escritório "sem papéis" onde

as tarefas básicas de manipulação da informação são automatizadas;

3) É visto como um apoio à decisão gerencial mediante o uso de ferramentas.

Hammer define AE como "o uso da tecnologia para aumentar a produtividade no escritório pela melhor realização das funções do escritório". (Hammer, 1980).

Segundo Kling, define-se AE como "o estudo de computarização para suportar ou substituir o trabalho feito nos escritórios". Entende-se computarização como a combinação do acoplamento de equipamento de comunicações/computador e as políticas/práticas associadas mediante as quais organiza-se o acesso ao equipamento, informação e recursos de suporte, como por exemplo treinamento. (kling,1985).

Segundo Wolfsdorf, define-se como "sendo a aplicação da tecnologia de Sistemas de Informação para aumentar a produtividade do escritório e propiciar a oportunidade de incrementar a tomada de decisões dos diretores, gerentes, supervisores, funcionários administrativos e secretárias que trabalham na empresa. AE visa a racionalização do uso de todas as informações e meios disponíveis no escritório, antes que a extinção de papéis do escritório ou a robotização dos funcionários". (Wolfsdorf,1988).

Cavellucci define-a como "um sistema integrado de comunicação e informação por computador que suporta procedimentos de gerência num ambiente de escritórios. Os sistemas de AE representam métodos estruturados de manipulação de processamento de textos de empresas e

comunicações através de uma rede integrada". (Cavellucci,1986).

Para Sassone, o valor da tecnologia de AE reside em seu poder para complementar ou incrementar o desempenho de gerentes e profissionais, antes que na sua capacidade de substituir mão de obra em tarefas simples. (Sassone,1986).

II.3. FUNÇÕES E SERVIÇOS DE UM SISTEMA DE AUTOMAÇÃO DE ESCRITÓRIO

Segundo a classificação de Cavellucci, as funções que integram um sistema de AE são:

- . Gerência de Atividades Básicas e Procedimentos;
- . Preparação de Arquivos e Recuperação de Documentos;
- . Apoio à Decisão;
- . Intercâmbio e Distribuição de Informações.

Tais funções incorporam os seguintes serviços :

a) Gerência de Atividades Básicas e Procedimentos:

- . Agenda Eletrônica;
- . Lembrete Eletrônico;
- . Gerência de recursos (salas de reuniões, equipamento, etc.);
- . Caixa de Entrada;
- . Linguagens de especificação de procedimentos de escritório;
- . Sistemas de Informação automatizados.

b) Preparação de Arquivos e Recuperação de Documentos:

- . Processamento de Texto;
- . Arquivamento / Recuperação de Documentos;

. Linguagens convencionais;

. Pesquisa de Documentos.

c) Apoio à Decisão:

. Computação Pessoal;

. Análise de Dados;

. Elaboração de Gráficos;

. Consulta a Banco de Dados.

d) Intercâmbio e Distribuição de Informações:

. Teleconferência;

. Videoconferência;

. Correio Eletrônico;

. Correio de Voz.

Muitos dos serviços de AE só poderão ser implementados na prática quando estiverem integrados com outros.

II.4. RECURSOS TECNOLÓGICOS

Como Recursos Tecnológicos para a Automação de Escritórios têm sido desenvolvidos inúmeras ferramentas de hardware e software capazes de suportar as tarefas rotineiras de um escritório; esta seção apresenta uma classificação destas.

II.4.1. FERRAMENTAS DE HARDWARE

Estas ferramentas podem ser classificadas segundo as tecnologias básicas de: Computação Digital, Telecomunicações e Equipamentos de Escritório.

- Tecnologia de Computação Digital: inclui computadores de grande porte, minis e micros; Entrada e Saida: teclado, vídeos, impressoras, traçadores gráficos, sintetizadores de voz, etc.
- Tecnologia de Comunicações: redes de comunicação, redes locais (as redes locais podem ser interligadas via redes de comunicação, gerenciadas pelo computador central, possibilitando a comunicação interdepartamental e a integração dos serviços de AE); Meios de Transmissão (telex, rede telefônica, satélites, etc.); tecnologias de transmissão (voz, dados, vídeo, fac-simile).
- Tecnologias de Equipamento de Escritório: teclados, microfilmagem (microfilmes e microfichas, máquinas copiadoras, etc.). (Carvalho,1986).

II.4.2. FERRAMENTAS DE SOFTWARE

Estas ferramentas podem ser classificadas em: Primitivas, Nativas, e de Prestígio.

- Primitivas: aquelas que devem ser necessárias tais como: a capacidade de preparação e recuperação de informação, interface amigável ao usuário, calendário de pessoal, linguagem procedural de escritórios, sistemas de informação automatizados.
- Nativas: aquelas que estão encaixadas no sistema, porém não acionadas diretamente, tais como: gerenciamento gráfico, gerência de microficha, capacidade multifuncional, memória.
- De Prestígio: aquelas que não são necessárias tais como projeto gráfico. (Chorafas,1982).

II.5. SISTEMAS INTEGRADOS

Um sistema integrado engloba as tecnologias de processamento de dados, telecomunicações e de equipamento de escritório através de uma rede de comunicação de dados a qual estarão conectados os equipamentos necessários à captura, transporte, armazenamento, processamento e apresentação da informação.

A integração de sistemas é um objetivo a ser permanentemente perseguido; a integração é um fator multidimensional, podendo existir em múltiplas formas: integração de ferramentas, integração dos meios (permitindo o manuseio da informação por dispositivos digitais, óticos, etc.), integração de tecnologias, integração através da distância (permitindo o acesso às informações independentemente da localização do usuário), integração através do tempo (equipamento de várias gerações podem conviver no mesmo sistema), integração do hardware e do software com o "orgware" (ambiente sócio-técnico da organização), integração de interface com o usuário (interface-padrão).

Nos software integrados se congregam mais de uma das funções das ferramentas de software citadas anteriormente, em um único pacote. (Carvalho,1986).

II.6. AMBIENTE DE ESCRITÓRIOS

Em um escritório, o trabalhador realiza atividades de manipulação de informação. Ele administra, recebe, edita, transforma, analisa, atualiza, classifica, arquiva,

recupera e distribui informação. A transmissão desta informação é feita por via oral, visual ou escrita.
Os trabalhadores de escritório processam informação e também geram novas informações.

Segundo Ellis, o escritório define-se como "um conjunto de procedimentos relacionados". Cada procedimento consiste de um conjunto de atividades ligadas mediante disposições temporais chamadas restrições de precedência. Para que uma atividade seja executada, pode-se precisar de depósitos de dados tais como arquivos e formulários. (Ellis, 1980b).

Dentro da Análise Estruturada de Sistemas define-se um empreendimento como um "conjunto lógico de funções que existem para fornecer um produto ou serviço".

Para Hammer, um escritório é "um componente de uma organização encarregado de realizar funções, definidas em termos dos objetivos e necessidades da organização". (Hammer,1980).

Lebensold define o escritório como "aquele componente de uma organização onde ingressa informação do exterior, desde o qual chega informação. É o lugar onde a informação é gerada ou transformada de forma que possa ser usada fora do escritório para produzir produtos, serviços e dinheiro. (Lebensold, 1982).

Segundo Chorafas, o escritório é o meio pelo qual se gerencia o fluxo de dados e textos necessários à operação do dia-a-dia. É basicamente uma fábrica de textos, dados e tomada de decisões. A informação que vêm do exterior dá inicio a uma resposta; essa resposta pode abranger ações específicas de uma ou mais pessoas e dependências. O resultado é apresentado na forma de uma carta, memo ou algum

outro formulário. (Chorafas,1982).

Segundo Lamersdorf, um escritório é uma organização de objetos (também chamados dados de escritório ou unidades de informação), procedimentos, atividades, localizações, comunicações e pessoal. (Lamersdorf,1986).

CARACTERÍSTICAS DO AMBIENTE DE ESCRITÓRIO

1. Os escritórios e suas atividades são distribuidos tanto em espaço como em tempo. Requerem uma coordenação contínua das atividades paralelas feitas em muitos lugares;
2. As atividades de escritório são às vezes rotineiras, embora estejam mudando frequentemente;
3. As atividades no escritório são interativas;
4. O trabalho de grupo envolve a participação simultânea de várias pessoas;
5. Os escritórios processam principalmente entidades simbólicas mais do que as físicas;
6. A informação manipulada no escritório vêm de muitas formas: na forma de texto e voz, ou de atividades de dados intensivos como planilhas;
7. A informação usada é desenvolvida tanto internamente como externamente. Informação interna é aquela cuja fonte é localizada na própria empresa;
8. A informação pode ser estruturada ou não. A informação estruturada é aquela rotineira, classificada, categorizada e formatada; é o tipo de informação tratada por sistemas convencionais de processamento de dados. As informações não estruturadas são por exemplo o texto livre, gráficos, mensagens verbais ou audiovisuais,

tabelas, números e estatísticas;

9. A informação é orientada; como no passado, para o desempenho da empresa; e também orientada para o futuro, como em projeções de vendas ou custos. (Hammer, 1980).

Um dos principais objetivos dos escritórios automatizados é resolver os problemas de manipulação de informação, os quais consomem tempo e são custosos. A manipulação de informação envolve basicamente as seguintes fases:

1) A criação de um documento;

2) Sua presença lógica e física;

3) Sua distribuição dentro da empresa;

4) Seu armazenamento;

5) Sua recuperação;

6) Sua distribuição externa.

Estas fases estão frequentemente intercaladas. A identificação apropriada destas fases requer uma reorganização de procedimentos e sistemas internos e uma revisão das comunicações externas. (Chorafas,1982).

A Figura 1 apresenta as Fases de Manipulação de Informação.

II.7. ESTAÇÃO DE TRABALHO

Uma estação de trabalho baseada em computador (micro ou mini) é um ambiente amigável ao usuário com o gerente e seus funcionários criando, recebendo, lendo e transmitindo informações através de dispositivos suportados pelo computador (vídeo, consoles, impressoras, etc.); os documentos são arquivados nos meios de suporte eletromagnéticos ou ópticos, e comunicados através de redes

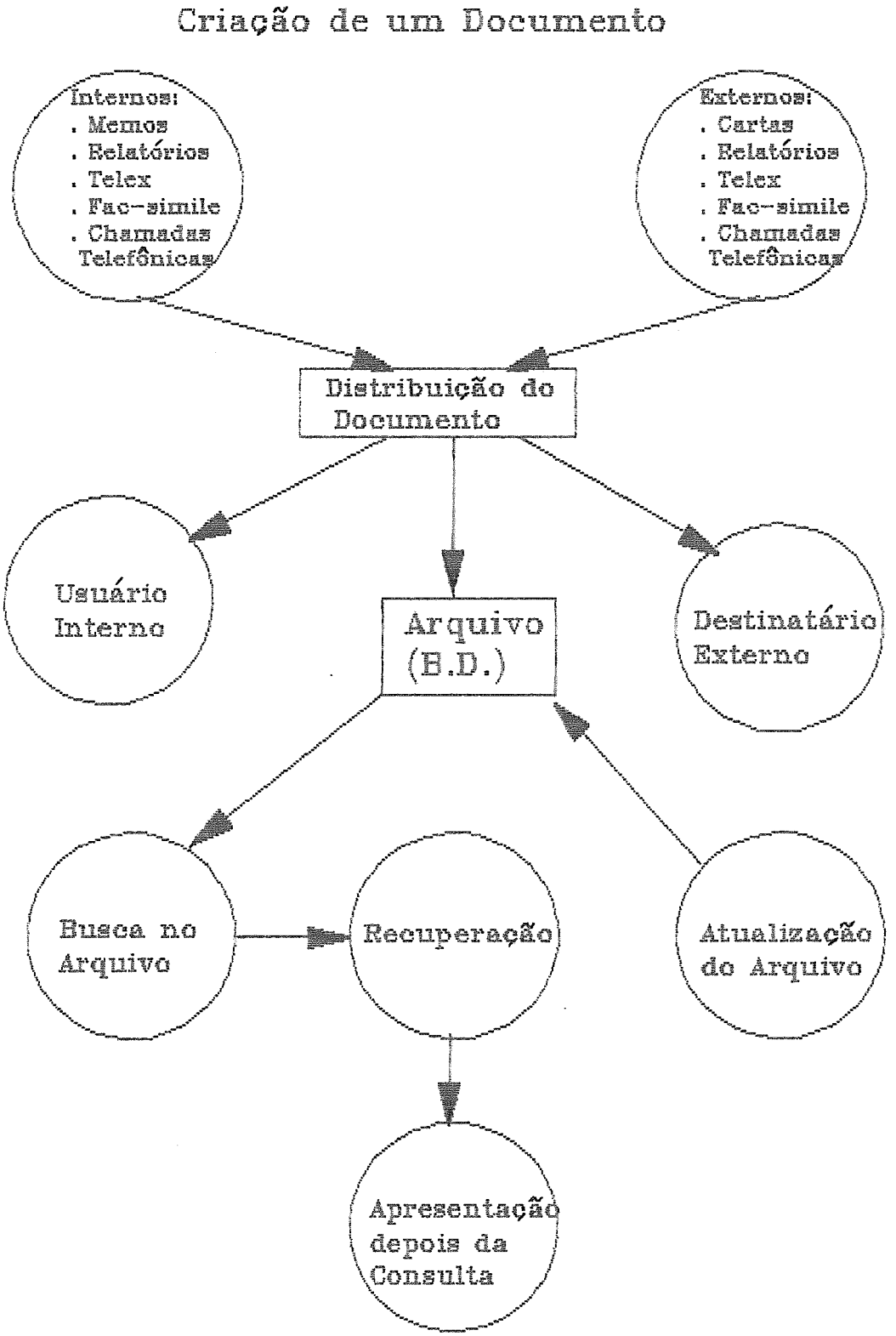


Figura 1. Fases de Manipulação de Informação

de telecomunicação.

A comunicação homem-máquina procura se manter rápida e eficaz com tempos de resposta razoáveis.

Além de melhorar a produtividade, um outro objetivo é fornecer um gerenciamento mais eficiente mediante a integração, organização, suporte e manutenção de vários tipos de serviços. (Chorafas,1982).

Os sistemas de escritório avançados são organizações complexas de objetos de escritório, procedimentos, atividades e localizações que incorporam estações de trabalho independentes com o pessoal de escritório, cooperando concorrentemente na execução das tarefas do escritório, contando com uma facilidade de comunicação. (Lamersdorf,1986).

Um ambiente de comunicações de uma corporação com estações de trabalho ligadas e gerenciadas como uma rede vai permitir que objetos pessoais de um usuário que precise trasladar-se a um outro local sejam transmitidos através da rede a uma estação alternativa, assim como permite ao usuário acessar a sua estação de trabalho desde qualquer nó da rede.

Estas estações possibilitam o acesso compartilhado a um banco de dados de informações comuns além do suporte de atividade em grupo.

Espera-se em um futuro próximo que seja corriqueiro o uso de voz em estações de trabalho que possibilitem a transmissão de mensagens sonoras (tais como: correio de voz, instruções para um assistente) e o reconhecimento de comandos verbais efetuados por um usuário autorizado. (Finch,1986).

A Figura 2 apresenta os tipos de serviços da Estação de Trabalho. (Chorafas,1982).

II.8. PROCEDIMENTOS DE ESCRITÓRIO

Segundo Hammer, um procedimento é uma estrutura por meio da qual são organizadas as tarefas individuais e as atividades a serem executadas pelos trabalhadores de escritório. O procedimento especifica quais passos vão ser tomados, por quem e em que ordem. (Hammer,1980).

Segundo Croft, eles são descrições das tarefas de escritório. (Croft,1984).

Segundo Tsichritzis, um procedimento consiste de três partes: uma condição, uma ação e uma notificação; a parte condição especifica os termos sob os quais é iniciado o procedimento; a parte ação especifica a operação executada pelo procedimento quando ele é iniciado; e a parte notificação que especifica os outros objetos a serem avisados quando conclui-se o procedimento. (Tsichritzis,1980).

Para Lamersdorf, as tarefas de escritório estão centradas ao redor de um conjunto de objetos ou dados de escritório; os quais são armazenados, recuperados e manipulados por um conjunto de procedimentos de escritório. Vários procedimentos de escritório podem ser agrupados em atividades complexas de escritório, que suportam a execução de uma tarefa de escritório de acordo com o objetivo global do escritório. (Lamersdorf,1986).

A maioria de procedimentos de escritório são "processos semi-estruturados". Estes processos abrangem tanto tarefas

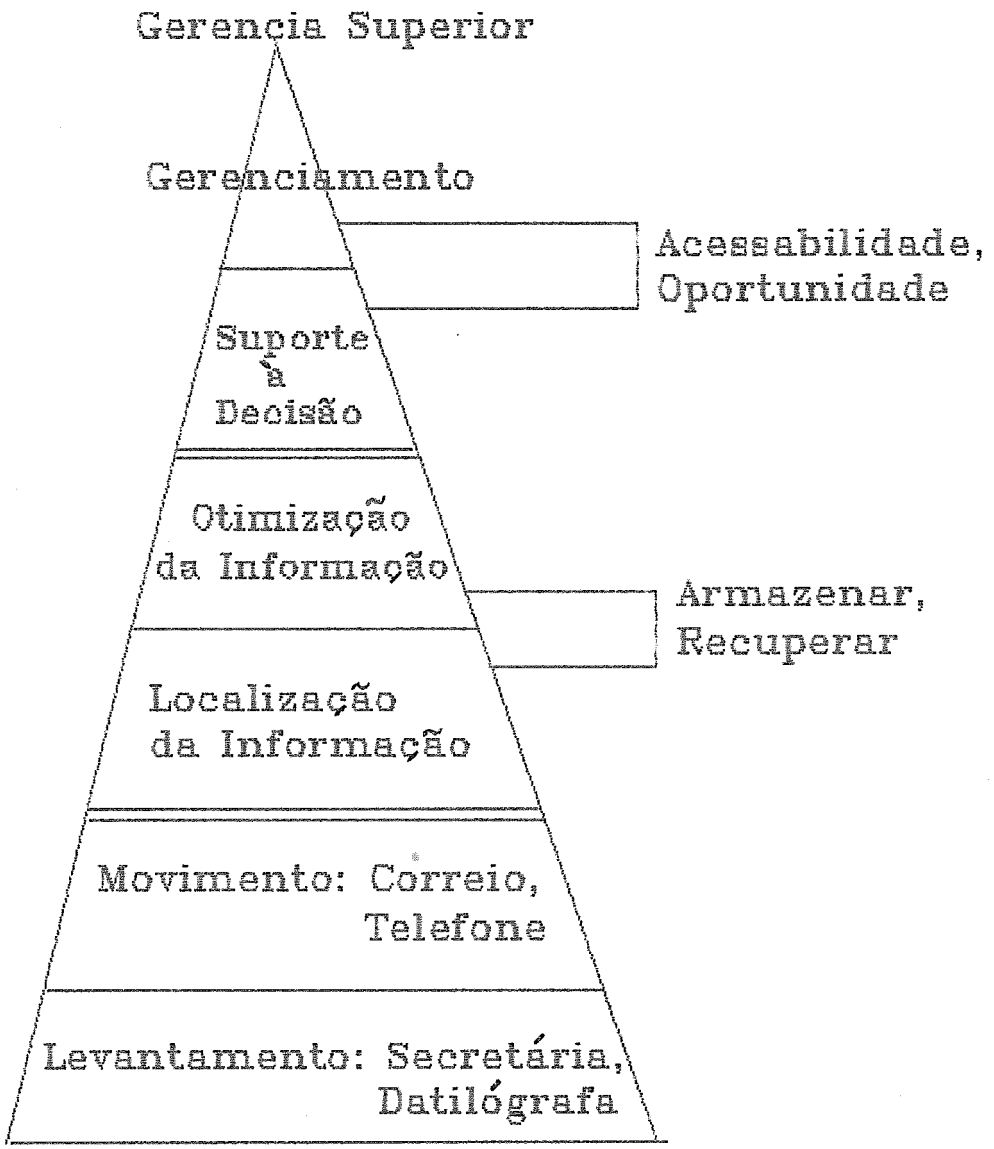


Figura 2. Tipos de Serviços da Estação de Trabalho

simples, altamente estruturadas e repetitivas (rotineiras como de processamento de informação) como tarefas não estruturadas e de mais alto nível (como tomada de decisões, gerência de informações, sequências de ações complexas e de interação com outro pessoal). (Hammer,1980), (Croft,1984).

Formalismos para procedimentos de escritório são usados para definir e analisar escritórios e para automatizar tarefas de escritório. São formalismos as linguagens de especificação que descrevem semânticas de escritório ou linguagens de descrição de eventos. (Croft,1984), (Lamersdorf,1986).

Os procedimentos, mais do que o "software" atuam como o ponto central da Automação de Escritórios. (Chorafas,1982).

II.9. SISTEMAS DE INFORMAÇÃO PARA ESCRITÓRIOS (SIEs)

Segundo Ellis, um SIE é composto de uma coleção de tarefas autônomas altamente interativas que são executadas em paralelo; estas tarefas incluem a preparação e gerência de documentos, de formulários, comunicação e apoio à tomada de decisões. (Ellis,1980b).

Segundo Carvalho, ele é um conjunto de operações básicas de manipulação de informação, organizadas de maneira sistemática, com o objetivo de efetuar racionalmente uma dada tarefa. (Carvalho,1986).

Segundo Hammer, ele é uma coleção integrada de componentes que suporta a operação de um procedimento de escritório. (Hammer,1980).

O maior desafio de um SIE é integrar os componentes para

reduzir a complexidade de interface do usuário ao sistema, controlar o fluxo de informação e aumentar a eficiência global do escritório. (Ellis,1980b), (Lee,1984).

Um SIE automatizado fornece melhores formas de comunicação e acesso à informação, além de velocidade, segurança e controles dos procedimentos de escritório. (Hammer,1980).

Os SIEs amíude apresentam-se ao usuário com interfaces para uma pequena coleção de aplicações genéricas; estas aplicações incluem editores de texto, correio eletrônico, etc.

Porém, dado que os escritórios existem para executar alguma função crítica dentro do contexto global de uma organização, um entendimento desta função conduzirá à definição de programas de aplicação especializados, as quais tratam procedimentos específicos a um escritório individual. Estes tipos de aplicações estão formando parte da maioria dos SIEs. (Zdonik,1984). Isto por sua vez implica, que o processo de análise e projeto do sistema tomem um papel primordial e que desenvolvam-se novas técnicas para facilitar a tarefa de programar estas aplicações.

Um software de aplicação-específica para um SIE tomará providência para o controle e fluxo do procedimento, administrará os dados e responsabilidades através das estações de trabalho, fornecerá uma variedade de facilidades de rastreamento e monitoramento permitindo que o procedimento como um todo seja efetivamente gerenciado e controlado.

Por outro lado, um SIE funcional fornecerá ao usuário de cada estação bastante poder computacional para executar

aplicações de processamento de dados e textos, assim como o acesso a grandes quantidades de informação, possibilitando que a informação possa ser compartilhada entre as estações de trabalho. (Hammer,1980).

II.9.1. DIFERENÇAS ENTRE SISTEMAS DE INFORMAÇÃO CONVENCIONAIS VERSUS "SIEs"

a) **Dados de escritório:** Um SIE deve suportar tipos de dados não estruturados contidos nos textos, anotações, gráficos, mensagens verbais e audiovisuais; deve suportar elementos de informação usados em grupos, como por exemplo o documento de tipo "carta".

b) **Dados de tempo:** são necessários para determinar o tempo de vida dos documentos ou das operações individuais, ou para se especificar a duração das atividades e as restrições de tempo; devem ser definidos como todos os operadores aplicáveis a eles. Precisa-se flexibilidade na definição de restrições de tempo.

c) **Atividades de escritório:** as atividades também são não-estruturadas; a mesma ação pode ser executada de vários modos; além disso, a maioria das atividades somente são executadas após que algum tipo de instrução tenha sido dada ao trabalhador. Precisa-se flexibilidade na execução de atividades.

d) **Interconexões de elementos:** existe um grande número de elementos de escritório relacionados através de várias conexões; geralmente os elementos necessários à execução do trabalho de escritório são distribuidos entre vários trabalhadores dentro do mesmo departamento, ou em

diferentes departamentos, ou fora do ambiente de escritório.

e) **Evolução do Escritório:** as restrições definidas sobre os elementos de escritório são as menos estáveis; as atividades também podem se modificar no tempo; os tipos de documentos também podem mudar, embora numa frequência mais baixa.

f) **Características de uso:** Os SIEs são altamente interativos; precisa-se projetar um sistema amigável ao usuário; algumas funções de conhecimento dentro do escritório saõ substituidos pelo uso destes sistemas.

g) **Filtro:** Uma das funções destes sistemas é filtrar a grande quantidade de dados para fornecer aos trabalhadores de escritório somente informação específica.

h) **Memorandos:** Através do planejamento automático de atividades, o sistema pode fornecer a função de comunicar aos trabalhadores as atividades a serem executadas e suas diferentes prioridades.

i) **Integração de funções:** está baseada na cooperação entre os elementos no escritório. Geralmente, um trabalho é feito por mais de um trabalhador em diferentes passos; embora, um trabalhador possa realizar várias funções ligadas entre sí.

j) **Impacto da Tecnologia:** As tecnologias dos SIEs ainda não estão estabelecidas e elas estão evoluindo rapidamente. (Bracchi,1984a).

II.10. IMPACTOS DA AUTOMAÇÃO DE ESCRITÓRIOS

A implantação da AE dentro de uma empresa afeta a organização de muitas formas. Este impacto vai ser analisado em três níveis: o da produtividade dos trabalhadores de escritório, o organizacional e o social.

II.10.1. IMPACTO NA PRODUTIVIDADE DOS TRABALHADORES DE ESCRITÓRIO

Segundo Toffler, os trabalhadores estão divididos em trabalhadores de "alta-abstração" composto de gerentes e profissionais, os quais executam principalmente tarefas não estruturadas como tomada de decisões; e os trabalhadores de "baixa-abstração" composto de funcionários, pessoal administrativo executando tarefas rotineiras. (Chorafas,1982).

O impacto dos sistemas de escritório automatizado na produtividade são:

a) Melhor uso dos recursos humanos - execução de mais trabalho com o mesmo número de pessoas;

b) Aumento do produto final;

c) Aumento da qualidade de decisão em trabalhos, produtos e serviços;

d) Aumento da eficiência - execução de tarefas em menos tempo;

e) Aumento da efetividade - melhoria na comunicação organizacional;

f) Aumento do controle do campo de ação - melhoria individual e flexibilidade organizacional.

(Chorafas,1982), (Clunie,1985).

II.10.2. IMPACTO ORGANIZACIONAL

Dado que o sistema não se limita a automatizar tarefas elementares como também procedimentos de escritório; o sistema vai automatizar completamente algumas funções implicando que os conteúdos dos cargos ou postos de trabalho sejam alterados. Alguns cargos podem ser extintos; por outro lado, novas funções e novos cargos são criados em decorrência da existência do sistema. A médio prazo é possível que este impacto leve a alterações na estrutura e modo de funcionamento da organização. (Carvalho,1986).

II.10.3. IMPACTO SOCIAL

Este abrange, entre outras, as áreas de educação, política, comércio internacional, desenvolvimento de novos produtos e as responsibilidades sociais.

Na área de educação, para o treinamento dos trabalhadores de escritório com as novas ferramentas de AE; na política, com o estabelecimento de uma política nacional de informática, dirigida para resolver o conflito econômico e social engendrado pela introdução de novas aplicações da tecnologia; uma análise do impacto potencial da AE pode fornecer pautas para o desenvolvimento de novos produtos; e a responsabilidade dos projetistas de sistemas para elaborar sistemas que apoiem o trabalho do pessoal de escritório, e dos meios de promover mudanças controladas da cultura organizacional. (Chorafas,1982), (Clunie,1985).

CAPITULO III

METODOLOGIAS E MODELOS PARA O PROJETO CONCEITUAL DOS "SIEs"

III.1. INTRODUÇÃO

Os SIEs estão cada vez mais afetados pelo uso de novas e sofisticadas tecnologias. Para suportar a análise e projeto dos SIEs, várias propostas de metodologias e modelos têm-se apresentados na literatura de pesquisas recentes. (Barbic,1985a).

As metodologias tentam especificar requisitos das fases anteriores à implementação, verificar a exatidão de um projeto e/ou serem usados como sistemas de projeto. (Ellis,1980b).

Há uma necessidade de metodologias sistemáticas que guiem o trabalho dos analistas de escritórios.

Os modelos SIEs por sua vez têm estendido os modelos tradicionais de sistemas de informação e de banco de dados incorporando as suas novas características. Eles são construidos no nível conceitual para que sejam tão independentes quanto possível das características de implementação. (Barbic,1985a).

III.2. METODOLOGIAS

Uma metodologia de análise permite fazer o processo de projeto mais fácil e seguro, acrescentando a racionalidade, qualidade e manutenibilidade na implementação dos SIEs.

O maior objetivo de uma metodologia é se obter uma descrição do escritório. O modelo usado na metodologia deve

descrever de maneira não ambígua tantos aspectos do escritório quanto possível, possibilitando ao pessoal do escritório validar o sistema, sugerir modificações e identificar inconsistências.

A metodologia deve agir como guia, provendo algumas soluções técnicas convenientes que levem em consideração o conjunto de problemas analisados do escritório, deve oferecer alguma indicação do critério a ser usado na avaliação de soluções possíveis e na eleição de ferramentas do projeto, além de prover interfaces diretas às fases de implementação. (Bracchi,1984a).

A seguir comentaremos algumas metodologias orientadas para SIEs.:

OAM/OSL: a metodologia OAM (Office Analysis Methodology) deriva diretamente da "abordagem funcional", que procura identificar as funções do escritório em termos da empresa; dá ênfase às atividades táticas e operacionais semi-estruturadas.

A linguagem OSL (Office Specification Language) representa a abordagem para o projeto de SIE. É uma linguagem de especificação inicial e implementação protótipo de requisitos. (Carvalho,1986), (Barbic,1985a).

MOBILE-Burotique: é uma meta-metodologia (conjunto de ferramentas de projeto), dá ênfase à análise de requisitos (sócio-técnicos) e opera num meta-nível, que determina os instrumentos corretos para o projeto, especificação de requisitos e para análise custo-benefício. Ele considera todas as fases do projeto de sistema. (Carvalho,1986), (Barbic,1985a).

OSIRIS: é uma metodologia para a especificação de procedimentos em um SIE. Esta metodologia decompõe o processo de projeto em duas fases: um projeto de procedimentos independentes de escritório e uma integração dos procedimentos.

O modelo semântico usado para representar um SIE incorpora elementos estruturais e de controle selecionados dos modelos SOS e ICN (Redes de Controle de Informação) respectivamente.

Para todos os elementos do modelo, uma representação gráfica foi desenvolvida, o que lhe permite ser amigável ao usuário. (Barbic,1985a).

INSYDE: o modelo e metodologia "Insyde" fornecem um meio para traduzir um conjunto de requisitos informais do usuário em especificações de dados e processos específicos.

A metodologia guia o projetista, primeiro através do processo de representar o ambiente de informação no modelo de fluxo de dados; e depois extrai desta especificação, os dados e requisitos procedurais necessários para implementar o sistema, os quais são expressos num modelo de banco de dados semântico incrementado por construções para especificar procedimentos.

O conceito unificado do modelo "Insyde" é o "evento", na forma de eventos de processos, eventos de aplicação e eventos de modelagem. (King,1985).

III.3. MODELOS

Modelos são abstrações limitadas da realidade, pois não é possível capturar todos seus aspectos e matizes. A limitação é expressa pela seleção de um subconjunto de atributos e estruturas relevantes ao sistema e por isso será feito o uso de modelos específicos.

O uso de modelos do escritório não pode ser evitado, cada trabalhador leva um modelo informal de suas atividades. Os modelos formais são úteis para propósitos de consistência, eles tendem a ser analíticos e explícitos. Os modelos informais são úteis para capturar aspectos sociais e de comportamento; eles tendem a ser implícitos e "ad-hoc". (Ellis, 1980a).

POR QUE MODELAR?

Os modelos matemáticos podem ser construidos em um nível de abstração bastante alto, de modo que as propriedades, generalizacões, diferenças individuais e inconsistências de sistemas complexos possam amiude ser descobertas, o que de outro modo não seriam aparentes.

Também porque sem uma modelagem não pode haver automatização.

Favorece-se a modelagem analítica dos SIEs porque estes sistemas são bastante dinâmicos (mudanças nos procedimentos, pessoal ou requisitos de escritório são frequentes), além disso a tecnologia destes sistemas está em uma etapa de formação e não existe uma teoria completa dos SIEs. (Ellis,1980a), (Ellis,1980b), (Carvalho,1986).

A seguir, apresentam-se duas classificações de modelos conceituais de SIEs segundo o tipo de visão do escritório: organizacional e técnica.

III.3.1. CLASSIFICAÇÃO DE MODELOS SEGUNDO A VISÃO ORGANIZACIONAL DO ESCRITÓRIO

É uma classificação de modelos comumente utilizada no estudo de sistemas de escritório e de organizações em geral. O modelo é escolhido em função do problema mais relevante da organização. (Ellis,1980b), (Carvalho,1986).

1. MODELOS DE FLUXO DE INFORMAÇÃO: eles tentam representar o trabalho de escritório em termos de unidades de informação (documentos, formulários) que fluem entre escritórios; eles nos permitem definir as operações executadas em cada unidade de informação, geralmente através de diagramas de fluxo. Estes modelos são os mais utilizados na análise de sistemas tradicionais.

   A ferramenta mais usada para construí-los é o Diagrama de Fluxo de Dados (DFD) da Análise Estruturada (Gane,1979) que consiste em um gráfico, onde são representados os fluxos dos dados do sistema de escritório e as transformações sofridas por estes dados.

2. MODELOS PROCEDURAIS OU FUNCIONAIS: eles dão ênfase ao trabalho de escritório orientado às tarefas, onde cada procedimento é projetado para executar uma tarefa particular, eles envolvem operações (passos procedurais) e operandos (unidades de informação), e também

identificam os papéis do pessoal de escritório.

A análise dos procedimentos permitem identificar possíveis ineficiências e gargalos. Estes modelos oferecem mais exatidão que os modelos de fluxo de informação. Para construi-los usam-se ferramentas de análise de sistemas tais como por exemplo:

- OSL (Hammer,1980): (Office Specification Language) é uma linguagem que descreve os procedimentos e problemas dos sistemas de escritório, permitindo ainda a especificação de soluções automatizadas; guarda alguma semelhanca com as linguagens de descrição de dados (DDL) utilizadas em SGBDs.
- ABL (Lebensold,1982): (Alternative Based Language) é uma linguagem poderosa para descrever paralelismos (eventos concorrentes e assíncronos) permitindo ao construtor do modelo descrever claramente e concisamente o comportamento do sistema.

3. **MODELOS DE BANCO DE DADOS**: eles tentam representar o trabalho do escritório em termos de banco de dados. Estes modelos fornecem construções especializadas para suportar aplicações de escritório; eles suportam a definição de objetos de escritório especializados, as ações primitivas (operações) que podem ser executadas sobre aqueles objetos e as regras que especificam vários tipos de restrições e relacionamentos.

Estes modelos conseguem gerenciar dados de escritório, os quais exibem uma estrutura diversa e complicada. (Harper,1986).

No capítulo IV são apresentados Modelos de Dados

para SIEs.

4. MODELOS DE APOIO À DECISÃO: eles estão baseados nas atividades de tomada de decisões dos gerentes e de outras pessoas do escritório. O modelo mais tradicional o trata como um processo de coleta e análise de informação; que permite ao usuário, através de ferramentas apropriadas, selecionar as informações, combiná-las, gerar índices, simular alternativas para novos cursos de ação.

A criação e manipulação de modelos tem sido um esforço de pesquisa em áreas de Ciências Gerenciais, Pesquisa Operacional, Econometria e outras.

Foram construídos Sistemas de Modelagem como: o SPSS para modelagem de problemas estatísticos e o MPX para modelagem em programação matemática.

Hoje, há uma tendência para utilizar novos conceitos na definição de modelos. Shen (Shen,1987) usa o termo "conhecimento" em vez de modelo, sugerindo um enfoque mais orientado por técnicas de Inteligência Artificial. Sol (Sol,1987) propõe abordar o problema de modelagem com uma "orientação a objetos". Usando esta abordagem, Konsynski (Konsynski,1986) propõe que modelos possam ser representados por objetos, com comportamento expresso por procedimentos e sujeitos a certas restrições impostas por assertivas.

Os Sistemas de Apoio à Decisão (SADs) são sistemas computacionais que têm facilidades para a construção e manipulação de modelos adequados ao problema em questão.

Nesta classificação de modelos também estão compreendidos os modelos de comportamento ou sociais e modelos sócio-técnicos.

III.3.2. CLASSIFICAÇÃO DE MODELOS SEGUNDO A VISÃO TÉCNICA DO ESCRITÓRIO

Esta classificação é baseada em elementos de escritório e o objetivo principal é a automação das atividades simples do escritório. Esta classificação reflete a diferente ênfase que os modelos dão aos elementos que constituem um escritório. (Ellis,1980b), (Bracchi,1984a), (Barbic,1985a), (Bracchi,1986).

Os exemplos a serem dados são modelos de especificação para a descrição conceitual de escritórios ou sistemas protótipos que usam este tipo de modelos na sua atividade.

1. MODELOS BASEADOS EM DADOS: A principal ênfase é colocada nos "dados". Os elementos básicos destes modelos são os tipos de dados e operações de manipulação de dados (armazenamento, recuperação, manipulação, transmissão); todas as atividades do escritório são modeladas como operações sobre os dados.

   Geralmente, os dados são representados por formulários em telas.

   O propósito principal é representar o escritório a partir dos objetos manipulados pelos trabalhadores do escritório (agentes); o fluxo de trabalho geral não está sob o controle do sistema.

Exemplos:

- OFFICETALK-ZERO (Ellis,1980b) : simula uma estação de trabalho baseada em microcomputador;
  elemento básico: formulários.
- OBE (Zloof,1982) : é uma extensão da linguagem de gerenciamento de banco de dados "QBE", que permite aos usuários descrever diretamente as suas aplicações;
  elementos básicos: formulários, objetos bidimensionais.
- OMEGA (Barber,1983) : é um sistema baseado em conhecimento;
  elemento básico: formulários.
- OFFIS (Konsynski,1982) : é um sistema para suporte de projeto de SIEs, dá ênfase à consistência e completude;
  elementos básicos: objetos, atributos, relações.

2. **MODELOS BASEADOS EM PROCESSOS**: A principal ênfase é colocada nas "atividades" executadas pelos trabalhadores de escritório (ou agentes). Dá-se ênfase na produção de uma visão integrada de todas as atividades executadas num escritório para executar certas tarefas, e não em dar a descrição de cada atividade.

   O principal objetivo do modelo é descrever o fluxo de controle do trabalho de escritório, e as regras que devem ser satisfeitas pelas atividades (regras de produção). O propósito é um controle geral do trabalho de escritório.

Exemplos:

- SCOOP (System for Computerization of Office Procedures): Sistema para modelar processos concorrentes assíncronos, usando redes PETRI aumentadas (APN) para especificar as ações e condições de tempo dos procedimentos de escritório, e usando sistemas de produção para monitorar o estado das instâncias do procedimento e determinar quando as ações têm que ser executadas.

  elementos básicos: procedimentos, transições, estados.

- ICN (Information Control Nets): O modelo dá ênfase ao fluxo de informação e à sua estrutura de controle; é uma técnica para o refinamento de especificação de requisitos;

  elementos básicos: procedimentos, atividades, depósitos.

3. **MODELOS MISTOS**: Estes modelos usam mais de um tipo de elemento como base para especificar o sistema. A maioria dos modelos recentes pertencem a esta categoria; eles permitem uma especificação mais completa dos elementos e seus relacionamentos.

