

---

Subject: Re: IDL & Sybase

Posted by [oet](#) on Fri, 04 Jun 1993 06:35:08 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

In article 677@physc1.byu.edu, goblec@physc1.byu.edu () writes:

> I am looking for some IDL code that can conduct Sybase queries and  
> return the results in IDL variables. We are running IDL on a  
> Sun. A front end for Sybase from IDL would be nice also. Does  
> anyone know of anyway to do this or programs that are available  
> from the net?  
> Thanks  
>  
> Clark Goble  
> goblec@theory.byu.edu  
>

Some times ago I wrote a simple function to access the Empress database. It is not for operational puroposes, only to get a data set from the database into an IDL Structure Array for further processing. For more sophisticated access try to write an RPC-Client which access the database and sends to data to an IDL started as RPC-Server. I've done something like this with the INGRES-Database. If you're interested in let me know.

Thomas

--  
|Thomas Oettli                Internet: Thomas.Oettli@sma.ch    |  
|Swiss Meteorological Institute       thomas.oettli@active.ch |  
|Kraehbuehlstr. 58            CServe: 100015.3543@compuserve.com |  
|8044 Zuerich                |

FUNCTION EMPCMD, database, sql\_statement, DEBUG = DEBUG  
;  
:+  
; NAME:  
; EMPCMD  
;  
; PURPOSE:  
;     Interface to the EMPRESS Database System. Loads output of dynamically  
;     submitted SQL-Statements into IDL-Structure-Array for further  
;     processing.  
;  
; CATEGORY:  
;     Database Access  
;  
; CALLING SEQUENCE:  
;     emp\_table=empcmd(database,SQL\_statement,[/DEBUG])  
;

```
; INPUTS:  
;  
; OPTIONAL INPUTS:  
;  
; KEYWORD PARAMETERS:  
;   DEBUG      if set time steps will be printed for performance  
;               analysis  
;  
; COMMENTS:  
;  
;  
; OUTPUTS:  
;  
; OPTIONAL OUTPUTS:  
;  
; COMMON BLOCKS:  
;  
; SIDE EFFECTS:  
;  
; RESTRICTIONS:  
;  
; PROCEDURE:  
;  
; EXAMPLE:  
;  
; SEE ALSO:  
;   imhelp  
;  
; MODIFICATION HISTORY:  
;  
;   IDLMETEO-Library  Swiss Meteorlogical Institute  
;  
;   Written by: Thomas.Oettli@sma.ch, 21-Dec-1992  
;  
;-
```

```
if keyword_set(DEBUG) then t0=systime(1)  
  
wordarray,sql_statement, tmparr          ; get first table definitions  
table_name = tmparr(where(tmparr EQ 'from')+1)  
msvalsep = getenv("MSVALSEP")  
descarr=strarr(1)                      ;  
  
if strpos(sql_statement, ' * ') EQ -1 then begin  
  attrarr=split(',',strcompress(strsub(sql_statement, $  
                                strpos(strupcase(sql_statement),'ECT ')+4, $  
                                strpos(strupcase(sql_statement),' FROM')-1), $  
                                /REMOVE_ALL),'s')
```

```

endif else attrarr=['*']

table_dict = { _table_dict, attr_name: string(replicate(32b,32)), $  

               attr_dname: string(replicate(32b,16))}

get_struct ='display '+ table_name + ' dump'  

spawn, ['empcmd',database,get_struct],tmparr,COUNT=sql_count, /NOSHELL

dict_table = replicate(table_dict, sql_count)

for i=0, n_elements(tmparr)-1 do begin  

  rowarr=split(msvalsep,tmparr(i),'s')

  dict_table(i).attr_name = rowarr(2)

  CASE rowarr(3) OF  

    'char': dict_table(i).attr_dname = 'A(1)'  

    'dec': dict_table(i).attr_dname = 'F'  

    'float': dict_table(i).attr_dname = 'F'  

    'integer': dict_table(i).attr_dname = 'I'  

  ELSE: print, 'Datatype conversion for this type currently not available!'  

  ENDCASE

endfor

if attrarr(0) NE '*' then begin  

  for i=0, n_elements(attrarr)-1 do begin  

    if i EQ 0 then descarr(0) = dict_table(where(dict_table.attr_name EQ attrarr(i))).attr_dname $  

      else descarr=push(descarr,dict_table(where(dict_table.attr_name EQ attrarr(i))).attr_dname)

  endfor
endif else begin  

  for i=0, n_elements(dict_table)-1 do begin  

    if i EQ 0 then begin  

      attrarr(0) = dict_table(0).attr_name  

      descarr(0) = dict_table(0).attr_dname  

    endif else begin

