SGBD Oracle - Implementação de Função para Fonetização em Português

SGBD Oracle - Implementação de Função para Fonetização em Português

por Edivaldo Vicente dos Santos

I – Introdução

Todos que trabalham com informática, e neste caso me dirijo principalmente aos profissionais que lidam com Sistemas de Gerenciamento de Banco de Dados(SGBDS), sabem da existência de centenas de funções úteis nos SGBDS que gostaríamos de utilizar, mas que por terem sido concebidas para outros países, com língua e sistemas métricos diferentes, se mostram inviáveis para utilização no nosso ambiente operacional. Uma maneira muito interessante de tratar essa limitação, existente na maioria dos SGBDS, é lançarmos mão da criação de nossas próprias funções, adequando-as ao nosso ambiente e necessidades. Vamos exemplificar esse procedimento com a criação de um procedimento armazenado em Java seu acesso como uma função pública no SGBD Oracle.

Por que implementar a função no SGBD e não na aplicação? Confesso que esta discussão esta muito em pauta e, que coloca muitas vezes os administradores de Banco de Dados em conflito com os Desenvolvedores de Aplicação, na minha modesta opinião o assunto deve ser tratado caso a caso com a devida isenção técnica levando em conta vários fatores que podem pesar pela opção do SGBD, tais como:

- A natureza da organização – para grandes Organizações, como Bancos Comerciais, seu maior patrimônio são os dados e pelo porte de suas organizações a mudança de SGBD não é uma opção constante, conheço organizações que trabalham a mais de 10 anos com o mesmo SGBD e todos os seus projetos para daqui a 5 anos incluem o mesmo SGBD. Neste caso a independência da aplicação com relação ao SGBD não é fator chave, inclusive a sugestão de alteração do SGBD por parte de desenvolvedores externos é visto com muitas reservas. Nestes casos o cliente espera justamente o contrário, que se obtenha o máximo de seu SGBD, valorizando seu investimento, que não é pequeno.

- Muitas organizações com SGBD centralizado, Federado ou não, e que possuem ambiente de desenvolvimento e produção muito heterogêneos, com base de dados única optam pela centralização das regras de negócio no SGBD para evitarem os risco de perda da "inteligência " da sua atividade e fortalecer a padronização sobre os diversos aplicativos que executam em paralelo. Dessa forma uma aplicações diferentes (Delphi-client-server, .NET ou Java) realizam exatamente o mesmo procedimento padronizado no Banco de Dados. Ao mesmo tempo em que o cliente domina a "inteligência" de seu negócio ele impõe uma padronização aos seus desenvolvedores, sem se utilizar de Servidores de Aplicação ou protocolos criados para essa finalidade.

Não estou querendo afirmar que sempre se devam criar as funções no SGBD, não se deve, mas que sempre se deve proceder à análise da melhor alternativa para cada caso.

I I – Função para busca fonética em Português

Um problema comum na localização de registros, principalmente nomes próprios, vem da maneira como palavras pronunciadas da mesma maneira possuem diversas grafias, por exemplo Rafael e Raphael, Valter e Walter, etc.. que tem exatamente a mesma pronuncia, mas que possuem grafias diferentes o que torna a busca com operadores relacionais e funções, como "like", ineficientes em muitos casos. Para a língua Inglesa existe, em diversos sistemas, a função Soundex (para maiores detalhes procure a documentação do seu SGBD e caso queira conhecer um pouco mais sobre este algoritmo uma ótima referência é ART OF COMPUTER PROGRAMMING - V.3 SORTING AND SEARCHING, KNUTH, DONALD ERVIN).

Antes de entrar na função propriamente dita, abrimos um parêntese, existem diversas abordagens a esse problema, como por exemplo: A função deve ser inserida no banco ou aplicativo?, Como proceder a fonetização?(devemos fonetizar o nome completo ou cada parte do mesmo?) Existe um modelo que maximiza a eficiência do algoritmo?. Estas questões extrapolam o escopo desse pequeno artigo no qual vamos nos concentrar com a função de fonetização, sua implementação no banco de dados e sua utilização fazendo às vezes da função "soundex". Só para deixar registro existe um alfabético fonético internacional, já reparou nos caracteres "esquisitos" de seu dicionário inglês-português?, infelizmente os lingüistas de língua portuguesa não se deram, ainda, a importância que seria a existência de padronização fonética em caracteres da língua portuguesa para a informática, isso quer dizer que em virtude de diferenças nestes algoritmos podemos ter diferenças de resultados, mas este é um assunto para outro momento, e um dos motivos que podem servir de justificativa para que a função fique centralizada no SGBD.

DevMedia group
Equipe DevMedia
Noticias/Dicas/Artigos pulicados.
Ver space do autor

I I I– Procedimento Java Armazenado

Como citado anteriormente iremos construir nossa função tomando como base procedimentos armazenados em Java, se possível leia novamente as edições 4 e 5 da SQLMagazine.