   As regras nestes modelos servem para especificar semântica dos elementos de escritório; abstrações são usadas para definir as estruturas dos elementos.

   Exemplos:

   - OFS (Tsichritzis,1980) : modelo baseado nas transformações de formulários, relações e mensagens;

     elementos básicos: formulários, mensagens,

procedimentos.

- IML (Richter,1981) : estrutura de controle e estrutura de dados são modelados do mesmo modo; elementos básicos: redes de transição/predicado inscrito.
- SOS (Bracchi,1984b) : modelo baseado numa descrição semântica de documentos, agentes e atividades, integrada através do uso de regras de controle para governar o trabalho de escritório.
  elementos básicos: agentes, documentos, "dossiers", atividades, regras.
- OPAS (Lum,1982) : O sistema fornece processamento de formulários abstratos, além de capacidades de especificação e execução de procedimentos de escritório. Um formulário abstrato é uma abstração de um formulário no vídeo com relacionamentos bem definidos entre os campos do formulário.

  Usa-se a linguagem FORMAL (Forms Oriented Manipulation Language) para especificar processos de formulários abstratos; a linguagem permite especificar operações tais como: criar, inserir, eliminar, atualizar, imprimir, consultar ou compor; descrever os dados que constituem o formulário de saida e as suas restrições (faixa de valores, valores nulos); qualificar o processo descrevendo a fonte dos dados e as condições a serem aplicadas na seleção das instâncias de formulários.

  O modelo de procedimentos de escritório usado define uma "atividade" como uma operação básica feita por uma máquina ou um humano numa estação de

trabalho, e um "procedimento" o define como um conjunto de atividades relacionadas estruturalmente a serem executadas de um certo modo.

Um procedimento depois de especificado, pode ser executado; para isto o sistema fornece um número de operações para assistir ao usuário na sua execução. elementos básicos: formulários, procedimentos.

- OFFICETALK-D (Ellis,1982) : é uma extensão de Officetalk-Zero e ICN. Nele, os procedimentos de escritório são modelados por Redes de Controle da Informação (ICNs); cada procedimento pode ser descrito mediante um conjunto de atividades, um conjunto de depósitos de informação, e um conjunto de mapeamentos entre atividades e depósitos.

  Um ICN é traduzido a uma representação interna armazenada num banco de dados entidade-relacionamento. Os tipos de entidades incluem "atividade" (unidades de trabalho), "tarefa" (transação individual), "ator" (trabalhador de escritório) e "role" (papéis desempenhados no escritório).

  Os tipos de relacionamentos incluem "status" (estado) entre atividades e tarefas, "precedência" entre atividades, "player" (atuação) entre atores e papéis, e "performer" (executor) entre papéis e atividades.

  Com esta estrutura, um procedimento de estação de trabalho obtem o trabalho a ser feito por um ator nessa estação independentemente do resto do sistema.

  O OFFICETALK-D é implementado num ambiente altamente distribuído, possibilitando aos usuários

compartilhar e comunicar dados de escritório e informação de controle.

Além disso, uma interface de usuário baseada em gráficos apresenta um escritório eletrônico a múltiplos usuários.

elementos básicos: formulários, procedimentos.

CAPITULO IV

MODELO DE DADOS PARA "SIEs"

IV.1. EXTENSÕES DO MODELO RELACIONAL

IV.1.1. LINGUAGEM PARA AUTOMAÇÃO DE ESCRITÓRIO: "QBE/OBE"

A Linguagem de Gerenciamento de B.D. "Query-By-Example" ("QBE")

"QBE" é uma linguagem relacional não procedural de alto nível e de fácil uso, com uma interface unificada e conveniente para CONSULTA, DEFINIÇÃO, ATUALIZAÇÃO, CONTROLE e SEGURANÇA do banco de dados.

Com esta linguagem os usuários podem diretamente descrever suas aplicações, gerar e manter aplicações íntegras.

A linguagem introduz o conceito de programação bi-dimensional que é apropriado para os usuários não programadores que procuram a automação interativa de suas aplicações. Os objetos de dados são tabelas relacionais bi-dimensionais.

Para executar uma operação no B.D., preenche-se um "exemplo" da operação em tabelas esqueletos (mostradas na tela) as quais podem estar associadas com as tabelas atuais do B.D.

Os dois conceitos básicos do "QBE" são:

- A "Programação" dentro de tabelas esqueletos bidimensionais;
- A diferença entre um "elemento constante" e uma variável "elemento exemplo" : os elementos exemplos são sublinhados e os elementos constantes não.

O conceito de variável "elemento exemplo", permite executar uma grande variedade de operações tais como: referências cruzadas entre campos de tabelas; formular condições sobre os valores dos campos; mover dados de um objeto a um outro; derivar novos campos, busca de texto, etc.

"QBE" fornece funções agregadas tais como: CNT., SUM., AVG., MAX., MIN., UNQ.(UNIQUE); operadores sobre programas ou dados dentro de objetos tais como: P. (imprime ou exibe na tela), I. (inserir), D. (eliminar), U. (atualizar) e G. (grupar). Além disso, permite formular restrições de integridade (sentenças CONSTR) e controle de segurança (sob sentenças de autorização AUTH).

Exemplo de Consulta com referências cruzadas:

Obter uma listagem dos nomes, salários e departamentos dos empregados que vendem "CANETA".

Esta consulta precisa pesquisar nas tabelas EMP e VENDAS:

EMP	NOME	SAL	DEPT
	P.	P.	P.DP

VENDAS	DEPT	ITEM
	DP	CANETA

O elemento exemplo DP presente em ambas tabelas indica que deve haver uma comparação entre o campo DEPT na tabela EMP e o campo DEPT na tabela VENDAS que vende "CANETA".

A Linguagem "Office Procedures By Example" ("OBE")

A linguagem "QBE" estendida é denominada "Office Procedures By Example" ("OBE"), é uma versão simplificada do "System for Business" ("SBA") (Zloof,1977),(De Jong,1980).

Na linguagem "OBE", os objetos de dados são mais gerais, tais como: tabelas, formulários, relatórios, grafos, menus, estruturas de B.D. hierárquicas IMS, documentos (de audio e fac-símile). A linguagem fornece facilidades estendidas tais como: objetos programas explícitos; um subsistema de comunicação pelo qual os usuários podem enviar e receber objetos de dados (operação SEND); e uma facilidade de gatilho ("trigger") para especificar operações (ações) a serem executadas automaticamente quando certas condições, tempo ou eventos ocorrem.

Com estas facilidades, os usuários podem enviar e receber Correio Eletrônico sob condições pelo banco de dados.

(Zloof,1981),(Zloof,1982),(Whang,1987).

IV.1.2. LINGUAGEM DE PROPÓSITO ESPECIAL PARA FORMULÁRIOS DE ESCRITÓRIO "SEDL"

"SEDL" é uma linguagem não procedural de alto nivel usada para definir a estrutura lógica de formulários de escritório e as restrições de integridade correspondentes a cada formulário.

Um formulário é visto como um conjunto ordenado de campos contíguos. Um campo ou mais devem servir para identificar um formulário particular denominando a CHAVE da forma (KEY_FIELD).

Os tipos de dados são: inteiros, reais, cadeias de caracteres.

Alguns campos são grupados para formar objetos maiores denominados ESTRUTURAS. Ex: DATA: structure (mês,dia,ano);

Outros campos ou grupos de campos denominam-se LISTAS. Ex: linhas_ordem [n] (...) n: número de linhas da ordem

Um formulário é considerado como um banco de dados relacional muito pequeno. Cada LISTA é considerada uma relação, as suas filas são denominadas ELEMENTOS DE LISTA e correspondem às tuplas no modelo relacional, as suas colunas são denominadas ATRIBUTOS.

Os campos simples são vistos como relações de uma tupla e um atributo, e as estruturas como relações de uma tupla e vários atributos.

A linguagem de manipulação de dados usada é o "SQL".

"SEDL" implementa quatro classes de testes de erro: Testes de domínio (sobre os valores de campos individuais); Testes de estrutura (sobre os campos de uma estrutura); Testes de elementos de lista (sobre os campos de um elemento de lista particular); e testes de formulários (sobre o formulário inteiro).

- **Especificação de Domínios e estruturas:** o usuário tem que definir os tipos abstratos de dados simples e os nomes dos atributos das estruturas.

  O usuário pode especificar um teste de domínio, do contrário será examinado por valores "default".

  Exemplo de teste de domínio:

```
salário:
        real (7,2)
        values: ( 0 thru 9999.99 )
        error: "salário deve ser positivo"
        suspect_values: ( 0 thru 1.00 )
        warning: "valores suspeitos"
```

  Os testes de estrutura formam um subconjunto dos testes de formulários.

- Especificação de Formulários: consiste de uma descrição obrigatória da estrutura lógica do formulário e de uma definição opcional de testes de elementos de lista e testes de formulário.

Exemplo:

```
ordem: form;
   key_field : (...);
   data_fields : (...);
   testes_elementos_lista:
      .
      .
   testes_formulário:
      .
      .
end form;
```

"SEDL" é construído sobre uma sentença assertiva que é uma expressão lógica seguida por uma ação a ser executada se a expressão calculada for falsa. Estas ações são: exibir uma mensagem de erro ou exibir um "warning".

```
Exemplo:  código >= 6  and  código <= 12
             error: "código deve estar entre 6 e 12";
```

As sentenças assertivas podem ser construídas condicionalmente com as sentenças IF-THEN-ELSE, CASE e combinadas com blocos DO-END.

Os Testes de Elementos de Listas são uma sequência de sentenças "ON".

```
Exemplo:      ON linhas_ordem element entry:
                 rowsum must equal preço * quantidade
                 error: "rowsum não balança";
```

Se for necessária uma assertiva sobre cada elemento, pode-se usar um bloco "DO-END".

Os testes de Formulários que envolvem listas têm uma estrutura mais complexa. A linguagem "SEDL" fornece construções derivadas das linguagens de manipulação de dados relacional que permitem ao usuário verificar toda ou parte de uma lista, combinar listas, construir listas desde campos

simples e expressões, e usar partes ou propriedades de listas em expressões lógicas e aritméticas.

Para construir expressões de lista usam-se: constantes de lista, especificadores de lista (SELECT-FROM-WHERE), operadores de conjuntos (INTERSECT, UNION, MINUS). A linguagem fornece funções agregadas (AVERAGE, SUM, MIN, MAX, COUNT, SCALAR), operadores de comparação unários (ASCENDING, UNIQUE, SERIES_OF) e binários (MUST BE IN, MUST EQUAL, CONTAINS).

```
Exemplos:( SELECT  parte#  from INVENTARIO ) ASCENDING
                error:  "as partes devem estar em ordem
                        ascendente";

        ( SELECT   parte#   from   ORDENS  ) MUST  BE  IN
        ( SELECT   parte#   from   INVENTARIO  )
              error:   "uma parte não tem entrada no
                       inventário";
```

(Ferrans,1982).

IV.1.3. MODELO DE DADOS "NT" DO SISTEMA DE GERENCIAMENTO DE FORMULÁRIOS "SPECDOQ"

O sistema "SPECDOQ" pode automatizar ou semi-automatizar a manipulação de documentos baseados em formulários de escritório convencionais. Os documentos manipulados podem combinar textos, diagramas, tabelas, mesmo como números e cadeias de caracteres.

O sistema faz um gerenciamento uniforme dos dados, em termos de um modelo de dados relacional não normalizado que é denominado "Nested table" (NT).

No modelo, os dados são representados em tabelas grupadas (de colunas e filas), e as operações de manipulação

de dados são algébricas e altamente abstratas.

A estrutura de "NT" é representada num esquema de árvore. Uma tabela N é definida como:

$$N = (\text{nome-N}, NS, NO)$$

nome-N é um nome de NT, NS é um esquema, e NO é uma ocorrência.

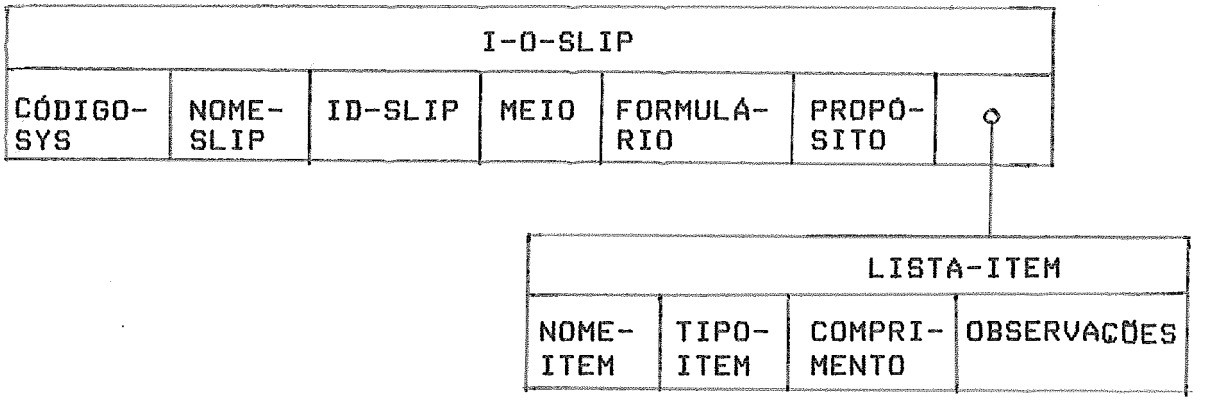
Um esquema NS é especificado por um grupo G chamado a RAIZ. A estrutura geral de um grupo G_i é:

$$G_i = (\text{nome-G}, F_1^i, \ldots, F_n^i, CG_1^i, \ldots, CG_m^i) \quad (n \geq 1, m \geq 0)$$

nome-G, é um nome de grupo, F_j^i ($1 \leq j \leq n$) é um campo, e CG_k^i ($1 \leq k \leq m$) é um grupo filho de G_i. O grupo CG_k^i pode ter à sua vez grupos filhos.

Cada campo tem um domínio de suas ocorrências de dados que pode ser: inteiro, real, cadeia de caracteres, texto e gráficos. Uma ocorrência NO (tabela de conteúdos de um NT) é um conjunto de "clusters" correspondentes à raiz.

Exemplo do esquema I-O-SLIP



I-O-SLIP e LISTA-ITEM são grupos, I-O-SLIP é a raiz, campos são por exemplo: CÓDIGO-SYS, NOME-ITEM.

As estruturas de dados hierárquicos de formulários podem ser mapeadas dentro de estruturas canônicas "NT".

As operações "NT" manipulam tabelas grupadas. Cada uma delas (exceto as operações CLEAR e DELETE) gera uma nova tabela.

O gerenciamento de dados ou o desenvolvimento da aplicação pode ser realizado através das operações de manipulação; como por exemplo a geração de relatórios e reorganização do B.D.

Uma sequência de operações que cumprem uma função pode ser armazenada e catalogada pelo sistema.

As operações "NT" são: PROJECT (selecionar e reordenar grupos ou campos especificados), NEST (grupar), FLATTEN (achatar grupos), ADD (inserir grupos/campos), TAKE (levantar "subclusters"), JOIN, SELECT, SORT, NAME (renomear grupos/campos), CLEAR (remover ocorrências), DELETE (eliminar tabela). (Kitagawa,1984).

IV.1.4. A LINGUAGEM DE CONSULTAS DE FORMULÁRIOS NATURAL ("NFQL")

A linguagem "NFQL" analisa um formulário para deduzir interpretações plausíveis, gerando depois consultas de banco de dados (para recuperar informação e atualizar arquivos) e especificações de funções (operações de cálculo) aplicáveis ao formulário. O sistema também deve ser capaz de fornecer as rotinas de leitura e validação dos dados ingressados por um usuário.

Os formulários manipulados pela linguagem não incluem informação completa sobre solicitações de recuperação e especificações de cálculos; em vez disso, são usados módulos

"inteligentes" para deduzir informação em falta.

A linguagem usa um modelo relacional para descrever o que deve ser feito para resolver relacionamentos computacionais.

O banco de dados de formulários define um formulário. Ele consiste da definição incompleta do formulário e da definição completa do formulário.

Definição Incompleta de Formulário:

Um formulário consiste de um título e quaisquer número de ítens com espaços vazíos a serem preenchidos. A definição inclui informação sobre o texto e a posição para o título, e a seguinte informação para os outros ítens: tipo de espaço (SINGLE se entrada-única, MULTIPLE se entrada-múltipla), frase chave, posição (par de inteiros que especificam a posição fila-coluna do primeiro caráter da frase chave), comprimento do espaço e a fonte ("U" se o valor do ítem é ingressado pelo usuário, "DB" se o valor é ingressado do B.D., e "C" se o valor é resultado de algum cálculo).

Definição Completa de Formulário:

Para completar a definição do formulário deve ser somado o seguinte: tipos de ítem, consultas de banco de dados e descrições computacionais.

Um tipo de ítem descreve o domínio das entradas esperadas para um ítem e fornece rotinas de entrada para ítens de formulários cujos valores são ingressados por um usuário.

Uma consulta de B.D. especifica o processo de recuperação para ítens cujos valores provêm do B.D. (ítens "DB"). Uma descrição de cálculo define a função e os

argumentos para ítens cujos valores são calculados (ítens "C"). As funções executadas sobre ítens são agregadas (tais como SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT) ou simples (+,*,/,-,%).

Estrutura de Dados (Data Frame):

Para o sistema de processamento de formulários, há uma correspondência única entre o tipo de um ítem e uma estrutura de dados. Uma estrutura de dados abrange o conceito de um ítem de dados com todas as suas propriedades essenciais. Cada estrutura de dados tem uma representação interna específica (STRING, INTEGER ou REAL), uma rotina de validação de entrada e uma rotina de transformação (para exibir os dados de saída); as funções simples e agregadas e as frases chave associadas a cada estrutura de dados. Frases chave, também estão associadas com funções.

Quando um formulário é analisado, cada frase chave seleciona 0, 1, 2 ou mais tipos de ítens possíveis. Se nenhum ítem é selecionado, o projetista deve decidir o tipo desejado; se mais de um tipo é selecionado, o tipo pode ser obtido pela informação de contexto ou o projetista é requerido para escolher o tipo. (Czejdo,1984).

IV.2. MODELO DE REDE "ENTIDADE-DOCUMENTO" DO SGBD "SOCRATE"

No Banco de Dados Textual desenvolvido, as descrições que contêm o objeto descrito, (somente descrições de textos eletrônicos) são denominados "Documentos", e as descrições de objeto-livre são denominadas "Entidades".

Entidades: O modelo de dados tem uma estrutura de rede. As entidades estão compostas de um conjunto de "Características" de vários tipos incluindo ligações (REFERS A).

Exemplo: A entidade Empregado é descrita pelas características: NOME de tipo ALPHA; EDADE de tipo INTEGER, etc.

Documentos: No modelo, os documentos estão ao mesmo nível que as entidades. Um documento é composto de um conjunto de "Características" e um "Texto". O Texto corresponde principalmente aos aspectos internos do documento (estrutura lógica e "layout") e as características aos seus aspectos externos (dados de identificação e manipulação do documento dentro da organização); as características descrevem o documento dentro de um ambiente de aplicação e o relacionam a outros documentos e entidades. Os textos estão compostos de uma cadeia de caracteres associada com uma árvore de "estrutura" (STRUCTURE) e uma "tabela de parametros" (PARAMETERS). A cadeia de caracteres é o conteúdo do texto, a estrutura em árvore descreve a sua estrutura lógica e os parametros descrevem suas partes variáveis.

A estrutura e parametros do texto estão determinados pelo tipo de texto (TEXT-TYPE).

Exemplo: empregados e seus contratos de trabalho estão representados pelas seguintes estruturas:

```
ENTITY  empregado
BEGIN
        nome ALPHA
          .
          .
        contrato  REFERS A  contrato-trabalho
END
```

```
DOCUMENT  contrato-trabalho   [contratojob]
BEGIN
        portador  REFERS AN  empregado
            .
            .
END
TEXT-TYPE  contratojob
       STRUCTURE
            .
            .
       PARAMETERS
         $referência : string
                .
                .
ENDTYPE
```

É possível compartilhar cadeias de caracteres entre textos que têm a mesma estrutura ou estruturas diferentes.

Operações no Banco de Dados

As operações podem ser feitas sobre entidades, documentos e textos.

Operações sobre Entidades: O sistema "SOCRATE" fornece uma linguagem de definição e uma linguagem de consulta. A primeira é usada para descrever a estrutura de entidades; e a segunda é usada para gerar instâncias de entidades, ingressar o ou atualizar valores de características, exibir valores de características e eliminar instâncias de entidades.

Operações sobre Documentos: envolvem operações para manipulação de características e textos. Os documentos como as entidades são selecionados através dos valores de suas características.

Para manipular características de documentos são usados os mesmos comandos fornecidos para entidades. A criação e eliminação de documentos são executados com os mesmos comandos usados em entidades, porém o sistema reage de modo diferente; por exemplo: no caso de eliminação, a estrutura e a tabela de parametros são removidos, porém a cadeia de

carateres do texto é mantida se ele era compartilhado com outros documentos.

É possível listar toda ou parte das instâncias de um documento.

Operações sobre Textos: Os textos podem ser manipulados através de suas estruturas e pode-se designar valores aos seus parametros. Várias operações são possíveis depois de se selecionar um texto, ou parte de um texto:

- Exibição e Impressão: Os textos são exibidos e impressos com um dado "layout" que pode ser definido pelo usuário. Cada padrão está associado com um tipo de texto. Se um elemento do texto não tem um "layout" associado, ele herda o "layout" do seu pai. O sistema gera um "layout" se nenhum padrão particular foi definido.
- Entrada de Texto: O ingresso de cadeia de caracteres de um texto é feito sob a direção do sistema, segundo a estrutura de árvore. Para cada parte do texto, o usuário pode escolher entre: deixar a parte vazía; copiar uma parte do mesmo texto ou do texto de outro documento; e digitar uma cadeia de carateres.
- Atualização: pode-se substituir parte de um texto; inserir uma nova parte; excluir uma parte opcional ou repetitiva; remover a cadeia de caracteres de uma parte.
- Designar valores aos parametros: os valores de parametros podem ser cadeias de caracteres ou "arrays" (segundo a declaração no tipo de texto). Os valores são designados dando o identificador do parámetro e seu valor, ou preenchendo os valores sobre o texto exibido no vídeo.
- Procedimentos de Correspondência: Para definir um procedimento de correspondência, o usuário deve: ingressar

um nome de procedimento; especificar o documento de saída; ingressar os nomes das entidades e de seus atributos, dos quais os dados serão extraidos, além da maneira de transformar e inserir os dados nos textos de saída.

(Kowarski,1982).

IV.3. MODELO ENTIDADE-RELACIONAMENTO DO SGBD "ODB"

O SGBD denominado "Office Data Base" ("ODB") é um sistema de recuperação e armazenamento de dados ajustados às necessidades de sistemas de escritório e estações de trabalho avançados.

O sistema reune um banco de dados Entidade-Relacionamento (E-R) com um sistema de arquivo hierárquico.

O modelo de dados é uma extensão do modelo E-R e fornece facilidades para suportar aplicações de escritório. O modelo de dados e sua interface de programação foram derivados do "NDB: Non-Programmer Data Base" (Winterbottom,1979).

O modelo consiste de três conceitos básicos:

1. Conjuntos e elementos de dados;
2. Relacionamentos e conexões;
3. Arquivos e diretórios.

- Um "elemento de dados" é uma unidade atômica do valor de dados. Representa uma entidade no B.D.. Cada elemento pode ser identificado por um identificador único (ID). O identificador permite recuperar o valor ou comprimento de um elemento, elimina-lo ou modificar seu valor. Exemplos: nome de uma pessoa, nome de uma publicação.

- Um "conjunto" é uma coleção homogênea de elemento de dados. Um conjunto pode ter um nome, embora é identificado por um ID único. O identificador permite inserir novos elementos no conjunto. Cada elemento de dados de um diretório "ODB" pertence a um conjunto. Exemplos: nomes de empregados, nomes de publicações.
- Um "relacionamento" é um mapeamento binário de um conjunto (conjunto fonte) a outro conjunto (conjunto objeto). O mapeamento pode ser: um-a-um, um-a-muitos, muitos-a-um, muitos-a-muitos. Um relacionamento pode ter um nome, embora também deve ser identificado por um ID único. Cada relacionamento tem um relacionamento inverso, o qual é mantido automaticamente pelo sistema. Exemplos: "Gerente é", "Editou".
- Uma "conexão" é uma instância de um relacionamento. Uma conexão não tem um nome ou um ID permanente. É identificada pela terna: ID do relacionamento, ID do elemento fonte e o ID do elemento objeto. Quando um elemento vai ser eliminado, o usuário tem a opção de elimina-lo se não existem conexões sobre ele ou remover todas suas conexões.
- Um "arquivo" é uma cadeia de bytes de comprimento variável. Usa-se para armazenar cadeias de textos longos, dados de gráficos/imagem ou audio digitado. São fornecidas operações "sub-string". Um arquivo é identificado por uma entidade armazenada num conjunto selecionado pelo usuário. Exemplo: arquivos podem armazenar o conteúdo de publicações.
- Um "elemento diretório" é o mecanismo pelo qual fornece-se uma hierarquia de diretórios e arquivos. Pode ser criado

em quaisquer conjunto, resultando na localização e inicialização de um diretório separado. Por sua vez, esse novo diretório pode conter arquivos e diretórios, permitindo assim uma árvore de diretório.

Um diretório é identificado pelo ID do seu elemento diretório no diretório pai.

- Um número de Conjuntos e Relacionamentos de Sistema formam o catálogo de dados definidos pelo usuário. Eles são Conjuntos e Relacionamentos predefinidos, os quais contêm os meta-dados de um diretório "ODB".

Os conjuntos "DBSET" e "DBREL" contêm os nomes de todos os conjuntos e relacionamentos respectivamente.

Relacionamentos predefinidos e seus Conjuntos Fonte e Objeto são:

```
DBMEM  :  DBSET  -->  *
DBFROM :  DBREL  -->  DBSET
DBTO   :  DBREL  -->  DBSET
```

O relacionamento "DBMEM" é usado para especificar todos os elementos no conjunto identificado pelo elemento fonte. Os relacionamentos "DBFROM" e "DBTO" são usados para especificar os conjuntos Fonte e Objeto de cada relacionamento.

Operações

Dado que o B.D. "ODB" é definido recursivamente, as operações de definição de dados são as mesmas que as operações de manipulação de dados.

A criação de um novo conjunto é feita inserindo um elemento no DBSET. Se um elemento do DBSET é eliminado, então o conjunto associado e todos os seus elementos são eliminados.

A criação de um novo relacionamento é feita inserindo o nome

do relacionamento no DBREL, e ligando-o ao elemento que representa o Conjunto Fonte no DBSET, via a relação DBFROM, e ao elemento que representa o Conjunto Objeto, via a relação DBTO.

O sistema fornece uma interface de consulta e atualização.

Operações de consulta: os operandos e o resultado de uma operação de consulta são uma lista de IDs de elementos.

Há 3 tipos básicos de consulta:

- Recuperação de um elemento específico, via o ingresso do seu ID;
- Recuperação dos membros de um Conjunto através de:

  1. ⟨ Nome do Conjunto ⟩ = ⟨ valor ⟩

     Se ⟨valor⟩ não é fornecido, recupera-se o conjunto íntegro

  2. ⟨ Nome do Conjunto ⟩ = ⟨valor1⟩ ... ⟨valor2⟩

     permite recuperar os elementos com prefixo entre ⟨valor1⟩ e ⟨valor2⟩

- Aplicação de Relacionamento: desde uma lista de elementos, identificada por uma consulta, os relacionamentos a seguir permitem obter outra lista

  ⟨ consulta ⟩.⟨ nome do relacionamento ⟩

  Ex: "EMP=SMITH.GERENTE.ESCREVEU"
  /* recuperará as publicações do gerente de Smith */

Os três tipos básicos de consulta podem ser combinados usando operadores de união ("+"), interseção ("*") e diferença ("-") de lista, assim como parêntesis para formar consultas mais complexas.

O sistema fornece funções para criar listas vazias, e para inserir e eliminar membros de suas listas (Choy,1984).

IV.4. MODELO DE DADOS RECURSIVO PARA MODELAR OBJETOS DE ESCRITÓRIO

Um modelo de dados recursivo (Lamersdorf,1984a) estendido via mecanismos de estruturação de dados adicionais para representar objetos abstratos e primitivas operacionais dos tipos adicionados é proposto por Lamersdorf (Lamersdorf,1984b), (Lamersdorf,1986).

O modelo permite representar objetos de dados não-formatados com componentes que variam, os quais estão semanticamente interrelacionados; isto é, como os requeridos nas aplicações de escritório.

Primitivas do Modelo de Dados Recursivo

O modelo permite a representação de objetos de dados estruturados de comprimento variável, geralmente grupados numa profundidade variável.

Para cada tipo de dados definido recursivamente ("RcsType") é fornecido um conjunto de diferentes "geradores de estrutura" ("Gen_i"). Os geradores de estrutura estão baseados em conjuntos limitados de componentes "tipo de dados" ("CompType_ij"), os quais a sua vez contêm outros tipos de dados recursivos ou contêm tipos de dados simples.

Componentes únicos podem ser identificados pelos "seletores de componentes" elementares ("sel_ij") para selecionar um componente de uma variável recursiva. Assim,

var_recursiva [sel_ij] = valor_componente_ij

A seguinte estrutura define um tipo de dados cujo conjunto de valores consiste de todos os valores de dados estruturados hierarquicamente.

```
TYPE  RcsType  =  (  Gen_1  (sel_11:  CompType_11;  ...  ;
                  sel_1n:  CompType_1n) / ... / ...
                  /  Gen_k  (sel_k1:  CompType_k1;  ...  ;
                  sel_km: CompType_km));
```

Exemplo de um Tipo Recursivo:

```
DocumentoType = (Carta (remetente,receptor: NomeType;
                        de,a: EndereçoType;
                        data_correio,      data_recebido:
                        DataType; conteúdo: TextoType)
                /Formulário  (...)/Memorando (...)/ ...);
NomeType = Nome (primeiro, sobrenome: PalavraType);
PalavraType = LIST OF CHAR;
```

O modelo fornece o suporte de um conjunto de "funções características" booleanas ("is-Gen_i") que permitem reconhecer um valor recursivo construído por meio do gerador "Gen_i"

```
is-Gen_i (var-recursiva) = TRUE
```

Extensões do Modelo de Dados Recursivo

O modelo apresenta notações breves para aquelas estruturas semânticas que representam conceitos que ocorrem frequentemente, tais como objetos ordenados (listas), série de objetos (conjuntos) e identificação de objetos (mapas); os quais são descritos a seguir:

- Objetos ordenados: através de um mecanismo de estruturação de dados do tipo "lista" de comprimento variável;

```
ListaType = LIST OF ElementoType;
```

Exemplo: ArquivoType = LIST OF DocumentoType;

As instâncias de lista única são geradas aplicando um gerador padrão para listas:

"ListaType (....)"

O modelo fornece primitivas de operações de lista tais como: seletores de elementos (Ex:LAST,REST,ELEMS), LENGTH

(número de elementos), VOID (prova booleana por zero elementos), APPEND (anexa elemento à lista), "lista [i]" (seleciona o i-ésimo elemento), etc.

- Série de Objetos: através da estruturação do tipo "conjunto" (SET). Nos conjuntos, a ordem dos elementos não é relevante e não se permite a duplicação de elementos.

ConjuntoType = SET OF ElementoType

Exemplo: GabineteArquivoType = SET OF GavetaType;

As instâncias de conjunto são geradas aplicando um gerador de conjunto padrão:

"ConjuntoType { .... }"

As restrições de conjunto estão baseadas em expressões de predicado de primeiro-ordem:

{ (elemento_id) IN (conjunto) SUCH THAT (predicado) }

- Identificação de Objeto: através do "mapeamento" de identificadores de objetos aos objetos a serem identificados. Mapas são conjuntos de pares que pertencem ao produto: Domínio x Contradominio; Domínio e Contradominio são conjuntos; dois pares de elementos não podem ter o mesmo valor de domínio.

MapaType = (DomínioType --) ContradominioType)

Exemplo: GavetaType = (IdArquivoType --) ArquivoType)
IdArquivoType = TOKEN

As instâncias de mapas são geradas usando um gerador de mapa:

MapaType = { d_1 --) r_1, ... , d_n --) r_n }

As operações sobre mapas são: DOM (conjunto de valores Domínio), RNG (conjunto de valores Contradominio), "+" (estensão do mapa), "&" (atualização do mapa), "mapa[argumento]" (aplicação de um mapa a um argumento).

- Integridade de Objetos de Dados: o modelo permite formular predicados "invariantes" (INVARIANT) booleanos com uma definição de tipo de dados, para impor restrições de integridade adicionais dependentes da aplicação.

Modelar Operações sobre Objetos de Escritório

No modelo proposto, as primitivas operacionais dos tipos de objetos estendidos já descritas, permitem modelar os aspectos operacionais dos SIEs. Operações mais poderosas sobre objetos recursivos podem ser definidos (recursivamente) em termos dos operadores elementares. Adicionalmente, uma seleção (algumas vezes complicada) pode ser expressada recursivamente usando uma definição "SELECTOR" parametrizada, genérica.

O caso mais simple de manipulação de objeto de dados é selecionar um objeto ou um componente de objeto, e substitui-lo por meio de uma operação de designação (usando o operador ":=").

A inserção, eliminação ou atualização de instâncias (componentes) de objetos é suportada pelos operadores de atualização orientados a conjuntos ("+","-","&") e pelos operandos construídos por meio das primitivas operacionais.

- Procedimentos de Escritório: O modelo fornece suporte adicional para modelar comunicação de dados, via as primitivas:

  SEND <valor-objeto-de-dado> TO <id-estação-de-trabalho>;

  RECEIVE <variável-objeto-de-dado> FROM
  <id-estação-de-trabalho>;

  A comunicação entre processos é permitida pelas primitivas de sincronização:

AWAIT <id-procedimento> ON <estação-de-trabalho>;

INITIATE <chamada-procedimento> ON <estação-de-trabalho>;

IV.5. MODELO DE ARQUITETURA DE DOCUMENTOS DE ESCRITÓRIO ORIENTADO A OBJETOS

O modelo proposto por Horak (Horak,1984) fornece a descrição das estruturas internas e correlações de documentos de modo misto independentes da aplicação, o que serve para facilitar o processamento e intercâmbio de documentos.

Um documento de modo misto tem informações tais como texto, dados, gráficos e imagens fotográficas dentro de páginas e parágrafos. (Rabitti,1985).

Pelo padrão ODA (Horak,1985), os documentos estão caracterisados por duas estruturas diferentes: a estrutura lógica e a estrutura "layout"; ambas estruturas são hierárquicas e compostas de objetos.

No modelo, uma CLASSE de documento é descrita pela "descrição genérica" que consiste de:

- Um conjunto de definições de tipos de objetos lógicos;
- Um conjunto de definições de tipos de objetos "layout";
- Um conjunto de regras de relação de correspondência.

A descrição da classe de documento pode conter em paralelo vários conjuntos de definições de objetos "layout", cada um dos quais define uma SUBCLASSE de Documento.

Classes de Documentos são por exemplo: cartas de negócios, relatórios, memorandos, formulários, etc.

Definições de Tipos de Objetos (lógicos ou "layout")

consistem de um nome de tipo e diferentes conjuntos de regras: Construção, Propriedade e Relação Intraestrutura.

- Regras de Construção: Devem ser capazes de expressar construções alternativas e recursão. Os construtores usados são: sequência, "array" e agregação.

  sequência: especifica uma ordem sequencial para os componentes de um objeto; os componentes são do mesmo tipo e são sequencialmente acessiveis (Ex: o corpo de uma seção consiste de uma sequência de parágrafos).

  "array" (n): especifica uma ordem uni ou bi-dimensional para os componentes de um objeto. Eles são acessados direitamente usando índices (Ex: tabelas).

  agregação: especifica uma ordem sequencial ou nenhuma ordem particular para os componentes de um objeto, os quais podem ser do mesmo tipo ou de tipos diferentes. Estes componentes são acessados direitamente pelos seus nomes (Ex: um corpo de documento pode consistir de um abstrato, uma tabela de conteúdos, a parte principal com seções, e uma lista de fontes bibliográficas).

- Regras de Propriedade: permitem avaliar os valores de propriedades dos objetos. As propriedades definem aspectos particulares dos objetos lógicos e "layout". Uma propriedade é de um certo "tipo" e tem um valor específico. (Ex: as "caixas" de objetos "layout" têm propriedades do tipo "tamanho" e "posição").

  Uma regra consiste de um tipo de propriedade e uma descrição de como se obter o valor de propriedade.

- Regras de Relação Intraestrutura: descrevem o tipo de relação e o valor de relação entre instâncias de tipos de objetos que podem ser estabelecidos automaticamente

(Exemplos de relações intraestrutura entre objetos "layout" são a relação "overlay" e a relação "flow link"; flow link: bloco parágrafo - bloco parágrafo).

As Regras de Relação de Correspondência descrevem as possíveis relações de correspondência entre instâncias de tipos de objetos lógicos e instâncias de objetos "layout". Podem existir várias regras para o mesmo tipo de objeto lógico.

A abordagem para regras de construção está baseada na produção de gramáticas de contexto livre, as regras de propriedade estão baseadas sobre regras evaluativas de atributos e as regras de relação baseadas sobre predicados.

IV.6. MODELOS DE DADOS SEMÂNTICOS

IV.6.1. MODELO DE DADOS SEMÂNTICO "ADABTPL" PARA SUPORTE DE ARQUITETURA DE DOCUMENTOS

O modelo de dados "ADABTPL" proposto por Croft (Croft,1987) está baseado na definição de restrições sobre tipos que permite que padrões de arquitetura de documentos ODA (Horak,1985) sejam definidos e garante que tipos de documentos individuais se ajustem àqueles padrões.

Além das facilidades básicas para definir agregação, generalização e "instantiation", o modelo de dados fornece um meio formal para definir restrições sobre tipos e especifica como estas restrições são herdadas e especializadas pelos subtipos.

Neste modelo, os tipos são construídos desde os tipos

abstratos de dados primitivos: tuplas, listas e conjuntos (sets) finitos, cada um dos quais tem uma semântica formal.

Os construtores de tipo disponíveis incluem: enumeração, indexação, agregação conjuntiva (usando o construtor "[ ]"), agregação disjuntiva usando uma união de tipos discriminada, derivação de um novo tipo, a partir de um tipo antigo (pelo uso de um predicado dentro de uma cláusula "WHERE"), refinamento ou enriquecimento de um novo tipo desde um tipo antigo (pelo uso da cláusula "WITH"), e uma construção de encapsulação para definir tipos abstratos de dados.

As operações no modelo são especificadas em 3 níveis:

- Os tipos abstratos de dados primitivos tais como SET têm operações associadas;
- Operações podem ser encapsuladas com tipos abstratos de dados definidos pelo usuário; e
- Transações de banco de dados.

O modelo tem semelhanças com os modelos de dados IFO (Abiteboul,1984) e GALILEO (Albano,1985).

No modelo, o esquema de banco de dados é visto como um conjunto de definições de tipos que culminam na definição do tipo objeto "banco de dados". As restrições são especificadas via cláusulas "WHERE" nas definições de tipos componentes e na definição do tipo banco de dados.

Um tipo é visto como um conjunto de todos os membros possíveis do tipo e é definido pelas propriedades necessárias que os membros devem ter para pertencer ao tipo. Os dois relacionamentos em tipos são: membro ou instância do tipo é um relacionamento de um objeto a um tipo, e o relacionamento subtipo/supertipo.

