ESPECIFICAÇÃO E IMPLEMENTAÇÃO DE UM SISTEMA DE REGRAS DE PRODUÇÃO FORTEMENTE ACOPLADO A BANCOS DE DADOS

Esther de Castro Pacitti

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. Peclro Manoel da Silveira, Ph.D.
(Presidente)

Prof.ª Slieila Regina Murgel Veloso, D.Sc.

Prof.ª Sueli Bandeira Teixeira Mendes, Ph.D.

Prof. Ysmar Vianna e Silva Filho, Ph.D.

RIO DE JANEIRO, RJ -BRASIL
OUTUBRO DE 1991

PACITTI, ESTHER DE CASTRO

Especificação e Implementação de um Sistema
de Regras de Produção Fortemente Acoplado a Bancos de Dados
[Rio de Janeiro], 1991
viii, 93p., 29,7cm. (COPPE/UFRJ, M.Sc.,
Engenharia de Sistemas e Computação, 1991)
Tese - Universidade Federal do Rio de Janeiro
1. Integração de Bancos de Dados e Sistemas Especialistas
2. Modelo Entidade-Relacionamento 3. Orientação a Objetos
I. COPPE/UFRJ II. Título (série).

Ao Luiz Eduardo.

AGRADECIMENTOS

A oportunidade de ter saúde física, mental, espiritual e sentimental sem as quais este trabalho não seria possível.

Ao meu marido Luiz Eduardo, pela importância que ele exerce em todos os setores da minha vida, sempre me encorajando e acompanhando.

Aos meus pais Tercio e Eunice, que proporcionaram as bases da minha educação, sempre me incentivando a atingir grades metas.

Aos meus amigos em geral, e em especia.1 ao Oswaldo e Luci pela amizade calorosa, presença e força.

Ao professor Peclro, pela valiosa.orientação e dedicação recebida.

A professora Sheila, pelo apoio e compreensão.

Ao Núcleo de Computação Eletrônica da UFRJ que possibilitou o desenvolvimento clesta tese.

Finalmente, a toclos aqueles, que em qualquer plano, consciente ou inconsciente contribuiram para a realização deste trabalho.

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

Especificação e Implementação de um Sistema de Regras de Produção Fortemente Acoplado a Bancos de Dados

Estlier de Castro Pacitti
Outubro de 1991

Orientador: Pedro Manoel da Silveira
Co-Orientador: Sheila Regina Murgel Veloso
Programa: Engenharia de Sistemas e Computação

O objetivo desta dissertação consiste no estudo, especificação e implementação de um ambiente que integra sistemas de regras de produção a bancos de dados, chamado QUICK-SHELL. Este ambiente permite que um usuário de um sistema especialista baseado em regras de produção tenha acesso aos dados armazenados num banco de dados de tal forma que se tenha a impressão do último ser uma extensão da base de fatos. A QUICK-SHELL está inserida num sistema de desenvolvimento de *software* denominado QUICK-DB. Este sistema utiliza o modelo Entidade-Relacionamento como base, e a forma de representação do conhecimento adotada na *shell* consiste nas triplas O-A-V. A adoção desses modelos de abstração permitiu a definição de termos, literais, regras e mecanismos de quantificação que abrigam de forma natural os elementos O-A-V e E-R.

No estudo teórico, algumas propostas de acoplamento e arquiteturas são descritas e analisadas a fim de enquadrar a nossa opção naquela que se mostre mais adequada. A partir deste ponto, apresenta-se a especificação do ambiente QUICK-SHELL através da definição dos elementos componentes e da linguagem de regras. Com base na especificação, descreve-se a implementação, que no nosso caso foi calcada na filosofia de orientação a objetos. Apresenta-se a estrutura interna da *shell* e do banco de dados, bem como os métodos de acesso ao mesmo. Visto isso, aborda-se os principais aspectos do tradutor QUICK-SHELL, cujo o objetivo é traduzir termos em comandos de orientação a objetos.

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

Specificatioii and Implementation of a Production Rule System Tightly Couplecl to Databases

Esther de Castro Pacitti

Thesis Supervisor: Pedro Manoel da Silveira
Co-Supervisor : Sheila Regina Murgel Veloso
Department : Systems Engineering and Computing

The main purpose os this dissertation is tl-ie specification and implementation of a software environment for production rules tightly coupled to databases, which is callecl QUICK-SHELL. The environment allows access to the storecl data by expert systems users in a natural way. The shell described is part of a wider system for database definition and manipulation built over the Entity Relationship Model, QUICK-DB. For the shell, the O-A-V model for knowledge representation has been used, in such a way that the terms, literals rules and deduction mechanismes coulcl fit together E-R and O-A-V elements.

In tl-ie tl-ieoretical basis, several proposals of architectures and coupling approaches have been analyzed in order to situate the present work. From this point, the whole environment is specifiecl, including tl-ie syntactical and semantics of tl-ie language. The internal structure of tl-ie sl-iell is presented along with tl-ie methods for database access, done through an object-oriented implementation. The main aspects of tl-ie rule translations are seen in detail.

Indice

I Introdução 1

1.1 Introdução 1

1.2 Histórico 2

1.3 Motivações 4

1.4 Contesto do Trabalho 6

1.5 Objetivos e Composição do Trabalho 9

I.6 Estrutura da Tese 10

II Fundamentos Teóricos 12

II.1 Introdução 12

11.2 Projetos de Bancos de Dados 12

II.2.1 Modelos de Dados 13

II.2.2 Abstrações no Projeto Conceitual 14

II.2.3 Modelo-Entidade Relacionamento 15

11.3 Sistemas Especialistas 16

11.4 Representação do Conhecimento 19

II.4.1 Redes Semânticas 20

II.4.2 Triplas O-A-V 21

II.4.3 Frames 21

II.4.4 Regias de Produção 22

II.4.5 Representação Interna. de Regras 23

11.5 Integração de Sisteinas Inteligentes e Bancos de Daclos 25

II.5.1 Bancos de Dados Dedutivos 26

II.5.2 Mecanismos Dedutivos para. Banco de Dados 26

II.5.3 Bancos de Dados para Sistemas Especialistas 27

II.5.4 Propostas de Acoplamento 27

II.5.5 Aspectos Arquiteturais de Acoplamento 31

II.6 Conclusões 33

III QUICK-SHELL: Uma Proposta de Integração **34**

III.1 Introdução 34

111.2 O Modelo E-R e suas Extensões 35

III.2.1 Propriedades de um esquema E-R 36
III.2.2 Operações sobre o Modelo E-R 38
III.3 Triplas O-A-V e Regras de Produção 39
111.4 O Ambiente de Integração 42
III.4.1 Regras e Literais na QUICK-SHELL 43
III.5 Aspectos Arquiteturais 48
III.5.1 Semântica da Linguagem de Regras 49
III.6 Exemplo de Utilização 50
III.7 Conclusões 52

IV Implementação 54
IV.1 Introdução 54
IV.2 Aspectos de Orientação a Objetos 55
IV.3 Orientação a Objetos na Quick-Shell 56
IV.3.1 Mecanismo de Inferência 57
IV.4 Banco de Dados: Objetos e Métodos 58
IV.4.1 Servidor E-R 58
IV.4.2 Métodos de Acesso 60
IV.5 Traduzindo Expressões para a Shell **O-O** 62
IV.5.1 Explicando a Tradução 64
IV.5.2 Algoritmo de Tradução e Estruturas de Dados 66
IV.6 Ferramenta de Implementação 71
IV.7 Conclusões 72

V Conclusões 74
V.1 Introdução 74
V.2 Contribuições 74
V.3 Migração para outros Bancos de Dados 75
V.4 Continuidade do Trabalho 76
V.5 Comparação coin Outros Trabalhos 78
Bibliografia 80
Apêndice A - Exemplo de Utilização 84
Apêndice B - Sintaxe da Linguagem de Regras 90

Capítulo I

Introducão

I.1 Introdução

O objetivo desta dissertação consiste no estudo, especificação e implementação de um ambiente que integra sistemas de regras de proclução a bancos de dados, chamado QUICK-SHELL. Este ambiente permite que um usuário de um sistema especialista baseado em regras de produção tenha acesso aos claclos armazenados num banco de claclos, de tal forma. que se tenha a impressão do último ser uma extensão da base de fatos. A QUICK-SHELL está inserida num sistema de desenvolvimento de *software* denominado QUICK-DB. Este sistema utiliza o modelo Entidade-Relacionamento como base, e a forma. de representação do conhecimento aclotacla na *shell* consiste nas triplas O-A-V. A adoção desses modelos de abstração permitiu a definição de termos, literais, regras *e* mecanismos de quantificação que abrigam de forma natural os elementos O-A-V e E-R.

Na confecção desta tese, buscou-se na literatura sistemas que se assemelhassem às classificações teóricas aqui apresentadas. Algumas aplicações foram encontradas, citadas no capítulo II, porém, em poucos casos comentou-se os aspectos teóricos de tais sistemas, isto é, como foram especificados os termos, literais, regras, etc. Os artigos que aludiam a estes fatos normalmente davam um enfoque teórico bastante complexo, ultrapassando os limites desejáveis neste trabalho. Tais sistemas em geral abordavam a integração de linguagens lógicas e bancos de claclos relacionais, no escopo de banco de dados dedutivos. Assim, o conteúdo do presente trabalho consiste numa proposta. de um ambiente simples, tangível e aberto a. extensões, que integra sistemas especialistas baseados em regras de produção a bancos de claclos. Tudo isso inserido num ambiente completo de projeto, manutenção e inanipulação de bancos de claclos construídos sobre o modelo Entidade-Relacionamento.

Como introdução, este capítulo aborda os fundamentos da tese. Inicialmente, apresenta-se um panorama histórico das áreas de representação do conhecimento e banco de claclos. O objetivo é mostras a. real tendência de integração destas áreas. Com isso, são reportadas as motivações básicas do presente trabalho e o contesto no qual está inserido. Em seguida, aborda-se os principais objetivos e a composição da monografia. Por fim, clecreve-se a estrutura da tese.

I.2 Histórico

Representação do Conhecimento

O fascínio causado pela inteligência humana e as suas mais versáteis facetas e formas de apresentação não constitui novidade para o mundo e, absolutamente, não é um fenômeno recente. Mas foi na década de 50 que se cunhou o termo Inteligência Artificial (IA) no contexto da computação. Apesar do entusiasmo dos pesquisadores, entre os quais teve destaque Alan Turing [1], essa. área de pesquisa não obteve expressão significativa na época., cleviclo às próprias limitações tecnológicas. Por exeinplo, os primeiros sistemas dedutivos, ein geral, não estavam vinculados a nenhum mecanismo específico de representação simbólica do conhecimento. Tais sistemas utilizavam linguagens aritméticas, como o Fortran, para implementar alguinas técnicas de inteligência artificial conhecidas até então. Além clisto, devido as restrições de *hardware* existentes, estes sistemas eram em geral custosos, lentos e não produziam resultados satisfatórios.

Junto com o desenvolvimento da tecnologia de microeletrônica que resultou numa nova geração de circuitos mais velozes, as pesquisas em inteligência artificial tomaram novos impulsos. Surgiram as linguagens de programação ein lógica [2], que se baseavam na teoria do cálculo de preclicaclos, e foram ainplamente empregadas no desenvolvimento de provadores automáticos de teoreinas. Atualmente, são utilizadas para especificações, como fundamentos para linguagens de consulta a banco de dados, na construção de compiladores, etc.

Na. década de 80, certo destaque foi dado ao desenvolvimento de sistemas especialistas. Estes são "prograinas que utilizam técnicas de inteligência artificial para resolver problemas complexos, que comportam razoável quantidade de conhecimento e experiência de especialistas em domínios específicos" [3]. A viabilização de tais sistemas exigia, entre outras coisas, modelos que abrigassem técnicas que

permitissem a dedução face ao conhecimento incompleto e o desenvolvimento de interfaces amigáveis que possibilitassem as interações necessárias à execução de um sistema especialista.

Assim, ganhou importância a questão da representação do conhecimento. Modelos de representação do conhecimento basicamente consistem num conjunto de convenções sintáticas e semânticas que tornam possível, através da abstração, a descrição de objetos, relacionamentos, procedimentos e meta-conhecimento. Devido às características destes modelos tornou-se viável o desenvolvimento de técnicas de raciocínio aproximado que, em geral, resolvem as questões mencionadas anteriormente.

Atualmente, percebe-se o emprego de sistemas especialistas em diversas áreas, e no Brasil podemos destacar seu emprego nas áreas de Telecomunicações, Petróleo/Geologia, Medicina, Construção Civil, etc [3]. Algumas de suas aplicações são muito específicas e por isso não necessitam de grandes bases de informação. Mas na medida em que as aplicações atendem às situações onde a base de fatos é robusta, como sistemas bancários ou financeiros, o número de dados necessários ao processo de dedução cresce substancialmente a ponto de interferir no processamento global do sistema especialista.. Isto é, o sistema,se torna lento, ou até busca dados errados. Assim, é necessário definir estratégias para a manipulação de bases de dados em memória secundária, ou então integrar o sistema especialista com um banco de dados previamente desenvolvido.

Bancos de Dados

As pesquisas na área de banco de dados tomaram-se significativas no início da década de 60. Apesar de ser uma área recente, contou desde o início com resultados práticos e visíveis. Os pontos centrais desta área estão relacionados primordialmente com os aspectos de implementação e as características operacionais dos sistemas de banco de dados.

Na primeira geração de [4] *software* para desenvolvimento de projetos de banco de dados, pouca ênfase era dada ao completo entendimento das propriedades estruturais dos dados, independentemente dos detalhes de implementação. Naqueles projetos, assumia-se que o projetista conhecesse e estivesse envolvido em todas as fases de desenvolvimento, centralizando todo o controle e decisões do projeto, e ao mesmo tempo tendo o papel de principal usuário.

Na segunda geração (tabela, I.1), as motivações estavam mais voltadas para a teoria de bancos de dados. Os modelos lógicos [5] emergiram como ferramenta

para descrição lógica dos bancos de claclos, entre os quais teve destaque o modelo relacional. Nesta. fase, grande ênfase era. dada à implementação, ao projeto de bancos de claclos, modelos de dados e linguagens de consulta. Apesar clos inoclelos lógicos proverem meios mais abstratos para descrever um banco de dados, eles ainda estavam vinculados a Sistemas Gerenciadores de Bancos de Dados (SGBD) [7] específicos, impossibilitando a. definição de um projeto puramente conceitual.

Neste sentido, na terceira geração, a preocupação se direcionou para a obtenção de técnicas de abstração e análise mais depuradas, na intenção de expressar ao máximo a. semântica. da informação. Acreditava.-se que os principais transtornos que ocorriam no desenvolvimento de bancos de dados eram causados pela falta de clareza no entendimento da natureza exata clos dados no nível abstrato. Em muitos casos, a clescrição do banco de claclos ficava relegada a. obscuras construções, muitas vezes embutidas nos programas, cujo objetivo era. descrever o sistema para o próprio computador, relegando a necessidade de ser clara para seus usuários a um segundo plano.

Atualmente, percebe-se claramente a. transição para uma nova geração de bancos de claclos. Uma das principais características percebidas consiste na manipulação de informação não convencional [8]. Neste aspecto e no contexto desta tese, o tipo de informação não convencional a ser manipulada consiste em conhecimento. Conhecimento deve estar associado a. mecanismos dedutivos para a derivação de informação numa 𝕝 ase de dados.

I.3 Motivações

Sistemas baseados em conhecimento e bancos de dados possuem características arquiteturais semelhantes. Ambos exigem a. escolha de estruturas apropriadas para representar o conhecimento e mecanismos eficientes para responder a perguntas ou deduzir novos conhecimentos a partir dos já existentes. No entanto, sistemas baseados ein conhecimento enfatizam dedução e representações complexas, enquanto que em bancos de claclos a ênfase reside na. eficiência. de acessos e na manipulação de estruturas, que são persistentes e armazenam grandes coleções de informações.

É inegável que a integração destas duas tecnologias traz benefícios para ambas. A contribuição fundamental da tecnologia de Inteligência Artficial consiste na obtenção de modelos semânticos mais ricos, isto é, um entendimento mais claro a respeito do conhecimento representado. Por outro lado, a contribuição da tecnologia de bancos de claclos propicia a habilidade de se realizar acessos eficientes a bases de conhecimento robustas e melhor estruturadas, com auxílio de controles

	Passado	Presente	Futuro
Requisitos de Proc.	Proc. Comer. Simples	Proc. de Daclos Compartilhados	Proc. de Daclos e Conhecimento
Modelos de Daclos Adotados	Modelos de Dados Clássicos, (Relacional, Rede, Hierárquico)	Modelos de Dados Semânticos	Em Aberto
SGBD de Suporte	Segunda Geração de SGBD	Terceira Geração de SGBD	Sistemas Gerenciadores de Bases de Conhec. e Dados

Tabela. I.1: Desenvolvimento Computacional e Semântico em BD

de concorrência, segurança e proteção de dados, acesso otimizado e gerenciamento de memória secundária.

A integração desses sistemas se efetua de diversas formas, dependendo do objetivo do projetista. No contexto desta tese, o enfoque se faz na integração de sistemas especialistas e bancos de claclos, no sentido dos primeiros terem acesso a estes. Acredita-se que, devido ao fato de ambas as áreas empregarem mecanismos de alsstração, seja possível a definição de termos, literais e regras que abrigam os elementos fundamentais de cada área. Neste sentido, optou-se pelos modelos Objeto-Atributo-Valor [9] e Entidade Relacionamento [11], correspondendo à forma de representação do conhecimento e modelo de claclos, respectivamente.

1.4 Contexto do Trabalho

O presente trabalho está inserido num projeto maior, UniversiData [28], desenvolviclo no Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro. O macro objetivo é a obtenção de um ambiente completo para projeto, manutenção e manipulação de bancos de claclos construídos sobre o Modelo-Entidade-Relacionamento. QUICK-DB, como é chamado o protótipo de software em construção, é composto por um conjunto de ferramentas de software que o tornam prático, simples e didático no desenvolvimento de aplicações de bancos de claclos.

QUICK-DB está. baseado na idéia de um ambiente uniforme e unificado, gráfico, onde todos os componentes referem-se ao Modelo E-R. Este enfoque permite que seus usuários sigam uma linha de trabalho consistente, simplificando a metoclologia e evitando a introdução de conceitos estranhos ao modelo básico.

Desc ição Geral

A idéia da utilização do Modelo E-R para definição e utilização de bases de claclos tem sua implementação calcada sobre um conjunto de ferramentas de software, conforme pode ser visto na figura 1.1. Montados sobre interfaces similares e amigáveis, caracterizando um ambiente integrado e homogêneo, esses módulos do sistema. propiciam ao usuário desfrutar plenamente das vantagens da presente metodologia.

A) DICIONÁRIO DE DADOS: Este componente é um dicionário de claclos orientado para o Modelo E-R, e tem como característica principal sua. plena capacidade

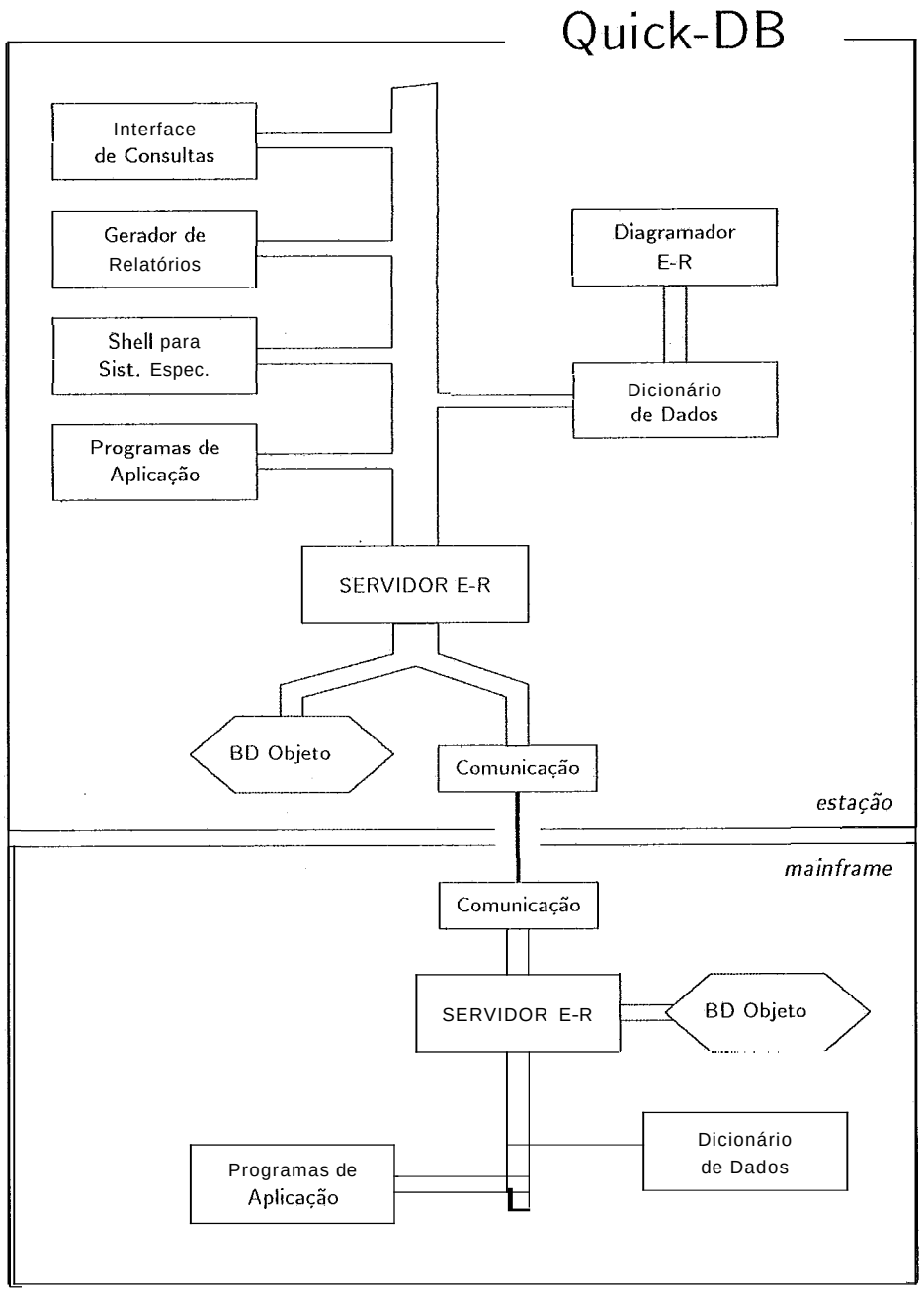

Figura I.1: Diagrama funcional

Figura I.1: Diagrama Funcional

de representar elementos do Modelo E-R. Essa.é uma.vantagem sobre outros dicionários de caráter geral. Por exemplo, poucos são os dicionários que tratam fielmente da questão de atributos herdados, nos casos de generalização.

