UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cristiano Agosti

Interface em Linguagem Natural para Banco de Dados: uma abordagem prática

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Luiz Fernando Jacintho Maia

Florianópolis, agosto de 2003

Interface em Linguagem Natural para Banco de Dados: uma abordagem prática

Cristiano Agosti

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração Sistemas de Conhecimento e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Raul S. Waslawick (coordenador)

Banca Examinadora

_____

Prof. Dr. Luiz Fernando Jacintho Maia (orientador)

_____

Prof. Dr. Ilson W. Rodrigues Filho

_____

Prof. Dr. Marco Antonio da Rocha

_____

Prof. Dr. João Bosco da Mota Alves

"Há homens que lutam um dia e são bons.
Há outros que lutam um ano e são melhores.
Há os que lutam muitos anos e são muito bons.
Porém, há os que lutam toda a vida.
Esses são os imprescindíveis."
(Bertolt Brech)

Para minha esposa Miriam,
meus pais Valdir e Neide
e minha irmã Ana Paula que me
acompanharam neste desafio.

Agradecimentos

A Unoesc campus Xanxerê, instituição a qual pertenço, por todo o incentivo recebido.

Ao professor orientador, Dr. Luiz Fernando Jacintho Maia, pelo seu companheirismo e incentivo na realização desta dissertação de mestrado.

Ao professor, Dr. Ilson W. Rodrigues Filho pela co-orientação prestada durante a elaboração deste.

Ao professor Msc. Davidson Mazzoco Davi, coordenador do curso de Tecnólogo em Informática, curso a qual leciono, por todo o apoio prestado e pela confiança dada ao meu trabalho.

Aos meus pais, e minha esposa pela força prestada nos momentos críticos desse trabalho.

A Deus, por permitir mais essa realização.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ATN – Rede de Transição Aumentada;

BD – Banco de Dados;

DBA – Administrador de Banco de Dados;

DCG – Gramática de Cláusula Definida;

EFL - Expressão Semântica Ambígua;

GUI – Interface Gráfica para Usuários;

IA – Inteligência Artificial;

IAP – Item Aproximado;

ILN – Interface em Linguagem Natural;

ILNBD - Interface em Linguagem Natural para Banco de Dados;

INESC – Instituto de Engenharia de Sistemas e Computadores;

IPR – Item Principal;

IS – Item Semântico;

LIL – Linguagem de Interrogação Lógica;

LN – Linguagem Natural;

LQL - Linguagem de Consulta do Masque;

MC – Módulo de Consulta;

MER – Modelo de Entidade e Relacionamento;

MG – Módulo Gerenciador;

MRL – Meaning Representation Language;

NP – Nome Próprio;

PLN – Processamento da Linguagem Natural;

RTN – Rede de Transição Recursiva;

SAdj – Sintagma Adjetival;

SAdv – Sintagma Adverbial;

SGBD – Sistema Gerenciador de Banco de Dados;

SN – Sintagma Nominal;

SP – Sintagma Preposicional;

SQL – Structured Query Language (Linguagem de Consulta Estruturada);

SV – Sintagma Verbal;

UA – Usuário Administrador;

UF – Usuário Final.

UML – Linguagem de Modelagem Unificada;

WML - Expressão Semântica não Ambígua;

Resumo

Este trabalho apresenta uma dissertação de mestrado desenvolvida dentro da linha de pesquisa Inteligência Computacional do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Será mostrado um estudo sobre o Processamento da Linguagem Natural e sua aplicação no acesso aos dados contidos em um Banco de Dados. Para que isso fosse possível foram analisados diversos softwares disponíveis no mercado e que possuem essa característica, além do estudo de trabalhos científicos desenvolvidos com esse objetivo.

Para finalizar apresenta-se uma arquitetura de interface, em linguagem natural, para acesso a um banco de dados relacional, que objetiva a obtenção de informações em relação à pesquisa de usuários que não tenham conhecimento da linguagem de consulta estruturada (SQL) e que não sejam familiarizados com as atuais interfaces visuais de consulta.

O trabalho foi integrado a um sistema usado para formular perguntas em linguagem natural através da internet.

Palavras-chave: Processamento da Linguagem Natural, Interface em Linguagem Natural para Banco de Dados, Linguagem de Consulta Estruturada (SQL).

Abstract

This work presents a master's degree dissertation developed inside of Computacional Intelligence research line of the Course of Masters degree in Computer Science of the Federal University of Santa Catarina.

It will be shown a study about of Natural Language Processing and its application in the access to the data stored in a database.

So that this was possible they had been analyzed a lot of softwares available in the market and that they possess this characteristic beyond the study of developed scientific works with this objective.

To finish an architecture of natural language interfaces to databases is presented, that serves as half of attainment of information the levels of research for users who does not have knowledge of the Structured Query Language (SQL) and that they are not made familiar to the current visual query interfaces.

The work was integrated system that is used to formulate questions in natural language through the internet.

Keywords: Natural Language Processing, natural language interfaces to databases, Structured Query Language (SQL).

Capítulo 1

Introdução

A comunicação com o computador foi sempre uma barreira difícil de ultrapassar. Os usuários para conseguirem tirar partido dos meios disponibilizados pelo computador, demoram a adaptar-se, e em muitos casos, têm que disponibilizar um valioso tempo para conseguir aprender todos os recursos que a tecnologia tem a oferecer. Essa limitação exige da parte dos projetistas de sistemas um esforço para sistematizar e facilitar a ligação entre sistema e usuário.

A necessidade de se buscar informações que se encontram em uma base de dados é cada vez mais comum. Os sistemas de gerenciamento de banco de dados (SGBD), mais comuns possuem uma linguagem que faz com que, se bem estudadas, possam extrair as informações do mesmo; no entanto, isso exige bastante conhecimento por parte do utilizador.

Neste sentido a busca de informações utilizando uma linguagem conhecida como a língua portuguesa, por exemplo, torna-se uma alternativa bem interessante.

1.1. Objetivos

1.1.1. Objetivo Geral

O objetivo deste trabalho é estudar as diversas formas de se fazer acesso ao banco de dados, utilizando para isso a linguagem natural e implementar um programa de computador para poder aplicar em um experimento prático o que foi estudado.

1.1.2. Objetivo Específico

- Descrição de técnicas de Processamento da Linguagem Natural;
- Estudo de formas de acesso ao banco de dados usando Linguagem Natural;
- Pesquisa sobre softwares comerciais e trabalhos científicos relacionados;
- Descrição sobre softwares comerciais e trabalhos científicos relacionados;
- Elaborar, a partir das considerações feitas sobre o que já existe em estudo, um sistema de acesso ao banco de dados em Linguagem Natural, independente do tipo de sistema gerenciador de banco de dados.

1.2. Justificativa

A possibilidade de existir um software que não necessite treinamento ao usuário e que consiga buscar as informações que procura em um sistema é o que justifica este trabalho. A possibilidade de o usuário perguntar o que ele precisa para um software.

1.3. Conteúdo deste Documento

Este documento foi desenvolvido da seguinte forma:

No capítulo 2 são apresentadas as formas de se fazer o processamento da linguagem natural, o detalhamento de uma arquitetura, e relatados problemas que esta área tem para resolver.

No capítulo 3 dá-se uma visão sobre como foram feitos os atuais trabalhos envolvidos com a utilização da linguagem natural para o acesso a dados em um banco de dados. São discutidas algumas arquiteturas existentes e os principais problemas envolvidos na mesma.

O capítulo 4 apresenta o estudo dos primeiros softwares criados nesta área, alguns softwares comerciais e também trabalhos científicos pesquisados.

Para o capítulo 5 será discutido uma proposta e um software para interface em linguagem natural para banco de dados desenvolvido a partir de conclusões da pesquisa.

No capítulo 6, apresentam-se as considerações finais, que relatam os resultados obtidos com este trabalho, bem como são propostos novos trabalhos.

Capítulo 2

Processamento da Linguagem Natural

2.1. Introdução

A busca por entender os mecanismos da língua iniciou-se com os primeiros estudos de gramática na Grécia antiga, tendo uma abordagem mais formal através dos estudos de Ferdinand de Saussure e desenvolveu-se notoriamente através dos trabalhos de Frege, Noam Chomsky e Richard Montague.

O interesse em dotar um sistema computacional com a capacidade de entender os objetivos do usuário em sua própria linguagem surgiu juntamente com os primeiros sistemas. Allan Turing, um dos maiores teóricos da computação, definia a inteligência dos computadores através da capacidade destes últimos em lidarem com a linguagem natural. A capacidade de processar linguagem natural, portanto, vem sendo pensada praticamente desde o advento dos computadores. Embora a máquina de Von Neumann tenha sido imaginada para aplicações numéricas, Turing já entendia o computador como um recurso com capacidades inteligentes, que o apoiaria em atividades como jogar xadrez ou teria, inclusive, habilidade para compreender e produzir linguagem natural.

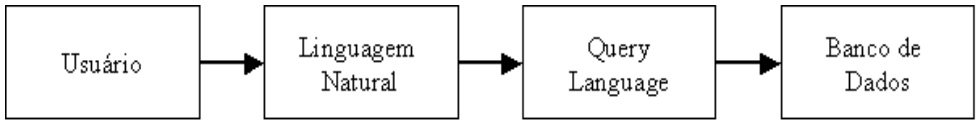
Neste capítulo é realizado um estudo sobre o Processamento da Linguagem Natural (PLN); serão discutidas as estruturas básicas necessárias a um sistema que utiliza o PLN como recurso de linguagem natural, bem como, suas formas de representação formal. Ao final serão relatados os principais problemas do PLN.

2.2. Conceito

O Processamento da Linguagem Natural (PLN) é um ramo da Inteligência Artificial (IA) que tem por objetivo interpretar e gerar textos em uma língua natural como, por exemplo, o Português.

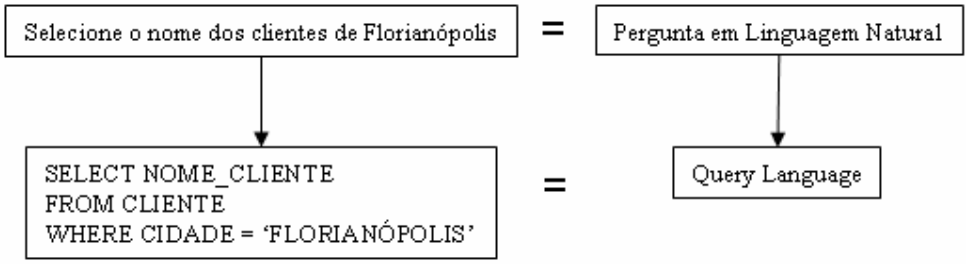
Uma das áreas de aplicação das técnicas do PLN é o acesso ao banco de dados, que normalmente é feito através do uso de "query languages", conforme demonstrado na figura 2.1, ou seja, uma linguagem de consulta ao banco de dados, em que uma das mais conhecidas e utilizadas é a linguagem de consulta estruturada SQL (Structured Query Language).

Figura 2.1 - Represetação da Query Language



A figura 2.2 demonstra um exemplo de uma Query Language, com uma pergunta feita em linguagem natural e sua representação usando uma query language.

Figura 2.2 - Exemplo de Query Language



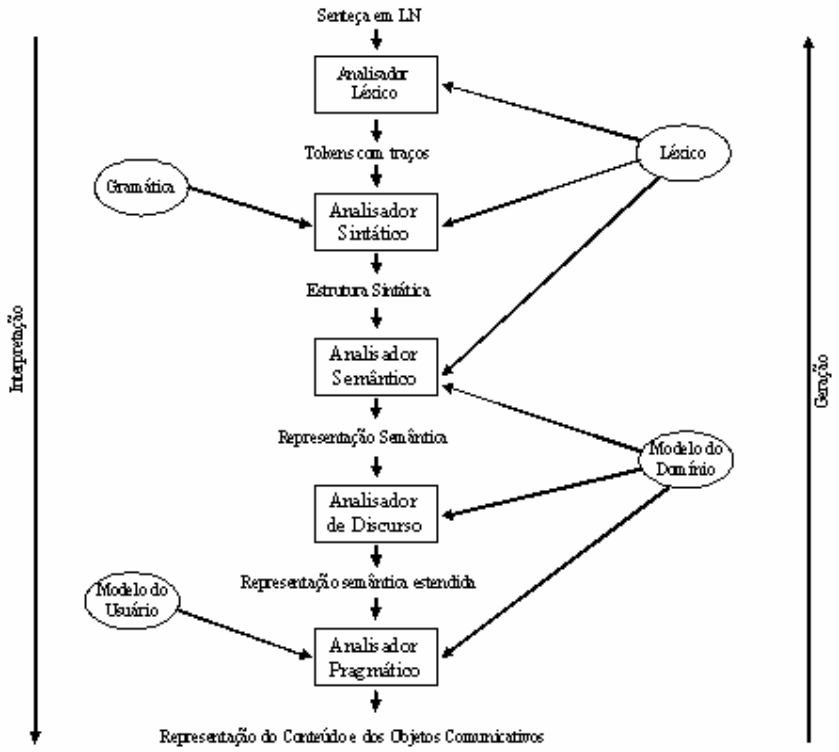
A pesquisa em processamento de linguagem natural tem dois objetivos:

a) construção de modelos computacionais da língua para facilitar a comunicação entre o ser humano e a máquina;
b) utilização do computador para validar as teorias lingüísticas.

O processo de compreensão da linguagem natural é composto por diversos componentes da figura 2.3 que são: análise morfológica, análise sintática, análise semântica, análise de discurso e análise pragmática. Os limites entre essas cinco fases

são normalmente bastante obscuros. As fases são às vezes executadas em seqüência e uma poderá precisar da assistência da outra.

Figura 2.3 - Arquitetura de um Sistema de Interpretação de Língua Natural



Fonte: Nunes et al.,1999.

2.3. Estruturas Auxiliares

Para que as fases do processamento da linguagem natural possam ser realizadas são necessárias estruturas auxiliares, como:

2.3.1. Léxico ou Dicionário

É uma estrutura fundamental para a maioria dos sistemas e aplicações. É a estrutura de dados contendo os itens lexicais e as informações correspondentes a estes itens. Em realidade, os itens que constituem as entradas em um léxico podem ser palavras isoladas (como lua, mel, casa, modo) ou composições de palavras, as quais

reunidas, apresentam um significado específico (por exemplo, lua de mel ou Casa de Cultura).

Existem vários formalismos para representação que estarão no léxico, e deverão ser escolhidos de acordo com a representação da gramática do sistema.

Abaixo um exemplo de como o léxico poderá ser representado:

cadeira

<categoria> = substantivo

<gênero> = feminino

<número> = singular

sentou

<categoria> = verbo

<tempo> = pretérito-perfeito

<número> = singular

<pessoa> = 3

2.3.2. Gramática

Conforme (Chomsky, N., 1965), existe por trás da língua, de um modo não palpável, um conjunto de generalizações, princípios e regras abstratas em número finito, que determinam as frases da língua, a sua gramaticalidade, suas propriedades e características. Esse conjunto altamente organizado chama-se gramática.

É uma estrutura que define, através de regras, quais são as cadeias de sentenças válidas em uma língua. Essa verificação é feita em termos de categorias sintáticas, e não de uma lista exaustiva de frases. Existem vários formalismos para representar as regras gramaticais. Segundo (Rabuske, R. A., 1995) uma gramática é formalmente representada pela quádrupla:

$G = (V_N, V_T, P, S)$, onde:

V_N é o conjunto das variáveis (não terminais).

V_T são símbolos terminais.

P são as regras de geração.

S é o símbolo inicial.

Sendo que, $\mathbf{V = V_N \cup V_T}$, é o conjunto finito que estabelece o vocabulário total.

Se relacionarmos com a língua portuguesa então poderíamos dizer que $\mathbf{V_N}$ são as categorias sintáticas (frase verbal, frase nominal, etc...), $\mathbf{V_T}$ são as palavras e conectivos, **P** são as regras gramaticais de uma frase, por exemplo, a frase deverá iniciar por artigo, seguida por um nome e finalizada com verbo, e por fim **S** que seria a própria "frase".

Abaixo um exemplo de formalismo:

FRASE_NOMINAL → Substantivo, Adjetivo
<Substantivo gênero> = <Adjetivo gênero>
<Substantivo número> = <Adjetivo número>

Essa regra indica que uma frase poderá conter um substantivo seguido de um adjetivo, por exemplo: cadeira azul. As restrições <Substantivo gênero> = <Adjetivo gênero> e <Substantivo número> = <Adjetivo número> determinam que o gênero e o número do adjetivo e do substantivo devem estar em concordância.

(Chomsky, N., 1965) estabelece uma hierarquia de gramáticas a partir do número e da natureza de símbolos que ocupam as posições das regras de produção que são: gramáticas regulares, livres de contexto, sensível ao contexto e irrestrita.

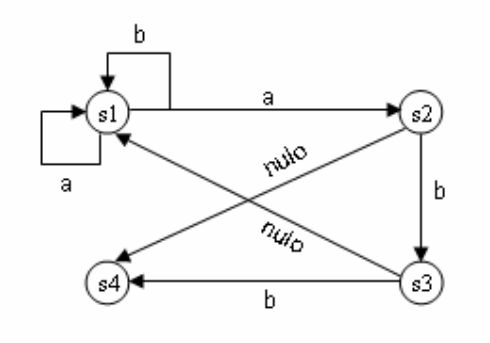
Gramáticas Regulares

Gramáticas Regulares (regular grammars) também conhecida como gramáticas do tipo 3, ou gramáticas de estados finitos (finite-state grammars), cujas regras se adequariam a uma de três formas: $\mathbf{V_N \rightarrow V_N\ V_T \mid V_T\ V_N \mid V_T}$. Estas gramáticas são bastante simples e facilmente reconhecidas, porém, apresentam um poder de expressão limitado, equivalente ao poder de expressão de um autômato finito.

Para (Palazzo, L. A. M., 1997), um autômato finito é uma máquina abstrata que lê, como entrada, um string de símbolos e decide se deve aceitar ou rejeitar o string lido. O autômato possui um certo número de estados e está sempre em um deles. Na figura

2.4 os círculos representam os "estados" do autômato; o mesmo muda de estado conforme a string de entrada for processada, e ele aceitará a mesma se após finalizada seu processamento a legenda dos arcos ao longo do caminho de transições corresponderem a string de entrada. O autômato apresentado é não-determinístico, pois a partir de um estado ele poderá passar para qualquer outro estado, respeitando a direção das setas; caso contrário poderíamos dizer que ele é um autômato finito determinístico.

Figura 2.4 - Autômato Finito Não-Determinístico



Fonte: Palazzo, L. A. M., 1997.

Gramáticas Livres de Contexto

Gramáticas livres de contexto (context-free grammars) ou gramáticas do tipo 2, cujas regras obedeceriam à sintaxe $\mathbf{V_N} \rightarrow \mathbf{x}$, em que **x** pode ou não ser um grupo de strings terminal são muito úteis no que tange à descrição de gramáticas em linguagem natural. Em geral, são mais poderosas que as regulares, permitindo a representação de linguagens com um certo grau de complexidade. No entanto, a dificuldade em expressar dependências simples (exemplo: concordância entre verbo e sintagma nominal) constitui um dos maiores problemas para sua utilização no tratamento da língua natural. Servem bem para descrever linguagens de programação, em especial para a construção de compiladores. Abordagens puramente livres de contexto não são suficientemente poderosas para captar a descrição adequada deste gênero de linguagem.

Gramáticas Sensíveis ao Contexto

Gramáticas sensíveis ao contexto (context-sensitive grammars), ou gramáticas do tipo 1, cujas regras seriam da forma $\mathbf{x} \rightarrow \mathbf{y}$, em que o comprimento de **y** é maior ou igual ao comprimento de **x**. Ainda assim, as gramáticas sensíveis ao contexto não abordam satisfatoriamente o tratamento de restrições gramaticais (Rich, E. 1993). O

impedimento para sua utilização, contudo, reside na questão do reconhecimento. O problema de decidir se uma sentença pertence a uma gramática sensível ao contexto é uma função exponencial sobre o tamanho da sentença, o que torna a implementação do procedimento de verificação uma questão complexa, do ponto de vista computacional.

Gramáticas Irrestritas

Gramáticas irrestritas (unrestricted grammars) ou gramáticas do tipo 0, cujas regras não seguiriam qualquer padrão. Segundo (Chomsky, N., 1965), as gramáticas de tipo 0 serviriam à descrição de qualquer tipo de língua, mas seu excessivo poder descritivo seria de pouca utilidade na compreensão dos fenômenos lingüísticos, porque estariam contempladas, na gramática, mesmo as sentenças que não pertencem à língua que se pretende descrever. O ideal seria a utilização de formalismos gramaticais menos poderosos, capazes de produzir apenas as sentenças efetivamente aceitáveis para uma determinada língua. No caso da maior parte das línguas naturais, incluído o português, acredita-se que as gramáticas livres de contexto sejam as mais adequadas.

A maioria dos trabalhos nesta área propõe, atualmente, trabalhar em modelos que se situem em um nível intermediário entre as gramáticas livres de contexto e as sensíveis ao contexto, aliando boa capacidade de representação, incluindo construções que permitam modelar dependências, e um modelo computacional viável.

A partir da idéia inicial de Chomsky, conforme (Rabuske, R. A., 1995), outros tipos de gramáticas e de mecanismos de representação foram desenvolvidos, dentre eles podemos citar: Gramáticas Transformacionais, Redes de Transição Aumentadas (ATNs), Gramáticas de Casos, Gramática Semântica e Gramáticas Lógicas.

Gramáticas Transformacionais

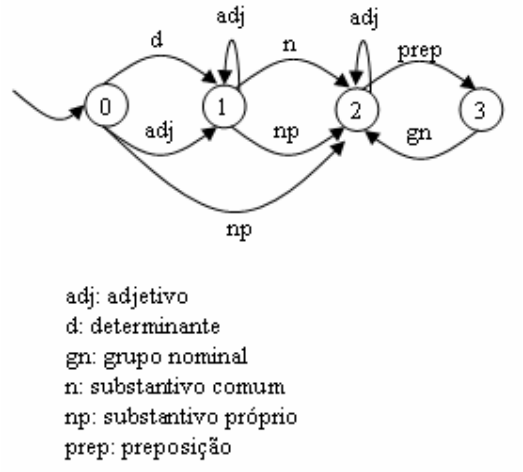
Segundo (Rabuske, R. A., 1995) a base desta gramática seria composta por três componentes, constituidores da competência lingüística tratados em separados, que são: o sintático, semântico, e fonológico. O componente básico de uma gramática livre de contexto, um conjunto de regras de reescrita na forma de árvores, fazendo intercalações ou eliminações de morfemas.

