Kölner Phonetik

In a German speaking environment, you need a language phonetic search implementation which is different from Oracles default soundex implementation.

We use the Oracle PL/SQL implementation of "Kölner Phonetik" below.

```
CREATE OR REPLACE PACKAGE PCK_PHONETIK AS
/*
|| $Header: $
||
|| Name   : PKG_PHONETIK
|| Aufgabe: An implementation of "Koelner Phonetik"
|| Autor  : 13.02.2017, Jan Schreiber
||
*/

  FUNCTION GET_CODE (
      I_STRING               IN VARCHAR2)
    RETURN VARCHAR2;
END PCK_PHONETIK;
/

CREATE OR REPLACE PACKAGE BODY PCK_PHONETIK AS
/*-----------------------------------------------------------------------------
||
|| 2017.02.13 Jan Schreiber: An implementation of "Koelner Phonetik".
|| 2017.02.14 Jan Schreiber: Renamed to PCK_PHONETIK
|| 2917.02.16 Jan Schreiber: Aligned with Java class Cologne Phonetic
||
*/-----------------------------------------------------------------------------
/*
  This work is based on Andy Theilers github package: https://github.com/deezaster/germanphonetic
  and Carsten Czarskis APEX implementation: https://apex.oracle.com/pls/apex/germancommunities/apexcommunity
/tipp/1502/index.html
  and the Java Commons 1.7 implementation ColognePhonetics from: http://archive.apache.org/dist/commons/codec
/source/commons-codec-1.7-src.tar.gz

  Source code: https://wiki.loopback.org/confluence/x/HoDR
*/
  function soundex_ger (strword IN VARCHAR2,
                        intlen IN NUMBER DEFAULT 255)
  return VARCHAR2  is

     word varchar2(255);
     wordlen number;
     checklen number;
     code varchar2(255);
     phoneticcode varchar2(255);
     intX number;

   begin

        if intlen is null
        then
          checklen := 255;
        else
          checklen := intlen;
        end if;

        Word := lower(substr(strWord,0,checkLen));
        if length(Word) < 1 then return 0; end if;
        Word := REGEXP_REPLACE(
        REPLACE(
          REPLACE(
            Translate(Word, 'vwjyäöüéèêàáç', 'ffiiaoueeeaac'),
                'ph', 'f'),
              'ß', 'ss'),
```

```
    '[^a-zA-Z]', Null);
wordlen := length(word);
if wordlen = 1
then
  word := word || ' ' ;
end if;

 if substr(word,1,1) = 'c'
 then
   case
     when substr(word,2,1) in ('a','h','k','l','o','q','r','u','x') then
        Code := '4';
     else
        Code := '8';
   end case;
   intX := 2;
else
   intX := 1;
end if;

while intX <= wordlen loop
   case
     when substr(word,intX,1) in ('a','e','i','o','u') then
        code := code || '0';
     when substr(word,intX,1) = 'b' or  substr(Word,intX,1) = 'p' then
        code := code || '1';
     when substr(word,intX,1) = 'd' or  substr(Word,intX,1) = 't' then
        if intX < wordlen then
           case
             when substr(word,intX+1,1) in ('c','s','z') then
                code := code || '8';
             else
                code := code || '2';
           end case;
        else
           code := code || '2';
        end if;
     when substr(word,intX,1) = 'f' then
        code := code || '3';
     when substr(word,intX,1) in ('g','k','q') then
        code := code || '4';
     when substr(word,intX,1) = 'c' then
        if intX < wordlen then
           case
             when substr(Word,intX+1,1) in ('a','h','k','o','q','u','x') then
                case
                  when substr(Word,intX-1,1) = 's' or substr(Word,intX-1,1) = 'z' then
                     code := code || '8';
                  else
                     code := code || '4';
                end case;
             else
                code := code || '8';
           end case;
        else
           code := code || '8';
        end if;
     when substr(word,intX,1) = 'x' then
        if intX > 1 then
           case
             when substr(word,intX-1,1) in ('c','k','x') then
                code := code || '8';
             else
                code := code || '48';
           end case;
        else
           code := code || '48';
        end if;
     when substr(word,intX,1) = 'l' then
        code := code || '5';
     when substr(word,intX,1) = 'm' or substr(word,intX,1) = 'n' then
```

```
               code := code || '6';
             when substr(word,intX,1) = 'r' then
               code := code || '7';
             when substr(word,intX,1) = 's' or substr(word,intX,1) = 'z' then
               code := code || '8';
             else
               code := code;
          end case;

          intX := intX + 1;
       end loop;

       phoneticcode := translate(regexp_replace(code, '(.)\1+', '\1'), '1234567890', '123456789');

      if  substr(code,1,1) = '0'
      then
       phoneticcode := '0' || phoneticcode;
      end if;
     return phoneticcode;

   end soundex_ger;

   function get_code
    (i_string IN VARCHAR2)
    return VARCHAR2
    is
     len number;
     i number;
     k number;
     in_string varchar2(4000);
     out_string varchar2(4000);
     key_length constant number := 4;
   begin
    i := 1;
    out_string := null ;
    in_string := regexp_replace(substr(translate(i_string,'.,-;','    '),1,4000),  '([[:cntrl:]])|(^\t)', '
');
    len := length(in_string) ;
    while i <= len loop
      k := InStr(in_string, ' ', i);
      case
         when (k = i) then
          i := i +1;
         when (k > i) then
          out_string := trim(out_string || ' ' || substr(soundex_ger(substr(in_string, i, k-i)),1,key_length)) ;
          i := k+1 ;
         when (k = 0) then
          out_string := trim(out_string || ' ' || substr(soundex_ger(substr(in_string, i)),1,key_length)) ;
          i := len +1;
      end case;
    end loop;
    return out_string;
   end get_code;
END PCK_PHONETIK;
/
```

The implementation table used is build on the article in German Wikipedia mentioned above:

Letter	Context	Code
A, E, I, J, O, U, Y		0
H		-
B		1
P	not before H	1
D, T	not before C, S, Z	2

F, V, W		3
P	before H	3
G, K, Q		4
C	in Anlaut before A, H, K, L, O, Q, R, U, X	4
C	before A, H, K, O, Q, U, X , but not behind S, Z	4
X	not behind C, K, Q	48
L		5
M, N		6
R		7
S, Z		8
C	behind S, Z	8
C	in Anlaut, but not before A, H, K, L, O, Q, R, U, X	8
C	not before A, H, K, O, Q, U, X	8
D, T	before C, S, Z	8
X	after C, K, Q	8