UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM RECONHECIMENTO FONÉTICO APLICADO AO BANCO DE DADOS ORACLE

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

FABIANO JOSÉ ROSSETTO

BLUMENAU, JULHO/2000

2000/1-18

PROTÓTIPO DE UM RECONHECIMENTO FONÉTICO APLICADO AO BANCO DE DADOS ORACLE

FABIANO JOSÉ ROSSETTO

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Marcel Hugo — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Marcel Hugo

_____

Prof. Roberto Hentzle

_____

Prof. Wilson Pedro Carli

DEDICATÓRIA

Dedico este trabalho a Deus, toda minha família e meus amigos pelo amor, amizade dedicados a minha pessoa e pelo incentivo e apoio de todos na elaboração deste trabalho.

Agradeço a Deus,
por me capacitar,
ao longo da academia.

AGRADECIMENTOS

Especialmente a Deus, por ter me dado e me mantido vivo para que chegasse até este ponto me mantendo inspirado e confiante na elaboração de todo projeto.

Agradeço também a minha família, por estarem sempre presente mesmo nos momentos mais difíceis que passei, tanto neste trabalho como durante todo o período acadêmico.

A meus amigos e todos as pessoas que me apoiaram e me ajudaram para que este trabalho, seja com conselhos, críticas construtivas e sugestões.

A meu orientador Marcel Hugo, pela supervisão, críticas, companheirismo e paciência que teve para comigo no decorrer deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELA

RESUMO

Este trabalho propõem-se a elaborar um protótipo que interligado à internet possa acessar uma base de dados ORACLE e acessar os registros de uma tabela através da pesquisa fonética. O objetivo deste é diminuir a quantidade de registros que estão perdidos ou não são mais encontrados em grandes bases de dados, principalmente quando estes registros só são acessados através de chaves alfanuméricos. Para a implementação do protótipo utilizou-se o *Oracle Web Server* e PL/SQL para acessar e gerar as páginas HTML dinamicamente.

ABSTRACT

This work aims to create a prototype to access a Oracle Database and the table records over the internet using fonetic search. In addition the objective is to decrease the quantity of the lost records or records not found in a large database, hainly those that the access key is composed by names. To develop the prototype it is going to be use the Oracle Web Server tool and PL/SQL language to access and generate the HTML pages dinamically.

1 INTRODUÇÃO

Com a revolução da informática as organizações estão cada vez mais dependentes dos computadores que são vastamente utilizados para gerar informações e guardá-las em banco de dados para serem utilizadas posteriormente em beneficio da própria organização. O problema é que uma parte destas informações nunca mais são utilizadas por serem às vezes redundantes, ou por se perderem por problemas de falha humana na absorção dos dados e não ser mais possível encontrá-las em uma grande base de informações. Segundo[ALP1999], este problema afeta principalmente organizações que utilizam nomes como principal chave de acesso ou às vezes como única forma disponível de acesso para encontrar dados, como nos exemplos a seguir:

a) **Auxílio a Lista** - nas Companhias Telefônicas;
b) **Pesquisa de Antecedentes Criminais -** na Polícia;
c) **Pesquisa de Maus Pagadores** - no Serviço de Proteção ao Crédito;
d) **Emissão de Certidões Negativas** - nos Cartórios.

Quando a manipulação de dados é feita de forma manual o problema é menor pois usa-se arquivos organizados em ordem alfabética sendo bastante comum em repartições públicas, comércio, escolas, cartórios, etc. Tais arquivos, compostos por fichas, são manipulados por seres humanos que têm a capacidade de análise como uma de suas características, associada à flexibilidade de pensamento e toda uma bagagem cultural herdada e ampliada através de gerações.

Na sociedade informatizada, porém, as informações estão armazenadas em meios magnéticos e sua recuperação fica sujeita ao rigor das comparações lógicas dos computadores digitais que, embora dispondo de sofisticados circuitos eletrônicos, só possuem a pobre cultura binária dos *bits*, sem nenhuma capacidade inerente de análise[ALP1999].

Por estes e outros fatores, seria interessante obter o conhecimento e o domínio da técnica de reconhecimento de palavras através de fonemas, permitindo que dados relevantes a grandes empresas fossem rapidamente resgatados ou encontrados para serem corretamente utilizados, trazendo com isto benefícios às organizações e também aos usuários.

Atualmente existe uma tendência de mercado em se dizer que qualquer problema será resolvido caso a empresa adquira um Banco de Dados. Naturalmente, em um ambiente com acesso constante aos dados (acesso concorrente), onde a segurança seja de vital importância e o desempenho da aplicação critico à empresa, sem dúvida a aquisição de um Banco de Dados "poderá" ser o primeiro passo na solução do problema.

Na empresa Souza Cruz, existem casos onde existem dificuldades em encontrar registros que não possam ser acessados através de código, este problema vem sendo contornado com a criação de listas de valores, porém não é a solução ideal. Este é um agravante que estimula a existência de um reconhecedor fonético que consiga contribuir para a diminuição deste problema. Com este protótipo, conseguir-se-á acessar dados na base de dados através do nome. Estas consultas tornarão os sistemas muito mais eficazes e economizarão tempo, eliminando cadastros duplicados e ou não encontrados.

A empresa TEClógica Sistemas em Informática contribuirá para definição, construção e testes. O protótipo estará na mesma para demonstração à futuros sistemas que possam utilizar desta nova tecnologia.

Para especificação do projeto físico e projeto lógico do protótipo será utilizada a ferramenta **CASE** *Designer* 2000, e para o desenvolvimento do protótipo linguagem estruturada PL/SQL, juntamente com técnicas de reconhecimento de padrões.

1.1 OBJETIVOS

Este trabalho tem por objetivo principal desenvolver um protótipo cuja função é buscar na base de dados nomes, endereços, ou qualquer dado desta natureza pelo método fonético. Este método tem por principais metas :

a) reduzir a redundância de cadastros;
b) diminuir o custo de recadastramento e com isto duplo controle;
c) aumentar o retorno de dados em consultas para encontrar registros duplicados e ou encontrar registros ainda cadastrados da maneira errada.

1.2 ESTRUTURA

O trabalho está dividido nos capítulos seguintes:

O capítulo 1 apresenta a introdução do trabalho, sendo dividida em: justificativa, objetivos do trabalho e estrutura que o mesmo possui.

No capítulo 2 descreve-se todos os conceitos e identificações de banco de dados Oracle que será utilizado para buscar os dados.

O capítulo 3 trata do assunto relacionada a lingüística, fonemas, fonetização, no âmbito da computação identificando suas características bem como seu funcionamento.

No capítulo 4 são mostrados os conceitos, protocolo e arquitetura do WEB SERVER que a Oracle utiliza como servidor de Web para fazer o acesso a base de dados.

Já no capítulo 5 são contextualizados alguns conceitos da metodologia de análise estruturada de sistemas e da ferramenta CASE utilizada, e a linguagem PL/SQL que foi escolhida para o desenvolvimento do protótipo.

No capítulo 6 fala sobre o protótipo, modo de funcionamento maneira de utilização do mesmo.

No capítulo 7 conclui-se o trabalho realizado identificando as dificuldades encontradas durante a realização do mesmo e sugere sugestões para o melhoramento e aperfeiçoamento do trabalho.

2 BANCO DE DADOS ORACLE

2.1 INTRODUÇÃO

O termo banco de dados é um nome da computação que designa uma coleção de informações. Esta coleção deve ser organizada para poder servir a uma finalidade específica [MOR1995]. Ou como [AUL1995], um banco de dados é um acervo de informações armazenadas segundo certos critérios de organização.

Já para [CER1995], banco de dados relacional é uma coleção de dados organizados e integrados armazenados em forma de tabelas interligadas por meio de chaves primárias e estrangeiras. Estas constituem uma representação natural dos dados, sem imposição de restrição ou modificações. Pode ser utilizada por todas as aplicações relevantes sem duplicação de dados, e sem a necessidade de serem definidos em programas.

Segundo [AUL1995], o ORACLE é um SGBDR - Sistema de Gerenciamento de Banco de Dados Relacional ou RDBMS - *Relational DataBase Management System*, que possibilita o armazenamento de dados em tabelas (relações). Estas relações são representações bidimensionais (linhas X colunas) dos dados, onde as linhas representam os registros e as colunas (atributos) são as partes de informação contidas no registro.

O ORACLE fornece um grande conjunto de ferramentas que permitem projetar e manter o banco de dados, com as quais os usuários podem acessar a base de dados. As principais ferramentas do ORACLE são o ORACLE *Forms*, ORACLE *Report*, ORACLE *Graphics*, *SQL*, *SQL*Plus*, *SQL*DBA* e ORACLE Web SERVER entre outras.

O ORACLE é mais que apenas um conjunto de programas que facilitam o acesso aos dados, podendo ser comparado a um sistema operacional sobreposto ao sistema operacional de computador onde reside. Possui suas próprias estruturas de arquivo, de *buffer*, áreas globais e uma capacidade de se ajustar muito além das capacidades fornecidas no sistema operacional. O ORACLE controla seus próprios processos, monitora seus registros, consistências e limpa a memória ao sair[ORA1998].

O banco de dados ORACLE quando está em operação (ativo), consiste de uma série de eventos acontecendo sobre o banco de dados. Segundo [ORA1992], os principais tipos de eventos que podem ocorrer são:

a) **transações**: unidade lógica de trabalho composta por um ou mais comandos SQL, executados por um único usuário. No ORACLE, os efeitos de uma transação podem ser aplicados definitivamente ao banco de dados (*commit*) ou desfeitos (*rollback*). Uma transação começa com a execução de um comando SQL pelo usuário e termina com a execução de um *commit* aplicando os efeitos da transação ao banco de dados ou com um *rollback* que desfaz os efeitos da transação;
b) **bloqueio (*lock*)**: bloqueios são mecanismos utilizados para controlar o acesso concorrente a um determinado recurso. Sua principal finalidade é evitar que uma transação destrua os efeitos de outra transação. Os *locks* podem ser de dois tipos, os causados pelos comandos do tipo DML (*DML Locks*) e os causados por comandos do tipo DDL (*DDL Locks*). Os *locks* são utilizados para realizar duas importantes metas, a **consistência** (assegura que os dados vistos ou alterados por um usuário não sejam alterados por outro enquanto a transação não for finalizada) e a **integridade** (assegura que os dados e estruturas reflitam todas as trocas em uma seqüência correta) dos dados. Os *locks* garantem a integridade dos dados, enquanto admitem um máximo acesso concorrente por usuários ilimitados;
c) **sessões (*sessions*)**: uma sessão especifica uma única conexão ao banco de dados via processo do usuário. Quando o usuário conecta-se à base de dados, precisa informar um nome (*username*) e uma senha (*password*) válidos para que a sessão seja estabelecida. Múltiplas sessões podem ser criadas por um mesmo usuário, mas cada uma delas é considerada como sendo uma sessão diferente.

Conforme[ORA1999], o banco de dados ORACLE é um banco de dados relacional, e possui tabelas, chaves primárias, chaves estrangeiras e o principal, possui um SGBD.

2.2 SISTEMA GERENCIADOR DE BANCO DE DADOS

Segundo[SAL1993], um SGBD ou DBMS - *DataBase Management System* é uma coleção de programas e utilitários que servem para organizar, armazenar, atualizar e recuperar

dados. O objetivo principal do SGBD é capturar os dados de forma que modelem adequadamente o mundo real.

Para armazenar os dados, o SGBD tem de oferecer algum tipo de serviço de definição de dados. Precisa também de um mecanismo interno para manter os dados armazenados e saber onde está cada elemento em particular. Claro que o SGBD faz mais do que simplesmente armazenar dados, é responsável também pela introdução ou inserção, classificação, restauração, atualização ou eliminação dos dados no banco de dados.

Contudo, ter os dados armazenados e reavê-los quando necessário, não serve para nada se não houver certeza da veracidade e precisão dos dados. Para isso o SGBD precisa fornecer algum tipo de integridade de dados, garantindo que os dados não sejam corrompidos no meio exterior, como por falhas de disco ou falta de energia. O SGBD tem ainda a difícil tarefa de proteger o banco de dados contra alterações não intencionais, causadas por usuários ou por aplicativos. Essa tarefa é particularmente importante para bancos de dados multiusuários, no qual um ou mais usuários podem, ao mesmo tempo, atualizar o mesmo dado. Neste caso, o SGBD deve garantir que somente uma das alterações realmente aconteça e deverá notificar os outros usuários sobre a mudança feita.

Conforme [SAL1993], para facilitar tudo isso, o SGBD oferece os seguintes serviços:

a) **definição dos dados** : fornece um método para definir e armazenar informações;
b) **manipulação dos dados** : fornece serviços que permitem ao usuário inserir, atualizar, eliminar e classificar dados;
c) **integridade dos dados** : fornece um ou mais métodos para a precisão dos dados.

2.3 OBJETIVOS DE UM BANCO DE DADOS

De acordo com [SAL1993], os objetivos de um banco de dados são :

a) permitir que várias aplicações compartilhem dados;
b) armazenar dados;
c) eliminar redundâncias;
d) reduzir o tempo de resposta;
e) permitir a integridade dos dados;
f) ter segurança.

2.4 CARACTERÍSTICAS DE UM BANCO DE DADOS

Segundo [MOR1995], várias características evidenciam um banco de dados

a) um banco de dados é uma coleção;
b) um banco de dados usa um padrão de organização consistente;
c) um banco de dados fornece respostas a consultas sobre informações previamente selecionadas.

2.5 ESTRUTURAS DO BANCO DE DADOS ORACLE

Segundo [ORA1992], os três principais aspectos do modelo relacional, no qual o ORACLE se baseia são as estruturas, as operações e as regras de integridade.

As estruturas são os objetos que guardam os dados de um banco de dados. Essas estruturas e os dados podem ser manipulados através de operações.

As operações são as ações que permitem aos usuários manipular os dados e as estruturas de um banco de dados. Essas operações devem aderir a um conjunto de regras de integridade predefinidas.

As regras de integridade são as leis que governam quais operações são permitidas nos dados e nas estruturas de um banco de dados, com o propósito de protegê-los de operações indevidas.

Um banco de dados ORACLE possui estruturas físicas e lógicas que são independentes entre si, isto é, as estruturas físicas podem ser gerenciadas sem afetar o acesso às estruturas lógicas.

2.5.1 ESTRUTURAS FÍSICAS

Na estrutura física de um banco de dados ORACLE, existem três tipos de arquivos gerenciáveis pelo sistema operacional, sendo eles :

a) arquivos de dados;
b) arquivos de *redo log*;
c) arquivos de controle.

2.5.1.1 ARQUIVOS DE DADOS

O banco de dados ORACLE possui um ou mais arquivos de dados que são utilizados para guardar fisicamente os dados que são inseridos alterados e consultados pelos usuários. Estes arquivos formam uma única unidade lógica que são as *tablespaces*.

2.5.1.2 ARQUIVOS DE LOG

Estes arquivos são importantes para um banco de dados pois em caso de falha de execução, em que pode até se perder o banco de dados por inteiro, por uma falha de disco rígido ou mesmo por uma falha mecânica em acesso a disco, estes arquivos de log, chamados na prática como ***redo log***, são recuperados e são executados criando novamente o banco com todas as informações até a ultima transação efetuada com sucesso.

2.5.1.3 ARQUIVOS DE CONTROLE

Estes arquivos têm registros de todas as estruturas físicas que compõe um banco de dados, o nome do banco de dados, a localização dos arquivos de dados, e arquivos de ***redo log***, e ainda a última data de alteração dos mesmos.

Estes arquivos são extremamente úteis para recuperação do banco de dados, é necessário sempre ter um *backup* destes arquivos.

2.5.2 ESTRUTURAS LÓGICAS

As estruturas lógicas são os objetos (tabelas, visões, índices, *clusters*, seqüências, *stored procedures* e ligações entre bancos de dados) que estão armazenados no banco de dados e são manipulados pelo usuário. Conforme [MOR1995], as estruturas lógicas são determinadas por uma ou mais *tablespaces* e pelos objetos dos esquemas dos usuários de um banco de dados.

Dentro das *tablespaces*, são criadas tabelas onde ficaram guardados todos os dados especificados para a tabela. Para o desenvolvimento do protótipo é essencial o entendimento de uma tabela. A tabela é formada por linhas e colunas, que são acessados pelos usuários, as

tabelas possuem apenas um nome dentro de um banco de dados, assim como as colunas possuem apenas um nome e um tipo dentro de uma tabela.

Sempre que se criar uma tabela pode-se inserir dados e também consultá-los, através de *queries* (pesquisa). Pode-se também alterá-los ou mesmo apagá-los tudo isto através de comandos de SQL.

2.6 LINGUAGENS DE PROGRAMAÇÃO DE BANCO DE DADOS

Um SGBD sofisticado não faz nada a não ser que os usuários tenham acesso aos dados. Um aplicativo de banco de dados é um programa de computador que permite ao usuário introduzir, alterar, eliminar e emitir relatórios com os dados. Os aplicativos, tradicionalmente feitos por programadores, são escritos em uma ou mais linguagens genéricas ou especializadas de programação. Entretanto, há uma tendência nos últimos anos em usar ferramentas de acesso a banco de dados orientado para o usuário, as quais simplificam a utilização do SGBD e eliminam a necessidade de programação personalizada.

Segundo [SAL1993], as linguagens utilizadas para criar os aplicativos de banco de dados podem ser agrupadas em três categorias: linguagens procedurais, linguagem SQL – *Linguage Query Structure* e todas as outras linguagens.

2.6.1 LINGUAGENS PROCEDURAIS

A grande maioria das linguagens de programação pode ser descrita como **procedural**. Quando o programador cria um aplicativo de banco de dados com uma dessas linguagens, precisa escrever o código como uma série de procedimentos (*procedures*). Cada *procedure* faz o trabalho de uma parte do aplicativo, como para consultar o banco de dados ou atualizar os dados. As diversas *procedures* são então reunidas por outras *procedures* de interface do usuário (menus) e colocadas nos locais apropriados do aplicativo.

As linguagens de programação de alto nível como o Pascal, Cobol e C, são procedurais e podem ser utilizadas para criar aplicativos de banco de dados através de uma interface de programação de aplicativo. Esta interface consiste de um conjunto padrão de funções (chamadas) que estende a linguagem para lhe dar acesso aos dados do SGBD.

As funções da interface de programação de aplicativo normalmente estão contidas em bibliotecas incluídas nos aplicativos ao serem compilados. A maioria dos fornecedores de SGBD tem essas bibliotecas disponíveis como parte do pacote de SGBD ou como uma opção de custo extra[ORA1992].

2.6.2 LINGUAGENS DE CONSULTA ESTRUTURADA (SQL)

Os sistemas de banco de dados devem oferecer recursos para que os usuários possam definir e consequentemente manipular os dados armazenados. O acesso a esses dados é feito através de comandos, os quais são interpretados e executados pelo SGBD. A linguagem SQL (*Structured Query Language* ou Linguagem de Consulta Estruturada) é uma simples e poderosa linguagem de acesso ao banco, usada por muitos sistemas de gerenciamento de banco de dados relacionais.

Segundo[MOR1995], existem duas categorias para os comandos SQL:

a) os comandos DDL (*Data Definition Language* ou Linguagem de Definição de Dados) permitem a definição, manutenção e a eliminação de objetos (como tabelas, índices, etc.) de um banco de dados. Essa categoria também inclui os comandos que são usados para dar privilégios ou direitos de acesso aos objetos de um esquema específico de um banco de dados. Segundo [DAT1991], esta definição deve apenas conter definições das informações, sem levar em consideração a estrutura de armazenamento e estratégias de acesso;
b) os comandos DML (*Data Manipulation Language* ou Linguagem de Manipulação de Dados) são usados para manipular os dados de um banco de dados, como por exemplo para consultar, inserir, alterar e eliminar linhas de uma tabela. Essa categoria de comandos inclui também aqueles usados para bloquear uma tabela ou visão e examinar o plano de execução de um comando SQL.