Redes de Transição Aumentadas ATNs

As ATNs são formas de representação para gramáticas de linguagens naturais. São representadas por grafos e estão baseadas nos mesmos autômatos finitos que emergem do processamento das gramáticas regulares. Elas são uma extensão das Redes de Transição Recursivas (RTNs) que têm seu potencial equivalente às gramáticas livres de contexto.

Como um autômato finito, uma rede ATN consiste em uma coleção de estados e arcos, um estado inicial distinto e um conjunto distinto de estados finais. Os estados são conectados uns aos outros pelos arcos, criando-se assim, um grafo direcionado ou rede. O nome no arco indica um símbolo terminal ou tipos de palavras que devem ocorrer na sentença de entrada para permitir a transição para o próximo estado. Uma sentença é dita aceita por uma rede ATN se existe uma seqüência de arcos conectando o estado inicial com o estado final que pode ser seguido pela sentença.

Figura 2.5 - Exemplo de Rede ATN



Fonte: Coulon, D., 1992.

A rede apresentada na figura 2.5 deverá ser capaz de analisar grupos nominais compostos por um nome comum ou próprio, um determinante, adjetivos, ou preposições.

A análise para a sentença “o grande livro azul de Pedro” ficará da seguinte forma (considerando substantivo, natureza, cor, forma, quantidade): inicialmente o autômato encontra-se no estado inicial representado na figura 2.5 como 0 (zero); segundo a natureza dos constituintes encontrados, seu estado será alterado ao longo do caminho

seguido no grafo, até atingir o estado final representado pelo número 2 (dois). Como a primeira palavra é um determinante então o arco “d” é executado e como natureza temos um artigo, considerando um léxico.

A seguir serão analisadas as próximas palavras, inclusive “de”, conduzindo ao estado 3 e com o seguinte resultado: adjetivos (grande, azul); substantivo (livro, pedro); preposição (de), características (tamanho, cor); natureza (objeto, pessoa).

Ao finalizar teremos após a saída do autômato:

Substantivo: livro

Natureza: objeto

Tamanho: grande

Cor: azul

Quantidade: 1

Substantivo: Pedro

Natureza: pessoa

Quantidade: 1

Conseguindo a rede ATN analisar a frase passada.

Gramáticas de casos

Segundo (Coulon, D., 1992) é um tipo de gramática que relaciona a interpretação sintática com a semântica. Sua existência deve-se ao fato de tentar solucionar alguns problemas deixados pela gramática transformacional.

As frases abaixo exemplificam a gramática:

João corta o papel.

Pedro corta o papel.

Essas frases poderão facilmente juntadas em João e Pedro cortam o papel. Essa transformação já não é possível com a sentença: A tesoura corta o papel, pois João e a tesoura cortam o papel não é aceitável.

A rede ATN pode ser utilizada como forma de implementação. A grande vantagem na sua utilização está no fato de que sua análise deixa muito mais claro o sentido da frase. No entanto, conforme (Rich, E., 1988), ainda não chegou a um consenso a respeito do número exato de casos que uma determinada língua pode possuir. Alguns exemplos são: agente (tipicamente animado, instigador da ação), factivo

(local do evento), meta (local para onde se desloca), tempo (tempo em que o evento ocorre ou ocorreu).

Gramáticas Lógicas

Conforme (Rabuske, R. A., 1995), entre os diversos tipos de gramáticas lógicas, as Gramáticas de Cláusulas Definidas (DCGs) são as que assumiram maior destaque. As mesmas são generalizações das gramáticas livres de contexto e são semelhantes à linguagem PROLOG.

Uma DCG para a frase "o jogador chuta a bola" é a seguinte:

sentença → frase_nominal, frase_verbal.

frase_nomimal → verbo, frase_nominal.

frase_verbal → verbo, frase_nominal.

artigo → [o].

artigo → [a].

nome → [jogador].

nome → [bolsa].

verbo → [chuta].

Para analisar se a frase é valida, ou não, basta chamar o predicado[1] "sentença", passando como argumento a frase, como no exemplo abaixo:

sentença([o,jogador,chuta,a,bola],X).

O argumento "X" é exigência do PROLOG, pois através do predicado "expand_term" traduz uma sentença como: senteça frase_nominal, frase_verbal em senteça(SO,S) :- frase_nominal(SO,SI), frase_verbal(SI,S).

Gramática de Montague

Conforme (Hinrichs, 1988) a arquitetura gramatical foi desenvolvida por Richard Montague, em 1973. Baseado no princípio da composicionalidade[2] de Frege, criou-se regras recursivas de representação sintática e semântica de expressões em linguagem natural, representadas no formato de árvore. A sintaxe foi tratada utilizando

[1] As propriedades dos objetos e os relacionamentos entre objetos, em Lógica, é feita através de uma estrutura chamada de predicado.

[2] Segundo Gottlog Frege, o princípio da Composicionalidade diz que, dada uma linguagem, a interpretação de construções complexas é determinada pela interpretação das partes individuais da linguagem e da maneira em que são colocadas em conjunto.

um conjunto de tipos lógicos, categorias sintáticas e as regras de geração de expressões da linguagem. Para dar conta de contextos oblíquos em frases foram criados os conceitos de intensão e extensão, além, de introduzir quantificadores temporais e operadores modais em sua representação semântica.

A extensão de uma expressão corresponde a sua notação, ou seja, aos valores verdadeiros de uma fórmula ou conjuntos no caso de predicados. A intensão é o "significado" intuitivo da expressão.

Por exemplo, na expressão, "O FHC é o FHC" e "O FHC é o presidente" possuem a mesma notação (verdadeiro), mas precedidas por um determinante mudam o "sentido", ou seja, a proposição da expressão como em "Necessariamente, o FHC é o FHC" e "Necessariamente, o FHC é o presidente".

Algumas tendências gramaticais, como a geração da estrutura gramatical da frase tem incorporado aspectos da gramática de montague.

Na figura 2.6, expressões sintáticas em inglês são traduzidas em uma linguagem de mais alto nível lógico, as quais são interpretadas em um modelo teórico. Em uma abordagem mais prática dos termos, um procedimento de tradução do inglês, em uma lógica que é baseada nos princípios da composicionalidade, tem especificamente duas partes: para cada item léxico, o procedimento tem que especificar uma tradução em um apropriado tipo de lógica e para cada regra sintática o procedimento tem que especificar como as traduções das entradas dos elementos combinam na tradução para as saídas dos elementos das regras sintáticas.

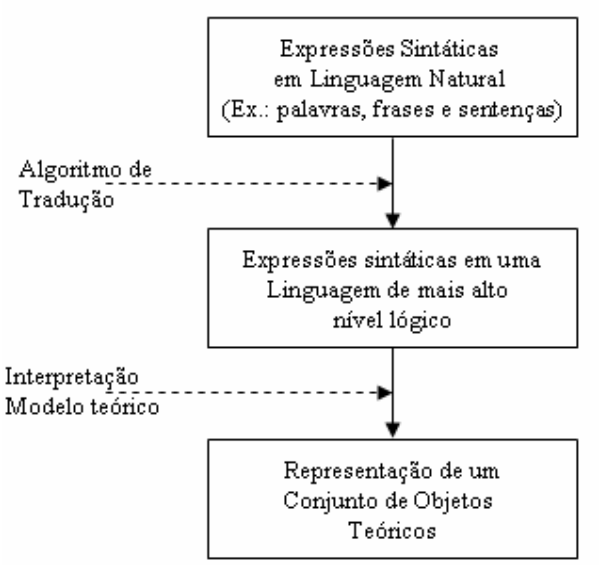
Além disso, a análise sintática e interpretação semântica podem proceder em paralelo, enquanto uma oração ou frase está sendo analisada gramaticalmente em acordo com as regras sintáticas e sua forma lógica pode ser computada por regras de tradução.

Abaixo a análise da frase "João procura um cachorro" segundo montague:

Se tomarmos como base um léxico que contenha substantivos comuns como "cachorro", podemos definir três regras F1, F2 e F3, tais que, se X é um substantivo, F1(X), F2(X) e F3(X) são os termos, "Todo X", "o X" e "um X", respectivamente formando novas sentenças. O segundo aspecto importante é a chamada regra de quantificação segundo a qual podemos obter uma sentença t* substituindo o primeiro pronome em t por um termo e fazendo os ajustes de gênero necessários. Desse modo, podemos obter uma sentença como "João quer um cachorro" tanto formando o verbo

intransitivo “quer um cachorro” para em seguida combiná-lo com “João” ou, alternativamente, “quantificando” “um cachorro” em “João quer ele”. Essas duas maneiras de formar a sentença vão corresponder aos sentidos não específico e específico da frase, respectivamente.

Figura 2.6 - Gramática de Montague



Fonte: Hinrichs, 1988.

2.3.3. Modelo do Domínio

Estrutura que fornece o contexto enciclopédico, armazenando conhecimento a respeito das entidades, relações, eventos, lugares, e datas do domínio, em algum formalismo de Inteligência Artificial (IA), como: Lógica de Predicados, Redes Semânticas, Frames, Scripts ou Hierarquias de tipos (Winston, 1992).

Através deste modelo o sistema terá subsídios para o correto inter-relacionamento semântico entre os componentes da frase, para a desambigüização lexical ou para a determinação de figuras de estilo ou figuras retóricas particulares, durante a análise do discurso.

2.3.4. Modelo do Usuário

Fornece o contexto interpessoal, armazenando conhecimento a respeito do usuário do sistema, como por exemplo: seus objetivos, planos, intenções, sua função, status, conhecimento do domínio, etc), através de representações como planejamento hierárquico ou atos da fala (Allen, 1995).

O grau de informação na geração textual depende do que é relevante ao leitor e, portanto, irá implicar em escolhas diversas de vocabulário, estruturas lingüísticas, etc.; o nível de conhecimento do assunto (superficial ou profundo) que o usuário apresenta pode levar a estruturas semânticas particulares, que, resultantes de um processo de parsing, podem auxiliar um sistema de consulta para, por exemplo, fornecer respostas em grau adequado de clareza.

2.3.5. Analisador Léxico

Conforme (Terra, E., 1992) a morfologia é uma parte da gramática que estuda a estrutura, processos de formação, flexão e classificação das palavras.

Em um sistema de interpretação da linguagem natural isto é feito por um módulo chamado analisador Léxico, morfológico ou Scanner. Sua finalidade é de identificar palavras (tokens) ou expressões isoladas em uma sentença, sendo esse processo auxiliado por delimitadores (pontuação e espaços em branco). As palavras identificadas são classificadas de acordo com sua categoria gramatical (exemplo: substantivos, pronomes, verbos, etc.). Esse processo denomina-se etiquetação.

Em relação à estrutura de formação das palavras, essas podem ser divididas em unidades significativas menores, ou morfemas, cuja manipulação pode gerar diversas variações para um mesmo vocábulo. São exemplos de morfemas: radicais, desinências nominais e verbais, prefixos e sufixos, etc. A manipulação de morfemas é especialmente interessante quando se deseja armazenar palavras de uma linguagem de forma condensada, em que ao invés de se armazenar todos os vocábulos, teriam apenas os morfemas necessários para sua criação. Essa idéia é muito atraente em nível computacional.

2.3.6. Analisador Sintático

A análise sintática em um sistema é feita através do analisador sintático ou parsing e consiste em criar uma árvore de derivação para cada sentença obtida no analisador léxico e mostrar como as palavras estão relacionadas entre si. Algumas seqüências de palavras podem ser rejeitadas se violarem as regras da linguagem sobre como as palavras podem ser combinadas. Tendo como exemplo, um analisador sintático do português rejeitaria a frase: "Ela foram a padaria hoje cedo".

A análise sintática de uma oração em português deve levar em consideração diversos tipos de sintagmas. Para (Savadovky, 1988) sintagmas são subdivisões intuitivas de orações de uma linguagem natural em que se percebe um significado claro. Cada sintagma tem uma palavra principal, que é chamada núcleo sintagmático e outras palavras dependentes desse núcleo, conforme tabela 1. Recursivamente, as palavras que acompanham o núcleo podem formar outros sintagmas.

Tabela 1 - Tipos de Sintagmas

Tipo de sintagma	Abreviação	Núcleo
Verbal	SV	Verbo (V)
Nominal	SN	Substantivo (Sub)
Preposicional	SP	Preposição (Prep)
Adjetival	SAdj	Adjetivo (Adj)
Adverbial	SAdv	Advérbio (Adv)

Fonte: Savadovisky, 1988

Na frase "João viu Maria" a palavra “Maria” forma um grupo com o verbo “viu”, isso é que caracteriza o sintagma verbal (SV) "viu Maria", que combinado com a palavra "João", nome próprio (NP) forma a sentença.

Para que um analisador possa processar essa frase faz-se necessário primeiro uma gramática para representar o fato que uma sentença (S) pode ser formada por um sintagma nominal (SN) seguido de um sintagma verbal (SV). A gramática deverá conter uma regra que permita combinar um verbo e um sintagma nominal para formar um sintagma verbal. Abaixo como ficaria a representação dessas regras, dentre outras:

S → SN, SV

SV → V, SN

Para determinar a estrutura, é preciso um léxico que relaciona as palavras com as categorias que pode lhe ser atribuídas:

Viu: V
Maria: NP
João: NP

Com essa gramática e esse léxico, pode-se identificar uma estrutura sintática para a sentença "João viu Maria", conforme a figura 2.7:

Figura 2.7 - Árvore Sintática para uma frase

Nos sistemas de processamento de linguagem natural, o maior problema é a transformação de uma frase potencialmente ambígua em uma não ambígua, a qual será utilizada pelo sistema.

2.3.7. Análise Semântica

A estrutura do sistema responsável por esta tarefa é o analisador semântico e consiste em analisar o sentido das palavras que foram agrupadas pelo analisador sintático.

A representação do significado das palavras apresenta diversas dificuldades. Pode-se mencionar a questão dos significados associados aos morfemas componentes de uma palavra (mercado, hipermercado), a questão da ambigüidade (tomar, em "tomar de alguém", em "tomar um banho" ou em "tomar suco"), ou a diferenciação entre significado e sentido ("casa", "minha casa").

A compreensão da relação entre as palavras é tão importante quanto a compreensão das próprias palavras. Enfoques formais para a semântica gramatical

tentam descrever o sentido de uma frase mediante a tradução de sua estrutura sintática para uma fórmula lógica-semântica. Como não existe uma correspondência imediata entre sintaxe e semântica, uma mesma estrutura sintática pode dar origem a diferentes representações semânticas.

As estruturas para as quais não seja possível um mapeamento podem ser rejeitadas, ou seja, se a frase não tiver algum sentido, será rejeitada. Por exemplo, na maioria dos universos, a frase "Verde a casa morre" seria rejeitada como semanticamente anômala.

2.3.8. Análise de Discurso

O significado de uma frase depende das frases que antecedem e pode influenciar os significados das frases que vêm depois dela. Por exemplo, a palavra "aquilo" na frase "Pedro não conseguiu aquilo" depende do contexto do discurso anterior, enquanto a palavra "Pedro" pode influenciar o significado de frases posteriores como "Ele sempre quis".

2.3.9. Análise Pragmática

É necessário fazer uma interpretação do todo e não mais analisar o significado de suas partes, do ponto de vista léxico e gramatical para determinar o que realmente se quis dizer. Por exemplo, a frase, "Você sabe que dia do mês é hoje?", deve ser interpretada como solicitação para que o dia do mês seja-lhe informado.

2.4. Problemas do PLN

Segundo (Savadovsky, 1988) um dos problemas mais difíceis de tratar computacionalmente será o da ambigüidade de linguagens naturais, ou seja, a existência de várias formas de entender uma mesma frase.

a) **Ambigüidade léxica:** ocorre quando uma palavra pode ser interpretada de diversas maneiras. Por exemplo, a sentença “João procurou um banco” a palavra “banco” pode se referir à procura de um banco financeiro ou de um lugar para se sentar. Normalmente é resolvida no contexto da frase;

b) **Ambigüidade sintática:** ocorre quando uma mesma sentença pode ser mapeada em mais de uma estrutura sintática válida. Por exemplo, “O menino viu o homem de binóculo”, esta frase possui dois tipos de interpretação, como se o menino estivesse de binóculo, ou também o homem estivesse de binóculo;

c) **Ambigüidade semântica:** ocorre quando temos mais de um significado para a mesma frase e acompanha a ambigüidade sintática, quando as diversas árvores sintáticas produzem análises semânticas válidas, como em: “Pedro viu Maria passeando” não se sabe se Pedro estava passeando e viu Maria, ou Pedro viu Maria passeando, na loja;

d) **Ambigüidade anafórica:** ocorre quando uma anáfora pronominal pode ser relacionada a duas ou mais palavras antecedentes e distintas. Por exemplo, “o ladrão entrou na casa do prefeito e tirou toda a sua roupa”, nesta frase a palavra “sua” pode estar relacionada com ladrão ou prefeito.

Além da ambigüidade outros tipos de problemas poderão aparecer durante o processamento da linguagem natural. O modelo a seguir exemplifica tal situação. As frases de uma língua são descrições incompletas das informações que pretendem transmitir. Por exemplo: há alguns cachorros lá fora. Há alguns cachorros no jardim. Há três cachorros no jardim. Nick, Dingo e Sheik estão no jardim. Nenhum programa envolvendo Linguagem Natural (LN) pode ser completo porque novas palavras, expressões e significados podem ser gerados com bastante liberdade, como: “Eu xeroco uma cópia para você”.

2.5. Conclusão

A questão PLN foi apresentada nesse capítulo ressaltando a importância e o esforço que a Inteligência Artificial apresenta no sentido de tornar os métodos de processamento dos computadores o mais próximo possível dos métodos do raciocínio humano.

Foi abordada uma estrutura básica inicial para o desenvolvimento de sistemas que utilizem o PLN para fazer a interação com o usuário, apesar de ser uma área que ainda não alcançou seus objetivos finais devido às limitações que as línguas oferecem ao processamento computacional.

Capítulo 3

Interface em Linguagem Natural

3.1. Introdução

Este capítulo vem apresentar uma explanação sobre Interface em Linguagem Natural (ILN) e Interface em Linguagem Natural para Banco de Dados (ILNBD).

3.2. Conceitos

Para que a comunicação entre duas ou mais entidades seja possível é necessário um veículo, ou um meio destas entidades compartilharem as mesmas informações. Este meio chama-se linguagem. "Uma linguagem é um conjunto de signos e símbolos que permitem um grupo social de comunicar e facilita o pensamento e as ações dos indivíduos" (Fischler, A., 1987).

Os seres humanos utilizam-se principalmente da linguagem natural para se comunicarem. Para (Savadovsky, 1988), a linguagem natural é uma das formas mais humanas de manifestação externa da atividade mental. Linguagem natural é a comunicação estruturada e inteligente entre pessoas.

Um dispositivo necessário para que haja a comunicação entre entidades é a interface. Uma interface é um dispositivo que serve de limite comum a várias entidades comunicantes, as quais se exprimem em uma linguagem específica a cada uma. Para que a comunicação seja possível, o dispositivo deve assegurar a conexão física entre as entidades e efetuar as operações de tradução entre os formalismos existentes em cada linguagem. Para (Thro, E., 1991), uma interface é um local para encontro ou interação.

A utilização de linguagem natural não garante que a interface seja natural. Isto é, fazer com que o usuário possa digitar seus comandos de acordo com seu vocabulário

coloquial facilita seu acesso ao computador, porém oferecer-lhe uma interface através da qual ele consiga dar entrada a esta mesma linguagem por voz ou escrita manual (ou ambos) seria mais próximo ao modo comum dele comunicar-se.

(Crane, H., 1993) aponta que a utilização de reconhecimento de voz e de escrita manual derrubarão as barreiras de teclados, mouses e Interfaces Gráficas para Usuários (GUIs), permitindo uma comunicação mais natural com o computador.

Existe um significante apelo na capacidade de fazer os computadores usarem a mesma linguagem que nós usamos no nosso dia-a-dia. Porém existem problemas na utilização de linguagens naturais nas interfaces: a ambigüidade das linguagens naturais. Uma solução para esse problema seria a restrição de sintaxe ou léxica embutida nas linguagens naturais utilizadas nas interfaces (Long, B., 1994).

A linguagem natural fornece instruções para o computador, de modo corrente, em qualquer língua.

Como é difícil reconhecer frases em linguagem natural, a interface é programada para reconhecer certas palavras, que são dicionarizadas e escolhidas em relação à utilização e finalidade do software.

A linguagem natural é uma das melhores técnicas para tornar o sistema mais intuitivo. Em vez de tentar lembrar os comandos o usuário entra com o que ele quer que aconteça. Entretanto existem dificuldades para se tratar da linguagem natural devido às ambigüidades da própria linguagem, mas uma possível solução é a utilização de menus com linguagem natural.

Quando sistemas utilizam restrições nas suas estruturas para limitar a ambigüidade, isso requer que o usuário aprenda quais estruturas são aceitáveis, tornando a linguagem natural nada mais que uma linguagem de comandos.

A linguagem natural é um estilo de interação muito difícil de ser implementado devido às várias possibilidades de linguagem natural utilizada, mas quando realizado com sucesso apresenta as seguintes vantagens: é fácil de usar porque a estrutura e o vocabulário são familiares ao usuário; a mesma linguagem pode ser utilizada para várias aplicações; deve existir pouco problema entre o intercâmbio de informações e aplicações; a linguagem natural é poderosa, pois permite várias formas de executar uma mesma ação, permite considerável flexibilidade em executar os passos de uma tarefa.

Para (Zwicker, R. & Reinhard, N., 1990) a utilização das interfaces em linguagem natural tem limitações sérias. O argumento de que elas facilitam o aprendizado pode ser enganoso, pois o usuário acaba tendo que aprender as limitações de cada uma delas. (Nickerson, R.S., 1986) questiona a comunicação em linguagem natural argumentando que não há maiores razões para assumir que o lado da máquina deva assemelhar-se ao lado humano da interface. O que é necessário são interfaces que maximizam a utilidade da máquina para o usuário, o que não é necessariamente obtido via linguagem natural.

Alguns aspectos da linguagem natural são: altamente desejável pela sua naturalidade, digitada pelo teclado, mas com redução de acentuação e pontuação podendo introduzir erros; é ambígua e vaga; aplicações usam um subconjunto restrito de uma linguagem natural existente; e usuários podem achar a interação inteligente.

Algumas desvantagens são: ambigüidade, imprecisa e com projeto de software complicado.