O exemplo seguinte define um tipo documento-simple, cada instância do tipo é uma tupla que tem pelo menos um autor

```
documentosimple-type =
        [título : text;
         autor : list of string;
         corpo : text]
        WHERE NOTEMPTY (autor);
```

Há duas maneiras de definir subtipos:

- Derivação: um subtipo é formado adicionando uma restrição a um tipo.

```
Exemplo: carta-type = documentosimple-type
                      WHERE LENGTH (autor) = 1;
```

O documento carta só tem 1 autor.

- Enriquecimento: é formado um novo tipo tupla adicionando um novo componente ou componentes a um tipo tupla existente.

```
Exemplo: relatório-type = documentosimple-type
                          WITH [data:string];
```

Os conjuntos ou relações de instâncias de tipos de objetos são declarados como tipos separados.

```
Exemplos: carta-rel = set of carta-type;
          simpledocs-rel = set of  documentosimple-type
                           WHERE   KEY   (simpledocs-rel,
                                          título, autor);
```

O modelo também fornece um construtor de tipo: união discriminada.

```
Exemplo:  seção-type = union [ P --> parágrafo-type,
                               F --> figura-type,
                               T --> tabela-type ];
```

Cada seção pode ser um parágrafo, figura ou tabela. Os símbolos P, F e T são os discriminadores do tipo e podem ser usados numa sentença CASE.

Para o refinamento de descrições de tipo com o propósito de produzir novas descrições de tipos, pode-se definir um TIPO PARAMETRIZADO, isto é, um tipo é definido usando uma

variável de tipo em lugar de um nome de tipo.

```
Exemplo:  relatório-geral-type (corpo-type) =
                    [ id-relatório : string;
                      cabeceira : cabeceira-type;
                      corpo : corpo-type ];
          relatório-sumário-type =
              relatório-geral-type (corpo-sumário-type);
          corpo-sumário-type =  ...
```

A flexibilidade para definir subtipos no modelo é uma característica essencial para representar padrões de arquitetura de documentos.

No modelo pode-se definir tipos recursivos através do construtor GENPLEX. Um tipo recursivo pode ser definido como sendo uma combinação de tupla recursiva e união discriminada.

No seguinte exemplo "lógico-composto-type" é um tipo recursivo

```
lógico-composto-type =
     GENPLEX
     [ lógico-básico-type ---> básico-type
                               WITH [ b-log-id : number ];
       log-comp-type  --->
           [ característica : exemplo-type;
             componentes:
                          list of lógico-composto-type  ]
       WHERE NOTEMPTY (componentes) ];
básico-type = union  [ P ---> parágrafo-type,
                       D ---> data-type,
                       A ---> autor-type,
                       T ---> título-type ];
```

As facilidades de tipos de dados abstratos no modelo fornecem uma capacidade para definir e encapsular operações sobre tipos.

As facilidades de definição e transações fornecidas pelo modelo podem ser usadas para descrever um padrão de arquitetura de documentos (ODA), relacionar tipos de documento particular àquele padrão e criar instâncias dos documentos. Tipos de documentos são criados como subtipos de um tipo ODA.

Os componentes lógicos de um tipo de documento são definidos como subtipos dos tipos padrões ODA usando predicados de restrição. As instâncias de documentos são criadas com operações "insert", "update" e "delete" fornecidas nas transações ADABTPL.

IV.6.2. MODELO DE DADOS DO "OBJECT MANAGEMENT SYSTEM" ("OMS")

"OMS" é um tipo de sistema de B.D. para escritórios que foi proposto por Zdonik (Zdonik,1984). O modelo de dados "OMS" fornece um conjunto de construções de modelagem dirigidos para resolver as questões básicas de gerência de informação para escritórios.

O modelo "OMS" é similar ao modelo "Semantic Database Model" (SDM) (Hammer,1981), embora ampliado para liga-lo mais proximamente às aplicações de escritório.

Objetos

Um objeto corresponde a uma entidade no domínio da aplicação com significado para o projetista. O modelo de objetos é similar ao do SMALLTALK (Goldberg,1981). Os objetos são instâncias de um ou mais tipos. Um tipo é um padrão que define o comportamento das suas instâncias pela especificação de um conjunto de operações que podem ser aplicados as suas instâncias. Um tipo pode estar relacionado a outro tipo como: Supertipo/Subtipo. Um subtipo herda todas as operações dos seus supertipos.

Classes

são conjuntos de objetos cujos membros são persistentes.

Elas podem ser Classes Base (definidas independentemente de outras classes, Ex: a classe de Empregado) e Classes Derivadas (definidas em termos de classes definidas previamente, Ex: a classe de Empregados Supervisores).

As classes derivadas podem ser definidas usando operações cujos resultados sejam conjuntos, tais como seleção de predicado, seleção manual, união, interseção e diferença de conjuntos.

Atributos

Descrevem as propriedades estáticas de um objeto e não têm uma existência independente do objeto. O nome do atributo deve ser único dentro do conjunto de atributos definidos para um tipo de objeto. Cada atributo tem uma classe denominada "classe valor", a qual contem valores do atributo.

A operação GET permite obter o valor do atributo e a operação SET permite dar um valor ao atributo.

Versões

O trabalho de escritório se caracterisa pela evolução dos objetos através de estados intermédios. O modelo fornece o "conjunto versão" que é uma coleção ordenada de versões de um objeto.

O mecanismo de versões permite aos usuários construir conjuntos versões lineares (as versões são coletadas numa sequência linear) e versões alternativas (versões cujo predecessor tem mais de um sucessor); também é possível gerar uma nova versão desde peças de várias alternativas.

As operações sobre Conjuntos de Versões são: Criar-Conjunto-Versão, Inserir-nova-versão, Eliminar-versão e Última-versão.

Os conjuntos versão são instâncias do tipo Conjunto e portanto dispoem de todas as operações sobre conjuntos (Iteração, Seleção-predicado, União, Interseção e Diferença).

O mecanismo de versão é usado pelo sistema para manter informação das mudanças: na definição de um tipo e na estrutura de hierarquia do tipo.

Conteúdo

Os objetos (objetos compostos) podem conter outros objetos como componentes.

Por exemplo: os componentes de um relatório podem ser os seus capítulos, apêndices e bibliografia.

É conveniente grupar estes componentes em conjuntos; assim, por exemplo: o conjunto componente capítulos consiste de um conjunto de objetos do tipo Capítulo.

O sistema permite o compartilhamento de objetos. Por exemplo: um capítulo pode ser componente de dois ou mais relatórios.

Para expressar uma mudança de conteúdo, usa-se o mecanismo de versão. Por outro lado, os valores de atributos de um objeto podem mudar sem causar que uma nova versão seja criada necessariamente.

Propagação de Versões

Criar novas versões dos componentes de baixo-nível de um objeto pode causar a criação de novas versões de cada um dos objetos de mais alto-nível que os contêm.

A definição do objeto especifica que componentes devem ter as suas mudanças propagadas.

Por exemplo: no esquema para a Classe Capítulos, uma definição do componente Parágrafos é:

Parágrafos: Set of Parágrafos
Propagate all changes

significa que modificações a quaisquer parágrafo de um capítulo causaria que seja criada uma nova versão daquele capítulo.

Referências

Uma referência é a forma em que se nomeia ou refere-se a outros objetos no B.D.

Quando é dado um valor a um componente de um objeto (via uma das operações "set-component" para aquele tipo de objeto), uma referência àquele valor é construída e conservada pelo objeto.

Uma referência é assim um outro tipo de objeto e tem operações associadas tais como: "Dereference" (retorna o objeto ou conjunto de objetos "referent") e "Criar-Referência".

Há dois tipos de referências: referências estáticas (refere-se a um objeto particular ou conjunto de objetos) e referências dinâmicas (o objeto ou conjunto de objetos pode variar com o tempo).

Herança

No modelo, os atributos de uma classe são herdados de suas superclasses; e também é possível herdar atributos via a hierarquia do conteúdo. Assim, um objeto pode herdar atributos dos objetos que o contêm.

Este tipo de herança é especificado pelo projetista no esquema pela expressão "extent" ligada ao atributo. Por exemplo: o atributo autor definido na classe Relatórios pode ser disponível aos objetos desde Capítulos até Parágrafos

Autor: Pessoa
Extent via Capítulos is through Parágrafos

V.6.3. MODELO DE DADOS BÁSICO PARA ESCRITÓRIO "MINSTREL-ODM"

No modelo hierárquico semântico denominado "MINSTREL-ODM" proposto por Harper (Harper,1986), os dados de escritório são vistos como objetos de escritório persistentes.

O modelo possui muitas características do modelo de dados funcional DAPLEX (Shipman,1981). Se compõe de: Estruturas, Restrições e Operações.

Estruturas

Incluem Entidades e Tipos de Entidades; Tipos de Dados; Propriedades e Relacionamentos.

- Entidades e Tipos de Entidades: entidades ou objetos são usados para modelar objetos de aplicações de escritório, os quais podem ser estações de trabalho, gabinetes, departamentos, empregados, etc.

  Um tipo de entidade é uma especificação de uma classe de objetos que possuem as propriedades e as características da classe.

  Os tipos são relacionados via generalização-especialização, que vai permitir representar "hierarquia de tipos".

  Exemplos:
  ```
  DECLARE TYPE  Pessoa: ENTITY
  DECLARE TYPE  Estudante: Pessoa
  DECLARE TYPE  Pós-Graduando: Estudante
  ```

  Instâncias de um tipo particular herdam as propriedades dos seus supertipos.

  A hierarquia de tipo pode-se representar num grafo acíclico.

  O modelo permite justapor tipos através da construção DECLARE OVERLAP.

Exemplo: DECLARE OVERLAP Estudante, Empregado;

/* um objeto poderia ser simultaneamente instâncias do tipo Estudante e Empregado */

O modelo permite declarar "tipos construídos" (tipos que são subtipos de mais de um tipo) usando os operadores INTERSECTION, UNION, XOR e DIFFERENCE.

```
Exemplo:  DECLARE  TYPE  ConferencistaEstudante:
                    INTERSECTION  Pós-Graduando,
                                  Conferencista;
```

- Tipos de Dados: o modelo fornece tipos de dados simples ou formatados (INTEGER, REAL, STRING e BOOLEAN), assim como tipos de dados não formatados (TEXT, AUDIO e IMAGE).

  Podem-se definir tipos de dados simples como subtipos de outros tipos, eles são declarados primeiro no esquema da aplicação.

  Exemplo: DEFINE TYPE TipoData: INTEGER

- Propriedades e Relacionamentos: são modeladas por "funções" que retornam valores de dados e valores de tipo entidade respectivamente.

  As funções podem retornar quatro tipos de valor:

  i) "single-valued" retornando um valor só (SINGLE);

  ii) "set-valued" retornando um conjunto de valores não duplicados (SET);

  iii) "list-valued" retornando listas de valores (LIST) é análogo ao conceito "sequência" em ODA (Horak,1985);

  iv) "array-valued" retornando uma lista de valores que podem ser indexados (ARRAY) também em ODA.

  As funções declaradas dentro de uma definição de tipo não têm argumentos na sua declaração.

```
Exemplos:  DECLARE  Nome: SINGLE string [1:1];
           DECLARE  Nos.Telefone:  SET string [0:];
```

As funções multi-argumentos são requeridas para modelar relacionamentos que envolvem mais de um objeto. Argumentos também podem ser associados na declaração.

```
Exemplo:  DECLARE  Graduação        (Estudante,
                   Curso(Estudante)):SINGLE Integer [0:1];
```

O modelo fornece "funções derivadas" para definir novas propriedades baseadas em propriedades existentes. Usa-se a sentença DEFINE.

```
Exemplo:      DEFINE     EmpregadoPor:    SET     INVERSE
                                  PessoalDo(Departamento);
```

Usando a técnica de composição funcional é possível construir novas funções a partir de funções existentes.

Restrições

Os objetos ou entidades devem estar organizados em tipos e um objeto de um tipo dado pode ter só aqueles atributos especificados para esse tipo ou seus supertipos.

As propriedades de um objeto usadas para sua identificação são denominadas "keys", são especificadas na declaração de tipo usando a sentença KEY.

As funções podem retornar valores únicos ou conjuntos, listas ou "arrays" de valores.

A cláusula de cardinalidade permite limitar o número de valores que uma função pode retornar.

```
    Exemplo: DECLARE Nos.Telefone: SET String [0:];
/*  limita o número de telefones desde 0 até um limite não
    especificado */
```

A existência de um objeto pode depender da existência de outro.

```
    Exemplo:  DECLARE  TYPE   ObjetoInformação:    DEPENDENT
                               ObjetoEscritório;
```

Uma função dependente significa que um valor da função não pode existir independentemente do objeto no qual a função é

declarada.

Exemplo: na declaração do tipo Gabinete:
DECLARE GavetasDo: DEPENDENT ARRAY Gaveta [1:];

/* se um gabinete é eliminado, as suas gavetas também são eliminadas */

Operações

- Sentenças de Controle e de Operações gerais:

FOR: construção que se usa para manipular objetos particulares;

RETURN: retorna um número de valores de dados;

FOR EACH: usa-se para dar "loop" através de um conjunto, lista ou "array".

Se um objeto é membro de vários tipos e é desejável aplicar funções definidas para um tipo diferente, usa-se o operador AS.

Usa-se o operador SUCH THAT para selecionar alguns valores de uma função ou instâncias de um tipo.

Exemplo:

```
FOR EACH Empregado SUCH THAT Salário(Empregado) > 100
    RETURN (Nome(Empregado), Salário(Empregado));
```

Variáveis de manipulação podem ser associados explícitamente com valores usando as sentenças SINGLE..IS, SET .. IS, LIST .. IS e ARRAY .. IS

Exemplo: SINGLE gavetax IS GavetasDo(gabinetex) [n]
/* gavetax: variável de manipulação */

- Sentenças para atualizar objetos:

FOR A NEW: usa-se para criar objetos;

SINGLE .. IS A NEW: permite ligar uma variável de manipulação a um novo objeto.

Uma sentença de criação pode ser seguido de um bloco de sentenças para dar valores às funções do novo objeto.

DELETE: usa-se para remover um objeto;

MOVE: usa-se para mover um objeto da sua posição na hierarquia de tipo.

- Sentenças para atualizar funções:

LET: usa-se para dar um novo valor a uma função; INCLUDE: acrescenta valores a um conjunto ou lista, usando as cláusulas BEFORE ou AFTER novos valores podem ser inseridos na mitade de uma lista; EXCLUDE: remove valores.

Exemplos: LET Nos.Telefone (JJ) = ["5521704","5521708"];
INCLUDE Nos.Telefone (JJ) = "5521905";

- Operações de Recuperação: a recuperação de dados é implícita, cada aplicação de função é uma chamada ao banco de dados para recuperar alguma informação.

- Operações Extra:

Funções agregadas tais como: COUNT, AVERAGE, TOTAL, MAX e MIN; CONTAINS é uma função que busca uma expressão "string" num valor de texto, se a expressão existe a função retorna o valor TRUE do contrário o valor FALSE.

IV.7. MODELOS DE DADOS ORIENTADOS A OBJETOS

IV.7.1. MODELO DE DADOS DO SISTEMA "SON OF OZ"

O sistema "Son of Oz" proposto por Nierstrasz (Nierstrasz,1985) é orientado a objetos, que pode ser visto como um banco de dados, no qual a inteligência está associada com os ítens de dados antes do que com os programas que os manipulam.

Um objeto é uma entidade com ESTRUTURA e COMPORTAMENTO. Os objetos são grupados em CLASSES com a mesma especificação

genérica (exemplos: jornais, pessoas).

As instâncias de objetos se caracterisam por ter identificadores de objetos únicos (OIDs).

Cada classe de objeto tem uma ESPECIFICAÇÃO que descreve a estrutura e comportamento das instâncias da classe.

A única interface legítima à estrutura do objeto é através do seu comportamento.

Estrutura

É a porção de dados de um objeto; os dados consistem de outros objetos tais como objetos simples: inteiros, números de ponto flutuante, booleanas, caracteres, cadeias de caracteres, listas e "arrays". Estes objetos simples são armazenados em variáveis instanciadas. Os objetos que são parte da estrutura de um outro objeto são chamados objetos dependentes (relação pai e filho). Os objetos independentes não têm pais.

Os objetos podem-se comunicar com outros objetos, se estes objetos são conhecidos. Os objetos são conhecidos se eles estão relacionados diretamente (pai e filho) ou se um objeto possui o "OID" do outro.

No modelo "Oz" (Twaites,1984), (Mooney,1984) é especificada a classe dos objetos que se quer ter como conhecidos e algumas condições gatilho ("trigger") para limitar a seleção do objeto conhecido.

Comportamento

O comportamento de um objeto descreve as circunstâncias sob as quais um objeto pode mudar seu estado ou ocasionar que algum outro objeto mude seu estado. Um objeto muda seu estado quando é criado, eliminado ou atualizado.

O comportamento de um objeto pode ser especificado por

um conjunto de REGRAS.

Cada classe de objeto tem as regras alpha e omega para criar e eliminar uma instância respectivamente.

As "regras-top" são similares a sub-regras exceto que não são invocadas explícitamente e as suas ações são executadas quando as suas condições gatilho ("trigger") fazem-se verdadeiras.

As regras podem enviar e receber listas de objetos como valores de argumentos ou valores retornados, exceito as "regras-top".

As regras também podem ter um conjunto de variáveis temporárias para manipular argumentos, calcular valores de retorno.

As regras são invocadas especificando: um objeto para ser endereçado, o nome da regra, uma lista de objetos argumentos e uma lista de variáveis para os objetos retornados.

```
Exemplo: x.add(1)  /* equivalente a  x+1;  add:nome da
                                              regra */
```

Especialização de Objetos

Um objeto especializado herda a estrutura e comportamento da sua superclasse. Embora, um objeto especializado pode restringir as regras da sua superclasse adicionando condições gatilho ("trigger") e especializar as classes das variáveis instanciadas. Os objetos especializados não devem violar as restrições impostas pela superclasse.

Define-se um tipo de objeto (Woo,1985),(Twaites,1984) como:

```
⟨tipo de objeto⟩ : ⟨superclasse⟩ {
        lista de conhecidos
        declarações de variáveis instanciadas
        regra  ALPHA
        regra1
          .
          .
        regran
        regra  OMEGA
        }
```

O formato para uma regra é:

```
⟨nome da regra⟩  (argumentos) {
        lista de conhecidos
        declarações de variáveis temporárias
        precondições ou condições "trigger"
        ações
    }  (valores retornados)
```

Os objetos são especialmente úteis para programar aplicações de SIEs.

- Documentos: estruturados hierarquicamente podem ser representados neste modelo. Várias versões de um documento podem ser armazenadas como um objeto único, então diferentes versões podem ser chamadas como vistas separadas. Informação de "layout" pode estar associada com os vários componentes do documento.
- Correio: mensagens de correio podem ser codificadas como objetos. Uma mensagem pode consistir de alguma informação de cabeceira, e uma coleção de objetos a serem enviados por correio.

  Por outro lado, mensagens inteligentes ou "imessages" usando objetos, entram num diálogo com seu receptor quando o usuário tenta ler o correio. Este tipo de mensagem seria usado para distribuir recursos ou responsabilidades entre o conjunto de usuários, ou para capturar informação.
- Coleção de conhecimento: objetos denominados "knos" ("knowledge, aquisition, dissemination and manipulation

objects") migram através de um sistema ou rede de sistemas de objetos, coletando e processando informação. Cada "kno" é uma instância de uma ou mais classes.
Num ambiente orientado a objetos, o monitoramento de objetos existentes por um "kno" externo pode ser completamente transparente (Tsichritzis,1985), (Tsichritzis,1987).

IV.7.2. MODELO DE DADOS DO SISTEMA "OPAL"

O sistema "OPAL" proposto por Ahlsen (Ahlsen,1984) está baseado num conceito chamado "packet" que gera sistemas objeto os quais estão integramente construídos por estruturas de objetos armazenados num banco de dados. Estes objetos representam programas e dados, e estão integrados uniformemente dentro do banco de dados.

Um "packet" representa um tipo de objeto com uma estrutura e propriedades particulares.
Uma instância de "packet" é descrita por um tipo de "packet" (ou "metapacket"). O tipo de "packet" tem atributos que mantêm as descrições de suas instâncias. Os atributos de usuário para uma instância estão caracterisados pelas estruturas de dados; o usuário também define os seus procedimentos de inicialização, procedimentos de acesso, privilégios de acesso e definições de restrições.

Um identificador único (PID) é fornecido pelo sistema para cada instância de "packet" gerada. O procedimento de inicialização designa valores a atributos que podem ser derivados de outros "packets" ou ser obtidos do ambiente.

Tipos de Dados

Os tipos de dados básicos são: INTEGER, REAL, TEXT, TEXTFILE, DOLLAR, DATE, TIME. Além disso, os tipos PACKETREF, TUPLE e TABLE.

- Packetref: um "packetref" é uma referência a uma instância de "packet". As referências podem ser usadas pelo projetista da aplicação e o usuário final. Uma variável deste tipo pode estar ligada a um tipo de "packet" ou não.
- Tuple: uma tupla representa um conjunto ordenado de um número predefinido de escalares de vários tipos.
- Table: uma tabela é um conjunto ordenado de instâncias de tipo tupla. É usado para representar tabelas bi-dimensionais em "formulários" de escritório.

```
Ex.:  linhasordem: TABLE (partno:INTEGER, preço:DOLLAR,
                          quantidade:INTEGER, rowsum:DOLLAR);
```

São disponíveis dois métodos de acesso: um modo orientado ao conteúdo:

```
Ex.:  SELECTAB partno FROM linhasordem WHERE
                                    quantidade < 50;
```

e um modo de acesso orientado à posição:

```
Exemplo:  linhasordem [2, 2]
          linhasordem [*, partno]  etc.
```

Os tipos de dados definidos pelo usuário são construídos, usando os tipos básicos e os estruturando em tuplas e tabelas. Podem ser definidas restrições, procedimentos de inicialização e procedimentos de acesso para cada tipo de dados. Uma definição de tipo é representado como um "packet".

Atributos de "Packet"

Um atributo controlado pelo sistema denominado FOLIO contem as refêrencias a outros "packets". Os dados do

"folio" são um conjunto ordenado de "packetrefs". É usado para construir agregações de "packets" (estruturas de instâncias).

Operações de "folio" são procedimentos definidos sobre todos os tipos de "packet". Exemplos destas operações são: FIRST, LAST, NEXT, PREVIOUS, EMPTY, SORT e APPEND.

```
Ex.: smithord: PACKETREF;
     smithord <--- SELECT ordem WHERE fornecedor =
                                          "Smith";
     IF NOT smithord.EMPTY THEN
                .
                .
```

Os atributos de aplicação de um tipo de "packet" são declarados pelo projetista. Cada atributo pode estar relacionado a uma expressão de geração. Estas expressões são denominadas "Attribute Sources" e são de dois tipos: "External Source" e "Internal Source".

"External Source" é usado só no tempo de geração; "Internal Source" é local ao "packet" e será automaticamente avaliado quando um atributo fonte seja alterado.

```
Ex.: PACKET ordem;
       ATTRIBUTE linhasordem:
          TABLE    (partno:INTEGER,     preço:DOLLAR,
                   quantidade:INTEGER,  rowsum:DOLLAR);
                   total: DOLLAR;
       INTSOURCE BEGIN
          linhasordem[#i, rowsum] <--
                        linhasordem [#i, preço] *
                        linhasordem  [#i, quantidade];
          total <--  SUM (linhasordem [*,rowsum]);
       END INTSOURCE;
     ENDPACKET ordem;
```

É possível ligar um procedimento de restrição a um atributo, o qual é evaluado cada vez que o atributo é escrito (ou lido) por um outro "packet".

"Templates" de "Packet"

O "layout" do "packet" é definido num "template" (molde). O "template" é um mecanismo de transformação que

permite a diferentes usuários ver os "packets" de diferentes maneiras. Um tipo de "template" é um tipo de "packet" especial, o qual é refinado pelo usuário (através de herança de propriedade) e é armazenado no banco de dados. Os "templates" também servem para transmitir os mecanismos de segurança e autorização no sistema OPAL.

Herança de Propriedade

O mecanismo de herança está baseado na "herança múltipla". A sintaxe "a,b PACKET c" significa que c herda as propriedades de a e b. A definição de atributo mais interno sempre anula uma definição de atributo de superclasse. Para atributos com a mesma denominação nas superclasses, têm-se uma ordem de herança "default" definida pela ordem prefixo (no exemplo: a, b). As regras de herança também abrangem procedimentos.

Versões de "Packet"

- Versões de Tipo de "Packet": as instâncias de diferentes versões de "packet" são possíveis de distinguir, quando seja necessário, por meio de um número de versão (ou um nome de versão dado pelo projetista).
- Versões de Instância de "Packet": o mecanismo é transparente e a versão de instância "default" é a última versão.

IV.7.3. MODELO DE DADOS O-O DO SGBD "IRIS"

O modelo de dados orientado-a-objetos (O-O) IRIS suporta abstrações estruturais de alto nível tais como classificação, generalização/especialização e agregação,

assim como abstrações no comportamento (Fishman,1987).

O modelo baseia-se em três construtores: Objetos, Tipos e Operações (Funções). Suporta propriedades genéricas e de herança, restrições, dados complexos ou não normalizados, operações definidas pelo usuário, controle de versão, inferência e tipos de dados estendidos.

As raizes do modelo baseiam-se no modelo DAPLEX (Shipman,1981) e na linguagem TAXIS (Mylopoulos,1980).

Objetos

Os objetos representam entidades e conceitos do domínio da aplicação. Possuem existência própria no B.D. e podem ser referenciados independentemente dos valores de seus atributos (integridade referencial).

Os objetos são classificados por tipo: objetos compartilhando propriedades comuns pertencem ao mesmo tipo. Podem ser: objetos literais que incluem inteiros, reais, booleanos e cadeias de caracteres, texto, audio, os quais são diretamente representáveis; ou objetos não-literais que são representados internamente através de identificadores gerados pelo sistema ("surrogates"). Ex.: pessoa, departamento.

Os objetos não-literais podem ser referenciados a partir de um programa de aplicação pelos valores de suas propriedades ou pelos seus relacionamentos com outros objetos.

Tipos e Hierarquia de Tipos

Os tipos são coleções nomeadas de objetos que compartilham propriedades comuns. São organizados numa estrutura de tipo que suporta generalização e especialização.

Assim, um tipo pode ser subtipo de um outro denominado o seu

supertipo, herdando as propriedades dele.

A estrutura de tipo IRIS é um grafo acíclico dirigido. Um tipo pode possuir múltiplos subtipos e múltiplos supertipos. Os tipos podem ser declarados como "disjuntos" ou "sobrepostos".

As propriedades definidas sobre diferentes tipos podem ser genéricas; isto é, podem ter nomes idênticos embora suas definições sejam diferentes.

Operações e Regras

As operações (ou funções) retornam uma coleção de resultados (ou o valor nulo). São definidas sobre os tipos e são aplicáveis sobre as instâncias dos tipos.

O modelo suporta operações definidas pelo usuário (Derrett,1985).

As propriedades (atributos) e relacionamentos entre objetos são expressados em termos de funções (possivelmente "multivalorados").

Uma função pode expressar propriedades de vários objetos, assim como retornar resultados múltiplos e complexos.

Exemplo: Designação: Professor ---> Curso x Semestre

Pode-se especificar restrições de cardinalidade sobre parametros e resultados de operações (Lyngbaek,1987).

O valor resultado pode ser especificado como requerido (REQUIRED), único (UNIQUE), etc.

O modelo permite definir novos tipos de dados com as suas operações associadas, os quais só podem ser definidos usando os tipos e operações fornecidos pelo sistema.

Consulta

O banco de dados pode ser consultado através do comando FIND, o mesmo que retorna todas as combinações de objetos que satisfazem algum predicado.

Os predicados podem conter chamadas a funções, constantes, variáveis, operadores de comparação e operadores lógicos.

```
Ex.: FIND g/empregado WHERE FORSOME d/departamento
     g = Gerente_Do (d) AND tamanho_departamento (d) > 20
       /* recupera todos os gerentes de Departamentos com
          mais de 20 empregados */
```

Operações Derivadas

Uma função pode ser derivada de outras funções (simples ou derivadas).

```
Ex.: DEFINE Supervisor_Do : empregado ---> empregado
     DERIVE Supervisor_Do (e) = FIND s/empregado WHERE
            s = Gerente_Do (Departamento_Do (e))
```

Regras

São funções e o modelo só suporta regras conjuntivas, não as recursivas.

```
Ex.: DEFINE avô (p/pessoa) = FIND a/pessoa WHERE
                                  a = pai (pai (p) )
```

Atualização

O modelo fornece os seguintes comandos:

SET: dá um novo valor a uma função;

ADD: acrescenta valores a uma função com múltiplos valores;

REMOVE: remove valores de uma função com múltiplos valores ou com um único valor.

IV.7.4. MODELO DE DADOS DE ESCRITÓRIO DE GIBBS

O modelo de dados de escritório proposto por Gibbs (Gibbs,1983), (Gibbs,1985), representa entidades do mundo real usando objetos classificados em tipos.

A estrutura do objeto consiste de uma hierarquia de propriedades e associações constituintes. Restrições adicionais são expressas proceduralmente usando gatilhos ("triggers"), definições de domínios. Os tipos de dados primitivos incluem: "integer", "real", "boolean", "char", "string"; e os novos tipos: "audio", "image", "text" e "digital". Os tipos "template" especificam a apresentação de um tipo objeto particular.

Tipo

É a especificação estrutural de uma classe de objetos. A especificação de um tipo identifica a estrutura de propriedades e de constituintes das instâncias do tipo.

Podem ser declaradas as propriedades simples (cujos valores são ítens de dados) e as propriedades compostas (cujos valores são objetos "características"). Assim, as propriedades são organizadas hierarquicamente.

Um objeto pode-se decompor em outros objetos denominados constituintes.

Pode ser especificada a restrição de unicidade (uma chave generalizada) de propriedades simples e constituintes.

As propriedades e constituintes podem ser monovalorados ou multivalorados.

Os tipos podem ser relacionados via especialização e generalização (supertipo/subtipo).

Exemplo: Empregado IS-A Pessoa;

Os relacionamentos "IS-A" formam um digrafo acíclico. As instâncias de um subtipo herdam as estruturas das propriedades de seus supertipos.

O modelo permite representar relacionamentos como E-R numa seção que define mapeamentos.

Nomes de mapeamentos, nomes de constituintes e nomes de propriedades podem-se concatenar para formar uma "cadeia" que é parecida com a composição funcional em DAPLEX (Shipman,1981).

As cadeias são evaluadas de esquerda a direita.

Exemplo: João: GerenciadoPor: DoDept.Título
/* refere-se ao título do departamento do gerente de João */

Operações

DEFINE: adiciona novas definições de tipo objeto, definições de tipo "template" e restrições definidas explicitamente.

- Operações de manipulação:

CREATE: cria uma nova instância de objeto; DESTROY: remove uma instância existente; COPY: cria um objeto com os mesmos valores de propriedades de um objeto existente; ASSERT: designa um valor a uma propriedade ou constituinte; REMOVE: remove valores de propriedades ou constituintes.

ADMIT: acrescenta um objeto a um tipo mais especializado; PROHIBIT: elimina-lo de um tipo.

- Operações de Consulta:

SET: é usada para selecionar um conjunto de objetos;

FOR: permite iteração sobre um conjunto;

OBJECT: é usada para selecionar um único objeto.

- Operações associadas com Transações:

TBEGIN, TEND, ABORT. A operação TBEGIN inicia uma transação que pode ser abortada pela operação ABORT ou cometida pela operação TEND.

EXPORT: preserva a ligação entre um identificador e um objeto para transações subseguintes.

O modelo suporta outras restrições usando as seguintes construções: Domínios, Gatilhos ("triggers") e Tipos "Template".

- Domínios: especificam tipos de dados restringidos ou mais refinados. A linguagem de definição de domínios permite especificar a estrutura de dados usada para armazenar elementos do tipo "dado" (representação interna) e os seus formatos permisíveis (representações externas); também como as operações para os elementos. Ex: pode-se definir o domínio DATA.
- Gatilhos ("triggers"): é um mecanismo de especificação de restrições que opera a nível de tipos objetos, é um grupo de operações a serem executadas quando é feita uma mudança ao banco de dados.

  Os gatilhos consistem de 4 componentes:

  1. padrão (pattern): identifica a operação que ativa o gatilho e declara restrições sobre os tipos de objetos que aparecem na operação;
  2. condição (condition): contem uma condição que deve ser satisfeita antes que a operação que ativa o gatilho seja executada, pode-se testar pre-condições e pós-condições;
  3. erro (error): contem um "string" (mensagem de erro) que é exibido quando a condição falha; e
  4. ação (action): lista de operações a serem executadas quando a condição é satisfeita.

- Tipo "template": é uma especificação da forma como um objeto ou conjunto de objetos vão ser apresentados num meio particular.

Há 3 categorias de "templates": i) de audio, ii) de imagem (gráfico) e iii) de texto.

Operações que envolvem instâncias de "template" são:

TEMPLATE: declara um identificador que é usado para referir-se a um "template" específico;

EMBED: executa o mapeamento de uma estrutura de objeto dentro de uma estrutura de "template";

PRESENT: emite o valor do "template";

ERASE: remove um "template" apresentado no vídeo.

CAPÍTULO V

PROPOSTA DE MODELO DE DADOS PARA SISTEMAS DE INFORMAÇÃO PARA ESCRITÓRIOS

V.1. INTRODUÇÃO

Este capítulo contém:

- Requisitos para um Modelo de Dados para SIEs.
- Definição de Modelo de Dados.
- Modelos de Dados Tradicionais VS Modelos de Dados Orientados a Objetos (O-O).
- Modelo de Dados O-O proposto para SIEs.
- Análise do Modelo de Dados Proposto.

V.2. REQUISITOS PARA UM MODELO DE DADOS PARA "SIE"S

Um escritório é um ambiente complexo e coloca demandas especiais sobre o modelo de dados a ser usado.
Os requisitos apresentados a seguir foram re-estabelecidos em função das características próprias dos SIEs já descritas em II.9.

1. O modelo de dados deve suportar o modelo conceitual dos usuários de SIEs.
2. Os dados de escritório exibem uma estrutura diversa e complexa:
   . Dados não formatados tais como texto, imagem, voz;
   . Dados de tempo necessários para determinar o tempo de vida dos documentos ou das operações individuais, ou para se especificar a duração das atividades e as restrições de tempo;

O usuário percebe os dados de escritório como "unidades de informação" ou "objetos de escritório" de vários tipos (como por exemplo o documento de tipo relatório), os quais devem ser acessados e manipulados como uma unidade lógica de processamento;

. Muitos objetos de escritório são construídos de outros objetos;

. Os objetos de escritório são usados frequentemente em muitas diferentes formas;

. O modelo deve ser capaz de suportar as estruturas lógicas e "layout" descritas no padrão ODA - "Office Document Architecture" (Horak,1985);

3. O modelo deve suportar um ambiente integrado de escritórios, onde os dados produzidos por uma aplicação sejam disponiveis para outras aplicações. Isto implica a integração de muitos tipos de informação.

4. O modelo deve suportar a evolução do escritório: as restrições definidas sobre os elementos de escritório mudam frequentemente no seu conteúdo e estrutura básica, as atividades também podem se modificar no tempo, para refletir novos conhecimentos ou condições. Deve-se permitir o desenvolvimento de novos tipos de objetos.

5. O modelo deve ser capaz de descrever relacionamentos existentes entre objetos, dado que existe um grande número de elementos de escritório relacionados através de várias conexões.

6. Muitos procedimentos de escritório envolvem manipulação de textos que podem durar horas ou dias. A maioria das atividades de escritório somente são executadas uma vez que algumas condições tenham sido satisfeitas;

7. As operações devem ser capazes de capturar o comportamento de atividades de escritório comuns tais como: criação, arquivamento, recuperação, atualização e distribuição de objetos de escritório;
8. O modelo deve fornecer um mecanismo que permita filtrar grande quantidade de dados, de modo a fornecer aos usuários somente informação específica.

V.3. DEFINIÇÃO DE MODELO DE DADOS

Segundo Codd (Codd,1981) um Modelo de Dados é uma combinação de três componentes:

1. Uma coleção de tipos de estrutura de dados;
2. Uma coleção de operadores ou regras de inferência, que são aplicados a quaisquer das instâncias válidas dos tipos de dados listados em (1) para recuperar ou derivar dados;
3. Uma coleção de regras de integridade geral, as quais implicitamente ou explicitamente definem o conjunto de estados de banco de dados consistente ou mudanças de estado ou ambos - estas regras são expressas algumas vezes como regras de inserção-atualização-eliminação.

Segundo Brodie (Brodie,1984) um Modelo de Dados é uma coleção de conceitos matematicamente bem definidos que ajudam a considerar e expressar as propriedades estáticas e dinâmicas de aplicações de dados intensivos.

Os conceitos constituintes de um modelo de dados estão dentro destas três categorias:

- Propriedades estáticas tais como objetos, propriedades de objetos (ou atributos), e relacionamentos entre objetos (ou seja uma classe particular de propriedades de objetos).
- Propriedades dinâmicas tais como operações sobre objetos, propriedades de operações, e relacionamentos entre operações (para formar transações).
- Regras de Integridade sobre objetos (i.é., estados do banco de dados) e operações (i.é., transições de estado).

Os Modelos de Dados fornecem primitivas para definir, manipular e consultar banco de dados. As primitivas podem ser usadas como uma linguagem de baixo nível introduzidas como sentenças de chamada numa linguagem hospedeira (como Cobol, Pascal, Lisp) ou projetada diretamente numa linguagem de programação de alto nível (tais como Daplex, Taxis).

Segundo Gibbs (Gibbs,1985) um Modelo de Dados é visto como uma linguagem de especificação para representações do mundo real. Isto é, dado um problema num domínio da aplicação, usamos um modelo de dados para especificar uma representação daquela porção do mundo real relevante ao problema.

A representação pode conter tanto um aspecto estrutural ou estático como um aspecto operacional, dinâmico.

V.4. MODELOS DE DADOS TRADICIONAIS VERSUS MODELOS DE DADOS ORIENTADO A OBJETOS (O-O)

Modelos de Dados Tradicionais

São os Modelos de Redes, Hierárquico e Relacional.

Os modelos Hierárquicos e de Redes representam objetos em segmentos ou registros que estão organizados, usando relacionamentos binários 1:N, como nodos em árvores e redes respectivamente. Fornecem somente operações primitivas e facilidades primitivas para percorrer um-registro-por-vez. Suas linguagens são mais representacionais (os usuários devem tratar com mais características de armazenamento e implementação).

O modelo relacional foi baseado no conceito matemático de uma RELAÇÃO, que é um conjunto de ênuplas. A estrutura de uma relação consiste de um número de ATRIBUTOS sobre DOMÍNIOS básicos. Uma tupla pode ser usada para representar tanto objetos, quanto relacionamentos N:M.

O cálculo e álgebra relacional fornecem uma facilidade de acesso para consultas orientadas a conjuntos e que também podem ser usadas para estabelecer restrições como ASSERTIVAS. (Brodie,1984), (Gibbs,1985).

Modelos de Dados Orientados a Objetos (O-O)

Os modelos de dados orientados a objetos (O-O) têm sido influenciados pela tecnologia de BD, Inteligência Artificial (AI) e Linguagens de programação O-O.

Segundo Bancilhon (Bancilhon,1988) as características inerentes aos modelos O-O são:

1. **Encapsulamento**: Os objetos são modelados pela estrutura de dados e pelas operações associadas. Um objeto tem uma parte interface e uma parte implementação. A parte interface é a especificação do conjunto de operações que podem ser executadas sobre o objeto. É a única parte visível do objeto. A parte implementação tem uma parte de dados e uma parte de operação.

   Uma vez definida a interface ao objeto, nenhuma operação diferente das especificadas nesta interface pode ser executada. Isto é certo tanto para atualização e consulta.