2.6.3 OUTRAS LINGUAGENS

Este grupo inclui as linguagens que não se adaptam perfeitamente às duas categorias anteriores. As mais comuns são as linguagens de programação orientada a objeto (OOP), como C++. As linguagens OOP representam uma abordagem completamente diferente da

programação, pois as ações são definidas pelos “objetos”, ao invés de serem por uma série de *procedures*.

Segundo [SAL1993], outro tipo de linguagem usada com banco de dados é a linguagem de macro (ou *script*). Essas linguagens não são exatamente linguagens de programação, na verdade, são listagens de toques de tecla, introduzidas pelo usuário no aplicativo, a fim de automatizar determinadas tarefas. São altamente específicas a um aplicativo em particular e normalmente são encontradas em pacotes de SGBD – Sistema Gerenciador de Banco de Dados, ou nos servidores de banco de dados.

2.7 O PROCESSAMENTO DOS COMANDOS SQL

Segundo[MOR1995], o banco de dados ORACLE, como outros bancos de dados, possui a linguagem SQL para fazer acesso a seus objetos. A linguagem SQL é padronizada e o SQL que o ORACLE recebe como padrão para acessar os dados estão de acordo com o padrão ANSI – *American National Standards Institute*. Assim todas as operações que envolvem o acesso aos dados são executados a partir de comandos SQL.

Para que o ORACLE processe um comando de SQL é necessário que o mesmo execute uma série de passos que podem ser divididos em três partes:

a) fase de análise;
b) fase de execução;
c) fase de busca de dados.

Na fase de análise o ORACLE verifica a sintaxe do comando, verifica se na área de memória, na qual se encontram outros comandos já executados e que ainda estão sendo guardados, não existe nenhum comando parecido ou mesmo igual ao comando executado. Existindo, verifica então o dicionário de dados para verificar as permissões de acesso, os privilégios de segurança, e o plano de execução do comando. Caso não exista o ORACLE determina o plano de execução mais eficiente para acessar os dados. Este comando executado pode ser compartilhado com outras aplicações que porventura venham utilizar do mesmo comando. Neste caso estará diminuindo, em caso de execução de outro comando igual ao anterior , o tempo de acesso já que não passará pela fase do plano de execução.

Na fase de execução caso o comando não tenha sido encontrado na memória, o plano de execução é aplicado aos dados, tendo o ORACLE a fazer leituras e gravações físicas e lógicas. Entretanto para o comando SELECT na fase de execução ocorre o congelamento da imagem da tabela afetadas pelo comando, para efeito de consistência de leitura. Ai então o ORACLE coloca em prática o plano de execução decidido na fase de análise.

Na fase de busca ocorre literalmente o acesso a base de dados, retirando o registro encontrado, fazendo uma cópia do mesmo e retornando-o ao processo que o solicitou.

2.8 OTIMIZADOR DO ORACLE

Segundo [MOR1995], existem várias maneiras de executar um mesmo comando SQL. E quem escolhe o método de acesso é o otimizador. Ele executa diversas tarefas, entre elas :

a) avalia expressões e condições;
b) pode transformar comandos SQL em comandos equivalentes, em algumas situações;
c) decide qual o melhor caminho de acesso aos dados armazenados em um banco de dados;
d) para aqueles comandos que acessam mais de uma tabela o otimizador decide como executar a mais eficiente combinação entre os dados das múltiplas tabelas.

3 FONEMAS

3.1 INTRODUÇÃO

O termo Fonética é aplicado desde o século XIX para designar o estudo dos sons da voz humana, examinando suas propriedades físicas independentemente do seu papel lingüístico de construir as formas da língua. Já os fundamentos da Fonologia (ou Fonêmica, como preferem dizer os anglos-saxões) se estabeleceram a partir do segundo decênio do século XX, na Europa e nos Estados Unidos da América do Norte [JAK1967].

Em contraste com a Fonética, que é uma ciência da natureza e diz a respeito aos sons da voz humana, a Fonologia tem por objeto os fonemas das línguas humanas. Por isso, os especialistas afirmam que há três modos principais de descrever os sons lingüísticos. Um som pode ser descrito sob o ponto de vista:

a) da sua composição;
b) da sua distribuição;
c) da sua função.

A Fonética trata do primeiro ponto de vista (a), ao passo que a Fonologia trata dos dois outros, (b) e (c).

Embora acrescentando que não existe uma concórdia absoluta sobre a área coberta por ambas as disciplinas, [LOP1997] mostra que a fonologia poderia ser apresentada como um modo de considerar-se a fonética: ela seria a fonética tratada dos pontos de vista funcional e estrutural.

Já [FAR1988], diz que todo e qualquer som capaz de estabelecer distinção de significado entre duas palavras de uma língua recebe o nome de fonema. Portanto o fonema não pode ser confundido com uma letra, na língua escrita apenas representamos fonemas através de sinais que são chamados de letras.

3.2 MORFEMAS

O estudo da morfologia de uma língua consiste em arrolar todas as unidades significativas e indicar as normas que regem a composição destas unidades mínimas para formar as palavras[HEC1994].

Morfema é a unidade formal mínima significativa, ou seja o menor elemento divisível que tenha significado. Aplicando a definição a um exemplo, **fazer** por exemplo, este existe três unidades mínimas significativas. 1 “FAZ” (sendo a raiz, carrega o significado nocional) 2 “E” (é a Segunda conjunção), e 3 “R” (sufixo indicador do infinitivo). Estes três são unidades indivisíveis mas mesmo assim possuem significado, portanto são consideradas como morfema.

Alomorfe é uma variação do Morfema, ou seja, a forma vária, porém mantém-se o mesmo significado.

Observando a tabela 01, pode-se identificar que a raiz é comum em todas elas , ou seja possui o mesmo significado sendo apenas a forma diferente, sendo então Alomorfes.

TABELA 01: DEMONSTRAÇÃO DOS ALOMORFES

Raiz	**Morfemas**	**Alomorfes**
Faz	Fazer	Fase
	Facada	Faca
Fiz	Física	Fizica
	Fizemos	Fisemos

Fonte[HEC1994].

3.3 MORFEMAS QUANTO AO SIGNIFICADO

Os morfemas dividem-se em duas formas:

a) MORFEMAS LEXICAIS : Os lexicais são as raízes das palavras, são os que dão o significado nocional, formando uma idéia em nossa mente;

b) MORFEMAS GRAMATICAIS : Os gramaticais são os que não formam uma imagem em nossa mente, apenas indicam relações gramaticais.

Segundo [HEC1994], outros autores como André Martinet, diz que a unidade mínima significativa é o **monema** , que se divide, quanto a semântica, em lexema e morfema. Já Bernard Pottier et Alii denominam **gramema** o morfema gramatical que é também a mesma coisa do que o morfema de Martinet. Outros utilizam o termo **semantema** para o morfema lexical, e ainda existe a possibilidade de se usar os termos **morfema objetivo** para a parte léxica e **morfema subjetivo** para a parte gramatical.

A tabela 02 esclarece melhor:

TABELA 02: DIFERENTES CONCEITOS SOBRE MORFEMAS

Terminologia geral	Morfema	1. Morfema lexical
		2. Morfema gramatical
Martinet	Monema	1. lexema
		2. morfema
Pottier e outros	Monema	1. lexema
		2. gramema
Outros	Morfema	1. morfema objetivo
		2. morfema subjetivo

Fonte[HEC1994].

3.4 MORFEMAS QUANTO A FORMA

Os morfemas quanto a sua forma dividem-se em outros três subgrupos que são os Livres, Presos e Dependentes.

3.4.1 LIVRES

Os morfemas livres são aqueles que não necessitam de outros morfemas para serem utilizados na comunicação, exemplo : **luz, sol, mar, ar.**

3.4.2 PRESOS

Os morfemas presos são os que necessitam de um ou mais morfemas para que ao serem utilizados na comunicação tenham um sentido, isto é, necessitam de mais de um morfema para formar uma palavra. Este morfema pode ser utilizado com outro morfema livre, ou um morfema livre e um ou mais presos, ou por mais de um morfemas, sendo todos presos.

Todos os afixos e a maioria das raízes são morfemas presos, ou seja, todos os gramemas são presos e a maioria dos lexemas também.

3.4.3 DEPENDENTES

Os morfemas dependentes são os que dependem de outro para servir na comunicação. Ex. falou-nos, o **nos** não é livre nem preso, neste caso o **nos** é dependente da palavra ou de outros morfemas **falou** ; **os** alunos, **os** é dependente de alunos, portanto as palavras clícicas são consideradas dependente.

3.5 O PROCESSO MORFOLÓGICO

Conforme [HEC1994], o objetivo da morfologia é arrolar todas as unidades mínimas significativas existentes numa língua e indicar as normas que regem a composição destas unidades mínimas para formar a palavra.

O padrão morfológico da língua portuguesa é indicado por dois tipos de unidades mínimas, sendo as **raízes** e **afixos**. No entanto é mais fácil conhecer todos os afixos do que conhecer todas as raízes que compõe a palavra portuguesa.

3.5.1 RAÍZES

De acordo com[HEC1994], as raízes são únicas. Portanto as palavras se originam pelo acréscimo dos diversos sufixos que a contornam e dão diferentes significados, além disso os sufixos ainda possuem a função de classificarem a palavra.

A raiz é uma unidade neutra, pois através da raíz de uma palavra não se consegue classifica-la. Para um reconhecimento fonético as raízes não podem ser levadas em

consideração pois geraria um grande número de canditados, e que muitas vezes não terá a menor importância com o que foi solicitado.

3.5.2 AFIXOS

São todas aquelas unidades lingüísticas que se ligam à raiz para completar uma forma livre ou uma palavra. O papel relevante dos afixos principalmente dos propostos na raiz são que além de serem unidades significativas, são também unidades funcionais, isto é exercem uma função classificatória na formação das palavras, como é mostrado na tabela 03.

TABELA 03: ORIGEM DA PALAVRA

Palavra	**Formação (Morfema)**	**Atribuições**
Estudar	**Stud**	Vogais
Estúdio	**Stud**	Vogais + acentuação
Estudo	**Stud**	Vogais

Fonte[HEC1994].

A primeira tarefa a ser realizada para a identificação fonética seria a identificação dos morfemas das palavras, ou dos nomes apresentados , e posteriormente os gramemas já que são um número bem reduzido. Na tabela 03, a palavra **estudar** tem seu meio exatamente igual as palavras **estúdio** e **estudo.** Na língua portuguesa após agregar a uma palavra vogais ou artigos, existe a possibilidade de formar outra com significado totalmente contrário a palavra inicial, este pode ser um problema para a identificação fonética.

Segundo[HEC1994], a formação da palavra é um processo. E este processo consiste em ajuntar os morfemas até conseguir uma forma livre, reconhecida como uma palavra. Um processo qualquer segue leis e normas. O gramático deve descobri-las através de uma pesquisa profunda. Estas leis serão lingüísticas e não leis físicas e nem morais. Este processo não é e não pode ser eterno, pois a palavra sofre mutação na medida em que o tempo vai passando. Para identificação da palavra, que será utilizado para reconhecer foneticamente um dado serão verificados os processos atuais, eliminando processos passados e futuros, já que não será levado em consideração a fonética de uma palavra daqui a algumas décadas.

3.6 GERAÇÃO DA PALAVRA

A palavra conforme[HEC1994], é uma unidade autoformal e auto-sêmica sentida pelos falantes nativos, e a seqüência fonética completa a palavra para que a mesma fique dicionarizada ou gramaticalizada.

Já [FAR1988], define uma palavra como sendo distinta em três partes, sendo elas quanto a sua função, quanto a sua forma e quanto a sua distribuição. Para efeito deste trabalho analisar-se-á apenas a definição que diz respeito a sua forma.

Segundo[FAR1988], quando diz respeito a forma, a palavra é composta potencialmente por dois ou mais tagmemas, o que para [HEC1994] significa morfema, sendo um manifestado por radical e outro manifestado por outros periféricos podendo ser afixos flexionais.

3.7 ABORDAGENS FONÉTICAS

Há três modos básicos, que constituem verdadeiras escolas, de descrever os sons da língua. Estes podem ser descritos da seguinte forma:

a) do ponto de vista da sua produção pelo aparelho fonador do remetente de signos;
b) do ponto de vista dos efeitos que eles provocam no ouvido do destinatário dos signos;
c) do ponto de vista das propriedades físicas das ondas sonoras que se propagam do remetente ao destinatário.

Segundo [LOP1997], o primeiro dos modos de descrição caracteriza a Fonética Articulatória (ou motriz). Tal método é o mais usado ainda hoje e, simultaneamente ainda, é, também, o mais antigo, remontando à antigüidade indiana, com a sua exigência de extrema acuidade na produção dos sons Sânscrito.

O método **b** caracteriza a Fonética Auditiva que é, igualmente, um estudo de longa tradição, remontando aos gregos.

O método c é o mais moderno entre todos os tipos de descrição. Ele caracteriza a Fonética Acústica, que se apoia nos registros das ondas sonoras feitas por diversos tipos de aparelhos (quimógrafo, espectógrafo, e outros).

É evidente que o emprego de aparelhos apresenta uma imensa vantagem sobre a compreensão dos fatos fonéticos que se pode lograr através do ouvido humano. Alguns especialistas afirmam que 30% dos fonemas são normalmente captados por nossos ouvidos de maneira inexata.

Deste modo, a Fonética Auditiva está longe de ter a precisão das análises laboratoriais em que se baseia a Fonética Acústica. Ocorre, no entanto, que a comunicação lingüística funciona perfeitamente bem com essa taxa de exatidão, que é compensada largamente pelos processos de redundância, de elipses fonéticas e *over-lapping*, normalizadores da compreensão inter-subjetiva.

Isso significa que, ao contrário do que se poderia pensar, uma análise fonética realmente científica deve incluir o estudo desses componentes aparentemente anormais (só aparentemente) e não excluí-los do exame.

A percepção da fala eqüivale a um ato de identificação que não se faz a base de uma única dimensão, mas sim a base de vários traços distintivos, entre os quais os fatores psicológicos.

3.8 CLASSIFICAÇÃO DOS FONEMAS EM PORTUGUÊS

De acordo com a Nomenclatura Gramatical Brasileira[LOP1997], as consoantes do português se descrevem levando em conta quatro critérios, de base articulatória:

a) quanto ao modo de articulação;
b) quanto ao ponto de articulação;
c) quanto ao papel das cordas vocais;
d) quanto ao papel das cavidades bucal e nasal.

Chegando à boca, a corrente de ar que provem dos pulmões podem ser totalmente bloqueada ou parcialmente bloqueada. Resultam daí os seguintes modos de articulação, em português:

a) fonemas oclusivos, são resultantes do bloqueamento, mas sempre momentâneo, da corrente de ar, em alguma parte da boca;
b) fonemas construtivos, são resultante do efeito de atrito a que se submete a corrente de ar, cuja o percurso é parcialmente bloqueado e se desvia, por isso, pelo canal um

formado pela língua;

c) fonemas líquidos, comumente subdivididos em:
   - fonemas vibrantes, são resultantes de brevíssimos e repetidos bloqueamentos parciais da corrente de ar, provocados por movimento vibratórios da língua (ao colidir com os dentes), do véu palatino, ou da úvula;
   - fonemas laterais, são resultantes do bloqueamento parcial da corrente de ar, que se escoa pelos lados da língua;
   - fonemas nasais, resultam da passagem de parte da corrente de ar para as fossas nasais, que atuam, conjuntamente com a boca, como caixa de ressonância.

3.8.1 FONEMAS VOCÁLICOS

De acordo com[JAK1967], vogais são fonemas sonoros resultantes da livre passagem da corrente de ar para a boca ou para a boca para as fossas nasais, órgãos estes que atuam como simples caixa de ressonância (órgãos ressonadores). Três propriedades características das vogais são:

a) as vogais apresentam o maior abrimento dos órgãos articulatórios; a boca fica normalmente aberta ou entreaberta ao pronunciar uma vogal;
b) as vogais apresentam o maior número de vibrações das cordas vocais por unidade de tempo (ou seja, têm a maior freqüência);
c) as vogais são os únicos fonemas do português a integrar o centro da saída.

São considerados, normalmente, cinco características para a descrição do sistema fonológico vocálico do português. Este processo de decisão não é inteiramente relevante; a zona de articulação, por exemplo não é distintiva na pronuncia brasileira, pois não existe duas vogais que se opunham com significados diferentes quando estão juntas, e nem sendo que uma tem mais valor que outra. Resumindo é possível discriminar duas vogais de acordo com tais traços, mas eles não são senão redundantes, subsidiários.

As vogais classificam-se em diferentes classes na língua portuguesa, veja a classificação conforme tabela 04.

TABELA 04: CLASSIFICAÇÃO DOS FONEMAS (VOGAIS)

VOGAIS ORAIS				**VOGAIS NASAIS**		
	Anteriores ou Palatais	**Central**	**Posteriores ou Velares**	**Anteriores ou Palatais**	**Central**	**Posteriores ou Velares**
Altas	/ i /		/ u /	/ ~i /		/ ~u /
Médias Centrípetas	/ e /		/ o /	/ ~e /		/ õ /
Médias Centrífugas	/ ê /		/ o /			
Baixa		/a /			/ ã /	
	Não Arredondadas		**Arredondadas**	**Não arredondadas**		**Arredondadas**

Fonte[JAK1967].

3.8.2 FONEMAS CONSONANTAIS

As consoantes por sua vez dependem da localização em uma palavra para se distinguirem a forma ou o som que produz. Se a consoante encontra-se entre duas vogais está terá o sons fechado, ou seja, será considerada como um fonema. Por outro lado se a consoante encontra-se entre outras consoantes ou mesmo no final de uma palavra a mesma dificilmente formará um fonema. Como mostra a tabela 05.

TABELA 05: CLASSIFICAÇÃO DAS CONSOANTES

Consoante	**Palavra**	**Fonema Considerado**
M	Saíram	Fonema não considerado
	Camelo	Fonema considerado
R	Instrução	Fonema não considerado
	Carro	Fonema considerado
	Marcos	Fonema não considerado

Fonte[LOP1997].

Para o estudo fonético deve-se considerar que uma parcela dos dados devem ser abstraídos, por isso deve-se levar em consideração que o estudo dos sons sem valor de um fonema pode transitar para os fonemas seguintes da cadeia falada.

Conforme[LOP1997], algumas palavras em português possuem um ressôo nasal, isto ocorre em palavras que possuem consoantes iguais. Como mostra a tabela 05, a palavra **carro** possui a consoante **R**, esta por sua vez é considerada, mas não é considerada como um todo pois a continuação da pronuncia, forma o mesmo som fonético.

Os sons das consoantes na língua portuguesa também tem um classificação específica, conforme tabela 06.

TABELA 06: CLASSIFICAÇÃO DOS FONEMAS (CONSOANTES)

Critério	**Classificação**	**Consoantes**
Modo de Articulação	Oclusivas	P,B,T,D,K,Q,C(a,o,u,l,r).
	Fricativas	F,V,SS,C(e,i,y),Ç,Z,S,X,Ch,Sh,J,G(e,i,y)
	Laterais	L,LH
	Vibrantes	R
Ponto de articulação	Bilabiais	P,B,M
	Labiodentais	Z,S,T,D,SS,C(e,i,y),Ç
	Alveolares	N,L,R
	Palatais	X,Ch,Sh,Lh,Nh,J,G(e,i,y)
	Velares	K,Q,C(a,o,u,l,r)R,G(a,o,u,l,r)
Papel das vocais	Surdas	P,T,K,Q,C,SS,Ç,F,X,Ch,Sh
	Sonoras	O restante

Papel das cavidades Bucal e Nasal	Nasais	M,N,Nh
	Orais	O restante

Fonte[LOP1997].

3.9 ALFABETO FONÉTICO

O alfabeto fonético é um conjunto de símbolos utilizados para efetuar uma transição fonológica e uma transição fonética de uma língua.

Existem também, o princípio do rendimento não-proporcional, propostos por [LOP1997], dizendo que qualquer decisão no tocante ao número de símbolos que deva conter um alfabeto é, em grande parte, arbitrária.

De acordo com [NOG1989], a representação dos fonemas que está na tabela 07.