```

```

attrarr=push(attrarr, dict_table(i).attr_name)
descarr=push(descarr, dict_table(i).attr_dname)
endelse
endfor
endelse

if keyword_set(DEBUG) then begin
    print, 'Time for initializing, definition of datatypes (Sec): ', $
        systime(1)-t0
    t0=systime(1)
endif

sql_statement = sql_statement + ' dump'           ; set to dump mode

                                ; loading data into
                                ; array of rows

spawn, ['empcmd', database, sql_statement],tmparr, $
        COUNT=sql_count, $
        PID=sql_pid, /NOSHELL

if keyword_set(DEBUG) then begin
    print, 'Time for loading data into row array (Sec): ', $
        systime(1)-t0
    t0=systime(1)
endif

strname = 'tmp'+ strcompress(string(sql_pid),/REMOVE_ALL)

create_struct, emp_table, strname ,attrarr, $
        descarr, $
        DIMEN=sql_count

for i=0, n_elements(tmparr)-1 do begin
    rowarr=split(msvalsep,tmparr(i),'s')
    for si=0, n_tags(emp_table)-1 do begin      ; tag_names are in relation
        ; to descarr, so we use the
        ; same counter

        CASE dict_table(si).attr_dname OF          ; check for correct type
            'A(1)': emp_table(i).(si) = rowarr(si)   ; conversion corresponding
                                              ; to the description array
            'F':  emp_table(i).(si) = float(rowarr(si))
            'I':  emp_table(i).(si) = fix(rowarr(si))

```

```

ELSE: print, 'Datatype conversion for this type currently not available!'
ENDCASE
endfor
      ; 1 row inserted into the emp_table (idl structure array)
endfor

if keyword_set(DEBUG) then begin
  print, 'Time for loading data from row array into struc array (Sec): ', $
    systime(1)-t0
  t0=systime(1)
endif

return, emp_table

```

END

---

```

;
;+
; NAME:
;   SPLIT
;
; PURPOSE:
;   Create dynamically arrays from a String, needs getwrd from
;   the fermi.jhuapl.edu - library as subfunction
;   This function is a rewriting of the same function in PERL for
;   IDL. A difference is, that we have to define the type of the
;   returned array
;
; CATEGORY:
;   String Processing
;
; CALLING SEQUENCE:
;   arr = split('<delimiter>', <stringvar>, '[s|i|f]');
; INPUTS:
; delimiter = any character defined as field separator
;   stringvar = text string to extract from.
;   [s|i|f]  = type definition for returned array:
;     's' = strarr() (string array)
;     'i' = intarr() (integer array)
;     'f' = fltarr() (floatarray)
;     (width of array will be generated automatically
;     by counting the delimiters)
;
; KEYWORD PARAMETERS:
; OUTPUTS:
;   arr = array of strings, integers or floats as defined in

```

```

;      third parameter
;COMMON BLOCKS:
;NOTES:
;      Needs the function getwrd.pro, which was added to idlmeteo from
;      the JHU-APL-Library
;
;EXAMPLES:
;IDL>txt='hello|world|we|are|ready'
;IDL>gagarr=split('|',txt,'s')
;IDL>print,gagarr(1)
;world
;
;IDL>txt='12|3|56|8'
;IDL>gagarr=split('|',txt,'i')
;IDL>print,gagarr(3)
;8
;IDL>print, gagarr
;12 3 56 8
;
;MODIFICATION HISTORY:
;Th. Oettli, 31 Jul, 1992 ---- Added CATEGORY definition 'Array'
;for administration of IDL-libraries
;
;Copyright (C) 1992, Thomas Oettli Swiss Meteorological Institute
;This software may be used, copied, or redistributed as long as it is not
;sold and this copyright notice is reproduced on each copy made. This
;routine is provided as is without any express or implied warranties
;whatsoever.
;-
;-----