B) DIAGRAMADOR E-R: O diagramador E-R é uma ferramenta gráfica para especificação de diagramas E-R. Trabalhando com o móclulo de dicionário de dados, o diagramador E-R oferece facilidades extremamente amigáveis para. a definição de esquemas para bases de claclos. O sistema trabalha com menus gráficos e direciona o usuário na sua tarefa. É aqui que se configura a "face" gráfica de um sistema. As demais ferramentas tomam o diagrama aqui definido como lsase, estabelecendo a.unidade visual que o sistema oferece.

C) GERADOR DE BANCOS DE DADOS: A partir do diagrama E-R, expresso no dicionário de claclos, são geradas descrições da base de dados para implementação nas instalações correntemente em uso. Da análise das construções definidas pelo usuário no diagrama E-R, este módulo gera a descrição mais adequada para implementação no SGBD objeto.

D) SERVIDOR DE BANCOS DE DADOS: Adicionalmente, existe um servidor de banco de claclos que atende às solicitações do usuário. As operações, originalmente especificadas sobre uma visão E-R do banco de claclos, são traduzidas para operações da implementação corrente e atendidas por este servidor. Os programas de aplicação tem acesso ao banco de claclos pela chamada a bibliotecas de funções que operam sobre o banco de claclos através do servidor.

E) BANCO DE DADOS AUTÔNOMO: O sistema é capaz de atuar como banco de dados autônomo, na própria estação de trabalho. Para.tanto, há um serviclor de banco de claclos, cuja interface é orientada a objetos, que manipula um banco de claclos local. Os objetos, atributos e métodos são gerados de tal forma que os usuários podem ter acesso ao banco facilmente e ao mesmo tempo contar com todas as facilidades de seus ambientes de programação preferidos.

F) MÓDULO DE CONSULTAS: No Projeto UniversiData pretende-se explorar o ambiente da interface gráfica para. que se ofereça ao usuário casual a possilsilidade de especificar consultas de modo simples e intuitivo. A idéia.é construir a expressão da consulta, isto é, sua especificação, num processo estreitamente ligado ao diagrama E-R, e dirigido pelo sistema. Além de possibilitar facilidades para.consulta,,essa implementação permite que se tenha uma interface

gráfica para sistemas de grande porte, como o DMS II da Unisys, por exemplo. O sistema, além da interface em si, faz uso da facilidade de acesso ao servidor de banco de claclos ligado através de uma linha de comunicação ao micro, possibilitando que, da interface gráfica, se tenha pleno acesso ao banco de dados [10].

G) GERADOR DE RELATÓRIOS: Este módulo é constituído por um conjunto de programas total ou semi-automáticos para geração de relatórios de média e baixa complexidade. Tal estratégia. resulta. numa menor dependência dos programas em relação às bases de dados implementadas, aumentando o grau de liberdade do administrador da base de claclos no que se refere aos aspectos de modelagem e estrutura do banco. Ou seja, pela. utilização de geradores de relatórios e outras faciliclacles do gênero, diminui-se o impacto que as mudanças na definição do banco de claclos possam causar nos programas de aplicação.

H) GERADORES DE APLICAÇÕES: Esta é uma. parte que visa estabelecer uma conexão entre a metodologia aqui adotada. e técnicas de construção de sistemas num ambiente de administração de dados. A idéia é aproveitar o ambiente de descrição clas bases de claclos, expressão de consultas e geração de relatórios e integrá-lo na. fase de definição da aplicação em si. Essa integração faz com que a especificação do sistema. se beneficie diretamente das funções embutidas no ambiente de apoio.

I) SHELL PARA SISTEMAS ESPECIALISTAS: A construção de Sistemas Especialistas tomou impulso significativo nos anos 80 e abriu uma nova e promissora linha de implementação para uma significativa classe de aplicações. Uma clas fragilidades dessas implementações, entretanto, tem sido a dificuldade, que existe na maior parte clas *shells* atualmente em uso, de se trabalhar com bases de fatos extensas. Assim, a interface de sistemas especialistas e bancos de claclos é interessante uma vez que é vantajoso dispor da união clas duas tecnologias, e é justamente o objeto de estudo da tese.

1.5 Objetivos *e* Composição do Trabalho

O objetivo do presente trabalho consiste em analisar os requisitos necessários ao desenvolvimento de ambientes que integram bancos de dados e sistemas especialistas, isto é, pesquisar as possíveis estratégias de acoplamento e arquiteturas. Com base neste estudo, é proposto um ambiente que acopla fortemente bancos de

claclos baseados no modelo E-R e *shells* desenvolvidas sob o paradigma Objeto-Atributo-Valor. Neste aspecto, são definidos termos, literais, regras, mecanismos de quantificação e inferência que agem sob os elementos da *shell* e banco de clados.

A característica principal cl0 ambiente aqui apresentado é a total integração entre os vários componentes e a absorção de termos e construções E-R de forma clara e simples. Além disso, a introdução de ordens de acesso e atributos referenciais ao inoclelo E-R muda sensivelmente o cenário no que tange às possibilidades de acesso a base de claclos. A semântica dos termos que podem ser construídos fica mais poderosa e é possível a especificação de varredura na base de claclos, de inoclo a promover a busca de instâncias que atendam as premissas das regraa.

Para concretizar a especificação, desenvolve-se uma implementação no contesto do QUICK-DB, clue corresponde ao item *i* clescrito anteriormente. As características mais importantes da implementação consistem da definição de uma linguagem de regras que abrigue naturalmente o termos, lite ais, regras, etc, para o ambiente proposto, e o einprego de programação orientada a objetos na geração de código. Internamente, os elementos inerentes à base de conhecimento e à estrutura do banco de claclos são definidos à luz da orientação a objetos. Neste contexto, a interface entre a *shell* e o banco de dados utiliza um conjunto de métodos que têm acesso, de forma natural, à base de claclos.

1.6 Estrutura da Tese

Esta tese está cliviclicla em cinco capítulos, incluindo este.

O segundo capítulo engloba os estudos de modelagem de claclos, principalmente no que concerne o moclelo E-R. São apresentadas e analisadas as principais formas de representação do conhecimento. Por fim, são abordadas as propostas de acoplamento e arquiteturas encontradas na literatura. Ao longo do testo, algumas referências a outros trabalhos de acoplamento são feitas a fim de ilustrar os conceitos apresentados.

O terceiro capítulo compreende a especificação de um ambiente que integra bancos de clados E-R e *shells* construídas sob o paradigma O-A-V. São apresentadas as extensões propostas ao moclelo E-R. Define-se termos, literais, regras, bem como a linguagem de regras. Mostra-se também como os elementos de um esquema E-R são manipulados por uma base de regras.

O quarto capítulo apresenta a proposta de implementação baseada na especificação do capítulo anterior. Uma ênfase especial é dada à programação orientada a objetos uma vez que esta técnica foi empregada. Neste sentido, mostra-se como

são organizados a, base de conhecimento, o banco de dados, a interface de acesso e o mecanismo de inferência. São descritos os principais aspectos do traclutor da linguagem de regras detalhando-se a forma como os termos, literais e regras são traduzidos para comandos de orientação a objetos.

O quinto capítulo descreve os principais resultados obtidos no trabalho como um todo. Algumas propostas de extensões são apresentadas e são clescritos os procedimentos necessários à migração para outros bancos de claclos.

No Apêndice A é apresentado um exemplo que engloba um esquema E-R associado a uma base de regras, que serve como ponto de referência para comentários ao longo do texto.

No Apêndice B é descrita a sintaxe da linguagem de regras especificada.

Capítulo II

Fundamentos Teóricos

II.1 Introdução

Neste capítulo vamos apresentar uma concisa revisão de conceitos teóricos que são fundamentais ao desenvolvimento desta monografia, quais sejam: projeto de banco de claclos, modelagem de dados, representação do conhecimento, sistemas especialistas e aspectos sobre a integração de banco de dados e sistemas de cleclução.

II.2 Projetos de Bancos de Dados

Esta seção descreve os princípios básicos de projetos de banco de clados, levando em consideração a. tendência atual de obter técnicas de abstração e análise mais depuradas, na intenção de melhor expressar a. semântica. da informação [12].

Neste contesto, cada fase do projeto total de bancos de claclos é resolvida de uma forma independente, através de métodos e técnicas específicas. Esta decomposição facilita, tanto para o usuário como para o projetista, a compreensão nítida e um desenvolvimento adequado do banco de claclos.

Quanto à decomposição, seguindo o padrão ANSI-SPARC, definiu-se os níveis conceitual, lógico e físico:

1. PROJETO CONCEITUAL: Consiste na especificação clos requerimentos desejaclos pelo usuário através de um modelo conceitual, produzindo o que é chamado de esqueina conceitual. O esqueina.conceitual é uma clescrição, em alto nível, das estruturas do banco de claclos independentemente do SGBD que for usado na implementação. O modelo coilceitual é a simbologia utilizada para a descrição clos esquemas, que na maior parte das vezes corresponde a uma linguagem.

2. PROJETO LÓGICO: Parte do esquema conceitual, e produz como resultado o esquema lógico, que consiste na descrição da estrutura do banco de claclos que pode

ser processada por um software de SGBD. Analogamente aos inoclelos conceituais, o moclelo lógico consiste numa simbologia para especificação de esquemas lógicos. Os mais utilizados pertencem as classes relacional, rede e hierárquico [5]. O projeto lógico é dependente da classe do moclelo lógico utilizado pelo SGBD.

3. PROJETO FÍSICO: Parte do esquema lógico e produz como resultado o esquema físico, que consiste na descrição da implementação do banco de daclos em memória secundária [7]. São descritas as estruturas de armazenamento e métodos de acesso. Deste modo, o projeto físico é suportado por um SGBD especifico. Existe uin processo de realimentação entre o projeto físico e o lógico, uma vez que decisões tomadas durante o projeto físico, na intenção de otimizar o desempenho, podem afetas a estrutura do esquema lógico.

No âmbito deste trabalho, maior destaque é dado ao projeto coilceitual. Em seguida serão abordadas algumas vantagens quando da sua utilização.

O projeto conceitual é independente do SGBD, o que o torna bastante estável. Este aspecto está associado a outras vantagens como a possibilidade da escolha a posteriori do SGBD, podendo o mesmo projeto conceitual ser suportado por diversos SGBD.

Na elaboração do projeto conceitual, requer-se uma grande interação junto aos usuários, no sentido de se entender os significados dos dados. Pelo fato desta fase não exigir um grande conhecimento técnico de bancos de claclos, a complexidade do diálogo entre projetistas e usuários é significativamente reduzida, possibilitando que o usuário aprenda o suficiente a ponto de colaborar na especificação, melhorando a qualidade do esquema e, consequentemente, produzindo o resultado desejado a um custo inais baixo.

Uma vez que o banco de dados esteja.em operação, o projeto coilceitual revela-se uma documentação clara, que facilita.a.compreensão do esquema e das operações definidas para estes.

II.2.1 Modelos de Dados

A modelagem do esquema coilceitual de um banco de claclos é de vital importância e deve refletir o mundo real das aplicações de forma.simples, completa e correta. Não é trivial a obtenção de tais propriedades quando se deseja que, adicionalmente, o modelo possa ser implementaclo eficientemente. Deve-se escolher uina inetoclologia de modelagem que abrigue estas preocupações e que seja de utilidade comprovada. Assim, não se pode expandir demasiadamente a liberdade de expressão do modelo, criando-se facilidades elegantes e vistosas, mas de difícil implementação, ou muito

menos torná-lo simplório ao ponto de inutilidade.

II.2.2 Abstrações no Projeto Conceitual

Os inoclelos conceituais, empregam mecanismos de abstração que possibilitam a representação de uma certa realidade. Analogamente à representação do conhecimento em IA, existem diversos inoclelos conceituais que podem ser empregados no projeto de bancos de dados de acordo com o objetivo do projetista [5]. Basicamente, os inoclelos conceituais utilizam os conceitos classificação, agregação e generalização.

CLASSIFICAÇÃO é a abstração utilizada no agrupamento de objetos que possuem características semelhantes. A clescriçã.odo conjunto de propriedades de uma classe é feita de uma só vez, de forma. concisa., independente de número de objetos. Matematicamente, esta abstração corresponde à pertinência de um elemento num conjunto (é membro de).

AGREGAÇÃO é a abstração que define uma nova. classe como coinposição de um conjunto de classes, que representam suas partes componentes. Matematicamente, este conceito correspoilcle à coinposição de conjuntos (é parte de)

Numa agregação, o número de classes que compõem a. mesma é dito ser o grau da agregação. Pela própria definição de agregação, é possível estabelecer relacionamentos entre as classes componentes, de tal forma que seja possível relacionar um elemento de uma. classe em n elementos de uma outra classe. A cardinalidade mínima de uma classe componente se refere ao número mínimo de mapeamentos que cada elemento da mesma pode ter numa agregação. Analogamente, a cardinalidade máxima corresponde ao número máximo de mapeamentos que cada elemento da classe pode ter no relacionamento.

GENERALIZAÇÃO é a abstração que define uma relação de subconjunto de duas ou mais classes. Classes são inseridas em uma. hierarquia de especialização, de tal forma que uma classe especializada herda todas as propriedades da classe mais geral a que ela imediatamente se subordina na hierarquia. Matematicamente, correspoilcle ao relacionamento de subconjuntos (é uma).

Observa-se que as três abstrações são independentes, sendo que nenhuma delas pode ser expressa em função das outras, e cada uma provê um mecanismo diferente de estruturação da informação.

Para que um modelo conceitual represente bem uma. realidade, é desejável que este seja expressivo no sentido de abrigar o maior número de conceitos possível. Ao mesmo tempo, ele deve ter um entendimento simples, e ser minimal. A mini-

malidade é alcançada se cada conceito presente no moclelo possui um significado distinto em relação a. todos os outros conceitos.

Esquemas concebidos à luz de um moclelo conceitual representam uma especificação formal de dados. Formalidade requer que toclos os conceitos do moclelo tenham uma única interpretação.

Várias disciplinas de modelagem têm sido introduzidas, embora a maior parte delas [6] tenha ficado apenas em protótipos experimentais e não tenham alcançado aplicação prática extensiva. Dentre esses modelos, destaca-se o Modelo Entidade-Relacionamento, que é descrito baseado nas três abstrações apresentadas.

II.2.3 Modelo Entidade Relacionamento

O moclelo E-R [11] foi introduzido por Peter Chen em 1976. E baseado em conceitos simples clue são satisfatórios para grande parte dos projetos convencionais de coinputaclores, e por este motivo se tornou bastante popular. Inicialmente, o modelo E-R abrigava apenas os conceitos de entidade, relacionamento e atributos. Mais tarde, outros conceitos como atributos compostos e generalizações foram introduzidos. O moclelo E-R pode ser expresso através de construções gráficas, mencionadas ao longo do testo. E111 seguicla.,são apresentados, informalmente, os principais elementos do modelo E-R.

ENTIDADE: Representa um o13jeto que pode ser distintamente identificado pelas suas propriedades (ou atributos).

CONJUNTOS DE ENTIDADES: Representam classes de objetos do mundo real. Uma entidade consiste num elemento desta. classe. Por simplificação, chamaremos conjuntos de entidades simplesmente de entidade. Graficamente são representadas por retangulos.

CONJUNTOS RELACIONAMENTOS: Representam agregações de uma ou mais entidades, onde são definidas relações entre estas. Função papel é a terminologia adotada para a identificação das entidades participantes num relacionamento. O grau de um relacionamento consiste no número de entidades que participam do mesmo. Por simplificação, chamaremos conjuntos de relacionamentos de relacionamentos apenas. Graficamente, são representados por losangos. Se R é um relacionamento que relaciona elementos de um conjunto de entidades E1 com elementos desse mesmo conjunto, então R é denominado *auto-relacionamento.*

ATRIBUTOS: Representam propriedades elementares de entidades e relacionamentos. Toda a informação extensional está associada aos atributos. Cada atributo tem uma cardinalidade máxima e mínima, correspondendo ao número máximo e mínimo de valores que um atributo pode possuir em uin dado instante. Quando uin atributo tem mais de um valor ele é dito ser *multivalorado*. Quando ele é expresso por um grupo de atributos que se completam para representar uma informação, ele é denominado *atributo composto*. Cada atributo está associado a um domínio particular, que é o conjunto de valores que poclem ser atribuídos ao mesmo. Graficamente, opcionalmente, aparecem na forma de círculos.

GENERALIZAÇÃO E ESPECIALIZAÇÃO: Generalização é o termo usado quando duas ou mais entidades podem ter algumas de suas características abstraídas e, num nível superior, representadas como uma entidade comum, a qual possui atributos que se aplicam aos diferentes tipos reunidos. O processo inverso se denomina especialização, onde se pode ir do geral para o específico. Hierarquia de tipos é, simplesmente, a. suposição de que um certo subtipo é sempre parte de um tipo superior, embora. nem todos os elementos do supertipo devem estar presentes em algum dos subtipos correspondentes.

II.3 Sistemas Especialistas

Esta seção aborda os conceitos básicos de sistemas especialistas no que se refere às suas aplicações e sub-partes componentes.

Sistemas Especialistas (SE) são "sistemas que utilizam técnicas de Inteligência A tificial para resolver problemas complexos, que comportam razoável quantidade de conhecimento e experiência de especialistas em domínios específicos" [3]. Pode-se constatar a aplicação e o desenvolvimento de sistemas especialistas ein áreas como:

a) diagnose: medicina, reatores, computadores;

b) análise de dados: geologia, química, determinação de causa-efeito em medicina;

c) análise: circuitos elétricos;

d) projeto: circuitos integrados, configuração de sistemas de computadores;

e) planejamento: robótica, planejamento de atividades;

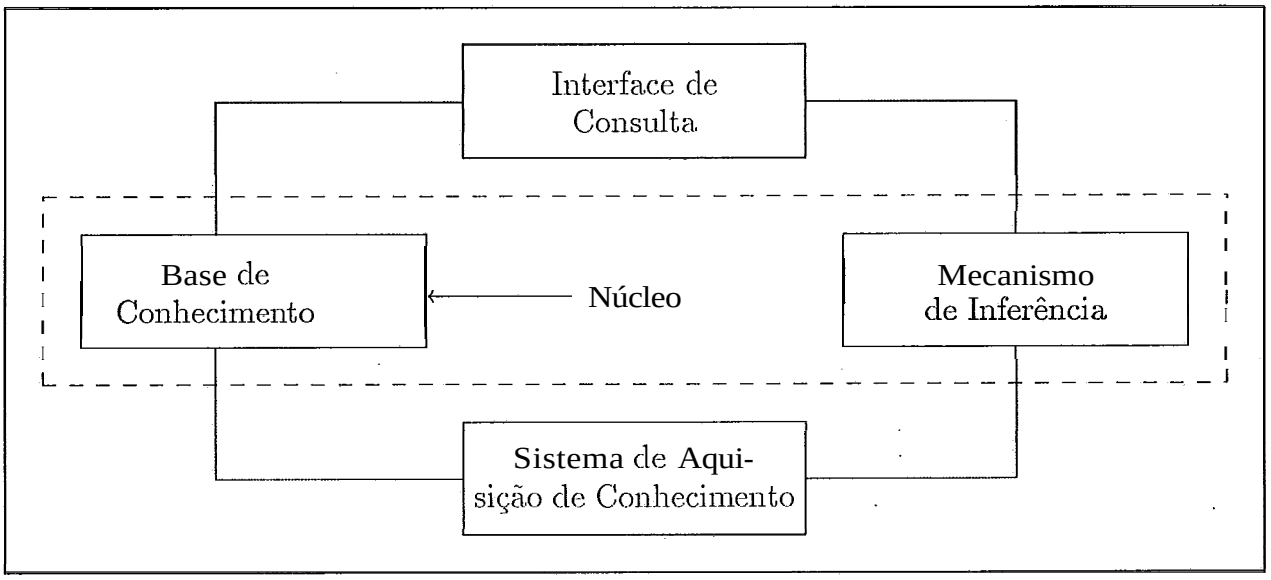

Figura 11.1: Diagrama de Blocos Simplificado dos Componentes de um Sistema Especialista

f) instrução: medicina, matemática;

g) aquisição de conhecimentos;

h) assessoria inteligente: área militar, medicina., energia nuclear;

i) automação de fábricas;

j) administração de projetos;

l) interpretação de imagens.

Um sistema especialista baseado em regras é composto pelos seguintes módulos (Figura 11.1):

- uma base de fatos,
- uma base de regras,
- um mecanismo de inferência ou lógica do raciocinio,
- uma interface com o usuário,
- um sistema de aquisição de conhecimento.