2. **Identidade de Objetos**: significa que um objeto tem uma existência que é independente do seu valor. Assim, dois objetos podem ser idênticos (são o mesmo objeto) ou iguais (têm o mesmo valor).

   Num modelo baseado em identidade, dois objetos podem compartilhar um componente. Assim, a representação gráfica de um objeto complexo é um DAG (Grafo Acíclico Direcionado). A atualização de um objeto é visto automaticamente por todos os objetos que referem-se a ele (isto é denominado Transparência Referencial).

3. **Tipos e Classes**: Um TIPO descreve um conjunto de objetos com as mesmas características. Corresponde-se à noção de um Tipo Abstrato de Dados (TAD). Em linguagens de programação, os tipos são usados em tempo de compilação para verificar a exatidão dos programas.

   A especificação de uma CLASSE é a mesma que de tipo, porém as classes são usadas para criar e manipular objetos de um mesmo tipo e não para verificar a exatidão

dos programas.

As Classes estão organizadas em uma hierarquia.

4. Herança: permite especializar objetos, fazendo-os compartilhar estrutura e comportamento relacionados a suas partes comuns.

   Um objeto que herda atributos e operações de outros objetos, tem por sua vez seus próprios atributos e operações.

   É uma ferramenta de modelagem poderosa porque dá uma descrição concisa e precisa do mundo, e pela reusabilidade através do compartilhamento de estrutura e comportamento entre objetos.

5. Polimorfismo Operacional: Para os casos em que requere-se utilizar o mesmo nome para operações diferentes usa-se o mecanismo de sobreposição ("overriding") pelo qual define-se a operação (ou método) no nível de tipo mais geral e redefine-se o corpo da operação para cada um dos tipos requeridos. Isto resulta num nome único denotando procedimentos diferentes ("overloading").

   Para um novo tipo inserido no sistema, o procedimento também é reusável. Para oferecer esta nova funcionalidade de sobreposição de operações, os nomes de operações só podem ser resolvidos em tempo de execução (isto é denominado "late binding").

Alguns modelos orientados a objetos têm, ainda, a seguinte característica:

Atributos Dinâmicos: atributos podem ser manualmente atualizáveis ou dinamicamente derivados de outros

atributos do objeto ou de outros objetos. Regras de avaliação de atributos podem ser especificadas junto com a descrição do objeto e podem ser ativadas por gatilhos ou mecanismos de controle da rede de dependências entre os atributos. (Hudson,1986a).

Graus de Liberdade são dedicados ao projetista para outras características tais como:

- O modelo computacional (uso de linguagens funcionais, linguagens lógicas, linguagens imperativos, etc.);
- Os construtores de objetos que o modelo usa estão abertos e a maioria de modelos diferem;
- O grau de uniformidade pode variar. Sob uma consideração uniforme: objetos, tipos e métodos são objetos; a manipulação de tipo e método é feita através de mensagens. De outro modo, tipos e métodos não são objetos e são manipulados por comandos especiais. (Bancilhon,1988).

Relação de Modelos O-O com Modelos de Dados Semânticos

Os Modelos de Dados Semânticos aparecem como uma integração das tecnologias de BD e Inteligência Artificial.

A noção de objetos complexos e hierarquia de tipos é comum tanto de Modelos Semânticos como de Modelos de Dados O-O. Embora os modelos semânticos ignorem as noções de encapsulação e polimorfismo operacional.

Os Modelos Semânticos estão mais interessados nas questões de modelagem conceitual e consultas do que em aspectos de sistema tais como a execução de programas de

aplicação. (Zaniolo,1986), (Bancilhon,1988).

Desvantagens de Modelos de Dados Tradicionais

Os Modelos de Redes e Hierárquico obrigam ao usuário a pensar em termos de conceitos relacionados ao computador (máquina) antes que na estrutura natural dos dados.
Estes modelos não têm um tratamento orientado a objetos apropriado para aplicações de escritório porque estão orientados para registros; as operações e suas restrições são centradas no conceito de registro e suas conexões.

Ainda, o modelo relacional está mais interessado na organização de dados para propósito de processamento, do que na estrutura natural dos dados.
O modelo relacional não tem a capacidade de representar a estrutura natural dos objetos dado que não distingue atributos de objetos de relacionamentos entre objetos, já que ambos são modelados pela mesma estrutura.
Somente permite a composição de objetos usando referências baseadas em conteúdo (ex: chaves); portanto um objeto não tem uma existência independente do seu valor e se tem problemas de "integridade referencial".

Os sistemas relacionais estão bem ajustados para o modo de consulta "ad hoc", porém não para o desenvolvimento da aplicação (o problema de "impedance mismatch"). Nenhuma das características dos SGBDs relacionais podem ser usadas para gerenciar os programas. Não têm ferramentas que facilitem a programação como herança e polimorfismo operacional.

Desvantagens de Modelos de Dados Orientados a Objetos (O-O)

Uma desvantagem de Modelos de Dados Orientados a Objetos (O-O) é a falta de um modelo comum porque não há um consenso claro sobre definições de conceitos, construtores de tipos, extensibilidade de tipos, generalização ou especialização de operações, polimorfismo de tipos, etc.

Outra desvantagem é a falta de uma teoria básica (fundação formal) para definir: as semânticas de tipos, identidade de objeto. E pelo enorme esforço de implementação em andamento. (Bancilhon,1988).

V.5. MODELO DE DADOS O-O PROPOSTO PARA "SIE"s

O modelo proposto a seguir fornece um conjunto de construções de modelagem que endereçam as questões básicas da gerência de informação para SIEs.

O modelo suporta objetos simples e complexos, relacionamentos entre objetos, restrições, atributos e constituintes de objetos, tipos de dados, construtores de tipos; regras de derivação que descrevem o comportamento do objeto; um mecanismo de "Molde" para modelar estruturas "layout" (físicas) de objetos; Operações sobre objetos, classes, versões; Operações de Recuperação ou Consulta; Operações de moldes e operações para gerenciar transações curtas e longas.

Objetos

Os objetos representam entidades e conceitos do domínio das aplicações de escritório, como por exemplo: objetos de escritório tais como: estações de trabalho, gabinetes,

arquivos, documentos, etc., e objetos de interesse na empresa tais como: departamentos, empregados, clientes.
Os objetos possuem existência própria no BD (identidade de objeto) e podem ser referenciados independentemente dos valores de seus atributos (possuem integridade referencial).

Um objeto pode ser parte ou componente de outro objeto (relacionamento "is-part-of") e a sua existência pode depender da existência do seu objeto "pai" (objeto DEPENDENTE) ou não (objeto INDEPENDENTE não tem "pai").
Um objeto pode ser componente de mais de um objeto (objeto compartilhado) como por exemplo: um capítulo pode ser parte de dois livros.

Tipos de Objetos, Hierarquia e Herança

Um TIPO é a especificação estrutural e comportamental de uma classe de objetos que cumprem as propriedades e características dessa classe.

Um objeto pode ser instância de mais de um tipo. Por exemplo um objeto de tipo Empregado também poderia ser do tipo Estudante.

A ocorrência de objetos RECURSIVOS é permitida, direta ou indiretamente, usando o próprio tipo de objeto dentro da sua especificação estrutural.

Os tipos são organizados numa estrutura de tipo que suporta generalização/especialização. Deste modo, um tipo pode ser subtipo de um outro tipo chamado seu supertipo (este relacionamento entre tipos é denominado "is-a"); herdando a estrutura e comportamento do supertipo.

Exemplo: Empregado: Pessoa

/* Empregado é-uma Pessoa, o supertipo do tipo Empregado é o tipo Pessoa */

A Hierarquia de Tipos pode ser representada por um DAG (grafo acíclico dirigido), isto é, suporta múltiplos supertipos.

Portanto, um tipo pode possuir múltiplos subtipos e múltiplos supertipos, suportando herança múltipla.

Exemplo: Documento: ObjetoSimples, ObjetoMóvel

/* Os supertipos do tipo Documento são os tipos Objeto Simples e Objeto Móvel */

Se um tipo herda propriedades de seus supertipos com os mesmos nomes daqueles definidos no tipo, o conflito é resolvido dando precedência às definições dentro do tipo sobre os de seus supertipos.

Se um tipo herda nomes de propriedades iguais de seus supertipos, o conflito é resolvido elegindo a definição do primeiro supertipo na lista de supertipos diretos definida para o tipo.

O grafo de tipos (supertipos e subtipos) pode evoluir dinamicamente, através da criação e remoção de tipos.

Os objetos especializados não devem violar as restrições impostas pelos supertipos.

As propriedades definidas sobre diferentes tipos podem ser genéricas; isto é podem ter nomes identicos embora suas definições sejam diferentes ("overriding").

O tipo "OBJETO" é o supertipo de todos os tipos.

Classes

A noção de Classe é similar àquela usada em OMS (Zdonik,1984). Uma classe é uma coleção de objetos persistentes (em contraste com objetos temporários das linguagens de programação O-O, eles podem ser invocados de

uma sessão a outra).

Cada tipo tem uma classe associada que coleta todas as suas instâncias. Um tipo é uma descrição e uma classe é uma coleção.

Se um objeto é membro de um classe, também é automaticamente um membro de todas as suas superclasses
Um objeto específico pode ser membro de um número arbitrário de classes.

As subclasses de uma classe podem ser disjuntas ou sobrepostas. Duas classes disjuntas não podem ser superclasses em comum de uma outra classe. Classes podem ser declaradas como sobrepostas ou disjuntas; se não se especifica assume-se que podem ser sobrepostas.

Exemplo: DECLARAR SOBREPOSTOS Empregado, Cliente

DECLARAR DISJUNTOS Empregado, Fornecedor

A Classe "OBJETO" corresponde a todo o banco de dados dado que contem todos os objetos conhecidos no sistema.

Classes Construídas

São definidas em termos de classes definidas previamente. Para que um objeto seja membro de uma classe construída, deve ser primeiro membro de alguma outra classe. O modelo permite declarar classes construídas usando operadores de conjuntos: INTERSEÇÃO, UNIÃO e DIFERENÇA.
Para assegurar operações de conjunto compatíveis, os operandos devem ser subclasses de uma classe (seja diretamente ou através de uma série de relacionamentos de subclasse).

Exemplo: CLASSE AnalistaProgramador: INTERSEÇÃO

Analista, Programador

Classes subconjuntos são coleções diferenciadas de objetos que têm significado especial para aplicações específicas. Os membros destas classes não têm estrutura e comportamento próprios.

Exemplo: CLASSE EmpregadoObservado: SUBCONJUNTO DE Empregado

Tipos de Dados

Os valores de dados estão organizados em tipos:

- Básicos ou Primitivos: tipos de dados simples ou formatados tais como: INTEIRO,REAL,STRING (cadeia de caracteres),BOOLEANA,TEMPO,DATA; e tipos de dados não formatados tais como TEXTO,VOZ e IMAGEM.
- Derivados ou Extensíveis: são tipos definidos pelo usuário, cuja definição encapsula as funções usadas para reconhecer elementos do tipo, transformar representações externas em internas, e as operações sobre eles.

Construtores de Tipos

São mecanismos de estruturação que permitem manipular coleções de valores.

CONJ ("SET"): manipula uma coleção ou conjunto de valores não duplicados.

Exemplo: Pessoal: CONJ Empregado;
NosTelefone: CONJ string(12);

LISTA: manipula uma coleção ordenada de valores (permite-se duplição). A idéia de Lista é analoga ao conceito de Seqüência em ODA (Horak,1985)

Exemplo: ConteúdoArquivo: LISTA ObjetoInformação;

VETOR: manipula uma lista de valores que podem ser indexados. É análogo ao conceito de "Array" em ODA (Horak,1985)

Exemplo: GavetasGabinete: VETOR DEPENDENTE Gaveta;

Seções de Definição do Tipo Objeto:

As seções de Atributos, Constituintes e Mapeamentos tomam as idéias do Modelo de Dados para SIEs proposto por Gibbs (Gibbs,1983) e as seções de Restrições e Regras tomam as idéias do Modelo proposto por Hudson (Hudson,1986b) para gerar sistemas de escritório.

Atributos: Cada objeto tem um grupo de valores de dados associados denominados atributos. O conjunto de tipos de atributos associados a um objeto é determinado pelo tipo de objeto.
Os tipos de dados dos atributos devem ser especificados. Os atributos podem ser intrínsecos e derivados, porém somente os atributos intrínsecos podem ser atualizados.

A cláusula ÚNICO serve para definir chaves que identificam unicamente ao objeto em atributos e constituintes.

A cláusula REQUERIDO significa que deve existir algum valor (restrição de cardinalidade mínima).

A cláusula ORDENADO [ASC/DESC] ordena em forma ascendente ou descendente, os valores de um atributo multivalorado.

Exemplo: CLASSE Departamento: ObjetoEmpresa

```
ATRIBUTOS:
        Nome: string(30) REQUERIDO;
        Endereço: String(35) REQUERIDO;
        NosTelefone: CONJ string(12);
        (Nome) ÚNICO;
```

Constituintes: Objetos constituintes podem ser objetos componentes (relacionamento "is part-of") e/ou outros objetos relacionados com o tipo, incluindo objetos derivados.

Os tipos dos objetos constituintes devem ser especificados.

Exemplo: CLASSE Departamento: Objeto Empresa

```
.
.

CONSTITUINTES:
        Gerente: Empregado;
        Pessoal: CONJ Empregado;
        (Gerente,Pessoal) ÚNICO;
```

A cláusula DEPENDENTE serve para indicar que um objeto componente não pode existir independentemente do objeto daquele tipo. Assim, se um objeto é removido, se executará automaticamente a remoção recursiva de todos os objetos referenciados como dependentes do objeto.

A cláusula ORDENADO [ASC/DESC] (atributo) ordena um objeto constituinte segundo os valores do atributo especificado em forma ascendente ou descendente.

Exemplo: CLASSE Arquivo: ObjetoOrganização

```
.
.

CONSTITUINTES:

     ConteúdoArquivo: LISTA Documento
          ORDENADO ASC Documento.DataCriação;
```

Mapeamentos: Esta seção define mapeamentos explícitos entre os objetos constituintes não derivados do tipo, os quais podem ser relacionamentos binários 1:1 e 1:M. O "argumento" do mapeamento aparece à direita do nome do mapeamento.

```
Exemplo:    CLASSE  Departamento: ObjetoEmpresa
                    .
                    .
            MAPEAMENTOS:
                  GerenteDo: Pessoal ---> Gerente;
                  DeptDo: Gerente ---> Departamento;
                  PessoalDo: Gerente ---> Pessoal;
```

Assim, por exemplo: o mapeamento "GerenteDo" quando é aplicado ao Pessoal de um departamento avalia o valor do Gerente daquele Departamento.

Restrições: As restrições explícitas são usadas para manter a noção de consistência além das expressas tipicamente usando o sistema de tipo e das cláusulas já definidas.
Usando predicado de restrições é possível construir um mecanismo poderoso de teste-de-restrições dentro do modelo. Deste modo, é possível para os objetos ativamente testar condições de erro quando possam ser violadas.

As restrições são condições booleanas (ASSERTIVAS) que referem-se aos componentes do tipo. As restrições são verificadas depois de alguma atualização, se uma restrição não é satisfeita, então o acesso é abortado e o objeto é armazenado a seu estado original.
Um subtipo herda as restrições do seu supertipo e novas restrições podem ser somadas.

```
Exemplo:  RESTRIÇÕES:
                  Salário > 1000  OR  Salário = NULO;
                  Quantidade  <=  Inventario.Quantidade;
```

Regras:

Regras de Derivação descrevem o comportamento do objeto na forma de atributos e objetos derivados, vão permitir que eles sejam definidos em termos de outros valores da mesma instância de um objeto e de valores de instâncias de objetos relacionados. Assim, estas regras (funções) descrevem como os objetos podem responder ao seu ambiente.

As regras de derivação são calculadas sobre demanda desde o estado do b.d. em curso. Estas funções são construídas usando operações lógicas, aritméticas, de iteração e de recuperação ou seleção.

```
Exemplo:  CLASSE   Ordem : Formulário
                ATRIBUTOS:
                      NoOrdem: Inteiro;
                      Total: Inteiro;
                CONSTITUENTES:
                      LinhasOrdem: CONJ DEPENDENTE
                                              LinhaOrdem;
                MAPEAMENTOS:
                      OrdemDe: LinhasOrdem ---> Ordem;
                REGRAS:
                     Total:=  SUM(LinhasOrdem.Subtotal);

          CLASSE   LinhaOrdem
                ATRIBUTOS:
                      NoParte: Inteiro;
                      Quantidade: Inteiro;
                      Preço: Inteiro;
                      Subtotal: Inteiro;
                      Porcentagem: Real;
                REGRAS:
                    Subtotal:= Preço * Quantidade;
                    Porcentagem:=  if  Subtotal  <  2000
                              then  Subtotal*0.05
                              else  Subtotal*0.07;
```

Versões de Objetos

A construção de um documento passa por sucessivas revisões de desenvolvimento. Uma revisão que é liberada para uso poderia ser chamada VERSÃO. As versões sofrem

modificações por erros sintáticos ou por modificações estratégicas (um documento pode ter mais de uma versão sendo trabalhada ao mesmo tempo). Este último caso pode-se chamar de Versões ALTERNATIVAS. As versões alternativas são descrições válidas simultaneamente.

O modelo propõe um mecanismo de versões de objetos que usa algumas das noções de versões para OMS (Zdonik,1984) onde objetos que pertencem à mesma classe podem ter diferentes números de versões. As versões são organizadas através de relacionamentos de derivação. Uma derivação linear é baseada na data de criação, portanto uma única versão é derivada da última existente. Em uma derivação hierárquica (em árvore) quaisquer número de novas versões podem ser derivadas de uma versão existente em qualquer tempo.

Cada versão pertence a somente um objeto (objeto genérico) e, portanto versões não são compartilhadas entre objetos.

As versões envolvem-se num relacionamento implícito com outras versões do mesmo objeto. Este relacionamento define um "grafo de versões" sobre o conjunto de versões do mesmo objeto.

As versões são numeradas na sequência de criação, que é o critério de ordenação. Um grafo de versões pode ser: Linear e Árvore.

Linear - Cada versão tem no máximo uma sucessora e uma predecessora. Corresponde a versões em série;

Árvore - Cada versão tem um número arbitrário de sucessoras e somente uma predecessora. Corresponde a versões alternativas.

Os objetos de uma classe podem ser declarados versionavéis.

Exemplo: CLASSE Empregado

.

.

VERSÕES: Linear;

Operações sobre Objetos

NOVO: cria uma nova instância de uma classe de objetos e o liga a um identificador. Automaticamente gera-se novas instâncias das superclasses da classe especificada.

Tem a forma:

NOVO Nome-da-Classe É identificador-objeto;

Exemplo: NOVO Empregado É Tom;

ELIMINAR: remove objetos de uma classe e de suas subclasses se eles não são usados como componentes de outros objetos. Tem a forma:

ELIMINAR Expressão1, ..., ExpressãoN;

"Expressão i" avalia um objeto ou uma sequência de objetos.

Exemplo: ELIMINAR Tom;

COPIAR: cria um ou mais objetos com os mesmos valores de um objeto já existente. Tem a forma:

COPIAR identificador-objeto1,...,identificador-objetoN DE identificador-objeto-fonte;

Exemplo: COPIAR y DE x;

INSERIR: designa valores iniciais a atributos e constituintes de um objeto. Tem a forma:

```
INSERIR  (identificador-objeto,
                (Atributo/Constituente:
                                valor/(valores),...));

Exemplo:  INSERIR (Tom,
                   (Nome: "Tom Telles",
                    Endereço:   "Rua  Miguel  Lemos
                                370/1010",
                    NosTelefone:   ("(0123)225876",
                                    "(0123)297115"),
                    Salário: 1050,
                    Experiência:      (Finanças,
                                       Mercado,
                                       Software)));
```

EXCLUIR: remove valores de atributos e constituintes de um objeto.

```
Exemplo:  EXCLUIR  Experiência (Tom) = Software;

          EXCLUIR  Experiência (Tom);
/*  remove todos os valores do conjunto
              Experiência */
```

INCLUIR: acrescenta valores a conjuntos ou listas de atributos e constituintes de um objeto.

```
Exemplo:       INCLUIR       Tom.NosTelefone      =
                                  "(0123)225885";
```

MEMBRO: Faz um Teste da classe do objeto. Tem a forma:

```
MEMBRO (identificador-objeto, Nome-da-Classe);
```

avalia-se "verdadeiro" se o objeto pertence à classe especificada, senão avalia-se "falso"

```
Exemplo:  MEMBRO (Tom, Empregado);
          /*  avalia-se "verdadeiro" se Tom é membro
              da classe Empregado  */
```

FAZER: modifica valores de atributos, constituintes ou funções eliminando qualquer valor prévio.

Tem a forma: FAZER variável = expressão;

Exemplo: FAZER Gerente(Vendas) = joão;

Operações sobre Classes

Operações que permitem acrescentar e eliminar classes dinamicamente: Classe e Remover Classe.

CLASSE: cria uma nova classe de objetos com as seções de definição de tipo já descritas.

REMOVER CLASSE: remove todos os objetos da classe denominada se a classe é independente de alguma outra classe e não tem membros ou subclasses (para simplificar a implementação). Tem a forma:

REMOVER CLASSE Nome-da-Classe;

Operações que permitem mudança dinâmica na classe de objetos: Acrescentar Classe e Deixa de ser da Classe.

ACRESCENTAR CLASSE ... A: inclue um ou mais objetos que pertencem a uma classe dentro de uma subclase determinada. Tem a forma:

ACRESCENTAR CLASSE Nome-da-Subclasse A Expressão1, ..., ExpressãoN;

"Expressão i" evalua um objeto ou uma sequência de objetos.

Exemplo: Supondo que Ivan e Jose são objetos da Classe Pessoa

ACRESCENTAR CLASSE Empregado A Ivan,Jose;

/* inclue aos objetos Ivan e Jose na classe mais especializada de Pessoa: Empregado */

DEIXA DE SER DA CLASSE: elimina um ou mais objetos de uma subclasse determinada. Estes objetos também são automaticamente eliminados das suas subclasses

respectivas se eles não são componentes de outros objetos. Tem a forma:

Expressão1,...,ExpressãoN DEIXA DE SER DA CLASSE Nome-da-subclasse;

Exemplo: Ivan DEIXA DE SER DA CLASSE Empregado;

Operações sobre Versões

São operações que manipulam o grafo de versões:

INSERIR VERSÃO: insere versão de um objeto dado seu identificador no grafo de versões após a última versão inserida. Tem a forma:

INSERIR VERSÃO (identificador-de-objeto [,n]);

O argumento opcional "n" (número de versão) é usado no caso de versões alternativas para identificar a qual das várias versões últimas vai ser ligada esta nova versão.

REMOVER VERSÃO: remove a versão de um objeto e suas extensões. Tem a forma:

REMOVER VERSÃO (identificador-de-objeto,n);

ULTIMA VERSÃO: retorna as versões do objeto mais recentes. (Somente uma versão se é uma versão linear; Um conjunto de versões se versões alternativas).

Tem a forma:

ULTIMA VERSÃO (identificador-de-objeto);

VERSÃO SEGUINTE: retorna a versão sucessora ou sucessoras de uma versão específica. Tem a forma:

VERSÃO SEGUINTE (identificador-de-objeto, n);

PREVIA VERSÃO: retorna a versão predecessora de uma versão específica. Tem a forma:

```
PREVIA VERSÃO (identificador-de-objeto, n);
```

RECUPERAR VERSÃO: retorna uma versão especifica. Tem a forma:

```
RECUPERAR VERSÃO (identificador-de-objeto, n);
```

Operações de Recuperação

O Modelo permite fazer consultas sobre classes de objetos e atributos de objetos, além de fornecer facilidades para referenciar relacionamentos (funções) comparável ao uso de composição funcional em DAPLEX (Shipman,1981).

Usa-se a notação "." para referenciar atributos de objetos.

```
Exemplo: Vendas.NosTelefone
/* refere-se aos Números de Telefone do Departamento de
   Vendas */

Exemplo: Gerente(Vendas)
/* refere-se ao Gerente do Departamento de Vendas */

Exemplo: DeptDo(GerenteDo(Tom)).Nome
/* refere-se ao Nome do Departamento do Gerente de Tom */
```

Para recuperar objetos, coleções de objetos ou atributos de objetos usa-se a sentença RECUPERAR que tem a seguinte forma:

RECUPERAR lista-objetivo [PARA CADA especificação-classe variável-manipulação [COMO especificaçao-classe] [,...]] [TAIS QUE predicado] [ORDENADO POR campo-lista [ASC/DESC] [,...]] [DENTRO DE nome-conjunto];

Os resultados da lista-objetivo podem ser objetos, coleções de objetos ou atributos de objetos.

A cláusula PARA CADA permite iterar sobre uma coleção de objetos.

A cláusula COMO é usada se um objeto é membro de várias classes e é desejável aplicar relacionamentos definidos para uma classe diferente da especificada.

A cláusula TAIS QUE permite especificar um predicado. Um predicado pode conter referências a relacionamentos, chamadas a funções, constantes, variáveis de manipulação, operadores de comparação simples ( ),<,=,<>,>=,<=), operadores de comparação que se aplicam a coleções (EM, NAO EM), operadores lógicos (AND, OR, MENOS) e operadores aritméticos.

A cláusula ORDENADO POR permite que os resultados da operação sejam dispostos numa sequência particular antes de serem exibidos. O ordenamento de um campo da lista pode ser ASC (ascendente, o valor "default") ou DESC (descendente).

A cláusula DENTRO DE especifica o nome do conjunto no qual um resultado vai ser armazenado, o qual pode ser referenciado posteriormente dentro da transação.

```
Exemplo:  RECUPERAR d.nome, e.nome, cursos(e) PARA CADA
                Empregado e COMO Estudante, Departamento d
                TAIS QUE e EM Pessoal(d) AND
                COUNT(Pessoal(d)) > 20;
/* recupera os nomes dos departamentos, nomes e cursos dos
   empregados dos departamentos que têm mais de 20
   empregados */

Exemplo:  RECUPERAR e PARA CADA Empregado e
                TAIS QUE e.Salário > 10000
                DENTRO DE BemPago;
/* recupera os identificadores de objetos tipo Empregado que
   ganham mais de 10000 armazenando-os no conjunto
   "BemPago" */
```

Moldes

Os objetos de escritório e da empresa são exibidos ou impressos como documentos com um certo "layout" (formato). Através do denominado "Processo de Apresentação" (Rabitti,1985), os documentos podem ser apresentados num meio físico (impressora,vídeo) mapeando a estrutura lógica dentro da representação externa. A informação necessária para este processo é denominada a ESTRUTURA "LAYOUT" (estrutura física) em ODA (Horak,1985) que mostra como os elementos de dados devem ajustar-se ao "layout" (formato) do documento no tempo da apresentação.

Estruturas "layout" são modeladas através do mecanismo de MOLDE proposto no modelo, o qual estende as características do mecanismo "template" Imagem e Texto do Modelo para SIEs de Gibbs (Gibbs,1983).

Um único objeto pode usar mais de um molde, permitindo

fornecer múltiplas representações externas para um mesmo objeto. Assim, é possível definir distintas visões de objetos para distintos usuários com restrições de autorização.

O elemento básico da estrutura "layout" é a CAIXA. Cada caixa é uma área bidimensional de forma retangular. Uma página corresponde a uma unidade no meio físico, é a área de referência usada para posicionar o conteúdo do documento.
As caixas podem ser compostas ou básicas. As caixas compostas estão constituídas de uma sequência de caixas básicas e/ou outras caixas compostas; estas caixas podem ser horizontalmente ou verticalmente divididas em áreas (caixas internas filas ou colunas).
As caixas básicas são também denominadas BLOCOS em ODA.

Usa-se a caixa predefinida "CaixaPag" para moldes de documentos que são exibidos em mais de uma tela. As caixas internas de "CaixaPag" determinam o formato do documento.

A sintaxe de um tipo de molde Imagem ou Texto tem a forma:

```
MOLDE TEXTO/IMAGEM NomeMolde PARA NomeClasse
ESTRUTURA:
    NomeCaixaComposta:        [LISTA/VETOR]    [FILA/COLUNA]
                        NomeCaixaBásica/NomeCaixaComposta;
    [NomeCaixaComposta:       [LISTA/VETOR]    [FILA/COLUNA]
              NomeCaixaBásica/NomeCaixaComposta [, ...] ...;
ATRIBUTOS DE CAIXA:
    NOME CAIXA: NomeCaixaComposta/NomeCaixaBásica,
    [TAMANHO CAIXA: largura (inteiro) [altura (inteiro)],]
    [CONTEÚDO: [`cadeia de caracteres´] atributo-objeto,]
```

[ESTILO TIPO: românico/itálico/negrito,]

[TAMANHO TIPO: script/ nota de roda-pé/ pequeno/ normal/ grande,]

[DADOS TEXTO,]

[ALINHAMENTO: C(centrado)/D(à direita)/E(à esquerda),]

[ESPAÇO ENTRE LINHAS: simple/dobre/inteiro,]

[OFFSET INICIAL [:inteiro],]

[ESPAÇO MARGEM SUPERIOR : parag(parágrafo)/ cap(capítulo)/ inteiro,]

[ESPAÇO MARGEM INFERIOR: inteiro,]

[ESPAÇO MARGEM ESQUERDA: inteiro,]

[ESPAÇO MARGEM DIREITA: inteiro,]

[QUEBRA PAGINA ANTES,]

[QUEBRA PAGINA DEPOIS,]

[APRESENTAÇÃO GRAFICA: bordas/negrito/sublinhado,]

[COR AMBIENTE: vermelho/azul/verde,];

A seção ESTRUTURA especifica a estrutura "layout" de um objeto que consiste de um número de caixas básicas ou compostas.

A cláusula LISTA/VETOR indica que a caixa especificada vai ser replicada (horizontalmente ou verticalmente). Usa-se para replicar caixas para formatos de objetos componentes de uma hierarquia ou para formatos de atributos multivalorados.

A cláusula FILA/COLUNA indica que a caixa composta está dividida horizontalmente ou verticalmente em áreas, isto é numa sequência de caixas filas ou caixas colunas respectivamente. Se não se especifica assume-se uma divisão em sequência de caixas filas.

A seção ATRIBUTOS DE CAIXA especifica os valores de atributos para cada caixa que aparece na seção ESTRUTURA.

Para atributos que não aparecem na especificação de uma caixa tais como: largura, estilo tipo, tamanho tipo, espaço entre linhas, margens; assume-se os valores atributos da caixa que o contem.

Descrição dos Atributos de Caixa

NOME CAIXA é o nome da caixa que aparece na seção ESTRUTURA;

TAMANHO CAIXA determina a largura e altura de área da caixa. A largura equivale ao número de caracteres numa linha, e a altura equivale ao número de linhas;

CONTEÚDO identifica o valor que aparece na caixa;

ESTILO TIPO "Românico" é o valor "default";

TAMANHO TIPO "Normal" é o valor "default";

DADOS TEXTO especifica que os dados são de tipo texto;

ALINHAMENTO indica se o valor vai ser centrado (C), alinhado à direita (D) ou alinhado à esquerda (E);

ESPAÇO ENTRE LINHAS fixa o espaçamento entre linhas de um texto. "Simple" é o valor "default";

OFFSET INICIAL estabelece a quantidade de espaço horizontal deixada ("indented") antes do início da primeira linha num parágrafo. Se a quantidade não for especificada, assumirá um

valor "default";

ESPAÇO MARGEM SUPERIOR especifica o número de linhas que devem ser deixadas em branco antes do início da primeira linha.

Usa-se os valores "parag" e "cap" para margens "default" de parágrafo e capítulo respectivamente;

ESPAÇO MARGEM INFERIOR especifica o número de linhas que devem ser deixadas em branco depois da última linha;

ESPAÇO MARGEM ESQUERDA estabelece a quantidade de espaço horizontal deixada na margem esquerda;

ESPAÇO MARGEM DIREITA estabelece a quantidade de espaço horizontal deixada na margem direita;

QUEBRA PAGINA ANTES realiza um quebre de página antes da exibição da caixa;

QUEBRA PAGINA DEPOIS realiza um quebre de página depois da exibição da caixa;

APRESENTAÇÃO GRAFICA indica um tipo especial de edição;

COR AMBIENTE indica um tipo de cor especial a ser usada.

Exemplo:

```
CLASSE  MemoVendas
ATRIBUTOS:
    Título:  string(30);
    DataMemo: string(10);
CONSTITUINTES:
    Autor: Pessoa;
    Corpo:  LISTA DEPENDENTE  Parágrafo;

CLASSE  Parágrafo
ATRIBUTOS:
    TextoParag: texto;

MOLDE TEXTO  MoldeMemoVendas  PARA  MemoVendas
ESTRUTURA:
    CaixaPag:     BlocoTit,    BlocoData,   BlocoAutor,
                  AreaCorpo;
    AreaCorpo:  LISTA  BlocoParag;
ATRIBUTOS DE CAIXA:
    NOME CAIXA:  BlocoTit,
      CONTEÚDO: ` TÍTULO  :  ´  Título;
    NOME CAIXA:  BlocoData,
      CONTEÚDO: ` DATA    :  ´  DataMemo;
    NOME CAIXA:  BlocoAutor,
      CONTEÚDO: ` AUTOR   :  ´  Autor.Nome;
    NOME CAIXA:  BlocoParag,
      CONTEÚDO:  Corpo.TextoParag,
      DADOS TEXTO,
      ESPACO MARGEM SUPERIOR: parag,
      OFFSET INICIAL;
```

Operações de Moldes

São operações de saída que envolvem instâncias de moldes.

MOLDE : declara um identificador (variável de manipulação) para um molde especifico. Tem a forma:

MOLDE Identificador-molde É NomeMolde;

Ex: MOLDE m É MoldeMemoVendas;

MAPEAR : executa o mapeamento de uma estrutura de objeto ou conjunto de objetos sobre uma estrutura de molde.

Os valores da estrutura do objeto são convertidos ao tipo de dados do molde especificado (texto ou imagem).

Tem a forma:

MAPEAR Identificador-objeto(s) EM identificador-molde;

Ex: MAPEAR Memo1 EM m;

APRESENTAR: realiza a apresentação física de um molde.

O molde pode ser apresentado na tela numa posição fixa determinada por um sistema de coordenadas (x,y). Se não se especifica se assume a posição (0,0); isto é, o canto superior esquerdo da página vai coincidir com aquele da tela. Tem a forma:

APRESENTAR identificador-molde [EM posição];

Ex: APRESENTAR m;

Transações

Os tipos de transações suportadas pelo modelo podem ser curtas e longas.

As transações curtas (padrões) são unidades de consistência e restauração. Os bloqueios gerados durante a transação são mantidos até que a transação é encerrada e a restauração no caso de falhas do sistema desfaz todas as atualizações feitas pela transação.

Para aplicações de escritório, também é preciso gerenciar transações longas que envolvem manipulação de textos em documentos que podem durar horas e ainda dias. Durante uma transação longa é possível requisitar vários objetos de segmentos públicos para um segmento privado do usuário, manipulá-los e então devolve-los aos segmentos públicos apresentando resultados parciais do processo. Simultaneamente, outros usuários podem ler (o estado anterior) os objetos de segmentos públicos.
Numa transação longa, a unidade de consistência para objetos requisitados é o tempo de vida dos bloqueios sobre eles, os quais são controlados pelo usuário. Em caso de falhas do sistema, o processo de restauração não desfaz as atualizações já feitas sobre os objetos que foram devolvidos aos segmentos públicos.

Operações para Gerenciar Transações Curtas

INICIAR TRANSAÇÃO: inicia uma transação curta dada pelo usuário.

TERMINAR TRANSAÇÃO: executa um fim normal de uma transação curta, realizando as atualizações físicas sobre os objetos no b.d., e retirando os bloqueios feitos pela transação.

ABORTAR TRANSAÇÃO: executa um fim abortivo de uma transação curta, desfazendo todas as atualizações e retirando os bloqueios sobre os objetos feitos pela transação.

Operações para Gerenciar Transações Longas

INICIAR TRANSAÇÃO LONGA: inicia uma transação longa dada pelo usuário.

TERMINAR TRANSAÇÃO LONGA: encerra uma transação longa, verificando que todos os bloqueios foram retirados do segmento público e que todos os objetos transferidos dentro do segmento privado foram removidos.

ABORTAR TRANSAÇÃO LONGA: encerra uma transação longa, desfazendo todos os bloqueios sobre os objetos públicos que ainda não foram atualizados via a operação DEVOLVER.

Esta operação não desfaz atualizações que já tenham sido realizadas.

REQUISITAR: permite que um objeto mantido num segmento público seja reproduzido e possivelmente alterado num segmento privado. Tem a forma:

REQUISITAR identificador-objeto PARA [ESCRITA / LEITURA]

O parámetro ESCRITA é especificado quando o usuário vai modificar o objeto. Outros usuários não podem requisitar um objeto que foi bloqueado para atualização.

Vários usuários podem simultaneamente requisitar um objeto para LEITURA, o que vai impedir que o objeto

seja alterado. Em ambos casos, outros usuários podem acessar ao objeto do segmento público para leitura somente.

DESFAZER REQUISIÇÃO: remove o objeto transferido do segmento privado, mantendo a versão original no segmento público, e retirando o bloqueio respectivo.

Tem a forma:

DESFAZER REQUISIÇÃO identificador-objeto

DEVOLVER: permite a atualização do objeto público, refletindo as alterações realizadas pelo usuário dentro do segmento privado; verifica que uma operação REQUISITAR sobre o objeto tenha sido feita previamente e retira o bloqueio respectivo. Tem a forma:

DEVOLVER identificador-objeto

V.6. ANÁLISE DO MODELO DE DADOS PROPOSTO

O Modelo de Dados O-O proposto apresenta as características necessárias, que permitem cumprir com os requisitos para um Modelo de Dados para SIEs já descritos em V.2..

1. O modelo conceitual dos usuários de SIEs é suportado através das construções para representar estrutura, comportamento e restrições aplicaveis aos dados do escritório.
2. O modelo permite representar dados de escritório através de:

   . Os tipos de dados não formatados: TEXTO, VOZ e IMAGEM;

   . Os tipos de dados TEMPO e DATA para gerenciar dados de tempo;

As noções de Objetos, Tipos de Objetos, Construtores de Tipos, Hierarquia de Tipos e Classes permitem perceber os dados de escritório como "objetos de escritório" de vários tipos, representar Objetos Complexos, e permitem que os objetos de escritório sejam usados frequentemente em muitas diferentes formas;

O mecanismo de "Molde" permite representar as estruturas "layout" descritas no padrão ODA - "Office Document Architecture".

3. A representação de objetos de tipo arbitrário e o conjunto de operações fornecido, permite a integração de muitos tipos de informação no escritório.
4. Evolução do Escritório: O mecanismo de versões permite gerenciar modificações à estrutura de tipo; as operações sobre Classes permitem acrescentar e eliminar classes dinamicamente, também como permitem a mudança dinâmica na classe de objetos.
5. O modelo permite descrever relacionamentos binários entre objetos através da seção "Mapeamentos" de definição de Tipo; Relacionamentos de Derivação através da noção de atributos e objetos derivados, supertipo/subtipo, classes construídas, e operações de recuperação.
6. O modelo permite o gerenciamento de transações longas, que envolvem manipulação de textos em documentos que podem durar horas e ainda dias.
7. As operações sobre objetos, classes, versões e operações de recuperação, permitem capturar o comportamento de atividades de escritório comuns.
8. Através do mecanismo de "Molde" é possível definir distintas visões dos objetos para distintos usuários.

CAPÍTULO VI

CONCLUSÕES

O modelo de dados orientado a objetos proposto para SIEs neste trabalho está influenciado pelas experiências de outros modelos, tais como: o modelo de Gibbs, Hudson, OMS, DAPLEX, Minstrel-ODM e IRIS.

