```
-- Function: get_core_namewa(character varying, character varying, character varying,
  character varying, character varying, integer, integer)

-- DROP FUNCTION get_core_namewa(character varying, character varying, character varying,
  character varying, character varying, integer, integer);

-- select get_core_namewa('merged','id','trade_name','merged','core_merged_names',1377500,
  10) ;

--
CREATE OR REPLACE FUNCTION get_core_namewa(character varying, character varying, character
  varying, character varying, character varying, integer, integer)
  RETURNS character varying AS
'

/*  Sample invocation:  SELECT get_core_name(^business^,^id^,^conformedname^, ^LEGAL^, 0,
  10000) ;
                       Where ^ are actually apostrophes.


    ********************************************************************************
    ********************************************************************************
    ALERTS !   ALERTS !   ALERTS !

    An interm TABLE MUST EXIST:

    -- DROP TABLE possibilities_4_value ;
    * CREATE TABLE possibilities_4_value (mightbe character varying(500), run_key integer
      ) ;
    -- ------------------------------------------------------------------------------
    The final TABLE MUST EXIST:

    * CREATE TABLE core_sourcemoniker_names (sourceid bigint ,
                                  legal_origval varchar(120), legal_trimval varchar(120),
                                  public_origval varchar(120), public_trimval
                                    varchar(120),
                                  entrytimestamp timestamp without time zone,
                                  updatetimestamp timestamp without time zone default
                                    now() ) ;
    There is a plural in the tablename.

    -- ------------------------------------------------------------------------------
    The following two tables must exist:

    * CREATE TABLE suffixes ( chktext VARCHAR(60), sortorder smallint ) ;

    chktext     | sortorder
    ------------+----------
    CORPORATION | 11
    COMPANY     | 7       ETC  (sortorder = length)
```

```
51
52      * CREATE TABLE punctuation ( chkpunct VARCHAR(20), sortorder smallint, puncttype
53        VARCHAR(20) ) ;
54
55      chkpunct            | sortorder | puncttype
56      --------------------+-----------+----------
57       .,/~@#$%^&*-_+=<>` | 19        | after       <- Note that there is a preceeding space
58                          | 2         | b4             The sortorder value illuminates for
59                            you
60                          | 1         | b4             the number of spaces before or after
61       ,                  | 3         | b4             the punctuation.
62       ,                  | 2         | b4             Like above, sortorder is the # of
63         chars.
64       ,                  | 1         | b4
65        ,                 | 2         | b4
66       ,                  | 3         | b4             Add in the same way for a period.
67                                                      There should be 13 records, at least.
68
69
70      ALERTS !   ALERTS !   ALERTS !
71      ********************************************************************************
72
73      ********************************************************************************
74  select * from core_merged_names
75  where  (select case when m1>m2 then m1 else m2 end
76          from   (select extract(epoch from max(entrytimestamp))::INT  as m1
77                       , extract(epoch from max(updatetimestamp))::INT as m2
78                  from   core_merged_names
79                 ) x
80         )
81  in (extract(epoch from updatetimestamp)::INT, extract(epoch from entrytimestamp)::INT) ;
82
83  select count(*) from core_merged_names where sourceid <= 2188593;  -- = 520378
84  select count(*) from      merged       where       id <= 2188593;  -- = 1377499
85
86  select sourceid, orig_name from core_merged_names order by sourceid ;
87
88
89      ********************************************************************************
90
91      Because of the time it takes for this to run, it is best to call it via a shell
92        script.