O sistema especialista como um toclo trata. de uma. classe de problemas, como por exemplo o diagnóstico de uma doença. Para cada paciente é necessário instanciar a base de fatos com um conjunto de assertivas referentes aos claclos pessoais do mesmo, que corresponde a uma instanciação de uma. classe de problemas. Nesta representação, são usadas normalmente redes semânticas, frames e triplas O-A-V.

A base de regras basicamente representa informações referentes à experiência do especialista. Este tipo de informação é normalmente espresso na forma de regras de produção. A composição da base de fatos e regras é denominada base de conhecimento. Nesta, são armazenados o conhecimento factual ou extensional, e o conhecimento heurístico ou especializado do domínio. Caracteriza-se por um alto grau de incompleteza, pois em geral não é possível capturar num modelo toclo o conl-ieciinento a. respeito de um domínio [13].

O sisteina de aquisição de conhecimento constitui o meio pelo qual são inseridas e retiradas informações, e testada a consistência. da base de conhecimento. E neste módulo que se faz a interação com o engenheiro do conl-iecimento, ou seja, o especialista de um problema de domínio específico.

Para manipular o conhecimento armazenado a. fim de deduzir e concluir fatos, definiu-se um conjunto de algoritmos que fazem com que a base de regras atue sobre a de fatos, modificando-a (inserindo novos fatos) até que, presumivelmente, se infira um fato que constitui uma resposta. ao problema em pauta [14]. O mecanismo de inferência deve ser capaz de manipular o conhecimento contido nas bases, seja qual for o domínio em questão, de uma forma. chamada preservadora de valores verdade, e aderente ao método de representação adotaclo. Cada algoritmo de busca corresponde a um tipo de motor de inferência, ou lógica de raciocínio. O funcionamento básico de um algoritmo de busca consiste em avaliar as regias de produção em função da base de fatos. Quando o algoritmo se depara com informações indefinidas, a interface com o usuário obtém do usuário, através de perguntas, os dados pendentes.

A interface com o usuário permite um diálogo entre o sistema. especialista e o usuário, de forma que o sistema possa. postulas perguntas a. respeito de informações indefinidas, dar respostas e, analogamente, que o usuário possa exigir explicações sobre como o sistema alcançou determinadas conclusões.

Como pode ser percebido na figura 11.1, o núcleo ou (shelí) de um sistema especialista é constituiclo pela base de conhecimento e pelo mecanismo de inferência. O sistema especialista é obtido quando o engenheiro do conhecimento, através do módulo de aquisição, preenche a base de regras e fatos e, subsequentemente, o sistema como um toclo é utilizado por um usuário casual, através da interface como

o usuário [15].

A nomenclatura *conhecimento* é empregada em sistemas especialistas devido aos seus aspectos dedutivos, e pela armazenagem de informação heurística, fruto de um saber especializado. Para representar o conhecimento inerente à base de fatos foram definidos diversos modelos simbólicos. A escolha de um determinado simbolismo irá depender da espécie de informação a ser manipulada. Quanto mais o inoclelo conseguir captar informações semânticas, mais fiel será a representação. Este processo de traduzir a semântica da informação para um modelo simbólico é conhecido como abstração. Esta técnica é empregada de forma semelhante em bancos de dados, na modelagem de dados, e em programação orientada a objetos. As sub-seções seguintes apresentam alguns modelos de representação do conhecimento utilizados no desenvolvimento de sistemas de *shells*.

11.4 Representação do Conhecimento

Representação do conhecimento, no contesto de IA, consiste basicamente num conjunto de convenções sintáticas e semânticas que torna possivel a descrição de objetos, relacionamentos, procedimentos e meta-conhecimento. A sintaxe especifica os símbolos que podem ser utilizados e como estes podem ser combinados. A semântica traduz o significado incorporado pelos simbolos e pela combinação destes [16].

Na área computacional, em geral, emprega-se as noções de representação do conhecimento em diversas áreas quais sejam: linguagens de programação, estruturas de dados e bancos de dados. Em Inteligência Artificial, é necessário obter modelos que possibilitam a manipulação de informação incompleta. Isto é, quando nem todo o conhecimento necessário está presente no início do processamento. Neste caso, o sistema precisa representar e utilizas novos conhecimentos a partir dos existentes, e/ou obtê-los externamente ao sistema.. Na primeira situação, o conhecimento é obtido através de mecanismos de inferência; na segunda, através de sensores, perguntas ao usuário, bancos de dados, etc.

Assim, ao se escolher uma forma de representação, deve-se escolher uma que seja suficientemente flexível para lidar com novos conhecimentos, tomando como base as seguintes características desejáveis [2]:

- adequação representacional: capacidade de representar os diferentes tipos de conhecimentos necessários,
- adequação inferencial: capacidade de manipular as estruturas de representação

de forma a obter novas estruturas correspondentes ao conhecimento inferido,

- eficiência inferencial: capacidade de incorporar informações que possibilitam direcionar os mecanismos de inferência,
- eficiência aquisicional: capacidade de adquirir novas informações com facilidade.

Em seguida, são apresentadas e avaliados os principais inoclelos de representação do conhecimento utilizados em Inteligência Artificial.

II.4.1 Redes Semânticas

Diversos modelos de representação do conhecimento baseiam-se nos conceitos encontrados nas redes semânticas, por ser este um modelo genérico, sendo, atualmente, o que mais se aproxima clos modelos desenvolvidos na área de cognição [16].

Uma rede semântica consiste de um grafo no qual os vértices representam objetos do mundo real, eventos, e os arcos simbolizam relações entre os vértices. Estas relações são do tipo: é uma., é parte de, pertence a, etc, expressando abstrações de generalização, especialização, agregação e classificação, já descritas.

Em termos de expansão, as recles semânticas são razoavelmente flexíveis. As inclusões que objetivam ampliar o conhecimento contido na rede não trazem grandes dificuldades. Nos casos onde as inclusões definem exceções sobre objetos que pertençam a uma generalização, é possível que as excecões definidas contradigam algumas propriedades previamente estabelecidas na generalização. Assim, deve haver mecanismos que a. cada inclusão verifique a. consistência da recle.

Em termos físicos, os vértices de uma. rede semântica são expressos, normalmente, por ponteiros. Se, por um lado, esta caracteristica é vantajosa por acelerar o processamento, por outro, a. perda. de um ponteiro pode trazer consequências desastrosas ao processo de dedução.

Uma vez que inexistem formalizações sobre a estrutura de recles semânticas, também não existem métodos formais e gerais de inferência. Alguns sistemas utilizam métodos de raciocinio chamados *matching* juntamente com derivações inerentes às propriedades de generalização e especialização. *Matching*, no contexto de recles semânticas, consiste em, a partir de uma pergunta feita à rede, montar o fragmento de rede correspondente àquela pergunta. Este fragmento é então sobreposto à rede para a localização clos vértices correspondentes. Uma vez encontrados, responde-se à pergunta..

II.4.2 Triplas (3-A-V

Na forma mais simplificada, uma rede semântica pode ser expressa ein função de triplas O-A-V (Objeto-Atributo-Valor). Neste caso, os olsjetos expressam conteitos do mundo real, os atributos constituem as propriedades de um olsjeto, e o valores caracterizam uma instanciação do olsjeto.

Em geral, este tipo de representação é adequado em áreas restritas do conhecimento, sendo possível estabelecer critérios formais de representação. Dada a simplicidade desta forma de representação, é possível a manipulação de crenças.

Normalmente, os objetos não possuem relações entre si, mas em algumas situações é conveniente organizar olsjetos através de relações hierárquicas. Este tipo de organização se clá quando o conhecimento pode ser clecoinposto em sub-objetos, ou seja, quando um atributo de um objeto envolve outro olsjeto. Um exemplo desta hierarquização consiste na descrição do corpo humano, onde para cada membro existe um conjunto de sub-membros, expressos também como olsjetos. Estas relacões hierárquicas são expressas na forma. de árvores, onde cada vértice de um nível inferior representa uma sub-parte do vértice superior. A raiz é empregada como ponto de partida para dedução e aquisição de informação [9].

II.4.3 Frames

Em representação do conhecimento, *frames* são clescritores de olsjetos, e sintaticamente se assemelham a tabelas de duas colunas. Na primeira, armazenam-se os nomes dos atributos que descrevem o objeto, e na segunda informa-se como o valor do atributo é obtido (*slots* ou unidade de informação).

A idéia subjacente consiste em prover meios para representar comportamentos complexos, como percepção visual e diálogo em linguagem natural, onde diversas situações são analisadas. Para isso, os frames são organizados na forma de grafos hierárquicos onde cada vértice é um frame. Nesses grafos, são definidos frames genéricos e particulares. O genéricos representam o conhecimento que o sistema possui *a prioi*. Os particulares, selecionados e preenchidos em tempo de processamento, representam o conhecimento necessário para se comportas frente a uma dada situação. As unidades de informação de um frame particular são preenchidas pelos seguintes mecanismos:

- por omissão, no caso de não haver informações que indiquem o valor pelo qual uma. unidade de informação pode ser preenchida, um valor *default* é assumido,

- pela especialização, as informações sobre o pai são herdadas pelos fraines filhos,
- pelo acionamento de um conjunto de regras que representam uma situação desejada,
- pela chamada a um procedimento,
- pela referência a outros frames, slots e sub-slots.

Após a escolha de um frame particular para representas uma situação, preenchem-se os detalhes referidos pelas unidade de informação. Pode ocorrer que o frame tenlia sido mal escolhido. Assim, muitos sistemas baseados em frames possuem mecanismos que permitem escolher o frame mais adequado quando não há uma correlação razoável entre as unidades de informação do frame escolhido e a situação real.

As propriedades de generalização, especialização, flexibilidade e indexação, vistas ein redes semânticas, também se aplicam aos frames. O que difere a primeira forma de representação da segunda. é o fato desta reunir em uma única estratégia duas formas complementares de armazenar fatos: procedimental e cleclarativa. Esta extensão, por um lado, enriquece a. representação e, por outro, torna a implementação mais difícil.

II.4.4 Regras de Produção

Regras de produção são formas procedimentais para representar o conhecimento lieurístico referente à experiência de um especialista. Devem estar associadas a uma base declarativa com informações que permitam a avaliação da regra. Uma regra de produção é composta por uma premissa. e uma. ação. Se a premissa for verdadeira a ação da regra é disparada. Normalmente, a. ação consiste numa atribuição à base de fatos. Uma premissa é composta por literais, que denotam expressões sobre objetos da base de fatos, que são conectados por operadores lógicos. Na avaliação de uma premissa os literais devem ser testados.

Sistemas que utilizam regras de proclução são chamados sistemas de proclução. Tais sistemas operam em ciclos que executam as fases de mat*ching*, identificação de quais produções têm a premissa satisfeita, resolução de conflito, seleção de uma produção, e execução da ação.

A naturalidade de expressão do conhecimento, a uniformidade e modularidade da base, são algumas características que estimularam o uso dos sistemas de

Se A e B então C

Se C e D então I

Se D ou E então H

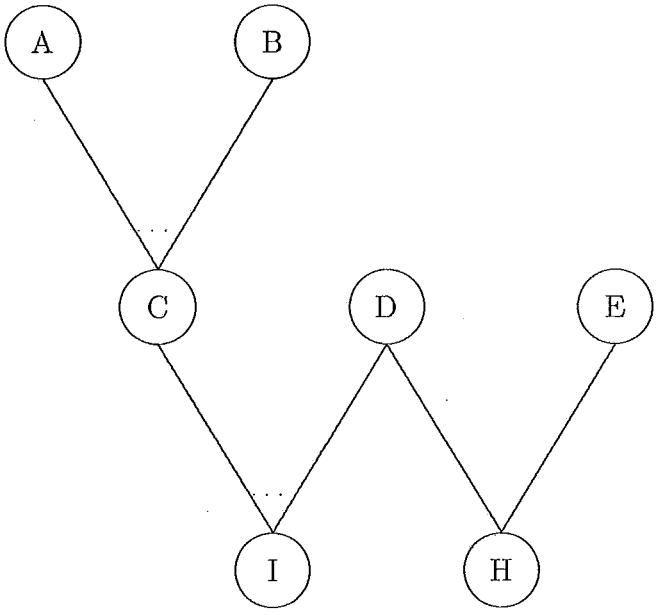

Figura 11.2: Exemplo de Encadeamento de Regras num Grafo E/OU

produção. Como desvantagem, podemos apontas a dificuldade de seguir o fluso de controle. Isto se deve ao fato, dos algoritmos de busca serem menos aparentes, isto é, o conhecimento algorítmico não é facilmente representável nesta forma. Percebe-se, também, que na medida em que um sistema cresce, a manutenção da modularidade fica comprometida., e os ciclos de *matching*, seleção e ação se tornam ineficientes, prejudicando o desempenho do sistema..

Desta forma, algumas técnicas para aumentar a eficiência foram desenvolvidas, das quais poclemos destacar a utilização de meta-produção e o estabelecimento de estratégias de seleção de produções [2].

II.4.5 Representação Interna de Regras

Do ponto de vista interno, a representação de regras se dá, normalmente, por meio de grafos E-OU. Uma regra é expressa por tantas arestas quantos literais ela possuir, todos conectados ao vértice que simboliza a ação. Para indicar que um

grupo de arestas participam de uma mesma regra. na forma de conjunção, utiliza-se um arco de circunferência unindo as arestas componentes conforme ilustrado na figura II.2. Nos casos de disjunção a representação é normal. A hierarquia no grafo é obtida à medida que um vértice representa uma ação, e ao mesmo tempo uin literal. Isto significa que o valor obtido por um objeto numa ação será utilizado num literal de uma outra regra.. Assim, o grafo representa também o encadeamento clas regras [17].

Num grafo E-OU, encontramos três categorias de vértices: os iniciais, intermediários e terminais. Os iniciais representam os literais que referenciam objetos que apenas aparecam em premissas de regras, ou seja, tais objetos nunca estão sujeitos a atualização. Os terminais são as ações em que os objetos referenciados, ein termos de atualização, aparecem apenas em ações de regras, ou seja, tais objetos nunca são consultados. Qualquer nó que não seja. inicial ou terminal é considerado intermediário.

Ein tempo de execução, o grafo E-OU é percorriclo de modo a procurar satisfazer as objetivos do sistema, especialista. A forma como este grafo é percorriclo irá determinar o tipo de motor que estiver sendo utilizado. Na utilização do motor baseado em *Encadeamento* Regressivo clas regras, o grafo será percorriclo a partir dos nós terminais que obtêm o valor objetivo, em direção aos nós iniciais. Quando o motor empregado for do tipo *Encadeamento* Progressivo, o grafo será percorriclo dos nós iniciais, testando os literais, em direção aos terminais que determinam o valor objetivo.

Existe uma. terceira. categoria de encadeamento, denominada *Misto*, em que o grafo E-OU é percorriclo nos dois sentidos. Inicialmente, o algoritmo trabalha de forma progressiva, inferindo novas informações até que não seja mais possivel prosseguir. A partir deste ponto o algoritmo funciona regressivamente, buscando novos dados e, se necessário, interage com o usuário. Uma vez obtidas as informações necessárias, o algoritmo volta a. funcionar de forma progressiva. Este processo de alternação de encadeamento se repete até que seja atingida alguma conclusão.

O encadeamento mais utilizado é o regressivo, e ocorre em situações onde o número de regras que concluem a mesma. ação é pequeno. Caso contrário, recomenda-se a utilização de encadeamento progressivo ou misto.

II.4.5.1 **Raciocínio** Aproximado

Uma clas principais características de sistemas especialistas é a capacidade que tais sistemas possuem de inferir conhecimento a partir de conhecimentos anteriores, e

adquirir novos conhecimentos. Essa facilidade, no entanto, permite o aparecimento de problemas como o gerenciamento da inferência em face de conhecimento insuficiente, e aquisição de novos conhecimentos quando tais conhecimentos contradizem o anterior.

O tratamento desses problemas constitui o que chamamos de Raciocínio Aproximado. Existem, basicamente, duas classes de métoclos que tentam resolver estas questões. A primeira classe consiste nos métodos não numéricos, que procuram estabelecer um formalismo para a incerteza. As lógicas não monotônicas tem destaque nesta classe, e aspectos como o raciocínio por default, circunscrição, abdução e sistemas de manutenção da verdade [2] tem sido usados.

A segunda classe consiste nos métodos numéricos, que procuram quantificar a incerteza. Pode-se destacar a teoria. da probabilidade, teoria da confirmação, utilizada no Mycin, teoria. da evidência. e teoria. da possibilidade [9].

11.5 Integração de Sistemas Inteligentes e Bancos de Dados

Sistemas baseados em conhecimento e SGBD possuem características arquiteturais semelhantes. Ambos baseiam-se na escolha de estruturas apropriadas para representas o conhecimento e mecanismos eficientes para responder às perguntas ou deduzir novos conhecimentos a partir dos já existentes. No entanto, sistemas baseados em conhecimento enfatizam dedução e representações complexas, enquanto que em SGBD a ênfase se dá na eficiência de acessos e na manipulação de estruturas, que são persistentes e armazenam informações de um universo específico [18].

É inegável que a. integração destas duas tecnologias traz benefícios para ambas. A contribuição fundamental da tecnologia de IA consistirá na obtenção de modelos semânticos mais ricos, isto é, um entendimento mais claro a respeito do conhecimento respresentado. Por outro lado, a contribuição da tecnologia de bancos de claclos propicia a, habilidade de se realizar acessos eficientes a lsases de conhecimento rolsustas e melhor estruturadas, com auxílio de controles de concorrência, segurança *e* proteção de claclos, acesso otimizado e gerenciamento de memória secundária [19]

Observando a figura I.1 podemos confirmar a. tendência de união de esforços destas áreas a fim de se alcanças sistemas gerenciadores de lsases de conhecimentos num sentido mais abrangente. Alguns esforços já. foram feitos para obtenção de

tais sistemas. Esta seção apresenta. os diversos aspectos na integração desta áreas relatando alguns trabalhos já realizados.

II.5.1 Bancos de Dados Dedutivos

Bancos de Dados Dedutivos [20] caracterizam a. área de pesquisa que visa a definição de bancos de dados por meio de formalismos lógicos. Pelo fato do modelo relacional ser o que mais se aproxima. da lógica de primeira ordem, maior esforço tem sido dedicado ao emprego cleste modelo nesta área. Neste sentido, existem duas formas [21] de se visualizar o modelo relacional através da lógica de primeira ordem. O primeiro enfoque consiste na concepção Teoria. Modelo . Neste abordagem, a lógica é usada como linguagem de consulta e o banco de clados relacional é um modelo de uma teoria de primeira, ordem. De fato, as expressões lógicas podem ser traduzidas em operações da algebra relacional.

O segundo enfoque consiste na Abordagem Declutiva.. Nesta visão, o mundo é representado através de uma teoria de primeira ordem com ou sem igualdade [21]. O banco de dados é composto por um conjunto de axiomas próprios e um conjunto de teoremas da primeira ordem. Os axiomas representam informações positivas, negativas e de dedução, bem como restrições de integridade. Os teoremas constituem, implicitamente, informação derivável. Consultas são tratadas como teoremas que devem ser provados com base nos axiomas e teoremas que constituem a teoria do banco de claclos. Como exemplos da abordagem dedutiva, destacamos o DATALOG [20] e o LDL (Logic-Basecl Data Language) [22].

Algumas dificuldades foram identificadas em sistemas que implementam a abordagem dedutiva,, das quais destacaremos algumas. A linguagem de programação baseada em lógica, em geral, é restrita, conforme visto em II.4.1, não oferecendo suporte para modularização, encapsulamento, e facilidades para construção de interfaces com o usuário. Em geral, tais sistemas são fechados, o que dificulta a comunicação com outros sistemas, contrariando a tendência atual de integração de sistemas heterogêneos. Por último, não existe atualmente um número significativo de aplicações que fazem uso do grande poder de expressão dos sistemas cleclutivos.

II.5.2 Mecanismos Dedutivos para Banco de Dados

Existem inúmeras situações onde mecanismos cleclutivos são interessantes de serem acoplaclos aos bancos de clados. Podemos identificar algumas aplicações que são abordadas a seguir [23].

Utilização de capacidade dedutiva nas interfaces com o usuário, principalmente na definição de vistas. No sistema KARMA [24], foi definida. uma interface inteligente que assiste ao usuário na formulação de consultas para um banco de claclos relacional, minimizando o conhecimento que este deve possuir a respeito do esquema do banco de claclos.

Métodos dedutivos também são empregados na otimização de consultas onde, dada uma consulta, o sub-sistema. de dedução identifica. o modo mais rápido de execução. Utilizam-se também mecanismos cleclutivos no gerenciamento de transações para a manutenção clas restrições de integridade do banco de dados. Outra aplicação se refere à obtenção do valor de um atributo virtual por mecanismos de dedução. Isto é, ao se calcular o valor de um atributo, clispasa-se um conjunto de regras para obter um determinado valor.

II.5.3 Bancos de Dados para Sistemas Especialistas

Com o progresso clas técnicas de Inteligência. Artificial, os sistemas especialistas se tornaram ferramentas práticas e amigáveis e atualmente vem ganhanho impulsos significativos nas diversas áreas do saber. Algumas de suas aplicações são muito restritas e por isso não necessitam de grandes bases de informação. Mas na medida em que as aplicações atendem a. aplicações que exigem grandes bases de fatos, como sistemas bancários e financeiros, o número de dados necessários ao processo de inferência cresce substancialmente a ponto de interferir no processamento do sistema especialista. Isto é, o sistema. se torna. lento, ou até busca dados errôneos. Nestes casos, é importante definir estratégias para a. manipulação de bases de claclos em memória secundária, ou então integras o sistema especialista com um banco de claclos previamente desenvolvido. Este tipo de ambiente traz o conforto clas facilidades inerentes aos sistemas especialistas, ou seja,, uma interface de alto n ível e técnicas de cleclução eficientes, e ao mesmo tempo usufrui clas potencialidades ofereciclas pelo banco de dados: gerenciamento de memória secundária, controle de concorrência e acesso [18].