O modelo propõe mecanismos para suportar representações de estrutura e comportamento de objetos de escritório, que irão facilitar a programação de aplicações de escritório.

O modelo é simples já que apresenta um número de características básicas, acrescentado por um conjunto de características especiais.

Características básicas: Mecanismos de abstração tais como: Tipos de Objetos, Construtores de Tipos, Hierarquia de Tipos, Relacionamentos entre objetos, Tipos de Dados, que suportam a modelagem de objetos complexos e altamente interrelacionados; Classes Base e Classes Construídas.

Características especiais: Tipos de Dados não formatados tais como: texto, voz e imagem; Tipos de Dados definidos pelo usuário; Regras de Derivação para encapsular o comportamento de um objeto, permitindo a definição de atributos e objetos derivados, os quais estão proximamente relacionados à noção de vistas do usuário (esquema externo); Suporte para Restrições de integridade: restrição de cardinalidade mínima, unicidade, restrições explícitas usando predicado de restrições; uma linguagem de consulta de alto nível, amigável ao usuário; Mecanismo de versões; um mecanismo denominado "Molde" que permite representar objetos

nos meios: imagem e texto, que possibilita definir distintas visões dos objetos para distintos usuários; Gerenciamento de transações longas através da requisição e devolução de objetos de segmentos públicos para um segmento privado, e a devolução de objetos de um segmento privado para um segmento público.

A restrição do modelo é a falta de facilidades para impor controle de acesso em um ambiente distribuído.

Extensões futuras: O modelo pode ser usado como o modelo de dados para um novo tipo de sistema de gerenciamento de B.D.; pode fornecer uma base para suportar uma variedade de facilidades de interface de usuários.

O processo de modelagem seria muito facilitado com o desenvolvimento de uma ferramenta automatizada, implementando este modelo, que proporcionasse ao usuário a possibilidade de executar esta tarefa diretamente no computador. Com isto as informações geradas poderiam ser utilizadas em fases posteriores do desenvolvimento, aumentando a consistencia e facilitando a correção de erros que por ventura possam existir.

REFERÊNCIAS BIBLIOGRAFICAS.

ABITEBOUL,S. e HULL,R. (1984); "IFO: A Formal Semantic Database Model"; Proceedings of the Third ACM Symposium on Principles of Database Systems, p.119-132.

AHLSEN,M., BJORNERSTEDT,A., BRITTS,S., HULTEN,C. e SODERLUND,L. (1984); "An Architecture for Object Management in OIS", ACM Trans. Off. Inf. Syst., Vol.2, No.3, p.173-196.

ALBANO,A., CARDELLI,L., e ORSINI,R. (1985); "Galileo: A Strongly-typed, Interactive Conceptual Language", ACM Trans. Database Systems, Vol.10, No.2, p.230-260.

BANCILHON,F. (1988); "Object-Oriented Database Systems", ACM SIGART-SIGMOD-SIGACT Symposium on Principles of DataBase Systems, p.152-162.

BARBER,G. (1983); "Supporting Organizational Problem Solving with a Workstation", ACM Trans. Off. Inf. Syst., Vol.1, No.1, p.45-67.

BARBIC,F., CERI,S., BRACCHI,G. e MOSTACCI,P. (1985a); "Modeling and Integrating Procedures in Office Information Systems Design", Information Systems, Vol.10, No.2, p.149-167.

BARBIC,F. e RABITTI,F. (1985b); "The Type Concept in Office Document Retrieval", Proceedings of Very Large Data Base, Stockholm, p.34-47.

BRACCHI,G. e PERNICI,B. (1984a); "The Design Requirements of Office Systems", ACM Trans. Off. Inf. Syst., Vol.2, No.2, p.151-170.

BRACCHI,G. e PERNICI,B. (1984b); "SOS: A Conceptual Model for Office Information Systems", Data Base, p.11-18.

BRACCHI,G. e PERNICI,B. (1986); "Trends in Office Modeling", Office Systems, Ed. A.A. Verrijn-Stuart e R.A. Hirschheim, North-Holland IFIP.

BRODIE,M. (1984); "On the Development of Data Models", On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages, Eds. Brodie M., Mylopoulos J., Schmidt J. - Springer-Verlag.

CARVALHO,R.F. (1986); Automação de Escritórios, Livros Técnicos e Científicos Editora S.A.

CAVELLUCCI, ZIMMERMANN, STRACK e GUERREIRO (1986); Automação de Escritórios, Cartgraf Editora Ltda. Campinas - SP.

CHORAFAS,D.M. (1982); Office Automation: The Produtivity Challenge, Prentice Hall, Inc., N.J..

CHOY,D.M., BAMFORD,R.J. e TUNG,F.C. (1984); "A Database Management System for Office Systems and Advanced Workstations", ACM SIGOA Newsletter, Vol.5, No.3, p.17-24.

CLUNIE,C.E., TORRES DE CLUNIE, G.E. e FEIGENBAUM D. (1985); La Oficina Informatizada, Relatório Técnico COPPE-UFRJ, ES-72/ Julho 1985.

CODD,E.F. (1981); "Data Models in Database Management", ACM Sigmod Record, Vol.11, No.2.

CROFT,W.B. e LEFKOWITZ,L.S. (1984); "Task Support in an Office System", ACM Trans. Off. Inf. Syst., Vol.2, No.3, p.197-212.

CROFT,W.B. e STEMPLE,D.W. (1987); "Supporting Office Document Architectures with constrained types", Proceedings of ACM SIGMOD, Annual Conference San Francisco May 27-29, p.504-509.

CZEJDO,B. e EMBLEY,D.W. (1984); "Office Form Definition and

Processing using a Relational Data Model", Proceedings of the Second ACM SIGOA Conference on Office Information Systems, Toronto, p.123-142.

DE JONG,P. (1980); "The System for Business Automation: A Unified Application Development System", Proceedings of IFIP Congress 80, Tokyo, p.469-474.

DERRETT,N., KENT,W. e LYNGBAEK,P. (1985); "Some Aspects of Operations in an Object-Oriented Database", IEEE Database Engineering Bulletin, Vol.4, p.302-310.

ELLIS,C.A. (1980a); "Office Modeling", Office Automation Conference Digest, p.53-55.

ELLIS,C.A. e NUTT,G.J. (1980b); "Office Information Systems and Computer Science", ACM Computing Surveys, Vol.12, No.1, p.27-60.

ELLIS,C.A e BERNAL,M. (1982); "Officetalk-D: An Experimental Office Information System", Proceedings ACM SIGOA Conference on Office Information Systems, Vol.3, Nos. 1,2, p.131-140.

FERRANS,J.C. (1982); "SEDL - A Language for specifying integrity constraints on Office Forms", Proceedings ACM SIGOA of Conference on Office Information Systems, Vol.3, Nos. 1 e 2, University of Pennsylvania, Philadelphia, June 21-23, p.123-130.

FINCH, J.H. (1986); "Security of Office Systems", Office Systems, Ed. A.A. Verrijn-Stuart e R.A. Hirschheim, North-Holland IFIP 1986.

FISHMAN, BEECH, CATE, CHOW, CONNORS, DAVIS, DERRETT, HOCH, KENT, LYNGBAEK, MAHBOD, NEIMAT, RYAN e SHAN (1987); "IRIS: An Object-Oriented Database Management System", ACM Trans. Off. Inf. Syst., Vol.5, No.1, p.48-69.

GANE,C. e SARSON,T. (1979); Structured Systems Analysis: Tools and Techniques, Prentice Hall, Inc., N.J..

GIBBS,S.J. (1982); "Office Information Models and the Representation of Office Objects", Proceedings ACM SIGOA Conference on Office Information Systems, Vol.3, Nos. 1 e 2, University of Pennsylvania, June 21-23, p.21-26.

GIBBS,S.J. e TSICHRITZIS,D (1983); "A Data Modeling Approach for Office Information Systems", ACM Trans. Off. Inf. Syst, Vol.1, No.4, p.299-319.

GIBBS,S.J. (1985); "Conceptual Modelling and Office Information Systems", Office Automation, Ed. D. Tsichritzis, Springer-Verlag, Berlin.

GOLDBERG,A. (1981); "Introducing the Smalltalk-80 System", Byte, August 1981, p.14-26.

HAMMER,M. e SIRBU,M. (1980); "What is Office Automation", Office Automation Conference Digest 1980, p.37-49.

HAMMER,M. e MCLEOD,D. (1981); "Database description with SDM: A Semantic Database Model", ACM Trans. on Database Systems, September 1981, p.351-386.

HARPER, DUNNION, SHERWOOD-SMITH e VAN RIJSBERGEN (1986); "Minstrel-ODM: A Basic Office Data Model", Information Processing & Management, Vol.22, No.2, p.83-107.

HEIMBIGNER,D. e MCLEOD,D. (1985); "A Federated Architecture for Information Management", ACM Trans. Off. Inf. Syst., Vol.3, No.3, p.253-278.

HORAK,W. e KRONERT,G. (1984); "An Object-Oriented Office Document Architecture Model for Processing and Interchange of Documents", Proceedings of the Second ACM-SIGOA Conference on Office Information Systems, p.152-160.

HORAK,W. (1985); "Office Document Architecture and Office Document Interchange Formats: Current Status of International Standardization", IEEE Computer, October 1985, p.50-60.

HUDSON,S.E. e KING,R. (1986a); "CACTIS: A Database System for specifying functionally defined data", International Workshop on Object-Oriented Database Systems, IEEE Computer Society.

HUDSON,S.E. e KING,R. (1986b); "A Generator of Direct Manipulation Office Systems", ACM Trans. Off. Inf. Syst., Vol.4, No.2, p.132-163.

KING,R. e MCLEOD,D. (1985); "A Database Design Methodology and Tool for Information Systems", ACM Trans. Off. Inf. Syst., Vol.3, No.1, p.2-21.

KITAGAWA,H., GOTOH.M., MISAKI,S. e AZUMA,M. (1984); "Form Document Management System SPECDOQ - Its Architecture and Implementation", Proceedings of the Second ACM SIGOA Conference on Office Information Systems, June 25-27, p.132-142.

KLING,R. (1985); "Conceptions of Office Automation", ACM SIGOA Bulletin, Vol.6, Nos.1,2, p.15-16.

KONSYNSKI,B., BRACKER,L. e BRACKER,W. (1982); "A Model for Specification of Office Communications", IEEE Trans. on Communications COM, Vol.30, No.1, p.27-36.

KONSYNSKI,B. e SPRAGUE,R. (1986); "Future Research Directions in Model Management", Decision Support Systems, No.2, p.103-109.

KOWARSKI,I. e LOPEZ,M. (1982); "The Document Concept in a Data Base", Communications of the ACM, June 1982, p.276-283.

LAMERSDORF,W. (1984a); "Recursive Data Models for Non-conventional Data Base Applications", Computer Data Engineering Conference (COMPDEC), IEEE Computer Society, April 1984.

LAMERSDORF,W., MULLER,G. e SCHMIDT,J.W. (1984b); "Language Support for Office Modelling", Proceedings of the Tenth International Conference on VLDB, Singapore, p.280-288.

LAMERSDORF,W., SCHMIDT,J.W. e MULLER,G. (1986); "A Recursive Approach to Office Object Modelling", Information Processing & Management, Vol.22, No.2, p.109-119.

LEBENSOLD,J., RADHAKRISHNAN,T. e JAWORSKI,W.M. (1982); "A Modelling Tool for Office Information Systems", Proceedings ACM SIGOA Conference on Office Information Systems, Vol.3, Nos. 1,2, University of Pennsylvania, Philadelphia, p.141-152.

LEE,A., WOO,C. e LOCHOVSKY,F.H. (1984); "OFFICEAID: An Integrated Document Management System", Proceedings of the Second ACM SIGOA Conference in Office Information Systems, p.170-180.

LUM,V.Y., CHOY,D.M. e SHU,N.C. (1982); "OPAS: An Office Procedure Automation System", IBM System Journal, Vol.21, No.3, p.327-350.

LYNGBAEK,P. e VIANU,V. (1987); "Mapping a Semantic Database Model to the Relational Model", Proceedings of ACM SIGMOD Record, Vol.16, No.3, p.132-142.

MOONEY,J. (1984); OZ: An Object-Based System for Implementing Office Information System, M.Sc. Thesis, Department of Computer Sciences, University of Toronto.

MYLOPOULOS, BERNSTEIN e WONG (1980); "A Language Facility for Designing Data Base - Intensive Applications", ACM

Trans. on Database Systems, Vol.5, No.2, p.185-207.

NIERSTRASZ,O.M. e TSICHRITZIS,D.C. (1985); "An Object-Oriented Environment for OIS applications", Proceedings of Very Large Data Base, Stockholm, p.335-345.

OLSON,M.H. e LUCAS,H.C.Jr. (1982); "The Impact of Office Automation on the Organization: Some implications for Research and Practice", Communications of the ACM, Vol.25, No.11, p.838-847.

RABITTI,F. (1985); "A Model for Multimedia Documents", Office Automation, Ed. D. Tsichritzis, Springer-Verlag, Berlin.

RICHTER,G. (1981); "IML - Inscribed Nets for Modeling Text Processing and Database Management Systems", Proc. Conf. on Very Large Data Base ACM, p.363-375.

SASSONE,P.G. e SCHWARTS,A.P. (1986); "Cost-Justifying OA", Datamation, February 1986.

SHEN,S. (1987); "Knowledge Management in Decision Support Systems", Decision Support Systems, No.3.

SHIPMAN,D.W. (1981); "The Functional Data Model and the Data Language DAPLEX", ACM Trans. on Database System, Vol.6, No.1, p.140-173.

SOL,H. (1987); "Conflicting Experiences with DSS", Decision Support Systems, No.3.

SOUZA,J.M. (1987); O Projeto de Engenharia de Dados, Relatório Técnico do Prog. de Eng. de Sistemas e Computação ES-134/87.

TSICHRITZIS,D. (1980); "OFS : An Integrated Form Management System", Proc. Conf. on Very Large Data Bases ACM, p.161-166.

TSICHRITZIS,D.C. (1985); "Objectworld", Office Automation,

Ed. D. Tsichritzis, Springer-Verlag, Berlin.

TSICHRITZIS, FIUME, GIBBS e NIERSTRASZ (1987); "KNOS, Knowledge Acquisition Dissemination and Manipulation Objects", ACM Trans. Off. Inf. Syst., Vol.5, No.1.

TWAITES,K.J. (1984); An Object-Based Programming Environment for OIS, M.Sc. Thesis, Dept of Comp. Sciences, University of Toronto.

WHANG, AMMANN, BOLMARICH, HANRAHAN, HOCHGESANG, HUANG, SOCKUT, SWEENEY, KHOVASANI e KRISHNAMURTHY (1987); "OFFICE-BY-EXAMPLE: An Integrated Office System and Database Manager", ACM Trans. Off. Inf. Syst., Vol.5, No.4, p.393-427.

WINTERBOTTOM,N. e SHARMAN,G.C.H. (1979); NDB: Non-Programmer Data Base Facility, IBM Technical Report TR-12-179, IBM U.K. Laboratories Ltd, England.

WOLFSDORF,P.J. (1988); Automação de Escritórios: Componentes Estratégicos, Ed. Mc Graw-Hill, Ltda., São Paulo.

WOO,C.C. e LOCHOVSKY,F.H. (1985); "An Object-Based Approach to Modelling Office Work", Database Engineering, Vol.4, IEEE Computer Society, p.14-22.

ZANIOLO, AIT-KACI, BEECH, CAMMARATA, KERSCHBERG e MAIER (1986); "Object Oriented Database Systems and Knowledge Systems", Expert Database Systems, Ed. Larry Kerschberg, p.50-65.

ZDONIK,S.B. (1984); "Object Management System Concepts", Proceedings of the Second ACM SIGOA Conference in Office Information Systems, June 25-27, p.13-19.

ZLOOF,M.M. e DE JONG,S.P. (1977); "The System for Business Automation (SBA): Programming Language", Communications

of the ACM, Vol.20, No.6, p.385-396.

ZLOOF,M.M. (1981); "QBE/OBE: A Language for Office and Business Automation", IEEE Computer, Vol.14, No.5, p.13-22.

ZLOOF,M.M. (1982); "OFFICE-BY-EXAMPLE: A Business Language that unifies Data, Word Processing and Electronic Mail", IBM Syst. Journal, Vol.21, No.3, p.272-304.

A P Ê N D I C E

EXEMPLO DE ESQUEMA PARA O SISTEMA "CURRICULUM"

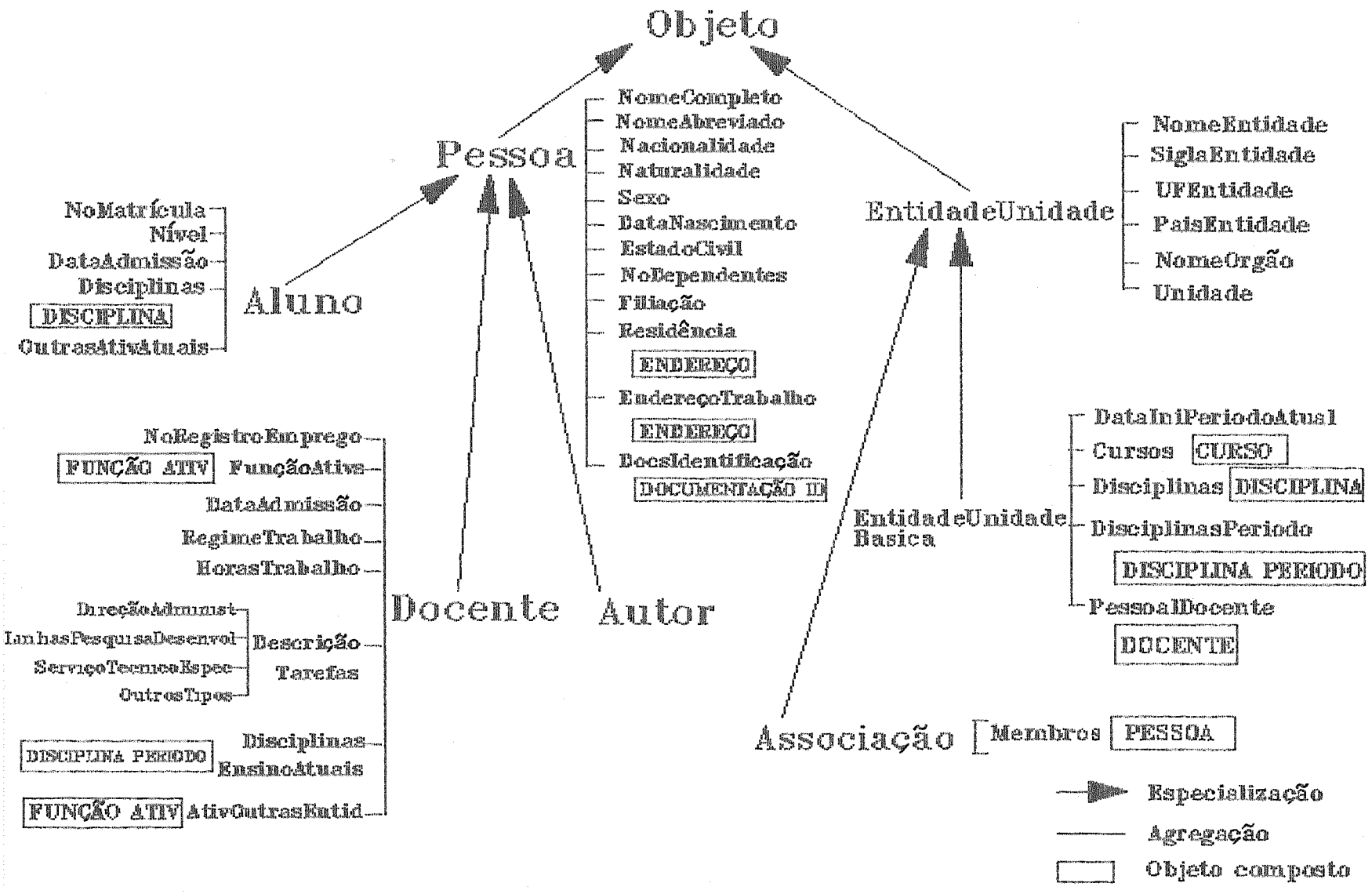


Figura 3: Esquema Objeto - Especializações e Agregações (Parte 1 de 3)

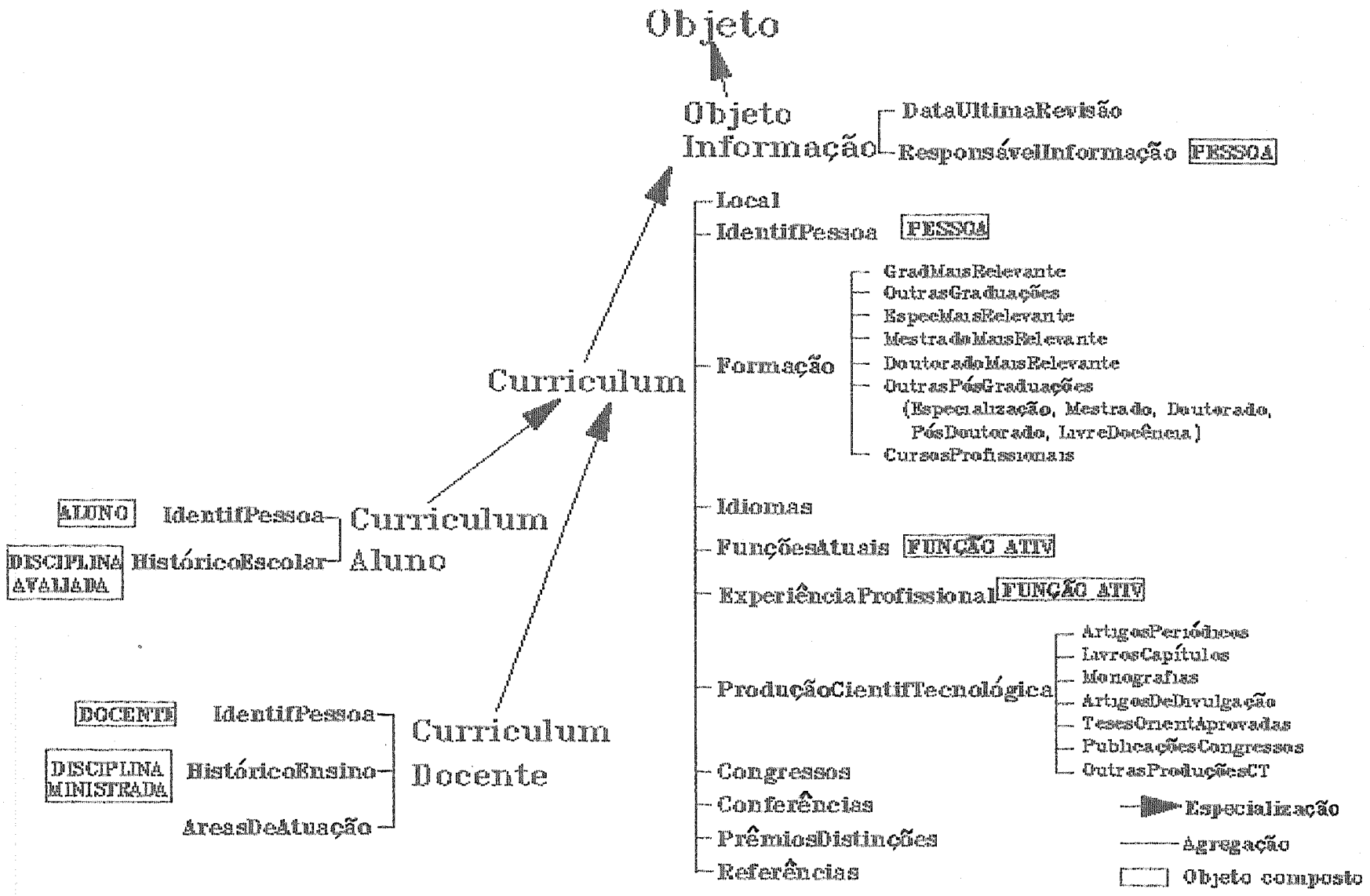


Figura 4: Esquema Objeto – Especializações e Agregações (Parte 2 de 3)

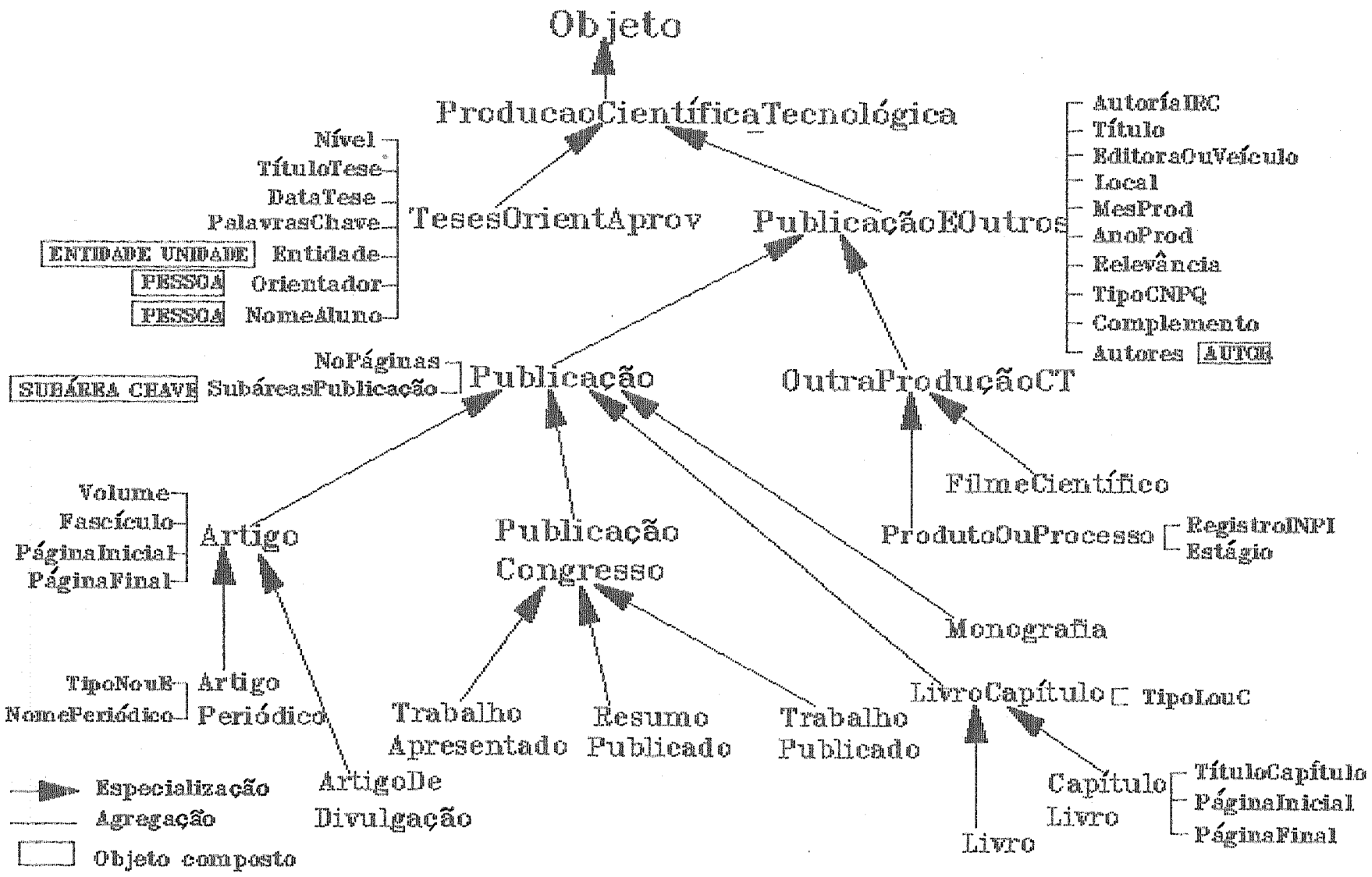


Figura 5: Esquema Objeto – Especializações e Agregações (Parte 3 de 3)

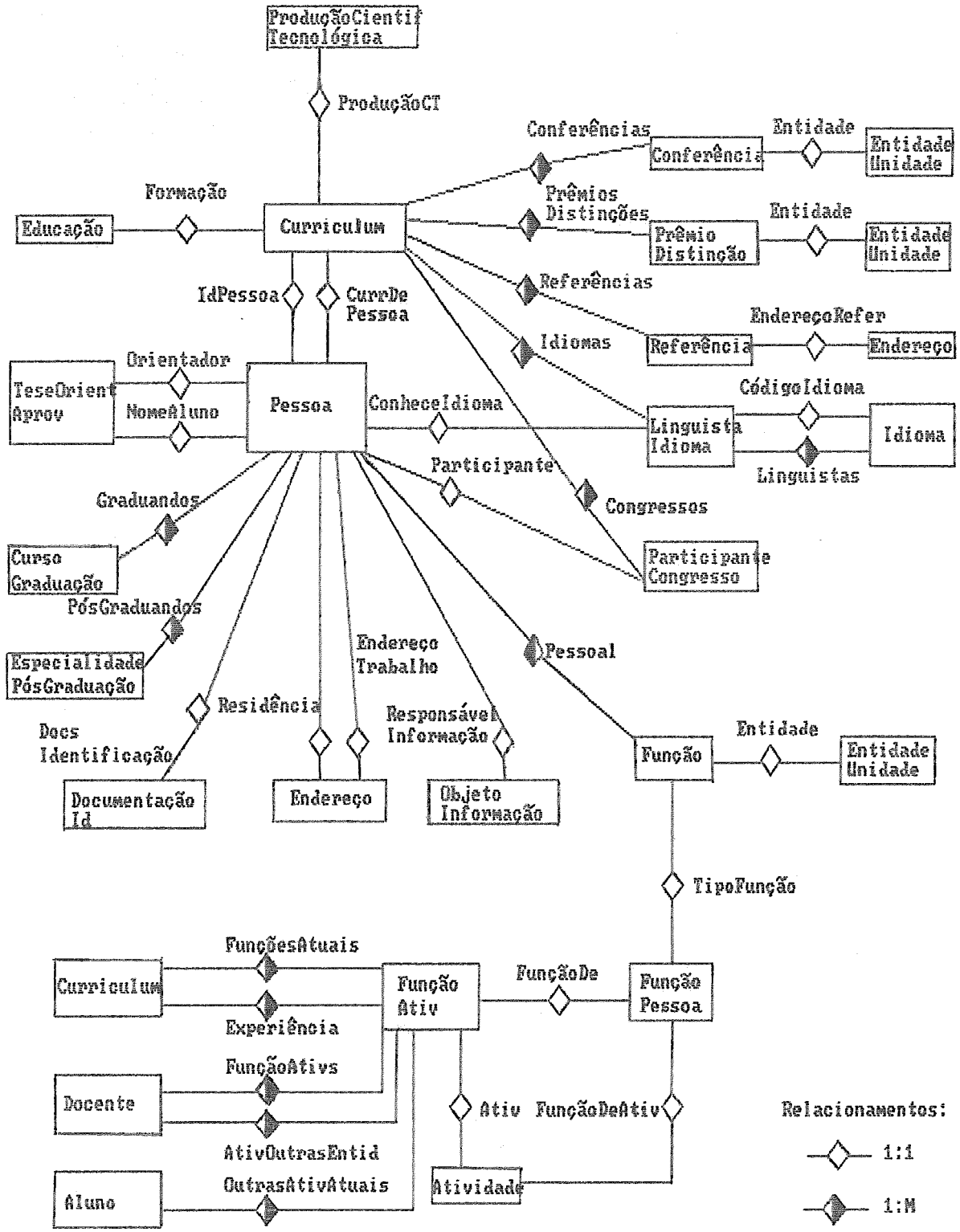


Figura 6: Diagrama Entidade-Relacionamento (Parte 1 de 4)

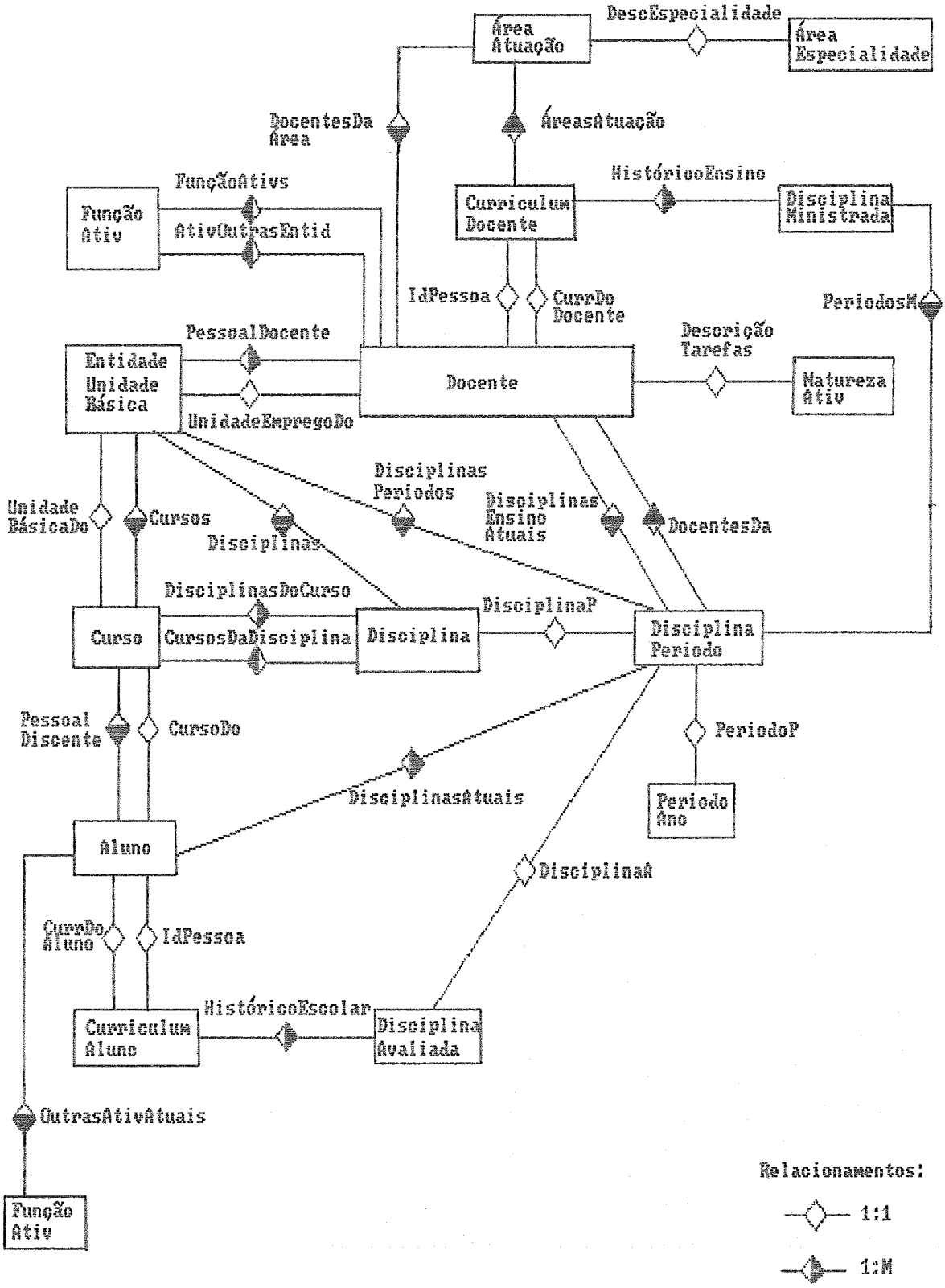


Figura 7: Diagrama Entidade-Relacionamento (Parte 2 de 4)

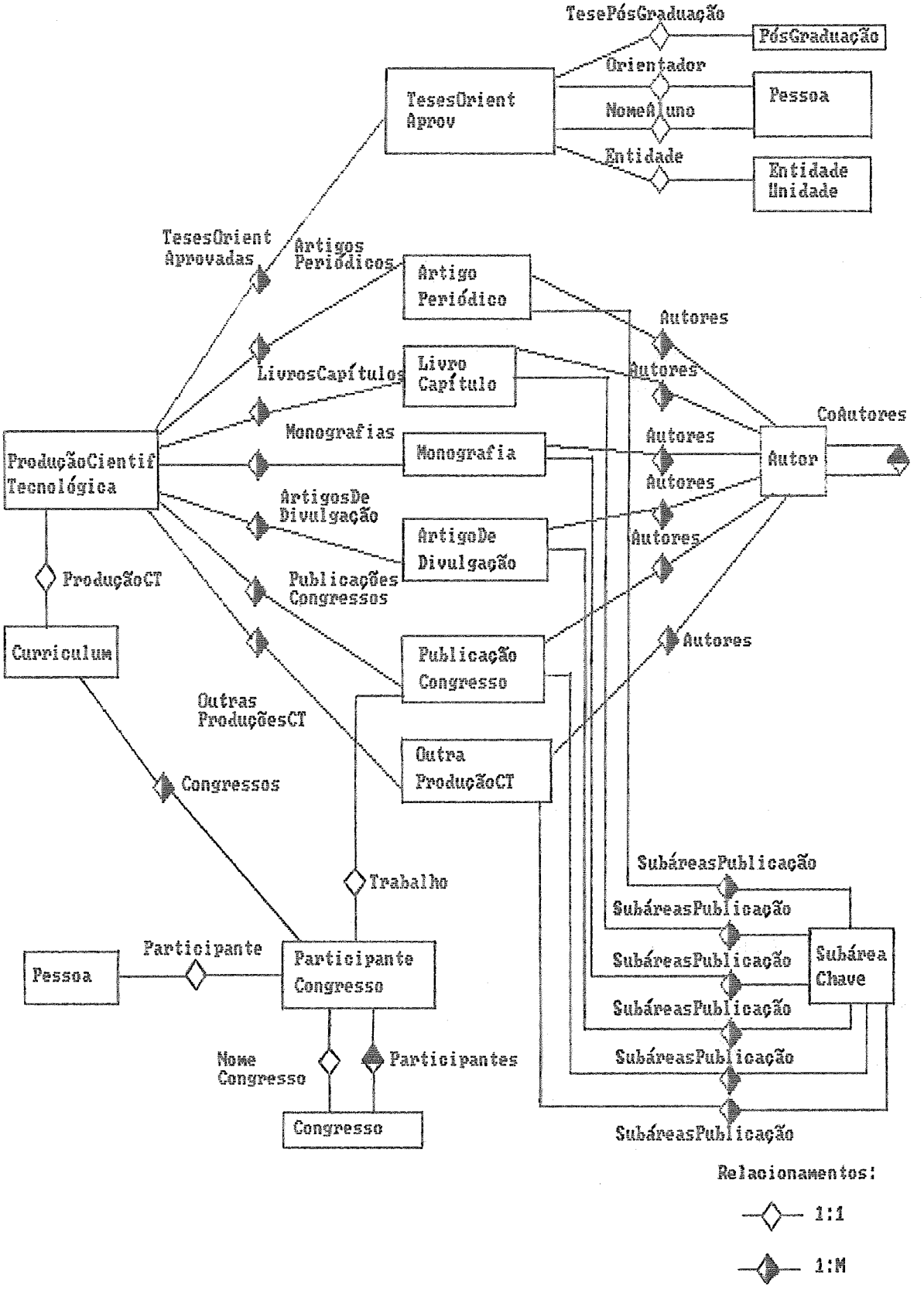


Figura 8: Diagrama Entidade-Relacionamento (Parte 3 de 4)