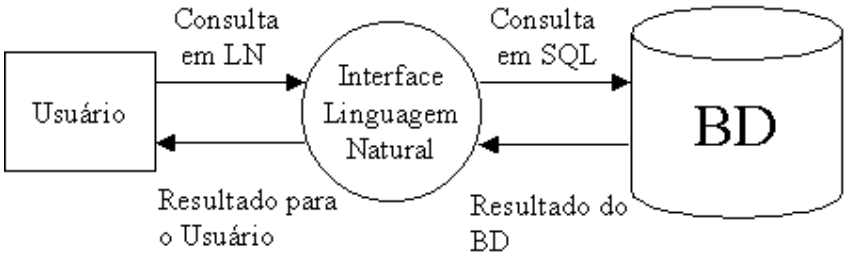
Para (Shneiderman, B., 1998) a linguagem natural apresenta a vantagem de aliviar a carga do usuário ter que aprender a sintaxe mas apresenta em contrapartida algumas desvantagens que são: requer diálogo claro; pode requerer muita digitação e pode não mostrar o contexto.

Segundo (Anick, Peter G., 1993) com o crescimento de tamanho e importância das bases de dados, os computadores necessitam ter meios de interpretar a linguagem natural. Dessa forma um usuário poderá encontrar facilmente algum argumento de pesquisa na base de dados, sem se preocupar com sua especificação exata - seja em termos de comandos de busca, seja em palavras a pesquisar - permitindo-lhe realizar suas pesquisas através do vocabulário que lhe é conhecido, sendo o computador responsável por oferecer-lhe sinônimos e ajuda direcionada ao argumento.

3.3. Interface em Linguagem Natural para Banco de Dados

Interface em linguagem natural para banco de dados (ILNBD) é um sistema que permite ao usuário obter informações armazenadas em banco de dados através do uso de comandos ou perguntas escritos em linguagem natural, como por exemplo, o português. Atualmente esses comandos são traduzidos em alguma linguagem formal de acesso a banco de dados, sendo SQL a mais utilizada, Figura 3.1. A área de acesso ao banco de dados foi o primeiro grande sucesso no desenvolvimento de aplicações que utilizam o processamento automático de linguagem natural (Russell & Norvig, 1998).

Figura 3.1 - Diagrama geral do Fluxo de uma ILN



3.4. Vantagens e Desvantagens das ILNBD

A maior motivação para a pesquisa e o desenvolvimento de ILNBDs como ferramentas de consulta são as vantagens, embora ainda não alcançadas por completo na prática, em relação às linguagens como SQL, interfaces baseadas em formulários e interfaces visuais. A principal vantagem é que usuários não necessitam aprender uma linguagem de comunicação artificial ou conhecer modelos lógicos dos sistemas gerenciadores de banco de dados a fim de elaborar suas consultas. A ILNBD ideal seria aquela que pudesse oferecer o uso de linguagem natural sem restrições, porém, no atual estado da arte, quando utilizamos a expressão "linguagem natural", estamos fazendo referência a dialetos que restringem a linguagem natural livre (Savadovsky, 1988).

Logo, ainda existe a necessidade de conhecer as funcionalidades e limitações de uma interface dessa natureza.

Outras características que tornam uma ILNBD a interface perfeita para usuários em geral são: a facilidade de formulação de perguntas que denotam negação ou quantificação e a utilização de expressões resumidas ou incompletas cujo significado é extraído do contexto do discurso (Reis et al.,1997). Por exemplo, existe uma facilidade em encontrar informações para questões como: "Qual departamento não tem programador?" ou "Quais são os departamentos da empresa?".

ILNBDs apresentam certas desvantagens em relação aos outros tipos de interfaces. Dentre elas, as mais apontadas são: o usuário não possui plena compreensão das limitações lingüísticas e semânticas impostas às suas consultas; quando uma pergunta é rejeitada, não é exposto com clareza se a mesma está fora do âmbito[3] lingüístico do sistema ou fora do modelo conceitual (Androutsopoulos et al., 1995); as interfaces que se apresentam como de propósito geral requerem longas etapas de configuração antes de serem utilizadas para uma aplicação particular (Copestake & Jones, 1989); alto custo de desenvolvimento em virtude da escassez de ferramentas profissionais robustas e integradas aos ambientes computacionais, e desenvolvedores qualificados; alto custo de evolução e extensão desses sistemas ao longo do seu ciclo de vida.

3.5. Arquitetura das ILNBD

Existem basicamente quatro modelos de arquiteturas para ILNBDs que são:

a) Sistemas baseados em Comparação de Padrões;
b) Sistemas baseados em Sintaxe;
c) Sistemas baseados em Gramática Semântica;
d) Sistemas baseados em Representação Intermediária.

a) Sistemas baseados em Comparação de Padrões

[3] Este ponto mostra a necessidade de pesquisas relacionadas a como o sistema em LN consegue entender o que o usuário deseja.

Esta arquitetura foi usada por alguns dos primeiros sistemas que faziam uso de uma interface em linguagem natural. Não utiliza analisadores sintáticos e nem gramáticas durante a interpretação de uma sentença e destina-se a aplicações com um conjunto pequeno de intenções. Sua implementação é fácil e o processamento de uma sentença é feito através do uso de um conjunto de padrões ou palavras chave. Mas a principal desvantagem está em sua simplicidade, o que não resulta em bons resultados ao usuário.

Como exemplo de interfaces que utilizam essa arquitetura podemos citar o programa ELIZA. Porém, muitos sistemas atuais utilizam variações do uso dessa técnica, tornando esta abordagem menos superficial e alcançado bons resultados.

O exemplo a seguir ilustra essa abordagem, na sua forma mais simples e dando ao tratamento de linguagem natural um caráter superficial, no desenvolvimento de interfaces para bancos de dados. Considere a seguinte relação como parte de um esquema de dados para uma Loja de Material de Construção:

Produto(Cod_Produto, Nome_Produto, Peso_Produto, Preço_Produto)

E que utilize o padrão abaixo para interpretar as perguntas de um usuário:
a) padrão:... "preço | quanto custa" .... "produto"... <Nome do Produto>
b) ação: Selecionar o preço do produto cujo Nome = <Nome do Produto>

Esse padrão especifica que se um usuário fizer uma pergunta que contenha a palavra "preço" ou a expressão "quanto custa" seguido da palavra "produto" mais o nome de um produto que exista na relação Produto, o sistema deverá localizar o preço do produto cujo nome foi mencionado na pergunta.

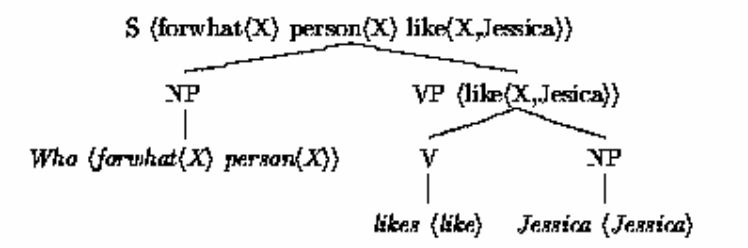
b) Sistemas Baseados em Sintaxe

Em sistemas baseados em sintaxe, o mapa entre a árvore gramatical e a representação semântica é um processo de somente dois passos. Primeiro uma árvore é

feita usando a gramática, e depois, um modulo de mapeamento associado a cada nó da árvore semântica a qual corresponde diretamente a consulta atual.

A Figura 3.2 mostra o método para a sentença “Quem gosta de Jéssica” (Who Likes Jéssica)?

Figura 3.2 - Árvore que representa a abordagem baseada em Sintaxe



Fonte: Généreux, M., 1999.

As representações semânticas são mostradas entre parênteses. O resultado da consulta associada com S é então usada para a consulta ao banco de dados. Devido ao fato das ILNBDs baseadas em sintaxe essencialmente usar a sintaxe para construir suas semânticas, existe usualmente linguagens e às vezes domínios específicos de acordo com (Androutsopoulos *et al*., 1995). A tendência é de existir dificuldade de aplicação prática, logo, questões como transportabilidade entre domínios e utilização de diferentes tipos de bancos de dados não podem ser contempladas através do uso dessa arquitetura.

c) Sistemas baseados em Gramática Semântica

A arquitetura abstrata dos sistemas baseados em gramática semântica é uma extensão da arquitetura dos sistemas baseados em sintaxe.

Basicamente, a diferença está na estrutura da gramática utilizada. Essa nova gramática, conhecida como gramática semântica, possui em sua estrutura elementos que não correspondem necessariamente a componentes sintáticos como verbos, nomes, sintagmas.

A vantagem da utilização desse tipo de gramática é que sua estrutura é concebida a partir de restrições semânticas. Dessa forma, o sistema pode assegurar que quando uma consulta for declarada válida pelo analisador sintático, uma sentença equivalente na linguagem de consulta do banco de dados poderá ser gerada. Pode-se

observar que a seguinte sentença: "Que rocha contém luminosidade?", embora sintaticamente correta em relação à gramática da língua portuguesa, não é aceita como válida, pois a gramática semântica impõe a restrição que o verbo conter deve ser seguido por nome que denomine uma substância e não uma radiação.

Essa abordagem foi adotada pelo sistema LADDER e seus sucessores, como o Q&A da Symantec (Copestake & Jones, 1989). Um outro tipo de interface e que também faz uso de gramáticas semânticas são os sistemas em linguagem natural baseado em menu.

Em sistemas baseados em menu, o usuário seleciona uma nova palavra ou sentença em um determinado menu, o sistema analisa a frase ou sentença utilizando a gramática semântica para determinar quais palavras ou sentenças podem seguir a sentença parcial a fim de formar uma consulta que possa ser entendida pelo sistema.

Sistemas baseados nessa arquitetura alcançam bons resultados quando o domínio da aplicação é relativamente limitado. No entanto, a reutilização da interface para outra aplicação exige a definição de uma nova gramática semântica.

d) Sistemas Baseados em Representação Intermediária

Ignorando alguns detalhes, a arquitetura da maioria das ILNBDs atual é semelhante a figura 3.3.

A sentença em linguagem natural informada pelo usuário passa pelo analisador da frase que envolve todas as etapas detalhadas no capítulo anterior. Após concluída esta etapa a sentença será traduzida em uma expressão intermediária. Segundo (Androutsopoulos et al., 1995) a expressão em linguagem intermediária, gerada no módulo aqui chamado de analisador intermediário, expressa formalmente o significado que o sistema atribui à pergunta em linguagem natural (Androutsopoulos et al., 1995).

A expressão intermediária é passada para um módulo chamado tradutor para linguagem de banco de dados que fará a conversão da mesma para uma linguagem que é suportada pela maioria dos Sistemas Gerenciadores de Banco de dados (SGBD). A expressão no idioma do banco de dados é então executada pelo SGBD que buscará atender ao pedido do usuário e mostrar-lhe o resultado do que foi solicitado.

Figura 3.3 - Arquitetura Geral das ILNBDs

Fonte: Androutsopoulos et al., 1995

A grande vantagem dessa abordagem é a modularidade empregada. Nessa arquitetura, podemos destacar dois módulos principais. O primeiro corresponde à parte lingüística e o segundo à interface com banco de dados. Essa característica favorece a transportabilidade da interface em diferentes níveis. Edite (Reis et al., 1997), um sistema que responde perguntas sobre informações turísticas, é um exemplo atual de ILNBD que utiliza essa arquitetura.

3.6. Problemas encontrados pelas ILNBDs

Embora o escopo das ILNBDs seja limitado pelas restrições de linguagem e de domínio de discurso, ainda faz-se necessário confrontar questões como transportabilidade entre domínios, identificação das limitações do sistema por parte dos usuários e problemas lingüísticos que normalmente ocorrem na maioria dos sistemas que trabalham na interpretação de linguagem natural. Dentre esses problemas, encontramos várias formas de ambigüidade introduzidas pela utilização de sentenças incompletas e mal formuladas (Androutsopoulos et al., 1995).

Sem contar com os problema lingüísticos abordados no capítulo anterior, lista-se os problemas encontrados no desenvolvimento de ILNBDs que aparecem como mais importantes para conseguir-se uma interface que seja aceita pelo usuário, como os que seguem:

a) Identificação de nomes próprios;
b) Transportabilidade;
c) Transparência para o usuário, (limitações e restrições ocultas).

a) Identificação de nomes próprios

Nomes próprios constituem um problema particular para as ILNBDs. Se um banco de dados em uma grande empresa possuir informações sobre milhares de funcionários o sistema precisa ter entradas em seu léxico de todos esses nomes. Inserir manualmente no léxico cada nome de empregado é uma tarefa tediosa, e isto significa também que o léxico teria que ser atualizado manualmente sempre que novos empregados entrarem na empresa. Uma possível solução é prover algum mecanismo que computasse automaticamente entradas no léxico de nomes formais que não aparecem no banco de dados. Nesse caso, a arquitetura de pré-processamento da figura 3.3 tem que ser modificada para permitir que este tenha acesso ao banco de dados. Essa abordagem tem a desvantagem que introduz um processamento adicional no banco de dados durante o pré-processamento. Em grandes bancos de dados esta procura pode se tornar cara.

Este problema também aparece em questões que não contém nomes próprios inseridos no banco de dados. Se, por exemplo, o banco de dados não contém o nome "João" a ILNBD deveria falhar ao fazer a análise gramatical e no seu lugar emitir uma mensagem de negação. Uma alternativa é empregar o pré-processamento baseado em comparação de padrões, porém, a procura por informação no banco de dados torna-se uma tarefa necessária.

b) Transportabilidade

Significa que a ILNBDs deverá ter a capacidade de ser utilizada por diversos banco de dados independente do domínio que possuem, sem a necessidade de reescrever

todo o módulo de processamento de linguagem natural ou modificar o mapeamento para novas estruturas de dados e criar um novo léxico.

c) Transparência para o usuário, (limitações e restrições ocultas)

O objetivo do uso de uma interface em linguagem natural é subtrair dos usuários a necessidade de aprender uma linguagem e vocabulário especializado ou conhecer a estrutura interna de um banco de dados. Porém, sistemas atuais trabalham com subconjuntos de uma linguagem natural, criando, dessa forma, limitações e restrições que devem ser compreendidas pelos usuários a fim de que a interação com sistema alcance o sucesso esperado. O termo "adaptability"[4] refere-se à medida da eficiência com a qual o usuário pode reconhecer e adaptar-se às limitações do sistema (Sethi, 1989).

Existem duas abordagens para minimizar o impacto de restrições desconhecidas pelo usuário. A primeira é tornar a interface dependente de um determinado domínio. Embora tal procedimento possa diminuir o número de perguntas que não podem ser processadas pelo sistema, torna-se imprescindível à existência de especialistas para extrair conhecimento de um domínio e configurar a interface. Além disso, tal solução vai de encontro à tentativa de reutilizar uma interface em diferentes aplicações. A outra abordagem é definir explicitamente um subconjunto restrito de uma linguagem natural a fim de expor naturalmente aos usuários as limitações e restrições do sistema. A interface PRE (Epstein, S. S., 1985) e sistemas baseados em menu adotaram essa postura e alcançaram sucesso em relação à adaptabilidade.

3.7. Conclusão

Este capítulo apresentou uma introdução a área de pesquisa sobre a interface em linguagem natural para Banco de dados com o objetivo de apresentar algumas das dificuldades existentes na implementação da mesma.

[4] O termo em inglês "adaptability" será referenciado no texto pela palavra em português "adaptabilidade"

As ILNBDs deverão ter a capacidade de ser utilizadas por diversos banco de dados independente do domínio que possuem, sem a necessidade de reescrever todo o módulo de processamento de linguagem natural ou modificar o mapeamento para novas estruturas de dados e criar um novo léxico.

Embora muitas ILNBDs desenvolvidas a partir dos anos 80 tenham demonstrado características importantes em relação ao processamento linguistico, elas não ganharam a aceitação comercial tão esperado, e ainda são tratadas como sistemas exóticos ou alvo de pesquisas, provalvelmente pela dificuldade na implementação dos problemas apresentados no item 3.6 (Problemas encontrados pelas ILNBDs).

A seguir serão apresentados alguns softwares que foram desenvolvidos sob esta concepção.

Sistemas em Linguagem Natural

4.1. Introdução

O objeto deste capítulo é apresentar uma relação de alguns software e trabalhos que foram desenvolvidos até o presente momento para que se tenha uma noção sobre a complexidade que existe, e para ampliar-se o conhecimento sobre a necessidade que ainda existe no que foi desenvolvido.

4.2. Os primeiros programas

4.2.1. ELIZA

Desenvolvido por (Weizenbaum, 1966) ELIZA, figura 4.1, foi o primeiro programa de computador a imitar a conversação humana, modelando o comportamento de um Terapeuta Rogeriano.

Os princípios usados pelo ELIZA são muito fáceis de descrever. Inicialmente, varre a entrada em busca de palavra-chave, que, quando detectadas, disparam sempre a mesma ação, devolvendo uma mensagem padrão, ou usando parte da entrada para construir uma mensagem.

Figura 4.1 - Interface do Sistema ELIZA traduzida

```
Como vai você. Por favor, relate seu problema.
>> Eu gasto muito tempo com computadores.
Por que você mencionou computadores?
>> Você é um.
Por favor, você acredita que eu sou um?
>> Sim.
Você parece estar certo.
>> Suponha que eu seja.
A quanto tempo você foi?
```

Fonte: Weizenbaum, 1966.

O programa também é baseado em uma estrutura gramatical simples e composto de regras que representam uma situação, como listado na figura 4.1, o que traz a ilusão que o programa possui alguma inteligência.

Qualquer sentença que o usuário digitasse seria varrida em busca de certas palavras ou frases tais como "meu" é transformado em "seu" ou "você é" em "eu sou". A finalidade dessas transformações simples é devolver ao usuário as sentenças que tenha introduzido, como se essas fossem geradas pelo programa.

4.2.2. LUNAR

Durante o ano de 1972 e 1973 foi desenvolvido o LUNAR (the Lunar Science Natural Language Information System), um protótipo construído por William Woods e sua equipe na NASA (National Aeronautics and Space Administration). Foi um dos trabalhos pioneiros para sistemas com característica de responder perguntas referentes a uma base de dados estruturada, com informações de análises químicas das rochas lunares e amostras do solo lunar coletadas por umas das expedições da missão Apollo a partir de uma representação de cálculo de predicados.

O sistema foi capaz de entender perguntas em inglês com quantificadores e anáforas. Esse sistema foi implementado na linguagem LISP e seu funcionamento original tem sido a base para a elaboração de técnicas para reconhecer estruturas morfológicas em domínios em que existem palavras desconhecidas e que tenham sido aplicadas em sistemas de recuperação como no projeto descrito pela Sun Microsystems[5].

Usa uma gramática ATN, motivada pela teoria de gramáticas transformacionais (Rabuske, R. A., 1995 apud Winograd, 1983).

O sistema não foi utilizado em operações reais apesar de conseguir responder a 78% das perguntas feitas a ele através de testes ao sistema (Russel & Norvig, 1998).

---

[5] Natural Language Technology in Precision Content Retrieval. Sun Microsystems Research. Techinal Reports. http://research.sun.com/techrep/1998/abstract-69.html

4.2.3. PLANES

Por volta de 1977, foi desenvolvido o PLANES, um sistema caracterizado por ter um parser baseado em uma Rede de Transição Aumentada ATN (Augmented Transition Networks). Para estender a gramática teria que agregar novas formas nas diferentes variações das orações (Wallace, M., 1984).

4.2.4. CHAT 80

Foi um sistema de ILNBD do início dos anos 80. Implementado em PROLOG, efetuava consultas a uma base de dados no próprio PROLOG (Androutsopoulos et al., 1995). O sistema de Fernando Pereira de 1983 gerava respostas às perguntas referentes a uma base de dados geográfica em inglês (Russel & Norvig, 1998).

O CHAT 80, figura 4.2, tinha como um dos seus principais objetivos tratar o problema da transportabilidade de interfaces entre domínios, tema muito explorado na década de 80 pelas pesquisas e protótipos na área de interfaces em linguagem natural para banco de dados.

Figura 4.2 - Interface do Sistema CHAT 80 traduzida

```
Q: Quais os países que possuem divisas com dois mares?
A: Egito, Irã, Israel, Arábia Saudita e Turquia.
Q: Quais são os países que desembocam no mar negro?
A: Romenia, União Soviética
Q: Qual é a área total dos países ao sul do equador sem a Austrália?
A: 10.228.000 milhas quadradas
Q: Qual é o oceano que faz divisa com os países da África e da Ásia?
A: Oceano Indico
```

Fonte: Russel & Norvig, 1998

4.2.5. JANUS

Em 1988, a BBN[6] e a ISI[7] desenvolveram o JANUS, figura 4.3, um sistema de interpretação e geração da linguagem natural em inglês que recuperava informação referente aos navios da frota do Pacífico da força naval dos Estados Unidos (Hinrichs,

[6] BBN significa (Bolt, Beranek and Newman) nome dos fundadores da empresa em 1948, maiores informações em http://www.bbn.com

[7] ISI significa (Information Sciences Institute) escola de engenharia da Califórnia, maiores detalhes em http://www.isi.edu

1988). A base de conhecimento continha, entre outras coisas, informações sobre os horários, localização e condições da disposição das frotas no pacífico.

Figura 4.3 – Tradução da Interface do sistema JANUS

```
a. O almirante desdobrou o navio?
b. Liste todos os navios que saem em 4/1/86
c. Quais navios C3 são agora C4?
d. Quando Vicent chegará no Havaí?
e. Quem era o comandante precedente a Frederick?
f. Todos os almirantes deverão comandar um navio hoje.
```

Fonte: Hinrichs, 1988.

Os exemplos de consultas efetuadas na figura 4.3 demonstram que muitas das informações são altamente dependente do tempo: navios alteram sua localização de acordo com seus percursos. Muito da informação que o usuário do JANUS estaria buscando seria dependente do tempo e uma reposta apropriada poderia ser dada se o tempo da questão e o tempo dos eventos em questão fossem levados em conta. Isto conseqüentemente gera a necessidade de uma representação semântica adequada para a análise da entrada em linguagem natural.

A teoria semântica adotada pelo JANUS é a arquitetura gramatical de Montague. A partir das considerações metodológicas, os três componentes (sintaxe, tradução, e modelo de interpretação teórica) que são de grande importância na teoria de Montague formam uma parte integrada aos componentes do sistema de interface em linguagem natural JANUS, figura 4.4.