A decisão de se usar procedimentos armazenados em Java no Oracle, e não uma linguagem como C/C++, é que os mesmos são executados originalmente no JVM do Oracle no espaço de endereço do banco de dados, com isso temos menor número de trocas de contexto entre processos ao nível de sistema operacional ao mesmo tempo em que o código Java esta sempre executando como "proprietário do software Oracle", detalhe o Oracle possui uma JDK embutido, Oracle9i - jdk 1.3 e o 10g - jdk 1.4 ambos Aurora. Para este exemplo utilizamos o Oracle 9i, creio que o mesmo funcionará sem maiores problemas para o Oracle 10G, porém não é possível implementar esses procedimento no Oracle Express Edition(XE) pois o mesmo não possui suporte para tal.

Neste exemplo iremos montar nossa classe Java fonetizar com a utilização do comando CREATE JAVA, este comando cria um objeto contendo um fonte de código Java ou uma Classe(maiores detalhes Oracle9i – SQL Reference – a96540.pdf) o comando utilizado terá a seguinte sintaxe:

CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED "Fonetica" As ...

O fonte da função de fonetização em Java, para a língua portuguesa que iremos utilizar é a disponibilizada pelo Instituto do Coração da Faculdade de Medicina da Universidade de São Paulo, que desenvolveu alguns componentes de fonetização em Java com suporte CORBA e está disponibilizando-os com código fonte aberto
(licença GNU) no Consórcio de Componentes de Software para Sistemas de Informação em Saúde (CCS-SIS). Estes componentes de fonetização foram utilizados na implementação do Serviço de Identificação de Pacientes (PIDS) e estão disponíveis em : http://www.incor.usp.br/spdweb/ccssis/fonetica/.

Visando simplificar a sua utilização usamos, apenas um pequeno fragmento desse pacote, mas fortemente indico a todos que o estudem por completo.

O Comando com o algoritmo Java completo está na **listagem 1(anexos)**.

I V – Procedimento Java Armazenado

Uma vez criada a classe temos que a tornar acessível para nossos usuários, isso se dá com a criação de uma função, obs.: Lembro que são necessárias as permissões e privilégios para a realização dessas tarefas.

CREATE OR REPLACE FUNCTION FONETIZAR (str VARCHAR) RETURN VARCHAR
AS LANGUAGE JAVA NAME 'Fonetica.fonetizar(java.lang.String) return java.lang.String';

Por fim agora podemos criar um sinônimo público para tornar disponível a todos os usuários do nosso banco a função que acabamos de criar.

CREATE PUBLIC SYNONYM FONETIZAR FOR FONETIZAR;

Neste momento podemos testar nossa função, imaginado uma tabela **FUNCIONARIO** , com o campo **nome** (varchar) a pesquisa poderia ser feita da seguinte forma:

```
SELECT NOME, CPF FROM FUNCIONARIOS
WHERE
FONETIZAR(NOME)=FONETIZAR("RAPHAEL")
```

Poderíamos obter como resultado tanto o funcionário "RAPHAEL" como "RAFAEL", uma outra customização possível seria a construção de uma tabela com os nomes "fonetizados" juntamente com um índice para a tabela FUNCIONARIO, com isso pode-se realizar uma busca mais rápida e com "fragmentos" do nome, poderíamos ter como retorno LUIZ RAPHAEL ou ROBERTO RAFAEL, juntamente com os resultados já retornados.

Espero que tenham gostado, e até a próxima.