II.5.4 Propostas de Acoplamento

Quatro propostas para estabelecer uma comunicação cooperativa entre componentes dedutivos e de dados num Sistema. Especialista foram iclentificaclas [25]. Paste-se clas faciliclacles iniciais e naturais de obtenção de claclos, evolui-se para um SGBD generalizado dentro do próprio sistema. especialista., implementando um conjunto de procedimentos que simulam o funcionamento de um SGBD. Passa-se para o

acoplamento fraco, onde é definido um canal entre o sistema especialista e o banco de claclos. Finalmente, apresenta.-se o acoplamento forte com um banco de clados que, neste caso, é visto como extensão do sistema especialista. Projetistas podem optar por uma das propostas dependendo do volume de dados, multiplicidade de uso dos claclos, características voláteis clos dados (qüão freqüentemente os claclos mudam), e clos requisitos de proteção e segurança.

Na primeira. proposta, que consiste na manipulação elementar de claclos no contexto de um sistema especialista, todos os dados são mantidos em memória principal e armazenados em estruturas de dados convencionais. Neste caso, rotinas específicas da aplicação para recuperação e modificação de claclos são implementadas.

Na. segunda proposta., que consiste em um SGBD generalizado no escopo de um sistema. especialista, a quantidade de dados armazenados na. base de fatos é grande a. tal ponto de não caber na memória. principal em tempo de execução. Neste caso, a manipulação elementar de dados não é suficiente. Algumas técnicas de gerenciamento externo de arquivos são então implementadas (índices secundários, dicionário de claclos, etc). Esta. solução tende a. produzir sistemas extensos, pois a maior parte das rotinas de gerenciamento de arquivos de um SGBD acabam sendo implementadas, aumentando consideravelmente a. quantidade de código gerado.

Outrossim, dependendo da multiplicidade do uso do banco de claclos e da extensão do mesmo, facilidades de SGBD podem ser tornar necessárias. Tais facilidades, que incluem vistas e janelas dinâmicas, se encontram disponíveis na. maioria clos SGBD relacionais modernos. Outra. facilidade consiste em integrar o sistema especialista com o dicionário de claclos de forma que sejam permiticlas consultas a respeito da estrutura do banco de clados. Este estágio de extensão parece ser normal para os sistemas especialistas que lidam com grandes bases de dados, embora nem todos os sistemas as acessem com o mesmo grau de sofisticação. Passa-se, então para. a terceira. proposta, denominada acoplamento fraco.

Bancos de Dados externos são normalmente robustos, altamente voláteis e utilizados por várias aplicações. Custos de armazenagem e manutenção de consistência proibiriam a duplicação das bases de claclos para. o uso exclusivo do sisteina especialista. O acoplamento fraco se refere à presença de um canal de comunicação entre os dois sistemas, pelo qual é feita. a extração clos claclos do banco de claclos existente para posterior armazenagem numa base de claclos interna do sistema especialista. A partir deste ponto, pode-se acessas os claclos conforme descrito na segunda proposta. Observa-se que são permitidas apenas operações de consulta.

Como exemplo de acoplamento fraco, menciona.-seo sistema MEDCLAIM [31],

que é um sistema de orientação pasa. tratamentos médicos. O sistema especialista armazena 120 regras que traduzem as estratégias pasa. tratamentos inéclicos e critérios de avaliação do quadro cl ínico de um paciente para a obtenção de um diagnóstico. O banco de claclos armazena, em 6000 registros, tanto o conhecimento médico como os dados relativos aos métodos de tratamento para 1000 doenças. No sistema RX [23], também para tratamentos inéclicos, a base de fatos é inicialmente preenchida com o conhecimento de um especialista, e subsequentemente atualizada com claclos provindos de um banco de claclos previamente instanciado.

A principal desvantagem cleste tipo de acoplamento consiste no fato dele não se adequar aos casos onde existem decisões dinâmicas. Isto é, durante a mesma sessão de execução de um sistema especialista, porções armazenadas em blocos de mémoria distintos podem ser referenciadas. Deste modo, acessos extras são solicitados gerando um processo de *swapping* que degrada o desempenho final do sistema. Em adição, se o banco de dados externo é continuamente atualizado, a consistência com o interno pode se tornar rapidamente obsoleta.

Assim, surge a. necessidade do acoplamento forte entre um sistema especialista e um SGBD externo. Neste tipo de acoplamento, o banco de dados externo é visto pelo sistema especialista como uma extensão cleste, sendo ideal para aplicações em tempo real. O modelo funcional genérico de um ambiente que adota o acoplamento forte pode ser visto na figura.11.3 [23]. Observamos que as funções efetuadas pelos processadores especializados podem ser integradas à própria linguagem de consulta do SGBD. Além disso, todos os claclos gerados pelos processadores também podem ser armazenados no banco de dados.

Abordagens cleste gênero enfrentam problemas relativos ao número de chamaclas ao banco de claclos, que em execução pode prejudicar o desempenho do sistema, devido ao tempo de acesso ao banco de dados, ou à complexidade clas consultas, que pode inviabilizar suas traduções para as linguagens de consulta existentes. As considerações mais relevantes na utilização do acoplamento forte consistem no gerenciamento inteligente do canal de comunicação, ou seja, quando e como utilizá-lo, e na. possibilidade de se efetuar operações de escrita no banco de claclos acoplado.

No acoplamento forte existem duas formas para. geração de consultas. A primeira, denominada estática, consiste em utilizas um pré-compilador que realiza a primeira avaliação da lógica do sistema. assumindo que todas as referências ao banco de claclos serão efetuadas, independente da premissa. Desta forma, o mapeamento dos termos que acarretam acessos ao banco de claclos pasa coilsultas ao mesmo, é feito em tempo de compilação, e na execução as consultas são invocadas

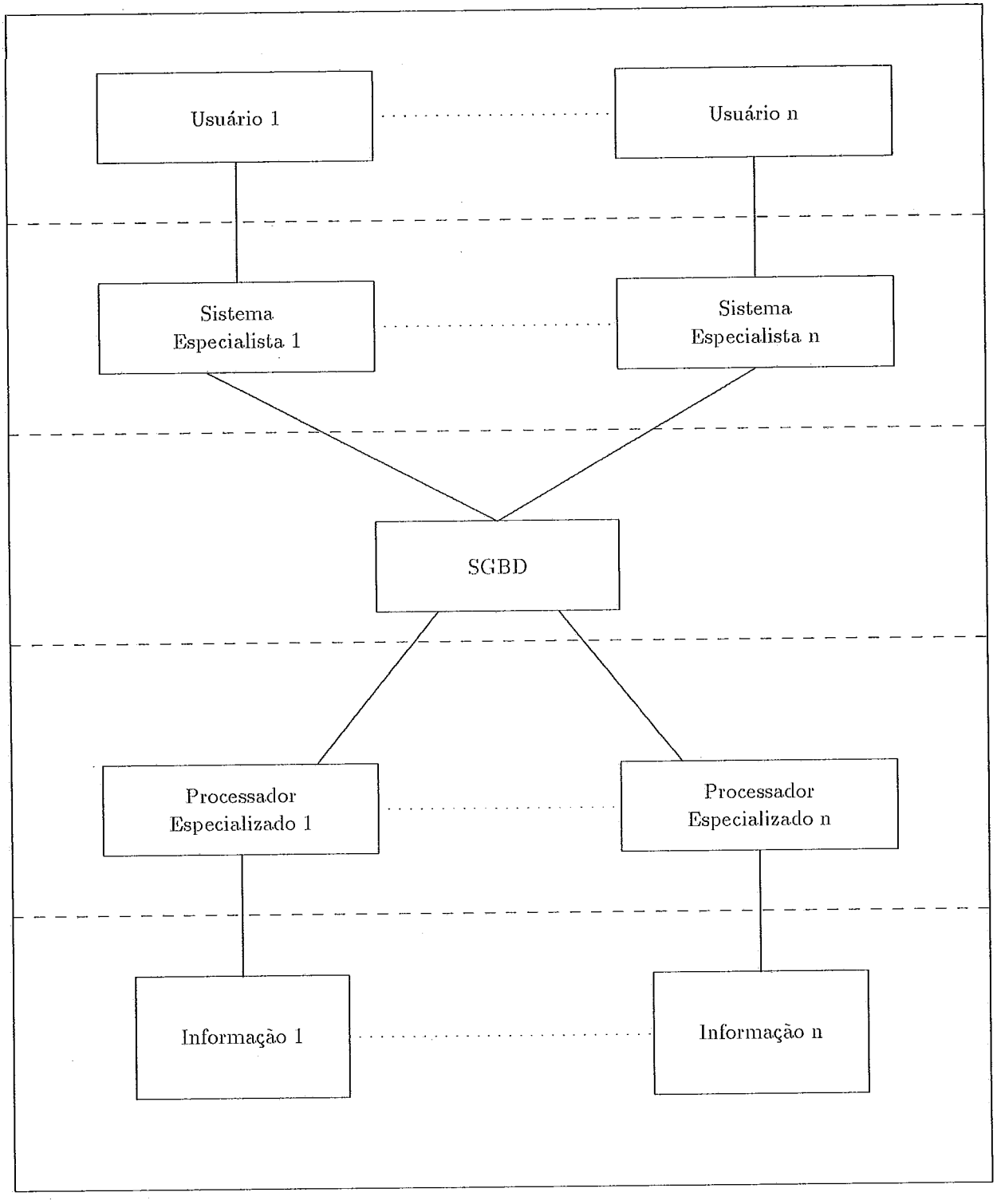

Figura II.1: Modelo Funcional de um Ambiente de Acoplamento Forte

diretamente quando a.regra em questão for disparada. Na segunda forma de implementação, denominada dinâmica, as consultas são geradas em teinpo de execução do sistema., de acordo com a execução das regras. Esta abordagem, apesar de minimizar o código gerado, pode resultar num sistema mais lento em termos de execução.

Destaca-se como exemplo de aplicação de acoplamento forte um sistema de produção automática de mapas [23]. O banco de dados mantém um modelo da superfície terrestre que deve refletir as mudanças ao longo do tempo. O modelo é continuamente atualizado com base nas imagens recebidas via satélite. Estas imagens devem ser analisadas por um sistema.especialista, através de comparações com imagens anteriores armazenadas no banco de dados. De posse dos dados gerados pelo sistema especialista, o modelo original é atualizado. Como sa ída, este sistema imprime um mapa de uma. determinada área., conforme especificado pelo usuário.

II.5.5 Aspectos Arquiteturais de Acoplamento

4 primeira vista,, a arquitetura mais natural no projeto de um acoplamento consiste num sistema de integração uniforme, escrito e manuseado por uma linguagem dedutiva com capacidade de programação, como Prolog, ou então por uma linguagem de consulta com capacidade dedutiva. Contudo, como cada uma destas linguagens é específica para determinados propósitos, é natural que surjam limitações em algumas situações. Além disto, este tipo de arquitetura contraria o propósito de se obter um sistema que usufrua das vantagens de cada sul]-sistema componente.

Considerando estes aspectos, três arquiteturas foram identificadas para o acoplamento de sistemas especialistas e bancos de dados, com base no artigo [25]. A categorização de cada arquitetura é baseada no local do processamento, e no controle da interações entre os sistemas.

A *primeira arquitetura* é totalmente distribuída, tanto no processamento como no controle. Os sistemas são auto-contidos, podendo operar de forma independente e, na medida em que são necessários acessos a dados ou a mecanismos dedutivos, o sisteina complementar é ativado. A interação se realiza.pela troca de mensagens e é caracterizacla por um relacionamento mestre e escravo entre o originador e receptor da mensagem, respectivamente.

Estes aspectos possibilitam a implantação de diversos estilos de aplicações e, devido à modularidade, a portabilidade é naturalmente obtida. Portabilidade se refere à possibilidade de substituição dos sistemas componentes. Contudo, cada

sistema terá. sempre que conhecer as capacidades do oiitro, justamente para ativar as mensagens adequadas. Apesar de oferecer os atrativos mencionados, esta arquitetura, principalmente no escopo do acoplamento fraco, possibilita a duplicação e inconsistência de claclos, conforme visto na sub-seção anterior.

Na *segunda arquitetura*, um clos sistemas assume o papel dominante, ou seja, ele controla o processamento e as interações entre os sistemas componentes. Esta abordagem é utilizada quando, no acoplamento, a. função de um clos sistemas é realçada. Dependendo da forma como for especificado, a. portabilidade pode fica- comprometida pois pode ocorrer que, na. especificação, o projetista tenda a implantar no sistema dominante as solicitações de funções do sistema complementar de uma forma não modular. Observa,-se que, pelo fato de um clos sistemas centralizar o controle, a possibilidade de duplicação e inconsistência de claclos é substancialmente reduzida.

Na terceira *arquitetura*, o processamento é distribuído mas o controle é efetuado por um sub-sistema dedicado, o supervisor. Em essência.,o supervisor realiza todos os procedimentos para o interfaceamento entre os sistemas e gerencia as interações. A principal vantagem se refere à facilidade de inclusão de novos subsistemas, pois os sistemas componentes ficam alheios a. todo processo de tradução de requisições, deixando esta tarefa para o supervisor, Nesta arquitetura, os problemas de duplicação e inconsistência de claclos também podem ocorrer, principalmente no acoplamento fraco.

Em cada uma. das arquiteturas apresentadas, devem existir meios para a. tradução das representações de conhecimento inerentes a. cada sistema., bem como das requisições de transação. No caso de uma. integração que acopla um sistema especialista escrito em PROLOG e um banco de claclos relacional, a tradução é facilitada devido à similiaridade destes modelos [26]. Nas situações onde isto não ocorre, faz-se necessário a realização de traduções intermediárias e, em algumas situações, otimizações nas requisições de transação são fundamentais ao bom desempenho do sistema como um todo.

11.6 Conclusões

Procurou-se, neste capítulo, abordar os principais conceitos relativos à representação de conhecimento e projeto conceitual de banco de claclos, dando enfoque ao modelo Entidade-Relacionamento. Verificou-se a. necessidade emergente de se integrar sistemas baseados em conhecimento e bancos de claclos, tanto no sentido de ampliar o dom ínio de atuação de um sistema especialista., como no de adicionar

facilidades dedutivas às operações próprias de banco de claclos.

Quatro propostas de acoplamento foram apresentadas, onde um enfoque especial foi dado para. os acoplamentos fraco e forte. Abordou-se três arquiteturas de acoplamento, analisando-se as vantagens e desvantagens de cada uma.

Alguns exemplos de integração foram citados ao longo do testo. Observou-se a necessidade de se implementar um tradutor entre os modelos inerentes ao sistemas especialista e banco de claclos. Esta tradução nem sempre é trivial, e em geral o desempenho do sistema como um todo fica comprometido.

Em qualquer modelo de acoplamento ou arquitetura adotada, deve-se sempre levar em consideração a atualização do conhecimento deduzido face à atualização de dados, principalmente no sistemas especialistas que armazenam bases de fatos em memória. secundária. Isto é, considerando que novos fatos foram deduzidos a partir dos dados armazenados no banco de dados, é preciso definir mecanismos que a cacla. atualização revejam os conhecimentos que foram obtidos a partir dos dados atualizados e subsequentemente repitam algumas procedimentos de cleclução.

Outro aspecto a considerar na integração de bancos de claclos e sistemas especialistas é que, atualmente, ambos dão enfoque a abstração conceitual. Se pensarmos num sistema genérico, que englobe as características de banco de claclos e sistemas dedutivos, sem que este sistema implicasse numa integração e sim numa nova filosofia de gerenciamento, ter íamos que pesquisar novos modelos que abrigassem as semânticas inerentes a cacla área, conforme indica. figura. I.1 [23].

Capítulo III

QUICK-SHELL: Uma Proposta de Integração

III.1 Introdução

Com base nos conceitos apresentados no capítulo anterior, este capítulo propõe, de fato, um ambiente de integração. A idéia consiste em especificar uma *shell* O-A-V que seja capaz de interagir com um banco de claclos baseado no Modelo E-R.

Normalmente, o modelo E-R é utilizado na modelagem conceitual de dados, pois possui uma grande faciliclacle de uso devido à intuitividade de seus conteitos. Na construção de bancos de claclos, após a modelagem conceitual, opta-se por um inoclelo lógico que serve de base para a. implementação. No contexto do projeto UNIVERSIDATA, [28], entretanto, a implementação do banco de claclos continua calcada na própria estrutura do modelo E-R. A simplicidade e clareza no entendimento e na manipulação do banco de dados, tanto por parte dos analistas e programadores, como por parte dos usuários, compensa, até certo ponto, as vantagens de otimização que geralmente são obtidas na utilização de um modelo lógico como alicerce de implementação.

Na escolha do modelo de representação do conhecimento para a armazenagem da base de fatos e regras, alguns aspectos foram levados em consicleração. Percebeu-se que um objeto no contexto O-A-V poclia ser visto como um elemento de um conjunto de entidades ou relacionamento. Neste contexto, um objeto comtitui uma instância de um conjunto de entidades ou relacionamento. O atributo se refere a um atributo da instância, e o valor corresponde ao valor do atributo E-R em questão. Esta semelhança é explorada no presente capítulo, e permite que a integração se dê através da definição de termos que englobam de forma natural os elementos O-A-V e E-R.

Inicialmente, este capítulo apresenta o modelo E-R e suas extensões de forma mais detalhada. Em seguida, descreve-se o moclelo O-A-V mais formalmente para facilitar a compreensão do ambiente de integração. Na apresentação deste, define-se termos, literais, regras, etc, que englobam os elementos O-A-V e E-R. Ein seguida, comenta-se os aspectos arquiteturais do sistema. e a. semântica da linguagem de regras. Para ilustrar o ambiente proposto, apresenta-se um exemplo prático de utilização. Por fim, são relatadas algumas conclusões.

111.2 O Modelo E-R e suas Extensões

Esta seção apresenta em detalhes o modelo E-R suportado pelo ambiente aqui apresentado. Nesta descrição, evitamos o uso de uma linguagem excessivamente formal a não ser quando necessário para a. posteiior compreenssão do ambiente QUICK-SHELL. Maiores detalhes a respeito da formalização do modelo E-R podem ser vistos em [27].

Elementos de Esquemas E-R

a) Constantes, identificadas por $c_1, ...c_n$ que expressam valores tais como números, cadeias alfanuméricas e outros objetos cujas interpretações não variam.

b) *Conjuntos* de Entidades e *Relacionamentos*, aqui identificados por E_1, $E_2, \ldots$ e R_1, $R_2, \ldots$ respectivamente. A cada relacionamento R está associado um número inteiro n. Este corresponde ao número de entidades que participam do relacionamento. $E_1, \ldots, E_n$ é a. lista. de entidades que participam de R, onde n é dito ser o grau de R.

c) Para cada relacionamento R há ainda um conjunto de papéis associados, da forma $\{p_1, \ldots, p_n\}$. Cada um desses papéis é de fato uma função da forma

$$p_i : R \rightarrow E_i$$

Ou seja, p_i é uma função que, aplicada a um elemento de um relacionamento, retorna um elemento da i-ésima entidade a ele associada. As funções papel são utilizadas para identificar quais os elementos das entidades participantes que estão associados nos relacionamentos.

d) *Domínios*, aqui identificados como V_1, V_2, ... são conjuntos de constantes e são utilizaclos para especificar os domínios dos atributos. Exeinplos são conjuntos de strings, inteiros, reais.

e) *Atributos* recebem a notação α_1, α_2, ... Cada atributo a é uma função da forina $a : X, \mathbf{I} \rightarrow$ V, onde X pode ser um conjunto de enticlacles ou uin relacionamento, **I** é um inteiro e V um domínio. Atributos podem ser multi-valorados ou simples, dependendo do número máximo de ocorrências associado a eles. Para atributos simples, este valor é **1** e $>$ **1** para multi-valorados. O número inteiro **I** indica qual a ocorrência que a função relativa. ao atributo deve retornar.

f) *Restrições de Integridade* são fórmulas bem-formadas e sua aplicação será comentada com mais detalhes logo adiante.

III.2.1 Propriedades de um esquema E-R

Algumas construções do Modelo E-R utilizadas caracterizam propriedades do esquema e, por consequência poclem ser consideradas como restrições de integridade. Estas são: *chaves, cardinalidades dos relacionamentos* e *entidades fracas.*

Chaves determinam combinações de atributos que unicamente identificam elementos de conjuntos de enticlacles e/ou relacionamentos. Para garantir que as chaves sejam corretamente implementadas é necessário respeitar a restrição de unicidade de seus valores.

As funções $Cmax$ e $Cmin$, identificam, para cada relacionamento e suas entidades associadas, as combinações permitidas no que refere ao número de elementos associados. Assim, podemos ter combinações $\mathbf{1} : 17, 12 : m$, etc. Novamente, a manutenção das cardinalidades recai no caso geral de manutenção de restrições de integriclade.

Entidades fracas são um caso especial de conjunto de entidades, tal que a identificação e existência de seus elementos dependem das instâncias de outras entidades. Como nos casos acima, isso pode ser considerado como uma restrição de integridade.

Ordens

Acesso ordenado não é suportado pelo Modelo E-R em sua definição original. Isto é devido à função inicial do inoclelo de servir como uma representação conceitual de dados, não havendo suporte para implementações físicas. Essa, entretanto, é uma importante característica dos bancos de dados e é extensivamente utilizada durante a programação. A obtenção de acesso ordenado a. bancos de dados geralmente envolve:

a) um domínio, que é um conjunto de elementos de entidades/relacionamentos que clevem ser ordenados;

b) uma expressão de seleção, que designa que elementos do domínio devem aparecer na ordem final;

c) os determinantes da ordenação, que são atributos ou outros termos do domínio da ordenação que devem determinar o critério de ordenação a ser seguido.