TABELA 07: ALFABETO FONÉTICO

Alfabeto	**Fonemas**	**Observação**
B	B	Sempre considerado.
C	C(a,o,u,l,r) K, Q.	Considerado C, somente a vogal posterior for a,o,u,l ou r o K e o Q sempre será considerado C.
D	D	Sempre considerado.
F	F	Sempre considerado.
G	G(a,o,u,l,r)	Considerado G somente se a vogal posterior for a,o,u,l ou r, senão é considerado J.
J	J, G(e,i,y)	A letra G é considerado J se a vogal posterior for e,i ou y.
L	L	Sempre considerado.
M	M	Sempre considerado.

N	N	Sempre considerado.
P	P	Sempre considerado.
R	R	Sempre considerado.
S	S, Z, Ç, SS, C(e,i,y)	Estas são sempre considerados S com exceção do C, que é considerado S somente se a vogal posterior for e, i ou y
T	T	Sempre considerado.
V	V, W	Sempre considerado V.
X	X, Ch, Sh	Sempre considerado X.

Fonte[NOG1989].

A associação fonética internacional adotou algumas medidas para uma melhor identificação dos fonemas, sendo elas :

a) uma medida de base que diz que cada diferença fônica perceptível deve corresponder a um único sinal gráfico. Trata-se portanto, de reproduzir sons (transcrição fonética) e não somente fonemas (transcrição fonológica);
b) A transcrição é feita num *cuntinuum.*

3.10 FONEMA E TRAÇOS DISTINTOS

De acordo com [GAS1998], Os fonemas são os resultados da segunda articulação, são uma subdivisão da saída. Concebidos não como sons, mas como simples traços sonoros agrupados em feixes, cujo valor não esta na sua uniforme realização da fala, mas na sua capacidade de caracterizar, mesmo com variações ocorrentes, uma dada forma da língua.

Isto significa que o fonema não está ligado de modo inviável e constante a um determinado som, mas sim a uma determinada forma de depressão. Os elementos da expressão apresentam um caracter discreto.

Na realidade, todo é transição na cadeia da fala, de tal modo que alguns autores a consideram uma série de alofones que devem ser filtrados em feixes de traços distintivos, ou seja, interpretados como fonemas.

Segundo [LOP1997], de modo prático pode-se dizer que são fonemas dois sons que, situados em idênticos ponto do mesmo contexto, fazem corresponder a uma diferença fonética entre eles e a uma diferença semântica qualquer entre as formas lingüisticas que eles integram.

Quando se fala em diferenças fonéticas, de qualquer modo, afirma-se que os fonemas se deixam analisar em unidades fonéticas mínimas e a esta situação denomina-se merisma, femas ou traços distintos.

Um merisma é a unidade mínima no plano da expressão. Continuando com [LOP1997], a idéia básica de traços distintivos, é que o receptor de uma mensagem, ao ouvir a onda sonora, vê-se frente a uma situação de dupla escolha e tem de eleger entre duas quantidades polares da mesma categorias sendo está grave ou aguda e compacto ou difuso, ou entre a presença ou ausência de uma determinada qualidade, sendo esta sonora ou surda e nasalizada ou não-nasalizada.

Assim, qualquer identificação de unidades fonológicas supõe uma eleição e o código é um código binário. Esta idéia esta de acordo com a teoria da informação, onde a mensagem se reduz a uma série de respostas do tipo verdadeiro/falso, tal como ocorre no código do teletipo.

Em virtude deste ato contatado, é que alguns especialistas definiram o fonema como cada particularidade fônica que se pode extrair da cadeia da fala como o menor elemento capaz de diferenciar unidades de significado.

O traço distinto cuja função é distinguir um fonema de outro será o traço distintivo pertinente do fonema considerado. Diz JAKOBSON [apud in, LOP1997] “Qualquer que seja o traço distinto que se tenha, sempre denotará a mesma coisa: que o morfema ao qual ele pertence não é igual a um morfema que tenha outro traço em seu lugar. Um fonema carece de uma pura alteridade.” A falta de denotação individual separa os traços distintivos e as suas combinações em formas de todas as demais unidades lingüísticas.

Ainda com JAKOBSON [apud in, LOP1997], é necessário evitar o equívoco de supor que o fonema encare uma significação positiva. Corresponde, única e exclusivamente, ao fato de haver distinção quanto a significação dos signos que eles integrem, sem que por isso fique determinado ou seja constante o conteúdo de tal distinção. Assim o fonema não tem um conteúdo, mas tem uma função que é a de opor-se a todos os demais fonemas do sistema a que ele pertence.

3.11 A SÍLABA

Os fonemas se agrupam em seqüência que obedecem a um padrão elementar, chamado sílaba. O termo sílaba é usado para designar um grupo unitário de fonemas. É através das sílabas que os fonemas, sob a forma de fones, para a instância de manifestação das línguas, no ato concreto da fala[MAT1979].

Ainda com [MAT1979], diz-se que do ponto de vista fisiológico, a sílaba é uma conseqüência natural dos limites rítmicos do funcionamento dos órgãos da fonação, de suas inspirações alternadas. A interrupção desse ritmo, para repouso dos órgãos, engendra as pausas na cadeia da fala. Tais pausas são utilizadas nas línguas sobretudo para exercer a função demarcadora das fronteiras entre as sílabas, os morfemas os vocábulos, os segmentos sintagmáticos das frases e entre enunciados inteiros.

3.12 O ACENTO

De acordo com [LOP1997], a definição mais precisa de acento é um realce de uma sílaba dentro de uma palavra, tomando a esta palavra como unidade acentual. Por si só o acento é incapaz de distinguir entre palavras de sentido diferentes. O mesmo individualiza apenas sílabas, operando sempre numa seqüência mínima de duas, das quais uma é **tônica** ou acentuada e a outra é **átona** ou não-acentuada.

Diferentemente também dos tons que representam um número igual ou superior a dois, o acento é único.

Ainda com [LOP1997], as regras que estabelecem o lugar do acento nas palavras, variam de língua para língua. Existem línguas, como o húngaro, em que o acento encontra-se fixado invariavelmente na primeira sílaba da palavra, ou como o francês que está sempre na

última sílaba, ou ainda o polonês que localiza-se na penúltima sílaba. Entretanto no caso do português ou mesmo do latim a acentuação é variável, e o lugar do acento é imprevisível.

3.13 PESQUISA FONÉTICA

Conforme [GAS1989], numa base de dados pode ser definida, facilmente, uma ou mais chaves de acesso com valores alfanuméricos. E uma consulta nestas bases através de valores não numéricos não é tão simples como a tradicional consulta por código. Esta consulta pode não obter sucesso por causa dos seguintes problemas:

a) erros de entrada de dados;
b) durante o cadastramento (na digitação de conteúdo da chave de acesso, o usuário pode cometer alguns erros, como troca de letras, como por exemplo S/Z, V/W, SS/Ç, maiúscula/minúscula, pulo de letras ou até palavras inteiras, entre outros);
c) campo com tamanho insuficiente obrigando o digitador a fazer abreviaturas;
d) durante a consulta, na maioria das vezes, o usuário não sabe o valor completo e exato do argumento de pesquisa.

Para solucionar os problemas acima, [GAS1989] propõe a elaboração de um algoritmo de fonetização ou pesquisa fonética. A fonetização é uma técnica baseada na análise dos fonemas de cada palavra, que visa amenizar os problemas ocasionados por erros e variações ortográficas e fonéticas, existentes entre os dados armazenados. Este algoritmo, combinado com uma boa estrutura de armazenamento de informações, pode resolver os problemas de consultas acima citados.

Hoje em dia com a evolução dos computadores é possível acessar dados a partir de um nome básico fornecido conforme tabela 08. O protótipo deste trabalho deve responder com nomes iguais ou foneticamente parecidos, juntamente com outras informações para diferenciação dos registros.

TABELA 08: EXEMPLO DE RESPOSTA

Xavier	Chaves	Rua Paranavaí	Blumenau
	Xavier	Rua XV de novembro	Blumenau
Celia	Celia	Rua Paranavaí	Campo Mourão

	Selia	Rua Curitiba	Curitiba
Vilson	Vilson	Rua Curitiba	Foz do Iguaçú
	Wilson	Rua Abel	Foz do Iguaçú

Fonte[NOG1989].

3.13.1 TRADUÇÃO FONÉTICA

Ainda com [GAS1989], e também com [NOG1989], a tradução fonética pode ser entendida como uma função que mapeia o conjunto de argumentos de pesquisa para o conjunto de códigos fonéticos gerados, a partir de uma análise dos sons produzidos pelas letras ou sílabas. Esta função não é biunívoca, ou seja, vários argumentos de pesquisa podem ser mapeados para um único código fonético (um sinônimo), como no exemplo de "CHAVES" e "XAVES", são sinônimos, foneticamente falando, mesmo sendo graficamente diferentes. Veja Tabela 09.

TABELA 09: TRADUÇÃO FONÉTICA

Palavra	**Código Fonético**
CHAVES	X0V1
XAVES	
HELIO	1L1
ELIO	

Fonte[NOG1989].

Em um processo de fonetização, nem todos os fonemas devem ser necessariamente convertidos. Uma conversão muito rigorosa pode dificultar a consulta.

3.13.2 SEPARAÇÃO SILÁBICA

A separação silábica é um processo de decomposição de uma palavra em grupos de letras consecutivas, denominados sílabas, as quais são pronunciadas em uma só expiração,

como um som ininterrupto. A separação de sílabas em computador não é tão simples, pois depende da pronúncia, da etimologia, dos elementos componentes e do contexto.

A análise dos sons de uma palavra eqüivale a análise dos sons produzidos por cada sílaba componente. Após o processo de separação, cada sílaba passa por um processo de padronização, eliminando-se as distorções ortográficas e fonéticas.

3.13.3 SEGMENTAÇÃO DA PALAVRA

O processo de separação silábica pode ser realizado através de uma análise em segmentos da palavra, formados por duas vogais consecutivas, incluindo as consoantes compreendidas entre elas. O vocabulário “pneumático” seria dividido nos segmentos “eu”, “uma”, ”ati”, ”ico”.

Comforme [GAS1998], sempre que houver separação silábica dentro de uma palavra esta ocorrerá dentro de um segmento, então a separação de sílabas de uma palavra eqüivale a separação dos segmentos que a compõe.

Pode-se classificar os segmentos quanto ao número de consoantes presentes entre as suas vogais. Segmentos sem consoantes poderão ou não conter um ponto de divisão silábica. Segmentos com uma ou mais consoantes sempre terão um ponto de divisão, dependendo apenas destas consoantes.

3.14 ESTRUTURA GERAL DOS SEGMENTOS

A estrutura geral dos segmentos segundo[LOP1997], comporta-se da seguinte maneira:

Vi(Ci ... Cn)Vf

Sendo:

Vi = vogal inicial;

Vf = vogal final;

Ci = Consoante inicial;

Cn = n-ésima consoante da esquerda para direita entre um segmento;

( ) = indica que a existência de consoantes é opcional.

Existem pelo menos cinco tipos de segmentos bem definidos e que possuem relevância para o estudo e compreensão da separação silábica.

3.14.1 SEGMENTO DO TIPO I

Estrutura : Vi Vf

Este segmento só é separado quando for um hiato. Um hiato é um encontro entre duas vogais pertencentes a sílabas diferentes. Enquanto os ditongos (2 vogais) e os tritongos (3 vogais) não são separados. Os grupos “ia”, “ei”, “io”, “ua”, “ue”, “uo”, quando átonos finais, também não são separados.

Um encontro vocálico pode ser identificado como hiato pela presença de um acento circunflexo ou agudo ou pela análise das vogais consoantes adjacentes, anteriores ou posteriores ao segmento.

3.14.2 SEGMENTO DO TIPO II

Estrutura : Vi C1 Vf.

Este segmento é composto por apenas uma consoante entre as vogais. A separação silábica, neste caso a mais simples, sempre será antes da consoante, ou seja, Vi – C1Vf como no exemplo : fa-da, ta-ba-co.

3.14.3 SEGMENTO DO TIPO III

Estrutura : Vi C1 C2 Vf

Este segmento são duas consoantes entre as vogais, porém existem exceções à regra que devem ser levadas em consideração, como exemplo da tabela 10.

TABELA 10: EXEMPLO DE SEGMENTO (TIPO III)

Condição	**Separação**	**Exemplos**
Se C2 = “H”	Vi – C1C2Vf	Ra- char; fi-lho.
Se C2 = “L” ou “R” e C1 <> “L”, “N”, “R”, “S”	Idem ao anterior	Li-vro, te-cla.
Outros casos	ViC1 – C2Vf	Al-vo; Ter-ra; Man-so

Fonte[NOG1989].

3.14.4 SEGMENTO DO TIPO IV

Estrutura: ViC1C2C3Vf

Neste segmento são três consoantes entre as vogais, e também possuem as exceções anteriores que devem ser levadas em consideração. Veja o exemplo na tabela 11.

TABELA 11: EXEMPLO DE SEGMENTO (TIPO IV)

Condição	**Separação**	**Exemplos**
C3 = “H”, “L” ou “R”	CiC1- C2C3Vf	Mês-cla; En-cher;
Outros casos	CiC1C2 – C3Vf	Abs-ces-so; obs-tá-cu-lo;

Fonte[NOG1989].

3.14.5 SEGMENTO DO TIPO V

Estrutura : ViC1C2C3C4Vf

Esta separação também é considerada simples pois na teoria não possui exceção. A separação silábica neste caso será feita sempre da seguinte forma : ViC1C2 – C3C4Vf.

Como no exemplo : subs-cre-ver; ins-tru-ção.

3.15 GERAÇÃO DO CÓDIGO FONÉTICO

Conforme [NOG1989], considera-se a fonetização de nomes com várias palavras (pessoas, produtos, endereços, etc.). estabelecendo os seguintes conceitos:

a) Nomes : nome completo que identifica a entidade. Exemplo: Luiz Carlos da Silva, Geladeira Cônsul Super Luxo, etc.;
b) Palavras : grupos de caracteres contínuos delimitado por um espaço em branco. Exemplo : Carlos, Super, Cônsul.

Para a geração do código fonético[NOG1989], define os seguintes passos:

a) efetuar a separação silábica de cada palavra;
b) as preposições “de”, “da”, “dos”, “das”, e a conjunção “e” devem ser eliminadas para facilitar a consulta;
c) abreviaturas muito curtas que geram apenas uma sílaba, também podem ser eliminadas;
d) para cada sílaba de cada palavra deve-se proceder da seguinte forma:
    - converter as vogais : Se as vogais forem desprezadas ocorrerá um grande número de códigos fonéticos gerados que serão iguais, porém a fala da palavra escrita não terá o menor sentido. Por outro lado se as vogais forem sempre consideradas pode ocorrer que todas as palavras parecidas foneticamente não sejam encontradas por diminuir consideravelmente a geração dos códigos fonéticos iguais;
    - converter as consoantes : Deve-se converter as consoantes iniciais de cada sílaba responsáveis pela formação dos sons, ignorando a letra “h” no início, as consoantes responsáveis apenas em formar os sons alveolares (“L” em “cla” e “tal”), vibrantes (“R” em “pra” e “par”), nasais (“M” em “som” e “N” em “senso”) e sibilantes (“S” em “nascer”).

Conforme[NOG1989], a melhor opção a ser implementada para uma rotina de reconhecimento fonético será:

a) localiza-se a penúltima sílaba nas palavras não monossílabas e a última sílaba nas monossílabas. Seleciona-se a vogal de menor classe, conforme tabela 12, sendo considerada a que mais se destaca;
b) em palavras que começam com “H”, ou vogal , considerar a vogal inicial;
c) Usar o critério da região de articulação conforme tabela 12.

TABELA 12 : CÓDIGO FONÉTICO DAS VOGAIS

Classe	**Classificação**	**Vogais**
0	Centrais	A
1	Anteriores	E , I , Y
2	Posteriores	O , U

Fonte[NOG1989].

Usar o critério do modo de articulação, papel das cordas vocais e papel das cavidades bucal e nasal conforme tabela 13.

TABELA 13 : MODO DE ARTICULAÇÃO DOS FONEMAS

CL	Fonemas	Modo Articulação	Ponto articulação	Cordas	Cavidades
B	B	Oclusivas	Bilabiais	Sonoras	Nasais
C	C(a,o,u,l,r) K, Q.	Oclusivas	Velares	Surdas	
D	D	Oclusivas	Linguodentais	Sonoras	
F	F	Fricativas	Labiodentais	Surdas	
G	G(a,o,u,l,r)	Oclusivas	Velares	Sonoras	
J	J, G(e,i,y)	Fricativas	Palatais	Sonoras	
L	L	Laterais	Alveolares		
M	M	Oclusivas	Bilabiais		Nasais
N	N	Oclusivas	Alveolares		Nasais

P	P	Oclusivas	Bilabiais	Surdas	
R	R	Laterais	Velares		
S	S, Z, Ç, SS, C(e,i,y)	Fricativas	Labiodentais		
T	T	Oclusivas	Linguodentais	Surdas	
V	V, W	Fricativas	Labiodentais	Sonoras	
X	X, Ch, Sh	Fricativas	Palatais	Surdas	

Fonte[LOP1997].

3.16 ROTINA DE ACESSO

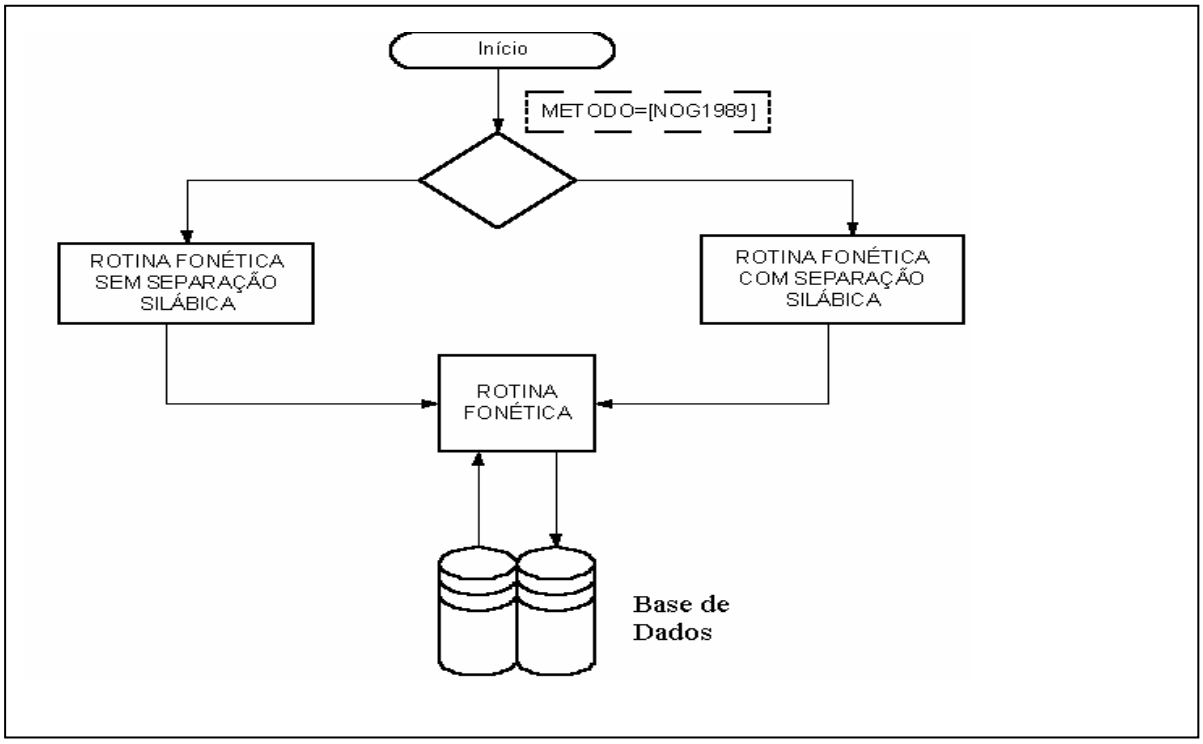
De acordo com[NOG1989], para fazer uma rotina de acesso é preciso:

a) traduzir o nome ou palavra informada par um código fonético;

b) através do código gerado buscar no índice correspondente as chaves candidatas e para cada chave candidata acessar o cadastro, pegar as informações necessárias e mostrar para o usuário fazer sua escolha.