V - Anexos

Listagem 1

```
CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED "Fonetica" AS
import java.util.*;
public class Fonetica {
 public static String fonetizar (String str) {
   //Fonetiza o string recebido como parametro e devolve
    //um outro string (que e o primeiro fonetizado)

    str = str.toUpperCase(); //todas as letras maiusculas
    str = removePrep(str); //remove as preposições
    str = removeAccentuation(str); //remove os acentos
    str = removeStrange(str); //remove caracteres diferentes de
                                    // A-Z, 0-9
    str = fonetize(str); //fonetiza o texto
   return str;
    }

        public static String fonetize (String str) {
            //Função que faz efetivamente a substituição de letras,
                 //fonetizando o texto

                 //matrizes de caracteres utilizadas para manipular o texto
            char[] foncmp = new char[256];
                char[] fonwrk = new char[256];
                 char[] fonaux = new char[256];
                 char[] fonfon = new char[256];

            int i, j, x, k, //contadores
```

```
            desloc, //posicao atual no vetor
                endfon, //indica se eh ultimo fonema
                copfon, //indica se o fonema deve ser copiado
                copmud, newmud; //indica se o fonema eh mudo

        //Vetor utilizado para armazenar o texto:
        //cada palavra do texto e armazenada em uma posicao do vetor
    Vector component = new Vector();

        i = 0;
    j = 0;//zera os contadores

        str = removeMultiple(str);
        //todos os caracteres duplicados sao eliminados
      //exemplo: SS -> S, RR -> R

component = strToVector(str);
//o texto eh armazenado no vetor:
//cada palavra ocupa uma posicao do vetor


for (desloc = 0; desloc < component.size(); desloc ++) {
//percorre o vetor, palavra a palavra

for (i = 0; i < 256; i++) {
    fonwrk[i] = ' ';
 fonfon[i] = ' ';//branqueia as matrizes
  }//for

  foncmp = component.elementAt(desloc).toString().toCharArray();
  fonaux = foncmp;
  //matrizes recebem os caracteres da palavra atual

j = 0;

  if (component.elementAt(desloc).toString().length() == 1) {
 fonwrk[0] = foncmp[0];
  //se a palavra possuir apenas 1 caracter, nao altera a palavra

 if (foncmp[0] == '_') {
     fonwrk[0] = ' ';
 //se o caracter for "_", troca por espaco em branco

 }//if
    else
      if ((foncmp[0] == 'E')||
          (foncmp[0] == '&')||
          (foncmp[0] == 'I')) {
   fonwrk[0] = 'i';
       //se o caracter for "E", "&" ou "I", troca por "i"

  }//if
  }//if
  else {
   for (i = 0; i < component.elementAt(desloc).toString().length(); i++)
 //percorre a palavra corrente, caracter a caracter

  if (foncmp[i] == '_')
       fonfon[i] = 'Y';     // _ -> Y
     else
       if (foncmp[i] == '&')
         fonfon[i] = 'i';     //& -> i
       else
         if ((foncmp[i] == 'E') ||
             (foncmp[i] == 'Y') ||
             (foncmp[i] == 'I'))
           fonfon[i] = 'i';     // E, Y, I -> i
         else
           if ((foncmp[i] == 'O') ||
         (foncmp[i] == 'U'))
             fonfon[i] = 'o';     // O, U -> u
      else
             if (foncmp[i] == 'A')
               fonfon[i] = 'a';      // A -> a
             else
               if (foncmp[i] == 'S')
                 fonfon[i] = 's';     // S -> s
               else
           fonfon[i] = foncmp[i];
                 //caracter nao eh modificado

    endfon = 0;
    fonaux = fonfon;

    //palavras formadas por apenas 3 consoantes
    //sao dispensadas do processo de fonetizacao
 if (fonaux[3] == ' ')
      if ((fonaux[0] == 'a') ||
          (fonaux[0] == 'i') ||
          (fonaux[0] == 'o'))
        endfon = 0;
      else
        if ((fonaux[1] == 'a') ||
            (fonaux[1] == 'i') ||
            (fonaux[1] == 'o'))
          endfon = 0;
        else
          if ((fonaux[2] == 'a') ||
```

```
            (fonaux[2] == 'i') ||
            (fonaux[2] == 'o'))
          endfon = 0;
        else {
          endfon = 1;
          fonwrk[0] = fonaux[0];
          fonwrk[1] = fonaux [1];
          fonwrk[2] = fonaux [2];
        }//else

if (endfon != 1) { //se a palavra nao for formada por apenas 3 consoantes...
  for (i = 0; i < component.elementAt(desloc).toString().length(); i++) {
  //percorre a palavra corrente, letra a letra

      copfon = 0;
      copmud = 0;
      newmud = 0;
      //zera variaveis de controle

      switch (fonaux[i]) {

        case 'a':  //se o caracter for a

          //se a palavra termina com As, AZ, AM, ou AN,
          //elimina a consoante do final da palavra
          if ((fonaux[i+1]== 's') ||
              (fonaux[i+1]== 'Z') ||
              (fonaux[i+1]== 'M') ||
              (fonaux[i+1]== 'N'))
            if(fonaux[i+2]!= ' ')
        copfon = 1;
            else {
              fonwrk[j] = 'a';
              fonwrk[j+1] = ' ';
              j++;
              i++;
            }//else
          else copfon = 1;
        break;

    case 'B':  //se o caracter for B

        // B nao eh modificado
        copmud = 1;
    break;

        case 'C':  //se o caracter for C

          x = 0;
          if (fonaux[i+1] == 'i')

      //ci vira si
          { fonwrk[j] = 's';
       j++;
            break;
          }//if

          //coes final vira cao
          if ((fonaux[i+1] == 'o') &&
              (fonaux[i+2] == 'i') &&
              (fonaux[i+3] == 's') &&
              (fonaux[i+4] == ' '))
          { fonwrk[j] = 'K';
            fonwrk[j+1] = 'a';