Uma orclem pode ser parcial ou total. A ordem é total quando todos os elementos do conjunto de entidades ou relacionamento, que é objeto da orclem, podem ser alcançados através da. ordem.

A ordem é parcial quando apenas alguns elementos de um conjunto de entidades ou relacionamento, que é objeto da ordem, podem ser alcançados através da orclem.

Atributos referenciais

São atributos que retornam referências a outras entidades do esquema. Este conceito é importante e extremamente útil, uma vez que elimina, em muitos casos, a necessidade de introduzirmos relacionamentos $1:n$ fictícios, que são melhor expressas com o presente conceito. Formalmente, atributos referenciais são funções da forma

$$\beta : E_1, I \rightarrow E_2$$

similarmente à definição no item e acima. Apenas, o contra-domínio de β é uma coleção de entidades E_2 e não um conjunto de valores V como naquele caso. Assim, os atributos referenciais, uma vez aplicados, retornam o ***i-ésimo*** elemento de uina coleção de instâncias de um determinado conjunto de entidades.

Analogamente, a função inversa

$$\beta^{-1} : E_2 \rightarrow E_1^*$$

relaciona a entidade de E_2 a uma coleção de elementos de E_1. Neste caso, o acesso aos elementos se faz através de uma ordem.

Generalização e especialização

Generalização é o termo usado quando duas ou mais entidades podem ter algumas de suas características abstraídas *e*, num nível superior, representadas como uma entidade comum, a qual possui atributos que se aplicam aos diferentes tipos reunidos. O processo inverso se denomina especialização, onde se pode ir do geral para o específico. Em todos os casos de especialização, a entidade que é subtipo herda os atributos de seu supertipo. No nosso caso, uma dada entidade pode ter apenas uma outra entidade como seu supertipo. Também não é permitido haver recursão: se a rede de generalizações fosse representada como um grafo, não haveria ciclos.

III.2.2 Operações sobre o Modelo E-R

Com o propósito de permitir operações de acesso, foram definidas quatro classes de operações:

a) avaliação de expressões lógicas;

b) avaliação de termos e funções;

c) operações de busca;

d) operações de atualização da base de dados.

No primeiro caso, aparecem as operações que correspondem à avaliação de expressões lógicas, especialmente o caso t E X, onde é testada a condição de pertinência do termo t em relação ao conjunto X. O resultado de um teste dessa natureza é um valor Óooleano, verdadeiro ou falso.

A avaliação de termos corresponde à busca. do valor de expressões formadas por atributos ou funções papel. Por exemplo, se a é conhecido, a expressão

$$nome(x)$$

pode ser avaliada se nome for um atributo do conjunto de entidades que é domínio para a variável x.

Operações de busca podem ser de dois tipos básicos: por chave e por orclem. No primeiro caso, dado um conjunto de valores que caracterizam uma chave de uma entidade ou relacionamento, é possível localizar o elemento ein questão. Na apresentação do ambiente de integração a. ser visto na seção III.4, este tipo de operação foi especificado como uma. função chave.

No segundo caso, é possível seguir certas ordens no acesso aos elementos de conjunto de entidades ou relacionamento. Por exemplo, observando o esquema E-R definido no Apêndice A, constata-se que é possível ter acesso aos elementos das entidades de COMISSÃO pela ordem alfabética. As ordens são definidas sobre o diagrama E-R e, quando da geração do banco de dados, são criadas as estruturas necessárias para sua manutenção e acesso.

Operações de atualização correspondem basicamente à inclusão e exclusão de elementos em entidades e relacionamentos e à alteração de atributos e funções.

Maiores detalhes sobre a utilização clas operações são vistas na descrição da implementação, no próximo capítulo.

III.3 Triplas O-A-V e Regras de Produção

Esta seção aborda com mais detalhes o modelo O-A-V e regras de produção. A partir clas definições aqui apresèntadas, é feita a proposta de integração que aparece na próxima seção.

No' presente contexto, a seguinte notação é utilizada para expressar objetos, atributos e valores:

Objetos: o, o_1, o_2...
Atributos: a, $\alpha_1, \alpha 2$...
Valores: v, $\upsilon_1, \upsilon_2, ...$

Na realidade a notação a no contexto O-A-V deveria.ser diferente da empregada no escopo E-R, mas por questões de didática optou-se por adotar o mesmo símbolo em ambos os casos.

A semântica de (o, a, v) é: o atributo a de um objeto o assume o valor v. De outra forma, pocleinos vislumbar os atributos como se fossem funções que aplicadas a objetos retomam valores. Assim, pode-se considerar a atribuição $\alpha(o) = \mathrm{v}$ equivalente à existência da tripla (o, a, v). Estabelece-se, então, a seguinte definição:

Definição **1** Uma *base* de fatos $\mathcal{B}$ é *constituída* por *um* conjunto *de* triplas *O-A-* V da forma (o, a, v).

Regras

Regras são compostas de ações e literais, e a seguinte notação é utilizada:
Regras: R, $\mathcal{R}_1, ..., \mathrm{R}$,
Ações: A, $\mathcal{A}_1, ...\mathrm{A}_n$.
Lite ais: C, $\mathcal{L}_1, ...\mathrm{C}_n$.

Uma regra R, toma a seguinte forma,:

$$\mathcal{A} \Leftarrow \mathcal{L}_1, ..., \mathcal{L}_k$$

O lado direito da regra corresponde à premissa, composta por literais, e o lado esquerdo corresponde à *conclusão*, que está associada a uma ação.

Literais

Literais são expressões da forma:

$$t_1 \theta t_2,$$

onde t_1 e t_2 são termos e θ um símbolo relacional. Como exemplo, menciona-se: $\alpha_1(o_1) = \alpha(o_2)$, $\alpha(o) \leq 10$, e $\alpha(o) \neq sen(90)$.

Ações

Uma. vez que a premissa. da regra. é avaliada como verdadeira, sua conclusão é ativada. Normalmente, isto equivale à execução de algum procedimento. O mais comum provoca a atualização de uma tripla na base de fatos, expresso por:

$$\alpha_1(o_1) \leftarrow \alpha_2(o_2)$$

ou

$$\alpha(o) \leftarrow \upsilon$$

Termos

Uin termo pode ser:

a) uma expressão da forma $\alpha(o)$, onde a é um atributo e o um objeto.

b) um símbolo de função f de grau n seguido por uma lista da forma. $(t_1, ...t,)$, onde t_i é um termo.

c) uma constante c, conforme definido no item *a* da seção *111.2.*

Avaliação de Regras e Literais

A avaliação de uma regra depende da avaliação de cada literal componente. Definiu-se a função g que retorna um valor no domínio {0,1,indefinido}, correspondendo aos valores falso, verdadeiro e a indefinição do valor, respectivamente. Esta função se aplica a literais e regras, a as avaliações se dão pelos seguintes critérios:

Definição 2 *Seja* L *um literal da forma* $t_1 \theta t_2$. *A avaliaçào de L é:*

$$g(\mathcal{L}) = \begin{cases} 0; & \text{se } t_1 \theta t_2 \text{ é falso;} \\ 1; & \text{se } t_1 \theta t_2 \text{ é verdadeiro;} \\ \textit{indefinido}; & \text{se } t_1 \text{ e } t_2 \text{ são incompatíveis} \end{cases}$$

Definição 3 Seja R uma regra da forma

$$\mathcal{A} \Leftarrow \mathcal{L}_1, ..., \mathcal{L}_k,$$

$k \geq 0$. Então a avaliação de R é:

$$g(\mathcal{R}) = \begin{cases} 0; & \text{se } g(\mathcal{L}_i) = 0 \text{ para algum i E } [1, \text{k}]; \\ 1; & \text{se } g(\mathcal{L}_i) = 1 \text{ para todo i E } [1, \text{k}]; \\ \textit{indefinido}; & \text{se } \exists i \text{ E } [1, k], \text{ tal que } g(\mathcal{L}_i) \text{ é indefinido;} \end{cases}$$

Fatores de Certeza

As triplas O-A-V podem carregas fatores de certeza. [29] que correspondem à probabilidade dos fatos serem verdadeiros ou falsos. Como consequência, os literais e regras também possuem fatores de certeza.. Como primeira experiência de integração, optou-se por não implementar fatores de certeza. Assim, os conceitos relativos não são abordados.

111.4 O Ambiente de Integração

Pela definição ??, observa-se que a noção de triplas O A-V pode ser empregada para os dados armazenados em um banco de dados. Nesta ótica a base de dados do banco de dados é constituída, por um conjunto de triplas O-A-V (o, a, v), onde o se refere a um elemento de um conjunto de entidades ou relacionamento em questão, a corresponde ao atributo, e v consiste no valor do atributo a. Do

mesmo modo, a semântica da expressão $\alpha(o) = \mathrm{v}$, neste contexto é: o atributo α do objeto o assume o valor v. Observe que uma base de claclos pode ser visualizada como um conjunto de triplas O-A-V, que são agrupadas em classes, agregações, generalização, etc. O que difere o ambiente de banco de dados de uma *shell* é o fato do primeiro possuir mecanismos potentes para. gerenciar eficazmente grandes bases de claclos, enquanto que o segundo se detem em técnicas de manipulação de conhecimento. Percebeu-se, então, que se fossem redefinidos os termos, literais e regras de modo a introduzir os elementos E-R. no escopo O-A-V, e acrescentado mecanismos de quantificação, poderíamos obter um ambiente de integração.

Normalmente, distingue-se duas classes de "conhecimento" em sistemas de segras de produção: a base de fatos e o conjunto de regras. Na QUICK-SHELL, a base de fatos divide-se em dois tipos: fatos representados através de triplas O-A-V e fatos oriundos da base de dados E-R [30]. Os elementos E-R aparecem nas regras na forma de termos, que são elucidados na próxima sub-seção.

III.4.1 Regras e Literais na QUICK-SHELL

Triplas **O-A-V**

A noção de triplas O-A-V não é alterada, permanecendo os conceitos definidos na seção anterior.

Termos

Os termos poclem ser categorizaclos em clois tipos:

TERMOS-E

Denotam instâncias de conjuntos de entidades e relacionamentos, e podem ser das seguintes formas:

a) uma variável v, que tem como domínio um conjunto de entidades ou relacionamento;

11) uma expressão $\rho(t)$, onde ρ é uma. função papel e t é um Termo-E;

c) uma expressão $\beta(t)$, onde β é um atributo referencial e t é um Termo-E;

cl) uma função chave $\Gamma(t_1,...t_n)$, onde t_i é um Termo-C. Dado um conjunto de valores compondo o identificador, uma função chave corresponde a um operaclor da base de dados que retorna a instância correspondente. A cada função chave está associado um conjunto de entidades ou um relacionamento, o qual será tomado por base na busca da instância correspondente.

e) uma constante E-R denotada por ζ, que identifica.uma instância de um conjunto de entidades ou relacionamento;

TERMOS-C

Denotam constantes e podem ser das seguintes formas:

f) uma expressão $\alpha(o)$, onde o é um objeto da base de fatos e a é um atributo de o;

g) um símbolo de função de grau n seguido por uma. lista da forma $(t_1,...,t,)$, onde t_i é um Termo-C. Esta construção permite o uso de funções *standard*, tais como *seno, coseno, trunc*, etc;

h) uma constante c, conforme clescrito na seção 111.2;

i) uma expressão $\alpha(t)$, onde a é um atributo de t, e t é um Termo-E';

Literais

Os literais, no ambiente QUICK-SHELL, permitem o uso de operadores relacionais e testes de pertinência. Testes de pertinência são necessários para verificar a existência de algum elemento num conjunto de entidades ou relacionamento. Os literais são definidos no mais alto nível possível, de modo a. tornar transparente para o usuário as operações específicas do banco de dados. Assim, foram definidas as seguintes formas para um literal C:

a) $t_1 \theta t_2$, onde 0 representa um operador relacional ($<, >, \geq, \leq, \neq$);

[1] O item i, estritamente falando, é da forma $\alpha(t,i)$, onde i referencia o i-ésiino valor da lista de valores de um atributo. O mesmo se aplica ao item c

11) t $\in X$, onde X representa um conjunto de entidades ou relacionamento;

c) $t_1 \in \alpha(t_2)$, onde a representa um atributo multivalorado.

Para os três literais definidos acima, t_i é um termo.

Regras Existencialmente Quantificadas

Regras são compostas de literais e ações. No contesto da QUICK-SHELL, uina regra R é definida conforme visto na seção anterior, isto é, sem variáveis, ou com presença de elementos E-R na forma de variáveis ou constantes, conforme explicitaclo abaixo:

$$\zeta_1 \in X_1, ..., \zeta_m \in X_m$$
$$v_1 \in X_m/O_1, ..., v_n \in X_n/O_n$$
$$\mathcal{A}(\zeta_1, ..., \zeta_m, v_1, ..., v_n) \leftarrow \mathcal{L}_1(\zeta_1, ..\zeta_m, v_1, ..., v_n), ..., \mathcal{L}_k(\zeta_1, ..., \zeta_m, v_1, ..., v_n),$$

ou

$$\zeta_1 \in X_1, ..., \zeta_m \in X_m$$
$$\mathcal{A}(\zeta_1, ..., \zeta_m) \leftarrow \mathcal{L}_1(\zeta_1, ...\zeta_m), ..., \mathcal{L}_k(\zeta_1, ..., \zeta_m),$$

ou

$$v_1 \in X_1/O_1, ..., v_n \in X_n/O_n$$
$$\mathcal{A}(v_1, ..., v_n) \leftarrow \mathcal{L}_1(v_1, ..., v_n), ..., \mathcal{L}_k(v_1, ..., v_n),$$

X_i representa o domínio da variável v_i ou constante ζ_i, que pode ser um conjunto de entidades ou relacionamento. O_i é uma referência a alguma ordem de acesso do banco de dados, seja total ou parcial.

As constantes $\zeta_1, ..., \zeta_m$ são inicializadas na. especificação da base, conforme definido pelo usuário. A nomenclatura *constante* é utilizada no sentido que seu valor se mantem inalterável após sua. inicialização.

As variáveis $v_1, ..., v$, são quantificadas existencialmente, porque têm seus domínios restritos às ordens estabelecidas, que são varridas por ocasião da avaliação da regra. Esta varredura faz com que as variáveis existenciais sejam instanciadas, passo a passo, até que a regra seja satisfeita ou se encerre a avaliação. A composição de declarações existênciais precedente a. uma regra corresponde à definição de caminhos que permitem operações de navegação pelas diversas entidades ou relacionamentos do esquema E-R em questão.

Observe que a primeira construção apresentada acima.engloba as duas últimas, mas por questões de didática optou-se por uma descrição mais detalhada.

Ações

As ações, no ambiente QUICIC-SHELL, permitem a atualizaqão dos objetos da base de fatos, recebendo como atribuição valores relativos a base de fatos, conforme visto na seção 111.3, ou valores obtidos pelo acesso a. base de dados, expresso por:

$$\alpha_{bf}(t) \leftarrow \alpha_{bd}(t)$$

onde bf e bd denotam um atributo da base de fatos e de dados, respectivamente, e t simboliza um termo.

Avaliação Literais e Regras

A avaliação de literais e regras no contexto da QUICIC-SHELL leva em comideração os termos-E e termos-C, conforme descrito em III.4.1, e as seguintes definições se aplicam:

Definição 4 *Seja $\mathcal{L}$ um literal da forma $t_1 \theta t_2$. Então, a avaliação de L é:*

$$g(t_1 \theta t_2) = \begin{cases} 0; & \text{se } t_1 \theta t_2 \text{ é falso;} \\ 1; & \text{se } t_1 \theta t_2 \text{ é verdadeiro;} \\ inclefinido; & \text{se } t_1 \text{ e } t_2 \text{ são incompatíveis} \end{cases}$$

Definição 5 *Seja $\mathcal{L}$ um literal da forma $t \in X$. Então a avaliação de $\mathcal{L}$ é:*

$$g(t \in \mathcal{S}) = \begin{cases} 0; & \text{se } t \text{ não é um elemento da instância de } X; \\ 1; & \text{se } t \text{ é um elemento da instância de } X; \\ indefinido; & \text{se } t \text{ não for compatível com} \end{cases}$$

Definição 6 *Seja L iim literal da forma t_1 E $\alpha(t_2)$. Então a avaliação de L é:*

$$g(t_1) \in \alpha(t_2) = \begin{cases} 0; & \text{se } t_1 \text{ não é um elemento da lista retornada por } t_2; \\ 1; & \text{se } t_1 \text{ é um elemento da lista retornada por } t_2; \\ inclefinido; & \text{se } t_1 \text{ não for compatível com } \alpha(t_2) \end{cases}$$

A avaliação de regras sem variáveis já foi descrita na seção anterior. A seguinte definição considera o uso de variáveis:

Definição 7 *Seja* R *uma regra da forma*

$$v_1 \text{ E } X_1/O_{m+1}, ..., v_n \text{ E } X_n/O_n$$
$$\mathcal{A}(v_1, ..., v_n) \leftarrow \mathcal{L}_1(v_1, ..., v_n), ..., \mathcal{L}_k(v_1, ..., v_n),$$

$k \geq 0$. Então a avaliação de R é

$$g(\mathcal{R}) = \begin{cases} 0; & \text{se } g(\mathcal{L}_j) = 0 \text{ para algum } j \text{ E } [1, \mathrm{k}]; \\ 1; & \text{se } g(\mathcal{L}_j) = \mathbf{1} \text{ para todo } \mathbf{j} \text{ E } [1, \mathrm{k}]; \\ indefinido; & \text{se } \exists j \text{ E } [1, k], \text{ tal que } g\ (\mathcal{L}_j) \text{ é indefinido;} \end{cases}$$

Estruturação das Regras

As regras de produção são estruturadas na. forma de grafos E/OU, e o único motor empregado atualmente é o regressivo. A definição de termos na forma de funções facilita consideravelmente o mecanismo de inferência, que será visto com detalhes no próximo capítulo.

111.5 Aspectos Arquiteturais

Esta seção considera os aspectos arquiteturais da QUICK-SHELL com base nos conceitos teóricos analisados no capítulo anterior.

No que se refere à forma de integração, a QUICK-SHELL adota a arquitetura distribuída. Este fato se deve a dois motivos básicos. O primeiro se refere ao desejo de se manter a indepêndencia clos sistemas componentes, impedindo que um mantenha controle sobre o outro. Esta característica fornece uma potencialidade para o sistema como um todo no sentido de ser viável o atendimento de requisições nos dois sentidos, isto é, tanto por parte do banco de dados como pelo sistema especialista.

O segundo motivo diz respeito ao modo de interação entre os sistemas, que se dá por meio de mensagens. O acesso às informações relativas aos diversos componentes de um esquema E-R. é feito por troca. de mensagens, que no nosso caso foi implementado como chamadas a. funções. De fato, ordens parciais e atributos referênciais podem ser vistos como funções que aplicadas a elementos ou ao próprio conjunto de entidades ou relacionamento, retornam listas ordenadas de identificaclores referentes ao conjunto de entidades ou relacionamento alvos da consulta. Da mesma forma, as ordens totais podem ser consideradas funções, que, aplicadas a um conjunto de entidades ou relacionamentos, retornam listas ordenadas clos elementos que compõem o mesmo.

De posse destas listas, obtem-se o identificador que será objeto da consulta, ou continua-se a navegação através da aplicação de outras ordens ou atributos referênciais sobre a lista de identificadores, e assim sucessivamente.

Em termos da base de fatos, o acesso ao valor de uma dupla O-A, conforme visto na seção **3,** pode ser vislumbrado também como uma função que, aplicada ao objeto, retorna um valor.

Levando-se em consideração estes aspectos, uma chamada a uma função de banco de clados pelo sistema especialista inicia uma transação onde o invocador é o mestre e o receptor é o escravo. O retorno da função é a reposta da mensagem, se houver, e indica o fim da transação.

Esta. convenção, na ótica do sistema. especialista, possibilitou a expressão de operações de navegação pelo banco de dados. Sob o ponto de vista do banco de dados, o emprego de funções facilitou consideravelmente a tradução das requisições, que é clescrita com detalhes no próximo capítulo.

III.5.1 Semântica da Linguagem de Regras

Com base nas considerações feitas, definiu-se uma linguagem de regras funcional que abrange de uma forma clara e intuitiva todos o conceitos teóricos apresentados. Funcionalidade, neste contexto, se refere a forma como os termos são expressos pela linguagem de regras, vide anexo A.

Através da linguagem são definidas a base de fatos e as regras, onde são armenadas as informações fornecidas pelo especialista. Sua tradução gera comandos numa linguagem intermediária que, em execução, efetua solicitações ao servidor E-R. Este, subsequentemente, acessa o banco de claclos objeto. Assim, o servidor E-R mapeia termos E-R em funções de acesso ao banco de claclos. Estas são definidas por ocasião da geração do dicionário de dados, na fase de diagramação. Maiores detalhes a respeito do servidor E-R são apresentados no próximo capítulo.

Assim, na especificação da base de fatos e regras, deve-se levar ein consideração os nomes utilizados nas definições de entidades, relacionamentos, ordens e os demais elementos do modelo E-R presentes no esquema original. Isto se justifica pelo fato clos métodos que implementam o acesso à base de claclos E-R utilizarem os mesmos nomes definidos no esquema.

Objetos da base de fatos, variáveis e contantes E-R. são declaradas anteriormente à base de regras. A declaração de um objeto consiste em identificar o nome do mesmo e clos atributos que o caracterizam. Os domínios clos atributos, no contexto da base de fatos são inteiro, real e alfanumérico. O valor de cada. atributo

pode ser obtido pelo processo de inferência, ou simplesmente através de uma pergunta ao usuário. Esta, diferenciação é também especificada. Na especificação de uma variável E-R, define-se o nome da mesma *e* ao domínio ao qual ela pertence. Com isso, é possível efetuar as operações de abertura de arquivos e relacionar os nomes empregados nas regras com seus respectivos métodos.