```

## FUNCTION SPLIT, DELIM, TXTSTR, TYPEDEF

```

if (n_params(0) ne 3) then begin
  print," Creates a [strarr|intarr|fltarr] from within a single string"
  print," Syntax: arr = split('<delimiter>', <stringvar>, '[s|i|f]')
  print,''
  print," delimiter = any character defined as field separator"
  print," stringvar = text string to extract from."
  print," [s|i|f] = type definition for returned array:"
  print,"           's' = strarr() (string array)"
  print,"           'i' = intarr() (integer array)"
  print,"           'f' = fltarr() (float array)"
  print,''
  print,' NOTE: Needs getwrd from the'
  print,'      fermi.jhuapl.edu - library as subfunction'

return, -1

```

```

endif

delim_byte=byte(delim)
txtstr_byte=byte(txtstr)
delim_count=0;
for i=0,(strlen(txtstr)-1) do begin
  if (txtstr_byte(i) eq delim_byte(0)) then begin
    delim_count = delim_count+1
  endif
endfor

if (TYPEDEF eq 's') then begin
  retarr = strarr(delim_count+1)
endif
if (TYPEDEF eq 'i') then begin
  retarr = intarr(delim_count+1)
endif
if (TYPEDEF eq 'f') then begin
  retarr = fltarr(delim_count+1)
endif

for i=0,delim_count do begin
  retarr(i)=getwrd(txtstr,i,delimiter=delim)
endfor

return, retarr

```

END

---

```

-----
;

;+
; NAME:
;   GETWRD
;
;
; PURPOSE:
;   Return the n'th word from a text string.
;
; CATEGORY:
;   String Processing
;
; CALLING SEQUENCE:
;   wrd = getwrd(txt, n, [m])
; INPUTS:
;   txt = text string to extract from.      in

```

```

; n = word number to get (first = 0 = def).  in
; m = optional last word number to get.    in
; KEYWORD PARAMETERS:
; Keywords:
;   LOCATION = l. Return word n string location.
;   DELIMITER = d. Set word delimiter (def = space).
;   /LAST means n is offset from last word. So n=0 gives
;     last word, n=-1 gives next to last, ...
;     If n=-2 and m=0 then last 3 words are returned.
;   /NOTRIM suppresses whitespace trimming on ends.
; OUTPUTS:
;   wrd = returned word or words.          out
; COMMON BLOCKS:
;   getwrd_com
; NOTES:
;   Note: If a NULL string is given (txt="") then the last string
;         given is used. This saves finding the words again.
;   If m > n wrd will be a string of words from word n to
;         word m. If no m is given wrd will be a single word.
;   n<0 returns text starting at word abs(n) to string end
;   If n is out of range then a null string is returned.
;   See also nwrds.
; MODIFICATION HISTORY:
;   Ray Sterner, 6 Jan, 1985.
;   R. Sterner, Fall 1989 --- converted to SUN.
;   R. Sterner, Jan 1990 --- added delimiter.
;   R. Sterner, 18 Mar, 1990 --- added /LAST.
;   R. Sterner, 31 Jan, 1991 --- added /NOTRIM.
; R. Sterner, 20 May, 1991 --- Added common and NULL string.
; Th. Oettli, 31 Aug, 1992 ---- Added CATEGORY definition 'Array'
;   for administration of IDL-libraries
;   Added to idlmeteo from the JHU-APL-Library nov-1992 oet@sma.ch
;
; Copyright (C) 1985, Johns Hopkins University/Applied Physics Laboratory
; This software may be used, copied, or redistributed as long as it is not
; sold and this copyright notice is reproduced on each copy made. This
; routine is provided as is without any express or implied warranties
; whatsoever. Other limitations apply as described in the file disclaimer.txt.
;-
;----- --
;
```