            fonwrk[j+2] = 'o';
            i = i + 4;
            break;
          }//if

          //ct vira t
          if (fonaux[i+1] == 'T')
            break;

          //   c vira k
      if (fonaux[i+1] != 'H')
          { fonwrk[j] = 'K';
            newmud = 1;

            //   ck vira k
       if (fonaux[i+1] == 'K')
          { i++;
            break;
          }//if

            else break;
          }//if

          //ch vira k para chi final, chi vogal, chini final e
      //chiti final

          //chi final ou chi vogal
      if (fonaux[i+1] == 'H')
            if (fonaux[i+2] == 'i')
              if ((fonaux[i+3] == 'a')||
                  (fonaux[i+3] == 'i')||
                  (fonaux[i+3] == 'o'))
                x = 1;
```

```
    // chini final
    else
      if (fonaux[i+3] == 'N')
        if (fonaux[i+4] == 'i')
          if (fonaux[i+5] == ' ')
            x = 1;

          else;
        else;
      else
        // chiti final
        if (fonaux[i+3] == 'T')
          if (fonaux[i+4] == 'i')
            if (fonaux[i+5] == ' ')
              x = 1;
    if (x == 1)
    { fonwrk[j] = 'K';
 j++;
      i++;
      break;
    }//if

   //chi, nao chi final, chi vogal, chini final ou chiti final
   //ch nao seguido de i
   //se anterior nao e s, ch = x
   if (j > 0)

     //sch: fonema recua uma posicao
if (fonwrk[j-1] == 's')
       { j--;
       }//if
     fonwrk[j] = 'X';
     newmud = 1;
i++;
   break;

   case 'D':  //se o caracter for D
     x = 0;

     //procura por dor
     if (fonaux[i+1] != 'o')
{ copmud = 1;
       break;
     }//if
     else
       if (fonaux[i+2] == 'R')
         if (i != 0)
           x = 1; // dor nao inicial
         else copfon = 1; // dor inicial
       else copfon = 1;  // nao e dor
     if (x == 1)
       if (fonaux[i+3] == 'i')
         if (fonaux[i+4] == 's') // dores
   if (fonaux[i+5] != ' ')
           x = 0;  // nao e dores
         else;
  else x = 0;
       else
         if (fonaux[i+3] == 'a')
           if (fonaux[i+4] != ' ')
    if (fonaux[i+4] != 's')
     x = 0;
          else
     if (fonaux[i+5] != ' ')
      x = 0;
           else;
         else;
        else x = 0;
     else x = 0;
     if (x == 1)
     { fonwrk[j] = 'D';
       fonwrk[j+1] = 'o';
       fonwrk[j+2] = 'R';
       i = i + 5;
     }//if
     else copfon = 1;
   break;

   case 'F':  //se o caracter for F

     //F nao eh modificado
     copmud = 1;
   break;

   case 'G':  //se o caracter for G

     //gui -> gi
     if (fonaux[i+1] == 'o')
       if (fonaux[i+2] == 'i')
       { fonwrk[j] = 'G';
         fonwrk[j+1] = 'i';
         j += 2;
         i +=2;
}//if
     //diferente de gui copia como consoante muda
 else copmud = 1;
```

```
        else

          //gl
          if (fonaux[i+1] == 'L')
            if (fonaux[i+2] == 'i')

              //gli + vogal -> li + vogal
              if ((fonaux[i+3]=='a')||
                  (fonaux[i+3]=='i')||
                  (fonaux[i+3]=='o'))
              { fonwrk[j] = fonaux[i+1];
      fonwrk[j+1] = fonaux[i+2];
                 j += 2;
      i += 2;
              }//if
              else

                //glin -> lin
                if(fonaux[i+3] == 'N')
      { fonwrk[j] = fonaux[i+1];
                   fonwrk[j+1] = fonaux[i+2];
       j += 2;
                    i += 2;
                  }/*if*/
                  else copmud = 1;
            else copmud = 1;
          else

            //gn + vogal -> ni + vogal
            if (fonaux[i+1] == 'N')
              if((fonaux[i+2]!='a')&&
       (fonaux[i+2]!='i')&&
                  (fonaux[i+2]!='o'))
      copmud = 1;
              else
              { fonwrk[j] = 'N';
                fonwrk[j+1] = 'i';
                j += 2;
                i++;
              }//else
            else

              //   ghi -> gi
              if (fonaux[i+1] == 'H')
                if (fonaux[i+2] == 'i')
                { fonwrk[j] = 'G';
                  fonwrk[j+1] = 'i';
                  j += 2;
                  i +=2;
                }//if
                else copmud = 1;
              else copmud = 1;
break;

    case 'H':  //se o caracter for H

    //H eh desconsiderado
break;

    case 'i':  //se o caracter for i

      if (fonaux[i+2] == ' ')

   //is ou iz final perde a consoante
   if (fonaux[i+1] == 's')
        { fonwrk[j] = 'i';
          break;
        }//if
        else
          if (fonaux[i+1] == 'Z')
          { fonwrk[j] = 'i';
            break;
          }//if

        //ix
  if (fonaux[i+1] != 'X')
    copfon = 1;
        else
          if (i != 0)
            copfon = 1;
    else

              //ix vogal no inicio torna-se iz
              if ((fonaux[i+2]=='a')||
                  (fonaux[i+2]=='i')||
                  (fonaux[i+2]=='o'))
              { fonwrk[j] = 'i';
      fonwrk[j+1] = 'Z';
                 j += 2;
                 i++;
      break;
              }//if
              else

              //ix consoante no inicio torna-se is
      if (fonaux[i+2]=='C' || fonaux[i+2]=='s') {
                 fonwrk[j] = 'i';
                 j++;
```

```
            i++;
            break;
           }//if
           else
             { fonwrk[j] = 'i';
              fonwrk[j+1] = 's';
              j += 2;
              i++;
              break;
             }//else
    break;

    case 'J':  //se o caracter for J

      //J -> Gi
      fonwrk[j] = 'G';
      fonwrk[j+1] = 'i';
      j += 2;
    break;

case 'K':  //se o caracter for K
      //KT -> T
      if (fonaux[i+1] != 'T')