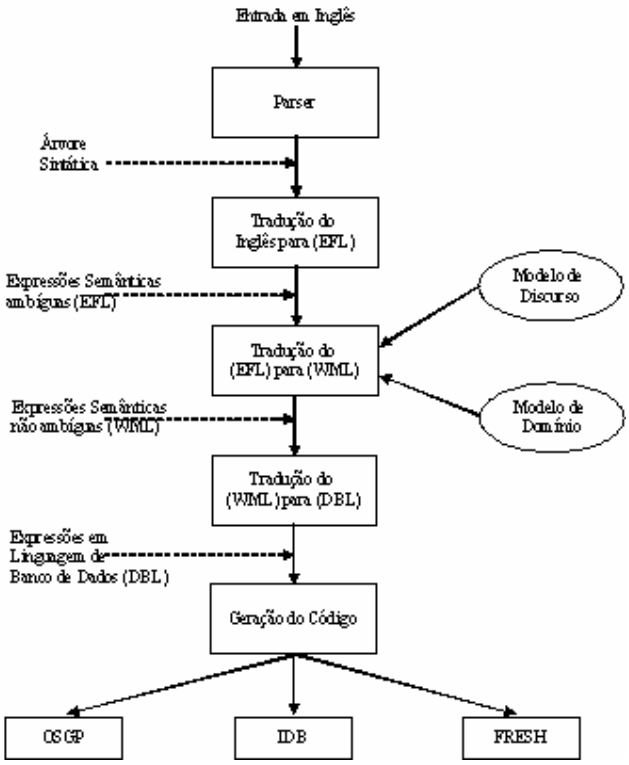
O inglês introduzido é alimentado no parser sintático do JANUS que produz uma árvore gramatical que é atribuído ao componente de tradução. O componente de tradução gera uma expressão lógica da entrada em inglês. Essa expressão lógica é avaliada em relação ao banco de dados, qual pode levar para ser uma procuração para um modelo parcial do real mundo, no caso de JANUS um modelo dos objetos e estados de negócios que pertencem a Frota do Pacífico.

No primeiro nível de tradução, todo artigo léxico em Inglês é traduzido em alguma expressão lógica de um Idioma formal Inglês-orientado chamado Expressões Semânticas Ambíguas (EFLs). No nível EFLs, nenhuma atenção é prestada ao contexto que pode contribuir para a interpretação do inglês introduzido.

Nenhuma tentativa é feita ao nível EFL para os significados léxicos não ambíguos. A desambiguação léxica é provida por um modelo de domínio que, entre

outras coisas, captura toda a informação sobre significados léxicos conhecidos ao sistema.

Figura 4.4 - Arquitetura JANUS



Fonte: Hinrichs, 1988.

O conhecimento de domínio mantém o sistema informado desses aspectos da pergunta-resposta de uma sessão de usuário que é pertinente para determinar a referência de questões dependentes do contexto como:

a) Pergunta: Onde os navios estavam ontem?

b) Resposta: No Oceano Índico.

c) Pergunta: Quanto tempo eles estiveram lá?

O modelo de discurso contém parâmetros sobre o tempo que a sessão de pergunta-resposta está levando. Nesse parâmetro é necessário calcular o referente aos advérbios de tempo como "ontem" na frase "a", além de manter a localização do modelo referente a frases nominais para descobrir que "eles" refere-se aos navios.

A entrada do modelo do domínio e do modelo do discurso conduz à tradução a uma língua do mundo-modelo, ou seja, expressão semântica não ambígua (WML), que seja traduzida por sua vez em expressões da língua de pergunta da base de

conhecimento apropriada: uma base de dados DB, um sistema especialista (FRESH), ou uma base de conhecimento OSGP da exposição.

4.2.6. ASK

O sistema ASK permitia que o usuário entrasse no sistema com novas palavras durante sua interação. Além de ser um sistema com ILNBD, também era um sistema administrador que se comunicava à base de dados, programas de correio eletrônico e outras aplicações de forma transparente para o usuário (Androutsopoulos et al., 1995).

4.2.7. Q&A

Q&A chegou a ser um sistema de ILNBD comercial baseado em uma gramática semântica que transformava a pergunta em uma ordem na base de dados e mostrava a informação na forma de tabela (Wallace, M., 1984).

Embora tenha apresentado um grande volume de vendas, seu sucesso não foi conseqüência direta da interface em linguagem natural, mas sim de suas características e facilidades como ferramenta de acesso a dados, que não tinham relação com linguagem natural, frente à concorrência (Copestake & Jones, 1989).

4.2.8. INTELLECT

Outro sistema ILNBD comercial, que utilizava uma gramática independente da aplicação, porém também tinha um dicionário que atribuía restrições a sua gramática. Tinha problemas para manusear com verbos dependentes do domínio como "voar". Não distinguia substantivos de verbos, só obedecia a restrições da aplicação, motivo pelo qual o usuário necessitava um pouco de conhecimento para fazer consultas, pois o sistema aceitava a inclusão de orações mal formuladas (Wallace, M., 1984).

Apesar de ter apresentado um sucesso razoável em relação ao processamento de linguagem natural, possuía um alto custo de aquisição e necessitava de um grande esforço por parte dos administradores para ser configurado.

4.3. Trabalhos Científicos

A seguir serão relatados alguns softwares que foram desenvolvidos a partir de trabalhos de pesquisas científicas.

4.3.1. Sistema Edite

(Filipe, P.P, 1999) diz que o sistema Edite permite formular consultas à base de dados de recursos turísticos dos quiosques multimídias do Instituto de Engenharia de Sistemas e Computadores (INESC) em Portugal.

O sistema Edite permite o usuário fazer consultas usando para isto a língua natural, apresentando as seguintes vantagens:

a) as perguntas a que, por alguma razão, o sistema não consegue responder podem ser guardadas e a causa do insucesso localizada e reparada. Deste modo o sistema está sempre em crescimento tornando-se cada vez mais robusto e competente;
b) são possíveis perguntas que envolvem quantificação – "Indique-me 2 locais com pesca no rio." - qualificação – "Indique-me um hotel perto." - ou negação – "Onde existem hotéis que não aceitam cartão Visa?";
c) são suportadas expressões permitindo ao utilizador exprimir-se de uma forma muito rápida – "O hotel Berna tem parque de estacionamento? E piscina?".

A arquitetura do sistema é baseada numa seqüência de processos, (análise morfológica, análise sintática, análise semântica e tradução) a que a pergunta em língua natural é submetida, até se obter a tradução para SQL. Se a análise inicial da frase encontrar uma interpretação válida para a pergunta é gerada uma interrogação na linguagem de representação intermédia, designada por linguagem de interrogação lógica (LIL) (Reis et al., 1997).

Finalmente é feita a tradução da interrogação em LIL para SQL gerando como resultado uma instrução SQL SELECT. A instrução SELECT resultante do processo de

tradução é passada ao sistema de gestão de base de dados que devolve os dados que constituem a resposta.

A grande desvantagem observada neste trabalho é que o sistema fica preso a um único domínio, ou seja, a recursos turísticos da região.

4.3.2. Masque/SQL

Em (Androutsoupoulos, I. 1992) MASQUE (Modular Answering System for Queries in English) significa módulo de sistemas de respostas para consultas em inglês. Também uma interface que faz ligação com banco de dados voltados a determinados domínios. Este sistema foi desenvolvido em Edinburgh na Alemanha e é descendente do CHAT-80.

Também possui uma linguagem de interpretação intermediária conhecida por Linguagem de Consulta do Masque (LQL) (Query Language of Masque).

Sua principal desvantagem é com relação a sua configuração inicial, ou seja, para que o usuário comece a trabalhar com o programa é necessária a realização de diversas etapas que tornam ele bastante complexo de ser utilizados em situações práticas.

4.4. Sistemas Comerciais

A seguir serão discutidas algumas das atuais interfaces em linguagem natural para acesso a banco de dados. A existência de interfaces comerciais dessa natureza como: Elf da Elf Software, EnglishQuery da Microsoft e Q&A da Symantec, subtraem a necessidade de aprendizado de uma linguagem de consulta formal, porém tais interfaces são desenvolvidas para esquemas de banco de dados em geral, o que as tornam ferramentas com expressiva necessidade de configuração antes de serem usadas. Para cada novo esquema a ser utilizado, todo um conjunto de entidades e relacionamentos lingüísticos precisam ser definidos.

Nenhum dos softwares acima é gratuito e a maior parte deles somente se encontra em sistemas operacionais Windows. Preço: licenças de $100,000 a $500,000 ou porcentagem de vendas melhoradas.

4.4.1. English Query

Este software foi projetado e implementado pela Microsoft como uma parte do software SQL Server. Consiste em dois componentes: um para converter consultas em Inglês para SQL e uma ferramenta para a configuração inicial do banco de dados.

O primeiro passo para construir uma aplicação no Microsoft English Query é criar o modelo semântico para o domínio do problema, ou seja, é necessário especificar como as entidades (substantivos) e os relacionamentos (verbos, adjetivos) mapeiam para entidade, atributos e relacionamentos no banco de dados criado. Se alguma entidade estiver relacionada a outra, então deve-se indicar o relacionamento entre elas, normalmente feito através da chave estrangeira[8]. Outro ponto a observar é a necessidade da presença da chave primária em todas as tabelas que servirão como possíveis consultas.

Ao se definir uma entidade atribui-se sinônimos para as tabelas e campos que estão associados a ela. Na figura 4.5 um formulário de interface com o usuário que está definindo palavras que identificam a entidade "Person" no exemplo foi atribuído às palavras "author" e "writer".

As entidades principais têm associadas a elas dois tipos de entidades menores com nomes e características. Os nomes indicam como a entidade é identificada em perguntas e declarações. Desta maneira cria-se nome de entidades para entidades principais que são representadas por todas as tabelas que o usuário tem alguma forma de identificação em perguntas. Esse procedimento é repetido para todos os atributos principais da tabela.

Depois de atribuídos as características para as entidades, pode-se efetuar perguntas como, por exemplo:

---

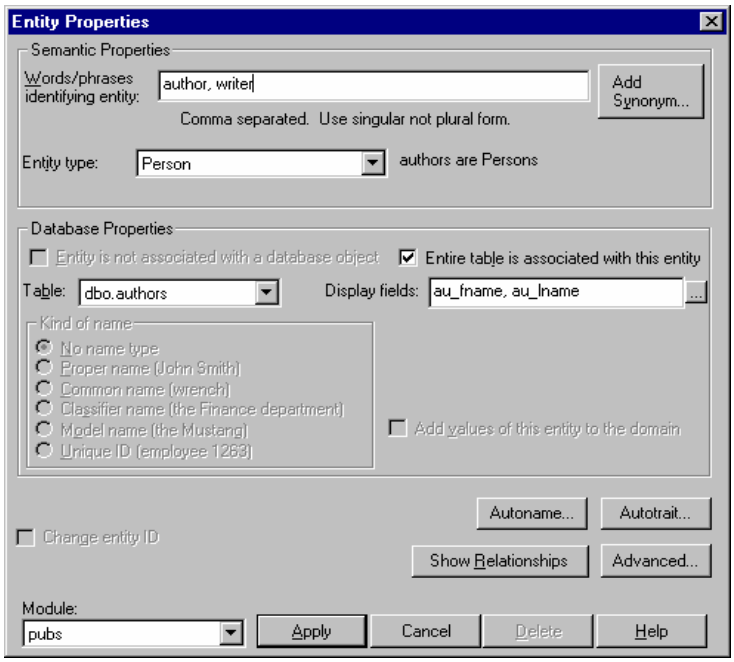
[8] Entidade é um conjunto de dados de um mesmo tipo como, por exemplo, Aluno. A entidade Aluno será formada por atributos (nome e código do aluno), código é uma chave primária por que é um atributo cujas informações não poderão ser duplicadas, a chave estrangeira é um atributo de uma entidade que é chave primária de outra entidade, no caso da entidade aluno poderíamos ter um atributo (código da cidade) que poderia ser considerado uma chave estrangeira.

“Que autores têm cidade Seattle?”,

“Mostrar os autores e suas cidades”,

“Quais autores são do país França”.

Figura 4.5 - Formulário com propriedades de uma Entidade



Fonte: Software English Query da Microsoft.

Mas para perguntas que sejam realmente interessantes, é necessário criar os relacionamentos entre as entidades como, por exemplo, “autores escrevem livros” e “editoras publicam livros”.

Após esse procedimento serão criadas frases para a relação. Entre os tipos de frases há:

Frases com verbo: "AUTORES ESCREVEM LIVROS",

Frases com preposição: "EDITORES ESTÃO NA CIDADE",

Frases adjetivas: "LIVROS SÃO POPULARES",

Frases de subconjunto: "ALGUNS LIVROS SÃO BEST-SELLER",

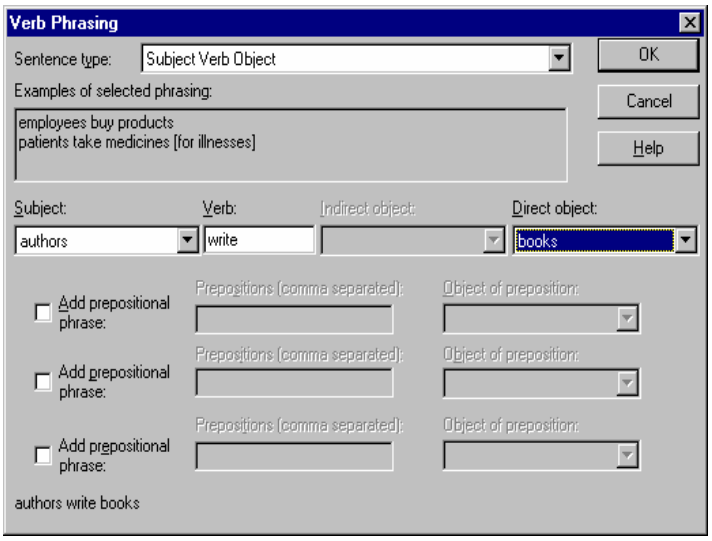
Frases com características: "LIVROS TÊM DIREITOS AUTORAIS".

A figura 4.6 mostra uma frase verbal definida na caixa de dialogo para "AUTOR ESCREVE LIVROS".

O administrador de banco de dados não necessita nenhum conhecimento de SQL para completar essas tarefas.

Uma vez que o banco de dados esteja configurado, o English Query pode traduzir questões na língua inglesa muito complexas para SQL com a capacidade de encontrar múltiplas tabelas e múltiplos campos. O exemplo abaixo demonstra uma declaração SQL complexa traduzida de uma questão em inglês.

Figura 4.6 - Formulário de criação de Frase Verbal



Fonte: Software English Query da Microsoft.

"What hotels in Hawaii have scuba?"

```
SELECT    dbo.HotelProperty.HotelName as "Hotel Name",
          dbo.HotelProperty.USReservationPhone as "Phone",
          dbo.HotelProperty.StreetAddress1 as "Street Address",
          dbo.HotelProperty.CityName as "City",
          dbo.HotelProperty.StateRegionName as "State or Region"
FROM      dbo.HotelProperty, AmenityNames, Amenities
WHERE     dbo.HotelProperty.StateRegionName='Hawaii'
AND       AmenityNames.Amenity='Scuba'
AND       AmenityNames.AmenityID=Amenities.AmenityID
AND       dbo.HotelProperty.HotelID=Amenities.Hotelid
```

O programa run-time pode ser instalado através de ferramentas como COM (Common Object Model) suportando ambientes como Visual C++, Visual Basic, Active

Server Pages (ASP). Isso habilita as questões inglesas a serem consultadas em websites criados com a tecnologia ASP.

O English Query somente está disponível para plataformas Windows com 32 bits e computadores que tenham a ferramenta de acesso ao banco de dados OLE DB como Oracle e Microsoft Access.

No SQL Server 2000, a criação, ou autoria, e a implantação de aplicações English Query foram amplamente simplificadas.

São proporcionados assistentes para automatizar o processo de criação dos modelos semânticos necessários para o English Query.

O novo Formato de Modelagem Semântica (SMF - Semantic Modeling Format) baseado pode ser usado como informação de modelo English Query. Usado com o Authoring Object Model, o SMF proporciona autoria via programação de modelos English Query.

A utilização do sistema está restrita ao sistema de desenvolvimento Microsoft Visual Studio, e está incluída no SQL Server 2000.

4.4.2. EASYASK

Desenvolvido por EasyAsk Incorporation, este é um pacote de aplicação para interface em linguagem natural para sites de e-commerce. Inicialmente desenvolvido pelo Dr. Larry Harris em 1995, foi chamado então de English Wizard.

Os usuários podem introduzir suas questões como palavras-chaves, frases ou perguntas em inglês completas. Esta pergunta do usuário é então pré-processada antes da tradução para o SQL.

O EasyAsk tem seu próprio dicionário e enciclopédias que ajudam corrigir erros de ortografia na entrada do usuário através da atribuição de sinônimos. As palavras ambíguas são corrigidas através de perguntas ao usuário sobre seu significado.

EasyAsk também gera códigos SQL complexo. Ele pode gerar sub-consultas, utilizar filtragens e agrupamentos. Ele também pode encontrar valores exatos ou outras condições especificas do SQL como "LIKE", "EXIST", "NOT EXISTS" e a cláusula "NULLS". Ele também pode reconhecer junção entre tabelas verificando a estrutura do

banco de dados. Alguns outros comandos SQL podem ser gerados como sub-consultas, datas, horas, perguntas de sim ou não, e traçar abreviaturas para seus valores formais.

Uma vez que a declaração do SQL é gerada e executada, a saída dos resultados do banco de dados poder ser apresentado ao usuário de muitas formas. Alguns destes formatos são apresentados em planilhas, tabelas de referência cruzadas, e gráficos. A saída desejada pode ser selecionada de uma lista ou incluída pela entrada do usuário.

Um exemplo de entrada do usuário poderia ser como "Pie chart of sales by region" (gráfico de setores para as vendas de uma região) o qual representa uma saída com dos dados no formato de gráfico de setores.

4.5. Conclusão

A partir do estudo realizado neste capítulo será proposto e implementado uma interface em linguagem natural que procura aproveitar o que cada um dos softwares estudados tem a oferecer, a proposta como os resultados alcançados são relatados no próximo capítulo.

Capítulo 5

Proposta Desenvolvida

5.1. Introdução

Este capítulo tem por objeto apresentar uma proposta de interface em linguagem natural para banco de dados. Serão descritas as ferramentas utilizadas, ligações entre elas, estratégias para resolução dos problemas enfrentados durante o desenvolvimento, algoritmos e lógicas criadas para a solução dos problemas.

5.2. Tecnologias Utilizadas

Dentre as tecnologias utilizadas para implementação do modelo proposto destaca-se: Linguagem de Modelagem Unificada (UML), Java Server Pages (JSP), Java Beans, PROLOG. A seguir uma introdução do que são cada umas delas e também como foi feita a ligação entre elas.

5.2.1. Java Server Pages (JSP)

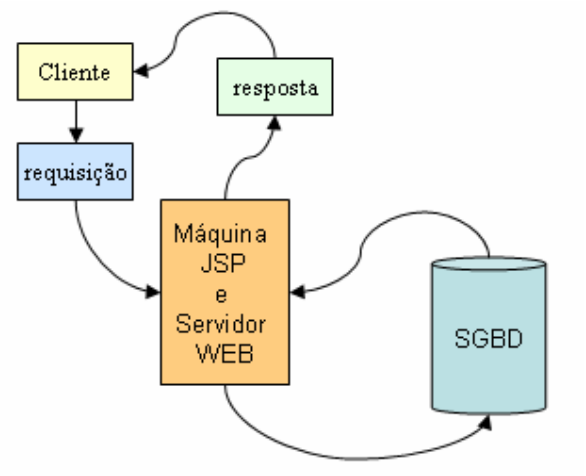
Conforme (Bomfim Júnior, F.T., 2002) o desenvolvimento de aplicações para web depende de alguma tecnologia capaz de gerar conteúdos dinamicamente. Aplicações baseadas em Java utilizam a tecnologia Java Server Pages (JSP) para este fim.

A tecnologia JSP é semelhante ao Microsoft Active Server Pages (ASP), porém tem a vantagem da portabilidade de plataforma; além disso permite produzir aplicações com acesso ao banco de dados, arquivos-texto, captação de informações a partir de formulários, visitantes e sobre o servidor, uso de variáveis e loops. Permite separar a

programação lógica (parte dinâmica) da programação visual (parte estática), facilitando o desenvolvimento de aplicações mais robustas, qm que o programador e designer podem trabalhar no mesmo projeto, mas de forma independente. Outra característica do JSP é produzir conteúdos dinâmicos com a possibilidade de reutilização.

Quando uma página JSP é requisitada pelo cliente através de um browser, esta página é executada pelo servidor, e a partir daí será gerada uma página HTML que será enviada de volta ao browser do cliente. A figura 5.1 ilustra esse funcionamento:

Figura 5.1 - Funcionamento do JSP



Fonte: Adaptado de (Komosinski, Leandro J., 2002).

Quando o cliente faz a solicitação de um arquivo JSP, é enviada uma requisição para a máquina JSP que envia a solicitação de qualquer componente (podendo ser um JavaBeans, servlet[9] ou enterprise Bean[10]) especificado no arquivo. O componente controla a requisição possibilitando a recuperação de arquivos em banco de dados ou outro dado armazenado; em seguida, passa a resposta de volta para a máquina JSP, que junto com o servidor WEB enviam a página JSP revisada para o cliente, e o usuário pode visualizar os resultados através do WEB browser. O protocolo de comunicação usado entre o cliente e o servidor pode ser HTTP ou outro protocolo.

JavaBeans são componentes de software que são projetados para serem unidades reutilizáveis sem serem modificados. Um modelo de componente é definido como um

[9] Para (Bomfim Júnior, F.T., 2002) Servlets são componentes, pequenos programas na linguagem Java, que geram as páginas visualizadas pelo navegador.
[10] Para (Bomfim Júnior, F.T., 2002) Enterprise JavaBeans possuem todas as características vistas em um JavaBean mas com componentes indicados para aplicações distribuídas.

conjunto de classes e interfaces na forma de pacotes Java que deve ser usado em uma forma particular para isolar e encapsular um conjunto de funcionalidades. Os componentes JavaBeans são também conhecidos como Beans.

5.2.2. Programação em Lógica PROLOG

Segundo (Clocksin, W.F. & Mellish, C.S., 1981) Prolog é uma linguagem de programção para computador que é usada para resolver problemas que envolvem objetos e o relacionamento entre objetos.

Para (Palazzo, L.A.M., 1997) a principal utilização da linguagem Prolog reside no domínio da programação simbólica, não-numérica. O advento da linguagem Prolog reforçou a tese de que a lógica é um formalismo conveniente para representar e processar o conhecimento. Seu uso evita que o programador descreva os procedimentos necessários para a solução de um problema, permitindo que ele expresse declarativamente apenas a sua estrutura lógica, através de fatos, regras e consultas.

A seguir serão esclarecidos os conceitos de fatos, regras e consultas conforme (Clocksin, W.F. & Mellish, C.S., 1981):

Fatos: consistem representar declaração de objetos, por exemplo na frase: "João gosta de Maria" é um fato e poderia ser representado através do predicado gosta_de(joao,maria).