A rotina de acesso fica melhor representada de acordo com a figura 01.

FIGURA 01: ROTINA DE ACESSO



4 WEB SERVER

4.1 INTRODUÇÃO

Este capítulo trata sobre conceitos e arquitetura que a ORACLE utiliza para gerenciar as aplicações na Internet.

Conforme[BRO1998], O *Oracle WebServer* possui uma arquitetura aberta, ideal para desenvolver produtos e negócios agrupando tecnologia de Internet, intranet ou *extranet* com acesso a base de dados Oracle. Sua arquitetura é forte o suficiente para apoiar aplicações ou negócios considerados críticos, tanto por complexidade de operações quanto ao tamanho, ou mesmo quantidade de acessos aos dados.

Ainda com [BRO1998], O OWS - *Oracle Web Server* proporciona mais funcionalidade do que um servidor de WEB tradicional. Pois é integrado com a base de dados ORACLE, permitindo que um procedimento armazenado de PL/SQL – Linguagem de Programação, gere páginas de HTML. Esta facilidade faz com que se consiga criar páginas em HTML dinâmicas, baseadas na entrada de dados, e dependendo de como os dados deverão ser tratados, gera-se uma nova página dinamicamente.

Para trabalhar com o OWS, é preciso ter uma ferramenta para configuração e administração, esta ferramenta que a ORACLE tem disponível é a OAS - *Oracle Applications Server*, que possui interface HTML, interface *applet* java para que o administrador possa trabalhar remotamente como localmente. O OAS é separado em duas partes:

a) OAS *Administrador*;
b) OAS *Utilities*.

A parte do administrador é acessada somente com conta e senha de um DBA – *Database Administrator*, isto porque a partir do administrador é que se torna possível inicializar ou finalizar o OAS *utilities*. Também é utilizado em tarefas administrativas como instalar o banco de dados e rodar *packages* para uma nova aplicação por exemplo.

A parte de utilitários é a parte onde os usuários utilizam sistemas, ou seja, é a parte que liga o usuário através de uma conexão HTTP – *Hipper Text Transfer Protocol*, de

qualquer browser ao OAS. Esta conexão pode solicitar uma página estática ou uma página dinâmica. Se a página solicitada for uma página dinâmica, o OAS cria está através de uma *stored procedure* e executa-a retornando a página que a *procedure* de banco chamada gera em HTML. Já no caso da solicitação feita pelo usuário ser uma página estática o OAS busca a página conforme endereço que o protocolo HTTP forneceu, que na verdade está em um diretório configurado no servidor.

O OWS possui um ambiente com características favoráveis sendo:

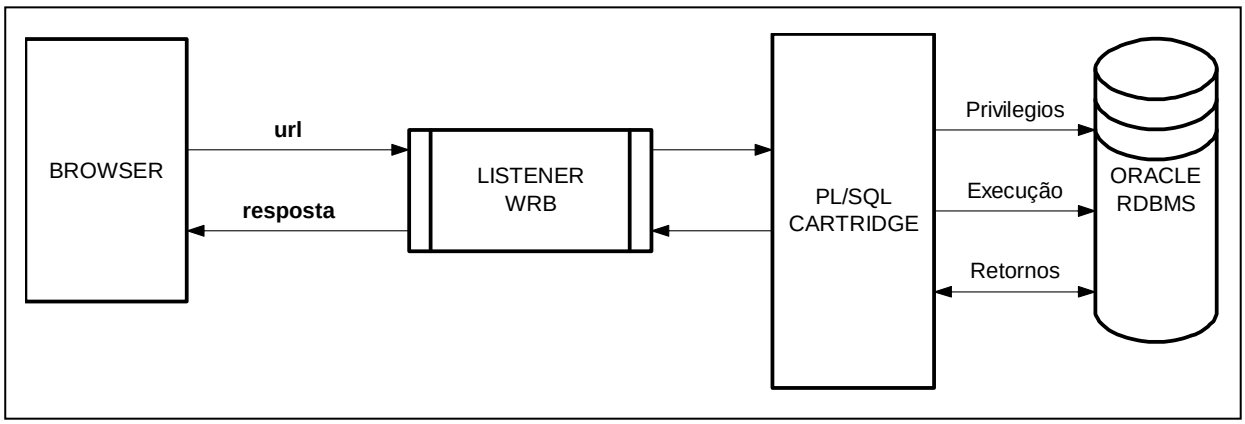
a) escalabilidade de empreendimento;
b) confiabilidade incomparável;
c) pode ser programado em vários idiomas;
d) plataforma portátil.

4.2 ARQUITETURA DO WEBSERVER

O *Oracle WebServer* é mais complicado do que os servidores da *Web* comuns, mas também é mais poderoso. Para funcionar corretamente, envolve-se com o servidor vários objetos e o servidor comunica-se entre eles, para manter as páginas e as solicitações de usuários sempre disponível. A arquitetura *WebServer* é constituído pelos seguintes componentes, que também são encontrados na figura 02.

a) *Listener* da *Web*;
b) CGI - *Common Gateway Interface*;
c) Interface WRB - *Web Request Broker* ;
d) Despachante WRB;
e) Agentes PL/SQL.

FIGURA 02: ARQUITETURA WEB SERVER



Fonte[BRO1998].

4.2.1 LISTENER DA WEB

O objeto chamado de *Listener* é responsável pela recepção de uma URL - *Uniform Resource Locator* de um *browser* denominado de Cliente, e o *Listener* também fica responsável em responder determinados dados para o Cliente conforme solicitação feita pelo mesmo. Sendo assim todas as solicitações feitas por usuários e todas as respostas recebidas, sendo páginas estáticas ou dinâmicas são recebidas e enviadas através do *Listener*.

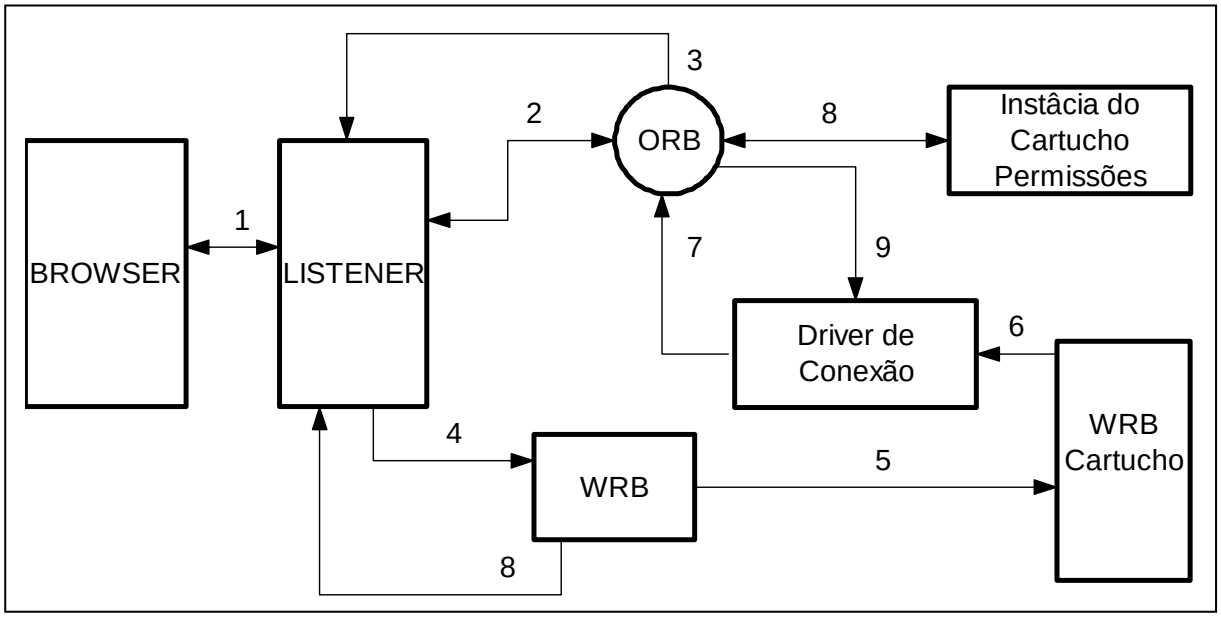
4.2.2 COMMON GATEWAY INTERFACE

A finalidade de um servidor da Web é responder aos pedidos de um cliente. Normalmente, as respostas são páginas HTML estáticas. O CGI proporciona uma interface com um programa residente no servidor que processa os dados recebidos através desta interface e tem como saída respostas em HTML ou páginas dinâmicas.

4.2.3 INTERFACE WEB REQUEST BROKER

A WRB é uma interface adicional que ativa a chamada de um programa executável pelo *Listener*. Quando o *Listener* identifica a necessidade de executar a interface WRB, delega o processamento para o despachante WRB (tópico 4.2.4), e retorna a processar pedidos de outros clientes. A interface WRB é específica do Oracle WebServer, enquanto que o protocolo CGI é um componente de todos os servidores WEB. Veja figura 03.

FIGURA 03: INTERFACE WEB REQUEST BROKER



Fonte[BRO1998].

4.2.4 DESPACHANTE WRB

O despachante é o encarregado de acompanhar os pedidos de WRB para posterior resposta. Esta é executada a partir de um conjunto de processos designados como *threds*, ou também chamada de WRBX. O WRBX comunica-se com uma aplicação em *background* chamada também de *Cartdridges* (Cartuchos). Um cartucho é uma aplicação especializada em implementar uma tarefa, que comunica-se com o WRBX, e também pode comunicar-se com o cliente através do *Listener*, utilizando uma API – *Applications Program Interface* de WRB aberta, ou seja que o *Listener* enxerga. O Despachante também é uma novidade no *Oracle WebServer*.

4.2.5 AGENTE PL/SQL

O agente PL/SQL é a referência final no processo de comunicação entre o cliente (navegador) e o servidor Oracle (WebServer). O agente é o encarregado de executar uma chamada eventual de um procedimento que fica armazenado no banco de dados ORACLE. Este procedimento cria uma página dinâmica de HTML como saída de dados e o agente envia está página dinâmica para o cliente utilizando o *Listener*.

Conforme [ORA1999], o agente é o principal meio de acesso que o Servidor WebServer tem para chegar a base de dados. Pois quando o *Listener* recebe uma URL e está URL está referenciando-se a um agente PL/SQL, o *Listener* chamará o agente e este fará uma nova ligação com a base de dados executando o procedimento contido no parâmetro da URL.

O procedimento a ser executado pode por sua vez também receber parâmetros. Estes parâmetros virão junto com a URL. Após o processamento da função é devolvido ao cliente uma página em HTML, que é gerada a partir de pacotes específicos que também são fornecidos com o servidor WebServer.

4.3 DESCRITOR DA LIGAÇÃO À BASE DE DADOS

Ao chamar um procedimento contido na base de dados, o agente PL/SQL necessita estabelecer uma conexão com a base, para se saber qual o nome do usuário, senha, nome do banco, por questões de segurança, entre outros. E estas informações estão contidas no DCD - Descritor da Ligação à Base de Dados. O que possui dentro de um DCD está descrito conforme tabela 14.

TABELA 14: DEMONSTRAÇÃO DO DESCRITOR DE DADOS

Nome do Campo	**Significado**
USUÁRIO	O nome do usuário. É necessário que os procedimentos relacionados com a chamada possam ser executados por este usuário.
SENHA	Senha do Usuário
ORACLE_HOME	Diretório onde o ORACLE está instalado no Sistema Operacional.
ORACLE_SID	Nome da base de dados onde deve-se conectar para acessar os procedimentos.
OWA_ERROR_PAGE	Local onde está uma página de erro, estática, que serve para que o servidor devolva para o usuário caso encontre algum erro do sistema operacional ou mesmo do banco de dados.
OWA_VALID_PORTS	Contém uma lista de portas válidas nas quais o listener

	responderá.
OWA_LOG_DIR	Diretório no servidor onde o agente PL/SQL escreverá, todos seus passos. Este server para o administrador diagnosticar erros que porventura venham a ocorrer.

Fonte[BRO1998].

Estes parâmetros contidos no DCD podem ser administrados pelo administrador a partir da página de administração do servidor *WebServer*. Esta página é criada quando é instalado o servidor, permitindo assim criar ou modificar o conteúdo do DCD.

4.4 CGI E WRB

Os agentes de PL/SQL podem ser chamados tanto a partir de interface CGI , como também através dos WRB. Em qualquer dos casos, o DCD é utilizado pois é necessário indicar as informações da ligação à base de dados. Então a dúvida qual das duas é a melhor opção?

Segundo[ORA1999], a interface WRB utiliza a WRBX - *Web Request Broker Extend* que é uma extensão do WRB e serve apenas para criar manter e cuidar de uma conexão com a base de dados do Oracle, e funciona em duas fases:

a) é estabelecida a ligação em si. A WRBX, ou seja, logo que é criada;
b) para cada pedido, a WRBX entrará em sessão na base de dados. Quando o pedido estiver concluído, a WRBX saíra da sessão, mas a ligação permanece ativa.

Ainda com [ORA1999], no caso da ligação ser feita através da interface CGI, o agente PL/SQL tem de concluir ambas as fases da ligação para cada pedido, o que é necessário porque cada pedido que é feito através da interface CGI, gera um novo processo, e para cada processo deve-se utilizar todos os passos.

Concluindo, sendo que a interface WRB, fica permanentemente conectada e que para cada chamada que recebe só necessita executar o segundo passo, e ainda levando em consideração que o primeiro passo é mais demorado, a utilização do WRB torna os pedidos mais rápidos e por isso é recomendado.

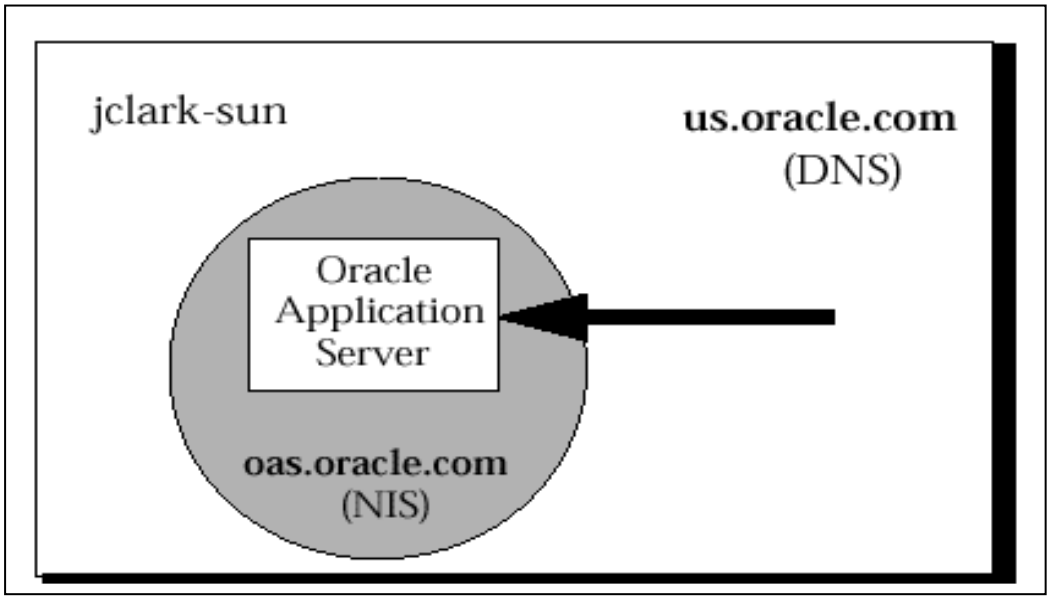
4.5 LOCALIZAÇÃO DOS OBJETOS DO BANCO DE DADOS

O ORACLE WebServer possui um serviço que tem a finalidade de procurar os nomes de objetos, diretórios virtuais, locais de imagens, entre outros. Este serviço é uma das funções da ORB, e verificar se o cliente tem acesso, ou se suas solicitações realmente existam mesmo no servidor. O ORB utiliza CORBA para comunicação entre seus vários processos, sendo que estes processos localizam, identificam e retornam ao solicitante afirmativo (*True*) ou negativo (*False*), no caso de êxito ou falha na procura.

Estes mesmo objeto também tem o poder de verificar se existe disponibilidade do servidor em responder a processo, no caso de existirem muitas solicitações feitas e o consumo de CPU do servidor estiver relativamente alto. O ORB pode negar-se a receber o pedido, retornando ao cliente o que contem no DCD, no campo relativo ao erro, configurado pelo administrador.

Quando o cliente tem acesso aos objetos o ORB retorna verdadeiro para o despachante que por sua vez executa as aplicações e retorna para o *Listener* o resultado da solicitação do cliente. Conforme figura 04, o localizador, identifica o OAS de acordo com o DNS informado pelo cliente.

FIGURA 04: ACESSO A BASE ATRAVÉS DO DNS

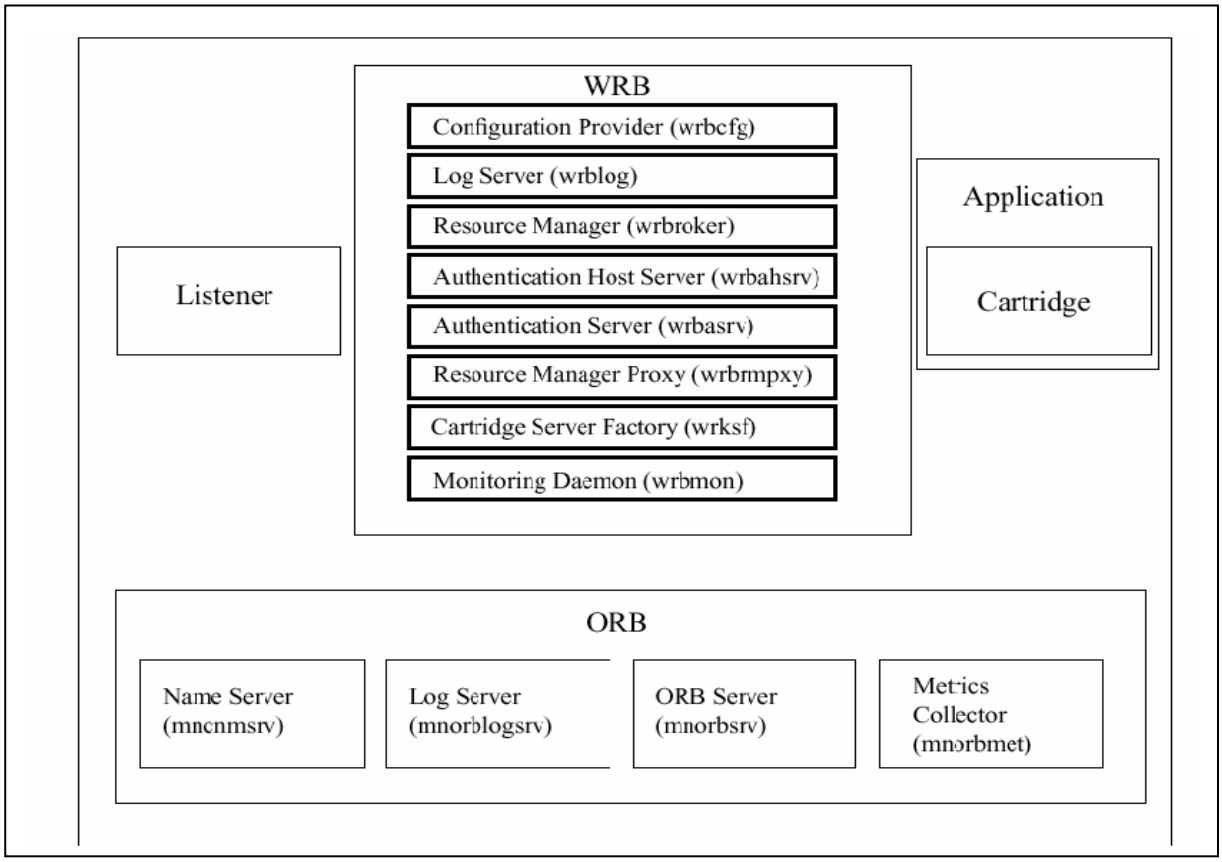


Fonte[ORA1998].