   copmud = 1;
      break;

    case 'L':  //se o caracter for L

 //L + vogal nao eh modificado
      if ((fonaux[i+1] == 'a')||
          (fonaux[i+1] == 'i')||
          (fonaux[i+1] == 'o'))
        copfon = 1;
      else

        //L + consoante -> U + consoante
        if (fonaux[i+1] != 'H')
    { fonwrk[j] = 'o';
          j++;
          break;
    }//if

         //LH + consoante nao eh modificado
    else
          if (fonaux[i+2] != 'a' &&
        fonaux[i+2] != 'i' &&
              fonaux[i+2] != 'o')
            copfon = 1;
           else

            //LH + vogal -> LI + vogal
            { fonwrk[j] = 'L';
        fonwrk[j+1] = 'i';
             j += 2;
             i++;
             break;
            }
    break;


    case 'M':  //se o caracter for M

 //M + consoante -> N + consoante
      //M final -> N
      if ((fonaux[i+1] != 'a' &&
           fonaux[i+1] != 'i' &&
           fonaux[i+1] != 'o') ||
    (fonaux[i+1] == ' '))
      { fonwrk[j] = 'N';
        j++;
      }//if

 //M nao eh alterado
      else copfon = 1;
    break;

    case 'N':  //se o caracter for N

      //NGT -> NT
      if ((fonaux[i+1] == 'G') &&
          (fonaux[i+2] == 'T'))
      { fonaux[i+1] = 'N';
  copfon = 1;
      }//if
      else

        //NH + consoante nao eh modificado
        if (fonaux[i+1] == 'H')
    if ((fonaux[i+2] != 'a')&&
            (fonaux[i+2] != 'i')&&
            (fonaux[i+2] != 'o'))
           copfon = 1;

         //NH + vogal -> Ni + vogal
         else
         { fonwrk[j] = 'N';
           fonwrk[j+1] = 'i';
```

```
      j += 2;
           i++;
         }
        else copfon = 1;
    break;

case 'o':  //se o caracter for o

      //oS final -> o
      //oZ final -> o
 if ((fonaux[i+1] == 's') ||
          (fonaux[i+1] == 'Z'))
        if (fonaux[i+2] == ' ')
        { fonwrk[j] = 'o';
          break;
        }//if
        else copfon = 1;
       else copfon = 1;
    break;

    case 'P':  //se o caracter for P

      //PH -> F
      if (fonaux[i+1] == 'H')
      { fonwrk[j] = 'F';
        i++;
        newmud = 1;
      }//if
      else
        copmud = 1;
    break;

    case 'Q':  //se o caracter for Q

 //Koi -> Ki (QUE, QUI -> KE, KI)
 if (fonaux[i+1] == 'o')
        if (fonaux[i+2] == 'i')
        { fonwrk[j] = 'K';
          j++;
          i++;
          break;
        }//if

      //QoA -> KoA (QUA -> KUA)
      fonwrk[j] = 'K';
      j++;
    break;

case 'R':  //se o caracter for R

      //R nao eh modificado
      copfon = 1;
    break;

    case 's':  //se o caracter for s

      //s final eh ignorado
      if (fonaux[i+1] == ' ')
        break;

      //s inicial + vogal nao eh modificado
 if ((fonaux[i+1]=='a')||
          (fonaux[i+1]=='i')||
          (fonaux[i+1]=='o'))
        if (i == 0)
   { copfon = 1;
          break;
        }//if
        else

          //s entre duas vogais -> z
          if ((fonaux[i-1] != 'a')&&
              (fonaux[i-1]!='i')&&
              (fonaux[i-1]!='o'))
    { copfon = 1;
            break;
          }//if
          else

            //SoL nao eh modificado
            if ((fonaux[i+1] == 'o') &&
                (fonaux[i+2] == 'L') &&
                (fonaux[i+3] == ' '))
            { copfon = 1;
              break;
            }//if

            else
            { fonwrk[j] = 'Z';
              j++;
              break;
            }//else

      //ss -> s
      if (fonaux[i+1] == 's')
        if (fonaux[i+2] != ' ')
```

```
        { copfon = 1;
          i++;
          break;
        }//if
        else
        { fonaux[i+1] = ' ';
    break;
        }//else

      //s inicial seguido de consoante fica precedido de i
      //se nao for sci, sh ou sch nao seguido de vogal
 if (i == 0)
        if (!((fonaux[i+1] == 'C') &&
      (fonaux[i+2] == 'i')))
          if (fonaux[i+1] != 'H')
            if (!((fonaux[i+1] == 'C') &&
                 (fonaux[i+2] == 'H') &&
                 ((fonaux[i+3] != 'a')&&
                  (fonaux[i+3]!='i')&&
                  (fonaux[i+3]!='o'))))
            { fonwrk[j] = 'i';
              j++;
              copfon = 1;
       break;
            }//if

      //sH -> X;
      if (fonaux[i+1] == 'H')
      { fonwrk[j] = 'X';
        i++;
        newmud = 1;
        break;
      }//if
 if (fonaux[i+1] != 'C')
      { copfon = 1;
        break;
      }//if

      //   sCh nao seguido de i torna-se X
 if (fonaux[i+2] == 'H')
        { fonwrk[j] = 'X';
          i += 2;
          newmud = 1;
          break;
        }//if
 if (fonaux[i+2] != 'i')
 { copfon = 1;
        break;
      }//if

      //sCi final -> Xi
      if (fonaux[i+3] == ' ')
      { fonwrk[j] = 'X';
        fonwrk[j+1] = 'i';
        i = i + 3;
        break;
      }//if

      //sCi vogal -> X
      if ((fonaux[i+3]=='a')||
          (fonaux[i+3]=='i')||
          (fonaux[i+3]=='o') )
      { fonwrk[j] = 'X';
        j++;
        i += 2;
        break;
      }//if

      //sCi consoante -> si
      fonwrk[j] = 's';
      fonwrk[j+1] = 'i';
      j += 2;
      i += 2;
    break;

    case 'T':  //se o caracter for T

      //TS -> S
      if (fonaux[i+1] == 's')
        break;

      //TZ -> Z
      else
        if (fonaux[i+1] == 'Z')
          break;
        else copmud = 1;
          break;

    case 'V':  //se o caracter for V
case 'W':  //ou se o caracter for W

      //V,W inicial + vogal -> o + vogal (U + vogal)
 if (fonaux[i+1] == 'a'||
          fonaux[i+1] == 'i'||
          fonaux[i+1] == 'o')
        if (i == 0)
  { fonwrk[j] = 'o';
```