A declaração de constantes E-R corresponde à instanciação de um elemento de um conjunto de entidades ou relacionamento obtida através da função chave.

Regras de produção são expressas na forma de se *então*. No caso de haver instanciações existenciais, estas são declaradas anteriormente a. regra em questão. Para cada. instanciação são definidos a. variável, o domínio da mesma e a ordem de acesso.

O comando DETERMINE é utilizado para discriminar as triplas O-A-V que devem ter seus valores determinados através da execução da *shell*. Opcionalmente, pode-se verificar a ordem das regras disparadas, através do comando MOSTRE.

A sintaxe da linguagem de regras pode ser vista com detalhes no apêndice B, e ao próximo capítulo abordaremos com detalhes questões de implementação.

III.6 Exemplo de Utilização

Esta seção apresenta um exemplo simplificado de utilização da QUICK-SHELL com base no esquema E-R e base de conhecimento clescritos no apêndice A.

Esquema E-R

O diagrama E-R consiste de uma. representação gráfica que expressa um modelo conceitual E-R sobre um domínio. Todos os elementos do modelo E-R são simbolizados por figuras geométricas conforme mencionado no capítulo II.

No Apêndice A encontramos um diagrama E-R que modela a organização da Sociedade Brasileira de Computação (SBC). A SBC coordena diversos eventos de computação realizados no país. Cada evento reúne pesquisadores de uma área específica.: Inteligência Artificial, Redes, Computação Gráfica, etc. Esses eventos são realizados na forma de congressos, onde cada congresso possui uma comissão organizadora, composta. por sócios da SBC, ou convidados. Os sócios da SBC são ein geral estudantes, ou profissionais e pesquisadores de informática e se encontram espalhados nas diveras regiões do país, de acordo com seu local de trabalho. Cada região possui um delegado.

SÓCIO é uma generalização dos tipos de associados da SBC. As especializações são as entidades ESTUDANTE e PLENO, representando cada tipo de sócio. O atributo referencial *regional* de SÓCIO retorna urna instância de REGIONAL. Inversamente, o atributo *sócio* de REGIONAL, quando aplicado a uma instância deste, retorna uma lista de instâncias de SOCIO. As ordens totais de acesso para SÓCIO foram definidas como alfabética e numérica. Assim, todas as instâncias de SÓCIO podem ser obtidas pela varredura dessas ordens.

Identifica-se também o relacionamento INSCRIÇÃO, que agrega as entidades COMISSÃO e SÓCIO. Neste relacionamento, uma comissão é composta por no mínimo, 0 e, no máximo, m sócios, e cada sócio pode participar em, no mínimo 0 e, no máximo n comissões. Na mesma figura, a ordem *inscrições* é parcial, porque acessa apenas as instâncias de INSCRIÇÃO associadas a uma determinada instância de COMISSÃO. Isto é, para cada comissão, *inscrições* atua como um atributo que retorna uma coleção de instâncias de INSCRIÇÃO associadas àquela comissão. De posse desta coleção, acessa-se, indiretamente, ALUNO e PLENO.

Base de Conhecimento

O sistema especialista SBC averigua se dado um sócio ele merece ou não ser premiado.

A regra. 1 obtém o número de inscrição do sócio na SBC. Esta. informação é colhida através da aplicação da função chave que recebe como parâmetros o domínio, a ordem de acesso e a chave que correspondem a SÓCIO, alfabética e *nome(cand)* respectivamente. Observe que o parâmetro chave referencia um atributo de objeto da base de fatos, cujo valor é obtido por uma pergunta ao usuário em tempo de execução da *shell.*

A primeira condição de premiação consiste no candidato ser delegado de alguma região, expresso pela regra 4.

Na segunda condição, codificadas pelas regras 5 e 6, verifica-se a titulação do candidato. Se ele for doutor, a. condição é automaticamente satisfeita. No caso dele ser mestre averigua-se o número de publicações do candidato. Se for superior a 10, a condição é satisfeita.

Na terceira condição considera.-se o tempo de participação do candidato na SBC, conforme pode ser visto nas regras 7 e 8. No caso de candidato com mestrado, o tempo exigido é seis anos. Se o candidato for doutor o tempo exigido é três anos.

Na quarta condição, que corresponde a. regra. 9, verifica-se se o candidato é integrante de alguma comissão organizadora.

Por fim, a quinta condição, expressa pelas regras 10 e 11, consiste na avaliação da participação do candidato na SBC de uma forma geral. Se o candidato for mestre exige-se que a participação dele seja excelente. No caso de ser doutor, a participação deve ser boa.. Observe que esta. informação é obtida através do atributo participacao do objeto *cand*, que no caso efetua. uma pergunta ao usuário por ocasião da execução da *shell*.

A regra 12, então, testa as cinco condições mencionadas anteriormente. Se todas as condições forem satisfeitas o candidato merece ser premiado.

O objeto cand tem como atributos o tipo do cacliclato, cloutor ou mestre, o nome e número de inscrição, a qualidade de participação e o sucesso da avaliação do mesmo.

O objeto *condição* registra o resultado da avaliação das cinco condições descritas anteriormente.

As variáveis da base são reg, com e ins, que tem como domínio REGIONAL, COMISSÃO e INSCRIÇÃO.

O comando USA especifica que a base de claclos a ser consultada é SBC. Na execução cleste comando, todas as inicializações com relação ao banco de claclos são efetuadas.

Na regra 4 percebemos uma. instanciação existencial pela ordem de busca alfabética. Na regra. 9 temos duas instanciações que correspondem a uma operação de navegação. Assim, para cada comissão, por ordem *alfabética*, obtem-se os sócios participantes, pela ordem parcial *inscriçôes*.

O objetivo do sistema especialista é expresso pelo comando DETERMINE, que no caso é *sucesso(cand)*.

111.7 Conclusões

Este capítulo apresentou uma proposta. de integração de *shells* O-A-V com bancos de claclos baseados no modelo Entidade-Relacionamento. A extensão proposta ao moclelo, através da introdução de atributos referenciais e ordens de acesso, visou enriquecer as operações de acesso e navegação, possibilitando que a implementação do banco de claclos continuasse baseda no moclelo E-R. Detalhou-se o modelo O-A-V num aspecto mais formal para. que em, seguida, fossem introduzidos os elementos necessários para efetuar o acoplamento.

Foi especificado a QUICK-SHELL, que é um ambiente que integra banco de claclos e sistemas especialistas baseados no moclelo E-R e O-A-V, respectivamente.

A adoção desses modelos permitiu a definição de termos, literais, regras e mecanismos de quantificação que abrigam de forma natural os elementos O-A-V e E-R. Este ambiente unifica componentes O-A-V e E-R, de forma que o usuário tenha a impressão que o banco de claclos é uma extensão da shell. Conforme pode ser percebido, a nossa proposta se baseia no acoplamento forte. A linguagem de regras abrange elementos da base de conhecimento e do banco de clados de uma forma clara e uniforme, tornando transparentes para o usuário operações específicas do banco de claclos. Todas as operações válidas num processamento convencional são acessíveis ao sistema especialista. A linguagem de regras também pode ser considerada como um espécie de linguagem de consultas.

A geração de consultas se dá de forma estática, pois as traduções dos termos-E e termos-C são efetuadas previamente à execução do sistema. Isto se justifica pela possibilidade do banco de claclos objeto não estas necessariamente residente no mesmo ambiente do sistema especialista. e, neste caso, se permitissemos o acesso dinâmico, a execução do sistema. seria eventualmente interrompida pela busca de claclos armazenados remotamente, prejudicando o desempenho final do sistema.

Apesar do presente trabalho explorar a integração de bancos de claclos e sistemas especialistas no aspecto do segundo manipulas os elementos do primeiro, podemos notar que a. utilização de funções como agentes de troca de mensagens facilitam também a manuseio dos elementos do sistema especialista pelo banco de claclos. Neste caso, o valor de um atributo do banco de claclos seria expresso por uma função, que, ao ser invocada, provocaria. o disparo das regras da *shell*.

Outros trabalhos apresentam abordagens distintas para. a consecução de tal acoplamento [25],[22], [32], [31]. Neste último, o autor descreve um sistema que permite atributos da base de claclos calculados através do processo de inferência. As bases de claclos, entretanto, não são genericamente E-R como no nosso caso, além de persistirem as limitações de acesso impostas originalmente pelo modelo.

Capítulo IV

Implementação

IV.1 Introdução

A Programação Orientada a Objetos (POO) vem tomando impulsos significativos nos últimos anos. As principais características de linguagens de programação orientadas a objetos se referem a abstração, modularização, encapsulamento e herança. Devido a semelhança destes aspectos com os vistos em modelagem de claclos, e na intenção de ampliar o universo dos tipos de dados possíveis de serem armazenados numa base de claclos, emergiu um nova linha de pesquisa que utiliza. técnicas de orientação a objetos em bancos de dados. Esta área de investigação é denominada bancos de claclos orientados a. objeto [34].

Levando em consideração estes fatos, a nossa. proposta. de implementação consiste em desenvolver uma *shell* cuja implementação é orientada a objetos. Isto é, os elementos da base de fatos e regras são especificados como objetos que interagein através de mensagens. O banco de dados, por sua vez, é orientado a objetos, apesar de ainda não possuir suporte pasa. a armazenagem de objetos complexos. O objetivo do tradutor QUICK-SHELL é geras código para a linguagem de orientação a objetos empregada, que implementa a. *shell* O-A-V e que executa as funções definidas para. o banco de dados.

Este capítulo inicia abordando as principais questões de POO. Em seguida, apresenta-se como estes conceitos foram aplicados na QUICK-SHELL, tanto no escopo da *shelb* como no que concerne ao banco de dados. Com isso, são explicados os mecanismos de traduções de termos, literais e regras pasa comandos de orientação a objetos. Os algoritinos de tradução são apresentados junto com algumas estruturas de dados. Por fim, comenta-se a ferramenta utilizada para o desenvolvimento do tradutor, juntamente com algumas conclusões.

IV.2 Aspectos de Orientação a Objetos

Programação Orientada a Objetos é um método de programação que tenta modelar o mundo real da forma mais natural possível. Este estilo de programação possui maior estruturação, modularidade, abstração e capacidade de "esconder" informação [35] em comparação com as diversas formas de programação estruturada utilizadas até agora.

Em POO, o analista ou programador busca ter uma visão de dados de sua aplicação ao invés de funções, como é feito normalmente em programação estruturada [36]. Nesta abordagem, tenta,-se modelar os elementos de uma aplicação em termos de objetos que se comunicam através de mensagens.

As propriedades básicas que caracterizam POO são *Encapsulamento*, Herança e Polimorfismo. O *encapsulamento* consiste na combinação de um registro convencional com procedimentos e funções, formando um novo tipo de estrutura e, consequentemente, manipulado como um *tipo* de *dado*, um Objeto. Este pode ser considerado uma entidade do mundo real que, pelo processo de abstração, é representado por este novo estilo de estrutura de dados.

Um objeto possui uma memória interna em que valores podem ser armazenados e modificados ao longo da vida do mesmo, e um comportamento, que consiste no conjunto de ações pré-definidas (métodos) através clas quais o objeto responde à demanda de processamento por parte de outros objetos. Ao receber uina mensagem, o objeto seleciona a, execução da ação que faz parte de seu comportamento. Após a execução, o objeto retorna o controle ao objeto que lhe enviou a mensagem.

Herança. diz respeito à definição de um determinado objeto e à reutilização deste para especificação de descendentes de uma forma hierárquica. Herdeiro é o que herda os atributos e ações de um outro objeto, dito ser seu parente. Um determinado objeto pode ter mais de um parente, Da mesma forma, um herdeiro pocle ser parente de outros herdeiros e assim sucessivamente. Vista de outra forma, a relação entre um herdeiro e seu parente pode ser considerada como uina relação é *um(a)*, utilizada em redes semânticas.

O conceito de herança é uma clas características fundamentais em POO. E através dela que se originam operações de especialização e generalização. Esta. faceta permite uma maior reutilização de código c amplia a.capacidade de extensão dos objetos primários definidos gerando outros mais complexos.

Polimorfismo é uma facilidade oferecida. na herança. que permite a reimplementação de uma ação herdada.

Para quem manuseia um objeto, não importa. como suas ações foram imple-

mentadas. Isto é, o que aparece externamente são os *serviços* que o objeto oferece. Para ilustrar, suponha. que uma. pilha seja um objeto oferecendo os *serviços* de *push e pop*. O modo como estas ações são implementadas ficam *escondidas* dos usuários deste objeto.

Para especificar um projeto utilizando a.filosofia de orientação a olsjetos, define-se o domínio da aplicação, para em seguida abstrair os olsjetos do sistema. Passa-se para fase de identificação dos atributos e ações de cada objeto, definindo as possíveis mensagens a serem trocadas. Feito isto, é verificado se existe algum dos conceitos oriundos da herança: especialização, generalização, e polimorfismo. Caso algum exista., organiza-se os objetos de tal forma. a. reutilizar ao máximo o código. Passa-se para fase de programação onde uma. linguagem adequada deve ser escolhida [37].

IV.3 Orientação a Objetos na Quick-Shell

No contexto da QUICK-SHELL, a.linguagem escolhida foi o Turbo Pascal a partir da versão 5.5 da Borland [35], que é uma versão com construções sob o paradigma de orientação por olsjetos. Desta. forma, ao longo deste capítulo algumas construções do Pascal são utilizadas para ilustar fielmente as soluções adotadas. É importante salientas que embora o ambiente escolhiclo seja o Pascal, as soluções expostas aqui são genéricas.

Utilizaremos a notação *objeto-b* e *atributo-b* para expressar objetos e atributos da base de fatos, e analogamente *objeto-p* e *atributo-p* no contexto de programação orientada a objetos. A filosofia.básica do sistema, na ótica de orientação a objetos, consiste em mapear cada objeto-11, e a base de regras como um todo, ein objetos-1s.

Cada objeto-1s é traduzido em um objeto-p que possui o mesmo nome do olsjeto-1s. Da mesma forma., os atributos que compõem o objeto-1s são exatamente os mesmos que descrevem o objeto-11. As ações, ou métodos, de cada objeto-p definido se referem às funções que obtêm os valores de cada atributo do referido objeto. Para exemplificar, considere o olsjeto *cand* da base de fatos SBC que pode ser visto no apeêndice A. A tradução fica sendo:

```
cand = object

 sucesso- :          string [STR];
 num- :              string [STR];
 nome-:              string [STR];
```

```
participacao-:          string[STR];
tipo-:                  string[STR];


function sucesso:       str20type;
function num:           str20type;
function nome:          str20type;
function participacao:  str20type;
function tipo:          str20type;

end;
```

*sucesso_*, num-, nome-, participacao- e tipo- são exemplos de atributos do objeto-p enquanto sucesso, *num*, nome, participacao e tipo são as ações que são disparadas para obter os valores de cada atributo respectivamente. Nas funções que implementam as ações, é averiguado se o valor do atributo em questão é obtido por pergunta ou inferência. No primeiro caso. uma pergunta é feita ao usuário, e a função retorna o valor obtido. No segundo caso, aciona-se o mecanismo de inferência.

IV.3.1 Mecanismo de Inferência

Como estamos num ambiente de orientação a objetos, é natural considerar a base de regras como um objeto-p, onde cada instância de regra.é representada por uma ação do tipo função. Esta escolha justifica-se pelo fato das regras serem traduzidas diretamente para comandos if then *else* da linguagem, ao invés de representá-las por meio de estruturas de dados mais complexas. O nome da ação está relacionado ao consequente da regra, ou seja, ao objeto da base que está recebendo a atribuição.

Para as regras da base SBC, o objeto-p *regra* é definido como:

```
regra = object
function cand_sucesso :         str20type;
function condicao-quinta :      str20type;
function condicao-quarta :      str20type;
function condicao-terceira :    str20type;
function condicao_segunda :     str20type;
function condicao_primeira :    str20type;
```

```
function cand_num:          str20type;
function cand_tipo:         str20type;

end;
```

Por exemplo, para se determinar *sucesso(cand)* dispara-se o método do objeto regra, *cand_sucesso*, que obtém o valor do referido atributo. Isto é, as funções que implementam as regras são geradas de acordo com as duplas A-V que surgem na ação de cada regra da base.

No corpo da base de regras é bastante comum encontar, nos consequentes, atribuições aos mesmos pares A-V, Nestes casos, as regras são agrupadas em uma mesma função, se a primeira falhar testa,-se a segunda. Isto equivale dizer que todas as regras que atribuem valores a pares A-V idênticos são agrupadas em uma mesma função do tipo ação.

Conforme já, mencionado, a geração das funções que implementam as regras se dá de forma direta através dos comandos *if then else*. Cada termo é traduzido para a composição de métodos que obtem o valor do atributo que estiver sendo referenciaclo.

Na tradução, ainda, são geradas chamadas a uma rotina de empilhamento que guarda a ordem de busca às regras de forma que o usuário possa, ao final da execução, verificar a ordem das regias disparadas, através do comando MOSTRE.

Cabe ressaltar que a utilização de grafos E/OU e do algoritmo de encadeamento regressivo se dá de forma implícita, pois a própria. forma de organização e representação do conhecimento através da orientação a objetos executa. esta tarefa de forma simples e estruturada.

IV.4 Banco de Dados: Objetos e Métodos

Esta seção descreve a. organização interna. do banco de dados à luz de orientação a objetos e apresenta os principais métodos que manipulam o mesmo.

IV.4.1 Servidor E-R

O acesso à base de claclos é feito através do servidor E-R figura IV.1, que armazena uma coleção de inétoclos aplicáveis às classes de objetos que formam o banco de dados. Neste contesto, conjuntos de entidades e relacionamentos são conside-

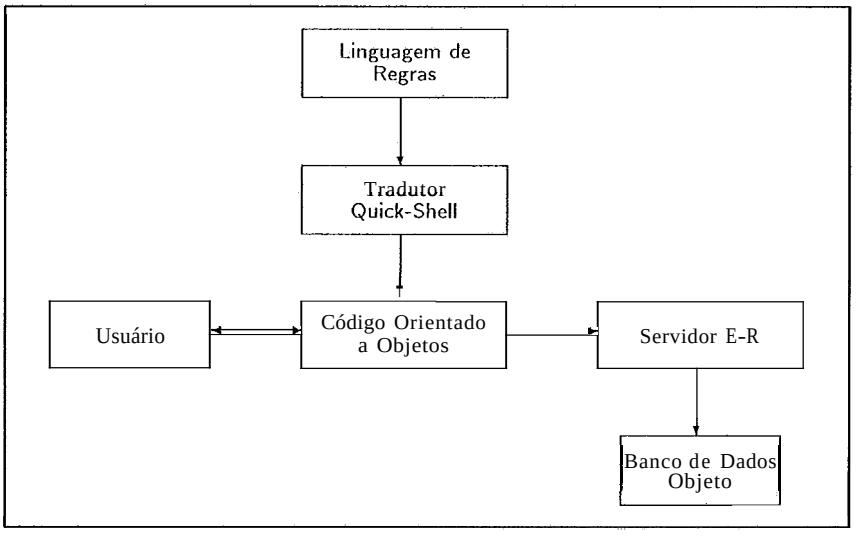

Figura IV.1: Modelo Lógico da Implementação

rados uma mesma classe de objeto. identificada parcialmente por T_ENTIDADE. As ordens também constituem uma classe de objetos que manipulam elementos dos conjuntos de entidades ou relacionamentos. identifiacadas por T_ORDEM. Cada elemento corresponde a uma instância definida em um conjunto de entidades ou relacionamento, também considerado como uma classe, indentificada por T_ELEMENTO. Finalmente, o banco de dados como um todo é também considerado uma *instância* de banco de dados, identificada por T_BANCO. Uma classe, no presente contexto, é considerada um tipo de objeto. Assim, para cada classe existe um conjunto de métodos vistos pelos usuários do servidor E-R, que possibilitam a inicialização, acesso e encerramento de manipulação de cada objeto. E importante ressaltar que existe também um conjunto de métodos que não aparecem para o usuário, e servem para manipulação interna de arquivos da base de dados.

Assim, a função do servidor E-R é mapear operações sobre o modelo E-R para procedimentos que implementam tais operações, de forma a acessar o banco de claclos. Nesta filosofia, o usuário tem a impressão de estar manipulando um banco de dados virtual, não se preocupando com informações de implementação do mesmo. Esta característica provê um alto grau de portabilidade, uma vez que a migração para outras máquinas não afeta o funcionamento da *shell*.

Além de traduzir operações, o servidor coleta informações resultantes da execução e envia-as ao módulo solicitante, viabilizando as operações de consulta.

Em seguida, serão descritos o principais métodos aplicáveis a cada classe de objetos definidos para o banco de dados.

IV.4.2 Métodos de Acesso

A *classe* banco de dados identifica. a. base de claclos como um todo, e aceita os seguintes métodos básicos:

OPENALL Este método abre acesso a todos os arquivos gerados no banco de dados a partir dos conjuntos de entidades, relacionamentos e ordens especificadas no esquema E-R;

CLOSEALL Este método encerra.o acesso aos arquivos abertos no método OPENALL;

CREATEALL Este método é responsável pela.geração de uma.base com instâncias vazias;

Uin objeto da classe banco de dados tem ainda um conjunto de atributos que servem para identificação e controles operacionais, que não são relevantes no presente trabalho.

A *classe conjunto*, associada ao objeto T_ENTIDADE, abriga operações genéricas a qualquer conjunto de entidade ou relacionamento, como :

CARDINALITY Obtém o número de elementos na instância. do conjunto;

OPENDATA Este métoclo é responsável por abrir acesso ao arquivo de claclos já definido;

CLOSE Este método encerra o acesso ao arquivo de dados;

CREATEDATA Este método gera. a estrutura de um arquivo de dados;

A definição de um determinado conjunto de entidades ou relacionamento envolve a definição de dois sub-objetos. O primeiro, que denominaremos de geral, herda os métodos de T_ENTIDADE, armazena as ordens de acesso, que são do tipo T_ORDEM, e oferece a função POINTER, que a partir de uma referência lógica a um elemento retorna um ponteiro para o mesmo. O segundo, denominado *específico*, herda os métodos de T_ELEMENTO, abriga os atributos que caracterizam o conjunto de entidade ou relacionamento em questão e os métodos de manipulação.