Ainda a respeito da figura 04**,** o *jclark-sun* pertence a dois domínios diferentes sendo eles tanto da oas.oracle.com (NIS) como também us.oracle.com(DNS). Quando o servidor é instalado em um ponto da rede, este recebe os valores do domínio de NIS e configura todos os arquivos necessários, ou seja, só conterá objetos que não contenham no outro domínio, isto porque o ORACLE WebServer possui a capacidade de distribuir seus objetos, tanto em domínios diferentes como em servidores diferentes[HUG1996].

As funções não se alteram e se o cliente faz uma solicitação, por exemplo, na rede de domínio NIS, mas os objetos necessários para a consulta estão em outro domínio o DNS, então o servidor fará os mesmos passos que anteriormente, porém pode-se verificar que os objetos podem não estar neste domínio, mas sabem que poderão estar em outro. Sendo assim, o retorno só será completo ao passar por todos os domínios cadastrados. Para melhor visualização veja figura 05.

FIGURA 05: DISTRIBUIÇÃO DOS OBJETOS DO ORACLE WEB SERVER



Fonte[ORA1998].

4.6 SEGURANÇA DO WEB SERVER

Freqüentemente, é desejável restringir acesso para certos recursos da rede ou para certas aplicações. Um exemplo desta necessidade é quando um servidor permite a seus clientes ter acesso a aplicações desde protótipo, trabalhos em desenvolvimento através de um *firewall* – Conjunto de hardware e software, com o objetivo de restringir acessos remotos ou mesmo para controlar trafego de rede. Para manter a confidência desta informação, o servidor deve fazer várias checagens sendo que uma delas é configurar seu servidor de rede para restringir acesso para recursos específicos.

A maioria dos servidores de rede está baseada em categorizar recursos em coleções de reinos. Um reino é uma coleção de recursos que têm segurança semelhante ou acesso/controle às exigências. Certos grupos podem ter acesso aceito ou acesso negado para certos reinos de dados. De acordo com [HUG1996], um grupo é uma coleção de usuários de um servidor. Existem também controles de acesso de nível físico significando que a permissão pode ser concedida ou recebida baseado na identidade da máquina que é conhecida costumeiramente como endereço IP, que faz o pedido, em vez de restringir através do usuário da máquina.

Ainda com [HUG1996], em uma transação HTTP, o servidor de rede compara o URL – *Universal Resource Location* pedido contra as listas de grupos protegidos que existem em cada servidor de recursos. Se o URL é protegido por segurança baseada no endereçamento IP, o servidor de rede confere o IP solicitante contra os grupos de IP considerados aprovados. Se o endereço IP solicitante não é autorizado, o servidor devolve uma mensagem de erro, caso contrário se a máquina do cliente pedido é autenticada corretamente ou se nenhuma segurança de endereço de cadeia é aplicada ao recurso, a segunda parte de controle de acesso é examinada.

Se o recurso de pedido é um sócio de um grupo que obriga controle de acesso, o servidor examina o cabeçalho de pedido para uma entrada de autorização. O cabeçalho presente, contém uma ou mais camadas de segurança, sendo, *username*, senha, e contra-senhas quando associado o anfitrião para o qual o pedido foi enviado. Se nenhum cabeçalho de apresentação está presente ou se um cabeçalho não fosse incluído do reino do qual o recurso presente é um sócio, o servidor devolve uma mensagem de erro dizendo que o pacote recebido para validação não possui cabeçalho, o servidor no caso solicita uma nova submissão

com as credenciais corretas. Para o cliente, isto resulta costumeiramente em uma caixa de diálogo que pede *username* e senha para o grupo.

Ao receber novamente o pedido com um *username* e senha para o grupo correto, o servidor examina o *username* e senha, em caso afirmativo o servidor então verifica o grupo que foi concedido acesso ao pedido. Se estes testes falham, retornam a mesma mensagem solicitando novamente a digitação de um *username* e senha. Se estes testes têm êxito, o servidor pode continuar com seu processo do pedido.

5 METODOLOGIA

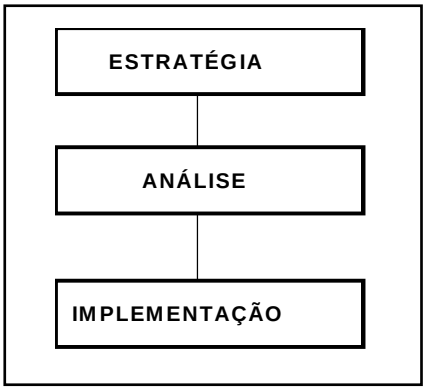
5.1 FASES DA METODOLOGIA

Geralmente a metodologia utilizada para desenvolvimento de sistemas gira em torno de análise estruturada, análise orientada a eventos, orientação a objetos. Estas ferramentas de metodologias ajudam o desenvolvedor a manter um padrão. Porém no desenvolvimento de uma rotina estes padrões não se encaixam, já que uma rotina possui características que diferem dos sistemas tradicionais. Tal fato, fez com que fosse adotada uma estratégia diferente, baseando-se na metodologia que a empresa Souza Cruz utiliza no desenvolvimento de rotinas específicas, como neste caso.

Normalmente, o processo de desenvolvimento de sistemas segue uma série de etapas e procedimentos. Uma série de métodos podem ser aplicadas a sistemas tradicionais ou sistemas de informação, porém, em certos casos estes métodos não se adaptam adequadamente.

A metodologia utilizada para o desenvolvimento do protótipo teve como base a Analise Estruturada mas com algumas modificações. As fases que compõem a metodologia utilizada para desenvolvimento são estratégia, análise e implementação, conforme mostra a figura 06.

FIGURA 06: ESTÁGIOS DO DESENVOLVIMENTO DO PROTÓTIPO



5.1.1 ESTRATÉGIA

Nesta fase foram definidos os objetivos e metas a serem alcançadas pela ferramenta. Também foram feitos vários estudos a respeito das palavras e seus padrões e principalmente as exceções. Isto foi fundamental para a determinação dos objetivos e das principais

atividades a serem realizadas pelo protótipo. Baseado nestas atividades, foi definida qual algoritmo seria utilizado.

Para o protótipo é de fundamental importância a rapidez e agilidade das respostas, porém, com os recursos convencionais isto nem sempre torna-se possível. Portanto notou-se a necessidade de uma maneira que agilizasse estas consultas, através da utilização dos recursos de internet e acesso a base de dados e com a ajuda das ferramentas de desenvolvimento que a própria ORACLE disponibiliza.

5.1.2 ANÁLISE

Nesta fase é feita a análise separadamente de cada item da área de atuação do protótipo, ou seja a maneira pelo qual o mesmo deve ser desenvolvido para prover uma maior portabilidade para outros bancos de dados sem necessidade de grandes alterações em sistemas que ainda não possuem está facilidade.

O banco de dados ORACLE possui uma função que na teoria deveria ter a mesma função de reconhecimento fonético. A função SOUNDEX, que gera um código parecido com o objetivo do trabalho, não se encaixa com as necessidades que a língua portuguesa exige, pois o código fonético gerado pela função é muito parecido em várias palavras escritas e que possuíam fonemas diferenciados.

A função SOUNDEX é uma função do banco de dados *Oracle* que recebe um parâmetro de entrada, que é a palavra, e retorna um código que é gerado a partir da análise deste parâmetro de entrada.

5.1.3 IMPLEMENTAÇÃO

Esta fase está melhor descrita no capitulo 06 que trata do desenvolvimento do protótipo.

5.2 A LINGUAGEM UTILIZADA

Segundo [FEV1995], a linguagem PL/SQL é uma linguagem baseada em blocos, e estes blocos por sua vez, podem ser seqüenciais, ou seja, um abaixo do outro, ou podem ser encadeados, sendo um dentro do outro. Existem outros tipo diferentes de blocos:

a) Blocos anônimos : são geralmente construídos de forma dinâmica e executados apenas uma vez;
b) Blocos nomeados : são os blocos anônimos, mas com uma identificação que fornece o nome ao bloco. Geralmente, também são construídos de forma dinâmica e são executados apenas uma vez;
c) Subprogramas : são procedimentos, pacotes, funções que são armazenados na base de dados. Estes blocos, geralmente, não são alterados depois de compilados e são executados muitas vezes. Este tipo de bloco será utilizado pelo protótipo para o reconhecimento de fonemas;
d) *Triggers* : são blocos nomeados que também são armazenados na base de dados. Também estes não são alterados depois de compilados e são executados muitas vezes. A diferença que é uma *Trigger* é executada implicitamente sempre que ocorre um evento a qual a *Trigger* está associada.

5.2.1 MODELO CLIENTE-SERVIDOR

Muitas aplicações que utilizam base de dados são especificadas no modelo cliente servidor. O funcionamento ocorre basicamente da seguinte maneira: o programa propriamente dito reside no cliente e envia pedidos de informações a um servidor que possui a base de dados. Estes pedidos resultam em muitas viagens na rede, sendo uma viagem a cada instrução de SQL que necessita. Um modo de melhorar o desempenho do programa e diminuir o trafego na rede é agrupar os comandos em blocos de PL/SQL e enviar para o servidor uma única vez, o resultado é um menor tráfego na rede e uma aplicação mais rápida.

Mesmo que a base de dados esteja na mesma máquina que a aplicação, segundo[FEV1995], a criação de pacotes de instrução SQL resulta num programa mais simples e que efetua menos chamada a base de dados.

5.2.2 FUNCIONALIDADE DO PL/SQL

As informações são transmitidas entre o PL/SQL e a base de dados através de variáveis, e estas variáveis nada mais são que uma localização na memória que pode ser lida por um programa e ou atribuída por outro. Cada variável definida tem um tipo respectivo que para fazer acesso a base deve ser compatível com o campo que foi convencionado na base de

dados. A programação PL/SQL também suporta vários tipos de ciclos. Um ciclo por sua vez é um comando que executa repetidas vezes os mesmos comandos.

5.2.2.1ESTRUTURA DE BLOCOS

A unidade básica em PL/SQL é um bloco. Todos os programas construídos em PL/SQL obrigatoriamente possuem blocos, que tem a facilidade de serem encadeados entre si. Normalmente cada bloco executa uma unidade de trabalho lógica no programa, separando as tarefas diferentes uma das outras. Um bloco possui a seguinte estrutura:

a) Bloco chamado como *Declarative Section* : Bloco opcional, onde se necessário são declarados variáveis, tipos, cursores;
b) Bloco chamado como *Executable Section* : Bloco não opcional, pelo menos um comando executável deve existir dentro do bloco. Neste bloco é onde desenvolve-se os algoritmos para melhor acesso a base de dados;
c) Bloco chamado como *Exception Handling Section* : Bloco opcional, neste bloco são tratados erros de execução do bloco anterior, e onde são retornados os próprios erros do banco de dados ORACLE.

O modelo que foi baseado para a criação do PL/SQL foi a linguagem de terceira geração ADA. Muitas das estruturas de códigos disponíveis em ADA, também são encontrados na linguagem PL/SQL. Como a estrutura dos blocos, outras funcionalidades encontradas em PL/SQL foram herdadas da linguagem ADA, tais como, tratamento de exceções, a sintaxe para declarar procedimentos e funções e os pacotes *(packages).* No decorrer deste capítulo poderão ser visualizadas tais semelhanças.

5.2.2.2 TRATAMENTO DE ERROS

A seção de tratamento de erros é responsável para responder a erros de *runtime* (momento de execução), que são encontrados pelo programa. Separando então as estruturas de bloco do código propriamente e as rotina de erro, tornam-se independente uma da outra.

5.2.2.3 CURSORES

Um cursor é utilizado para processar várias linhas obtidas da base de dados. Quando utiliza-se um cursor, o programa pode acessar dentro de um comando de repetição uma linha de cada vez, trabalhar com a mesma e passar para a próxima linha.

5.3 PL/SQL DINÂMICO

Um dos pacotes instalados junto com ORACLE a partir da versão 7 é o pacote chamado de DBMS_SQL, neste pacote contém funções que executam em tempo de *runtime*, por exemplo, é possível escolher uma tabela um campo e algumas condições para restrição na busca de dados para alguma operação sem precisar compilar o programa novamente. Para compreender a modo de funcionamento do pacote DBMS_SQL, necessitamos examinar diferença entre o SQL dinâmico e SQL estático.

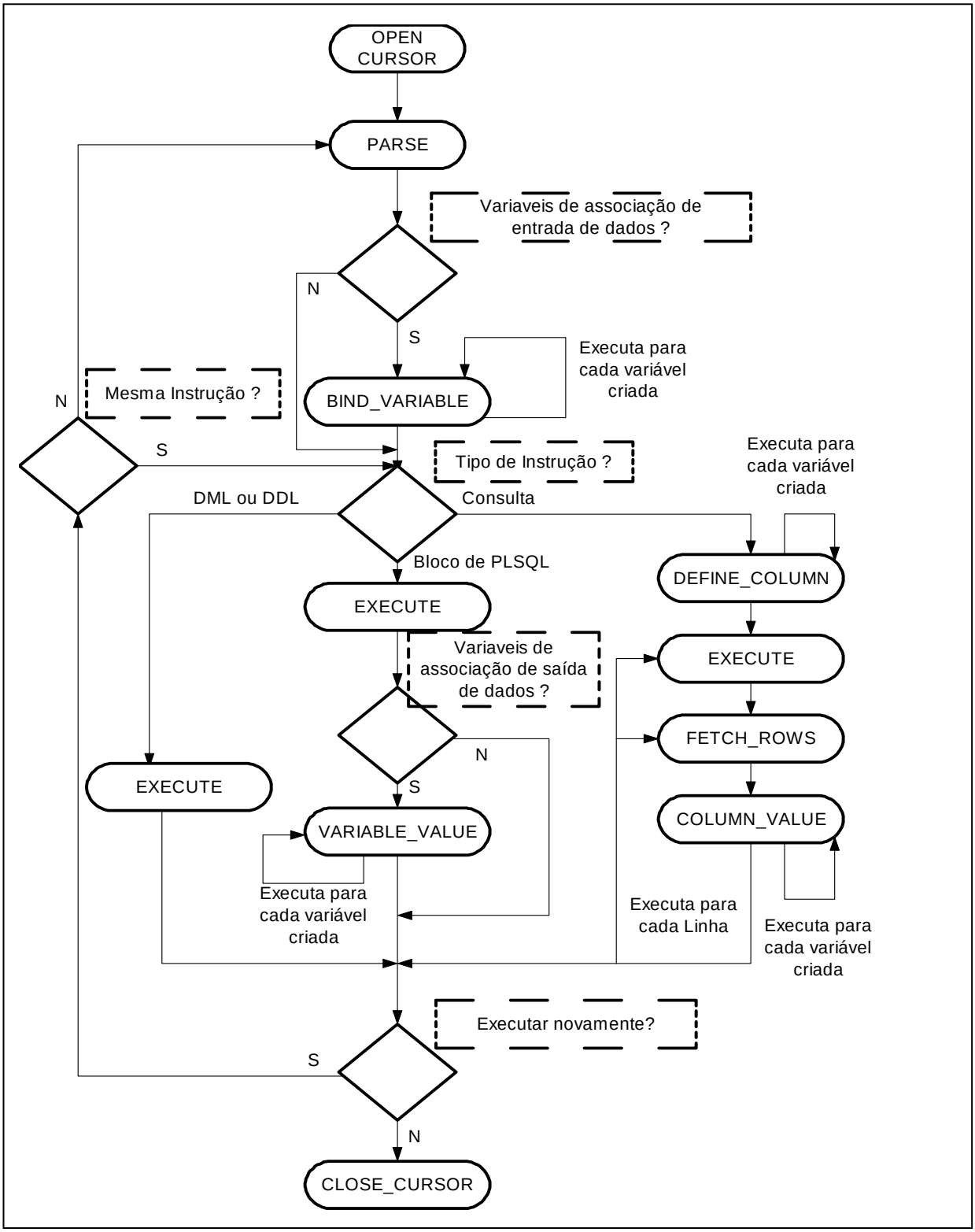
5.3.1 SQL DINÂMICO X SQL ESTÁTICO

O SQL estático é o que costumeiramente utiliza-se em nossos blocos de PL/SQL, onde o compilador conhece a instrução no mesmo momento em que se está compilando o programa. Já no caso do SQL dinâmico, utiliza-se o pacote DBMS_SQL, este recebe uma cadeia de caracteres e compila esta cadeia como se estivesse sendo uma entrada de dados no ambiente SQL/PLUS, ou seja, passa por um verificador léxico, sintático e semântico. Após, executa o comando e retorna os valores conforme comando passado na cadeia de caracteres. Veja na tabela 15, o exemplo.

TABELA 15: COMPARAÇÃO ENTRE SQL DINÂMICO E SQL ESTÁTICO

SQL Estático	SQL Dinâmico
UPDATE emp SET nom_emp = ‘XXXX’ WHERE cod_emp = 1;	Var_emp := ‘NNNN’; Var_campo := ‘dsc_emp = YYYYYY’; ‘UPDATE ’||var_emp|| ’ SET ’ || var_campo || ‘ WHERE cod_emp = 1’;

FIGURA 07: FLUXOGRAMA DO FUNCIONAMENTO DO PACOTE DBMS_SQL



Fonte[ORA1999].

Na figura 07, pode-se verificar como o pacote DBMS_SQL funciona, as chamadas, qual a ordem dos comandos, as seqüências utilizadas para o procedimento, entre outros.

Os tipos de dados que os cursores dinâmicos executam são definidos na chamada de um procedimento, este executa o cursor de acordo com parâmetros recebidos. O tipo de dado que o cursor retornará pode ser diferente entre uma execução e outra. Uma breve descrição de cada fator será descrito conforme[ORA1999]:

a) OPEN_CURSOR: Existe uma semelhança com comando que abre um cursor no SQL estático, todas as instruções SQL são executadas dentro de um cursor. No PL/SQL dinâmico, o processamento dos cursores é controlado por si. Quando é feito a chamada OPEN_CURSOR este retorna um número de identificação, que é utilizado para identificar a área de contexto em que a instrução será processada. Todas as chamadas subsequentes utilizarão o mesmo número. Isto possibilita executar mais de um cursor dinâmico ao mesmo tempo, apenas com números de identificação diferentes;
b) PARSE: A análise de uma instrução implica em enviá-la para o servidor, onde a sintaxe e a semântica da instrução são verificadas. Se a instrução for uma consulta, o plano de execução também é determinado neste ponto;
c) BIND_VARIABLE: É a associação de uma variável a um repositório de valores tornando-se semelhante ao processo de associação que o PL/SQL utiliza no SQL estático. Um repositório com valores especiais são passados na cadeia de caracteres para que o pacote DBMS_SQL consiga executar a instrução e colocar o resultado em um endereço de memória grande o suficiente ou seja, a associação é o ato de associar este repositório de valores a uma variável real e informar o DBMS_SQL sobre o tipo e comprimento da variável. Esta associação é efetuada somente para variáveis de entrada de dados;
d) DEFINE_COLUMN: A definição de uma variável de saída de dados é semelhante à associação de uma variável de entrada de dados. Neste caso, as variáveis de saída de dados são o resultado de uma consulta. DEFINE_COLUMN identifica o tipo e o comprimento das variáveis de PL/SQL que receberão os dados, quando forem obtidos por FETCH_ROWS. Esta só é utilizado em variáveis de saída;
e) EXECUTE: Quando o comando não for de consulta, o comando EXECUTE é que executará a instrução e devolverá o número de linhas processadas. Para uma

consulta, EXECUTE determina o conjunto ativo de dados que serão alterados, posteriormente os dados são extraídos com o comando FETCH_ROWS. Para todas as instruções, as variáveis de associação são examinados na operação de EXECUTE;

f) FETCH_ROWS: Cada chamada de FETCH_ROWS devolverá mais dados do servidor. Os dados serão convertidos nos tipos de dados especificados por DEFINE_COLUMN;
g) VARIABLE_VALUE: Esta rotina é utilizada para determinar o valor de uma variável de associação, se for modificada pela instrução. Isto só é utilizado quando a instrução é um bloco de PL/SQL;
h) COLUMN_VALUE: Depois de chamar FETCH_ROWS, COLUMN_VALUE é utilizado para devolver realmente os dados. São necessários variáveis do mesmo tipo que as especificadas em DEFINE_COLUMN. COLUMN_VALUE só deve ser utilizado para consultas;
i) CLOSE_CURSOR: Quando o processo é concluído, o cursor é fechado. Fazendo com que todos os recursos utilizados pelo cursor sejam liberados.