Regras: são usadas quando se quer dizer que um fato depende de um grupo ou de outros fatos. Por exemplo, para expressarmos a seguinte regra: "João gosta de alguém que gosta de vinho", poderia ser representada como sendo: gosta_de(joao,X) :- gosta_de(X,vinho).

Consulta: uma vez criado um programa com fatos e regras precisamos obter informações sobre eles, ou seja, efetuarmos uma consulta. Por exemplo: desejamos saber quem gosta de "João" a consulta seria: ? – gosta_de(joao,X).

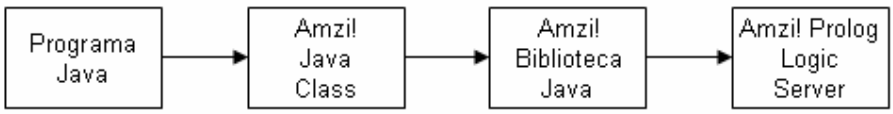
Os fatos e regras também poderão ser chamados de predicados em prolog.

5.2.3. Ligação entre Java e PROLOG

A linguagem de programação Amzi Prolog possui uma interface simples para geração de aplicações de console. Porém, aplicativos profissionais e de uso geral (como é o caso de Sistemas Especialistas, Interfaces para Linguagem Natural) necessitam apresentar uma interface mais amigável para o usuário.

Baseado nisso, foi desenvolvida uma biblioteca de funções na qual está incorporada à máquina de inferência da linguagem Prolog, chamada de Logic Server.

Figura 5.2 - Ligação entre Java e PROLOG



Fonte: Adaptado de (www.amzi.com).

O Amzi Logic Server é escrito em C/C++ e possui uma biblioteca dinâmica (DLL sob o Windows, compartilhando biblioteca sob Solaris, Unix do SDI e Linux) que pode ser chamada por qualquer aplicação que possa chamar uma biblioteca dinâmica. O Java Class é uma interface no Prolog Logic Server, projetado para uso do Java.

Para construir uma classe em Java cujos métodos não são implementados em Java, há a necessidades de construir uma nova biblioteca que ofereça suporte às chamadas pelo programa Java. Esse foi o primeiro passo ao implementar o Amzi Java Class. Os elos de um programa Java com a biblioteca básica Amzi são mostradas na figura 5.2.

Amzi Java é a nova biblioteca que conecta o Amzi Java Class ao Amzi Logic Server. Além desta arquitetura apresentada, há um número de recursos do Java Class que foi introduzida de acordo com as características distintas de Java.

O Amzi Logic Server foi encapsulado para duas linguagens de programação orientada a objeto, C++ e Delphi. Essas classes permitem aos desenvolvedores derivar suas próprias aplicações a classes específicas que encapsulam serviços do Prolog. A implementação do Java segue o mesmo padrão dessas classes.

A classe do Logic Server inclui todos os métodos que oferecem ao desenvolvedor controle total acima da engenharia do Prolog. Esses incluem métodos para:

a) setup, iniciar, reset e fechar a ferramenta Prolog;

b) abrir e/ou consultar programas do Prolog;
c) emitir perguntas ao Prolog;
d) aceitar ou não aceita termos do Prolog;
e) conversão entre string do Java e termos do Prolog;
f) obter valores das variáveis Java de átomos do Prolog, strings e números;
g) construir e decompor listas do Prolog;
h) construir e decompor estruturas do Prolog;
i) retornar informação sobre erros.

Para que ocorra a comunicação entre o Java e o Prolog há a necessidade de que o arquivo criado na interface do prolog passe pelo processo de compilação, recurso esse oferecido pelo prolog. A compilação vai gerar um novo arquivo com extensão "*.plm" esse arquivo deverá passar por outro processo que é o link, processo que irá gerar um novo arquivo com extensão "*.xpl" que é o responsável por conter os dados criados no programa em prolog e que a linguagem Java no caso conseguirá buscar as informações.

5.2.4. Linguagem de Modelagem Unificada (UML)

A Linguagem de Modelagem Unificada (UML), segundo (Silva,D.M.,2001), é uma linguagem para especificação, documentação, visualização e construção de sistemas orientados a objetos, sendo considerada uma das linguagens para modelagem de sistemas orientados a objetos, mais expressivas de todos os tempos. Através dos diagramas oferecidos por essa linguagem, é possível representar sistemas de softwares sob diversas perspectivas de visualização.

Para (Furlan, J.D., 1998) a UML traz uma proposta para uma padronização no projeto e análise de sistemas orientados à objetos; ela é um modelo de linguagem, não um método. Sua notação foi desenvolvida por Grady Booch, James Rumbaugh e Ivan Jacobson. A UML define uma notação e um meta-modelo. A notação são todos os elementos de representação gráfica vistos no modelo, ou seja, é a sintaxe do modelo de linguagem. A notação do diagrama de classe define a representação de itens e conceitos tais como: classe, associação e multiplicidade. Um meta-modelo é um diagrama de classe que define de maneira mais rigorosa a notação.

A utilização desta notação veio a facilitar a documentação do programa implementado, a descrição detalhada será descrita adiante.

5.3. Modelo Proposto

O modelo proposto tem por objetivo disponibilizar uma ferramenta que torne possível a usuários a consulta a qualquer banco de dados utilizando a linguagem natural, sem que para isso o mesmo tenha que conhecer conceitos sobre banco de dados, análise de sistemas ou linguagem SQL. Neste caso a linguagem natural utilizada será a língua portuguesa.

Entrando em mais detalhes sobre o que foi desenvolvido procuraram-se formas alternativas para que o usuário inconscientemente possa gerar ao banco de dados consultas SQL em mais específico o comando SELECT. Por ser um sistema que estará disponível na internet considerou-se que o foco principal será apenas a busca de informações e não será possível a manipulação e nem definição de novos dados no banco de dados, ou seja, a inclusão, exclusão, alteração de dados, criação e modificações de entidade[11] não serão tratadas.

O comando SELECT tem por objetivo buscar informações em um banco de dados envolvendo uma ou mais entidades do mesmo.

A sintaxe completa do comando SELECT pode ser vista modelo abaixo:

SELECT	<lista de atributos>, <funções de agrupamento>
FROM	<lista de entidades>
WHERE	<lista de restrições>,
	<lista de relacionamentos entre as entidades>
AND	<lista de restrições>,
	<lista de relacionamentos entre as entidades>
OR	<lista de restrições>,
	<lista de relacionamentos entre as entidades>
GROUP BY	<lista de atributos>

[11] Entidade - Identifica o objeto de interesse do sistema e tem "vida" própria, ou seja, a representação abstrata de um objeto do mundo real sobre o qual desejamos guardar informações. Exemplo: Clientes, Fornecedores, Alunos, Funcionários, Departamentos, etc.

HAVING <lista de restrições>

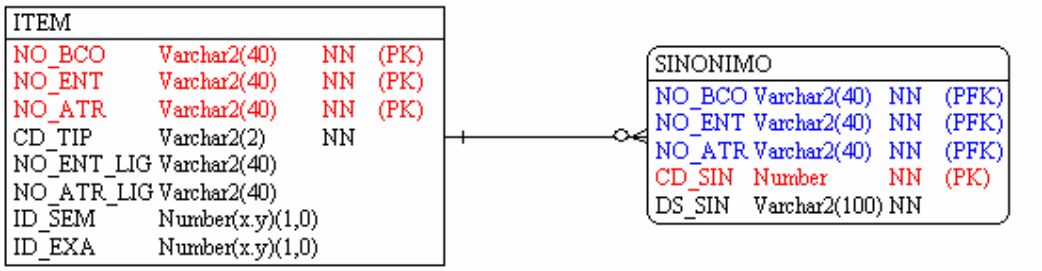
ORDER BY <lista de atributos>

Para que o programa consiga montar o comando SELECT através das informações passadas pelo usuário final de forma transparente, houve a necessidade de montar um banco de dados de sinônimos que será configurado por um usuário que tenha conhecimento da lógica do negócio, ou seja, uma pessoa que conheça sobre o que o banco de dados tenha a oferecer de informações.

O objetivo deste banco é buscar as informações de qualquer outro tipo de banco de dados armazenando nestas duas entidades quais são as entidades, atributos[12], relacionamentos[13] e os sinônimos atributos a eles.

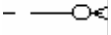
A figura 5.3, representa o Modelo de Entidade e Relacionamento[14] (MER) do banco de dados criado, havendo apenas duas entidades item e sinônimo, um relacionamento entre elas representando uma ligação de 1 item para muitos (N) sinônimos.

Figura 5.3 – Modelo de Entidade e Relacionamento do Sistema



Fonte: Dados da Pesquisa.

Legenda:

Símbolo	Significado
	Lado 1
	Lado N e chave primária e estrangeira
	Lado N e chave estrangeira
(FK)	Chave Estrangeira (Foreign Key)
(PK)	Chave Primária (Primary Key)

[12] Atributo - Informações que desejamos guardar sobre a entidade. Exemplo: Nome do aluno, Número da turma, Endereço do fornecedor, Sexo do funcionário, etc.

[13] Relacionamento - Representa a associação entre os elementos do conjunto de uma entidade com outra entidade.

[14] Consiste em mapear o mundo real do sistema em um modelo gráfico que irá representar o modelo e o relacionamento existente entre os dados.

(PFK)	Chave Primária e Estrangeira (Primary and Foreign Key)

Abaixo uma descrição detalhada do que é e para que serve cada uma das entidades acima mencionadas, assim como, os seus atributos.

A entidade Item (ITEM), tabela 5.1, representa cada entidade e atributo do banco de dados

Tabela 5.1 - Entidade Item

Atributo	Descrição	Função
NO_BCO	Nome do Banco	Nome do Banco de Dados em que a informação está armazenada.
NO_ENT	Nome da Entidade	Nome da Entidade.
NO_ATR	Nome do Atributo	Indica o nome do campo ou do atributo retirado do banco de dados.
CD_TIP	Código do tipo	Código do tipo de Item, aqui será feita uma classificação do item, podendo assumir os seguintes códigos: 1 para atributo, 2 para entidade, 3 para chave primária, 4 para chave estrangeira, 5 para chave primária e estrangeira.
NO_ENT_LIG	Nome da Entidade de Ligação	Serve para indicar qual a entidade o atributo que é chave estrangeira está relacionado.
NO_ATR_LIG	Nome do Atributo de Ligação	Indica dentro da entidade de ligação qual é o atributo que está relacionado.
ID_SEM*	Identificação de Semântico	Indica se o sinônimo é um item semântico ou não.
ID_EXA*	Identificação de Exato	Indica se o sinônimo é um item exato ou não.

* estes atributos serão esclarecidos com mais detalhes a diante.

A entidade Sinônimo (SINONIMO), tabela 5.2, representa os sinônimos que um atributo ou uma tabela podem ter. Quanto mais sinônimos forem atribuídos maior será a probabilidade de o usuário localizar o que esteja procurando.

Tabela 5.2 - Entidade Sinônimo

Atributo	Descrição	Função
NO_BCO	Nome do Banco	Chave estrangeira primária.
NO_ENT	Nome da Entidade	Chave estrangeira primária.
NO_ATR	Nome do Atributo	Chave estrangeira primária.
CD_SIN	Código Sinônimo	Código seqüencial que identifica um sinônimo do item
DS_SIN	Descrição do sinônimo	Palavra que o usuário deverá utilizar para identificar um determinado atributo ou tabela no banco de dados.

Para um melhor esclarecimento do que foi desenvolvido torna-se necessário o conhecimento de alguns termos especiais criados.

Usuário Final (UF): é a pessoa que utilizará o sistema, mas não possui conhecimento sobre análise de sistema, banco de dados ou SQL. Sua função é trabalhar com o sistema, efetuando perguntas em linguagem natural. O único requisito é que seja uma pessoa que tenha conhecimento sobre ambiente de negócio do sistema, para que tenha alguma noção sobre o tipo de questionamento poderá fazer ao programa.

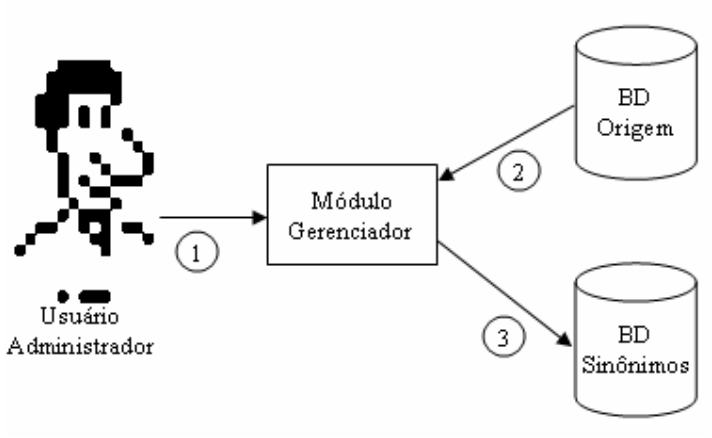
Usuário Administrador (UA): é a pessoa que possui as mesmas funções de um UF, mas que ficará responsável pela configuração inicial e manutenção do sistema para que os UFs possam a partir daí utilizar o sistema. Deve conhecer sobre a estrutura interna do sistema para que possa atribuir sinônimos às entidades e atributos. A pessoa mais indicada para tal tarefa será um Administrado de Banco de Dados (DBA).

O **Módulo Gerenciador (MG)** fará a busca do nome de todas as entidades e atributos relacionados a um determinado banco de dados para que o UA possa fazer a configuração dos sinônimos atribuídos aos mesmos. Uma entidade ou atributo poderá ter mais que um sinônimo, quando mais detalhado for sua configuração mais abrangente será o dicionário disponível ao usuário final para a elaboração das perguntas em linguagem natural ao banco de dados. A figura 5.4 como funcionará o Módulo Gerenciador.

Em 1 o UA entrará com os sinônimos para os atributos e entidades de um determinado banco de dados. Banco de dados (BD) Origem representa qualquer banco de dados.

BD Sinônimos é o local em que estarão armazenados todos os sinônimos atribuídos pelo UA. O número 2 indica a leitura da estrutura do banco de dados de origem para tornar disponível ao UA a configuração dos sinônimos. Em 3 significa a gravação dos sinônimos atribuídos pelo UA.

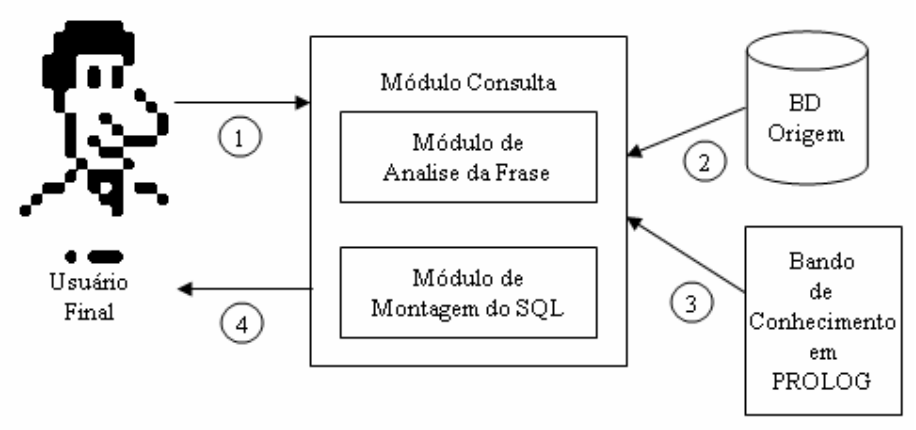
Figura 5.4 – Funcionamento do Módulo Gerenciador



Fonte: Dados da Pesquisa.

O **módulo de Consulta (MC)**, figura 5.5, é o responsável por fazer a interface com o UF e seu objetivo é transformar a entrada em linguagem natural do UF em um comando SELECT do SQL. Para que isso seja possível foram desenvolvidos dois módulos auxiliares denominados Módulo de Análise da Frase e Módulo de Montagem do SQL.

Figura 5.5 - Funcionamento do Módulo Consulta



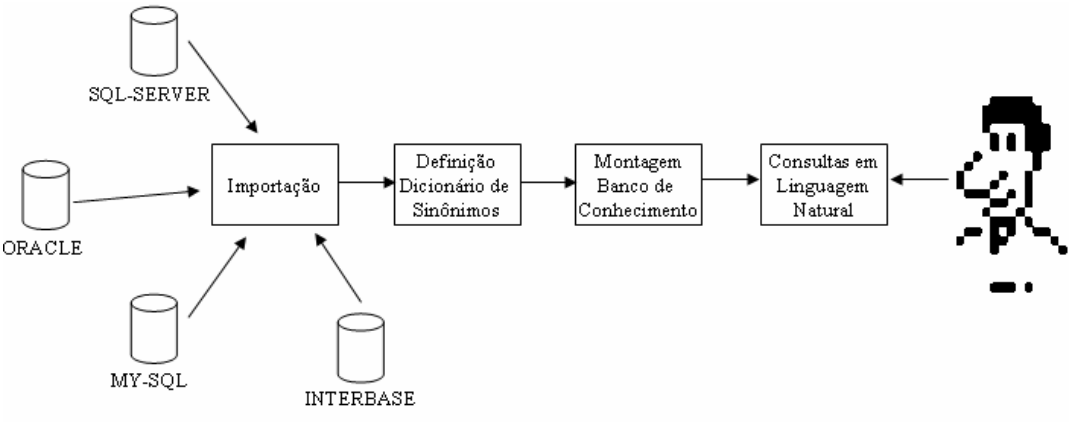
Fonte: Dados da Pesquisa.

Na figura 5.5, o número 1 representa a entrada em LN feita pelo UF.

O número 4 representa o resultado obtido pelo pergunta, caso o sistema consiga ou não responder a solicitação feita pelo UF. O número 2 indica a busca dos dados no BD de Origem. E em 3 a busca pelos sinônimos das entradas. O módulo de Análise de Frase fará a separação dos elementos da frase original passada pelo UF.

A configuração inicial do sistema envolve alguns passos que deverão ser feitos pelo UA. A figura 5.6 demonstra as etapas iniciais.

Figura 5.6 – Configuração Inicial do Sistema



Fonte: Dados da Pesquisa.

A seguir uma descrição sobre o que será realizado em cada etapa da configuração inicial do sistema. Apesar de os módulos implementados estarem disponíveis ao usuário por Internet recomenda-se fazer sua execução localmente ou em um computador próximo ao servidor em que o programa está instalado, por serem processamentos que dependendo do tamanho do banco a ser importado poderá levar um certo tempo.

5.3.1. Módulo de Importação

Esta é a etapa inicial de todo o processo. Sua finalidade é buscar em um banco de dados, identificados pelo UA, as informações sobre: entidades, atributos, relacionamentos entre as entidades, chaves primárias, chaves estrangeiras, chaves primárias e estrangeiras. Nesse momento é que estaremos inserindo na entidade Item, tabela 5.1, todas essas informações.

A busca das informações no Sistema Gerenciador de Banco de Dados[15] (SGBD) é feita com a ajuda do próprio Dicionário de Dados do Banco. O dicionário de dados é uma das mais importantes partes de um SGBD: ele consiste em um conjunto de entidades e visões provenientes da leitura referente ao Banco de Dados. O dicionário contém informações como nome de usuários, direitos e privilégios que eles possuem,

[15] Software que fará o gerenciamento das informações contidas em um Banco de Dados, por exemplo Oracle, MY-SQL, SQL-server, etc...

nome de entidades, atributos, restrições aplicadas a cada entidade. Ele é criado quando o próprio banco de dados é criado, e mantém-se atualizado após cada mudança na estrutura do mesmo.

Se considerarmos o SGBD Oracle, dentre as diversas estruturas que armazenam informações a respeito do próprio banco, podemos citar as que foram utilizadas para buscar as informações necessárias, como:

j) ALL_TAB_COLUMNS: restrições sobre as entidades que se tem acesso;
k) ALL_CONSTRAINTS: contém informações sobre as entidades que se tem acesso;
l) ALL_CONS_COLUMNS: informações sobre acessíveis atributos para as restrições definidas.

Por restrições ou "constraints" entende-se como uma limitação ou conjunto de limitações que um atributo ou uma entidade podem sofrer, como, por exemplo, não poder conter valor nulo, não poder repetir, etc...

O conceito de dicionário de dados existe em praticamente todos os SGBDs disponíveis no mercado; o que diferencia de um para o outro é o nome da entidade na qual estarão armazenando as informações a respeito do banco, tornando possível a importação de outros bancos de dados que possuem um dicionário interno.

5.3.2. Módulo de Definição do Dicionário de Sinônimos

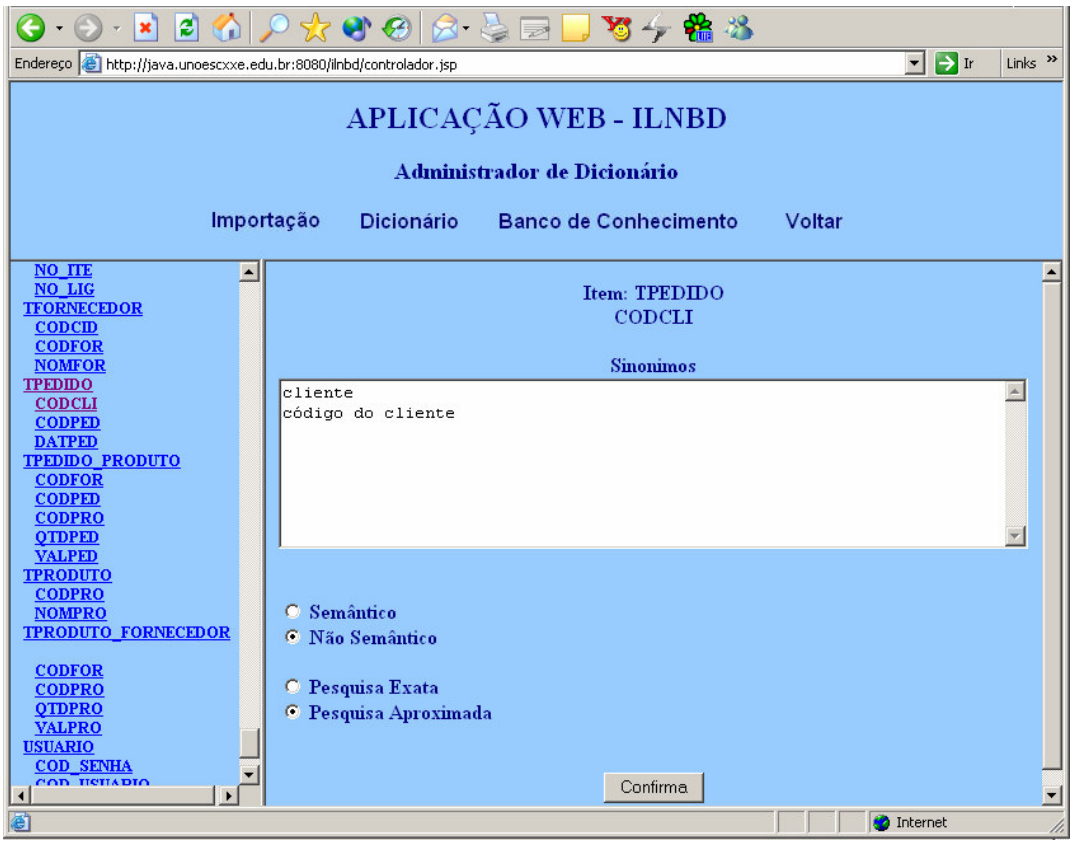
Depois de finalizada a etapa de importação passa-se para a segunda etapa: a definição do dicionário de sinônimos. Nesse momento o UA estará atribuindo sinônimos para todas a entidades e atributos importados.