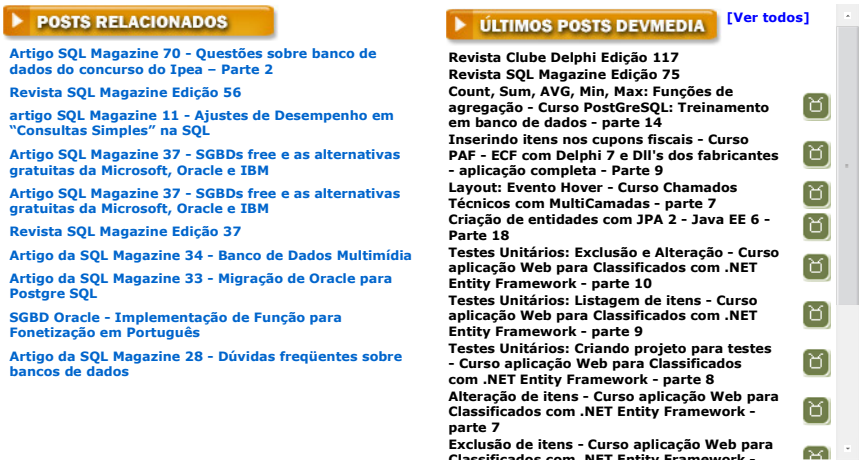
```
            j++;
           }//if

           //V,W NAO inicial + vogal -> V + vogal
      else
           { fonwrk[j] = 'V';
             newmud = 1;
           }//else

         else
         { fonwrk[j] = 'V';
           newmud = 1;
         }//else
       break;

       case 'X':  //se o caracter for X

    //caracter nao eh modificado
         copmud = 1;
  break;

       case 'Y':  //se o caracter for Y
  //Y jah foi tratado acima
       break;

       case 'Z':  //se o caracter for Z

         //Z final eh eliminado
         if (fonaux[i+1] == ' ')
           break;

         //Z + vogal nao eh modificado
    else
           if ((fonaux[i+1] == 'a')||
        (fonaux[i+1] == 'i')||
              (fonaux[i+1] == 'o'))
             copfon = 1;

           //Z + consoante -> S + consoante
           else
      { fonwrk[j] = 's';
             j++;
           }//else
           break;

  default: //se o caracter nao for um dos jah relacionados

         //o caracter nao eh modificado
         fonwrk[j] = fonaux[i];
    j++;
       break;
     }//switch

     //copia caracter corrente
     if (copfon == 1)
 { fonwrk[j] = fonaux[i];
       j++;
     }//if

     //insercao de i apos consoante muda
 if (copmud == 1)
       fonwrk[j] = fonaux[i];
     if (copmud == 1 || newmud == 1)
     { j++;
       k = 0;
       while (k == 0)
         if (fonaux[i+1] == ' ')
    //e final mudo
           { fonwrk[j] = 'i';
      k = 1;
           }//if
           else
             if ((fonaux[i+1]=='a')||
          (fonaux[i+1]=='i')||
                (fonaux[i+1]=='o'))
        k = 1;
             else
               if (fonwrk[j-1] == 'X')
               { fonwrk[j] = 'i';
                 j++;
                 k = 1;
               }//if
               else
                 if (fonaux[i+1] == 'R')
                   k = 1;
                 else
                   if (fonaux[i+1] == 'L')
                     k = 1;
                   else
             if (fonaux[i+1] != 'H')
                   { fonwrk[j] = 'i';
                     j++;
                     k = 1;
             }//if
                   else i++;
         }

}//for
```

```
        }//if
       }//else

      for (i = 0; i < component.elementAt(desloc).toString().length() + 3; i++)
     //percorre toda a palavra, letra a letra

      //i -> I
        if (fonwrk[i] == 'i')
          fonwrk[i] = 'I';
        else

          //a -> A
          if (fonwrk[i] == 'a')
          fonwrk[i] = 'A';
          else

           //o -> U
           if (fonwrk[i] == 'o')
            fonwrk[i] = 'U';
            else

             //s -> S
             if (fonwrk[i] == 's')
              fonwrk[i] = 'S';
              else

               //E -> b
               if (fonwrk[i] == 'E')
                fonwrk[i] = ' ';
                else

                 //Y -> _
                 if (fonwrk[i] == 'Y')
                  fonwrk[i] = '_';

     //retorna a palavra, modificada, ao vetor que contem o texto
       component.setElementAt(str.copyValueOf(fonwrk), desloc);
       j = 0; //zera o contador
     }//for

     str = vectorToStr(component);
     //remonta as palavras armazenadas no vetor em um unico string

     str = removeMultiple(str);
     //remove os caracteres duplicados

    return str.toUpperCase().trim();
        }



    public static String removePrep(String str) {
     int i,j;
      Vector palavra = new Vector();
      palavra = strToVector(str);
      String prep[] =
{"DEL","DA","DE","DI","DO","DU","DAS","DOS","DEU","DER","E","LA","LE","LES","LOS","VAN","VON","EL"};

     for (i = 0; i < palavra.size(); i++) {
        for (j = 0; j < prep.length; j++) {
         if (palavra.elementAt(i).toString().compareTo(prep[j]) == 0) {
             palavra.removeElementAt(i);
             i--;
           }
         }
       }
       return vectorToStr(palavra);
   }


        public static String removeMultiple (String str) {
  //Retira do texto carateres que estao multiplicados:
  // ss -> s, sss -> s, rr -> r

        char[] foncmp = new char[256];
        //matriz de caracteres que armazena o texto sem duplicatas

        char[] fonaux = new char[256];
        //matriz de caracteres que armazena o texto original

        char[] tip = new char[1]; //armazena o caracter anterior

        int i, j;  //contadores

        j = 0;
        tip[0] = ' ';
        fonaux = str.toCharArray();
        //a matriz de caracteres recebe o string original

       for (i = 0; i < str.length(); i++) {
       //percorre o texto, caracter a caracter

          //elimina o caracter se ele for duplicata e
          //nao for numero, espaco ou S
          if ((fonaux[i] != tip[0]) || (fonaux[i] == ' ')
              ||((fonaux[i]>='0') && (fonaux[i]<='9'))
              ||((fonaux[i]=='S')&&(fonaux[i-1]=='S')&&
```

```
            ( (i>1) && (fonaux[i-2]!='S')))) {
          foncmp[j] = fonaux[i];
          j++;
          }

          tip[0] = fonaux[i];
          //reajusta o caracter de comparacao
        }

        //o string recebe o texto sem duplicatas
      str = str.copyValueOf(foncmp);

        return str.trim();
      }//removeMultiple


  public static String removeAccentuation (String str) {
//Substitui os caracteres acentuados por caracteres nao acentuados

        char aux[] = new char[256];
        //matriz de caracteres onde o texto eh manipulado

        int i;  //contador

        aux = str.toCharArray();
        //matriz recebe o texto

        for (i = 0; i < str.length(); i++) {
        //percorre o texto, caracter a caracter