6 PROTÓTIPO DO RECONHECEDOR FONÉTICO

Neste fase do trabalho, descreve-se os aspectos de desenvolvimentos do protótipo, ou seja, o planejamento, desenvolvimento, testes e o modo de funcionamento. Este protótipo tem o objetivo de mostrar duas formas de busca: uma baseada na teoria de [NOG1989], e outra sendo uma variante desta teoria.

6.1 BASE DE DADOS

Os dados utilizados para o desenvolvimento e testes do protótipo foram selecionados a partir de várias tabelas da base de dados da TEClógica Serv. em Informática Ltda. Estes dados foram importados para uma tabela contendo nomes e endereços, tanto de clientes como fornecedores. Estes dados serão utilizados apenas para a validação final, na entrega do protótipo sem importância no contexto do trabalho.

TABELA 16: FORMATO DOS DADOS RETIRADOS DA BASE

CÓDIGO	Um número seqüencial, pois não existe relevância para o trabalho.
NOMES E OU ENDEREÇOS	Dados de tamanhos diferenciados que são tratados pelo protótipo apresentado.

6.2 ANÁLISE DO PROTÓTIPO

O protótipo terá por finalidade principal localizar palavras no banco de dados através de comparação fonética. Para isto ele vai analisar uma palavra ou frase que recebe como parâmetro de quem estiver utilizando o sistema. Transformar esta palavra ou frase em um código que o protótipo consiga entender, e buscar em uma base de dados que já foi previamente escolhida pelo próprio usuário, todos os nomes foneticamente parecidos ou idênticos, consultando em duas formas distintas: a primeira que é conforme a proposta por [NOG1989], e a segunda que é uma variante a esta proposta.

Existe as opções de buscar apenas os nomes que se iniciam com o que foi digitado, ou mesmo nomes que possuem a palavra digitada ou ainda que finalizam com a palavra digitada. Estes por sua vez todos parametrizados e para cada tipo diferente de forma de acesso.

6.3 ESPECIFICAÇÃO DA ROTINA FONÉTICA EM FLUXOGRAMA

As figuras 08, 09, 10 representam a rotina fonética na forma de diagrama, e o modo de implementação.

FIGURA 08: DIAGRAMA DA ROTINA FONÉTICA

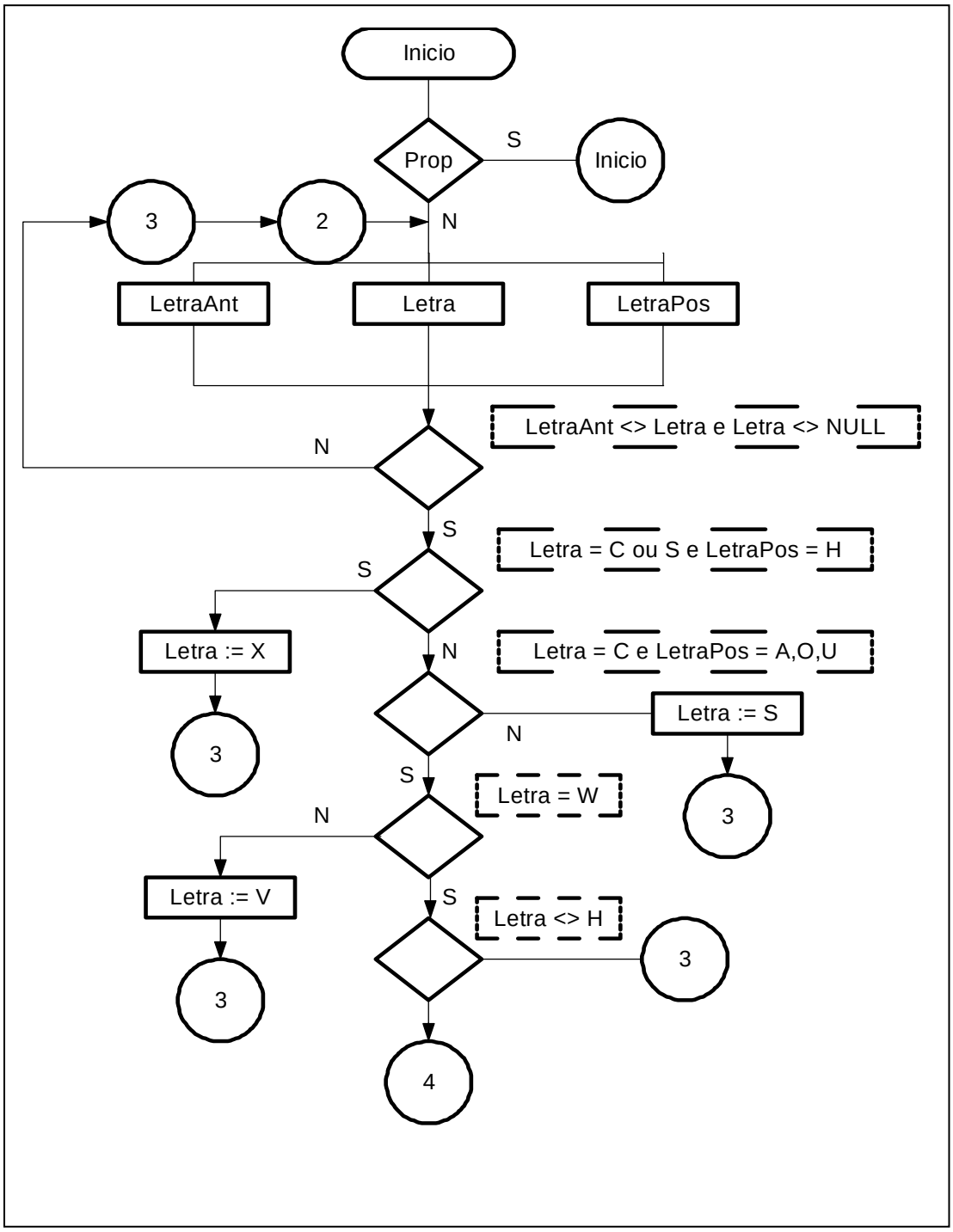


FIGURA 09: DIAGRAMA DA ROTINA FONÉTICA

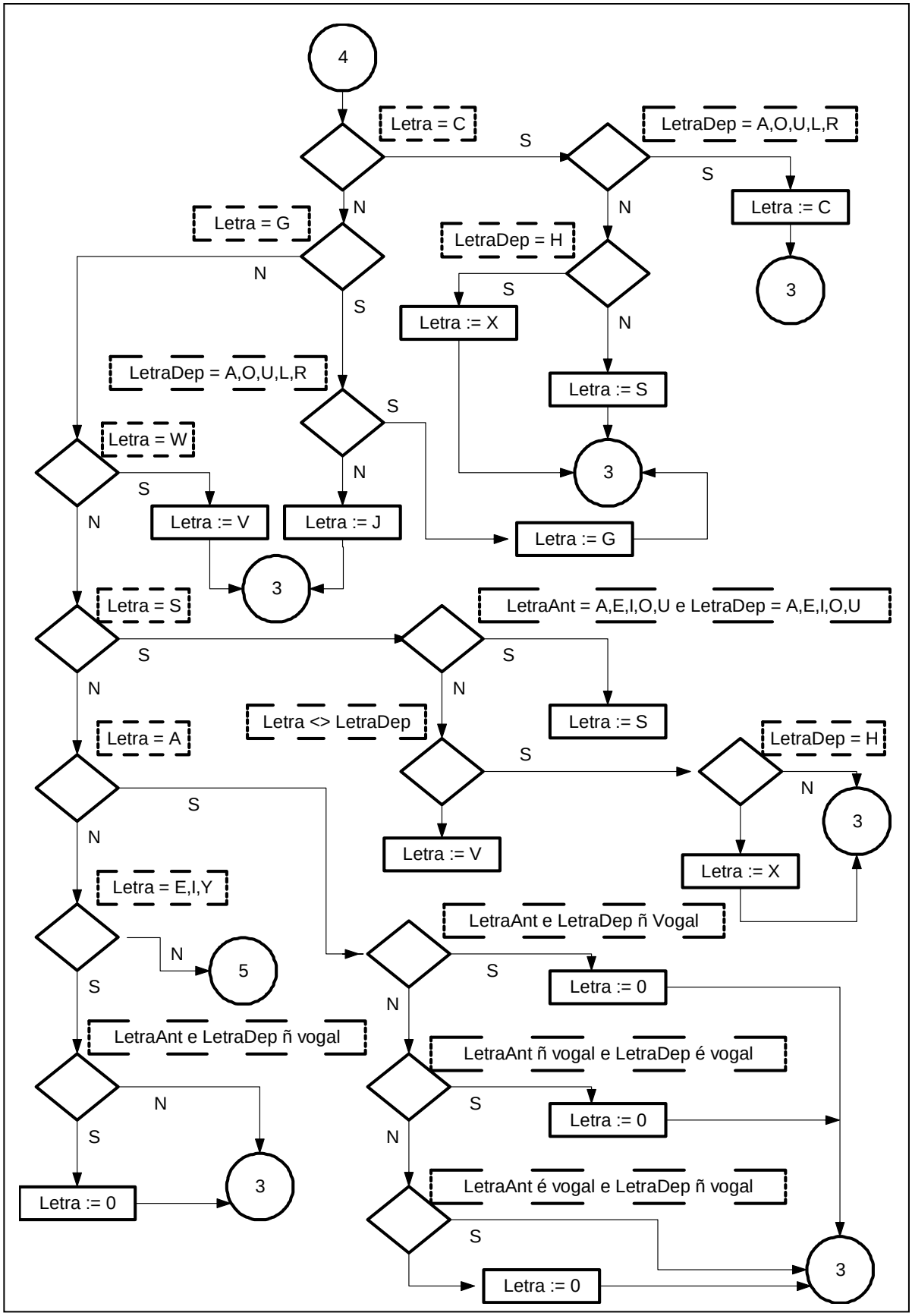
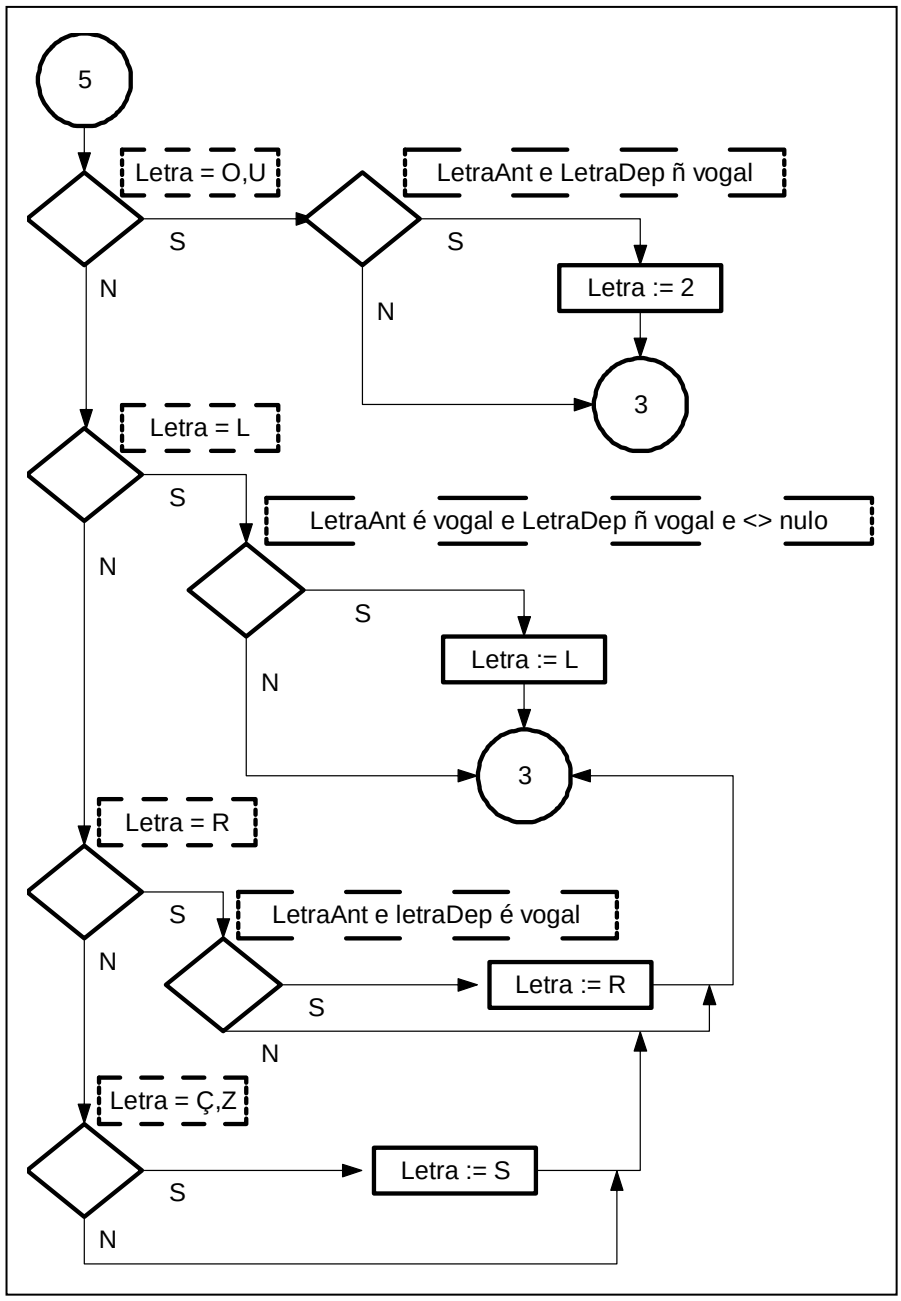
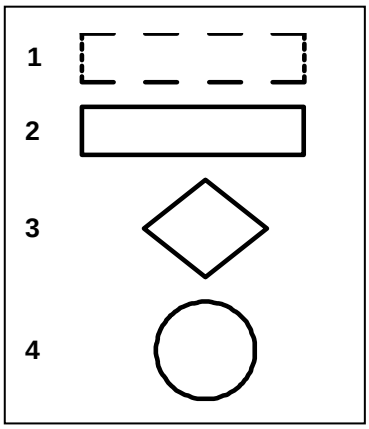


FIGURA 10: DIAGRAMA DA ROTINA FONÉTICA



6.3.1 DOCUMENTAÇÃO DO FLUXOGRAMA

FIGURA 11 : DOCUMENTAÇÃO DA ROTINA FONÉTICA



As itens acima relacionados fazem parte do fluxograma da especificação da rotina fonética, estas partes serão documentadas a seguir, com objetivo de melhorar o entendimento e a compreensão:

a) Item 01 : Significa área de descrição da condição, está sempre associado a um item do tipo 04 (visto a seguir), de acordo com a condição verifica-se se o resultado é verdadeiro ou falso e segue-se o procedimento;
b) Item 02 : Significa área de atualização de fonemas, neste objeto é que é recebido o novo fonema ou a letra pela qual deve-se considerar o fonema, conforme tabela 07 do alfabeto fonético visto no capítulo 3.
c) Item 03 : Significa a condição, a decisão a ser tomada conforme as três letras recebidas na entrada e a condição que escolhe o próximo passo.
d) Item 04 : Significa Inicio, ou final de ação, de acordo com a direção da seta. Se a seta termina no objeto, significa final. Se a seta inicializa no objeto, significa que se esta inciando o processo.

6.4 ESPECIFICAÇÃO DA ROTINA FONÉTICA EM CONOTAÇÃO BNF

As notações apresentadas até o momento descrevem linguagens de programação por meio das gramáticas. Neste ponto introduzir-se-a algumas notações para descrever reconhecedores.

Os diagramas de estados são a mais simples e, talvez, a mais eficiente, representação de reconhecedores. Esses diagramas representam graficamente as transições de máquinas de estados finitos, ou seja, os autômatos finitos.

Algumas notações adicionais, permitem que os diagramas de estados representem não apenas as linguagens regulares, mas também as linguagens sensíveis ao contexto. Na verdade, essas notações adicionais são apenas a representação dos elementos particulares aos autômatos de pilha, de modo que um diagrama passará a ser composto de vários diagramas interligados por transições que associam dois estados pelo consumo de outro diagrama e não apenas de um átomo, tais transições são graficamente representadas por uma linha tracejada.

Como exemplo, considere as produções pertencentes a uma gramática para representação de expressões que obedecem a ordem de precedência. Segundo a notação BNF – *Backus-Naur Form*, tais produções podem ser escritas assim:

<regra> ::= SE <condição> ENTÃO <fonema>

<condição>::= <cláusula> | <cláusula> E <cláusula>

<cláusula> ::= <vogal><vogal>

<cláusula> ::= <vogal><consoante><vogal>

<cláusula> ::= <vogal><consoante><consoante><vogal>

<cláusula> ::= <vogal><consoante><consoante><consoante><vogal>

<cláusula> ::= <vogal><consoante><consoante><consoante><consoante><vogal>

<fonema>::= <tabela07>

<consoantes>::=<b,c,...,z,>

<vogais>::=<a,e,i,o,u>

6.5 FUNCIONAMENTO DO PROTÓTIPO

O protótipo foi desenvolvido em quatro módulos, sendo cada um dos módulos descritos a seguir:

a) primeiro módulo: independente, este é responsável por criar a tela de alteração da tabela escolhida. Não tem ligação nenhum com outros módulos;
b) segundo módulo: independente, este é responsável por gerar os códigos fonéticos da tabela e colunas específicas;
c) terceiro módulo: independente, responsável construção da página de consulta e preparação dos parâmetros para a consulta;
d) quarto módulo: independente, responsável em gerar o código fonético de um parâmetro recebido e devolver ao procedimento chamado.

6.5.1 VANTAGENS DO DESENVOLVIMENTO MODULARIZADO

Existem várias vantagens em um desenvolvimento modularizado não só neste caso mas em um âmbito geral[RAM1999], confirma o seguinte:

a) manutenção: para futuras manutenções, que são muito comuns em software, o desenvolvimento modular facilita por ser independente e ao ser alterado o módulo dificilmente este causará impacto em outros módulos ao qual é ligado. A exceção fica no caso de ser alterados a quantidade de parâmetros ou mesmo, ser alterado o tipo dos parâmetros, já que neste caso os módulos comunicam-se apenas através de parâmetros;
b) independência: no caso de desenvolvimento de sistemas mais complexos e grandes, várias pessoas podem desenvolver paralelamente, módulos diferentes e no final agrupá-los, ganhando em tempo e agilidade.

6.5.2 DESVANTAGENS DO DESENVOLVIMENTO MODULARIZADO

Da mesma forma que o desenvolvimento modularizado traz vantagens existem desvantagens que podem ser muito agravantes se não forem contornadas, ainda com [RAM1999] as desvantagens são:

a) manutenção : no caso de um sistema for mal modularizado, ou seja, não se tornar

independente, ao haver manutenção no módulo, este pode causar falhas em outros, sendo que dificilmente será detectado a curto prazo. Podendo assim causar danos muito maiores;

b) independência: no caso de não haver total independência os módulos não poderão, ou não funcionarão de acordo se forem feitos por mais de uma pessoa, ou ainda ficará muito mais difícil de ser desenvolvido por um grupo maior.