A classe elemento, associada ao objeto T-ELEMENTO, armazena os métodos que obtem identificadores físicos c lógicos de um elemento de um determinado conjunto de entidades E ou de um relacionamento R. Os principais métodos são:

REFERENCE Obtém a referência física. a um elemento lógico ;

SURROGATE Obtém a referência.lógica.que identifica unicamente um elemento dentro da base.

Os métodos de manipulação são responsáveis pelo armazenamento, atualização, deleção e instanciação de cada elemento, conforme descrito abaixo:

STORE Armazenagem de um elemento;

UPDATE Atualização de um elemento;

DELETE Remoção de elemento;

INSTANTIATE Instancia um elemento do conjunto a.partir dos dados gravados na base.

Por fim, a. classe ordem trata do acesso à base de dados. Uma instância da. classe ordem pode ser imaginada como uma. lista de referências a. elementos da base. Assim, pode haver diversas ordens definidas sobre um conjunto de entidades ou relacionamento. Através de cada uma dessas ordens, pode-se acessar os elementos segundo critérios dc ordenação que são definidos por ocasião da confecção do projeto lógico do banco de claclos. A classe ordem, associada ao objeto T_ORDEM, admite os seguintes métodos:

RESET Posiciona no início da ordem;

NEXT Obtem o próximo elemento;

EOO Testa o fim de lista.;

SEARCH Utlizado na busca. de um elemento que contenha atributo igual ou posterior a determinada chave;

FIND Utilizado na busca de um elemento que contenha exatamente determinada chave;

OPEN Prepara para acesso através da ordem;

CLOSE Encerra o acesso pela ordem;

DELETEKEY Remove a, associação de um elemento a uma determinada chave;

CHANGEKEY Altera a associação;

ADDKEY Insere associação.

IV.5 Traduzindo Expressões para a Shell O-O

A figura IV.1 expressa o modelo lógico da implementação. O objetivo é, a partir da base de regras e fatos, expressos através da linguagem de regras, gerar código orientado a objetos que acesse os métodos definidos para o servidor E-R e implemente a *shell* O-A-V. O tradutor QUICK-SHELL analisa a sintaxe e semântica de cada termo envolvido, formando os comandos orientados a objetos adequados.

Na regra 9 do Apêndice A, duas declarações existenciais especificam os domínios e as ordens de obtenção de cada instância das variáveis existenciais com relação aos seus respectivos domínios. Para cada instância de variável obtida testa-se a condição da regra, até que a ação seja disparada ou a regra falhe. Parece natural que a estrutura de código gerado para as regras com instanciações existencias seja constituída por *whiles* aninhados. A passagem de um *while* externo para um mais interno se dá pela ordem definida na especificação da instanciação, conforme pode ser visto na Figura IV.2.

De outra forma, as ordens podem ser vistas como um tipo especial de atributos que simbolizam funções que retornam listas de identificadores de instâncias de um domínio específico. Conforme visto, uma *ordem* é um *Objeto* que encapsula uma memória interna e um conjunto de procedimentos e funções. Assim, todos os atributos do tipo *ordem* possuem comportamentos similares no sentido de que os mesmos métodos se aplicam. Como exemplo, podemos citar os métodos NEXT, que retorna o identificador correspondente à próxima instância na ordem, e FIND, que dada uma chave retorna o identificador associado. Como a manipulação de uma ordem envolve uma busca, definiu-se os métodos RESET, que reinicializa uma ordem, e EOO, que verifica se a busca numa determinada ordem deve ser encerrada.

De posse do identificador, aplica-se o método *instancia* ou a função *ponteiro*. Estas retornam a instância propriamente dita ou o ponteiro para a instância, respectivamente.

Na codificação das regras, uma ordem é traduzida conforme as transições mostradas abaixo:

a) $X.O \rightarrow X.O$;

b) $O(t) \rightarrow X.pointer(t') \wedge .O.$

onde t simboliza um termo. A primeira expressão representa uma ordem total, que é ilustrada no exemplo da figura IV.3 por COMISSÃO.alfabética, que especifica,

no domínio COMISSÃO, a busca pela ordem alfabética de *com*. A segunda expressão representa uma ordem parcial, ilustrada. no exemplo por *inscrições(com)*, que especifica, no domínio INSCRIÇÃO, a. busca por ordem de inscrições de *ins*. A ordem de acesso às instâncias de INSCRIÇÃO é definida na especificação lógica do banco de dados.

No corpo das regras, podemos perceber os termos

$$num(cand) \text{ e } primeira(condicao)$$

que são denominados termos simples. Termos do tipo

$$num(socio(ins)) \text{ e } num(delegado(reg))$$

são denominados termos compostos. Termo simples, como o próprio nome cliz, é composto apenas por um termo. Termos compostos correspondem a termos que são formados a partir de sub-termos. É importante ressaltar que, em ambos os casos, o nome mais externo é sempre um atributo.

A estrutura do cócligo gerado para cada tipo de termo aparece abaixo:

a) $v \rightarrow v$

b) $\rho(v) \rightarrow v.\rho$

c) $\beta(v) \rightarrow v.\beta$

d) $\Gamma(t_1, t_2, t_3, ...t_n) \rightarrow X.POINTER(X.O.FIND(t_3, ...t_n)), \quad n \geq 3$

e) $\alpha(o) \rightarrow o.\alpha$

f) $f(t_1, ..., t_n) \rightarrow f(t_1, ...t_n)$

g) $\alpha(t) \rightarrow X.pointer(t') \wedge .att$

Os itens a, b e e referem-se às traduções clas variáveis e termos simples. O item c cliz respeito à traclução de atributos referenciais. O item d refere-se à tradução da função chave, onde t_1 representa o domínio, t_2 a ordem de acesso e $t_3...t_n$ os termos que identificam a. chave primária.. O item f cliz respeito à tradução de funções. O item g corresponde à tradução de termos compostos, onde X representa o domínio de t, sendo t um termo. A identificação de X é obtida através de uma consulta ao dicionário de dados, uma vez que X é o domínio de t. f' representa a. tradução do termo original t.

```
ord00 := COMISSAO.alfabetica;
ord00.reset;
while not ord00.eoo do
begin
        com.instantiate(ord00.next);
        ord01 := com.inscricoes^;
        ord01.reset;
        while not ord01.eoo do
        begin
            ins.instantiate(ord01.next);
            if ( ob1.num = SOCIO.pointer(ins.socio)^.num ) then
            begin
                condicao_quarta := 'TRUE';
                goto 002;
            end;
        end;
end;

        .

002: end;
```

Figura. IV.2: Tradução de uma. regra

IV.5.1 **Explicando a** Tradução

P u a clarear os conceitos apresentados em IV.5, descreveremos a coclificação da regra 3, conforme mostrado na Figura IV.2. De modo a. não estender excessivamente este trabalho, assume-se que o leitor esteja familiarizado com programação orientada a objetos *e* Pascal. Entretanto, maiores esclarecimentos a. respeito podem ser facilmente obtidos da. documentação do Turbo Pascal da Borland, a partir da versão 5.5 [35].

A variável *ord00* armazena o objeto referente à ordem total *alfabética* de COMISSÃO. Inicializa-se esta ordem

ord00.reset

e enquanto a.lsusca pela ordem não encerrar

while not ord00.eoo

obtem-se a próxima. instância de COMISSÃO

com.instantiate(ord00.next)

A instância obtida. fica armazenada em *com,*. De posse desta instância, aplica-se o métoclo referente à ordem parcial *inscriçôes* em *com,* e armazena-se o resultado em *ord01.* Inicializa-se esta. ordem

ord01 := com.inscricoes^;
ord01.reset

e enquanto a. busca pela ordem parcial dc *inscricoes* não encerrar

while not ord01.eoo

obtem-se a próxima instância de INSCRIÇÃO

ins.instancia(ord01.next)

Note que apenas as instâncias de INSCRIÇÃO que estejam relacionadas a com são obtidas através da ordem em questão. Cada instância obtida fica armazenada em *ins.* Observe ainda que o método *instantiate* faz com que a. variável referenciada (no caso *zns)* seja. instanciada com o objeto cujo identificador interno é fornecido ao método como parâmetro, no caso uma referência lógica de um elemento.

Com as variáveis com e *ins* instanciadas. testa-se a condição da regra. A codificação do corpo da regia consiste em comparar se o número do candidato com doutorado está na lista. dos participantes de comissões. A veracidade deste fato implica no candidato já ter participado de alguma comissão.

IV.5.2 Algoritmo de Tradução e Estruturas de Dados

Esta seção apresenta os algoritmos de tradução e as principais estruturas de dados empregadas no tradutor. Nesta descrição, a notação *att* é utilizada para expressar de uma forma genérica a, p, β. São apresentados apenas os campos e procedimentos mais relevantes para a compreensão da implementação.

As estruturas de dados mais relevantes são *Termo*, *Literal*, *Declaração* e *Corpo*.

A estrutura *Termo* consiste em uma lista encadeada de todos os sub-termos componentes:

Termo

tipo	contexto	nome	prox	domínio	ordem	largs

tipo : Especifica se é termo simples, composto, variável, constante inteira, real, alfanumérica ou ordem total;

contexto : Especifica se é um termo do banco de dados ou da base de fatos;

nome : Especifica o nome referente a *att;*

prox : No caso de termo composto, aponta para o próximo sul]-termo.

Os últimos três campos são necessários para a tradução de funções chaves:

domínio : Especifica o nome da entidade ou relacionamento em que a instanciação se dará;

ordem : Especifica o nome da ordem de busca.;

largs : Ponteiro para a lista dos termos que formam a chave.

Os algoritmos que realizam a tradução de termos, figura IV.3, IV.4, IV.5 recebe como parâmetro a estrutura de dados de um termo e processa as informações contidas nesta a fim de gerar o código orientado a objetos, conforme descrito na seção IV.5.

A geração do código das regras consiste também em chamadas ao algoritmo Traduz-Termo, pois os elementos da mesma são também termos. Internamente,

```
Traduz_Termo(termo)
Se termo for do tipo constante
            retorna(nome da constante);
Se termo for do tipo ordem total
            retorna(nome da ordem total);
Se termo for do tipo variavel
            retorna(nome da variavel);
Se termo for do tipo simples
            retorna("v.att");
            fim_termo = VERDADE;
Se termo for simples do tipo chave
            retorna(concat(Traduz_Chave(sub_termo),"A.att");
Se TERMO-COMPOSTO = VERDADE e termo for do tipo chave
            retorna(Traduz_Chave(termo))
            fim-termo ='VERDADE
Se termo for composto
            traduz_termo_composto(termo);
Fim
```

Figura. IV.**3:** Algoritmo Traduz-Teimo

Traduz_Termo_Composto(Termo)

```
TERMOXOMPOSTO = VERDADE;
empilha (att);
X = dominio de termo;
enfilera(X);
concat(ord,traduz_termo(termo→prox);
Se fim-traducao
          retorna(ord);
string = "".
Enquanto houver elementos na fila
          desenfilera(X);
          concat(string,X".ponteiro(");
concat(string,ord".)");
Enquanto houver elementos na pilha
          desempilha(att);
          concat(string,"^.att");
          Se topo-pilha ≠ base-pilha
                    concat(string,")");
Se fim_termo
          fim-traducao = VERDADE;
retorna(string)
Fim
```

Figura IV.4: Algoritmo Traduz_Termo_Composto

Traduz_Chave(termo)

```
string = ""
X = Traduz-Termo(t1);
ORDEM = Traduz-Termo(t2);
concat(string,"("X".ponteiro("X"."ORDEM".FIND(");

Para i = 3 ate n faça
        salva-contexto();
        ti' = Traduz-Termo(ti);
        restaura-contexto(),
        Se i < n
            concat(string,ti', ");
concat(string,")))");
retorna(string);
Fim
```

Figura. IV.5: Algoritmo Traduz_Chave

uma regra é composta por duas estruturas, a primeira capta as informações referentes a declaração da mesma, se houver. c a segunda armazena dados a respeito do corpo da regra :

Declaração

termo1	termo2	termo3	prox

termo1 : Aponta para o termo referente à variável objeto da ordem;

termo2 : Aponta para o termo referente ao domínio da ordem;

termo3 : Aponta para o termo referente à ordem de busca.;

prox : Aponta. para a próxima instanciação;

Todos os campos clescritos são do tipo termo, já descrito.

A estrutura corpo armazena informações a respeito de todos os elementos que caracterizam uma regra,:

Corpo

rclec	rlit	ract	prox

rdec : Aponta para as declarações da. regra.:

rlit : Aponta para os literais da regras;

ract : Aponta, para. a ação da regra.:

prox : Aponta para a próxima regra.;

A estrutura literal comporta. informações a respeito de todos os literais da regra:

Literal

ltere	op	lterd	prox

ltere : Aponta. para o termo a esquerda.;

op : especifica. o operador relacional;

lterd : Aponta para o termo a. direita;

prox : Aponta para o próximo literal;

Os campos ltere e lterd são do tipo termo.

IV.6 Ferramenta de Implementação

Os geradores dc analisadores sintáticos são ferramentas que auxiliam no desenvolvimento de sistemas que contenham alguma análise sintática. Atualmente, é comum a presença de analisadores sintáticos não só em compiladores e interpretadores, mas também em montadores, depuradores, editores de testo, bancos de dados, planilhas eletrônicas e em diversas aplicações que procuram dotar a sua entrada de comandos ou programação de uma interface amigável.

Estes geradores, também conhecidos como compiladores de compiladores, são usados para minimizar o esforço de implementação dos analisadores sintáticos ao longo do desenvolvimento de um compilador ou interpretador. Isto é, em geral usamos o inétodo de semântica orientada pela sintaxe. método que torna a gramática muito sensível a qualquer mudança na definição da linguagem feita ao longo do seu desenvolvimento, obrigando muitas vezes a modificar o analisador sintático, o que pode ser feito rapidamente se estivermos utilizando um gerador.

No contesto deste trabalho adotou-se o YACC (Yct Another Compilei. of Compilers) como ferramenta de suporte para a implementação do tradutor QUICKSHELL. O arquivo de entrada do YACC contém basicamente a declaração de uma gramática que define a linguagem que o analisador sintático gerado reconhecerá. A classe de gramáticas para a qual o YACC consegue gerar uma analisador sintático é a LALR(1) : Look *Ahead* Left to *right Right most derivation* [39].

Para a geração de um analisador sintático o usuário prepara uma especificação que inclui regras gramaticais que descrevem a estrutura da linguagem, rotinas que são invocadas no reconhecimento de regras: que correspondem às ações semânticas, e código para o fornecimento dos elementos básicos da linguagem, tokens, que comtitui o analisador lésico. Nas ações semânticas são desencadeados os procedimentos de geração de código.

O YACC foi desenvolvido em C portátil e gera como saída um analisador sintático escrito também em C. Internamente, o YACC consiste em uma máquina

de estados finitos com uma pilha, capaz de ler o próximo *token* de entrada,. O topo da pilha armazena o estado corrente c ações disponíveis são *shift*, *reduce*, accept e *error*. Existem também mecanismos para o reconhecimento de ambiguidade e tratamento de erros. Maiores detalhes a respeito da características deste gerador podem ser vistos em [38].

IV.7 Conclusões

Este capítulo mostrou os principais aspectos da implementação da QUICK-SHELL. As facilidades encontradas em programação orientada a objetos, aplicadas tanto em representação do conhecimento como em banco de dados, proporcionaram meios eficientes para a consecução de uma implementação ao mesmo tempo simples e robusta.

Tem-se no momento uma versão experimental em funcionamento, capaz de gerar programas completos que utilizam a *shell* de acesso aos dados. Também está em curso a implementação de uma interface amigável, que obtem informalmente do usuário a definição da base de fatos c regras. De posse das estruturas de dados obtidas. invoca-se as rotinas de geração de código.

O fato do tradutor do QUICK-SHELL gerar código compatível com o Pascal dá grande liberdade ao usuário, considerando-se que o código pode sei modificado, estendido e ligado a outros sistemas previamente desenvolvidos. Esta facilidade pode ser considerada bastante oportuna, dado que atualmente existem várias áreas de pesquisa em informática que desenvolvem projetos que exigem a utilização de sistemas especialistas internamente ao software desenvolvido.

Adicione-se a isto a possibilidade de, sem mudar substancialmente a estrutura do sistema, mudar a linguagem de programação na qual são gerados os programas que executam a. *shell.* Por exemplo, a. utilização da linguagem C++ está. em estudos, uma. vez que as transformações requeridas para a. migração são facilmente exeqüíveis. Isto proporcionaria ao sistema alto grau de portabilidade.

Fatores de certeza., associados aos atributos da base de fatos, podem ser incluidos sem grandes alterações. Como ainda não foram definidas operações de atualização no banco de dados, assumiria-se. fator de certeza. igual a 1 para qualquer fato obtido do mesmo. Os fatores de certeza clos atributos dos objetos da base de fatos seriam definidos pelo próprio usuário por ocasião da definição da mesma. Em termos práticos, isto implica em associar cada atributo-p a um fator de certeza, armazenando esta informação mim registro dentro da própria estrutura que define o objeto-p à qual o atributo pertence. Quanto à hierarquização clos

objetos-b, as próprias facilidades de herança encontradas em POO viabilizariam rapidamente a implementação desta faceta.

Pretendeu-se também, neste trabalho, introduzir uma nova forma de implementação de *shells*, que mostrou ser mais fácil de ser desenvolvida e apresentou eficiência na execução.

Capitulo V

Conclusões

V.1 Introdução

O objetivo deste capitulo é realizar uma análise conclusiva dos principais pontos abordados nesta dissertação. São apresentadas as contribuições obtidas e, em seguida, analisa-se os requisitos necessários à migração da. QUICK-SHELL para outros bancos de dados hospedeiros. Algumas extensões ao presente trabalho são sugeridas e, por fim, são feitos comentários comparativos a. respeito de oritros trabalhos correlatos.

V.2 Contribuições

O trabalho aqui desenvolvido apresenta três resultados principais: a. análise clos diversos aspectos de integração de bancos de dados e sistemas dedutivos, a especificação de um ambiente de integração baseado nos conceitos teóricos, e a concretização da especificação através da construção de um protótipo.

Análise

Verificou-se a tendência dos SGBD incorporarem mecanismos dedutivos no processamento de informação, bem como a necessidade de introduzir mecanismos de gerenciamento de memória secundária para sistemas especialistas, devido à expansão da base de fatos destes. No contesto desta tese, o enfoque ficou na integração de bancos de dados com sistemas especialistas. Foi feito um estudo preliminar sobre representação do conhecimento e modelagem de dados a. fim de proporcionar maior embasamento teórico. Algumas propostas de acoplamento e arquiteturas

foram descritas e analisadas a fim de enquadrar a. nossa proposta. naquela que se mostrasse mais adequada.

Especificação

O presente trabalho apresentou a especificação de um ambiente que acopla fortemente bancos de dados e sistemas especialistas. baseados no modelo E-R e O-A-V, respectivamente. A adoção desses modelos de abstração permitiu a definição de termos, literais, regras e mecanismos de quantificação que abrigam de forma natural os elementos O-A-V e E-R. Na realidade, as triplas 0-A-V podem ser entendidas também como instâncias de um conjunto de entidades ou relacionamento que participam de uma. base de fatos. Assim, poderíamos ter definido a base de fatos internamente ao banco de dados, formando uma base global.

Esta alternativa, apesar de simples. descaracterizaria a. independência da base de conhecimento com relação ao banco de dados. impedindo a. construção de sistemas especialistas que não fazem acesso ao mesmo. Além disso, como não é normal o emprego de fatores de certeza em bases de dados, o manuseio de conhecimento incompleto dificilmente seria. empregada no sistema como um todo.

Implementação

Ao longo deste trabalho, enfocou-se os aspectos comuns de programação orientada a objetos, representação do conhecimento e modelagem de dados. As três áreas tratam de abstração utilizando principalmente os conceitos de classes, hierarquias e herança. Desta forma,, a utilização de orientação a objetos na implementação gerou uma estrutura que facilitou bastante a. consecução desta fase. Como foi mencionado no capítulo III, a própria definição da base de regras como objeto e a forma. como elas são geradas já. implementa o algoritmo de encadeamento regressivo sem ter sido necessário implementá-lo explicitamente. Do ponto de vista dos bancos de dados, a implementação orientada. a. objetos proporcionou meios para a definição de uma interface de acesso simples e genérica.

V.3 Migração para outros Bancos de Dados

A proposta de integração apresentada, apesar de ser específica. pelo fato de estar vinculada ao modelo E-R e extensões, pode utilizar como banco de dados hospecleiro um SGBD previamente desenvolvido. Para, a realização desta migração,

mantendo o aspecto de acoplamento forte, dois requisitos fazem-se necessários. O primeiro consiste em gerar uma interface entre o banco de dados e o usuário de forma que este tenha a impressão que o banco de dados lógico continue calcado no modelo E-R. Provavelmente, esta interface consistiria. numa linguagem que alsrigasse os conceitos e operações do modelo E-R e que, uma vez traduzida, gerasse comandos do banco de dados hospedeiro. No contesto da QUICK-SHELL, a linguagem de regras seria traduzida em duas etapas: para a. linguagem intermediária *e,* em seguida, para os comandos do banco de dados em questão.

O segundo requisito consiste em desenvolver um Servidor E-R, analogamente ao descrito no capítulo IV que tivesse como função principal mapear os termos E-R em funções de acesso ao banco de dados, seguindo o raciocínio anterior.