Os sinônimos funcionarão como palavras-chave que serão consideradas nas perguntas em linguagem natural. A partir das palavras-chave identificadas na frase é que o software conseguirá encontrar quais são as entidades e atributos envolvidos na pergunta feita ao sistema.

A figura 5.7 mostra como ficou o layout do módulo implementado.

Aqui aparecem dois novos conceitos criados para que fosse possível tratar alguns problemas encontrados na hora da montagem do comando SELECT.

Figura 5.7 - Módulo de Definição do Dicionário de Sinônimos



Fonte: Dados da Pesquisa.

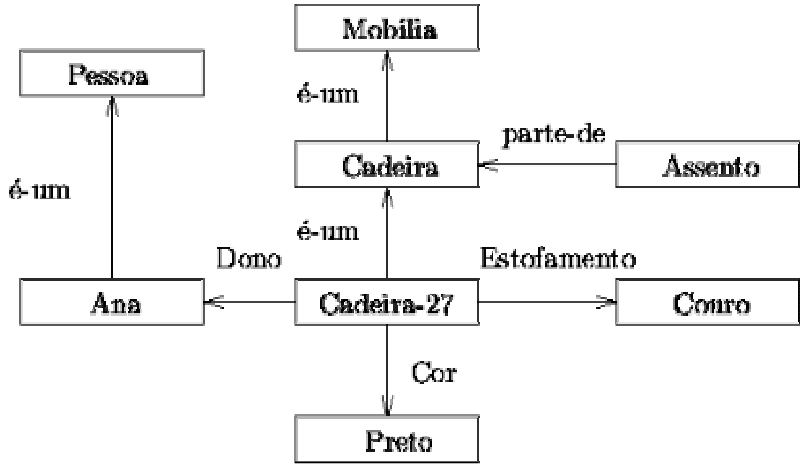
Item Semântico (IS): se um determinado item for marcado como sendo semântico então o programa fará um tratamento especial para este item, ou seja, se for semântico então significa que deverão ser criadas estruturas ou predicados usando o conceito de Redes Semânticas.

Segundo (Rich, E., 1988), rede semântica é um nome utilizado para definir um conjunto heterogêneo de sistemas. Em última análise, a única característica comum a todos estes sistemas é a notação utilizada: uma rede semântica consiste em um conjunto de nós conectados por um conjunto de arcos. Os nós em geral representam objetos e os arcos, relações binárias entre esses objetos. Mas os nós podem também ser utilizados para representar predicados, classes, palavras de uma linguagem, entre outras possíveis interpretações, dependendo do sistema de redes semânticas em questão.

A figura 5.8 mostra um exemplo gráfico de Rede Semântica, apesar de não ser uma forma comum de trabalhar “computacionalmente”; linguagens como o PROLOG facilitam sua representação e implementação. Poderíamos logicamente representar da

seguinte maneira: é_um(cadeira,mobília). Esta representação está indicando que a cadeira é uma mobília e a mesma regra vale para as outras ligações da figura.

Figura 5.8 - Exemplo de Rede Semântica



Fonte: Adaptado de (Rich, E., 1988).

No caso a rede semântica foi utilizada para resolver o seguinte tipo de problema: supondo que o usuário fez a seguinte pergunta ao sistema: "QUAL É O PREÇO DO FEIJÃO?", para a pessoa que fez a pergunta fica claro que feijão é uma informação que pertenceria a uma entidade PRODUTO que estaria armazenada no banco de dados. No entanto para que o computador possa entender isso é necessário representar tal informação. Para finalizar, se o atributo NOME_DO_PRODUTO for configurado como sendo um Item Semântico então será gerado uma representação para o atributo do tipo "é_um" no caso do exemplo ficaria: é_um(feijão, produto). Assim toda vez que o usuário informar a palavra "feijão" no texto o programa irá saber que a entidade envolvida é a PRODUTO.

Outro problema defrontado é com relação ao operador LIKE do SQL. Sua função é a de poder fazer pesquisas aproximadas sobre o valor desejado. Deseja-se, por exemplo, procurar no banco de dados o cliente de nome "João", mas não se tem certeza do seu sobrenome nesse caso a pesquisa SQL, usando como filtragem o operador LIKE no lugar do operador igual "=", irá facilitar a pesquisa. Através da utilização do LIKE podemos listar todos os clientes que possuem no seu nome a palavra "João" bem diferente de se tivéssemos utilizando o operador igual em que a procura teria que ser exatamente igual ao que estivesse armazenado no banco. Para saber se a pesquisa deve

ser feita usando o igual ou LIKE o usuário terá que definir sobre o item se ele sofrerá uma "pesquisa aproximada" com LIKE ou "pesquisa exata", usando o sinal de igualdade. Neste trabalho esse o item será chamado de "**Item Aproximado**" (IAP).

Uma outra situação enfrentada está relacionada com a falta de informações que o usuário poderá estar passado em sua pergunta em linguagem natural. Por exemplo, para a pergunta: "Quais são os produtos cadastrados", inicialmente o sistema não tem como identificar na entidade produto quais os atributos deverão ser mostrados para o usuário, ou seja, não se saber até o momento se devemos mostrar o código do produto, seu nome, ou qualquer outro tipo de informação. Para tratarmos essa situação o UA tem disponível uma opção para indicar se o atributo da entidade é um atributo principal ou não. Caso for considerado principal e na hora da pergunta do usuário o sistema não conseguir identificar nenhum atributo serão mostrados os atributos indicados como "**item principal**" (IPR).

5.3.3. Módulo de Montagem do Banco de Conhecimento

Este será o terceiro passo da configuração inicial do sistema. Neste momento será gerado pelo sistema o Banco de Conhecimento, ou seja, serão gerados todos os predicados em PROLOG identificados pelos: item semântico (IS), item principal (IPR) e item aproximado (IAP).

Abaixo exemplo de alguns predicados gerados:

a) e_um($PEDRO$,$NOMCLI$,$TCLIENTE$,$LIKE$);
b) e_um($JOINVILLE$,$NOMCID$,$TCIDADE$,$LIKE$);
c) sin(atributo,$TPRODUTO$,$NOMPRO$,$PRODUTO$);
d) sin(atributo,$TPRODUTO$,$NOMPRO$,$NOME DO PRODUTO$);
e) sin(entidade,$TCIDADE$,$CIDADE$);
f) sin(entidade,$TCIDADE$,$CIDADES$);
g) relaciona($TCIDADE$,$TCLIENTE$,2,$CODCID$,$CODCID$);
h) relaciona($TPEDIDO$,$TPED_PROD$,1,$CODPED$,$CODPED$).

Em "a" o nome do predicado "e_um" significa que a palavra "PEDRO" é um nome de cliente "NOMCLI" encontra-se na entidade "TCLIENTE" e que a forma de

pesquisa que deverá ser utilizada sobre este atributo é aproximada usando o operador LIKE do SQL.

A mesma regra vale para o predicado exemplificado pela letra “b”, “JOINVILLE” é um nome de cidade “NOMCID”, armazenada na entidade “TCIDADE” e com pesquisa aproximada.

Para que no momento da análise da frase não seja necessário estar pesquisando toda vez no banco de dados de sinônimos, optou-se em gerar predicados também para os mesmos o que tornará bem mais rápido este processo. Abaixo exemplos da regra para os sinônimos:

Os predicados das letras “c” e “d” significam que “NOMPRO” é um atributo da entidade “TPRODUTO” e para a primeira regra o sinônimo associado a ele é a palavra “PRODUTO” e na segunda é “NOME DO PRODUTO”.

O predicado “sin” além de representar os sinônimos dos atributos também estará representando os sinônimos das entidades como nos exemplos das letras “e” e “f”: em “e” significa que a entidade “TCIDADE” possui como sinônimo a palavra “CIDADE” e em “F” seu plural representado pela palavra “CIDADES”.

Eventualmente a pergunta digitada pelo usuário poderá conter referência a apenas algumas entidades do banco de dados, e poderão existir outras entidades intermediárias a elas. Para que o sistema consiga saber quais são todas as entidades envolvidas em uma pergunta houve a necessidade da criação do predicado relaciona. Esse predicado será responsável por armazenar as informações sobre os relacionamentos entre todas a entidades do banco de dados e será útil na situação apresentada neste parágrafo.

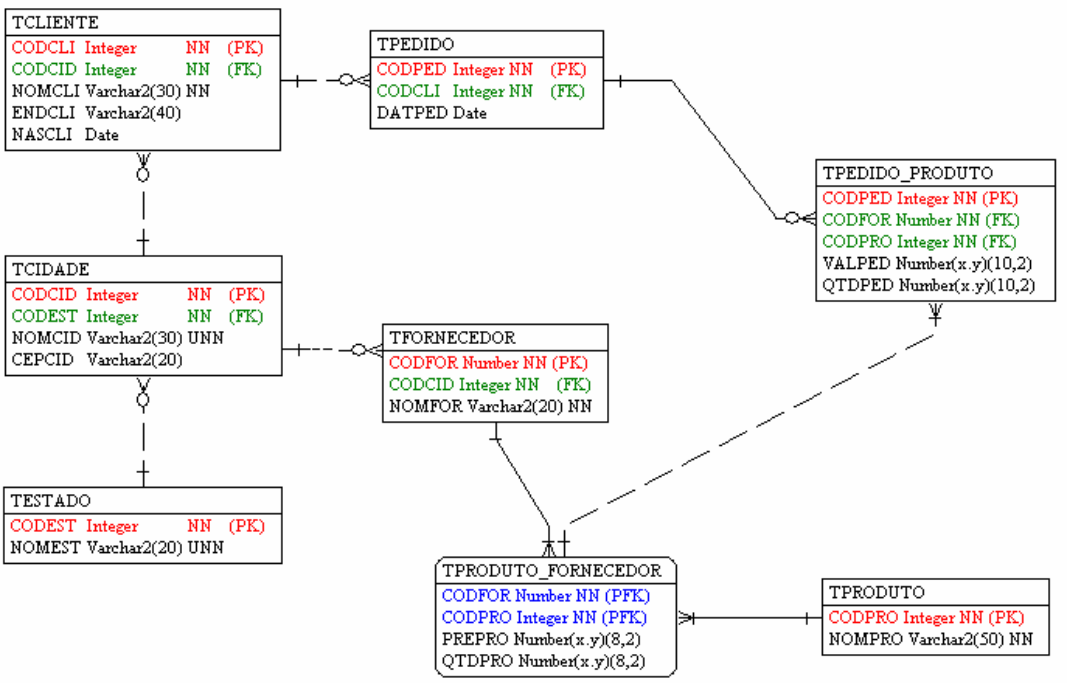
Caso as entidades identificadas na pergunta do usuário não estejam próximas ou se existirem entidades intermediárias então o predicado “relaciona” conterá as informações para fazer a busca de todas as entidades envolvidas e que deverão aparecer no comando SELECT.

O exemplo do predicado relaciona pode ser visto nas letras “g” e “h”; o significado deles são: em “g” a entidade “TCIDADE” está relacionada com “TCLIENTE” e o peso atribuído ao relacionamento é (2) dois por ser um relacionamento 1:N entre “TCIDADE ” e “TCLIENTE” (1 cidade para N “vários” clientes), o código da cidade “CODCID” é o atributo da entidade “TCIDADE” que está

relacionado ao atributo “CODCID” da entidade “TCLIENTE”. O que difere a interpretação da letra “h” com relação a “g” é o fato de que as entidades e atributos envolvidos apesar de se tratar de um relacionamento de grau 1:N entre a entidade “TPRODUTO” e “TPED_PROD” (itens do pedido) mas neste caso a chave (CODPED) é estrangeira primária na entidade “TPED_PROD”, por isso ter considerado o peso com o valor 1 (um).

Abaixo na figura 5.9, um exemplo de MER que exemplifica o problema identificado.

Figura 5.9 - Exemplo de Modelo de Entidade e Relacionamento



Fonte: Dados da Pesquisa.

Supondo que a pergunta fornecida pelo usuário envolva as entidades “TPRODUTO” e “TCLIENTE” neste caso existem duas possibilidades de caminhos que podem ligar as duas entidades que são:

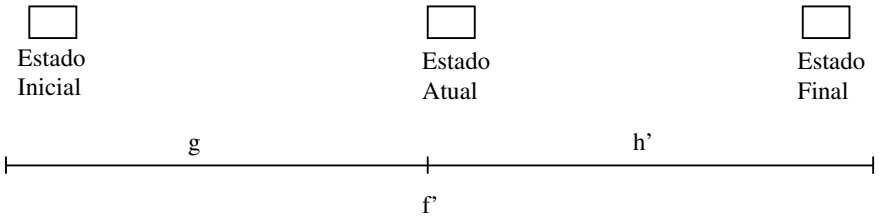
a) caminho 1 (um), iniciar em TPRODUTO, TPRODUTO_FORNECEDOR, TFORNECEDOR, TICIDADE E TCLIENTE;

b) caminho 2 (dois) partindo de TPRODUTO, TPRODUTO_FORNECEDOR, TPEDIDO_PRODUTO, TPEDIDO e TCLIENTE.

As duas possibilidades servem como caminhos, no entanto se levarmos em consideração a realidade da situação apresentada, não tem sentido utilizar o caminho 1 (um) mas sim a lista das entidades envolvidas no caminho 2 (dois). Para que o sistema consiga achar o caminho certo foram considerados os pesos na hora da criação dos predicados de relacionamento, esses pesos serão atribuídos dependendo do tipo de relacionamento que existe entre as entidades. O caminho que será escolhido será aquele que possuir a menor distância ou nesse caso menor peso.

A estratégia de busca heurística utilizada para resolver esse problema foi a busca pela melhor escolha, que se caracteriza pelo fato de cada etapa do processo de busca escolher-se o nó mais promissor, gerado até aquele momento, figura 5.10.

Figura 5.10 - Função Heurística f'



Fonte: Adaptado de (Rabuske, R. A., 1995).

A Função Heurística f' tem como objetivo estimar os méritos de cada nó gerado.

$$\mathbf{f' = g + h'}$$

a) **f'** é a função heurística e representa uma estimativa de custo para sair do estado inicial ao estado final;
b) **g** é a soma dos custos desde o estado inicial até o estado atual (corrente);
c) **h'** é a estimativa de custos para sair do estado atual e chegar ao estado final.

Obs.: no estado Final h' = 0 e g = 0 para o estado Inicial.

O funcionamento começa a partir de um estado inicial qualquer: aplica-se os operadores para o problema que faz a geração de novos filhos e para a escolha de qual utilizar aplica-se às heurísticas apresentadas anteriormente. Aquele nó que possuir o menor valor de f' será o mais promissor; portanto a partir deste se não for o estado final serão gerados novos filhos (sucessores).

Os nós que foram gerados, ou seja, que foi aplicado a ele um operador é colocado em uma lista chamada de LISTA_DE_NÓS_ABERTOS. Aqueles nós que forem gerados, e que forem selecionados por possuírem menor custo serão colocados em uma LISTA_DE_NÓS_FECHADOS.

Na figura 5.11 a descrição do algoritmo de Busca pela Melhor Escolha:

Figura 5.11 - Algoritmo de Busca pela Melhor Escolha

1) Criar uma lista de nós abertos contento o estado inicial do problema;
2) Até que a lista de nós abertos (ABERTOS) esteja vazia ou que o estado meta tenha sido alcançado faça;
3) Se o primeiro elemento for o nó destino pare;
4) Senão
   i. Tire o 1º elemento de ABERTOS;
   ii. Gerar seus sucessores, e o colocar em ABERTOS;
   iii. Se este nó ainda não tenha sido expandido colocar numa lista de nós fechados (FECHADOS);
   iv. Se este nó já está em FECHADOS, verificar qual é o melhor (que tem menor g – custo);
   v. Deixe o melhor, atualizar o custo para chegar ao nó atual;
   vi. Registrar o pai dos novos elementos de ABERTOS;
5) Ordenar ABERTOS segundo o custo f' estimado para chegar ao destino;
6) Fim Até.

Fonte: Adaptado de (Rabuske, R. A., 1995).

Conforme visto em (Rabuske, R. A., 1995), se h' for obtida por um estimador ótimo, então o algoritmo convergirá diretamente para a solução, sem desperdícios. Se h' for sempre menor ou igual a h, então certamente o algoritmo encontrará um caminho ótimo até uma meta (teorema da admissibilidade). No caso do algoritmo implementado foi considerado este teorema como forma de se chegar à solução ótima.

A partir deste momento o sistema está pronto para receber as consultas em linguagem natural pelo usuário final.

5.3.4. Módulo de Consulta

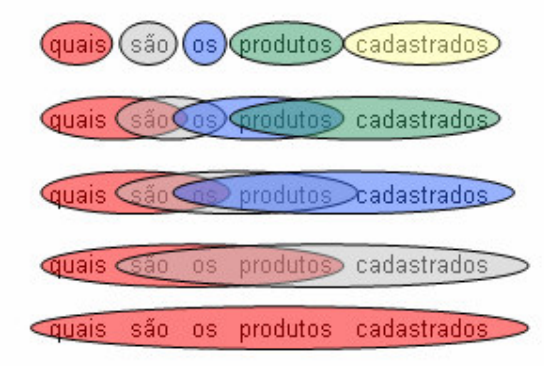
Depois de feita a configuração inicial do sistema o UF poderá utilizar o sistema para fazer suas consultas em linguagem natural. Só será permita a entrada de perguntas que possam ter como resposta uma consulta SQL elaborada pela cláusula SELECT, ou

seja, o usuário final não poderá efetuar nenhuma operação de atualização de dados, criação de tabelas através da interface.

Para que esta etapa fosse disponibilizada sentiu-se a necessidade de se montar um algoritmo para a análise da frase passada pelo usuário; isso foi feito a partir de um módulo chamado de Análise da Frase.

O primeiro passo da análise será a identificação das palavras-chave e para otimizar este processo foi elaborado um algoritmo que analisa todas as palavras da frase, agrupando-as a cada nova análise. A figura 5.12 demonstra, através de um desenho, a lógica utilizada. O primeiro passo serão testadas todas as palavras individualmente; no segundo as mesmas serão agrupadas de duas em duas e assim por diante até que seja analisada um grupo com todas as palavras, sempre começando da primeira palavra à esquerda. A figura abaixo representa a análise feita sobre a frase: "quais são os produtos cadastrados".

Figura 5.12 – Análise das Palavras-chaves



Fonte: Dados da Pesquisa.

Já a figura 5.13 estão listadas todos grupos gerados e analisados na frase. A cada novo agrupamento é verificado se a palavra existe ou não no dicionário de sinônimos gerados pelo predicado "sinônimo" do PROLOG; caso algum sinônimo existir então se faz uma verificação para ver se este representa uma entidade ou um atributo do banco de dados; as demais palavras serão ignoradas.

Depois de repassados por todos os grupos de palavras então partimos para o próximo passo que é a verificação de entidades intermediárias aquelas identificadas na etapa anterior. Esse processo é feito usando o algoritmo de busca pela melhor escolha que foi detalhada anteriormente.

As entidades, atributos e relacionamentos identificados são armazenados em vetores dinâmicos, que são estruturas que armazenam informações e são capazes de aumentar sua capacidade de armazenamento à medida que os dados estão sendo inseridos. Estes vetores serão utilizados na etapa de montagem do SQL.

Figura 5.13 - Listas com todas as palavras geradas

Frase: “quais são os produtos cadastrados”
quais
são
os
produtos
cadastrados
quais são
são os
os produtos
produtos cadastrados
quais são os
são os produtos
os produtos cadastrados
quais são os produtos
são os produtos cadastrados
quais são os produtos cadastrados

Fonte: Dados da Pesquisa.

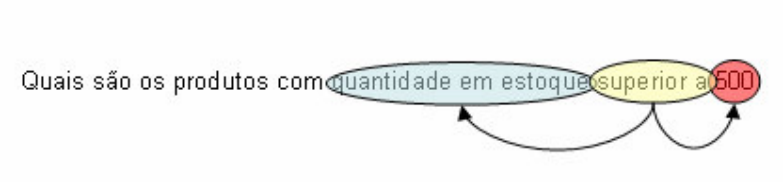
A montagem do comando SQL será feita considerando as partes do comando SELECT; cada rotina esta relacionada ao tratamento de uma das cláusulas do comando. Neste momento já são conhecidos quais são as entidades envolvidas na pergunta; então monta-se a cláusula FROM e os atributos na cláusula SELECT. Com os relacionamentos podem-se montar as condições de ligações das entidades na cláusula WHERE.

Caso a etapa de análise da frase identificar restrições relacionadas aos comandos maior ou igual, maior, menor ou igual, menor, igual e diferente serão inseridas também na cláusula WHERE as filtragens. Este processo será chamado de **Análise de Restrições de Quantidades**. Para que o mesmo possa ser inserido no comando SELECT, respeitando os padrões SQL, usou-se o seguinte procedimento: se for identificada na frase palavra-chave relacionada a restrições então se identifica em qual posição na frase ela foi encontrada. A partir deste momento é percorrida a palavra à esquerda e à direita, começando da posição encontrada; caso a pesquisa para esquerda

encontrar algum atributo de uma entidade e a pesquisa para a direita encontrar algum valor numérico então significa que o usuário em sua pergunta está querendo efetuar uma filtragem em sua resposta desejada.