6.6 TELAS DO PROTÓTIPO

Os próximos tópicos mostrarão a forma das telas, e a funcionalidade de cada uma. Existem três módulos que foram implementados separadamente. Cada um dos módulos representa uma tela, e possuem uma função distinta, que será visto no decorrer do tópico.

6.6.1 TELA DE ABERTURA

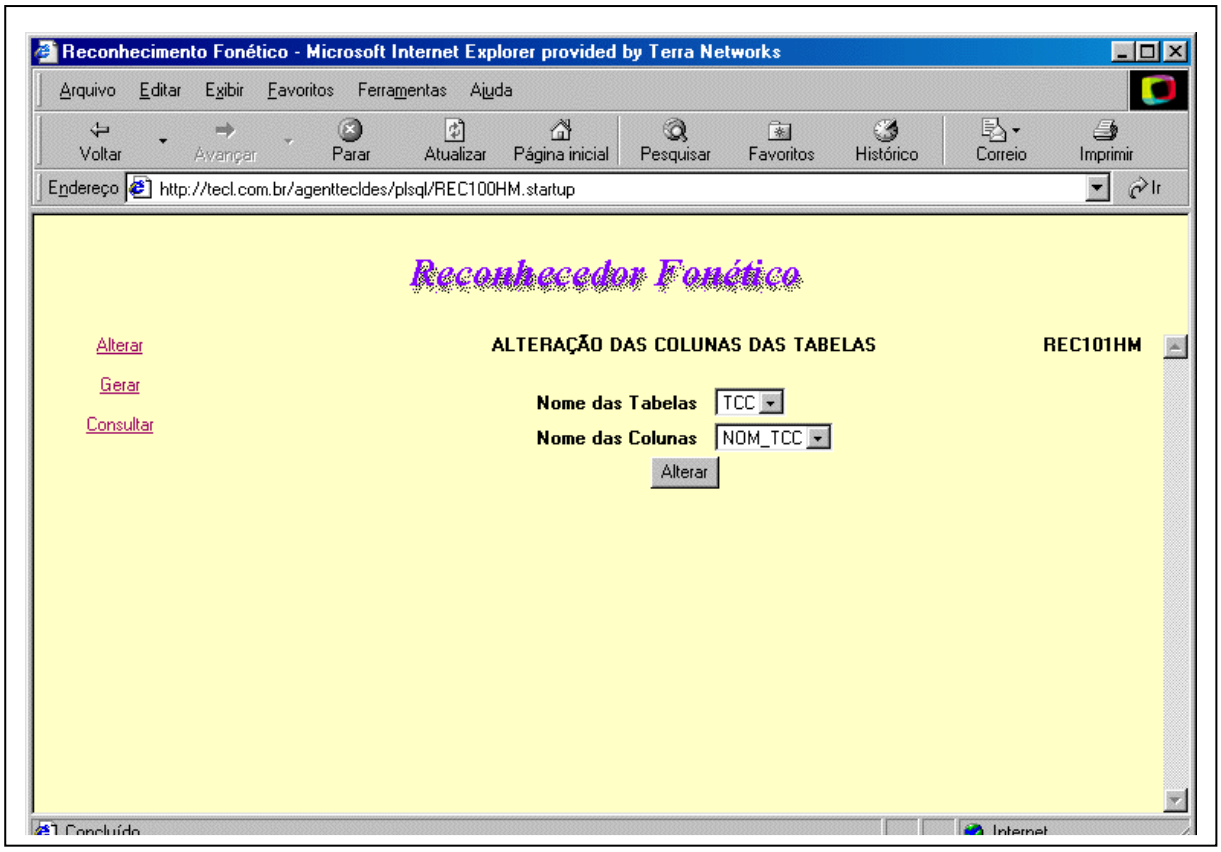
Nesta tela apenas terá a apresentação do protótipo, juntamente com o menu para os próximos módulos, conforme figura 11. O código fonte utilizado para gerar a tela de abertura pode ser visualizado conforme anexo 01.

FIGURA 12 : TELA DE ABERTURA

6.6.2 TELA DE MODIFICAÇÃO DA TABELA

Esta tela tem a finalidade de modificar a estrutura física da tabela, criando dois novos atributos, sendo um para o código fonético de [NOG1989], e o outro para a variante implementada que servirá para receber os códigos fonéticos que serão gerados em outro módulo, conforme figura 12.

FIGURA 13: TELA DE MODIFICAÇÃO DA TABELA



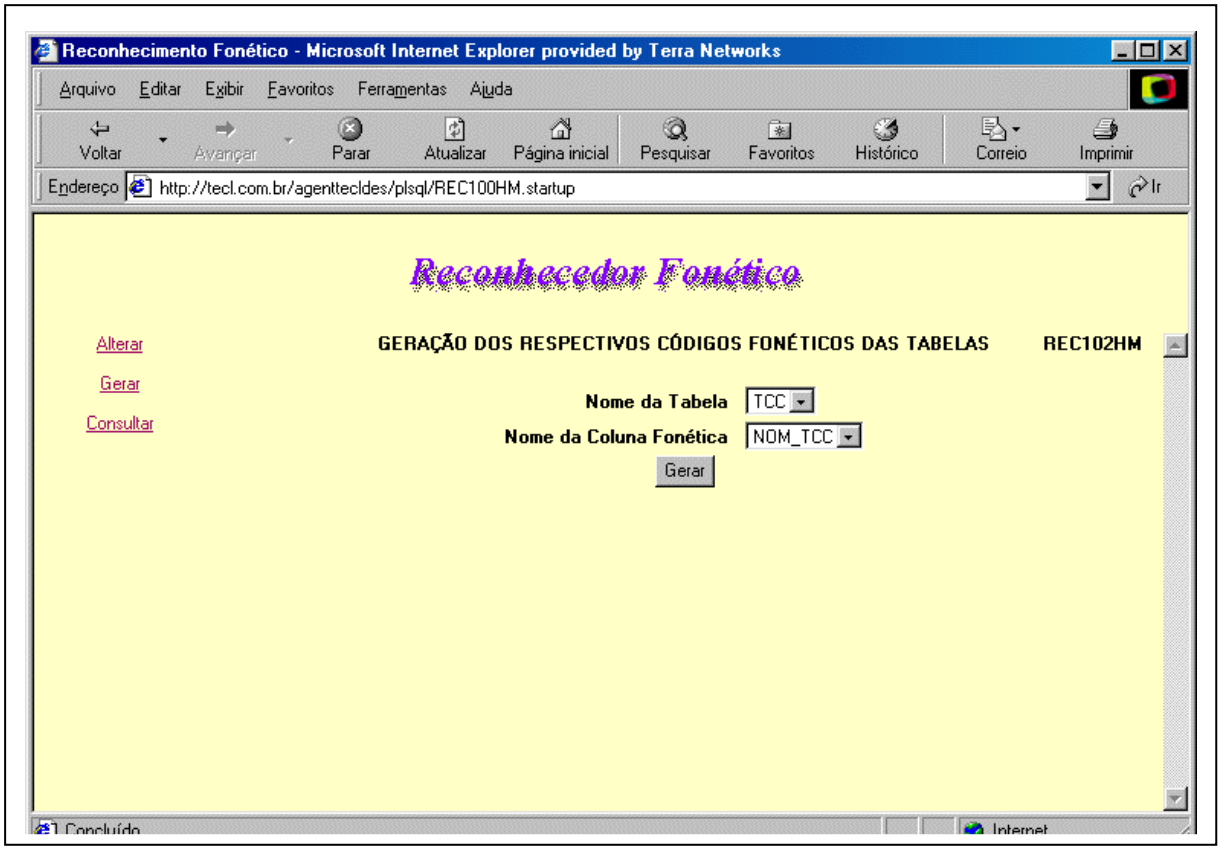
Esta tela é carregada com todas as tabelas que o usuário conectado possui acesso. Ao escolher uma tabela o mesmo buscará do banco todas as colunas que a tabela possui que sejam do tipo caracter. Escolhe-se então a coluna que se deseja gerar o código fonético.

Ao clicar no botão o mesmo executará a rotina que criará um novo atributo, a fim de receber os códigos fonéticos da tabela. Veja código de criação de uma nova coluna de acordo com anexo 02.

6.6.3 TELA DE GERAÇÃO DO CÓDIGO FONÉTICO

Este módulo possui o objetivo de buscar todos os dados do atributo escolhido pelo usuário de uma determinada tabela, e para cada linha retornada chama a rotina de geração do código fonético, e atribui o resultado a coluna gerada no módulo anterior. Veja figura 13.

FIGURA 14: TELA DE GERAÇÃO DO CÓDIGO FONÉTICO



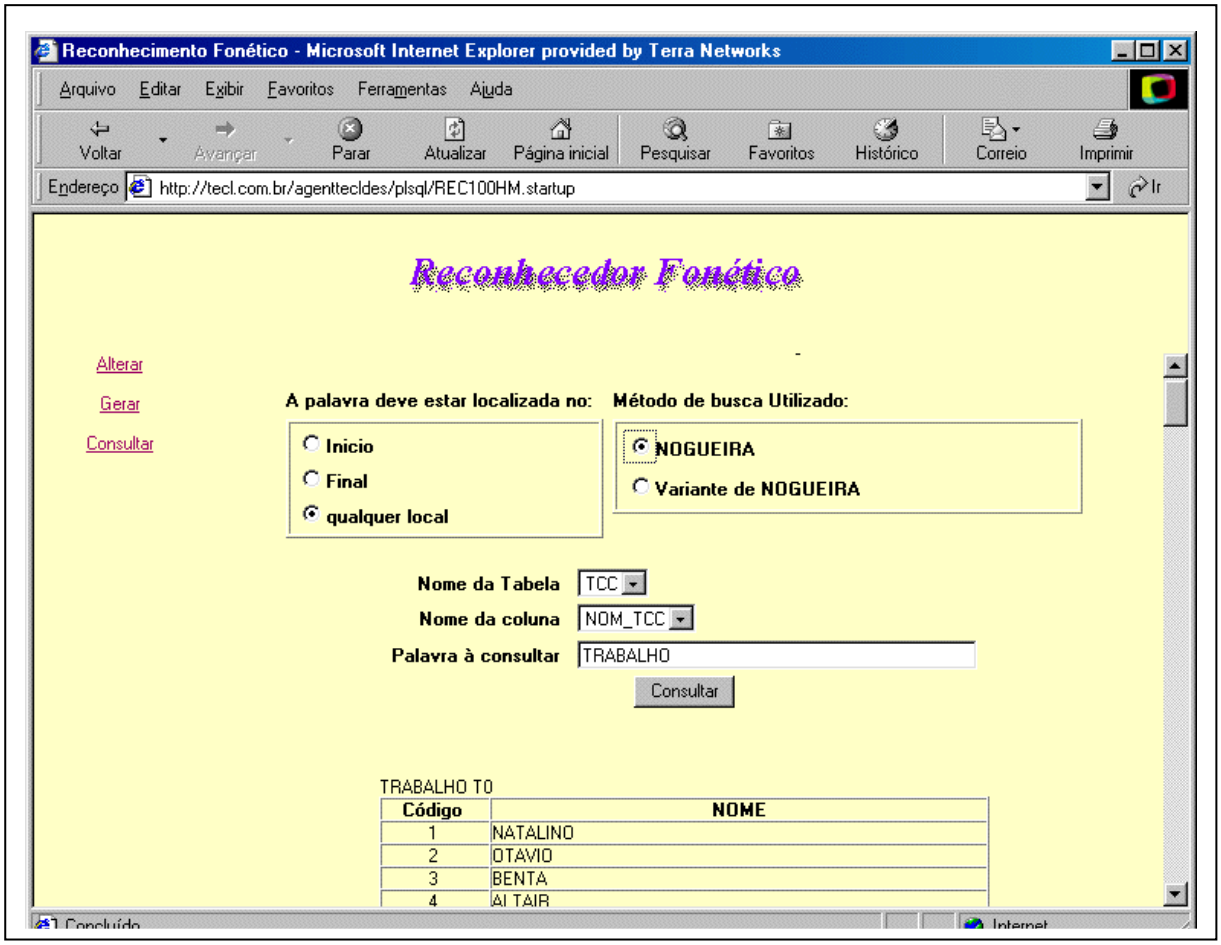
Ao entrar nesta tela, esta tem o mesmo procedimento que a tela anterior, trazendo todas as tabelas criadas abaixo da conta conectada. Quando o usuário escolher uma tabela da lista o protótipo buscará no banco de dados todas os atributos da tabela que sejam do tipo caracter e ainda possuem um atributo para receber o código gerado.

Ao selecionar o atributo que receberá o código fonético e clicar no botão "**gerar**" a rotina executa a função de buscar os dados, através do SQL dinâmico e gera o código fonético atribuindo-o ao respectivo atributo. Veja o código fonte conforme anexo 03.

6.6.4 TELA DE CONSULTA

Este módulo é a tela principal, onde é feito a pesquisa através do código fonético gerado nos módulos anteriores. Além de montar a página dinamicamente com os nomes vindos do banco de dados. Conforme figura 14.

FIGURA 15: TELA DE CONSULTA



Este módulo por sua vez possui as mesmas características dos módulos anteriores, pois a nível de protótipo, é necessário que diga ao sistema de qual coluna deve fazer a consulta, ou seja, ao entrar na tela de consulta a tela carregará também as tabelas geradas abaixo da conta que esta conectado.

Ao escolher uma tabela será feito uma consulta a base, buscando as respectivas colunas para que o usuário possa escolher.

Ainda possui a possibilidade de escolher para procurar tudo que se inicie com a palavra solicitada, ou mesmo tudo que termine com a mesma ou ainda que se encontre em qualquer posição da palavra procurada.

Além da possibilidade de optar por duas rotinas de reconhecimento, ou seja, podendo haver diferenças entre uma rotina e outra, retornando portanto, para um nome digitado, resultados diferentes. Veja como foi construído a página, conforme anexo 05.

6.7 TESTES EXECUTADOS COM O PROTÓTIPO

Após o término da implementação foram executadas várias baterias de testes com a aplicação. Estes testes foram executados diretamente dentro de uma rede intranet, ou seja, com um tráfego muito pequeno, e com uma velocidade de transferência muito superior a da internet normal. Contudo chegamos aos seguintes resultados, conforme tabela 17.

TABELA 17: RESULTADO DOS TESTES DA APLICAÇÃO

Palavra	**Método**	**Quantidade de Registros Retornados**
MAURO	NOGUEIRA	563
	VARIANTE	2
MATHEUS	VARIANTE	14
	NOGUEIRA	2
ABEL	VARIANTE	65
	NOGUEIRA	112
HUGO	VARIANTE	262
	NOGUEIRA	262
EVERALDO	VARIANTE	15

	NOGUEIRA	252

Conforme a tabela 17, a quantidade de registros retornados dar-se-á com a opção existente no protótipo de buscar registros com o respectivo código fonético em qualquer ponto da palavra, contudo isto trouxe muitas palavras que não tinham muito a ver com a palavra solicitada para pesquisa. A tabela 18 mostra com maior clareza.

TABELA 18: EXEMPLO DAS PALAVRAS RETORNADAS

PALAVRA SOLICITADA	**RETORNO**	**[NOG1989]**	**VARIANTE**
HUGO	HUGOLINO	X	X
	UGOLINO	X	X
	HUGO	X	X
	LONGUINHO	X	
MAURO	MARIO	X	X
	MANOEL	X	
	MARIA	X	X
	MAXIMINO	X	
	LUCIMAR	X	
	WALDEMAR	X	X

Outro teste realizado com o protótipo foi para comparar os métodos utilizados e para verificar qual seria o mais exato. O resultado obtido é que 20% dos registro possuíam um código fonético diferente, e 80% possuíam o mesmo código. O método mais exato ficou com o método variante de Nogueira, em compensação o método de Nogueira chegou mais próximo da realidade, pois trazia sempre mais registros do que solicitado. A variante do

método de Nogueira se tornou mais rápido, pois utiliza de menos rotinas para ser gerado, mas este fato não alterou as considerações já que o tempo perdido pelo método de Nogueira não chega a ser significante.

7 CONSIDERAÇÕES FINAIS

7.1 CONCLUSÕES

O presente trabalho aborda a importância da pesquisa fonética em uma consulta feita ao banco de dados ORACLE e como o uso do reconhecimento fonético na consulta pode trazer muitos benefícios a organização tornando-a uma ferramenta muito eficaz.

Ao finalizar este trabalho, concluí-se que uma rotina fonética é essencial para o desenvolvimento de sistemas utilizados nos dias atuais, principalmente em sistemas que possuem em uma palavra a chave para chegar-se a um determinado registro.

Outra situação importante que deve ser considerada, é que ao desenvolver um sistema, que será utilizado uma rotina de reconhecimento fonético, esta função deve ser considerada desde as primeiras fases de um projeto. Deve ser definido de maneira global e que tenha um acesso a qualquer momento no sistema. Se este for mal definido poderá trazer conseqüências cruciais, do tipo alguma das consultas poderão não ter acesso a rotina, fazendo com que o sistema não fique padronizado, não obtendo os resultados esperados.

O protótipo apresenta duas formas de geração do código fonético, uma proposta por [NOG1989], e outra sendo uma variação desta primeira proposta. Considerando testes feitos e os resultados de consultas feitas pelos dois métodos, verifica-se que o segundo método é mais eficaz em casos de erros de digitação, no entanto verificou-se que o segundo método também é mais ágil, pois ignora todo o tratamento dado a rotina de separação silábica, tendo com isto um enorme ganho de tempo em caso de geração de muitos nomes.

Porém a proposta que [NOG1989] sugere encontra um maior número de registros em cada consulta, fazendo com que registros sejam mais facilmente identificados, o que não acontece com o segundo método, pois este considera praticamente todas as vogais e consoantes, tendo o resultado um código fonético maior, consistindo em menos registros nas buscas.

7.2 LIMITAÇÕES DO PROTÓTIPO

Na realidade o protótipo não consegue identificar exceções que ocorrem na língua portuguesa no geral, assim como gírias ou mesmo nomes regionais, pois este necessita de um novo estudo individualmente para encontrar um comportamento.

Também é uma grande deficiência deste protótipo um reconhecimento fonético de outras línguas, até porque comparando os sons produzidos pela escrita na língua portuguesa, em outra língua terá sons diferentes.

7.3 DIFICULDADES ENCONTRADAS

Algumas dificuldades que devem ser consideradas e que foram encontradas no decorrer do trabalho, é a existência de bibliografia referente a fonemas e ou fonética, mas que tratam o assunto voltando-o para a fala, a voz.

Outro tópico que deve ser levado em consideração é o fato da língua portuguesa possuir uma enormidade de gírias, ou exceções fazendo com que um planejamento estruturado não obtenha sucesso.

7.4 SUGESTÕES

A rotina fonética hoje é efetivamente compatível com o banco de dados ORACLE, uma sugestão seria de ser criado um componente que pudesse ser compatível a qualquer outro banco de dados, e ou para ser implementado em qualquer outra linguagem de programação, para que realmente está rotina fique modularizada e para qualquer aplicação.

Outra sugestão seria criar esta rotina, mas voltada a outra língua como por exemplo o inglês que é muito utilizado por nós e nem sempre encontramos o que queremos por não saber como se escreve.