93      See script named:  "get_core_name.sh".
94
95  */
96
97  DECLARE
98    TableName alias for $1 ;
99    KeyColName  alias for $2 ;
100   ValueColName  alias for $3 ;
```

```
  TblSrcTyp       alias for $4 ;
  TblDest         alias for $5 ;
  vOffset   alias for $6 ;
  vLimit    alias for $7 ;
  aRec    RECORD ;
  afield    varchar(120) ;
  origfield varchar(120) ;
  sid   bigint ;
  ComnWord  RECORD ;
  punctRecA RECORD ;
  PunctB4Token  RECORD ;
  token   varchar(120) ;
  punct_b4  varchar(20) ;
  punct_after varchar(20) ;
  chkstr    varchar(100) ;
  TableSrcType    varchar(20) ;
  TableDest       varchar(20) ;
  sufpos    INT ;
  aval    varchar(120) ;
  best_trimmed  varchar(120) ;
  aCount    INT DEFAULT 0 ;
  I   INT ;
  L               INT  DEFAULT 0 ;
  t   INT ;
  tl1   INT ;
  tl2   INT ;
  tl3   INT ;
  tl4   INT ;
  ExecuteString varchar(500) ;
  UpdateString  VARCHAR(500) ;
  UpdateStringGo  VARCHAR(500) ;
  InsertString  VARCHAR(500) ;
  InsertStringGo  VARCHAR(500) ;
  HaveAValue  INT ;
  SidFirst  INT ;
  SidLast   INT ;
  SkipCo          INT ;

  RightNow  TIMESTAMP WITHOUT TIME ZONE ;
  InitSecs0 INT DEFAULT 0 ;
  ElapSecs0 INT DEFAULT 0 ;
  InitSecs  INT DEFAULT 0 ;
  ElapSecs  INT DEFAULT 0 ;
  SumElapSecs1  INT DEFAULT 0 ;
  SumElapSecs2  INT DEFAULT 0 ;
  ElapSelectSecs  INT DEFAULT 0 ;
  PctCheckUpdate  INT DEFAULT 0 ;
  PctDoingUpdate  INT DEFAULT 0 ;


```

```
151
152 BEGIN
153
154
155   -- Debugging levels
156   t := 4 ;
157   tl1 := t ;
158   tl2 := tl1-1 ;
159   tl3 := tl2-1 ;
160   tl4 := tl3-1 ;
161
162   -- ---------------------------------------------------------------
163   TableDest := TblDest ;
164   TableSrcType := TblSrcTyp ;  -- No checking on this (yet)
165
166   IF upper(TableSrcType)=''MERGED'' OR lower(TableSrcType)=''merged'' THEN
167     TableSrcType=''MERGED'' ;
168   ELSIF upper(TableSrcType)=''SPDB'' OR lower(TableSrcType)=''spdb'' THEN
169     TableSrcType=''SPDB'' ;
170   ELSIF upper(TableSrcType)=''DDDB'' OR lower(TableSrcType)=''dddb'' THEN
171     TableSrcType=''DDDB'' ;
172   ELSE
173     RAISE EXCEPTION ''This SQL Script currently can only INSERT or UPDATE cleaned names
174       into core_xxxx_names for destinations designaged as "MERGED" or "SPDB" or "DDDB"
175       Name Types, (the 4th parameter).'' ;
176   END IF ;
177
178
179   IF TableDest=''core_merged_names'' THEN
180     InsertString := ''INSERT INTO core_merged_names '' ;
181     UpdateString := ''UPDATE core_merged_names SET orig_name = '' ;
182   ELSIF TableDest=''core_spdb_names'' THEN
183     InsertString := ''INSERT INTO core_spdb_names '' ;
184     UpdateString := ''UPDATE core_spdb_names SET orig_name = '' ;
185   ELSIF TableDest=''core_dddb_names'' THEN
186     InsertString := ''INSERT INTO core_dddb_names '' ;
187     UpdateString := ''UPDATE core_dddb_names SET orig_name = '' ;
188   END IF ;
189
190   InsertString := InsertString || ''(sourceid, orig_name, trim_name, entrytimestamp)
191                                VALUES ( '' ;
192
193   -- ---------------------------------------------------------------
194
195   SELECT INTO InitSecs0  secs_get() ;
196
197   select into punct_after chkpunct from punctuation where puncttype=''after'' ;
198
199   SELECT INTO ExecuteString lib__gen_execstring1(TableName, KeyColName, ValueColName,
200     vOffset, vLimit) ;
```

```
201
202   -- ========================================================================
203   -- ========================================================================
204   FOR aRec IN EXECUTE ExecuteString LOOP
205
206
207
208     aCount := aCount + 1 ;
209
210     sid    := aRec.ID  ;
211     afield := aRec.ValueCol;
212 --  PERFORM RNotice3(1,''aCount'',aCount::VARCHAR,''sid'',sid::VARCHAR,''afield'',afield)
213   ;
214
215     IF length(afield) != 0 THEN
216
217       origfield := afield ;
218       IF aCount = 1 THEN
219         SidFirst := sid ;
220         SELECT INTO ElapSelectSecs  secs_get_elapsed(InitSecs0) ;
221       ELSIF aCount = vLimit THEN
222         SidLast := sid ;
223       END IF ;
224
225       afield := trim(both '' '' from afield) ;
226       afield := trim(both '''''''' from afield) ;
227       afield := trim(both '' '' from afield) ;
228
229       INSERT INTO possibilities_4_value (run_key, mightbe) VALUES(InitSecs0,afield) ;
230
231 --  PERFORM Rnotice1(1,''Raw Data, afield'',afield) ;