A figura 5.14 ilustra um exemplo prático para este fato, na frase “quais são os produtos com quantidade em estoque superior a 500”. Após feita a análise da frase a palavra-chave “superior a” irá identificar uma restrição equivalente ao sinal “maior”. Então a partir da posição de onde foi encontrada a palavra-chave é feita a busca pelo atributo de uma entidade e pela quantidade, uma à esquerda e outra à direita, conforme a figura. Neste caso o atributo quantidade em estoque foi identificado como atributo e 500 como valor numérico, criando assim uma nova restrição ao SQL que será “AND QUANTIDADE_EM_ESTOQUE > 500”

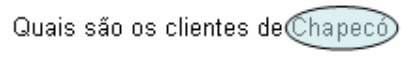
Figura 5.14 - Análise de Restrições de Quantidades



Fonte: Dados da Pesquisa.

A etapa de análise da frase também poderá identificar os itens semânticos (IS); caso isto ocorrer ele também será considerado como uma nova restrição a ser inserida no comando SQL. Este fato chamaremos de **Análise de Restrição Semântica** e a figura 5.15 ilustra um exemplo. Para a frase “Quais são os clientes de Chapecó”, caso UA tenha configurado o atributo “NOMCID” (nome da cidade) como sendo um item semântico, então a palavra “Chapecó” será considerada como sendo um “nome de cidade” da entidade “cidade” adicionando ao SQL a restrição “AND NOMCID = ‘CHAPECÓ’”, caso for uma pesquisa exata ou “AND NOMCID LIKE ‘%CHAPECÓ%’”, caso for aproximada.

Figura 5.15 - Análise de Restrição Semântica



Fonte: Dados da Pesquisa.

A mesma resolução será aplicada para identificar as cláusulas ORDER BY, GROUP BY e funções de agrupamento como SUM, AVG, MIN, MAX, COUNT, DISTINCT do SQL.

A figura 5.16, mostra a interface com o Usuário final, bem como o resultado de uma pergunta em linguagem natural.

Figura 5.16 - Interface com o Usuário

Fonte: Dados da Pesquisa.

A frase também passa por uma análise gramatical antes do processo de geração do código SQL. Para que isso se torne possível foi utilizado o dicionário on-line de palavras criado pelo JSpell[16]. Conforme (Almeida *et al.*, 1993) o JSpell é um módulo analisador léxico-morfológico genérico de domínio público, parametrizado por dicionário (lista de palavras) e regras de flexão utilizado como corretor ortográfico.

Conforme (Almeida *et al.*, 1994) grande parte das aplicações em linguagem natural necessitam de um mecanismo de classificação léxico, que dada uma palavra obtenha-se informações a seu respeito, sua origem, categoria gramatical e outros elementos (features) como gênero, número, etc...

[16] Para maiores detalhes veja em http://natura.di.uminho.pt/~jj/pln/pln.html

O dicionário tem sua origem no projeto Natura que tem como principal objetivo criar e disponibilizar recursos de Processamento de Linguagem Natural com particular ênfase na língua Portuguesa.

A mesma representação utilizada para as classes gramaticais será a mesma que a utilizada no JSpell facilitando a integração e posteriores atualizações feitas no dicionário de palavras do mesmo.

Abaixo uma lista com as representações utilizadas:

a) CAT – categoria;
b) G – gênero: m – masculino; f – feminino; _ - ambos;
c) N – número: s – singular; p – plural; _ - ambos;
d) P – pessoa: 1; 2; 3;
e) T – tempo
f) TR – Transitividade: GR – Grau: FSEM – função semântica;

Categorias Gramaticais consideradas no dicionário:

a) nc - nome comum;
b) np – nome próprio;
c) adj – adjetivo;
d) a_nc – adjetivo e ou nome comum;
e) v – verbo;
f) art – artigo;
g) ncard – numeral cardinal;
h) nord – numeral ordinal;
i) nmult – numeral multiplicativo;
j) nfrac – numeral francionário;
k) prep – preposição;
l) adv – advérvio;
m) ppes – pronome pessoal;
n) pdem – pronome demonstrativo;
o) ppos – pronome possessivo;
p) pind – pronome indefinido;
q) prel – pronome relativo;
r) pint - pronome interrogativo;

s) conj – conjunção;

t) in – interjeição.

No total foram geradas 38962 palavras com suas respectivas classificações, a partir desse dicionário e após a entrada em linguagem natural do usuário a frase será verificada se esta gramaticalmente correta. Caso a frase não esteja correta será gerada uma lista contento a frase e as classes gramaticais identificadas, onde a partir desta lista o UA poderá fazer a geração de novas gramáticas para que em próximas entradas com as mesmas características o software consiga reconhecer a frase como gramaticalmente correta.

Abaixo uma lista com alguns exemplos gerados:

a) vt,ordenar;

b) vn,estimar;

c) vi,intervir;

d) npm,euclides;

e) npf,natália;

f) nm,edifício;

g) nm,canoeiro.

Esta lista encontra-se em um arquivo do tipo texto e que através do sistema são passadas para predicados utilizando as regras gramáticas do DCG conforme descrito no item 2.3.2 (Gramática).

É importante resaltar que apesar da grande quantidade de dados geradas pelo dicionário Jspell o prolog se portou de forma a conseguir uma performance excelente no momento em que foi solicitado a busca ao seu banco de informações. A grande dificuldade encontrada foi que o prolog não permitiu sua geração a partir da sua interface de desenvolvimento devido a quantidade de dados, ele não permitiu fazer a compilação e linkagem dos mesmos como descrito no item 5.2.3 (Ligação entre Java e PROLOG). Para solucionar essa deficiencia teve-se que criar um arquivo com extensão “*.xpl” vazio no prolog e um arquivo texto “*.txt” contendo o dicionário das palavras do JSpell . Feito isso foi criada uma classe no Java que faz a leitura do arquivo texto e gera-se dinamicamente em forma de predicados através do comando “assert” do prolog todos os predicados necessários, mais detalhes em A.6 (Classe GramaBean). Dessa forma conseguimos criar a base de conhecimento, esse processo deverá ser executado

toda vez que se desejar importar novas palavras ao dicionário ficando disponível automaticamente para todos os usuários do sistema.

5.4. Arquitetura do Modelo

Toda a programação que utiliza a tecnologia JSP foi desenvolvida utilizando os conceitos da Programação Orientada a Objetos, ou seja, todo código fonte está organizado em classes Java ou pequenos programas que poderão ser reutilizados para outros sistemas, que poderão ser desenvolvidos, e que desejem disponibilizar aos seus usuários a facilidade de consultas feitas em linguagem natural.

A descrição detalhada das principais classes implementadas assim como os diagramas de Classes são detalhados no apêndice A (Diagramas UML).

5.5. Resultados Práticos

Antes de disponibilizar o sistema para usuários finais foram feitos alguns teste para validar o modelo proposto. Abaixo uma lista de algumas perguntas feitas ao sistema e o retorno dado. As questões foram feitas baseados no modelo de banco de dados da figura 5.9.

Pergunta: “Quais são os produtos cadastrados?”.

Resposta:

```
SELECT NOMPRO
FROM TPRODUTO
```

Pergunta: “Mostre os produtos ordenados por código?”.

Resposta:

```
SELECT NOMPRO
FROM TPRODUTO
ORDER BY CODPRO
```

Pergunta: “Quais são os clientes de Xanxerê?”.

Resposta:

```
SELECT NOMCLI
FROM TCLIENTE, TCIDADE
WHERE TCLIENTE.CODCID = TCIDADE .CODCID
AND TCIDADE.NOMCID LIKE '%XANXERÊ%'
```

Pergunta: "Quais são os produtos com quantidade em estoque superior a 500?"

Resposta:

```
SELECT NOMPRO
FROM TPRODUTO
WHERE TPRODUTO, TPRODUTO_FORNECEDOR
AND TPRODUTO.CODPRO = TPRODUTO_FORNECEDOR.CODPRO
AND TPRODUTO_FORNECEDOR > 500
```

Pergunta: "Quais são os produtos adquiridos pelo João?".

Resposta:

```
SELECT NOMPRO
FROM TPRODUTO, TPRODUTO_FORNECEDOR, TPEDIDO_PRODUTO, TPEDIDO,
TCLIENTE
WHERE TPRODUTO.CODPRO = TPRODUTO_FORNECEDOR.CODPRO
AND TPRODUTO_FORNECEDOR.CODFOR = TPEDIDO_PRODUTO.CODFOR
AND TPRODUTO_FORNECEDOR.CODPRO = TPEDIDO_PRODUTO.CODPRO
AND TPEDIDO_PRODUTO.CODPED = TPEDIDO.CODPED
AND TPEDIDO.CODCLI = TCLIENTE.CODCLI
```

Pergunta: "Quantos pedidos foram feitos entre 01/01/2003 a 30/06/2003".

Resposta:

```
SELECT COUNT(*)
FROM TPEDIDO
WHERE TPEDIDO.DATPED BETWEEN '01/01/2003' AND '30/07/2003'
```

Pergunta: "Quantos pedidos foram no primeiro semestre de 2003?".

Resposta:

O software não conseguiu entender esta pergunta. Para esta situação poderia-se criar um dicionário de termo como por exemplo: primeiro semestre = <atributo> BETWEEN '01/01'+AnoCorrente AND '30/07'+AnoCorrete.

Pergunta: “Qual é o total dos pedidos feitos pelo João?”.

Resposta:

```
SELECT SUM(TPEDIDO_PRODUTO.VALPED)
FROM TPEDIDO_PRODUTO, TPEDIDO, TCLIENTE
WHERE TPEDIDO_PRODUTO.CODPED = TPEDIDO.CODPED
AND TPEDIDO.CODCLI = TCLIENTE.CODCLI
AND TCLIENTE.NOMCLI LIKE ‘%JOÃO%’
```

O software não conseguiu entender que o total do pedido será feito utilizando a soma do valor do pedido multiplicado pela quantidade pedido e somou apenas ao valor de cada item do pedido. Como solução poderíamos criar um dicionário de fórmulas especificando que total_do_pedido = QTDPED * VALPED.

Pergunta: “Quantidade de Clientes por Cidade”.

Resposta:

```
SELECT TCLIENTE.NOMCID, COUNT(*)
FROM TCLIENTE
GROUP BY 1
```

O software não foi capaz de reconhecer necessidade de subconsultas.

5.6. Conclusão

Com relação ao Java utilizou-se como uma ferramenta para fazer a interface amigável com o usuário e possibilitou que o código implementado possa estar sendo utilizado em outras aplicações que necessitem do PLN como forma de acesso ao banco de dados, bastando o programador passar a perguntas em linguagem natural para a classe e ela retornará o código SQL ou os dados pesquisados. Desde que o sistema consiga ter acesso à linguagem Java, não importando se for um sistema Cliente-Servidor ou um sistema para a Internet.

Outro ponto importante a ser considerado está relacionado com a linguagem PROLOG que para o PLN tornou o sistema bem mais ágil e rápido para a análise da

frase, além de facilitar a implementação da identificação de palavras-chave e o método de busca pela melhor escolha.

O sistema encontra-se instalado no servidor da Unoesc Campus Xanxerê e poderá ser acessado pelo endereço http://www.unoescxxe.edu.br:8080/ilnbd. E está configurado para responder a questões relacionadas com o modelo apresentado neste capítulo.

A intenção é utilizá-lo junto a alguns setores do campus bem como disponibiliza-los para os alunos para obter consultas

Capítulo 6

Considerações Finais

Apresentou-se neste trabalho uma proposta e implementação de um sistema em Linguagem Natural para Banco de Dados desenvolvido para comprovação do modelo proposto. Acreditamos ter alcançado um grande avanço no que tange o assunto sobre interfaces em linguagem natural para Banco de Dados.

Deparamos com uma série de dificuldades que aos poucos foram se acertando, mas que, no entanto não esgota de forma alguma o conjunto de aplicações e implementações que poderão ser desenvolvidas.

O sistema desenvolvido possui uma série de aplicações práticas que poderão utilizar a ILNBD como forma de encontrar a resposta desejada.

6.1. Contribuições do Trabalho

Após o estudo sobre problemas envolvidos no desenvolvimento de Interfaces em Linguagem Natural este trabalho apresenta uma proposta de desenvolvimento de uma interface que é independente de um domínio específico, ou seja, foi proposto e desenvolvido um sistema que permite a transportabilidade exigida no desenvolvimento desses tipos de softwares. O mesmo software poderá ser utilizado para qualquer domínio e qualquer tipo de SGBD que a linguagem Java consiga se comunicar.

Com o desenvolvimento da interface apresenta-se uma sugestão para o problema da transparência para o usuário onde evita-se a necessidade de ter que aprender uma linguagem ou uma ferramenta que exige um treinamento mais avançado.

Utilizou-se um léxico que por ser aberto ao público está em constante evolução e é facilmente adaptado ao sistema desenvolvido porque possui as mesmas regras gramaticais. Servindo como uma proposta para a resolução de uma outra dificuldade encontrada no desenvolvimento deste tipo de sistema.

6.2. Sugestões para Trabalhos Futuros

O assunto desenvolvido nesta dissertação dá margem a outras pesquisas relacionadas à área. A seguir, são relacionadas algumas sugestões que se julgam pertinentes e que não foram abordadas neste trabalho e poderão ser acrescidas no mesmo:

a) tratamento sobre ambigüidades, léxica, sintática, semântica e anafórica;

b) tratamento sobre a análise de discurso onde o usuário poderá entrar com diversas frases e o sistema deverá entender sobre o assunto que está envolvido;

c) estudo sobre as formas como os resultados poderão ser apresentados para o usuário;

d) geração automática do plural das palavras;

e) processo de recuperação baseado na similaridade através do matching parcial entre o ajuste da situação fornecido inicialmente em linguagem natural, e as regras armazenadas na base de conhecimento do programa, usando uma medida de similaridade.

Referências Bibliográficas

[Allen, 1995] ALLEN, J. **Natural Language Understanding**. 2nd edition. CA: Benjamim Cummings, 1995.

[Almeida *et al.*, 1993] ALMEIDA, JOSÉ JOÃO DIAS DE ALMEIDA, PINTO ULISSES. **Manual do Utilizador do JSpell.** Departamento de Informática. Universidade do Minho. Campus de Gualtar. <http://natura.di.uminho.pt/~jj>. Acessado em: abril, 2003.

[Almeida *et al.*, 1994] ALMEIDA, JOSÉ JOÃO DIAS DE ALMEIDA, PINTO ULISSES. **JSpell – um modulo para Análise Léxica Genérica de Linguagem Natural.** Departamento de Informática. Universidade do Minho. Campus de Gualtar. <http://natura.di.uminho.pt/~jj>. Acessado em: abril, 2003.

[Androutsopoulos, I., 1992] ANDROUTSOPOULOS, I. **Interfacing a Natural Language Front-End to a Relational Database.** Universidade de Edinburgh. 1992

[Androutsopoulos *et al*., 1995] ANDROUTSOPOULOS, I., RITCHIE, G. D., THANISCH, P. **Natural Language Interfaces to Databases: An Introduction.** DAÍ Research Paper Nº. 709, Dep. of Artificial Intelligence, Edinburgh University, Scotland, UK.

[Anick, Peter G., 1993] ANICK, PETER G. **Integrating natural language processing and information retrieval in a troubleshooting help desk.** In IEEE Expert, p. 9-17, Dec 1993.

[Bomfim Júnior, F. T., 2002] BOMFIM JÚNIOR, FRANCISCO TARCIZO. **JSP: a Tecnologia Java na Internet.** Editora Érica. São Paulo, 2002. ISBN: 85-7194-876-3.

[Copestake & Jones, 1989] COPESTAKE, A., JONES, K. S. **Natural Language Interfaces to Databases.** Computer Laboratory, University of Cambridge, 1989.

[Coulon, D., 1992] COULON, DANIEL. **Informática e Linguagem Natural: uma visão geral dos métodos de interpretação de textos escritos.** Brasília: IBICT; Rio de Janeiro: Senai, 1992. ISBN: 85-7013-021-X.

[Chomsky, N., 1965] CHOMSKY, N. **Aspects of the theory of syntax.** Cambridge, Mass: The MIT Press.1965.

[Clocksin, W.F. & Mellish, C.S., 1981] CLOCKSIN, WILLIAM F. & MELLISH, CHRISTOPHER S. **Programming in Prolog.** Springer-Verlag. Nova Yorque. ISBN: 3-540-58350-5.

[Crane, H., 1993] CRANE, HEWITT D., RTISCHEV, DIMITRY**. Pen and voice unite**. In BYTE, p. 98-102, Oct 1993.

[Epstein, S. S., 1985] EPSTEIN, S.S. **Transportable Natural Language Processing through Simplicity. The PRE System.** ACM Transaction on Office Information Systems 3, 1985.

[Fischler, A., 1987] FISCHLER, MARTIN A., FIRSCHEIN, OSCAR. **Intelligence: the eye, the brain, and the computer.** Menlo Park: AddisonWesley, 1987

[Filipe, P.P.,1999] FILIPE, PENA, PORFÍRIO. **Sistema de Interrogações em Língua Natural para Base de Dados Modelo Conceptual, Aquisição de Vocabulário e Tradução.** Universidade Técnica de Lisboa. Julho de 1999.

[Filipe & Mamede, 1998] FILIPE, P., MAMEDE, N. **Databases and Natural Language Interfaces.** Prentice Hall, December 1998.

[Furlan, J.D., 1998] FURLAN, JOSÉ DAVI. **Modelagem de Objetos Através da UML.** São Paulo, Makron Books, 1998.

[Genereux M., 1999] GENEREUX, M. **A comparative evaluation of graph and term unification in a Natural Language Interface system for Database Queries.** United Kingdom, University of Essex, 1999.

[Tennant, H. 1981] TENNANT, HARRY. **Natural Language Processing. An Introduction to an Emerging Techonology.** Petrocelli Books, Inc. United States. 1981. ISBN: 0-89433-100-0.

[Hinrichs, 1988] HINRICHS, ERCHARD W. **Tense, Quantifiers and Contexts. Computacional Ligüistics.** Vol. 14, No. 2, Junho, University of Illinois 1988.

[Komosinski, Leandro J., 2002] KOMOSINSKI, LEANDRO J. **Aplicações Cliente-Servidor via Web usando Java.** Apostila do curso de Ciência da Computação UFSC.

[Lima, T. S., 2001] LIMA, TIAGO SANTOS. **Semântica de Montague.** Disponível em: <http://www.inf.ufpr.br/~tiago/public/texts/montague.ps.gz>. Acessado em: julho, 2003.

[Long, B., 1994] LONG, B. **Natural Language as an Interface Style**. Disponível em: <http://www.dgp.utoronto.ca/~Byron>. Acessado em: abril,2002.

[Nascimento, A., 2001] NASCIMENTO, ANDRÉ. **Uma Interface para consulta a Sistemas de Suporte à Decisão Baseado em Linguagem Natural, casamento de padrões e Metadados.** Brasil. Campina Grande, 2001.

[Nickerson, R.S., 1986] NICKERSON, R. S. **Using Computers: Human Factors in Information Systems.** MIT Press, 1986.

[Nunes et al.,1999] NUNES, M. G. V., SILVA, B. C. D. **Introdução ao Processamento das Línguas Naturais. Instituto de Ciências Matemáticas e de Computação.** São Carlos, SP, Junho 1999. ISBN - 0103-2577.

[Palazzo, L. A. M., 1997] PALAZZO, LUIZ, A. M. **Introdução à Programação PROLOG.** Editora da Universidade Católica de Pelotas. Pelotas, RS. 1997.

[Rabuske, R. A., 1995] RABUSKE, RENATO ANTONIO. **Inteligência Artificial.** Editora da UFSC, Florianópolis, SC. 1995.

[Reis et al., 1997] REIS, P., MATIAS, J., MAMEDE, N. **Edite - A Natural Language Interface to Databases: a New Dimension for an Old Approach.** Proceeding of the Fourth International Conference on Information and Communication Technology in Tourism, ENTER'97, Edinburgh, Escócia. Springer-Verlag, 1997.

[Rich, E., 1988] RICH, E. **Inteligênia Artificial.** Tradução Newton Vasconcellos; revisão Nizam Omar. McGraw-Hill, São Paulo 1988.

[Rich, E., 1993] RICH, E.; KNIGHT, K. **Inteligência Artificial.** Makron Books, 1993.

[Russel & Norvig, 1998] RUSSEL, S. J., NORVIG, P. **Artificial Intelligence: A Modern Approach.** Prentice Hall, December 1998.

[Savadovsky, 1988] SAVADOVSKY, P. **Introdução ao Projeto de Interfaces em Linguagem Natural.** São Paulo: SID Informática, 1988.

[Sethi, R., 1989] SETHI, R. **Programming languages: concepts and constructs.** AddisonWesley, Reading, Massachussetts, 1989. ISBN 0-201-10365-6.

[Shneiderman, B, 1998] SHNEIDERMAN, B. **Designing the user interface: strategies for humancomputer interaction.** Reading, Addinson-Wesley, 3 ed. 1998.

[Silva, M.D., 2001] SILVA, DOUGLAS MARCOS DA. **Guia de consulta rápida UML.** Editora NovaTec. São Paulo – SP. 1ª Edição. 2001. ISBN 8-575-22004-7.

[Sterling, L. & Shapiro, E., 1994] STERLING, LEON & SHAPIRO, EHUD. **The art of Prolog. Advanced Programming Techiniques.** The MIT Press. Cambridge Massachusetts. London, England. 1994. ISBN 0-262-19338-8.

[Terra, E., 1992] TERRA, ERNANI. **Curso Prático de Gramática.** Editora Scipione; São Paulo 1992.

[Thro, E., 1991] THRO, ELLEN. **The artificial intelligence dictionary. The Lance. A. Leventhal Microtrend Series.** San Marcos: Microtrend, 1991.

[Zwicker, R. & Reinhard, N., 1990] ZWICKER, R. & REINHARD, N. **Interfaces Inteligentes: perspectivas para novas formas de aprendizado e uso de**

sistemas. Revista Brasileira de Computação, v. 5, n. 3, p. 17-25, Rio de Janeiro, Jan/Mar 1990.

[Wallace, M., 1984] WALLACE, MARK. **Communicating with databases in natural language.** Ellis Horwood Series. Artificial Intelligence, 1984.

[Watson, 1999] WATSON, M. **NLBean version 4: A Natural Language Interface for Databases.** Disponível em: <http://www.markwatson.com/opensource/nlbeandoc.htm>. Acessado em: Janeiro 2002.

[Weizenbaum, 1966] WEIZENBAUM, JOSEPH. **ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine.** Communications of the Association for Computing Machinery 9, 1966.

[Winston, 1992] WINSTON, P.H. **Artificial Intelligence.** Addison-Whesley, 1992.