ANEXO 01

```
/* executa as ações do formulário */
PROCEDURE ACTIONQUERY (P_NM_TABELA IN VARCHAR2 := NULL
 ,P_NM_COLUNA IN VARCHAR2 := NULL ,P_ACAO IN VARCHAR2 := NULL
 ,P_BOTAO IN VARCHAR2 := NULL )  IS
-- Program Data
W_DS_SQL_STMT VARCHAR2(2000) := NULL;
W_ID_CURSOR INTEGER;
W_ID_RETORNO INTEGER;
W_NM_CAMPO VARCHAR2(2000) := NULL;
W_DS_ERRO VARCHAR2(2000) := NULL;
BEGIN
------------------------------------------------
-- Objetivo: criar uma nova coluna para recepção do código fonético
-- Autor   : Fabiano José Rossetto
-- Data    : 05/05/2000
------------------------------------------------
--
w_ds_erro := NULL;
IF (p_acao = 'l') THEN
    RAISE localiza;
END IF;
w_nm_campo := p_nm_coluna || '_FONEMA';
if (p_acao = 's') then
```

```
 begin
    w_ds_sql_stmt := 'alter table '|| p_nm_tabela ||' add ('|| w_nm_campo ||'varchar(2000))';
     w_id_cursor   := dbms_sql.open_cursor;
     dbms_sql.parse (w_id_cursor, w_ds_sql_stmt, dbms_sql.V7);
     w_id_retorno  := dbms_sql.execute(w_id_cursor);
     dbms_sql.close_cursor(w_id_cursor);
 exception
   when others then
       w_ds_erro := substr(sqlerrm(sqlcode), 1, 80);
       raise erro;
 end;
end if;
-- CHAMA EXPLICITAMENTE TELA DE CONSULTA
w_ds_erro := 'Operação executada com Sucesso.!!';
RAISE localiza;
EXCEPTION
 WHEN localiza THEN
      rec001hm.formquery(p_nm_tabela,p_nm_coluna,w_ds_erro);
 WHEN erro THEN
      rec001hm.formquery(p_nm_tabela,p_nm_coluna,w_ds_erro);
 WHEN others THEN
     ROLLBACK;
     w_ds_erro := SQLERRM(SQLCODE);
     rec001hm.formquery(p_nm_tabela,p_nm_coluna,w_ds_erro);
END ACTIONQUERY;
```

ANEXO 02

```
/* executa as ações do formulário */
PROCEDURE ACTIONQUERY (P_NM_TABELA IN VARCHAR2 := NULL
 ,P_NM_COLUNA IN VARCHAR2 := NULL ,P_ACAO IN VARCHAR2 := NULL
 ,P_BOTAO IN VARCHAR2 := NULL )  IS
-- Program Data
W_DS_SQL_STMT VARCHAR2(2000) := NULL;
W_ID_CURSOR INTEGER;
W_ID_RETORNO INTEGER;
W_NM_CAMPO VARCHAR2(2000) := NULL;
W_DS_ERRO VARCHAR2(2000) := NULL;
BEGIN
------------------------------------------------
-- Objetivo: criar uma nova coluna para recepção do código fonético
-- Autor   : Fabiano José Rossetto
-- Data    : 05/05/2000
------------------------------------------------
--
w_ds_erro := NULL;
IF (p_acao = 'l') THEN
    RAISE localiza;
END IF;
w_nm_campo := p_nm_coluna || '_FONEMA';
if (p_acao = 's') then
```

```
 begin
    w_ds_sql_stmt := 'alter table '|| p_nm_tabela ||' add ('|| w_nm_campo ||' varchar(2000))';
     w_id_cursor   := dbms_sql.open_cursor;
     dbms_sql.parse (w_id_cursor, w_ds_sql_stmt, dbms_sql.V7);
     w_id_retorno  := dbms_sql.execute(w_id_cursor);
     dbms_sql.close_cursor(w_id_cursor);
 exception
   when others then
       w_ds_erro := substr(sqlerrm(sqlcode), 1, 80);
       raise erro;
 end;
end if;
-- CHAMA EXPLICITAMENTE TELA DE CONSULTA
w_ds_erro := 'Operação executada com Sucesso.!!';
RAISE localiza;
EXCEPTION
 WHEN localiza THEN
      rec001hm.formquery(p_nm_tabela,p_nm_coluna,w_ds_erro);
 WHEN erro THEN
      rec001hm.formquery(p_nm_tabela,p_nm_coluna,w_ds_erro);
 WHEN others THEN
     ROLLBACK;
     w_ds_erro := SQLERRM(SQLCODE);
     rec001hm.formquery(p_nm_tabela,p_nm_coluna,w_ds_erro);
END ACTIONQUERY;
```

ANEXO 03

```
/* executa a ação do form e gera os códigos fonéticos; */
PROCEDURE ACTIONQUERY (P_NM_TABELA IN VARCHAR2 := NULL
 ,P_NM_COLUNA_FONEMA IN VARCHAR2 := NULL
,P_ACAO IN VARCHAR2 := NULL ,P_BOTAO IN VARCHAR2 := null )  IS
-- Program Data
W_CD_FONEMA VARCHAR2(2000) := NULL;
W_DS_ROWID VARCHAR2(240) := NULL;
W_ID_CURSOR INTEGER;
W_ID_CURSOR_HANDLER INTEGER;
W_ID_RETORNO INTEGER;
W_DS_ERRO VARCHAR2(2000) := NULL;
W_CD_FONEMA_UPD VARCHAR2(2000) := NULL;
W_NM_COLUNA VARCHAR2(240) := NULL;
W_DS_ROWID_UPD VARCHAR2(240) := NULL;
W_NM_CAMPO VARCHAR2(240) := NULL;
W_SQL_SELECT VARCHAR2(2000) := NULL;
W_DS_SQL_STMT VARCHAR2(2000) := NULL;

BEGIN
-----------------------------------------------
-- Objetivo: Atualizar os dados da tabela em relação ao Fonemas.
-- Autor   : Fabiano José Rossetto
-- Data    : 07/05/2000
```

```
-----------------------------------------------
--
w_ds_erro := NULL;
IF (p_acao = 'l') THEN
   RAISE localiza;
END IF;

w_nm_coluna:= SUBSTR (p_nm_coluna_fonema, 1, INSTR(p_nm_coluna_fonema, '_FONEMA' )  -1 );
w_nm_campo := p_nm_coluna_fonema;
IF (p_acao = 's') THEN
    w_sql_select := 'SELECT '|| w_nm_coluna || ', rowid FROM '|| p_nm_tabela;
    w_id_cursor_handler := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(w_id_cursor_handler,w_sql_select, DBMS_SQL.v7);
    DBMS_SQL.DEFINE_COLUMN(w_id_cursor_handler, 1, w_cd_fonema, 2000);
    DBMS_SQL.DEFINE_COLUMN(w_id_cursor_handler, 2, w_ds_rowid, 240);
    w_id_retorno := dbms_sql.execute(w_id_cursor_handler);
  LOOP
     IF DBMS_SQL.FETCH_ROWS(w_id_cursor_handler) > 0 THEN
        DBMS_SQL.COLUMN_VALUE(w_id_cursor_handler, 1, w_cd_fonema);
        DBMS_SQL.COLUMN_VALUE(w_id_cursor_handler, 2, w_ds_rowid);
        w_cd_fonema_upd:= REC100PN.rtn_rec_fonema(w_cd_fonema);
        w_ds_rowid_upd := w_ds_rowid;
       BEGIN
          w_ds_sql_stmt := 'update '||p_nm_tabela||' set '||
```

```
            w_nm_campo||' = :w_cd_fonema
            WHERE ROWID = :w_ds_rowid';
           w_id_cursor   := DBMS_SQL.OPEN_CURSOR;
           DBMS_SQL.PARSE(w_id_cursor, w_ds_sql_stmt, dbms_sql.V7);
           DBMS_SQL.BIND_VARIABLE(w_id_cursor,':w_cd_fonema',w_cd_fonema_upd);
           DBMS_SQL.BIND_VARIABLE(w_id_cursor,':w_ds_rowid',w_ds_rowid_upd);
           w_id_retorno  := DBMS_SQL.EXECUTE(w_id_cursor);
           DBMS_SQL.CLOSE_CURSOR(w_id_cursor);
         EXCEPTION
           WHEN OTHERS THEN
             w_ds_erro := SUBSTR(SQLERRM(SQLCODE), 1, 80);
             RAISE erro;
         END;
    ELSE
      EXIT;
    END IF;
  END LOOP;
DBMS_SQL.CLOSE_CURSOR(w_id_cursor_handler);
END IF;
COMMIT;
-- CHAMA EXPLICITAMENTE TELA DE CONSULTA
w_ds_erro := 'Operação executada com Sucesso.!!';
RAISE localiza;
EXCEPTION
  WHEN localiza THEN
```

```
      rec002hm.formquery(p_nm_tabela,p_nm_coluna_fonema,w_ds_erro);
   WHEN erro THEN
      rec002hm.formquery(p_nm_tabela,p_nm_coluna_fonema,w_ds_erro);
   WHEN OTHERS THEN
      ROLLBACK;
      w_ds_erro := SQLERRM(SQLCODE);
     rec002hm.formquery(p_nm_tabela,p_nm_coluna_fonema,w_ds_erro);
END ACTIONQUERY;
```

ANEXO 04

```
/* Tela do form de consulta na WEB. */
PROCEDURE FORMQUERY (P_NM_TABELA IN VARCHAR2 := NULL
 ,P_NM_COLUNA IN VARCHAR2 := NULL ,P_NM_CONSULTA IN VARCHAR2 := null
 ,P_FLG_INICIO IN VARCHAR2 := NULL ,P_DS_ERRO IN VARCHAR2 := NULL ) IS
-- Program Data
W_DS_ERRO VARCHAR2(240) := NULL;
W_SQL_SELECT VARCHAR2(2000) := NULL;
W_CD_COUNT NUMBER(5) := NULL;
W_ID_CURSOR_HANDLER INTEGER;
W_ID_RETORNO INTEGER;
W_NM_COLUNA VARCHAR2(240) := NULL;
W_NM_CAMPO VARCHAR2(2000) := NULL;
W_CD_FONEMA VARCHAR2(2000) := NULL;
-- PL/SQL Block
BEGIN
-----------------------------------------
-- Objetivo: Consultar dados de uma base Oracle através de busca fonética.
-- Autor   : Fabiano J. Rossetto
-- Data    : 02/05/2000
-----------------------------------------
----------------------
-- CABEÇALHO DA PÁGINA
----------------------
```

```
htp.HtmlOpen;
htp.HeadOpen;
  htp.Title('Reconhecimento Fonético');
  htp.meta('Content-Type',NULL,'text/html; charset=iso-8859-1');
htp.HeadClose;
------------------
-- CORPO DA PÁGINA
------------------
w_ds_erro := p_ds_erro;
htp.BodyOpen;
htp.centerOpen;
htp.tableOpen('border="0" cellpadding="0" cellspacing="0" width="680"');
  htp.tableRowOpen;
     htp.tableHeader('&'||'nbsp;','left',cattributes=>'width="15%"');
     htp.tableHeader('CONSULTA CONSIDERANDO CODIFICAÇÃO FONÉTICA '
‘,'center',cattributes=>'width="70%"');
     htp.tableHeader('REC103HM','right',cattributes=>'width="15%"');
  htp.tableRowClose;
htp.tableClose;
htp.formOpen('REC003HM.actionquery','post','_self',NULL,'name="rec003hm"');
htp.tableOpen('border="0" cellpadding="0" cellspacing="0" width="680"');
  htp.tableRowOpen;
if p_flg_inicio = 1 then
      htp.tableHeader(htf.formRadio('p_flg_inicio',1,'CHECKED')||'Selecionar a partir do
Inicio'||'&'||'nbsp;','left',cattributes=>'width="15%"');
```

```
      htp.tableHeader(htf.formRadio('p_flg_inicio',2,null)||'Selecionar a partir do Final'||'&'||'nbsp;','left',cattributes=>'width="15%"');

      htp.tableHeader(htf.formRadio('p_flg_inicio',3,null)||'Selecionar a partir do Meio'||'&'||'nbsp;','left',cattributes=>'width="15%"');

elsif p_flg_inicio = 1 then

      htp.tableHeader(htf.formRadio('p_flg_inicio',1,null)||'Selecionar a partir do Inicio'||'&'||'nbsp;','left',cattributes=>'width="15%"');

      htp.tableHeader(htf.formRadio('p_flg_inicio',2,'CHECKED')||'Selecionar a partir do Final'||'&'||'nbsp;','left',cattributes=>'width="15%"');

      htp.tableHeader(htf.formRadio('p_flg_inicio',3,null)||'Selecionar a partir do Meio'||'&'||'nbsp;','left',cattributes=>'width="15%"');

    else

      htp.tableHeader(htf.formRadio('p_flg_inicio',1,null)||'Selecionar a partir do Inicio'||'&'||'nbsp;','left',cattributes=>'width="15%"');

      htp.tableHeader(htf.formRadio('p_flg_inicio',2,null)||'Selecionar a partir do Final'||'&'||'nbsp;','left',cattributes=>'width="15%"');

      htp.tableHeader(htf.formRadio('p_flg_inicio',3,'CHECKED')||'Selecionar a partir do Meio'||'&'||'nbsp;','left',cattributes=>'width="15%"');

    end if;

  htp.tableRowClose;

htp.tableClose;

htp.TableOpen('border="0"');

  htp.tableRowOpen;

    htp.tableHeader('Nome da Tabela '||'&'||'nbsp;','right',NULL,NULL,NULL,NULL,'NOWRAP');


htp.tableData(htf.formSelectOpen('p_nm_tabela',NULL,1,cattributes=>'onChange=Localizar()')||rec100hm.cb_tabela(p_nm_tabela,'N'),'left',cattributes=>'colspan="4"');
```

```
    htp.formSelectClose;
  htp.tableRowClose;
  IF p_nm_tabela IS NOT NULL THEN
    htp.tableRowOpen;
      htp.tableHeader('Nome da Coluna' ||'&'||'nbsp;' ,'right' ,NULL ,NULL, NULL, NULL,
'NOWRAP');
      htp.tableData(htf.formSelectOpen('p_nm_coluna',NULL,1,cattributes=>'
')||rec100hm.cb_coluna(p_nm_tabela,p_nm_coluna,'N'),'left',cattributes=>'colspan="4"');
      htp.formSelectClose;
    htp.tableRowClose;
    htp.tableRowOpen;
      htp.tableHeader('Nome para Consultar '||'&'||'nbsp;' ,'right', NULL, NULL, NULL, NULL,
'NOWRAP');
      htp.tableData(htf.formText('p_nm_consulta',  '30','50',p_nm_consulta),  'left',cattributes=>'
colspan="4"');
    htp.tableRowClose;
  ELSE
    htp.formHidden('p_nm_coluna',NULL);
    htp.formHidden('p_nm_consulta',NULL);
  END IF;
htp.TableClose;
htp.formHidden('p_acao',null);
htp.p('<INPUT TYPE="BUTTON" name= "p_botao" VALUE= "Consultar" onClick=
Salvar()>');
htp.formClose;
htp.br;
```

```
htp.br;
if p_nm_consulta is not null then
  htp.tableOpen('border="1" cellpadding="0" cellspacing="0" width="680"');
    htp.tableRowOpen;
      htp.tableHeader('Código'||'&'||'nbsp;','left',cattributes=>'width="15%"');
      htp.tableHeader(' NOME ','center',cattributes=>'width="70%"');
    htp.tableRowClose;

  w_nm_coluna := p_nm_coluna ||'_FONEMA';
  w_cd_fonema := REC100HM.rtn_rec_fonema(p_nm_consulta);
if p_flg_inicio = 1 then
     w_cd_fonema := '%'||w_cd_fonema;
  elsif p_flg_inicio = 2 then
     w_cd_fonema := w_cd_fonema||'%';
  else
     w_cd_fonema := '%'||w_cd_fonema||'%';
  end if;
w_sql_select := 'SELECT ' || p_nm_coluna || ' FROM ' || p_nm_tabela || '
                where ' || w_nm_coluna || ' like ' || "" || w_cd_fonema || "";
w_id_cursor_handler := DBMS_SQL.OPEN_CURSOR;
DBMS_SQL.PARSE(w_id_cursor_handler,w_sql_select, DBMS_SQL.v7);
DBMS_SQL.DEFINE_COLUMN(w_id_cursor_handler, 1, w_nm_campo, 2000);
w_id_retorno := dbms_sql.execute(w_id_cursor_handler);
w_cd_count := 1;
LOOP
```

```
  IF DBMS_SQL.FETCH_ROWS(w_id_cursor_handler) > 0 THEN
      DBMS_SQL.COLUMN_VALUE(w_id_cursor_handler, 1, w_nm_campo);
     htp.tableRowOpen;
      htp.tableData(w_cd_count||'&'||'nbsp;','left',cattributes=>'width="15%"');
      htp.tableData(w_nm_campo,'center',cattributes=>'width="70%"');
     htp.tableRowClose;
   ELSE
    EXIT;
   END IF;
   w_cd_count := w_cd_count + 1;
 END LOOP;
DBMS_SQL.CLOSE_CURSOR(w_id_cursor_handler);
htp.tableClose;
END IF;
htp.CenterClose;
-------------------
-- RODAPÉ DA PÁGINA
-------------------
htp.bodyClose;
htp.htmlClose;
--
-- VERIFICA SE EXISTEM ERROS DE OUTRAS PROCEDURES E MOSTRA-OS AO USUÁRIO
IF (w_ds_erro IS NOT NULL) THEN
 rec100hm.alerta(w_ds_erro);
```

```
END IF;
EXCEPTION
  WHEN OTHERS THEN
    w_ds_erro := SQLERRM(SQLCODE);
    rec100hm.alerta(w_ds_erro);
END FORMQUERY;
```

REFERÊNCIAS BIBLIOGRÁFICAS

[ALP1999] ALPHA, Manoel. **Fonetizador para a língua portuguesa.** 1999, Endereço eletrônico : http://www.fonetica.com.br. Data da consulta : 20/02/2000.

[AUL1995] AULT, Michael R. **Oracle 7.0 : Administração & Gerenciamento.** Rio de Janeiro: Infobook, 1995.

[BRO1998] BROWN, Bradley D, NIEMIEC, Richard J, TREZZO, Joseph C. **Oracle Application Server Web Toolkit Reference**. São Paulo : Osborne McGraw-Hill, 1998.

[CER1995] CERÍCOLA, Vincent Oswald. **Oracle : Banco de Dados Relacional e Distribuído - Ferramentas para Desenvolvimento.** São Paulo : Makron Books, 1995.

[FAR1988] FARRACO E MOURA. **Gramática Fonética e Fonemas morfologia sintaxe estilística.** São Paulo : Ática, 1988.

[FEV1995] FEVERSTEIN, Steven. **Oracle PL/SQL Programming Database Management Systens.** Rio de Janeiro : O' Reilly & Associates, 1995.

[GAS1998] GASTALDI, Hermínio. **Protótipo de uma aplicação para controlar postagem de catálogos com enfoque em residências.** Blumenau 1998. Monografia (Bacharel Ciências da Computação) – Centro de Ciências Exatas e Naturais, FURB.

[HEC1994] HECKLER, Evaldo. **Estrutura das palavras:** Famílias Morfologia Análise Origem. São Leopoldo RS : Unisinos, 1994.

[HUG1996] HUGO, Toledo Jr. **Oracle Network Connecting:** Oracle products the internet. São Paulo : Osborne McGraw-Hill, 1996.

[JAK1967] JAKOBSON, Roman. **Fonema e Fonologia.** Rio de Janeiro : Livraria Acadêmica, 1967.

[LOP1997] LOPES, Edward. **Fundamentos da lingüistica contemporânea.** São Paulo : Cultrix, 1997.

[MAT1979] MATTOS, Rinaldo. **Metodologia de Análise Gramatical**. Rio de Janeiro : Vozes Ltda, 1979.

[MOR1995] MORAIS, Rinaldo de Oliveira. **Oracle7 – Server : Conceitos Básicos.** São Paulo : Érica, 1995.

[NOG1989] NOGUEIRA, Antônio R. **Pesquisa Fonética.** In Tecninfo (Blumenau : 1989 pág. 107 -121).

[ORA1992] ORACLE, Handbook. **Oracle 7 Server Administrator Guide.** Redwood City : Oracle Corporations, 1992.

[ORA1998] ORACLE, Handbook. **Writing Applications for Oracle Mobile Agents.** Redwood City : Oracle Corporations, 1992.

[ORA1999] ORACLE, Handbook. **PL/SQL User's Guide and Reference.** Redwood City : Oracle Corporations, 1992.

[RAM1999] RAMALHO, José Antônio. **Oracle 8i : Ideal para desenvolvedor.** São Paulo : Berkeley, 1999.

[SAL1993] SALEMI, Joe. **Guia PC Magazine para Bancos de Dados Cliente/Servidor.** Rio de Janeiro : Infobook, 1993.