       switch (aux[i])
          { case 'É':
          aux[i]='E';  //É -> E
              break;
         case 'Ê':
              aux[i]='E';  //Ê -> E
              break;
            case 'Ë':
              aux[i]='E';  //Ë -> E
              break;
         case 'Á':
              aux[i]='A';  //Á -> A
          break;
            case 'À':
              aux[i]='A';  //À -> A
              break;
            case 'Â':
              aux[i]='A';  //Â -> A
              break;
            case 'Ã':
              aux[i]='A';  //Ã -> A
              break;
            case 'Ä':
              aux[i]='A';  //Ä -> A
              break;
            case 'Ç':
              aux[i]='C';  //Ç -> C
              break;
            case 'Í':
              aux[i]='I';  //Í -> I
              break;
            case 'Ó':
              aux[i]='O';  //Ó -> O
              break;
            case 'Õ':
              aux[i]='O';  //Õ -> O
              break;
            case 'Ô':
              aux[i]='O';  //Ô -> O
              break;
            case 'Ö':
              aux[i]='O';  //Ö -> O
              break;
            case 'Ú':
              aux[i]='U';  //Ú -> U
              break;
            case 'Ü':
              aux[i]='U';  //Ü -> U
              break;
            case 'Ñ':
              aux[i]='N';  //Ñ -> N
              break;
            }
        }
        str = str.copyValueOf(aux).trim();
      //o string recebe o texto sem acentuacao

      return str;
      }//removeAccentuation

  public static String removeStrange (String str) {
//Elimina os caracteres que NAO sejam alfanumericos ou espacos

        char[] foncmp = new char[256];
        //matriz de caracteres que armazena o texto original
```

```
  char[] fonaux = new char[256];
  //matriz de caracteres que armazena o texto modificado

  int i, j,  //contadores
      first;  //indica se exitem espacos em branco antes do primeiro
              //caracter: se 1 -> existem, se 0 -> nao existem

j = 0;
  first = 1;
  fonaux = str.toCharArray();
//matriz de caracteres recebe o texto

for (i = 0; i < 256; i++)
    foncmp[i] = ' ';
 //branqueia a matriz de caracteres

  for (i = 0; i < str.length(); i++) {
  //percorre o texto, caracter a caracter

    //elimina os caracteres que nao forem alfanumericos ou espacos
    if (((fonaux[i]>='A')&&
        (fonaux[i]<='Z')) ||
       ((fonaux[i]>='a')&&
        (fonaux[i]<='z')) ||
       ((fonaux[i]>='0')&&
         (fonaux[i]<='9')) ||
        (fonaux[i] == '&')  ||
        (fonaux[i] == '_')  ||
        ((fonaux[i] == ' ') && first == 0)) {
      foncmp[j] = fonaux[i];
   j++;
      first = 0;
     }//if
  }//for
  str = str.valueOf(foncmp);
//string recebe o texto da matriz de caracteres

return str.trim();
}//removeStrange


 public static Vector strToVector(String str) {
//armazena o texto de um string em um vetor onde
 //cada palavra do texto ocupa uma posicao do vetor

str = str.trim();

  char[] fonaux = new char[256];
//matriz de caracteres que armazena o texto completo

  char[] foncmp = new char[256];
  //matriz de caracteres que armazena cada palavra

  Vector component = new Vector();
  //vetor que armazena o texto

String aux = new String();

  int i, j,  //contadores
   pos,   //posicao da matriz
      rep,   //indica se eh espaco em branco repetido
      first; //indica se eh o primeiro caracter

first = 1;
  pos = 0;
  rep = 0;

  fonaux = str.toCharArray();
//matriz de caracteres recebe o texto

for (j = 0; j < 256; j++)
    foncmp[j] = ' ';
 //branqueia matriz de caracteres

  for (i = 0; i < str.length(); i++) {
  //percorre o texto, caracter a caracter

    //se encontrar um espaco e nao for o primeiro caracter,
    //armazena a palavra no vetor
    if ((fonaux[i] == ' ') && (first != 1)) {
   if (rep == 0) {
       component.addElement(aux.copyValueOf(foncmp).trim());
       pos = 0;
       rep = 1;
       for (j = 0; j < 256; j++)
         foncmp[j] = ' ';
     }//if
    }//if

 //forma a palavra, letra a letra, antes de envia-la a uma
    //posicao do vetor
    else {
      foncmp[pos] = fonaux[i];
   first = 0;
      pos++;
      rep = 0;
    }//else
  }//for
```

```
        if (foncmp[0] != ' ')
          component.addElement(aux.copyValueOf(foncmp).trim());

          return component;
        }//strToVector


        public static String vectorToStr(Vector vtr) {
      //converte o texto armazenado em um vetor para um unico string

          char[] foncmp = new char[256];
          //matriz de caracteres que armazena o texto completo

          char[] auxChar = new char[256];
          //matriz de caracteres que armazena cada palavra

        String auxStr = new String();
          String str = new String();
        int i, j, desloc;

          desloc = 0;  //deslocamento dentro da matriz

        for (i = 0; i < 256; i ++)
              foncmp[i] = ' ';
            //branqueia a matriz de caracteres

          for (j = 0; j < vtr.size(); j++) {
          //percorre o vetor, palavra a palavra

            auxStr = (vtr.elementAt(j)).toString().trim();
            //string recebe a palavra armazenada pelo vetor

            auxChar = auxStr.toCharArray();
            //matriz de caracteres recebe a palavra armazenada no vetor

         for (i = 0; i < auxStr.length(); i++)
         //percorre a matriz, caracter a caracter

             foncmp[desloc + i] = auxChar[i];
           desloc = desloc + auxStr.length() + 1;
         }//for

         str = str.valueOf(foncmp);
         //string recebe o texto completo

         return str.trim();

      }//vectorToStr
}
```