232
233       -- ============================================================================
234       -- Check for punctuation that might just be hanging out at the end of the Name
235
236       aval := trim(trailing punct_after from afield) ;
237
238       if length(afield) != length(aval) then
239         INSERT INTO possibilities_4_value (run_key, mightbe) VALUES(InitSecs0,aval) ;
240         aval := trim(both '' '' from aval) ;
241         afield := aval ;
242       end if ;
243
244 --  PERFORM RNotice2(1,''Just after JUST punct at top'','' '',''afield'',afield) ;
245
246
247       -- ============================================================================
248
249       FOR I IN 1..2 LOOP          -- There are often multiple occurrances of punct b4
250         suffixes to remove
```

```
251
252         IF length(afield) != 0 THEN
253
254           -- ---------------------------------------------------------------
255           -- Loop through common words that might or might not be there, or
256           --   they are there but with other abbreviations
257
258           FOR ComnWord IN SELECT * FROM suffixes order by sortorder desc LOOP
259
260             token := ComnWord.chktext ;
261 --  PERFORM RNotice2(1,''I'',I::text::varchar,''token Checking'',token) ;  --
262
263             -- ---------------------------------------------------------------
264             -- Check for the text that might be a necessary part of the company name
265
266             SkipCo := 0 ;
267             IF 1 = 2 THEN                -- We do not want to do this for WA because did
268               not for SPDB or DDDB
269             L := length(afield) ;
270             IF substr(afield,L-4+1) = ''& CO'' OR substr(afield,L-9+1) = ''& COMPANY''
271               THEN
272               SkipCo := 1 ;
273             END IF ;
274             END IF ;
275
276             IF NOT (SkipCo = 1 AND (token = ''CO'' OR token = ''COMPANY'')) THEN
277
278               -- ---------------------------------------------------------------
279               -- Loop through the combinations of periods and commas and spaces
280               --   looking for them as part of STUFF before the common text to eliminate
281
282               FOR PunctB4Token IN SELECT * FROM punctuation where puncttype = ''b4'' order
283                 by sortorder desc LOOP
284
285                 punct_b4 := PunctB4Token.chkpunct ;
286                 aval := '''' ;
287
288 --  PERFORM Rnotice3(1,''I'',I::VARCHAR,''token'',token,''punct_b4'',punct_b4) ;
289
290                 chkstr := punct_b4 || token ;
291
292                 IF substring(afield from length(afield)-length(chkstr)+1) = chkstr THEN
293
294                   aval := substr(afield,1,length(afield)-length(chkstr)) ;
295                   aval := trim(both '' '' from aval) ;
296                   afield := aval ;
297
298                   aval := trim(trailing punct_after from afield) ;
299                   IF length(afield) != length(aval) THEN
300                     afield := aval ;
```

```
                END IF  ;

                INSERT INTO possibilities_4_value (run_key, mightbe)
                  VALUES(InitSecs0,aval) ;

              END IF ;       -- We have above some punctuation before the suffix word

            END LOOP ;       -- EOL Punctuations before suffix word

          END IF ;           -- Skip the special case where we want to keep the token

--  PERFORM RNotice2(1,''i'',i::TEXT::VARCHAR,''afield'',afield) ;

          -- ------------------------------------------------------------
          --  JUST the core mightbe the token w/o looking for punctuation before it

          IF I = 1 AND SkipCo = 0 THEN

            aval=''''  ;
            chkstr := token  ;
            sufpos=strpos(afield,chkstr) ;

            IF sufpos != 0 AND sufpos = length(afield)-length(chkstr)+1 THEN

              aval := substr(afield,1,sufpos-1) ;
              aval := trim(both '' '' from aval) ;
              afield := aval ;
              INSERT INTO possibilities_4_value (run_key, mightbe)
                VALUES(InitSecs0,aval) ;

            END IF ;      -- Get the suffix word by itself out of the field

          END IF ;        -- Skip because of special combination

        END LOOP ;        -- FOR tokens

      END IF ;            -- If have length