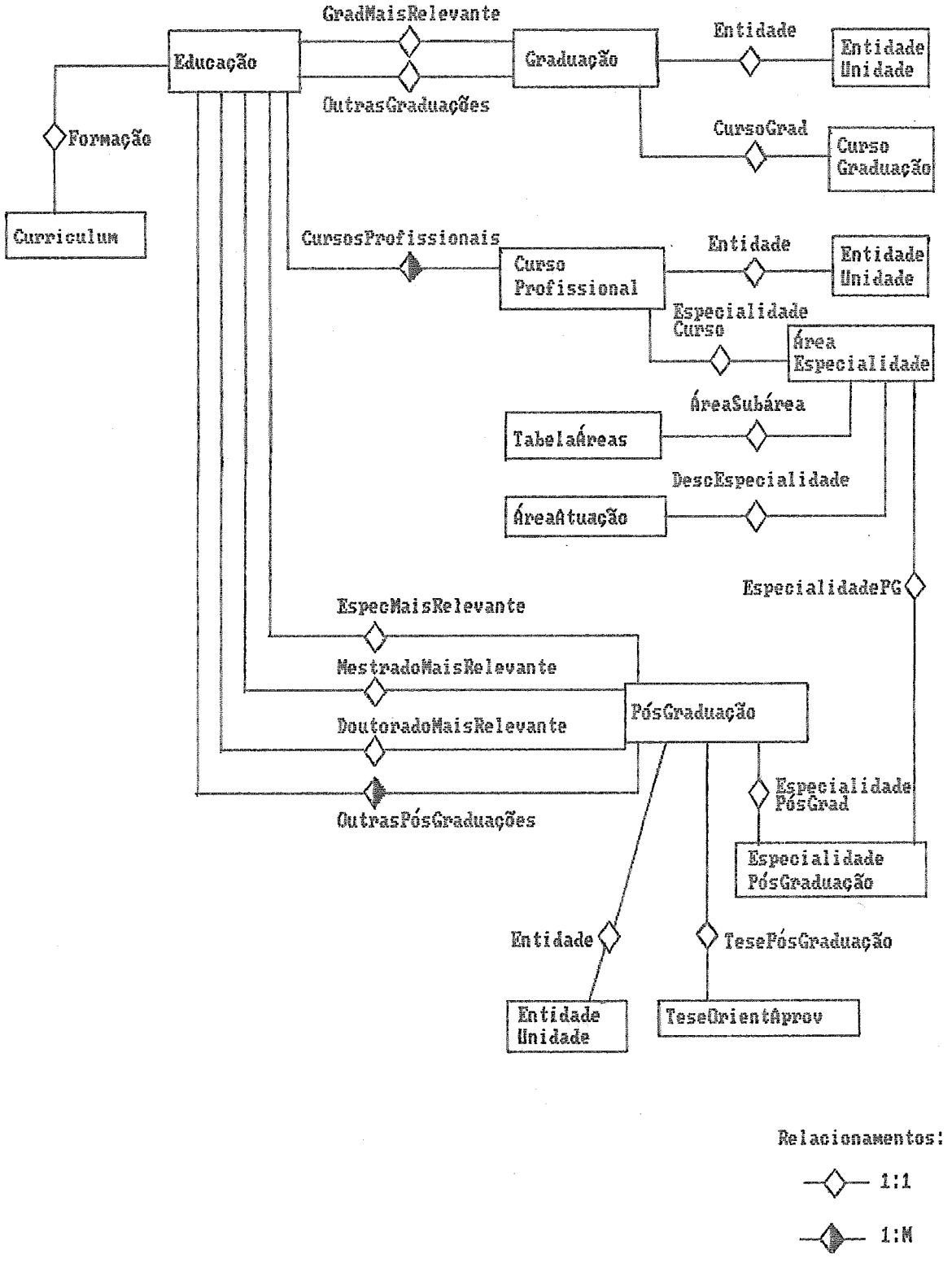


Figura 9: Diagrama Entidade-Relacionamento (Parte 4 de 4)

```
Classe Pessoa: Objeto
 Atributos:
     NomeCompleto: string( ) REQUERIDO;
     NomeAbreviado: string( ) REQUERIDO;
     Nacionalidade: string( ) REQUERIDO;
     Naturalidade: string( );
     DataNascimento: data REQUERIDO;
     Sexo: string(1) REQUERIDO;
     EstadoCivil: string(1) REQUERIDO;
     NoDependentes: inteiro;
     Filiação: string( );
     DescriçãoCivil: string( );
     SexoCNPQ: vetor[1..2] string(1);
     (NomeCompleto) ÚNICO;
 Constituintes:
     DocsIdentificação: DEPENDENTE DocumentaçãoId
                                          REQUERIDO;
     Residência: DEPENDENTE Endereço REQUERIDO;
     EndereçoTrabalho: Endereço;
     (DocsIdentificação) ÚNICO;
     (Residência) ÚNICO;
 Restrições:
     Sexo = "M" OR Sexo = "F";
     EstadoCivil É EM ( "S", "C", "V", "O" );
 Regras:
     DescriçãoCivil   :=    if   EstadoCivil  =   "S"   then
           "Solteiro"   else  if  EstadoCivil  =  "C"   then
           "Casado"     else  if  EstadoCivil  =  "V"   then
           "Viúvo"      else  "Outro";
     SexoCNPQ[1] := if Sexo = "M" then "X" else " ";
     SexoCNPQ[2] := if Sexo = "F" then "X" else " ";
 Versões: Linear;

Classe Docente: Pessoa
 Atributos:
     NoRegistroEmprego: inteiro REQUERIDO;
     DataAdmissão: data REQUERIDO;
     RegimeTrabalho: string(1) REQUERIDO;
     HorasTrabalho: inteiro REQUERIDO;
     RegimeCNPQ: vetor[1..3] string(1);
     (NoRegistroEmprego) ÚNICO;
 Constituintes:
     FunçãoAtivs:  CONJ  DEPENDENTE FunçãoAtiv ORDENADO DESC
                  Ativ(FunçãoAtiv).RelevânciaAtiv  REQUERIDO;
     DisciplinasEnsinoAtuais: CONJ DisciplinaPeriodo;
     DescriçãoTarefas:  DEPENDENTE NaturezaAtiv  REQUERIDO;
     AtivOutrasEntid:  CONJ  DEPENDENTE FunçãoAtiv  ORDENADO
                     DESC    Ativ(FunçãoAtiv).RelevânciaAtiv;
     FunçãoEntid: Fun,çãoPessoa;
     OutrasFunções: CONJ FunçãoPessoa;
     PesquisaEntid: CONJ Atividade;
     PesquisaOutrasEntid: CONJ Atividade;
     BancasExamEntid: CONJ Atividade;
     BancasExamOutrasEntid: CONJ Atividade;
     AdmTécnicoEntid: CONJ Atividade;
     AdmTécnicoOutrasEntid: CONJ Atividade;
     DocênciaAtualCOPPE: FunçãoPessoa;
     DocênciaAtualOutros: CONJ FunçãoPessoa;
     AtivsCNPQ: CONJ FunçãoAtiv;
```

```
    AssociaçõesAtuais: CONJ FunçãoPessoa;
    (FunçãoAtivs) ÚNICO;
    (DescriçãoTarefas) ÚNICO;
    (AtivOutrasEntid) ÚNICO;
Restrições:
    (RegimeTrabalho   =  "P"  AND  HorasTrabalho  >  0  AND
     HorasTrabalho < 40) OR  RegimeTrabalho = "C"  OR
                             RegimeTrabalho = "D";
    HorasTrabalho > 0  AND  HorasTrabalho  <= 40;
    PeriodoP(DisciplinasEnsinoAtuais).DataInício =
         UnidadeEmpregoDo(Docente).DataIniPeriodoAtual;
    FunçãoDe(AtivOutrasEntid).DataTérmino = NULO;
    FunçãoDe(FunçãoAtivs).DataTérmino = NULO;
Regras:
    RegimeCNPQ[1]  := if RegimeTrabalho = "P" then "X" else
                                                    " ";
    RegimeCNPQ[2]  := if RegimeTrabalho = "C" then "X" else
                                                    " ";
    RegimeCNPQ[3]  := if RegimeTrabalho = "D" then "X" else
                                                    " ";
    FunçãoEntid := RECUPERAR FunçãoDe(FunçãoAtivs);
    OutrasFunções  :=  RECUPERAR  FunçãoDe(AtivOutrasEntid)
          ORDENADO POR
          FunçãoDe(AtivOutrasEntid).DataInício ASC;
    PesquisaEntid   :=  RECUPERAR   Ativ(p)  PARA  CADA   p
          FunçãoAtivs TAIS QUE  Ativ(p).Código = "2";
    PesquisaOutrasEntid  := RECUPERAR  Ativ(a) PARA CADA  a
          AtivOutrasEntid TAIS QUE Ativ(a).Código = "2";
    BancasExamEntid  :=  RECUPERAR  Ativ(a)  PARA  CADA   a
          FunçãoAtivs   TAIS  QUE   Ativ(a).Código  =   "8"
          ORDENADO POR Ativ(a).DataInício ASC;
    BancasExamOutrasEntid  := RECUPERAR Ativ(a) PARA CADA a
          AtivOutrasEntid  TAIS QUE  Ativ(a).Código  =  "8"
          ORDENADO POR Ativ(a).DataInício ASC;
    AdmTécnicoEntid   :=  RECUPERAR  Ativ(a)  PARA  CADA  a
          FunçãoAtivs   TAIS QUE  Ativ(a).Código =  "1"  OR
          Ativ(a).Código  =  "5"  OR  Ativ(a).Código =  "9"
          ORDENADO POR Ativ(a).DataInício ASC;
    AdmTécnicoOutrasEntid := RECUPERAR Ativ(a) PARA CADA  a
          AtivOutrasEntid TAIS QUE  Ativ(a).Código = "1" OR
          Ativ(a).Código  =  "5"  OR  Ativ(a).Código =  "9"
          ORDENADO POR Ativ(a).DataInício ASC;
    DocênciaAtualCOPPE := RECUPERAR FunçãoDe(a) PARA CADA a
          FunçãoAtivs  TAIS QUE
          TipoFunção(FunçãoDe(a)).Tipo = "D" AND
       Entidade(TipoFunção(FunçãoDe(a))).Unidade = "COPPE";
    DocênciaAtualOutros := RECUPERAR FunçãoDe(a) PARA  CADA
          a  AtivOutrasEntid  TAIS QUE
          TipoFunção(FunçãoDe(a)).Tipo = "D" AND
          Entidade(TipoFunção(FunçãoDe(a))).Unidade  NOT  =
                                    "COPPE"  ORDENADO  POR
          FunçãoDe(a).DataInício ASC;
    AtivsCNPQ  :=  RECUPERAR a PARA CADA a  AtivOutrasEntid
          TAIS    QUE      Ativ(a).Código   >=    "1"    AND
                           Ativ(a).Código    <=   "6"    AND
                           Ativ(a).DataTérmino = NULO;
    AssociaçõesAtuais := RECUPERAR  FunçãoDe(a) PARA CADA a
          AtivOutrasEntid  TAIS  QUE Ativ(a).Código  =  "6"
          ORDENADO POR Ativ(a).DataInício ASC;
```

```
Classe Aluno: Pessoa
 Atributos:
     NoMatrícula: string(8) REQUERIDO;
     Nível: string(1) REQUERIDO;
     DataAdmissão: data REQUERIDO;
     (NoMatricula) ÚNICO;
 Constituintes:
     DisciplinasAtuais: CONJ DisciplinaPeriodo;
     OutrasAtivAtuais: CONJ DEPENDENTE FunçãoAtiv ORDENADO
            DESC Ativ(FunçãoAtiv).RelevânciaAtiv;
     (OutrasAtivAtuais) ÚNICO;
 Restrições:
     Nível = "M"  OR  Nível = "D";
     PeriodoP(DisciplinasAtuais).DataInício =
         UnidadeBasicaDo(CursoDo(Aluno)).DataIniPeriodoAtual;
     FunçãoDe(OutrasAtivAtuais).DataTérmino = NULO;


Classe Autor: Pessoa;


Classe ObjetoInformação: Objeto
 Atributos:
     DataUltimaRevisão: data REQUERIDO;
 Constituintes:
     ResponsávelInformação: Pessoa;
 Versões: Linear;


Classe DocumentaçãoId
 Atributos:
     CarteiraId: string( );
     OrgãoEmissorId: string( );
     UFCarteira: string(2);
     DataCarteira: data;
     CPF: string( );
     Passaporte: string( );
     RegistroProfissional: string( );
 Restrições:
     UFCarteira É EM TabelaUF.Código  OR  UFCarteira = NULO;

Classe Endereço
 Atributos:
     Rua: string( ) REQUERIDO;
     Bairro: string( ) REQUERIDO;
     Cidade: string( ) REQUERIDO;
     UF: string(2);
     CEP: string(5);
     Telex: string( );
     Telefone: string( );
 Restrições:
     UF  É EM  TabelaUF.Código  OR  UF = NULO;
     CEP É numerico;

Classe TabelaUF: ObjetoInformação
 Atributos:
     Código: string(2) REQUERIDO;
     Descrição: string( ) REQUERIDO;
```

```
Classe NaturezaAtiv
 Atributos:
     DireçãoAdminist: texto;
     LinhasPesquisaDesenvol: texto;
     ServiçoTécnicoEspec: texto;
     OutrosTipos: texto;


Classe EntidadeUnidade: Objeto
 Atributos:
     NomeEntidade: string( ) REQUERIDO;
     SiglaEntidade: string( );
     UFEntidade: string(2);
     PaisEntidade: string( );
     NomeOrgão: string( );
     Unidade: string( ) REQUERIDO;
     (NomeEntidade,Unidade) ÚNICO;
 Restrições:
     UFEntidade É EM TabelaUF.Código  OR  UFEntidade = NULO;


Classe EntidadeUnidadeBasica: EntidadeUnidade
 Atributos:
     DataIniPeriodoAtual: data REQUERIDO;
 Constituintes:
     Cursos: CONJ DEPENDENTE Curso REQUERIDO;
     Disciplinas: CONJ Disciplina REQUERIDO;
     DisciplinasPeriodos: CONJ DisciplinaPeriodo;
     PessoalDocente: CONJ Docente;
     (Cursos) ÚNICO;
     (Disciplinas) ÚNICO;
     (DisciplinasPeriodos) ÚNICO;
     (PessoalDocente) ÚNICO;
 Mapeamentos:
      UnidadeEmpregoDo: PessoalDocente --->
                              EntidadeUnidadeBasica;
      UnidadeBasicaDo: Cursos ---> EntidadeUnidadeBasica;
      DocentesDa: DisciplinasPeriodos ---> PessoalDocente;
      DisciplinasDoCurso: Cursos ---> Disciplinas;
      CursosDaDisciplina: Disciplinas ---> Cursos;
 Restrições:
      DataIniPeriodoAtual <= Data( );
 Versões: Linear;


Classe Curso
 Atributos:
     NomeCurso: string( ) REQUERIDO;
     (NomeCurso) ÚNICO;
 Constituintes:
     PessoalDiscente: CONJ Aluno;
     (PessoalDiscente) ÚNICO;
 Mapeamentos:
     CursoDo: PessoalDiscente ---> Curso;
```

```
Classe Disciplina
 Atributos:
     CódigoDisci: string(6) REQUERIDO;
     NomeDisci: string( ) REQUERIDO;
     Créditos: real REQUERIDO;
     (CódigoDisci) ÚNICO;
 Restrições:
     Créditos >= 0;


Classe PeriodoAno
 Atributos:
     Periodo: string(1) REQUERIDO;
     AnoP: string(2) REQUERIDO;
     DataInício: data REQUERIDO;
     (Periodo,AnoP) ÚNICO;
 Restrições:
     Periodo >= "1" AND Periodo <= "4";
     AnoP >= "00" AND  AnoP <= AnoS(Data( ));


Classe DisciplinaPeriodo
 Constituintes:
     DisciplinaP: Disciplina REQUERIDO;
     PeriodoP: PeriodoAno REQUERIDO;
     (DisciplinaP,PeriodoP) ÚNICO;


Classe DisciplinaAvaliada
 Atributos:
     Avaliação: string(1) REQUERIDO;
 Constituintes:
     DisciplinaA: DisciplinaPeriodo REQUERIDO;
 Restrições:
     Avaliação É EM ("A", "B", "C", "D", "I");


Classe DisciplinaMinistrada
 Constituintes:
     DisciplinaM: Disciplina;
     PeriodosM: CONJ DisciplinaPeriodo  ORDENADO ASCEN
           PeriodoP(DisciplinaPeriodo).DataInício REQUERIDO;
 Regras:
     DisciplinaM :=  DisciplinaP(PeriodosM);


Classe Função
 Atributos:
     Tipo: string(1) REQUERIDO;
     Nome: string( ) REQUERIDO;
 Constituintes:
     Entidade: EntidadeUnidade REQUERIDO;
     Pessoal: CONJ Pessoa;
 Restrições:
     Tipo = "D"  OR  Tipo = "O";
```

```
Classe Atividade
 Atributos:
      Código: string(1) REQUERIDO;
      DataInício: data REQUERIDO;
      DataTérmino: data;
      EspecificaçãoAtiv: string( );
      RelevânciaAtiv: inteiro REQUERIDO;
 Restrições:
      Código >= "1"  AND  Código <= "9";
      RelevânciaAtiv >= 0 AND RelevânciaAtiv < 10;


Classe FunçãoPessoa
 Atributos:
      DataInício: data REQUERIDO;
      DataTérmino: data;
      Salário: real;
 Constituintes:
      TipoFunção: Função REQUERIDO;
 Restrições:
      Salário > 0.  OR  Salário = NULO;


Classe FunçãoAtiv
 Constituintes:
      FunçãoDe: FunçãoPessoa  REQUERIDO;
      Ativ: Atividade  REQUERIDO;
 Mapeamentos:
      FunçãoDeAtiv: Ativ ---> FunçãoDe;


Classe ÁreaAtuação
 Constituintes:
      DescEspecialidade: ÁreaEspecialidade REQUERIDO;
      DocentesDaÁrea: CONJ  Docente;
      (DescEspecialidade) ÚNICO;


Classe Curriculum: ObjetoInformação
 Atributos:
      Local: string( );
      TotProdução: vetor [1..17] inteiro;
      IndiceIdiomasCNPQ: vetor [1..8] inteiro;
      DescriçãoIdiomasCNPQ: vetor [1..8] string(10);
      FalaIdiomasCNPQ: vetor [1..8] string(6);
      LêIdiomasCNPQ: vetor [1..8] string(6);
      EscreveIdiomasCNPQ: vetor [1.8] string(6);
 Constituintes:
      IdPessoa: Pessoa  REQUERIDO;
      Formação: DEPENDENTE  Educação  REQUERIDO;
      Idiomas: CONJ  LinguistaIdioma;
      FunçõesAtuais:  CONJ  DEPENDENTE  FunçãoAtiv   ORDENADO
                      DESC Ativ(FunçãoAtiv).RelevânciaAtiv;
      Experiência: CONJ DEPENDENTE  FunçãoAtiv  ORDENADO
                      DESC Ativ(FunçãoAtiv).RelevânciaAtiv;
      Conferências:  CONJ DEPENDENTE Conferência ORDENADO ASC
                      Conferência.DataConferência;
      ProduçãoCT: DEPENDENTE ProduçãoCientifTecnológica;
```

```
    Congressos: CONJ  ParticipanteCongresso ORDENADO DESC
         NomeCongresso(ParticipanteCongresso).AnoCongresso;
    PrêmiosDistinções:   CONJ  DEPENDENTE   PrêmioDistinção
                  ORDENADO ASC  PrêmioDistinção.DataPrêmio;
    Referências: CONJ DEPENDENTE Referência;
    ExperiênciaPesquisa: CONJ Atividade;
    ExperiênciaFunções: CONJ FunçãoPessoa;
    ExpEnsinoGraduação: CONJ Atividade;
    ExpEnsinoPósGrad: CONJ Atividade;
    ExpBancasExam: CONJ Atividade;
    ExpAdmTécnica: CONJ Atividade;
    ExpDocênciaCOPPE: CONJ FunçãoPessoa;
    ExpDocênciaOutros: CONJ FunçãoPessoa;
    ExpAtivsCNPQ: CONJ FunçãoAtiv;
    AssociaçõesAnteriores: CONJ FunçãoPessoa;
    (IdPessoa) ÚNICO;
    (Formação) ÚNICO;
    (FunçõesAtuais) ÚNICO;
    (Experiência) ÚNICO;
    (Conferências) ÚNICO;
    (ProduçãoCT) ÚNICO;
    (PrêmiosDistinções) ÚNICO;
    (Referências) ÚNICO;
Restrições:
    FunçãoDe(FunçõesAtuais).DataTérmino = NULO;
    FunçãoDe(Experiência).DataTérmino <> NULO;
Mapeamentos:
    CurrDePessoa: IdPessoa ---> Curriculum;
Regras:
    TotProduçãoC1]:= ProduçãoCT.TotalProduçãoCTC1];
    TotProduçãoC2]:= ProduçãoCT.TotalProduçãoCTC2];
    TotProduçãoC3]:= ProduçãoCT.TotalProduçãoCTC7];
    TotProduçãoC4]:= ProduçãoCT.TotalProduçãoCTC8];
    TotProduçãoC5]:= ProduçãoCT.TotalProduçãoCTC9];
    TotProduçãoC6]:= ProduçãoCT.TotalProduçãoCTC10];
    TotProduçãoC7]:= ProduçãoCT.TotalProduçãoCTC11];
    TotProduçãoC8]:= ProduçãoCT.TotalProduçãoCTC12];
    TotProduçãoC9]:= ProduçãoCT.TotalProduçãoCTC3];
    TotProduçãoC10]:= ProduçãoCT.TotalProduçãoCTC4];
    TotProduçãoC11]:= Formação.TotalTesesM;
    TotProduçãoC12]:= Formação.TotalTesesD;
    TotProduçãoC13]:= ProduçãoCT.TotalProduçãoCTC5];
    TotProduçãoC14]:= ProduçãoCT.TotalProduçãoCTC6];
    TotProduçãoC15]:= COUNT (e PARA CADA e Experiência TAIS
                  QUE   Ativ(e).Código  = "7") +  COUNT  (f
                  PARA   CADA  f  FunçõesAtuais  TAIS   QUE
                  Ativ(f).Código = "7");
    TotProduçãoC16]:= COUNT (e PARA CADA e Experiência TAIS
                  QUE   Ativ(e).Código  = "8") +  COUNT  (f
                  PARA   CADA  f  FunçõesAtuais  TAIS   QUE
                  Ativ(f).Código = "8");
    TotProduçãoC17]:= ProduçãoCT.TotalProduçãoCTC13];
    ExperiênciaPesquisa  := RECUPERAR  Ativ(e) PARA CADA  e
         Experiência TAIS QUE  Ativ(e).Código = "2";
    ExperiênciaFunções :=
         RECUPERAR FunçãoDe(Experiência) ORDENADO POR
         FunçãoDe(Experiência).DataInício ASC;
```

```
ExpEnsinoGraduação  :=  RECUPERAR Ativ(e) PARA  CADA  e
     Experiência   TAIS   QUE   Ativ(e).Código  =   "3"
     ORDENADO POR  Ativ(e).DataInício ASC;
ExpEnsinoPósGrad  :=  RECUPERAR Ativ(e) PARA  CADA  e
     Experiência   TAIS   QUE   Ativ(e).Código  =   "4"
     ORDENADO POR  Ativ(e).DataInício ASC;
ExpBancasExam   :=  RECUPERAR  Ativ(e)  PARA   CADA   e
     Experiência   TAIS   QUE   Ativ(e).Código  =   "8"
     ORDENADO POR  Ativ(e).DataInício ASC;
ExpAdmTécnica   :=   RECUPERAR  Ativ(e)  PARA  CADA   e
     Experiência  TAIS  QUE  Ativ(e).Código  =  "1"  OR
     Ativ(e).Código  =  "5"  OR  Ativ(e).Código  =  "9"
     ORDENADO POR  Ativ(e).DataInício ASC;
ExpDocênciaCOPPE :=  RECUPERAR FunçãoDe(e) PARA CADA  e
     Experiência TAIS QUE
     TipoFunção(FunçãoDe(e)).Tipo = "D"  AND
    Entidade(TipoFunção(FunçãoDe(e))).Unidade = "COPPE"
     ORDENADO POR  FunçãoDe(e).DataInício ASC;
ExpDocênciaOutros :=  RECUPERAR FunçãoDe(e) PARA CADA e
     Experiência TAIS QUE
     TipoFunção(FunçãoDe(e)).Tipo = "D"  AND
     Entidade(TipoFunção(FunçãoDe(e))).Unidade   NOT  =
     "COPPE" ORDENADO POR  FunçãoDe(e).DataInício  ASC;
ExpAtivsCNPQ  :=  RECUPERAR e PARA CADA  e  Experiência
     TAIS    QUE    Ativ(e).Código    >=    "1"     AND
                    Ativ(e).Código    <=    "6";
AssociaçõesAnteriores  :=  RECUPERAR  FuncaoDe(e)  PARA
     CADA  e Experiência TAIS QUE Ativ(e).Código =  "6"
     ORDENADO POR FunçãoDe(e).DataInício ASC;
IndiceIdiomasCNPQ :=
  BEGIN
    var:vetor[1..8] inteiro; k:inteiro; j:inteiro;
    k:=0; j:=5; FOR m:= 1 TO 8  var[m]:= 99;
    FOR EACH i IN Idiomas
     BEGIN
      k:= k+1;
      if CódigoIdioma(i).Código >= 1 AND
         CódigoIdioma(i).Código <= 5     then
            var[CódigoIdioma(i).Código] := k
       else if j < 8 then BEGIN j:=j+1; var[j]:=k; END;
     END;
    RETURN (var);
  END;
DescriçãoIdiomasCNPQ :=
  BEGIN
    vardes:vetor[1..8] string(10); i:inteiro;
    vardes := spaces;
    FOR j := 1 TO 8
     BEGIN
      i:= IndiceIdiomasCNPQ[j];
      if NOT (i = 99 OR j < 6) then
       vardes[j]:=  CódigoIdioma(Idiomas[i]).Descricao;
     END;
    RETURN (vardes);
  END;
```

```
      FalaIdiomasCNPQ :=
        BEGIN
          varfala:vetor[1..8] string(6); i:inteiro;
          var1: vetor[1..4] string(2); varfala := spaces;
          FOR j := 1 TO 8
           BEGIN
            var1:= spaces; i:= IndiceIdiomasCNPQ[j];
            if NOT (i = 99 OR j < 6) then
               var1[Idiomas[i].Fala]:= "X ";
               varfala[j] := var1;
           END;
          RETURN (varfala);
        END;
      LêIdiomasCNPQ :=
        BEGIN
          varlê:vetor[1..8] string(6); i:inteiro;
          var2: vetor[1..4] string(2); varlê := spaces;
          FOR j := 1 TO 8
           BEGIN
            var2:= spaces; i:= IndiceIdiomasCNPQ[j];
            if NOT (i = 99 OR j < 6) then
               var2[Idiomas[i].Fala]:= "X ";
               varlê[j] := var2;
           END;
          RETURN (varlê);
        END;
      EscreveIdiomasCNPQ :=
        BEGIN
          varescreve:vetor[1..8] string(6); i:inteiro;
          var3: vetor[1..4] string(2); varescreve := spaces;
          FOR j := 1 TO 8
           BEGIN
            var3:= spaces; i:= IndiceIdiomasCNPQ[j];
            if NOT (i = 99 OR j < 6) then
               var3[Idiomas[i].Escreve]:= "X ";
               varescreve[j] := var3;
           END;
          RETURN (varescreve);
        END;


Classe CurriculumAluno: Curriculum
 Constituintes:
      IdPessoa: Aluno REQUERIDO;
      HistóricoEscolar:   CONJ   DisciplinaAvaliada  ORDENADO
        ASCEN
        PeriodoP(DisciplinaA(DisciplinaAvaliada)).DataInício;
 Restrições:
      DisciplinaA(HistóricoEscolar).DataInício <
      UnidadeBasicaDo(CursoDo(IdPessoa)).DataIniPeriodoAtual;
 Mapeamentos:
      CurrDoAluno: IdPessoa  ---> CurriculumAluno;
```

```
Classe CurriculumDocente: Curriculum
 Constituintes:
     IdPessoa: Docente REQUERIDO;
     ÁreasAtuação: CONJ ÁreaAtuação REQUERIDO;
     HistóricoEnsino: CONJ DisciplinaMinistrada;
     Funções: CONJ FunçãoPessoa;
     DocênciaCOPPE: CONJ FunçãoPessoa;
     DocênciaOutros: CONJ FunçãoPessoa;
     Associações: CONJ FunçãoPessoa;
     PesquisaFunçõesAtuais: CONJ Atividade;
     BancasExaminadoras: CONJ Atividade;
     AtivsAdmTécnicas: CONJ Atividade;
 Mapeamentos:
     CurrDoDocente: IdPessoa  ---> CurriculumDocente;
 Regras:
     TotProduçãoC15]:= COUNT (e PARA CADA e Experiência TAIS
                    QUE   Ativ(e).Código  = "7") +  COUNT  (f
                    PARA  CADA f  FunçãoAtivs(IdPessoa)  TAIS
                    QUE Ativ(f).Código = "7") + COUNT (g PARA
                    CADA g AtivOutrasEntid(IdPessoa) TAIS QUE
                    Ativ(g).Código = "7");
     TotProduçãoC16]:= COUNT (e PARA CADA e Experiência TAIS
                    QUE    Ativ(e).Código  =  "8")  +
                    COUNT (BancasExamEntid(IdPessoa)) +
                    COUNT (BancasExamOutrasEntid(IdPessoa));
     Funções          :=       ExperiênciaFunções         UNIÃO
        FunçãoEntid(IdPessoa) UNIÃO OutrasFunções(IdPessoa);
     DocênciaCOPPE       :=       ExpDocênciaCOPPE        UNIÃO
        DocênciaAtualCOPPE(IdPessoa);
     DocênciaOutros      :=      ExpDocênciaOutros        UNIÃO
        DocênciaAtualOutros(IdPessoa);
     Associações       :=      AssociaçõesAnteriores      UNIÃO
        AssociaçõesAtuais(IdPessoa);
     PesquisaFunçõesAtuais := PesquisaEntid(IdPessoa)     UNIÃO
        PesquisaOutrasEntid(IdPessoa);
     BancasExaminadoras   :=   ExpBancasExaminadoras      UNIÃO
        BancasExamEntid(IdPessoa)                         UNIÃO
        BancasExamOutrasEntid(IdPessoa);
     AtivsAdmTécnicas       :=      ExpAdmTécnica         UNIÃO
        AdmTécnicoEntid(IdPessoa)                         UNIÃO
        AdmTécnicoOutrasEntid(IdPessoa);
```

```
Classe Educação
 Atributos:
     TotalTesesM: inteiro;
     TotalTesesD: inteiro;
 Constituintes:
     GradMaisRelevante: DEPENDENTE Graduação REQUERIDO;
     OutrasGraduações: CONJ DEPENDENTE Graduação
         ORDENADO DESC Graduação.Prioridade;
     EspecMais Relevante: DEPENDENTE PósGraduação;
     MestradoMaisRelevante: DEPENDENTE PósGraduação;
     DoutoradoMaisRelevante: DEPENDENTE Pós Graduação;
     OutrasPósGraduações: CONJ DEPENDENTE  PósGraduação
         ORDENADO DESC PósGraduação.Prioridade;
     CursosProfissionais: CONJ DEPENDENTE CursoProfissional
         ORDENADO DESC CursoProfissional.Prioridade;
     TeseMestrado: PósGraduação;
     TeseDoutorado: PósGraduação;
     OutrasTesesMestDout: CONJ PósGraduação;
     TesesMestDout: CONJ PósGraduação;
     PósGraduações: CONJ PósGraduação;
     (GradMaisRelevante) ÚNICO;
     (OutrasGraduações) ÚNICO;
     (EspecMaisRelevante) ÚNICO;
     (MestradoMaisRelevante) ÚNICO;
     (DoutoradoMaisRelevante) ÚNICO;
     (OutrasPósGraduações) ÚNICO;
     (CursosProfissionais) ÚNICO;
 Regras:
     TotalTesesM := COUNT(MestradoMaisRelevante TAIS QUE
           TesePósGraduação(MestradoMaisRelevante)  NÃO  É
           NULO)  +  COUNT(m PARA CADA m OutrasPósGraduacoes
           TAIS  QUE  EspecialidadePósGrad(m).Nível = 2  AND
           TesePósGraduação(m) NÃO É NULO);
     TotalTesesD  := COUNT(DoutoradoMaisRelevante  TAIS  QUE
           TesePósGraduação(DoutoradoMaisRelevante)  NÃO  É
           NULO)  +  COUNT(d PARA CADA d OutrasPósGraduacoes
           TAIS  QUE  EspecialidadePósGrad(d).Nível = 3  AND
           TesePósGraduação(d) NÃO É NULO);
     TeseMestrado  := RECUPERAR  MestradoMaisRelevante  TAIS
           QUE TesePósGraduação(MestradoMaisRelevante) NÃO É
           NULO;
     TeseDoutorado  := RECUPERAR DoutoradoMaisRelevante TAIS
           QUE TesePósGraduação(DoutoradoMaisRelevante)  NÃO
           É NULO;
     OutrasTesesMestDout   :=   RECUPERAR  p  PARA  CADA  p
           OutrasPósGraduações          TAIS            QUE
           (EspecialidadePósGrad(p).Nível     =     2     OR
           EspecialidadePósGrad(p).Nível      =    3)    AND
           TesePósGraduação(p) NÃO É NULO;
     TesesMestDout := TeseDoutorado UNIÃO TeseMestrado UNIÃO
           OutrasTesesMestDout;
     PósGraduações    :=    DoutoradoMaisRelevante     UNIÃO
           MestradoMaisRelevante  UNIÃO  EspecMaisRelevante
           UNIÃO OutrasPósGraduações;
```

```
Classe Graduação
 Atributos:
     DataInício: data REQUERIDO;
     DataObtenção: data;
     Prioridade: inteiro REQUERIDO;
 Constituintes:
     CursoGrad: CursoGraduação REQUERIDO;
     Entidade: EntidadeUnidade REQUERIDO;
 Restrições:
     Prioridade >= 0  AND  Prioridade < 10;


Classe PósGraduação
 Atributos:
     DataInício: data REQUERIDO;
     DataObtenção: data;
     Prioridade: inteiro REQUERIDO;
 Constituintes:
     TesePósGraduação: TeseOrientAprov;
     EspecialidadePósGrad: EspecialidadePósGraduação
                                        REQUERIDO;
     Entidade: EntidadeUnidade REQUERIDO;
 Restrições:
     Prioridade >= 0  AND  Prioridade < 10;


Classe CursoProfissional
 Atributos:
     NomeCurso: string( ) REQUERIDO;
     DataInício: data REQUERIDO;
     DataTérmino: data REQUERIDO;
     NoHoras: inteiro REQUERIDO;
     Prioridade: inteiro REQUERIDO;
 Constituintes:
     EspecialidadeCurso: ÁreaEspecialidade REQUERIDO;
     Entidade: EntidadeUnidade REQUERIDO;
 Restrições:
     NoHoras > 0  AND  NoHoras < 360;
     Prioridade >= 0  AND  Prioridade < 10;


Classe ÁreaEspecialidade
 Atributos:
     Especialidade: string( );
 Constituintes:
     ÁreaSubárea: TabelaÁreas REQUERIDO;


Classe CursoGraduação
 Atributos:
     Curso: string( ) REQUERIDO;
     (Curso) ÚNICO;
 Constituintes:
     Graduandos: CONJ Pessoa;
```

```
Classe EspecialidadePósGraduação
 Atributos:
     Nível: inteiro;
     NívelCNPQ: vetor[1..5] string(1);
     DescriçãoNível: string( );
 Constituintes:
     EspecialidadePG: ÁreaEspecialidade REQUERIDO;
     PósGraduandos: CONJ Pessoa;
 Restrições:
     Nível >= 1  AND  Nível <= 5;
 Regras:
     NívelCNPQ[Nível] := "X";
     DescriçãoNível  := if Nível = 1  then  "Especialização"
                   else  if Nível = 2  then  "Mestrado"
                   else  if Nível = 3  then  "Doutorado"
                   else  if Nível = 4  then  "PósDoutorado"
                   else  if Nível = 5  then  "LivreDocência";
```

```
Classe ProduçãoCientifTecnológica
 Atributos:
     TotalProduçãoCT: vetor[1..13] inteiro;
 Constituintes:
     ArtigosPeriódicos:  CONJ  ArtigoPeriódico ORDENADO DESC
                                  AnoProd,      Relevância;
     LivrosCapítulos:  CONJ   LivroCapítulo   ORDENADO  DESC
                                  AnoProd,      Relevância;
     Monografias: CONJ  Monografia  ORDENADO DESC
                                  AnoProd,      Relevância;
     TesesOrientAprovadas:  CONJ DEPENDENTE  TeseOrientAprov
                                  ORDENADO  DESC  DataTese;
     ArtigosDeDivulgação:  CONJ  ArtigoDeDivulgação ORDENADO
                             DESC   AnoProd,    Relevância;
     PublicaçõesCongressos:     CONJ     PublicacaoCongresso
                   ORDENADO  DESC   AnoProd,   Relevância;
     OutrasProduçõesCT:  CONJ OutraProduçãoCT ORDENADO  DESC
                                                  AnoProd;
     Artigos: CONJ Artigo ORDENADO DESC AnoProd;
     ProduçõesComplementCNPQ:     CONJ     PublicaçãoEOutros
                                  ORDENADO  DESC   AnoProd;
     (TesesOrientAprovadas) ÚNICO;
 Regras:
     TotalProduçãoCT[1]  := COUNT(ArtigosPeriódicos TAIS QUE
                   ArtigosPeriódicos.TipoNouE = "N");
     TotalProduçãoCT[2]  := COUNT(ArtigosPeriódicos TAIS QUE
                   ArtigosPeriódicos.TipoNouE = "E");
     TotalProduçãoCT[3]  :=  COUNT(LivrosCapítulos TAIS  QUE
                   LivrosCapítulos.TipoLouC = "L");
     TotalProduçãoCT[4]  := COUNT(LivrosCapítulos  TAIS  QUE
                   LivrosCapítulos.TipoLouC = "C");
     TotalProduçãoCT[5]  := COUNT(TesesOrientAprovadas  TAIS
                   QUE  TesesOrientAprovadas.Nível = "M");
     TotalProduçãoCT[6]  := COUNT(TesesOrientAprovadas  TAIS
                   QUE  TesesOrientAprovadas.Nível = "D");
     TotalProduçãoCT[7]  := COUNT(ArtigosDeDivulgação);
     TotalProduçãoCT[8] := COUNT(PublicaçõesCongressos  TAIS
                 QUE  PublicaçõesCongressos.TipoCNPQ = "2";
     TotalProduçãoCT[9]  := COUNT(PublicaçõesCongressos TAIS
                 QUE  PublicaçõesCongressos.TipoCNPQ = "3";
     TotalProduçãoCT[10] := COUNT(PublicaçõesCongressos TAIS
                 QUE  PublicaçõesCongressos.TipoCNPQ = "4";
     TotalProduçãoCT[11] := COUNT(OutrasProduçõesCT TAIS  QUE
                   OutrasProduçõesCT.TipoCNPQ   =    "5");
     TotalProduçãoCT[12] := COUNT(OutrasProduçõesCT TAIS  QUE
                   OutrasProduçõesCT.TipoCNPQ   =    "6");
     TotalProduçãoCT[13] := COUNT(OutrasProduçõesCT TAIS  QUE
                   OutrasProduçõesCT.TipoCNPQ   =    "7");
     Artigos := ArtigosPeriódicos UNIÃO ArtigosDeDivulgação;
     ProduçõesComplementCNPQ  :=  ArtigosDeDivulgação  UNIÃO
            PublicaçõesCongressos  UNIÃO OutrasProduçõesCT;
```

```
Classe ProduçãoCientíficaTecnológica: Objeto;


Classe PublicaçãoEOutros: ProduçãoCientíficaTecnológica
 Atributos:
     AutoriaIRC: inteiro REQUERIDO;
     Titulo: string( ) REQUERIDO;
     Local: string( );
     EditoraOuVeículo: string( );
     MesProd: string(2);
     AnoProd: string(2) REQUERIDO;
     TipoCNPQ: string(1);
     Complemento: string( );
     Relevância: inteiro REQUERIDO;
     Autoria: vetor[1..3] string(1);
 Constituintes:
     SubáreasPublicação: vetor[1..2] SubáreaChave;
     Autores: CONJ Autor;
 Restrições:
     AutoriaIRC >= 1  AND  AutoriaIRC <= 3;
     Relevância >= 0  AND  Relevância < 10;
     (MesProd  >=  "01" AND MesProd <= "12")  OR
                           MesProd = NULO;
     AnoProd >= "00"  AND  AnoProd <=  AnoS(Data( ));
     (TipoCNPQ  >=  "1" AND TipoCNPQ <= "7") OR
                           TipoCNPQ = NULO;
 Regras:
     Autoria[AutoriaIRC] := "X";
 Mapeamentos:
     CoAutores: Autores --->  Autores;


Classe Publicação: PublicaçãoEOutros
 Atributos:
     NoPaginas: inteiro;
 Constituintes:
     SubáreasPublicação: vetor[1..2] SubáreaChave;


Classe Artigo: Publicação
 Atributos:
     Volume: inteiro;
     Fascículo: inteiro;
     PáginaInicial: inteiro;
     PáginaFinal: inteiro;


Classe ArtigoPeriódico: Artigo
 Atributos:
     TipoNouE: inteiro REQUERIDO;
     NomePeriódico: string( ) REQUERIDO;
     NouE: vetor[1..2] string(1);
 Restrições:
     TipoNouE = 1  OR  TipoNouE = 2;
 Regras:
     NouE[TipoNouE] := "X";
```

```
Classe ArtigoDeDivulgação: Artigo
 Restrições:
     TipoCNPQ = "1";


Classe PublicaçãoCongresso: Publicação
 Restrições:
     TipoCNPQ = "2" OR TipoCNPQ = "3" OR TipoCNPQ = "4";


Classe TrabalhoApresentadoCongresso: PublicaçãoCongresso
 Restrições:
     TipoCNPQ = "2";


Classe ResumoPublicadoCongresso: PublicaçãoCongresso
 Restrições:
     TipoCNPQ = "3";


Classe TrabalhoPublicadoCongresso: PublicaçãoCongresso
 Restrições:
     TipoCNPQ = "4";


Classe LivroCapítulo: Publicação
 Atributos:
     TipoLouC: string(1) REQUERIDO;
     LivroOuCap: vetor[1..2] string(1);
 Restrições:
     TipoLouC = "L"  OR  TipoLouC = "C";
 Regras:
     LivroOuCap[1] := if TipoLouC = "L" then "X" else " ";
     LivroOuCap[2] := if TipoLouC = "C" then "X" else " ";


Classe Livro: LivroCapítulo
 Restrições:
     TipoLouC = "L";


Classe CapítuloLivro: LivroCapítulo
 Atributos:
     TítuloCapítulo: string ( ) REQUERIDO;
     PáginaInicial: inteiro;
     PáginaFinal: inteiro;
 Restrições:
     TipoLouC = "C";


Classe Monografia: Publicação;


Classe OutraProduçãoCT: PublicaçãoEOutros
 Restrições:
     TipoCNPQ = "5" OR TipoCNPQ = "6" OR TipoCNPQ = "7";
```

```
Classe ProdutoOuProcesso: OutraProduçãoCT
 Atributos:
     RegistroINPI: string(12);
     Estágio: inteiro;
 Restrições:
     TipoCNPQ = "5"  OR  TipoCNPQ = "6";
     (Estágio )= 1 AND Estágio (= 3) OR Estágio = NULO;

Classe Filme Científico: OutraProduçãoCT
 Restrições:
     TipoCNPQ = "7";


Classe SubáreaChave
 Atributos:
     CódigoSubárea: string(12) REQUERIDO;
     PalavrasChave: vetor[1..3]  string( );
 Restrições:
     CódigoSubárea É EM  TabelaÁreas.CódigoSubárea;


Classe TeseOrientAprov: ProduçãoCientíficaTecnológica
 Atributos:
     Nível: string(1) REQUERIDO;
     TítuloTese: string( ) REQUERIDO;
     DataTese: data REQUERIDO;
     PalavrasChave: vetor[1..3] string( );
     NívelTese: vetor[1..2] string(1);
     DescriçãoNível: string( );
 Constituintes:
     Entidade: EntidadeUnidade REQUERIDO;
     Orientador: Pessoa REQUERIDO;
     NomeAluno: Pessoa REQUERIDO;
 Restrições:
     Nível = "M"  OR  Nível = "D"  OR  Nível = "O";
 Regras:
     NívelTese[1] := if Nível = "M" then "X" else " ";
     NívelTese[2] := if Nível = "D" then "X" else " ";
     DescriçãoNível  := if Nível = "M" then "Mestrado"  else
               if Nível = "D" then "Doutorado" else " ";


Classe TabelaÁreas: ObjetoInformação
 Atributos:
     CódigoSubárea: string(12) REQUERIDO;
     DescriçãoÁreaSubárea: string( ) REQUERIDO;
     (CódigoSubárea) ÚNICO;
```

```
Classe ParticipanteCongresso
 Atributos:
     DescriçãoFunção: string( ) REQUERIDO;
 Constituintes:
     NomeCongresso: Congresso REQUERIDO;
     Participante: Pessoa REQUERIDO;
     Trabalho: PublicaçãoCongresso;
     (NomeCongresso, Participante) ÚNICO;

Classe Congresso
 Atributos:
     TítuloEvento: string( ) REQUERIDO;
     Local: string( ) REQUERIDO;
     MesCongresso: string(2) REQUERIDO;
     AnoCongresso: string(2) REQUERIDO;
     (TítuloEvento) ÚNICO;
 Constituintes:
     Participantes: CONJ  ParticipanteCongresso;
     (Participantes) ÚNICO;
 Restrições:
     MesCongresso >= "01"  AND  MesCongresso <= "12";
     AnoCongresso    >=   "00"    AND     AnoCongresso    <=
                                          AnoS(Data( ));


Classe LinguistaIdioma
 Atributos:
     Fala: inteiro REQUERIDO;
     Lê: inteiro REQUERIDO;
     Escreve: inteiro REQUERIDO;
 Constituintes:
     CódigoIdioma: Idioma REQUERIDO;
     ConheceIdioma: Pessoa REQUERIDO;
 Restrições:
     Fala  >= 1  AND  Fala  <= 4;
     Lê  >= 1  AND  Lê  <= 4;
     Escreve  >= 1  AND  Escreve  <= 4;


Classe Idioma
 Atributos:
     Código: inteiro REQUERIDO;
     Descrição: string( ) REQUERIDO;
     (Código) ÚNICO;
 Constituintes:
     Linguistas: CONJ  LinguistaIdioma;


Classe Associação: EntidadeUnidade
 Constituintes:
     Membros: CONJ  Pessoa;


Classe Conferência
 Atributos:
     NomeConferência: string( ) REQUERIDO;
     DataConferência: data REQUERIDO;
 Constituintes:
     Entidade: EntidadeUnidade REQUERIDO;
```

```
Classe PrêmioDistinção
 Atributos:
     NomePrêmio: string( ) REQUERIDO;
     DataPrêmio: data REQUERIDO;
 Constituintes:
     Entidade: EntidadeUnidade REQUERIDO;


Classe Referência
 Atributos:
     NomeReferência: string( ) REQUERIDO;
 Constituintes:
     EndereçoRefer: Endereço REQUERIDO;
```

```
MOLDE TEXTO  Curriculum Resumo PARA CurriculumDocente
ESTRUTURA:
    CaixaPag:            BlocoTit,          AreaNomeEndereço,
                AreaDadosPessoais,              AreaEducação,
                AreaAtividades,               AreaExperiência,
                AreaPublicações,              AreaReferências;
    AreaNomeEndereço:      BlocoNome,      BlocoTitEndereço,
                BlocoRuaBairro,    BlocoCEP,    BlocoCidade,
                BlocoUF;
    AreaDadosPessoais:     BlocoSubtit1,    BlocoDataNasci,
                BlocoNatural, BlocoTelefone;
    AreaEducação:      BlocoSubtit2,       AreaPósGraduação,
                BlocoSuperior,                BlocoTitCursos,
                AreaCursosProfissionais;
       AreaPósGraduação:   VETOR  [1..2] BlocoPósGraduação;
       AreaCursosProfissionais:          VETOR          [1..3]
                BlocoCursoProfissional;
    AreaAtividades:     BlocoSubtit3,     BlocoTitPesquisa,
                AreaExpPesquisa,AreaPesquisaFunçõesAtuais,
                BlocoTitEnsino,               AreaHistEnsino;
       AreaExpPesquisa: VETOR [1..4] BlocoExpPesquisa;
       AreaPesquisaFunçõesAtuais:                      LISTA
                BlocoPesquisaFunçõesAtuais;
       AreaHistEnsino: LISTA BlocoHistEnsino;
    AreaExperiência:       BlocoSubtit4,       AreaFunções;
       AreaFunções: LISTA BlocoFunção;
    AreaPublicações:      BlocoSubtit5,     BlocoTitLivros,
                AreaLivrosCapítulos,       BlocoTitArtigos,
                AreaArtigos;
       AreaLivrosCapítulos:          VETOR          [1..4]
                BlocoLivroCapítulo;
       AreaArtigos:  VETOR [1..3] BlocoArtigo;
    AreaReferências: BlocoSubtit6, AreaNomeReferências;
       AreaNomeReferências:      VETOR    [1..4]     COLUNA
                BlocoReferência;
ATRIBUTOS DE CAIXA:
    NOME CAIXA:  BlocoTit,
        ESPAÇO MARGEM SUPERIOR: 3,
        CONTEÚDO: "RESUMO DE CURRICULUM VITAE",
        ALINHAMENTO: centrado,
        RENDIÇÃO GRAFICA: negrito;

    NOME CAIXA:  AreaNomeEndereço,
        ALINHAMENTO: centrado;
    NOME CAIXA: BlocoNome,
        CONTEÚDO: "NOME: " IdPessoa.NomeCompleto;
    NOME CAIXA:  BlocoTitEndereço,
        CONTEÚDO: "ENDEREÇO PARA CORRESPONDENCIA";
    NOME CAIXA:  BlocoRuaBairro,
        CONTEÚDO:  CONCAT(Residência(IdPessoa).Rua,";",
                          Residência(IdPessoa).Bairro);
    NOME CAIXA:  BlocoCEP,
        CONTEÚDO: "CEP: " Residência(IdPessoa).CEP;
    NOME CAIXA:  BlocoCidade,
        CONTEÚDO: "Cidade: " Residência(IdPessoa).Cidade;
    NOME CAIXA:  BlocoUF,
        CONTEÚDO: "UF: " Residência(IdPessoa).UF;
```

```
NOME CAIXA:  AreaDadosPessoais,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: BlocoSubtit1,
    ESPAÇO MARGEM ESQUERDA: 1,
    ESPAÇO MARGEM SUPERIOR: 3,
    CONTEÚDO: "I   DADOS PESSOAIS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA:  BlocoDataNasci,
    CONTEÚDO:          "DATA NASCIMENTO: "
                IdPessoa.DataNascimento;
NOME CAIXA:  BlocoNatural,
    CONTEÚDO: "NATURAL: " IdPessoa.Naturalidade;
NOME CAIXA:  BlocoTelefone,
    CONTEÚDO:           "TELEFONE: "
                Residência(IdPessoa).Telefone;

NOME CAIXA:  AreaEducação,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA:  BlocoSubtit2,
    ESPAÇO MARGEM ESQUERDA: 1,
    ESPAÇO MARGEM SUPERIOR: 3,
    CONTEÚDO: "II  EDUCAÇÃO",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA:  BlocoPósGraduação,
    CONTEÚDO:
      CONCAT(EspecialidadePósGrad(PósGraduações
           (Formação)).DescriçãoNível, ":",
      EspecialidadePG(EspecialidadePósGrad
        (PósGraduações(Formação))).Especialidade, ",",
      Entidade(PósGraduações(Formação)).NomeEntidade,
      ",",AnoS(PósGraduações(Formação).DataInício),
      "-",AnoS(PósGraduações(Formação).DataObtenção));
NOME CAIXA:  BlocoSuperior,
    CONTEÚDO: "Graduação: "
      CONCAT(CursoGrad(GradMaisRelevante(Formação)).
                  Curso,",",
      Entidade(GradMaisRelevante(Formação)).
                  NomeEntidade,",",
      AnoS(GradMaisRelevante(Formação).
                  DataInício),"-",
      AnoS(GradMaisRelevante(Formação).
                  DataObtenção));
NOME CAIXA:  BlocoTitCursos,
    CONTEÚDO: "Outros Cursos Profissionais: ";
NOME CAIXA:  BlocoCursoProfissional,
    CONTEÚDO:
      CONCAT(CursosProfissionais(Formação).
                   NomeCurso, ",",
      Entidade(CursosProfissionais(Formação)).
                   NomeEntidade, ",",
      DataS(CursosProfissionais(Formação).
                   DataInício), "-",
      DataS(CursosProfissionais.(Formação).
                   DataTérmino));

NOME CAIXA:  AreaAtividades,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: BlocoSubtit3,
    ESPAÇO MARGEM SUPERIOR: 3,
```

```
    ESPAÇO MARGEM ESQUERDA: 1,
    CONTEÚDO:
      "III ATIVIDADES ACADEMICAS E CIENTIFICAS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA:  BlocoTitPesquisa,
    CONTEÚDO: "- ATIVIDADES DE PESQUISA: ";
NOME CAIXA:  BlocoExpPesquisa,
    CONTEÚDO: CONCAT(ExperiênciaPesquisa.
            EspecificaçãoAtiv,",",
      AnoS(ExperiênciaPesquisa.DataInício),"-",
      AnoS(ExperiênciaPesquisa.DataTérmino));
NOME CAIXA: BlocoPesquisaFunçõesAtuais,
    CONTEÚDO:   CONCAT(PesquisaFunçõesAtuais.
                EspecificaçãoAtiv,";",
    AnoS(PesquisaFunçõesAtuais.DataInício), "-",
    AnoS(PesquisaFunçõesAtuais.DataTérmino));
NOME CAIXA:  BlocoTitEnsino,
    CONTEÚDO: "- ATIVIDADES DE ENSINO NA UFRJ :";
NOME CAIXA:  BlocoHistEnsino,
    CONTEÚDO:
      CONCAT(DisciplinaM(HistóricoEnsino).NomeDisci,
      ",", PeriodoP(FIRST(PeriodosM(HistóricoEnsino))).
                                    AnoP,"-",
          PeriodoP(LAST(PeriodosM(HistóricoEnsino))).
                                    AnoP);

NOME CAIXA: AreaExperiência,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: BlocoSubtit4,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 1,
    CONTEÚDO: "IV  EXPERIÊNCIA PROFISSIONAL",
    RENDIÇÃO GRAFICA: NEGRITO;
NOME CAIXA: BlocoFunção,
    CONTEÚDO:
    CONCAT(TipoFunção(Funções).Nome, ",",
    Entidade(TipoFunção(Funções)).NomeEntidade, ",",
    AnoS(Funções.DataInício), "-",
    AnoS(Funções.DataTérmino));


NOME CAIXA: AreaPublicações,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: BlocoSubtit5,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 1,
    CONTEÚDO: "V   PUBLICAÇÕES RECENTES",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoTitLivros,
    CONTEÚDO: "- LIVROS OU CAPÍTULOS: ";
NOME CAIXA: BlocoLivroCapítulo,
    CONTEÚDO:
    CONCAT(LivrosCapítulos(ProduçãoCT).Título, "-",
    LivrosCapítulos(ProduçãoCT) COMO CapítuloLivro.
                        TítuloCapítulo,",",
    LivrosCapítulos(ProduçãoCT))AnoProd);
NOME CAIXA: BlocoTitArtigos,
    CONTEÚDO: "- ARTIGOS DE PERIÓDICOS:";
```

```
NOME CAIXA: BlocoArtigo,
    CONTEÚDO:
    CONCAT(ArtigosPeriódicos(ProduçãoCT).Título, ",",
    ArtigosPeriódicos(ProduçãoCT).NomePeriódico, ",",
    ArtigosPeriódicos(ProduçãoCT))AnoProd);


NOME CAIXA: AreaReferências,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: BlocoSubtit6,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 1,
    CONTEÚDO: "VI  REFERÊNCIAS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoReferência,
    CONTEÚDO: CONCAT(Referências.NomeReferência,",",
       EndereçoRefer(Referências).Telefone,";");
```

```
MOLDE TEXTO  Curriculum CEPG PARA CurriculumDocente
ESTRUTURA:
    CaixaPag:          BlocoTit,          AreaDadosPessoais,
                AreaDocsIdentif,                AreaFormação,
                AreaAtividades,                 BlocoSubtit5,
                AreaCongressos,                 BlocoSubtit6,
                AreaAssociações,  BlocoSubtit7, AreaPrêmios,
                BlocoSubtit8,               AreaConferências,
                AreaPublicações;
    AreaDadosPessoais:      BlocoSubtit1,       BlocoNome,
                BlocoFiliação,          BlocoNacionalidade,
                BlocoNatural,  BlocoDataNasci,  BlocoEstado,
                BlocoNoDepend,              BlocoResidência,
                BlocoEndereçoTrabalho;
    AreaDocsIdentif: BlocoSubtit2, BlocoCarteira, BlocoCIC,
                BlocoRegProfissional,  BlocoRegistroEmprego,
                BlocoDataAdmissão;
    AreaFormação:       BlocoSubtit3,       BlocoGraduação,
                BlocoSubtitPósGrad, AreaPósGrad;
       AreaPósGrad: LISTA  BlocoPósGraduação;
    AreaAtividades:    BlocoSubtit4,   BlocoSubTitDocência,
                BlocoSubtitCOPPE,         AreaDocênciaCOPPE,
                BlocoSubtitDocOutros,    AreaDocênciaOutros,
                AreaAtivsEnsino,              BlocoTitTeses,
                BlocoTitTesesReali,     AreaTesesRealizadas,
                BlocoTitTeseOrient,     AreaTesesOrientadas,
                BlocoTitBancasExam,  AreaBancasExaminadoras,
                BlocoTitAdmTécnica, AreaAtivsAdmTécnicas;
       AreaDocênciaCOPPE: LISTA BlocoDocênciaCOPPE;
       AreaDocênciaOutros: LISTA BlocoDocênciaOutros;
       AreaAtivsEnsino:                 BlocoTitAtivsEnsino,
                BlocoTitGraduação,          AreaEnsinoGrad,
                BlocoTitPósGradEntid,   AreaHistóricoEnsino,
                BlocoTitPósGradOutros,
                AreaEnsinoPósGradOutros;
          AreaEnsinoGrad: LISTA  BlocoEnsinoGrad;
          AreaHistóricoEnsino: LISTA  BlocoDisciplina;
          AreaEnsinoPósGradOutros:  LISTA
                BlocoEnsinoPósGradOutros;
       AreaTesesRealizadas: LISTA BlocoTeseRealizada;
       AreaTesesOrientadas: LISTA BlocoTeseOrientada;
       AreaBancasExaminadoras: LISTA BlocoBancaExaminadora;
       AreaAtivsAdmTécnicas: LISTA BlocoAtivAdmTécnica;
    AreaCongressos: LISTA BlocoCongresso;
    AreaAssociações: LISTA BlocoAssociação;
    AreaPrêmios: LISTA BlocoPrêmio;
    AreaConferências: LISTA BlocoConferência;
    AreaPublicações:      BlocoSubtit9,     BlocoTitLivros,
                AreaLivrosCapítulos,        BlocoTitMonogs,
                AreaMonografias,           BlocoTitArtigos,
                AreaArtigos;
       AreaLivrosCapítulos: LISTA BlocoLivroCapítulo;
       AreaMonografias: LISTA BlocoMonografia;
       AreaArtigos: LISTA BlocoArtigo;
ATRIBUTOS DE CAIXA:
    NOME CAIXA:  BlocoTit,
        CONTEÚDO: "CURRICULUM VITAE",
        ALINHAMENTO: centrado,
        RENDIÇÃO GRAFICA: negrito;
```

```
NOME CAIXA:  AreaDadosPessoais,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: BlocoSubtit1,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "1. DADOS PESSOAIS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoNome,
    CONTEÚDO:       "1.1. Nome Completo: "
                  IdPessoa.NomeCompleto;
NOME CAIXA: BlocoFiliação,
    CONTEÚDO:       "1.2. Filiação: "
                  IdPessoa.Filiação;
NOME CAIXA: BlocoNacionalidade,
    CONTEÚDO:       "1.3. Nacionalidade: "
                  IdPessoa.Nacionalidade;
NOME CAIXA: BlocoNatural,
    CONTEÚDO:       "1.4. Naturalidade: "
                  IdPessoa.Naturalidade;
NOME CAIXA: BlocoDataNasci,
    CONTEÚDO:       "1.5. Data Nascimento: "
                  IdPessoa.DataNascimento;
NOME CAIXA: BlocoEstado,
    CONTEÚDO:       "1.6. Estado Civil: "
                  IdPessoa.DescriçãoCivil;
NOME CAIXA: BlocoNoDepend,
    CONTEÚDO:       "1.7. Número de Dependentes: "
                  IdPessoa.NoDependentes;


NOME CAIXA: BlocoResidência,
    CONTEÚDO:       "1.8. Residência: "
      CONCAT(Residência(IdPessoa).Rua, ",",
      Residência(IdPessoa).Bairro, ",","CEP: ",
      Residência(IdPessoa).CEP, ",",
      Residência(IdPessoa).Cidade, ",", "UF: ",
      Residência(IdPessoa).UF, ",", "Telefone: ",
      Residência(IdPessoa).Telefone);
NOME CAIXA: BlocoEndereçoTrabalho,
    CONTEÚDO:       "1.9. Local de Trabalho: "
      CONCAT(EndereçoTrabalho(IdPessoa).Rua, ",",
      EndereçoTrabalho(IdPessoa).Bairro, ",","CEP: ",
      EndereçoTrabalho(IdPessoa).CEP, ",",
      EndereçoTrabalho(IdPessoa).Cidade, ",", "UF: ",
      EndereçoTrabalho(IdPessoa).UF, ",", "Telex: ",
      EndereçoTrabalho(IdPessoa).Telex,",","Telefone:",
      EndereçoTrabalho(IdPessoa).Telefone);


NOME CAIXA:  AreaDocsIdentif,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA:  BlocoSubtit2,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "2. DOCUMENTOS DE IDENTIFICAÇÃO",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA:  BlocoCarteira,
    CONTEÚDO:  "2.1. Carteira de Identidade: "
      DocsIdentificação(IdPessoa).CarteiraId;
```

```
NOME CAIXA:  BlocoCIC,
    CONTEÚDO:  "2.2. CIC: "
      DocsIdentificação(IdPessoa).CPF;
NOME CAIXA:  BlocoRegProfissional,
    CONTEÚDO:  "2.3. Registro Profissional: "
      DocsIdentificação(IdPessoa).RegistroProfissional;
NOME CAIXA:  BlocoRegEmprego,
    CONTEÚDO:  "2.4. No de Registro na UFRJ: "
            IdPessoa.NoRegistroEmprego;
NOME CAIXA:  BlocoDataAdmissão,
    CONTEÚDO:  "2.5. DataAdmissão na UFRJ: "
            DataS(IdPessoa.DataAdmissão);

NOME CAIXA: AreaFormação,
    ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: BlocoSubtit3,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "3. FORMAÇÃO E TÍTULOS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA:  BlocoGraduação,
    CONTEÚDO:  "3.1. Graduação: "
      CONCAT(CursoGrad(GradMaisRelevante(Formação)).
                  Curso,",",
      Entidade(GradMaisRelevante(Formação)).
                  NomeEntidade,",",
      AnoS(GradMaisRelevante(Formação).
                  DataInício),"-",
      AnoS(GradMaisRelevante(Formação).
                  DataObtenção));
NOME CAIXA: BlocoSubtitPósGrad,
    CONTEÚDO: "3.2. Pós Graduação: ";
NOME CAIXA: AreaPósGrad,
    ESPAÇO MARGEM ESQUERDA: 11;
NOME CAIXA: BlocoPósGraduação,
      CONCAT(EspecialidadePósGrad(PósGraduações
           (Formação)).DescriçãoNível, ":",
      EspecialidadePG(EspecialidadePósGrad
        (PósGraduações(Formação))).Especialidade, ",",
      Entidade(PósGraduações(Formação)).NomeEntidade,
      ",",AnoS(PósGraduações(Formação).DataInício),
      "-",AnoS(PósGraduações(Formação).DataObtenção));


NOME CAIXA: AreaAtividades,
    ESPAÇO MARGEM ESQUERDA: 11;
NOME CAIXA: BlocoSubtit4,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "4. ATIVIDADES PROFISSIONAIS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoSubtitDocência,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO: "4.1. CARREIRA DE MAGISTÉRIO";
NOME CAIXA: BlocoSubtitCOPPE,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO: "4.1.1. Na COPPE: ";
NOME CAIXA: BlocoDocênciaCOPPE,
```

```
    CONTEÚDO:     "- "
      CONCAT(TipoFunção(DocênciaCOPPE).Nome, ",",
      AnoS(DocênciaCOPPE.DataInício));
NOME CAIXA: BlocoSubtitDocOutros,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO: "4.1.2. Externa à COPPE: ";
NOME CAIXA: BlocoDocênciaOutros,
    CONTEÚDO:     "- "
      CONCAT(TipoFunção(DocênciaOutros).Nome, ",",
      Entidade(TipoFunção(DocênciaOutros)).
                                  NomeEntidade, ",",
      AnoS(DocênciaOutros.DataInício));
NOME CAIXA: BlocoTitAtivsEnsino,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO: "4.2. ATIVIDADES DE MAGISTÉRIO";
NOME CAIXA: BlocoTitGraduação,
    CONTEÚDO: "a) GRADUAÇÃO: ";
NOME CAIXA: BlocoEnsinoGrad,
NOME CAIXA: BlocoTitPósGradEntid,
    CONTEÚDO: "b) PÓS GRADUAÇÃO NA UFRJ: ";
NOME CAIXA: BlocoDisciplina,
    CONTEÚDO:     "- "
      CONCAT(DisciplinaM(HistóricoEnsino).NomeDisci,
      ",", PeriodoP(FIRST(PeriodosM(HistóricoEnsino))).
                              AnoP, "-",
      PeriodoP(LAST(PeriodosM(HistóricoEnsino))).AnoP);
NOME CAIXA: BlocoTitPósGradOutros,
    CONTEÚDO: "c) PÓS GRADUAÇÃO EXTERNA À UFRJ";
NOME CAIXA: BlocoEnsinoPósGradOutros,
    CONTEÚDO:     "- "
      CONCAT(ExpEnsinoPósGrad.EspecificaçãoAtiv, ",",
      Entidade(TipoFunção(FunçãoDeAtiv
            (ExpEnsinoPósGrad))).NomeEntidade, ",",
      AnoS(ExpEnsinoPósGrad.DataInício), "-",
      AnoS(ExpEnsinoPósGrad.DataTérmino));
NOME CAIXA: BlocoTitTeses,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO: "4.3. TESES";
NOME CAIXA: BlocoTitTesesReali,
    CONTEÚDO: "a) REALIZADAS: ";


NOME CAIXA: BlocoTeseRealizada,
    CONTEÚDO:     "- "
      CONCAT(TesePósGraduação(TesesMestDout(Formação)).
                    TituloTese, ",",
      EspecialidadePósGrad(TesesMestDout(Formação)).
                    DescriçãoNível, ":",
      EspecialidadePG(EspecialidadePósGrad
        (TesesMestDout(Formação))).Especialidade,",",
      Entidade(TesesMestDout(Formação)).NomeEntidade,
      ",", AnoS(TesesMestDout(Formação).DataObtenção));
NOME CAIXA: BlocoTitTeseOrient,
    CONTEÚDO: "b) ORIENTADAS: ";
NOME CAIXA: BlocoTeseOrientada,
    CONTEÚDO:     "- "
      CONCAT(TesesOrientAprovadas(ProduçãoCT).DataTese,
      ",",  TesesOrientAprovadas(ProduçãoCT).
                    DescriçãoNível, ",",
```

```
      TesesOrientAprovadas(ProduçãoCT).TituloTese, ",",
      NomeAluno(TesesOrientAprovadas(ProduçãoCT)).
                        NomeAbreviado);
NOME CAIXA: BlocoTitBancasExam,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO:
      "4.4. PARTICIPAÇÃO EM BANCAS EXAMINADORAS";
NOME CAIXA: BlocoBancaExaminadora,
    CONTEÚDO:     "- "
      CONCAT(BancasExaminadoras.EspecificaçãoAtiv, ",",
      Entidade(TipoFunção(FunçãoDeAtiv
            (BancasExaminadoras))).NomeEntidade, ",",
      AnoS(BancasExaminadoras.DataTérmino));
NOME CAIXA: BlocoTitAdmTécnica,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO:
      "4.5.   ATIVIDADES  TÉCNICAS,  ADMINISTRATIVAS  E
              MISSÕES NO EXTERIOR";
NOME CAIXA: BlocoAtivAdmTécnica,
    CONTEÚDO:     "- "
      CONCAT(TipoFunção(FunçãoDeAtiv
            (AtivsAdmTécnicas)).Nome, ",",
      AtivsAdmTécnicas.EspecificaçãoAtiv, ",",
      Entidade(TipoFunção(FunçãoDeAtiv
            (AtivsAdmTécnicas))).NomeEntidade, ",",
      AnoS(AtivsAdmTécnicas.DataInício), "-",
      AnoS(AtivsAdmTécnicas.DataTérmino));


NOME CAIXA: BlocoSubtit5,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "5. PARTICIPAÇÃO EM CONGRESSOS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoCongresso,
    ESPAÇO  MARGEM ESQUERDA: 5,
    CONTEÚDO:     "- "
      CONCAT(NomeCongresso(Congressos).TituloEvento,
      ",", NomeCongresso(Congressos).Local, ",",
      NomeCongresso(Congressos).AnoCongresso, ",",
      Congressos.Descrição, "-",
      TrabalhoCongresso(Congressos).Titulo);


NOME CAIXA: BlocoSubtit6,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "6. ASSOCIAÇÕES CIENTÍFICAS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoAssociação,
    ESPAÇO  MARGEM ESQUERDA: 5,
    CONTEÚDO:     "- "
      CONCAT(Entidade(TipoFunção(Associações)).
                        NomeEntidade, ",",
      Entidade(TipoFunção(Associações)).PaisEntidade);
```

```
NOME CAIXA: BlocoSubtit7,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "7. DISTINÇÕES, PRÊMIOS E CONDECORAÇÕES",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoPrêmio,
    ESPAÇO  MARGEM ESQUERDA: 5,
     CONTEÚDO:      "- "
      CONCAT(PrêmiosDistinções.NomePrêmio, ",",
      Entidade(PrêmiosDistinções).NomeEntidade, ",",
      AnoS(PrêmiosDistinções.DataPrêmio));

NOME CAIXA: BlocoSubtit8,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "8. CONFERÊNCIAS ",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoConferência,
    ESPAÇO  MARGEM ESQUERDA: 5,
    CONTEÚDO:      "- "
      CONCAT(Conferências.NomeConferência, ",",
      Entidade(Conferências).NomeEntidade, ",",
      AnoS(Conferências.DataConferência));

NOME CAIXA: AreaPublicações,
    ESPAÇO MARGEM ESQUERDA: 11;
NOME CAIXA: BlocoSubtit9,
    ESPAÇO MARGEM SUPERIOR: 3,
    ESPAÇO MARGEM ESQUERDA: 2,
    CONTEÚDO: "9. TRABALHOS PUBLICADOS",
    RENDIÇÃO GRAFICA: negrito;
NOME CAIXA: BlocoTitLivros,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO: "9.1. LIVROS E CAPÍTULOS";
NOME CAIXA: BlocoLivroCapítulo,
    CONTEÚDO:     "- "
      CONCAT(LivrosCapítulos(ProduçãoCT).Título, "-",
      LivrosCapítulos(ProduçãoCT)   COMO
             CapítuloLivro.TítuloCapítulo, ",",
      LivrosCapítulos(ProduçãoCT).AnoProd);
NOME CAIXA: BlocoTitMonogs,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO: "9.2. MONOGRAFIAS";
NOME CAIXA: BlocoMonografia,
    CONTEÚDO:      "- "
      CONCAT(Monografias(ProduçãoCT).Título, ",",
      Monografias(ProduçãoCT).AnoProd);
NOME CAIXA: BlocoTitArtigos,
    ESPAÇO MARGEM ESQUERDA: 5,
    CONTEÚDO:
      "9.3. ARTIGOS ORIGINAIS E REVISÃO CRÍTICA";
NOME CAIXA: BlocoArtigo,
    CONTEÚDO:      "- "
      CONCAT(Artigos(ProduçãoCT).Título, ",",
      Artigos(ProduçãoCT) COMO ArtigoPeriódico.
                           NomePeriódico, "-",
      Artigos(ProduçãoCT).EditoraOuVeículo, ",",
      Artigos(ProduçãoCT).AnoProd, ",",
      Artigos(ProduçãoCT).NoPaginas);
```

```
MOLDE TEXTO  CurriculumCNPQ  PARA  Curriculum Docente
ESTRUTURA:
 CaixaPag:          AreaIdentif,          AreaAtivPrincipal,
         AreaOutrasAtivAtuais,                  AreaAtuação,
         AreaAtivsAnteriores,   AreaFormação,   AreaPrêmios,
         AreaIdiomas,                AreaIndicadoresProduc,
         AreaArtigosPeriódicos,        AreaLivrosCapítulos,
         AreaTesesOrient,           AreaProdCTComplementar;
 AreaIdentif:   BlocoNome,   AreaDocsIdentif,  AreaIdentif1,
         AreaEndereço1, AreaEndereço2;
   AreaDocsIdentif:    COLUNA    BlocoCPF,    BlocoCarteira,
         BlocoOrgão, BlocoUF, BlocoData, BlocoPassaporte;
   AreaIdentif1:     COLUNA    BlocoDataNasci,    BlocoSexo,
         BlocoNacionalidade;
   AreaEndereço1: COLUNA BlocoRua, BlocoBairro;
   AreaEndereço2:    COLUNA   BlocoCidade,    BlocoUFCidade,
         BlocoCEP, BlocoTelefoneResidência;
 AreaAtivPrincipal:     BlocoUnidade,     BlocoOrgãoUnidade,
         AreaEntidade,    AreaEndereçoU1,    AreaEndereçoU2,
         AreaFunçãoPrincipal, AreaDescriçãoTarefas;
   AreaEntidade: COLUNA BlocoNomeEntid, BlocoDataAdmissão;
   AreaEndereçoU1: COLUNA BlocoRuaEntid, BlocoBairroEntid;
   AreaEndereçoU2:   COLUNA  BlocoCidadeEntid,   BlocoUFEnt,
         BlocoCEPEntid, BlocoTelexEntid, BlocoFoneEntid;
   AreaFunçãoPrincipal:     COLUNA     BlocoFunçãoPrincipal,
         BlocoRegimeTrabalho;
   AreaDescriçãoTarefas:    BlocoAdminist,    BlocoPesquisa,