POSTS RELACIONADOS

- Artigo SQL Magazine 70 - Questões sobre banco de dados do concurso do Ipea – Parte 2
- Revista SQL Magazine Edição 56
- artigo SQL Magazine 11 - Ajustes de Desempenho em "Consultas Simples" na SQL
- Artigo SQL Magazine 37 - SGBDs free e as alternativas gratuitas da Microsoft, Oracle e IBM
- Artigo SQL Magazine 37 - SGBDs free e as alternativas gratuitas da Microsoft, Oracle e IBM
- Revista SQL Magazine Edição 37
- Artigo da SQL Magazine 34 - Banco de Dados Multimídia
- Artigo da SQL Magazine 33 - Migração de Oracle para Postgre SQL
- SGBD Oracle - Implementação de Função para Fonetização em Português
- Artigo da SQL Magazine 28 - Dúvidas freqüentes sobre bancos de dados

ÚLTIMOS POSTS DEVMEDIA [Ver todos]

- Revista Clube Delphi Edição 117
- Revista SQL Magazine Edição 75
- Count, Sum, AVG, Min, Max: Funções de agregação - Curso PostGreSQL: Treinamento em banco de dados - parte 14
- Inserindo itens nos cupons fiscais - Curso PAF - ECF com Delphi 7 e Dll's dos fabricantes - aplicação completa - Parte 9
- Layout: Evento Hover - Curso Chamados Técnicos com MultiCamadas - parte 7
- Criação de entidades com JPA 2 - Java EE 6 - Parte 18
- Testes Unitários: Exclusão e Alteração - Curso aplicação Web para Classificados com .NET Entity Framework - parte 10
- Testes Unitários: Listagem de itens - Curso aplicação Web para Classificados com .NET Entity Framework - parte 9
- Testes Unitários: Criando projeto para testes - Curso aplicação Web para Classificados com .NET Entity Framework - parte 8
- Alteração de itens - Curso aplicação Web para Classificados com .NET Entity Framework - parte 7
- Exclusão de itens - Curso aplicação Web para

Participe! Inclua um comentário