V.4 Continuidade do Trabalho

A importância deste trabalho reside não somente nos resultados descritos anteriormente, mas em seu potencial de continuidade. Há várias extensões num trabalho desta natureza, entre as quais podemos citar as seguintes.

Interface com o Usuário

Da forma como a QUICIC-SHELL foi especificada, o sistema de aquisição de conhecimento corresponde às construções definidas pela linguagem de regras. Deste modo, o usuário tem que conhecer a priori a sintaxe e semântica da linguagem. Este fato, as vezes, torna desagradável o manuseio da *shell,* pois é natural a ocorrência de erros sintáticos e semânticos, principalmente para um usuário que não esteja suficientemente integrado na área de informática. Para superar esta clificulclacle, é conveniente desenvolver uma interface amigável de alto nível, que obtem informalmente as mesmas informações que seriam obtidas através das construções da linguagem de regras.

Inclusão de Fatores de Certeza

Este trabalho pode ser continuado pela inclusão de mecanismos de raciocínio aproximado sobre a base de conhecimento, permitindo o manuseio de informação incompleta.. Porém, como se trata de uma integração com banco de dados, é preciso

criar mecanismos similares na base de claclos. Isto C, de alguma forma, os claclos do banco de claclos devem estar associados a fatores de certeza. Atualinente, existe este conceito de modo implícito, e os fatores possíveis são 1 ou 0, correspondendo a existência ou não de determinado dado. Esta extensão ampliaria o universo de aplicações a utilizarem os serviços da QUICK-SHELL, além de ser uma área de pesquisa bastante fértil.

Estruturação dos Objetos da Base de Fatos

Outra extensão enriquecedora consiste na estruturação dos objetos O-A-V na forma de hierarquias conforme descrito no capítulo II. Desta forma., torna-se possível a armazenagem *e* manipulação de objetos mais complexos na base de fatos. A utilização de orientação a. objetos na implementação facilita. esta, extensão além de viabilizar a. implementação de *tipo de* dados no próprio ambiente da *shell.* Isto é, a definição de um objeto equivaleria a. especificação de um *tipo* Se dado e, conforme a situação, possuiria um determinado número de instâncias.

Atualizações da Base de Dados

Bancos de clados são normalmente orientados para o processamento de transações, que são caracterizadas por unidades de processamento. Entre o início de uina transação e seu final, o banco de dados pode passar por estados inconsistentes, embora seu estado resultante deva ser um estado consistente. Uina transação também caracteriza os elementos para a. recuperação de um banco de claclos por ocasião de falhas, cancelando a. transação ou executando-a completamente. Se considerarmos que as ações resultantes de uma regra. podem gerar alterações na base de claclos, será necessário considerar a ativação de procedimentos de *begin-transaction* e *end-transaction*, de modo a comunicar à base de claclos os limites de consistência clas operações e instruí-lo quanto ao procedimento de recuperação. Embora este problema resulte em algumas pequenas alterações na coclificação clas regras aqui introduzidas, ele não compromete a especificação proposta [40].

Migração para outros Ambientes

No momento em que a implementação e suas extensões atingirem um grau de maturidade razoável, deve-se estimulas a sua transposição para outros ambientes, respeitando os requisitos mencionados na seção V.3.

Armazenamento de Tipos de Dados Complexos na Base de Dados

As aplicações ditas não convencionais envolvem objetos complexos cuja estrutura se configura como uma composição hierárquica de partes. Os componentes de um objeto podem ser, por si, também objetos complexos. Além disso, pode haver uma composição recursiva de objetos e mesmo sobreposição de sub-objetos da hierarquia. A evolução natural do QUICK-DB consiste em adotar a filosofia de bancos de claclos orientados a objetos conforme aludido acima.

Desta. forma, torim-se necessário reavaliar os aspectos de representação do conhecimento na QUICIC-SHELL. A definição de tipos mais complexos com os respectivos métodos, no escopo do paradigma O-A-V, resolve satisfatoriamente esta questão. Outra proposta consiste na migração para. um modelo de representação mais complexo como redes semânticas.

V.5 Comparação com Outros Trabalhos

Em [4] foi mencionado que não há consenso sobre qual seria a. arquitetura ideal para um sistema que integra elementos dedutivos e de gerenciamento de dados. Existem idéias isoladas, a maior parte delas tendendo a obter rapidamente uma. implementação que permita conduzir experimentos e, de um modo geral, partindo dos aspectos de implementação para chegar a um modelo geral.

A maior dificuldade provém do fato de não ser possível ainda caracterizar precisamente o que é conhecimento e muito menos especificar que tipo de conhecimento deve ser absorvido pelos sistemas que acoplam sistemas especialistas e bancos de claclos. Outros problemas surgem, tais como o de adequar mecanismos de representação e inferência a,cada tipo de domínio. Além destes fatores, a implementação de tais sistemas ainda é um tópico pouco explorado.

Buscou-se na literatura sistemas que se assemalhassem às classificações teóricas aqui apresentadas. Algumas aplicações foram encontradas, citaclas no capítulo II. Porém, em poucos casos comentou-se os aspectos teóricos de tais sistemas. Isto é, como foram especificados os termos, literais. regi as. mecanismo de inferência, etc. Os artigos que aludiam a estes fatos normalmente davam um enfoque teórico bastante complexo, ultrapassando os limites desejáveis neste trabalho. Tais sistemas, em geral, abordavam a integração de linguagens lógicas e bancos de clados relacionais, no contesto de bancos de dados dedutivos.

O trabalho que mais se aproximou a nossa proposta é clescrito em [31]. O autor clescreve uin sistema que permite atributos da base de claclos calculados através

do processo de inferência. As bases de dados, entretanto, não são genericamente E-R como no nosso caso, além de persistirem as limitações de acesso impostas originalmente pelo modelo.

Assim, acredita-se que o conteúdo desta tese consiste de um trabalho onde a principal contribuição foi justamente uma proposta de um ambiente simples, tangível e aberto a extensões, que integra sistemas especialistas e bancos de dados. Tudo isso inserido num ambiente completo de projeto, manutenção e manipulação de bancos de dados construídos sobre o modelo Entidade-Relacionamento.

Bibliografia

[1] MANNA, Z.: *Mathematical Theory of Computation*, McGraw, Computes Science Series, 1974.

[2] RICH, E.: *Artificial Intelligence*, McGraw-Hill, 1983, USA

[3] LOPES, P. L.: Sistemas Especialistas; *Anais do XIX Congresso Nacional de Informática*; pp: 65-70;

[4] BRAGA, J. L.; MELO, R. N.: Sistemas para Gerenciamento de Bases de Conhecimento: Uma Evolução de SGBD; *Anais do Terceiro Simpósio Brasileiro de Banco de Dados;* 1988, pp: 180-190.

[5] SETZER, TV.: *Projeto Lógico e Projeto Físico de Bancos Se Dados*, V Escola de Computação, Belo Horizonte, 1986.

[6] FURTADO, A.L; NEUHOLD, E.J.: *Formal Techniques for Data Base Design*, Springer-Verlag, 1986, Berlin Heidelberg.

[7] DATE, C. J.: *Introduction to Database Systems* 4, ed. Reading: Addison-Wesley, 1986 U.S.A.

[8] MELO, R. M.: *Bancos de Dados não Convencionais: A Tecnologia do BD e suas áreas de Aplicação*; VI Escola de Computação, Campinas-UNICAMP, 1988.

[9] HARMON, P.; KING, D.: *Expert Systems*, John Wiley & Sons, Inc., USA, 1955.

[10] TRINKENREICH, H.: *Aspectos da Implementação de Interfaces Gráficas para Consultas a Bancos de Dados*; Tese de Ms.c., Programa de Engenharia de Sistemas, COPPE/UFRJ; Maio de 1991.

[11] CHEN, P.: The Entity-Relationship Model - Toward a Unified View of Data, *ACM Transactions on Database Systems 1*, (1976).

[12] PECKHAM, M.; MARYANSKI, F.: Semantic Data Models, *ACM Computing Surveys 20*, 3(September 1988).

[13] CLANCY, W.J.: *Knowledge-Based Tutoring: The GUIDON Program*, MIT Press, Cambridge, Massachusetts. 1987.

[14] HORN, N.V.: *Understanding Expert Systems*, Bantam Books Inc., New York, 1986.

[15] O'SHEA, T.; EISENSTADT: *Artificial Intelligence, Tools, Techniques and Applications*, Harper and Row, New York, 1984.

[16] SHIRAI, **Γ;** TSUJI, J,: *Artificial Intelligence Concepts, Techniques and Applications*; Pitman Press Ltd, England, 1982.

[17] NILSSON, N. J.: *Principles of Artificial Intelligence*, Sringer-Verlag, New York 1982.

[18] BRACHMAN, R. J.; LEVESQUE, H. J.: Tales form the Far Side of Krypton, Lessons for Expert Database Systems from Knowledge Representation; *Proceedings of the First International Workshop* In: Kerschberg,L. (ed), 1987, pp: 3-21.

[19] DEERING, M.: Database Suport for Storage os AI Reasoning Knowledge; *Expert Database Systems Proceedings of the First International Workshop* In: Kerschberg,L. (ed), 1986, pp: 527-537.

[20] GRADARIN, G.; VALDURIEZ, P.: *Relational Databases and Knowledge Bases*, Addison-Wesley Publishing Company, 1989.

[21] NAQVI, S. A.: *Logic and Database Systems*, Islamorada Surveys, 1985, pp: 58-65.

[22] SHALOM, T.; ZANIOLO, C.: LDL: A Logic-Based Data-Language; *Proceedings of the Twelfth International Conference on Very Large Data. Bases*, 1986, pp: 33-57.

[23] SMITH, J. M.: Expert Database Systems: A Database Perspective; *Expert Database Systems Proceedings of the First International Workshop* In: Kerschberg,L. (ed), 1986, pp: 3-15.

[24] BOSE, P,K; RAJINIKANTH, M.: KARMA: Knowledge-Based Assistant to Database System; *Proceeedings of the Second Conference om Artificial Intelligence Applications*, Dezembro. 1985. pp 467-473.

[25] JARKE, MATTHIAS, VASSILIOU, YANNIS: Coupling Expert Systems with database Management Systems. In: Reitman, TV. (ed). *Artificial Intelligence Applications for Business*, Norwood, NJ:Ablex, 1985, cap. 5 p. 65-85.

[26] ABARBANEL, R,M.; WILLIAMS, M.D.: A Relational Representation for Knowledge Bases, *Expert Database Systems; Proceedings of the First International Workshop* In: Kerschberg,L. (ed), 1987, pp: 191-205.

[27] SILVEIRA, P.; M.: A Formalization of The E-R Model, *Proceedings of the IX Conferencia Internncionnl de la Sociedad Chilena de la Ciencia de la Computation*, Santiago, Chile, 1989.

[28] SILVEIRA, P. M.: Procedural Data Manipulation Operations for the E-R Model, *Proceedings of the XVI Conferência Latino Americana de Informática*, Assunção, Paraguay, 1990.

[29] WINSTON, P. H.: *Artificial Intelligence*, Addison-Wesley, 1984, U.S.A.

[30] BRODIE, M.L; MYLOPOULOS, J.: *On Knowledge Base Management Systems*, Springer-Verlag, 1986, New York.

[31] KERSCHRBERG, L.: KORTEX: An Expert Database System Shell for knowledge based entity relationship model, *Int. Conf. on the Entity/Relationship Approach*, pp 174-187, 1989.

[32] ICERSCHERBERG, L.: Expert Database Systems: Knowledge/Data Management Environments for Intelligent Information Systems, *Information Systems* Vol. 15, No. 1, pp 151-160. 1990.

[33] ZANIOLO, C.: Object Oriented Database Systems and Knowledge Systems; *Expert Database Systems; Proceedings from the First International Workshop* In: Kerschberg,L. (ed), 1986, pp: 49-65.

[34] DITTRICH, K. L.: Object-Oriented Database Systems, A Workshop Report; *Proceedings of the 5th ER Conference*, Dijon; North Holland Publ. Group; 1986.

[35] BORLAND: Turbo Pascal 5.5; *Object Oriented Programming Guide*; Scotts Valley, Ca, 1989.

[36] MEYER and BERTRAND : *Object-Oriented Software Construction* - New York Printice-Hall, 1988.

[37] TADAO, T.; HANS, K. E. L.: *Programaçâo Orientada a Objetos,* VII Escola de Computação; São Paulo; 1990.

[38] JOHNSON, S. C.: *Yacc: Yet Another Compiler-Compiler*; Bell Laboratories, Murray Hill, New Jersey, 1978.

[39] AHO, A.V.; ULLMAN, J,D.: *Principles of Compiler Design*; Addison-Wesley Publishing Company, 1977.

[40] PACITTI, E.C., SILVEIRA, P.M.: Um *Ambiente que Integra Sistemas Especialistas e Bancos de Dados Baseados no Modelo E-R* , Relatório Técnico NCE-24/90, Outubro/90.

APÊNDICE A - EXEMPLO DE UTILIZAÇÃO

```
% Base SBC
% Objetos


cand = objeto
     tipo :  alfa [INFERI ;
     participacao :  alfa [PERG];
     nome :  alfa [PERG];
     num :  alfa [INFER];
     sucesso :  alfa [INFERI ;
fim


condicao = objeto
         primeira:  alfa [INFERI ;
         segunda:  alfa[INFERI
         terceira :  alfa [INFER];
         quarta :  alfa [INFERI;
         quinta alfa [INFER];
fim


% variaveis

reg :  REGIONAL;
com :  COMISSAO;
ins :  INSCRICOES;

USA SBC
#

Regra 1

Se 'TRUE' = 'TRUE'
entao
   num(cand) = num(^(SOCIO,alfabetica,nome(cand)))
```

Regra 2

```
Se titulacao(∧(SOCIO,numerica,num(cand))) = >DOUTORADO>
entao
   tipo(cand) = 'DOUTOR'
```

Regra 3

```
Se titulacao(∧(SOCIO,numerica,num(cand))) = 'MESTRADO'
entao
   tipo(cand) = 'MESTRE'
```

Regra 4

```
Para algum reg em REGIONAL por REGIONAL.alfabetica
Se num(delegado(reg)) = num(cand)
entao
   primeira(condicao) = 'TRUE'
```

Regra 5

```
Se tipo(cand) = 'DOUTOR'
entao
   segunda(condicao) = 'TRUE'
```

Regra 6

```
Se tipo(cand) = 'MESTRE' e
   publicacoes(∧(SOCIO,numerica,num(cand))) > 10
entao
   segunda(condicao) = 'TRUE'
```

Regra 7

```
Se tipo(cand) = >DOUTOR' e
   tempo(∧(SOCIO,numerica,num(cand))) > 3
entao
   terceira(condicao) = 'TRUE'
```

Regra 8

```
Se tipo(cand) = >MESTRE e
   tempo(∧(SOCIO,numerica,num(cand))) > 6
entao
   terceira(condicao) = 'TRUE'
```

Regra 9

```
Para algum com em COMISSAO por COMISSAO.alfabetica
Para algum ins em INSCRICAO por inscricoes(com)

Se num(cand) = num(socio(ins) )
entao
   quarta(condicao) = 'TRUE'
```

Regra 10

```
Se tipo(cand) = 'MESTRE' e
   participacao(cand) = >EXCELENTE>
entao
   quinta(condicao) = 'TRUE'
```

Regra 11

```
Se tipo(cand) = 'DOUTOR' e
   participacao(cand) = 'BOA'
entao
   quinta(condicao) = 'TRUE'
```

Regra 12

```
Se primeira (condicao)  = 'TRUE' e
   segunda (condicao)  = 'TRUE' e
   terceira (condicao)  = 'TRUE' e
   quarta (condicao)  = 'TRUE' e
   quinta (condicao)  = 'TRUE'
entao
   sucesso(cand) = 'TRUE'
```

```
#

% Comandos
Determine :  sucesso(cand);
mostre
```

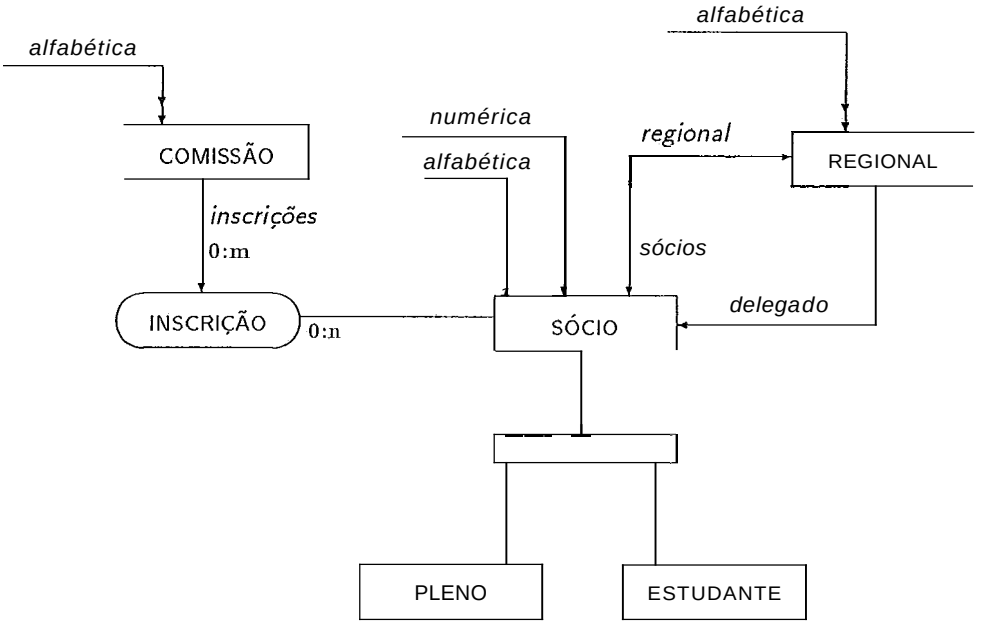

Figura .1: Exemplo de um Esquema E-R

Entidades	COMISSÃO
	SÓCIO
	REGIONAL
	PLENO
	ESTUDANTE
Relacionamentos	INSCRIÇÃO
Atributos	num(SÓCIO) → inteiro
	titulação(SÓCIO) → alfa
	publicações(SÓCIO) → inteiro
	nome(SÓCIO) → alfa
	tempo(SÓCIO) → int
	endereço(SÓCIO) → alfa
	cep(SÓCIO) → alfa
	inscrição(SÓCIO) ▪→ alfa
	código(COMISSÃO) → alfa
	nome(COMISSÃO) → alfa
	curso(ESTUDANTE) → alfa
	término(ESTUDANTE) → alfa
	ocupação(PLENO) → alfa
	taxa(INSCRIÇÃO) → real
	ano(INSCRIÇÃO) → alfa
	nome(REGIONAL) → alfa
Atributos Referenciais	regional(SÓCIO) ▪→ REGIONAL
	sócios(REGIONAL) ⇒ SÓCIO
	delegado(REGIONAL) → SÓCIO
Ordens	alfabética ⇒ COMISSÃO
	inscrições(COMISSÃO) ⇒ INSCRIÇÃO
	alfabética ⇒ SÓCIO
	numérica ⇒ SÓCIO
	alfabética ⇒ REGIONAL
Generalizações	PLENO é um SÓCIO
	ESTUDANTE é um SÓCIO
Funções Papel	comissão(INSCRIÇÃO) ▪→ COMISSÃO
	sócios(INSCRIÇÃO) → SÓCIO

Figura .2: Elementos do Esquema E-R exemplo

APÊNDICE B - SINTAXE DA LINGUAGEM DE REGRAS

```
programa    :   universo '#' cregra '#'cinst '#'comando EOFILE
            |   universo '#' cregra '#' comando EOFILE
            ;

cregra      :   cregra regra
            |   regra
            ;

regra       :   nregra ccleclaracao SE cliteral ENTAO acao
            |   nregra SE cliteral ENTAO acao
            ;

ccleclaracao :  cdeclaracao declaracao
            |   declaracao
            ;

declaracao  :   PARA ALGUM termo EM termo POR termo
            ;

cliteral    :   cliteral E literal
            |   literal
            ;

literal     :   termo op termo
            ;

ord         :   ID
            ;

var_er      :   ID
            ;
```

```
acao :      ID '(' ID ')' '=' termo
     |      ID
     ;

termo:      ID
     |      ID '.' ID
     |      ID '(' args ')'
     |      ACONST
     |      ICONST
     ,

args :      ID
     |      ID '(' args ')'
     |      ^ '(' ID ',' ID ',' cargs ')'
     ;

cargs :     cargs ',' chargs
     |      chargs
     ;

chargs      ID
     |      ID '(' chargs ')'
     |      ICONST
     |      ACONST
     '

nregra      REGRA ICONST
     ,

op   :      LT
     |      LE
     |      GT
     |      GE
     |      NE
     |      EQ
     ;
```

```
universo    cobj cvar basebd
        |   cobj
        ;

cobj    :   cobj obj
        |   obj
        ;

cvar    :   cvar var
        |   var
        ;

obj     :   ID '=' OBJ catrs FIM
        ;

catrs   :   catrs atr
        |   atr
        ;

atr     :   ID ':' tipo '[' infr ']' ';'
        |   ID ':' tipo ';'
        ;

tipo    :   I

        |   ALFA
        ;

infr    :   INFER
        |   PERG
        ;

var     :   var_er ':' ID ';'
        ;

basebcl     USA ID
        ;
```

```
comando   DETE CTE MOSTRE
      |   DETECTE
      ,

cinst :   cinst inst
      |   inst
      |   '#'
      ,
inst  :   TAL QUE ID ':' ID '=' ACONST POR ord ';'
      |   TAL QUE ID ':' ID '=' ICONST POR orcl ';'
      ,

CTE   :   CTE TE
      |   TE
      ,

TE    :   ID '(' ID ')' ';'
      ;

}%
```