      -- ========================================================================
      -- Check for the after-mightbe punctuation once more

      aval := trim(trailing punct_after from afield) ;
      aval := trim(both '' '' from aval) ;

      if length(afield) != length(aval) then
        INSERT INTO possibilities_4_value (run_key, mightbe) VALUES(InitSecs0,aval) ;
        afield := aval ;
      end if ;

```

```
351       END LOOP ;          -- I LOOP
352
353
354       -- ============================
355
356 --  PERFORM rnotice2(1,''afield'',afield,''length'',length(afield)::TEXT::VARCHAR) ;
357
358       SELECT INTO best_trimmed         mightbe
359       FROM          possibilities_4_value
360       WHERE         run_key = InitSecs0
361       AND           length(mightbe) = ( SELECT MIN(length(mightbe)) FROM
362         possibilities_4_value )
363       LIMIT 1 ;  -- <===  THIS IS NOT GREAT -- We might have a better one of = length
364
365       DELETE FROM possibilities_4_value WHERE run_key = InitSecs0 ; -- vacuum full
366         possibilities_4_value ;
367
368       -- ====================================
369
370       SELECT INTO InitSecs  secs_get() ;
371
372       IF TableDest = ''core_merged_names'' THEN
373         SELECT INTO HaveAValue 1 FROM core_merged_names WHERE sourceid = sid ;
374       ELSIF TableDest = ''core_spdb_names''   THEN
375         SELECT INTO HaveAValue 1 FROM core_spdb_names   WHERE sourceid = sid ;
376       ELSIF TableDest = ''core_dddb_names''   THEN
377        SELECT INTO HaveAValue 1 FROM core_dddb_names   WHERE sourceid = sid ;
378       END IF ;
379
380       SELECT INTO ElapSecs  secs_get_elapsed(InitSecs) ;
381       SumElapSecs1 := SumElapSecs1 + ElapSecs ;
382       RightNow := timeofday()::TIMESTAMP WITHOUT TIME ZONE ;
383
384       -- select get_core_namewa(''raw_special_databases'',''id'', ''bus_name'',
385         ''spdb'',0,10)
386       -- select get_core_namewa(''raw_dddb'',                ''id'',
387         ''companyname'',''dddb'',0,10)
388
389 --  PERFORM
390   Rnotice3(1,''sid'',sid::TEXT::VARCHAR,''origfield'',origfield,''best_trimmed'',
391   best_trimmed) ;
392
393       IF HaveAValue IS NULL THEN
394
395         InsertStringGo := InsertString ||  sid || '', '' || quote_literal(origfield) ||
396           '', '' || quote_literal(best_trimmed) || '', '' || quote_literal(RightNow) || ''
397           ) ; '' ;
398
399 --  PERFORM Rnotice1(1,''InsertStringGo'',InsertStringGo) ;
400
```

```
401         EXECUTE InsertStringGo ;
402
403       ELSE
404
405         UpdateStringGo :=  UpdateString || quote_literal(trim(origfield)) || ''
406                         , trim_name = '' || quote_literal(best_trimmed) || ''
407                 , updatetimestamp = '' || quote_literal(RightNow) || ''
408                   WHERE sourceid =  '' || sid || '' ; '' ;
409
410 --  PERFORM Rnotice1(1,''UpdateStringGo'',UpdateStringGo) ;
411
412         SELECT INTO InitSecs  secs_get() ;
413
414         EXECUTE UpdateStringGo ;
415
416         SELECT INTO ElapSecs  secs_get_elapsed(InitSecs) ;
417         SumElapSecs2 := SumElapSecs2 + ElapSecs ;
418
419       END IF ;    -- Check whether to do an insert or an update
420
421     END IF ;      -- We started with something
422
423     --  ================================================================================
424
425   END LOOP ;        -- afield
426
427   SELECT INTO ElapSecs0  secs_get_elapsed(InitSecs0) ;
428
429   IF ElapSecs0 = 0 THEN
430     PctCheckUpdate := 0 ;
431     pctDoingUpdate := 0 ;
432   ELSE
433     PctCheckUpdate := round(100*SumElapSecs1/ElapSecs0, 0) ;
434     PctDoingUpdate := round(100*SumElapSecs2/ElapSecs0, 0) ;
435   END IF ;
436
437
438   RETURN ''Cleaned '' || ValueColName || '' values for rows '' || vOffset || '' to '' ||
439     vOffset+vLimit || '', '' || KeyColName || '' values from '' || SidFirst || '' to '' ||
440     SidLast || ''.  SELECTing the data took '' || ElapSelectSecs || ''seconds, looking for
441     updates used '' || PctCheckUpdate || ''% and performing updates was '' ||
442     PctDoingUpdate || ''% of '' || ElapSecs0 || '' total seconds.'' ;
443
444 --END ; -- The Fantom END for BEGIN for function get_core_name()
445
446 END ;  '
447   LANGUAGE 'plpgsql' VOLATILE;
448
449 -- select
450   get_core_namewa('merged','id','trade_name','merged','core_merged_names',853350,5) ;
```