FUNCTION GETWRD, TXTSTR, NTH, MTH, help=hlp, location=ll,\$  
 delimiter=delim, notrim=notrim, last=last

common getwrd\_com, txtstr0, nwds, loc, len

if (n\_params(0) lt 1) or keyword\_set(hlp) then begin  
 print, " Return the n'th word from a text string."

```

print,' wrd = getwrd(txt, n, [m])'
print,' txt = text string to extract from.      in'
print,' n = word number to get (first = 0 = def). in'
print,' m = optional last word number to get.   in'
print,' wrd = returned word or words.        out'
print,' Keywords:'
print,' LOCATION = l. Return word n string location.'
print,' DELIMITER = d. Set word delimiter (def = space).'
print,' /LAST means n is offset from last word. So n=0 gives'
print,' last word, n=-1 gives next to last, ...'
print,' If n=-2 and m=0 then last 3 words are returned.'
print,' /NOTRIM suppresses whitespace trimming on ends.'
print,' Note: If a NULL string is given (txt="") then the last string'
print,' given is used. This saves finding the words again.'
print,' If m > n wrd will be a string of words from word n to'
print,' word m. If no m is given wrd will be a single word.'
print,' n<0 returns text starting at word abs(n) to string end'
print,' If n is out of range then a null string is returned.'
print,' See also nwrds.'
return, -1
endif

```

if n\_params(0) lt 2 then nth = 0 ; Def is first word.  
IF N\_PARAMS(0) LT 3 THEN MTH = NTH ; Def is one word.

```

if strlen(txtstr) gt 0 then begin
  ddel = '' ; Def del is a space.
  if n_elements(delim) ne 0 then ddel = delim ; Use given delimiter.
  TST = (byte(ddel))(0) ; Del to byte value.
  X = BYTE(TXTSTR) NE TST ; Non-delchar (=words).
  X = [0,X,0] ; 0s at ends.

  Y = (X-SHIFT(X,1)) EQ 1 ; Diff=1: word start.
  Z = WHERE(SHIFT(Y,-1) EQ 1) ; Word start locations.
  Y2 = (X-SHIFT(X,-1)) EQ 1 ; Diff=1: word end.
  Z2 = WHERE(SHIFT(Y2,1) EQ 1) ; Word end locations.

```

```

txtstr0 = txtstr ; Move string to common.
NWDS = TOTAL(Y) ; Number of words.
LOC = Z ; Word start locations.
LEN = Z2 - Z - 1 ; Word lengths.
endif else begin
  if n_elements(nwds) eq 0 then begin ; Check if first call.
    print,' Error in getwrd: must give a '+$'
    'non-NULL string on the first call.'
    return, -1 ; -1 = error flag.
  endif
endelse

```

```

if keyword_set(last) then begin ; Offset from last.
  lst = nwds - 1
  in = lst + nth ; Nth word.
  im = lst + mth ; Mth word.
  if (in lt 0) and (im lt 0) then return, " ; Out of range.
  in = in > 0 ; Smaller of in and im
  im = im > 0 ; to zero.
  if (in gt lst) and (im gt lst) then return," ; Out of range.
  in = in < lst ; Larger of in and im
  im = im < lst ; to be last.
  ll = loc(in) ; Nth word start.
  return, rtrim(strmid(txtstr0,ll,loc(im)-loc(in)+len(im)), 2)
endif

```

```

N = ABS(NTH) ; Allow nth<0.
IF N GT NWDS-1 THEN RETURN," ; out of range, null.
ll = loc(n) ; N'th word position.
IF NTH LT 0 THEN GOTO, NEG ; Handle nth<0.
IF MTH GT NWDS-1 THEN MTH = NWDS-1 ; Words to end.

```

```

if keyword_set(notrim) then begin
  RETURN, STRMID(TXTSTR0,ll,LOC(MTH)-LOC(NTH)+LEN(MTH))
endif else begin
  RETURN, rtrim(STRMID(TXTSTR0,ll,LOC(MTH)-LOC(NTH)+LEN(MTH)), 2)
endelse

```

```

NEG: if keyword_set(notrim) then begin
  RETURN, STRMID(TXTSTR0,ll,9999)
endif else begin
  RETURN, rtrim(STRMID(TXTSTR0,ll,9999), 2)
endelse

```

END

---