Apêndice A

Diagramas UML

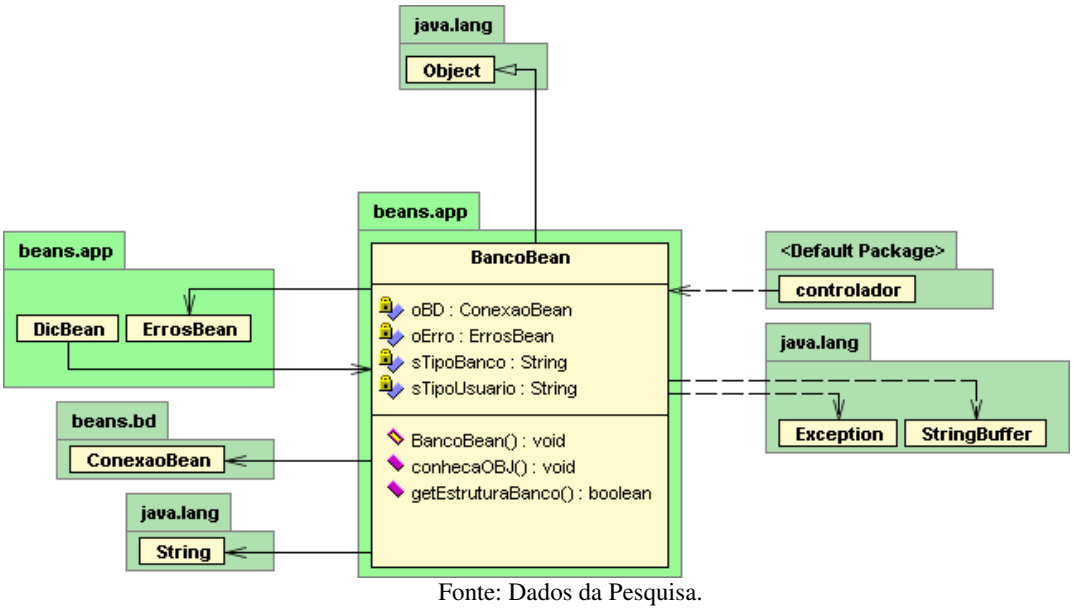
A.1. Introdução

Nesse apêndice são apresentadas as principais classes implementadas durante a o desenvolvimento do programa. Para cada classe serão apresentados sua finalidade e os métodos e atributos mais importantes.

A.1. Classe BancoBean

A Figura A.1 mostra como ficou o diagrama.

Figura A.1 - Diagrama da Classe BancoBean



Fonte: Dados da Pesquisa.

A finalidade da Classe BancoBean é fazer a interligação do sistema com um banco de dados, independente de qual seja, Oracle, Interbase, My-SQL, SQL-Server. O

principal método dessa classe é o método “getEstruturaBanco” e é dele a responsabilidade de fazer a leitura do dicionário de dados do banco definido pelo usuário. Os atributos “sTipoBanco” e “sTipoUsuário” armazenarão os dados sobre qual o banco foi escolhido pelo usuário e qual o nome do usuário do banco.

Essa classe inicialmente será usada para fazer a importação dos dados do banco do usuário para o sistema de consultas em LN. É nesse momento que o sistema conhece o nome de todas a entidades, atributos e relacionamentos do banco de dados.

A.2. Classe DicBean

Depois de feita a importação o usuário começa a definir os sinônimos para cada entidade e atributo e essa é a finalidade da classe “DicBean”. Essa classe conseguirá através dos métodos “getAtributo”, “getEntidade” e “getRelacionameto” buscar quais são as entidades, atributos e relacionamentos e armazenará também as informações dos sinônimos passados pelo usuário.

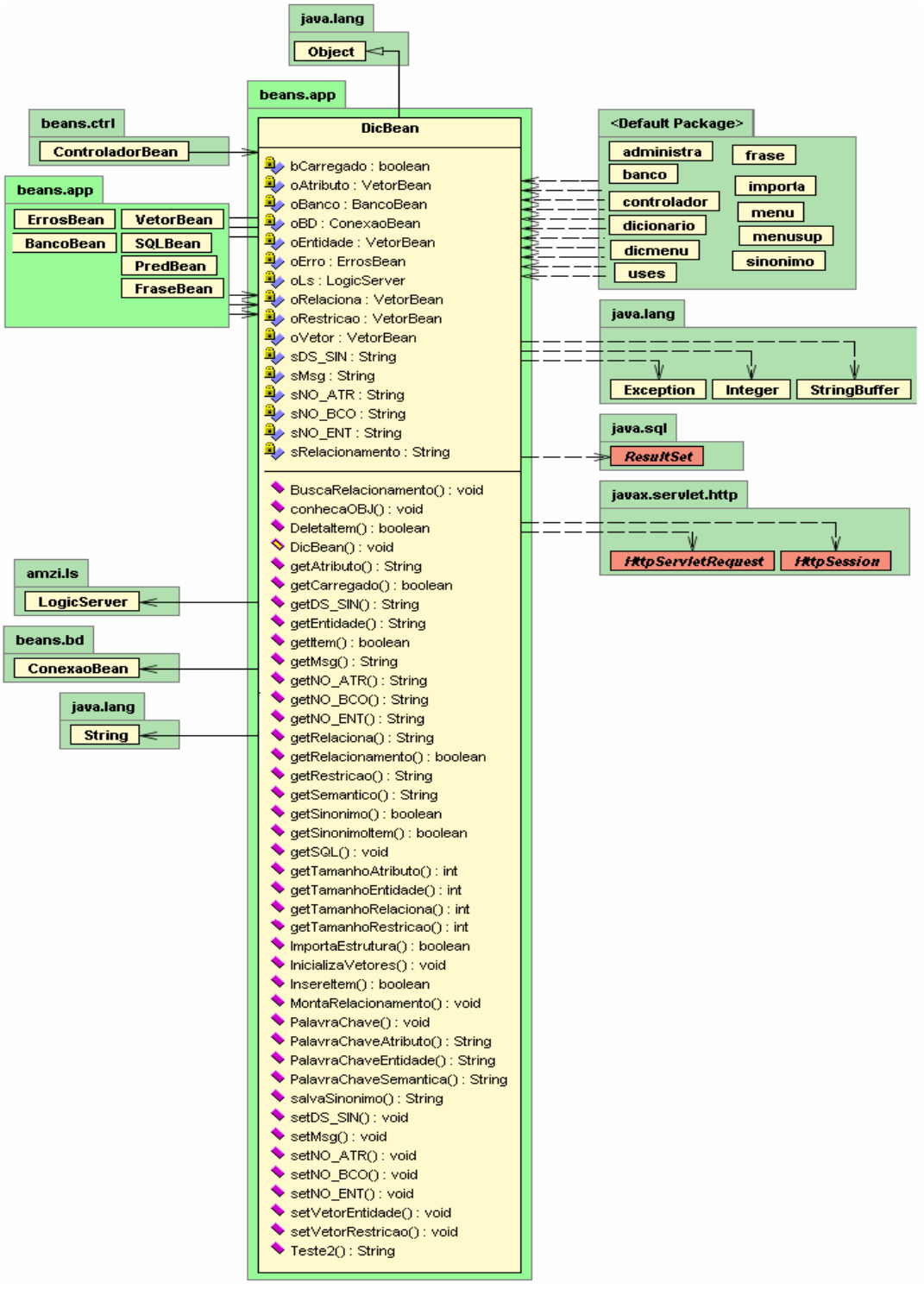
Outra finalidade desta classe é identificar as palavras-chave da pergunta em LN e relacioná-la a uma informação do banco. Isso foi implementado nos métodos cujas iniciais são identificados pelo nome “PalavraChave”.

Na fase de análise da frase a classe “DicBean” também será utilizada pois ela armazenará em estruturas do tipo vetor todas as entidade, atributos e relacionamentos identificadas.

O método “MontaRelacionamento” é que fará a interligação entre todas as entidades envolvidas em uma consultas nele foi implementado o método de busca pela melhor escolha descrita no item 5.3 (Modelo Proposto). O algoritmo foi implementado utilizando a linguagem PROLOG e essa classe fará apenas a chamada para que seja possível fazer a procura por todas a entidades do banco.

A Figura A.2 mostra como ficou o diagrama da classe.

Figura A.2 – Diagrama da Classe DicBean



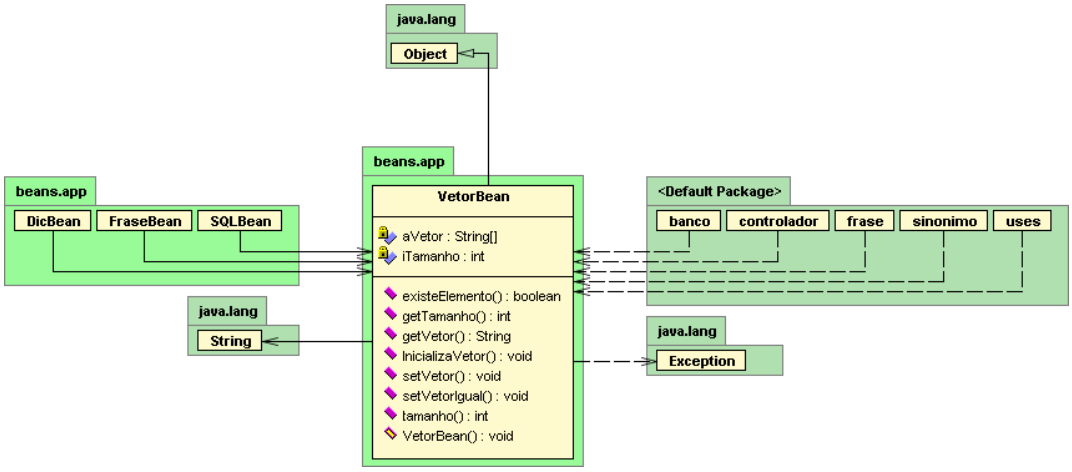
Fonte: Dados da Pesquisa.

A.3. Classe VetorBean

Como em várias partes do programas precisou-se usar estruturas do tipo vetor e os mesmos poderiam ser de tamanhos diferentes optou-se em montar uma classe para trabalhar com este tipo de dado.

A finalidade dessa classe é de fazer a inicialização de um vetor, redimensionamento, atribuição dos valores, verificação de valores existentes. Tornando assim uma tarefa genérica. A Figura A.3 mostra o diagrama.

Figura A.3 – Diagrama da Classe VetorBean



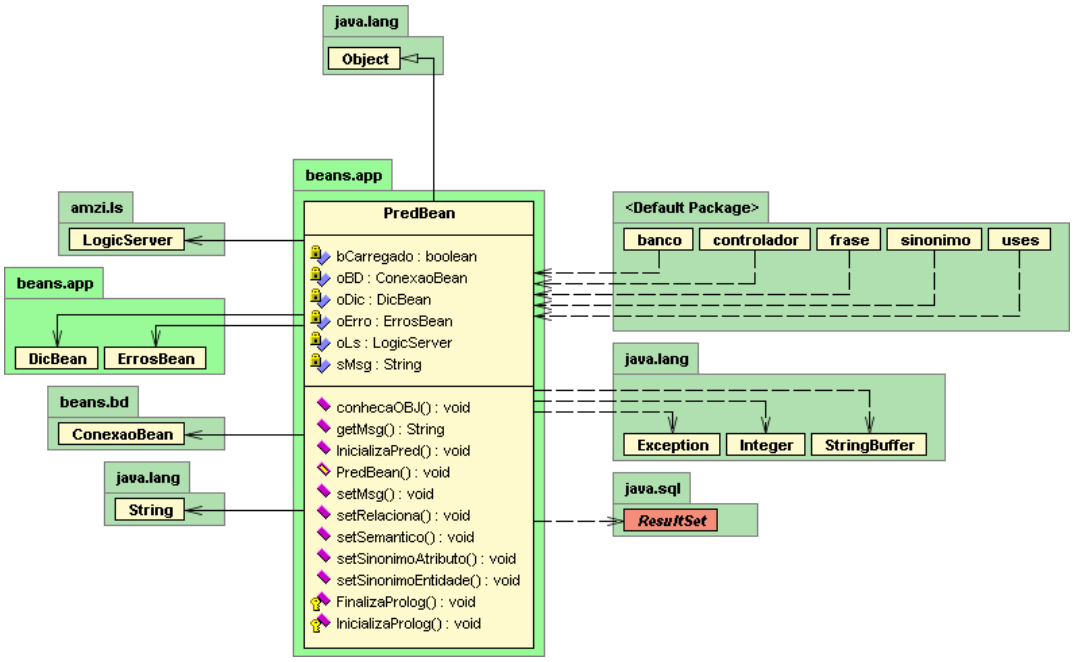
Fonte: Dados da Pesquisa.

A.4. Classe PredBean

A análise da frase é um processo que envolve entre outras funções a busca por palavras-chave, no entanto, se isso fosse feito diretamente sobre o SGBD geraria um fluxo de rede enorme, pois a cada palavra identificada na frase existiria a necessidade de se fazer uma consulta a um banco de dados. Para resolver este problema criou-se a classe “PredBean” que transformará as informações sobre o banco no formato de predicado na linguagem PROLOG evitando assim a necessidade da busca direto ao SGBD. Todo o processo será feito através de consultas a um arquivo em PROLOG o que tornou a análise uma etapa bastante rápida de ser executada.

A ligação do Java com PROLOG é feita utilizando o método “InicializaProlog”. Essa classe utilizada a classe “LogicServer”, que é da própria AMZI[17], para fazer a ligação entre as duas linguagens. A Figura A.4 mostra o diagrama.

Figura A.4 – Diagrama da Classe PredBean



Fonte: Dados da Pesquisa.

A.5. Classe FraseBean

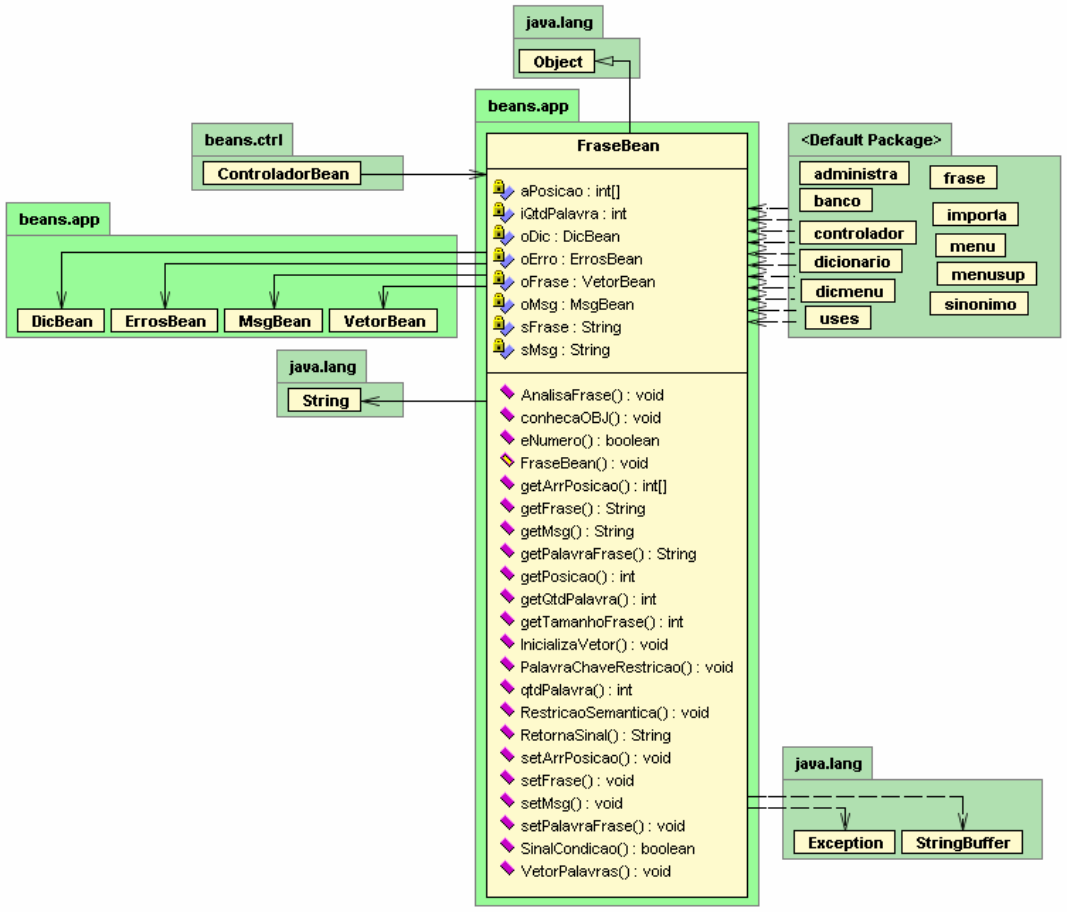
Todo processamento da frase é feito utilizando essa classe ela. Sua finalidade é separar a frase em grupos, como descrito no item 5.3.4 (Módulo de Consulta). Foram elaborados métodos para identificar se uma palavra é número “eNumero”, armazenar a posição de cada palavra “getArrPosicao”, verificar o tamanho da frase, quantidade de palavras, o tipo de sinal envolvido (maior, menor, maior igual, etc...).

As restrições do comando SELECT serão identificadas nesse momento também.

A Figura A.5 mostra o diagrama.

[17] AMZI é uma das versões disponíveis da linguagem PROLOG

Figura A.5 – Diagrama da Classe FraseBean



Fonte: Dados da Pesquisa.

A.6. Classe GramaBean

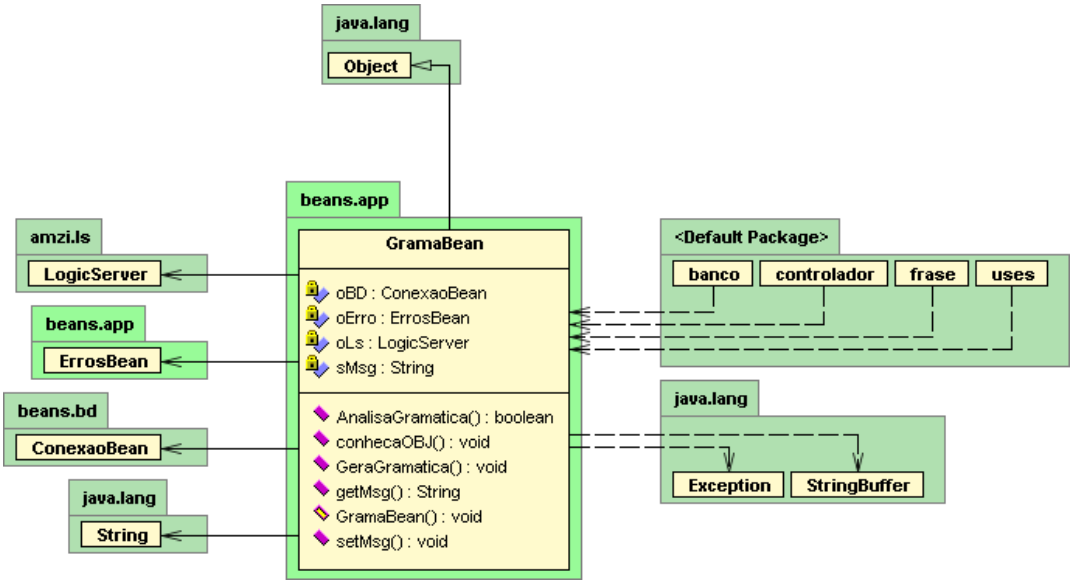
A etapa de análise da gramática foi implementada na classe “GramaBean”. Sua finalidade é classificar cada uma das palavras digitada pelo usuário e transformá-la em uma gramática válida após uma análise do UA.

O método “GeraGramatica” é que criará um DCG conforme descrito no item 2.3.2 (Gramática), os dados contidos no léxico JSpeel também é gerado por este método.

O método “AnalisaGramatica” é o responsável por verificar cada uma das palavras da frase identificar sua classificação a partir do léxico e gerar uma nova frase

gramatical que se não existir estará disponível para o UA fazer a geração dessa nova classe para o sistema para que possa analisar corretamente a frase.

Figura A.6 – Diagrama da Classe GramaBean



Fonte: Dados da Pesquisa.

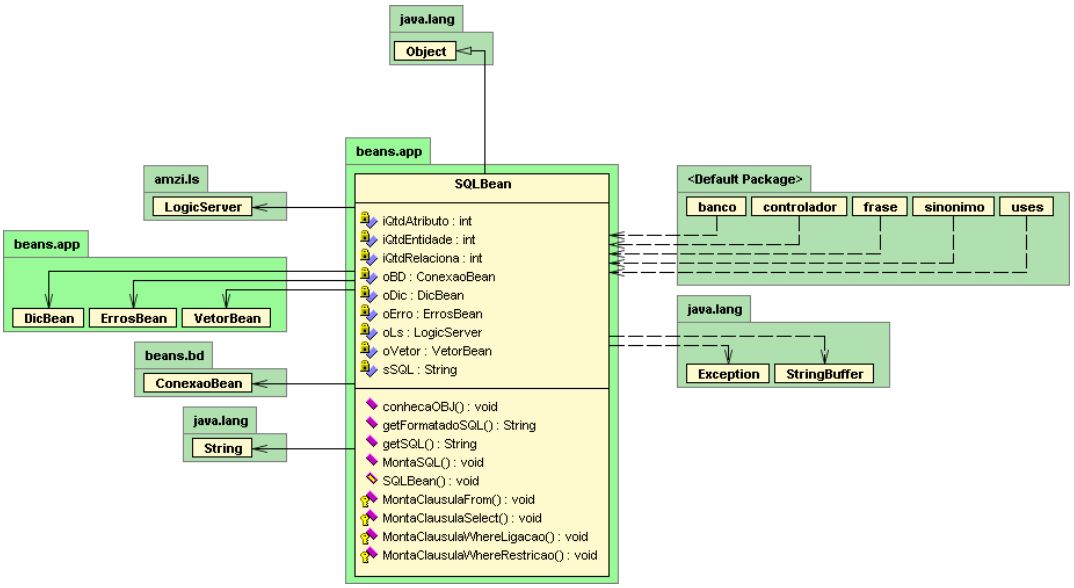
A.7. Classe SQLBean

Por fim a classe que fará a geração do código SQL do comando “SELECT”. A criação do código SQL é feita através das informações geradas pelas outras classes como pode ser visto na

Figura A.7 em pacotes padrões (“default pakages”).

A responsabilidade da execução do código SQL como o retorno da informação encontrada no banco dados para o UF também foi atribuído a essa classe.

Figura A.7 – Diagrama da Classe SQLBean



Fonte: Dados da Pesquisa.

A.8. Conclusão

O objetivo a ser implementado ficou bem mais claro após o estudo inicial e modelagem utilizando como técnica a UML. Pode-se assim ter uma visão do que se pretendia desenvolver facilitando a maneira com que as classes foram programadas.