         BlocoGrad, AreaEnsinoPósGrad, BlocoServiçoTécnico;
      AreaEnsinoPósGrad: LISTA COLUNA BlocoDisciplina;
 AreaOutrasAtivAtuais: VETOR [1..3] AreaFunçãoAtiv;
   AreaFunçãoAtiv:   BlocoEntid1,   AreaEntid2,  AreaFunção,
         BlocoEspecificação;
   AreaEntid2:    COLUNA   BlocoSiglaEntid,    BlocoUFEntid,
         BlocoPaisEntid;
   AreaFunção:     COLUNA     BlocoFunção,     BlocoCodAtiv,
         BlocoDataAtiv;
 AreaAtuação: VETOR[1..5] AreaConhecimento;
   AreaConhecimento:           COLUNA           BlocoSubárea,
         BlocoEspecialidade;
 AreaAtivsAnteriores: VETOR[1..3]   AreaFunçãoAnterior;
   AreaFunçãoAnterior:     BlocoEntidAnt1,    AreaEntidAnt2,
         AreaFunçãoAnt, BlocoEspecAnt;
   AreaEntidAnt2:    COLUNA    BlocoSiglaAnt,    BlocoUFAnt,
         BlocoPaisAnt;
   AreaFunçãoAnt:    COLUNA   BlocoFunçãoAnt,   BlocoCodAnt,
         BlocoDataIniAnt, BlocoDataTermAnt;
 AreaFormação:        AreaGraduação,       AreaEspecialização,
         AreaMestrado, AreaDoutorado, AreaOutrosPósGrad;
   AreaGraduação: BlocoCurso, BlocoEntidGrad, AreaDadosGrad;
      AreaDadosGrad:   COLUNA  BlocoPaisGrad,   BlocoUFGrad,
         BlocoAnoIniGrad, BlocoAnoObtGrad;
   AreaEspecialização:   AreaSubáreaEspec,  BlocoEntidEspec,
               AreaDadosEspec;
      AreaSubáreaEspec:          COLUNA         BlocoCodEspec,
         BlocoDescriEspec;
      AreaDadosEspec:  COLUNA BlocoPaisEspec,  BlocoUFEspec,
         BlocoAnoIniEspec, BlocoAnoObtEspec;
   AreaMestrado:       AreaSubáreaMest,      BlocoEntidMest,
         AreaDadosMest,  BlocoOrientadorMest, BlocoTeseMest,
```

```
        AreaPalavrasMest;
      AreaSubáreaMest: COLUNA BlocoCodMest, BlocoDescriMest;
      AreaDadosMest:  COLUNA BlocoPaisMest, BlocoAnoIniMest,
        BlocoAnoObtMest;
      AreaPalavrasMest: VETOR[1..3] BlocoPalavraMest;
  AreaDoutorado:      AreaSubáreaDout,      BlocoEntidDout,
        AreaDadosDout,  BlocoOrientadorDout, BlocoTeseDout,
        AreaPalavrasDout;
      AreaSubáreaDout: COLUNA BlocoCodDout, BlocoDescriDout;
      AreaDadosDout:  COLUNA BlocoPaisDout, BlocoAnoIniDout,
        BlocoAnoObtDout;
      AreaPalavrasDout: VETOR[1..3] BlocoPalavraDout;
  AreaOutrosPósGrad: VETOR[1..2]  AreaPósGrad;
      AreaPósGrad:        AreaSubáreaPG,        BlocoEntidPG,
        BlocoNívelEspec,   AreaDadosPG,   AreaOrientadorPG,
        AreaTesePG, AreaPalavrasPG;
       AreaSubáreaPG: COLUNA BlocoCodPG, BlocoDescriPG;
       AreaDadosPG:   COLUNA  BlocoNívelMest,   BlocoPaisPG,
        BlocoAnoIniPG, BlocoAnoObtPG;
       AreaOrientadorPG:        COLUNA       BlocoNívelDout,
        BlocoOrientadorPG;
       AreaTesePG: COLUNA AreaNiveis, BlocoTesePG;
        AreaNiveis: BlocoNívelPósDout, BlocoNívelLivreDoc;
       AreaPalavrasPG: VETOR[1..3] BlocoPalavraPG;
AreaPrêmios: VETOR[1..2] AreaPrêmio;
  AreaPrêmio: BlocoTítuloPrêmio, AreaEntidPrêmio;
     AreaEntidPrêmio:        COLUNA       BlocoEntidPrêmio,
        BlocoAnoPrêmio;
AreaIdiomas: VETOR[1..4] BlocoIdiomas;
AreaIndicadoresProduc: VETOR[1..17] BlocoIndicadorProduc;
AreaArtigosPeriódicos: VETOR[1..19] AreaArtigos;
  AreaArtigos:      AreaTipoTítulo,      AreaAutoresArtigo,
        BlocoPeriódico,                  AreaEditoraArtigo,
        AreaSubáreasArtigo;
    AreaTipoTítulo:         COLUNA         BlocoTipoArtigo,
        BlocoAutoriaArtigo, BlocoTítuloArtigo;
    AreaAutoresArtigo: LISTA COLUNA BlocoAutorArtigo;
    AreaEditoraArtigo:  COLUNA  BlocoEditora,  BlocoVolume,
        BlocoFascículo,     BlocoPagIni,     BlocoPagFinal,
        BlocoAnoArtigo;
    AreaSubáreasArtigo: VETOR[1..2] COLUNA AreaCódPalavras;
      AreaCódPalavras: BlocoCódArtigo, AreaPalavrasArtigo;
        AreaPalavrasArtigo: VETOR[1..3] BlocoPalavraArtigo;
AreaLivrosCapítulos: VETOR[1..9] AreaLivros;
  AreaLivros:       AreaTítuloLivro,      AreaAutoresLivro,
        AreaEditoraLivro,              AreaAutoresCapítulo,
        AreaSubáreasLC;
    AreaTítuloLivro:    COLUNA   BlocoLC,   BlocoAutoriaLC,
        BlocoTítuloLivro, BlocoPágs;
    AreaAutoresLivro: LISTA COLUNA BlocoAutoresLivro;
    AreaEditoraLivro:       COLUNA       BlocoLocalEditora,
        BlocoEditoraLivro,                   BlocoAnoLivro,
        BlocoTítuloCapítulo,                BlocoPagIniCap,
        BlocoPagFinalCap;
    AreaAutoresCapítulo: LISTA COLUNA BlocoAutoresCapítulo;
    AreaSubáreasLC: VETOR[1..2] COLUNA AreaCódLC;
      AreaCódLC: BlocoCódLivro, AreaPalavrasLC;
        AreaPalavrasLC: VETOR[1..3] BlocoPalavraLC;
AreaTesesOrient: VETOR[1..14] AreaTesesAprov;
```

```
   AreaTesesAprov:     AreaAlunoTese,     BlocoTitTeseAprov,
        AreaEntidTese;
     AreaAlunoTese:        COLUNA       BlocoNívelTeseAprov,
        BlocoNomeAluno;
     AreaEntidTese:          COLUNA          BlocoEntidTese,
        BlocoAnoTeseAprov;
 AreaProdCTComplementar: VETOR[1..14] AreaProdComplement;
   AreaProdComplement:     AreaTipoProd,    BlocoEventoProd,
        AreaDadosProd;
     AreaTipoProd:  COLUNA BlocoTipoProd,  BlocoAutoriaProd,
        BlocoTítuloProd;
     AreaDadosProd:   COLUNA  BlocoLocalProd,  BlocoMesProd,
        BlocoAnoProd,      BlocoRegProd,      BlocoEstágio,
        BlocoComplemento;
ATRIBUTOS DE CAIXA:
 NOME CAIXA: AreaIdentif,
   ESPAÇO MARGEM ESQUERDA: 11,
   ESPAÇO MARGEM DIREITA: 3,
   ESPAÇO ENTRE LINHAS: dobre;
 NOME CAIXA: BlocoNome,
   ESPAÇO MARGEM SUPERIOR: 16,
   TAMANHO CAIXA: 66,
   CONTEÚDO: IdPessoa.NomeCompleto;
 NOME CAIXA: BlocoCPF,
   TAMANHO CAIXA: 20,
   CONTEÚDO: DocsIdentificação(IdPessoa).CPF;
 NOME CAIXA: BlocoCarteira,
   TAMANHO CAIXA: 14,
   CONTEÚDO: DocsIdentificação(IdPessoa).CarteiraId;
 NOME CAIXA: BlocoOrgão,
   TAMANHO CAIXA: 6,
   CONTEÚDO: DocsIdentificação(IdPessoa).OrgãoEmissorId;
 NOME CAIXA: BlocoUF,
   TAMANHO CAIXA: 3,
   CONTEÚDO: DocsIdentificação(IdPessoa).UFCarteira;
 NOME CAIXA: BlocoData,
   TAMANHO CAIXA: 8,
   CONTEÚDO:
     DataS(DocsIdentificação(IdPessoa).DataCarteira);
 NOME CAIXA: BlocoPassaporte,
   TAMANHO CAIXA: 15,
   CONTEÚDO: DocsIdentificação(IdPessoa).Passaporte;
 NOME CAIXA: BlocoDataNasci,
   TAMANHO CAIXA: 16,
   CONTEÚDO: CONCAT(" ",DiaS(IdPessoa.DataNascimento), "  ",
                        MesS(IdPessoa.DataNascimento), "  ",
                        AnoS(IdPessoa.DataNascimento));
 NOME CAIXA: BlocoSexo,
   TAMANHO CAIXA: 11,
   CONTEÚDO: CONCAT(IdPessoa.SexoCNPQ[1], "      ",
                    IdPessoa.SexoCNPQ[2]));
 NOME CAIXA: BlocoNacionalidade,
   CONTEÚDO: IdPessoa.Nacionalidade;
 NOME CAIXA: BlocoRua,
   TAMANHO CAIXA: 48,
   CONTEÚDO: Residência(IdPessoa).Rua;
 NOME CAIXA: BlocoBairro,
   TAMANHO CAIXA: 18,
   CONTEÚDO: Residência(IdPessoa).Bairro;
```

```
NOME CAIXA: BlocoCidade,
  TAMANHO CAIXA: 31,
  CONTEÚDO: Residência(IdPessoa).Cidade;
NOME CAIXA: BlocoUFCidade,
  TAMANHO CAIXA: 4,
  CONTEÚDO: " " Residência(IdPessoa).UF;
NOME CAIXA: BlocoCEP,
  TAMANHO CAIXA: 12,
  CONTEÚDO: Residência(IdPessoa).CEP;
NOME CAIXA: BlocoTelefoneResidência,
  TAMANHO CAIXA: 19,
  CONTEÚDO: Residência(IdPessoa).Telefone;

NOME CAIXA: AreaAtivPrincipal,
  ESPAÇO ENTRE LINHAS: dobre,
  ESPAÇO MARGEM ESQUERDA: 11,
  ESPAÇO MARGEM DIREITA: 3;
NOME CAIXA: BlocoUnidade,
  ESPAÇO MARGEM SUPERIOR: 3,
  TAMANHO CAIXA: 47,
  CONTEÚDO: UnidadeEmpregoDo(IdPessoa).NomeUnidade;
NOME CAIXA: BlocoOrgãoUnidade,
  TAMANHO CAIXA: 66,
  CONTEÚDO: UnidadeEmpregoDo(IdPessoa).NomeOrgão;
NOME CAIXA: BlocoNomeEntid,
  TAMANHO CAIXA: 53,
  CONTEÚDO: UnidadeEmpregoDo(IdPessoa).NomeEntidade;
NOME CAIXA: BlocoDataAdmissão,
  TAMANHO CAIXA: 9,
  CONTEÚDO: CONCAT("  ", MesS(IdPessoa.DataAdmissão),
                   "   ", AnoS(IdPessoa.DataAdmissão));
NOME CAIXA: BlocoRuaEntid,
  TAMANHO CAIXA: 47,
  CONTEÚDO: EndereçoTrabalho(IdPessoa).Rua;
NOME CAIXA: BlocoBairroEntid,
  TAMANHO CAIXA: 19,
  CONTEÚDO: EndereçoTrabalho(IdPessoa).Bairro;
NOME CAIXA: BlocoCidadeEntid,
  TAMANHO CAIXA: 22,
  CONTEÚDO: EndereçoTrabalho(IdPessoa).Cidade;
NOME CAIXA: BlocoUFEnt,
  TAMANHO CAIXA: 6,
  CONTEÚDO: " " EndereçoTrabalho(IdPessoa).UF;
NOME CAIXA: BlocoCEPEntid,
  TAMANHO CAIXA: 11,
  CONTEÚDO: EndereçoTrabalho(IdPessoa).CEP;
NOME CAIXA: BlocoTelexEntid,
  TAMANHO CAIXA: 7,
  CONTEÚDO: EndereçoTrabalho(IdPessoa).Telex;
NOME CAIXA: BlocoFoneEntid,
  TAMANHO CAIXA: 20,
  CONTEÚDO: EndereçoTrabalho(IdPessoa).Telefone;
NOME CAIXA: BlocoFunçãoPrincipal,
  TAMANHO CAIXA: 37,
  CONTEÚDO: TipoFunção(FunçãoEntid(IdPessoa)).Nome;
NOME CAIXA: BlocoRegimeTrabalho,
  TAMANHO CAIXA: 29,
  CONTEÚDO: CONCAT("    ",   RegimeCNPQ[1],  "          ",
            RegimeCNPQ[2], "          ", RegimeCNPQ[3]);
```

```
NOME CAIXA: AreaDescriçãoTarefas,
  ESPAÇO MARGEM ESQUERDA: 23;
NOME CAIXA: BlocoAdminist,
  TAMANHO CAIXA: 54 7,
  ESPAÇO MARGEM SUPERIOR: 4,
  CONTEÚDO: DescriçãoTarefas(IdPessoa).DireçãoAdminist,
  DADOS TEXTO;
NOME CAIXA: BlocoPesquisa,
  TAMANHO CAIXA: 54 7,
  CONTEÚDO:
    DescriçãoTarefas(IdPessoa).LinhasPesquisaDesenvol,
  DADOS TEXTO;
NOME CAIXA: BlocoGrad,
  TAMANHO CAIXA: 54 7;
NOME CAIXA: AreaEnsinoPósGrad,
  TAMANHO CAIXA: 54 7;
NOME CAIXA: BlocoDisciplina,
  TAMANHO CAIXA: 30,
  CONTEÚDO:
  DisciplinaP(DisciplinasEnsinoAtuais(IdPessoa)).NomeDisci;
NOME CAIXA: BlocoServiçoTécnico,
  TAMANHO CAIXA: 54 7,
  CONTEÚDO: DescriçãoTarefas(IdPessoa).ServiçoTécnicoEspec,
  DADOS TEXTO;

NOME CAIXA: AreaOutrasAtivAtuais,
  ESPAÇO MARGEM ESQUERDA: 9,
  ESPAÇO MARGEM DIREITA: 10,
  ESPAÇO MARGEM SUPERIOR: 3;
NOME CAIXA: AreaFunçãoAtiv,
  ESPAÇO MARGEM SUPERIOR: 1;
NOME CAIXA: BlocoEntid1,
  TAMANHO CAIXA: 62,
  CONTEÚDO: Entidade(TipoFunção(FunçãoDe(AtivsCNPQ
                    (IdPessoa)))).NomeEntidade;
NOME CAIXA: BlocoSiglaEntid,
  TAMANHO CAIXA: 21,
  CONTEÚDO: Entidade(TipoFunção(FunçãoDe(AtivsCNPQ
                    (IdPessoa)))).SiglaEntidade;
NOME CAIXA: BlocoUFEntid,
  TAMANHO CAIXA: 3,
  CONTEÚDO: Entidade(TipoFunção(FunçãoDe(AtivsCNPQ
                    (IdPessoa)))).UFEntidade;
NOME CAIXA: BlocoPaisEntid,
  TAMANHO CAIXA: 22,
  CONTEÚDO: Entidade(TipoFunção(FunçãoDe(AtivsCNPQ
                    (IdPessoa)))).PaisEntidade;
NOME CAIXA: BlocoFunção,
  TAMANHO CAIXA: 53,
  CONTEÚDO: TipoFunção(FunçãoDe(AtivsCNPQ
                    (IdPessoa))).Nome;
NOME CAIXA: BlocoCodAtiv,
  TAMANHO CAIXA: 3,
  CONTEÚDO: Ativ(AtivsCNPQ(IdPessoa)).Código;
NOME CAIXA: BlocoDataAtiv,
  CONTEÚDO:
    CONCAT(MesS(Ativ(AtivsCNPQ(IdPessoa)).DataInício),
     "  ", AnoS(Ativ(AtivsCNPQ(IdPessoa)).DataInício));
NOME CAIXA: BlocoEspecificação,
```

```
   TAMANHO CAIXA: 62,
   CONTEÚDO: Ativ(AtivsCNPQ(IdPessoa)).EspecificaçãoAtiv;


NOME CAIXA: AreaAtuação,
   ESPAÇO MARGEM SUPERIOR: 2,
   ESPAÇO MARGEM ESQUERDA: 5;
NOME CAIXA: AreaConhecimento,
   ESPAÇO MARGEM SUPERIOR: 1;
NOME CAIXA: BlocoSubárea,
   TAMANHO CAIXA: 14,
   CONTEÚDO: AreaSubárea(DescEspecialidade(ÁreasAtuação)).
                        CódigoSubárea;
NOME CAIXA: BlocoEspecialidade,
   TAMANHO CAIXA: 36,
   CONTEÚDO: DescEspecialidade(ÁreasAtuação).
                        Especialidade;

NOME CAIXA: BlocoAreaAtivsAnteriores,
   ESPAÇO MARGEM SUPERIOR: 4,
   ESPAÇO MARGEM ESQUERDA: 8,
   ESPAÇO MARGEM DIREITA: 10;
NOME CAIXA: AreaFunçãoAnterior,
   ESPAÇO MARGEM SUPERIOR: 1;
NOME CAIXA: BlocoEntidAnt1,
   TAMANHO CAIXA: 62,
   CONTEÚDO: Entidade(TipoFunção(FunçãoDe
                        (ExpAtivsCNPQ))).NomeEntidade;
NOME CAIXA: BlocoSiglaAnt,
   TAMANHO CAIXA: 21,
   CONTEÚDO: Entidade(TipoFunção(FunçãoDe
                        (ExpAtivsCNPQ))).SiglaEntidade;
NOME CAIXA: BlocoUFAnt,
   TAMANHO CAIXA: 3,
   CONTEÚDO: Entidade(TipoFunção(FunçãoDe
                        (ExpAtivsCNPQ))).UFEntidade;
NOME CAIXA: BlocoPaisAnt,
   TAMANHO CAIXA: 22,
   CONTEÚDO: Entidade(TipoFunção(FunçãoDe
                        (ExpAtivsCNPQ))).PaisEntidade;
NOME CAIXA: BlocoFunçãoAnt,
   TAMANHO CAIXA: 45,
   CONTEÚDO: TipoFunção(FunçãoDe
                        (ExpAtivsCNPQ)).Nome;
NOME CAIXA: BlocoCodAnt,
   TAMANHO CAIXA: 3,
   CONTEÚDO: Ativ(ExpAtivsCNPQ).Código;
NOME CAIXA: BlocoDataIniAnt,
   TAMANHO CAIXA: 7,
   CONTEÚDO: CONCAT(MesS(Ativ(ExpAtivsCNPQ).DataInício),
              " ", AnoS(Ativ(ExpAtivsCNPQ).DataInício));
NOME CAIXA: BlocoDataTermAnt,
   TAMANHO CAIXA: 7,
   CONTEÚDO: CONCAT(MesS(Ativ(ExpAtivsCNPQ).DataTérmino),
              " ", AnoS(Ativ(ExpAtivsCNPQ).DataTérmino));
NOME CAIXA: BlocoEspecAnt,
   TAMANHO CAIXA: 62,
   CONTEÚDO: Ativ(ExpAtivsCNPQ).EspecificaçãoAtiv;
```

```
NOME CAIXA: AreaFormação,
  ESPAÇO MARGEM ESQUERDA: 20,
  ESPAÇO MARGEM DIREITA: 4;
NOME CAIXA: AreaGraduação,
  ESPAÇO ENTRE LINHAS: dobre;
NOME CAIXA: BlocoCurso,
  TAMANHO CAIXA: 41,
  CONTEÚDO: CursoGrad(GradMaisRelevante(Formação)).Curso;
NOME CAIXA: BlocoEntidGrad,
  TAMANHO CAIXA: 56,
  CONTEÚDO: Entidade(GradMaisRelevante(Formação)).
                        NomeEntidade;
NOME CAIXA: BlocoPaisGrad,
  TAMANHO CAIXA: 22,
  CONTEÚDO: Entidade(GradMaisRelevante(Formação)).
                        PaisEntidade;
NOME CAIXA: BlocoUFGrad,
  TAMANHO CAIXA: 5,
  CONTEÚDO: Entidade(GradMaisRelevante(Formação)).
                        UFEntidade;
NOME CAIXA: BlocoAnoIniGrad,
  TAMANHO CAIXA: 6,
  CONTEÚDO: "  " AnoS(GradMaisRelevante(Formação).
                        DataInício);
NOME CAIXA: BlocoAnoObtGrad,
  TAMANHO CAIXA: 6,
  CONTEÚDO: "  " AnoS(GradMaisRelevante(Formação).
                        DataObtenção);
NOME CAIXA: BlocoCodEspec,
  TAMANHO CAIXA: 12,
  CONTEÚDO:
    AreaSubárea(EspecialidadePG(EspecialidadePósGrad
      (EspecMaisRelevante(Formação)))).CódigoSubárea;
NOME CAIXA: BlocoDescriEspec,
  TAMANHO CAIXA: 30,
  CONTEÚDO: EspecialidadePG(EspecialidadePósGrad
       (EspecMaisRelevante(Formação))).Especialidade;
NOME CAIXA: BlocoEntidEspec,
  TAMANHO CAIXA: 56,
  CONTEÚDO:
    Entidade(EspecMaisRelevante(Formação)).NomeEntidade;
NOME CAIXA: BlocoPaisEspec,
  TAMANHO CAIXA: 22,
  CONTEÚDO:
    Entidade(EspecMaisRelevante(Formação)).PaisEntidade;
NOME CAIXA: BlocoUFEspec,
  TAMANHO CAIXA: 5,
  CONTEÚDO:
    Entidade(EspecMaisRelevante(Formação)).UFEntidade;
NOME CAIXA: BlocoAnoIniEspec,
  TAMANHO CAIXA: 6,
  CONTEÚDO:
    "  " AnoS(EspecMaisRelevante(Formação).DataInício);
NOME CAIXA: BlocoAnoObtEspec,
  TAMANHO CAIXA: 6,
  CONTEÚDO:
    "  " AnoS(EspecMaisRelevante(Formação).DataObtenção);
NOME CAIXA: AreaSubáreaMest,
  ESPAÇO ENTRE LINHAS: dobre;
```

```
NOME CAIXA: BlocoCodMest,
  TAMANHO CAIXA: 12,
  CONTEÚDO:
    AreaSubárea(EspecialidadePG(EspecialidadePósGrad
     (MestradoMaisRelevante(Formação)))).CódigoSubárea;
NOME CAIXA: BlocoDescriMest,
  TAMANHO CAIXA: 30,
  CONTEÚDO: EspecialidadePG(EspecialidadePósGrad
      (MestradoMaisRelevante(Formação))).Especialidade;
NOME CAIXA: BlocoEntidMest,
  TAMANHO CAIXA: 56,
  CONTEÚDO: Entidade(MestradoMaisRelevante(Formação)).
                                  NomeEntidade;
NOME CAIXA: AreaDadosMest,
  ESPAÇO ENTRE LINHAS: dobre;
NOME CAIXA: BlocoPaisMest,
  TAMANHO CAIXA: 27,
  CONTEÚDO: Entidade(MestradoMaisRelevante(Formação)).
                                  PaisEntidade;
NOME CAIXA: BlocoAnoIniMest,
  TAMANHO CAIXA: 6,
  CONTEÚDO:
    "  "  AnoS(MestradoMaisRelevante(Formação).DataInício);
NOME CAIXA: BlocoAnoObtMest,
  TAMANHO CAIXA: 6,
  CONTEÚDO:
    " " AnoS(MestradoMaisRelevante(Formação).DataObtenção);
NOME CAIXA: BlocoOrientadorMest,
  TAMANHO CAIXA: 56,
  CONTEÚDO: Orientador(TesePósGraduação
           (MestradoMaisRelevante(Formação))).NomeCompleto;
NOME CAIXA: BlocoTeseMest,
  TAMANHO CAIXA: 56 2,
  CONTEÚDO: TesePósGraduação(MestradoMaisRelevante
              (Formação)).TítuloTese;
NOME CAIXA: AreaPalavrasMest,
  ESPAÇO ENTRE LINHAS: dobre;
NOME CAIXA: BlocoPalavraMest,
  TAMANHO CAIXA: 33,
  CONTEÚDO: TesePósGraduação(MestradoMaisRelevante
              (Formação)).PalavrasChave;

NOME CAIXA: AreaSubáreaDout,
  ESPAÇO ENTRE LINHAS: dobre;
NOME CAIXA: BlocoCodDout,
  TAMANHO CAIXA: 12,
  CONTEÚDO:
    AreaSubárea(EspecialidadePG(EspecialidadePósGrad
     (DoutoradoMaisRelevante(Formação)))).CódigoSubárea;
NOME CAIXA: BlocoDescriDout,
  TAMANHO CAIXA: 30,
  CONTEÚDO: EspecialidadePG(EspecialidadePósGrad
      (DoutoradoMaisRelevante(Formação))).Especialidade;
NOME CAIXA: BlocoEntidDout,
  TAMANHO CAIXA: 56,
  CONTEÚDO: Entidade(DoutoradoMaisRelevante(Formação)).
                                  NomeEntidade;
NOME CAIXA: AreaDadosDout,
  ESPAÇO ENTRE LINHAS: dobre;
```

```
NOME CAIXA: BlocoPaisDout,
  TAMANHO CAIXA: 27,
  CONTEÚDO: Entidade(DoutoradoMaisRelevante(Formação)).
                                    PaisEntidade;
NOME CAIXA: BlocoAnoIniDout,
  TAMANHO CAIXA: 6,
  CONTEÚDO:
    " "  AnoS(DoutoradoMaisRelevante(Formação).DataInício);
NOME CAIXA: BlocoAnoObtDout,
  TAMANHO CAIXA: 6,
  CONTEÚDO:
    " " AnoS(DoutoradoMaisRelevante(Formação).DataObtenção);
NOME CAIXA: BlocoOrientadorDout,
  TAMANHO CAIXA: 56,
  CONTEÚDO: Orientador(TesePósGraduação
      (DoutoradoMaisRelevante(Formação))).NomeCompleto;
NOME CAIXA: BlocoTeseDout,
  TAMANHO CAIXA: 56 2,
  CONTEÚDO: TesePósGraduação(DoutoradoMaisRelevante
                          (Formação)).TítuloTese;
NOME CAIXA: AreaPalavrasDout,
  ESPAÇO ENTRE LINHAS: dobre;
NOME CAIXA: BlocoPalavraDout,
  TAMANHO CAIXA: 33,
  CONTEÚDO: TesePósGraduação(DoutoradoMaisRelevante
                          (Formação)).PalavrasChave;

NOME CAIXA: AreaOutrosPósGrad,
  ESPAÇO MARGEM ESQUERDA: 11;
NOME CAIXA: AreaSubáreaPG,
  ESPAÇO MARGEM ESQUERDA: 20;
NOME CAIXA: BlocoCodPG,
  TAMANHO CAIXA: 12,
  CONTEÚDO:
    AreaSubárea(EspecialidadePG(EspecialidadePósGrad
     (OutrasPósGraduações(Formação)))).CódigoSubárea;
NOME CAIXA: BlocoDescriPG,
  TAMANHO CAIXA: 30,
  CONTEÚDO: EspecialidadePG(EspecialidadePósGrad
      (OutrasPósGraduações(Formação))).Especialidade;
NOME CAIXA: BlocoEntidPG,
  ESPAÇO MARGEM ESQUERDA: 20,
  TAMANHO CAIXA: 56,
  CONTEÚDO: Entidade(OutrasPósGraduações(Formação)).
                                    NomeEntidade;
NOME CAIXA: BlocoNívelEspec,
  TAMANHO CAIXA: 9,
  CONTEÚDO: EspecialidadePósGrad(OutrasPósGraduações
                    (Formação)).NívelCNPQ[1];
NOME CAIXA: BlocoNívelMest,
  TAMANHO CAIXA: 9,
  CONTEÚDO: EspecialidadePósGrad(OutrasPósGraduações
                    (Formação)).NívelCNPQ[2];
NOME CAIXA: BlocoPaisPG,
  TAMANHO CAIXA: 27,
  CONTEÚDO: Entidade(OutrasPósGraduações(Formação)).
                                    PaisEntidade;
NOME CAIXA: BlocoAnoIniPG,
  TAMANHO CAIXA: 6,
```

```
  CONTEÚDO:
    " "  AnoS(OutrasPósGraduações(Formação).DataInício);
NOME CAIXA: BlocoAnoObtPG,
  TAMANHO CAIXA: 6,
  CONTEÚDO:
    " " AnoS(OutrasPósGraduações(Formação).DataObtenção);
NOME CAIXA: BlocoNível1Dout,
  TAMANHO CAIXA: 9,
  CONTEÚDO: EspecialidadePósGrad(OutrasPósGraduações
                  (Formação)).NívelCNPQ[3];
NOME CAIXA: BlocoOrientadorPG,
  TAMANHO CAIXA: 56,
  CONTEÚDO: Orientador(TesePósGraduação
      (OutrasPósGraduações(Formação))).NomeCompleto;

NOME CAIXA: BlocoNívelPósDout,
  TAMANHO CAIXA: 9,
  CONTEÚDO: EspecialidadePósGrad(OutrasPósGraduações
                  (Formação)).NívelCNPQ[4];
NOME CAIXA: BlocoNível1LivreDoc,
  TAMANHO CAIXA: 9,
  CONTEÚDO: EspecialidadePósGrad(OutrasPósGraduações
                  (Formação)).NívelCNPQ[5];
NOME CAIXA: BlocoTesePG,
  TAMANHO CAIXA: 56 2,
  CONTEÚDO: TesePósGraduação(OutrasPósGraduações
                        (Formação)).TítuloTese;
NOME CAIXA: AreaPalavrasPG,
  ESPAÇO MARGEM ESQUERDA: 20,
  ESPAÇO ENTRE LINHAS: dobre;
NOME CAIXA: BlocoPalavraPG,
  TAMANHO CAIXA: 33,
  CONTEÚDO: TesePósGraduação(OutrasPósGraduações
                        (Formação)).PalavrasChave;

NOME CAIXA: AreaPrêmio,
  ESPAÇO MARGEM SUPERIOR: 1,
  ESPAÇO MARGEM ESQUERDA: 9,
  ESPAÇO MARGEM DIREITA: 11;
NOME CAIXA: BlocoTítuloPrêmio,
  TAMANHO CAIXA: 60 2,
  CONTEÚDO: PrêmiosDistinções.NomePrêmio;
NOME CAIXA: BlocoEntidPrêmio,
  TAMANHO CAIXA: 53,
  CONTEÚDO: Entidade(PrêmiosDistinções).NomeEntidade;
NOME CAIXA: BlocoAnoPrêmio,
  TAMANHO CAIXA: 7,
  CONTEÚDO: PrêmiosDistinções.DataPrêmio;
NOME CAIXA: AreaIdiomas,
  ESPAÇO MARGEM SUPERIOR: 3,
  ESPAÇO MARGEM ESQUERDA: 4,
  ESPAÇO MARGEM DIREITA: 14;
NOME CAIXA: BlocoIdiomas,
  ESPAÇO MARGEM SUPERIOR: 1,
  TAMANHO CAIXA: 63,
  CONTEÚDO:
    CONCAT(DescriçãoIdiomasCNPQ[i],  FalaIdiomasCNPQ[i],
    " ",  LêIdiomasCNPQ[i],  " ",  EscreveIdiomasCNPQ[i],
    "   ", DescriçãoIdiomasCNPQ[i+4], FalaIdiomasCNPQ[i+4],
```

```
    " ", LêIdiomasCNPQ[i+4], " ", EscreveIdiomasCNPQ[i+4]);

NOME CAIXA: AreaIndicadoresProduc,
  ESPAÇO MARGEM SUPERIOR: 3,
  ESPAÇO MARGEM ESQUERDA: 60;
NOME CAIXA: BlocoIndicadorProduc,
  ESPAÇO MARGEM SUPERIOR: 1,
  TAMANHO CAIXA: 5,
  CONTEÚDO: TotProdução;


NOME CAIXA: AreaArtigosPeriódicos,
  ESPAÇO MARGEM SUPERIOR: 15,
  ESPAÇO MARGEM ESQUERDA: 10,
  ESPAÇO MARGEM DIREITA: 4;
NOME CAIXA: AreaArtigos,
  ESPAÇO MARGEM SUPERIOR: 1;
NOME CAIXA: AreaTipoTítulo,
  TAMANHO CAIXA: 80 3;
NOME CAIXA: BlocoTipoArtigo,
  TAMANHO CAIXA: 14,
  CONTEÚDO: CONCAT("      ",
    ArtigosPeriódicos(ProduçãoCT).NouE[1], "   ",
    ArtigosPeriódicos(ProduçãoCT).NouE[2]);
NOME CAIXA: BlocoAutoriaArtigo,
  TAMANHO CAIXA: 10,
  CONTEÚDO:
    CONCAT(ArtigosPeriódicos(ProduçãoCT).Autoria[1],
    "   ", ArtigosPeriódicos(ProduçãoCT).Autoria[2],
    "   ", ArtigosPeriódicos(ProduçãoCT).Autoria[3]);
NOME CAIXA: BlocoTítuloArtigo,
  TAMANHO CAIXA: 42,
  CONTEÚDO: ArtigosPeriódicos(ProduçãoCT).Título;
NOME CAIXA: AreaAutoresArtigo,
  TAMANHO CAIXA: 80 3;
NOME CAIXA: BlocoAutorArtigo,
  CONTEÚDO: CONCAT(Autores(ArtigosPeriódicos(ProduçãoCT)).
                        NomeAbreviado, ";");
NOME CAIXA: BlocoPeriódico,
  TAMANHO CAIXA: 66,
  CONTEÚDO: ArtigosPeriódicos(ProduçãoCT).NomePeriódico;
NOME CAIXA: BlocoEditora,
  TAMANHO CAIXA: 36,
  CONTEÚDO: ArtigosPeriódicos(ProduçãoCT).EditoraOuVeículo;
NOME CAIXA: BlocoVolume,
  TAMANHO CAIXA: 8,
  CONTEÚDO: ArtigosPeriódicos(ProduçãoCT).Volume;
NOME CAIXA: BlocoFascículo,
  TAMANHO CAIXA: 6,
  CONTEÚDO: ArtigosPeriódicos(ProduçãoCT).Fascículo;
NOME CAIXA: BlocoPagIni,
  TAMANHO CAIXA: 6,
  CONTEÚDO:  "  "
    ArtigosPeriódicos(ProduçãoCT).PáginaInicial;
NOME CAIXA: BlocoPagFinal,
  TAMANHO CAIXA: 6,
  CONTEÚDO:  "  "
    ArtigosPeriódicos(ProduçãoCT).PáginaFinal;
```

```
NOME CAIXA: BlocoAnoArtigo,
  TAMANHO CAIXA: 4,
  CONTEÚDO: ArtigosPeriódicos(ProduçãoCT).AnoProd;
NOME CAIXA: BlocoCodArtigo,
  TAMANHO CAIXA: 33,
  CONTEÚDO: SubáreasPublicação(ArtigosPeriódicos
                (ProduçãoCT)).CódigoSubárea;
NOME CAIXA: BlocoPalavraArtigo,
  TAMANHO CAIXA: 33,
  CONTEÚDO: SubáreasPublicação(ArtigosPeriódicos
                (ProduçãoCT)).PalavrasChave;

NOME CAIXA: AreaLivrosCapítulos,
  ESPAÇO MARGEM SUPERIOR: 11,
  ESPAÇO MARGEM ESQUERDA: 10,
  ESPAÇO MARGEM DIREITA: 4;
NOME CAIXA: AreaLivros
  ESPAÇO MARGEM SUPERIOR: 1;
NOME CAIXA: AreaTítuloLivro,
  TAMANHO CAIXA: 80 3;
NOME CAIXA: BlocoLC,
  TAMANHO CAIXA: 14,
  CONTEÚDO: CONCAT("      ",
    LivrosCapítulos(ProduçãoCT).LivroOuCap[1], "   ",
    LivrosCapítulos(ProduçãoCT).LivroOuCap[2]);
NOME CAIXA: BlocoAutoriaLC,
  TAMANHO CAIXA: 10,
  CONTEÚDO:
    CONCAT(LivrosCapítulos(ProduçãoCT).Autoria[1],
    "   ", LivrosCapítulos(ProduçãoCT).Autoria[2],
    "   ", LivrosCapítulos(ProduçãoCT).Autoria[3]);
NOME CAIXA: BlocoTítuloLivro,
  TAMANHO CAIXA: 96,
  CONTEÚDO: LivrosCapítulos(ProduçãoCT).Título;
NOME CAIXA: BlocoPágs,
  TAMANHO CAIXA: 12,
  CONTEÚDO: LivrosCapítulos(ProduçãoCT).NoPaginas;
NOME CAIXA: AreaAutoresLivro,
  TAMANHO CAIXA: 80 2;
NOME CAIXA: BlocoAutoresLivro,
  CONTEÚDO: CONCAT(Autores(LivrosCapítulos(ProduçãoCT)
                          COMO Livro).NomeAbreviado, ";");
NOME CAIXA: AreaEditoraLivro,
  TAMANHO CAIXA: 80 4;
NOME CAIXA: BlocoLocalEditora,
  TAMANHO CAIXA: 23,
  CONTEÚDO: LivrosCapítulos(ProduçãoCT).Local;
NOME CAIXA: BlocoEditoraLivro,
  TAMANHO CAIXA: 37,
  CONTEÚDO: LivrosCapítulos(ProduçãoCT).EditoraOuVeículo;
NOME CAIXA: BlocoAnoLivro,
  TAMANHO CAIXA: 6,
  CONTEÚDO: LivrosCapítulos(ProduçãoCT).AnoProd;
NOME CAIXA: BlocoTítuloCapítulo,
  TAMANHO CAIXA: 120,
  CONTEÚDO: LivrosCapítulos(ProduçãoCT)
             COMO  CapítuloLivro.TítuloCapítulo;
NOME CAIXA: BlocoPagIniCap,
  TAMANHO CAIXA: 6,
```

```
  CONTEÚDO:  " "
    LivrosCapítulos(ProduçãoCT) COMO CapítuloLivro.
                        PáginaInicial;
NOME CAIXA: BlocoPagFinalCap,
  TAMANHO CAIXA: 6,
  CONTEÚDO:  " "
    LivrosCapítulos(ProduçãoCT) COMO CapítuloLivro.
                        PáginaFinal;
NOME CAIXA: AreaAutoresCapítulo,
  TAMANHO CAIXA: 80 2;
NOME CAIXA: BlocoAutoresCapítulo,
  CONTEÚDO: CONCAT(Autores(LivrosCapítulos(ProduçãoCT)
                    COMO CapítuloLivro).NomeAbreviado, ";");
NOME CAIXA: BlocoCódLivro,
  TAMANHO CAIXA: 33,
  CONTEÚDO: SubáreasPublicação(LivrosCapítulos
                   (ProduçãoCT)).CódigoSubárea;
NOME CAIXA: BlocoPalavraLC,
  TAMANHO CAIXA: 33,
  CONTEÚDO: SubáreasPublicação(LivrosCapítulos
                   (ProduçãoCT)).PalavrasChave;

NOME CAIXA: AreaTesesOrient,
  ESPAÇO MARGEM SUPERIOR: 17,
  ESPAÇO MARGEM ESQUERDA: 10,
  ESPAÇO MARGEM DIREITA: 4;
NOME CAIXA: AreaTesesAprov,
  ESPAÇO MARGEM SUPERIOR: 1;
NOME CAIXA: BlocoNívelTeseAprov,
  TAMANHO CAIXA: 13,
  CONTEÚDO:  CONCAT("      ",
    TesesOrientAprovadas(ProduçãoCT).NívelTese[1], "   ",
    TesesOrientAprovadas(ProduçãoCT).NívelTese[2]);
NOME CAIXA: BlocoNomeAluno,
  TAMANHO CAIXA: 53,
  CONTEÚDO: NomeAluno(TesesOrientAprovadas(ProduçãoCT)).
                                    NomeCompleto;
NOME CAIXA: BlocoTitTeseAprov,
  TAMANHO CAIXA: 80 3,
  CONTEÚDO: TesesOrientAprovadas(ProduçãoCT).TítuloTese;
NOME CAIXA: AreaEntidTese,
  TAMANHO CAIXA: 80 3;
NOME CAIXA: BlocoEntidTese,
  TAMANHO CAIXA: 41,
  CONTEÚDO: Entidade(TesesOrientAprovadas(ProduçãoCT)).
                                    NomeEntidade;
NOME CAIXA: BlocoAnoTeseAprov,
  TAMANHO CAIXA: 7,
  CONTEÚDO:  CONCAT("   ",
    AnoS(TesesOrientAprovadas(ProduçãoCT).DataTese));


NOME CAIXA: AreaProdCTComplementar,
  ESPAÇO MARGEM SUPERIOR: 19,
  ESPAÇO MARGEM ESQUERDA: 10,
  ESPAÇO MARGEM DIREITA: 4;
NOME CAIXA: AreaProdComplement,
  ESPAÇO MARGEM SUPERIOR: 1;
```

```
NOME CAIXA: AreaTipoProd,
  TAMANHO CAIXA: 80 3;
NOME CAIXA: BlocoTipoProd,
  TAMANHO CAIXA: 10,
  CONTEÚDO: "       "  ProduçõesComplementCNPQ(ProduçãoCT).
                                                TipoCNPQ;
NOME CAIXA: BlocoAutoriaProd,
  TAMANHO CAIXA: 10,
  CONTEÚDO:
    CONCAT(ProduçõesComplementCNPQ(ProduçãoCT).Autoria[1],
     "    " ProduçõesComplementCNPQ(ProduçãoCT).Autoria[2],
     "    " ProduçõesComplementCNPQ(ProduçãoCT).Autoria[3]);
NOME CAIXA: BlocoTítuloProd,
  CONTEÚDO: ProduçõesComplementCNPQ(ProduçãoCT).
                                              EditoraOuVeículo;
NOME CAIXA: AreaDadosProd,
  TAMANHO CAIXA: 80 3;
NOME CAIXA: BlocoLocalProd,
  TAMANHO CAIXA: 38,
  CONTEÚDO: ProduçõesComplementCNPQ(ProduçãoCT).
                                                Local;
NOME CAIXA: BlocoMesProd,
  TAMANHO CAIXA: 4,
  CONTEÚDO: ProduçõesComplementCNPQ(ProduçãoCT).
                                                MesProd;
NOME CAIXA: BlocoAnoProd,
  TAMANHO CAIXA: 4,
  CONTEÚDO: ProduçõesComplementCNPQ(ProduçãoCT).
                                                AnoProd;
NOME CAIXA: BlocoRegProd,
  TAMANHO CAIXA: 13,
  CONTEÚDO: ProduçõesComplementCNPQ(ProduçãoCT)
              COMO ProdutoOuProcesso.RegistroINPI;
NOME CAIXA: BlocoEstágio;
  TAMANHO CAIXA: 7,
  CONTEÚDO: ProduçõesComplementCNPQ(ProduçãoCT)
              COMO ProdutoOuProcesso.Estágio;
NOME CAIXA: BlocoComplemento,
  TAMANHO CAIXA: 66,
  CONTEÚDO: ProduçõesComplementCNPQ(ProduçãoCT).
                                              Complemento;
```