
Subject: Re: IDL and SQL

Posted by [Andy Loughe](#) on Fri, 08 Nov 2002 19:36:17 GMT

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

M. Katz wrote:

- > Has anyone out there made any efforts to link IDL to an SQL database?
- > Even if the actions were write-only from IDL, I'm interested.
- >
- > Thanks,
- >
- > M. Katz

Here's something I put together once to read output from MySQL into IDL.

Nothing very fancy... very brute force...

```
function get_mysql_vars, datain, heading, floatit=floatit,  
longit=longit, info=info
```

```
;  
;  
; Obtain variable names from a MySQL database query, and  
; store the associated values into identically named  
; variables within an IDL data structure.  
;  
;  
; Originator: Andrew F. Loughe :: 11 OCT 2000  
;  
;  
; ASSUMPTIONS:  
; 1) That the MySQL query results in a 2-D table sent to STDOUT.  
; 2) That the command issued was mysql -t ('|' delimited data).  
;  
;  
; PARAMETER:  
; datain : Parameter containing the table of actual data.  
; heading : Parameter containing the variables to process (MySQL table  
heading).  
;  
;  
; KEYWORDS:  
; floatit : converts those variables listed, to type=float.  
; longit : converts those variables listed, to type=long.  
; info : only return the variable names (MySQL table heading).  
;  
;
```

```
on_error, 2
```

```
;; start_time = systime(1)      ; Let's do some MySQL-like timing  
of the read
```

```

usage="data = get_mysql_vars(data, heading, floatit='area, yy',
longit='id', /info)"
if (N_params() lt 2)          then message, usage

num_vars = N_elements(heading) ; Number of variables (fields) to process
num_rows = N_elements(datain)  ; Number of rows (records) in the table
snum_rows = strtrim(num_rows, 2)
if (num_rows ge 100000L) then print, '**** ' + strtrim(num_rows,2) + $
                                ' records may take awhile ****'

; Announce to the user which variables are being processed.
format = "(a, a, " + strtrim(num_vars-1,2) + "(", ',a))"
print, '---> Database variables(' + strtrim(num_vars,2) + '): ', $
      heading, format=format
if (KEYWORD_SET(info)) then begin
  print, '**** NOTHING RETURNED. INFO. ONLY!!!'
  return, -999
endif

; Find location of all '|' characters and create a format statement for
reads.
test_str = datain(0)          ; Create format for: record = datain(0)
test_len = strlen(test_str)

; Store output STRING variables into a DATA STRUCTURE called dataS.
; Then copy results into a structure which can contain LONGS and FLOATS
(dataout).
cmd1      = 'dataS = {'      ; Structure used with reads
cmd2      = 'dataout = {'    ; Structure for mapping dataS --->
dataout
convert_cmd = "              ; Command for mapping dataS ---> dataout

for ii = 0L, num_vars-1L do begin ; Loop through all the variables
  str_var = heading(ii)        ; Get current variable name to process

; Initially, read all data as STRINGS. Make command to create the
structure of strings.
  format_type = 'a'
  cmd1        = cmd1 + str_var + ":' '"
  if (ii lt num_vars-1L) then cmd1 = cmd1 + ','
  if (ii eq num_vars-1L) then cmd1 = cmd1 + '}' ; data: All string reads

; Convert some data to longs and floats? Make command to create new
structure (dataout).
  iv = where( str_var eq longit, countlng ) ; dataout: longs?
  iv = where( str_var eq floatit, countflt ) ; dataout: floats?
  case 1 of
    (countlng gt 0): begin      ; LONGS into dataout

```

```

        cmd2      = cmd2 + str_var + ':lonarr(' +
snum_rows + ')'
        convert_cmd = convert_cmd+'dataout.'+str_var+'='+ $
            'long(TEMPORARY(dataS.' + str_var
+ '))'
        end
        (countflt gt 0): begin      ; FLOATS into dataout
            cmd2      = cmd2 + str_var + ':fltarr(' +
snum_rows + ')'
            convert_cmd = convert_cmd+'dataout.'+str_var+'='+ $
                'float(TEMPORARY(dataS.' + str_var
+ '))'
        end
    else:      begin      ; STRINGS into dataout
        cmd2      = cmd2 + str_var + ':strarr(' +
snum_rows + ')'
        convert_cmd = convert_cmd+'dataout.'+str_var+'='+ $
            'strtrim(TEMPORARY(dataS.' +
str_var + '),2)'
    end
endcase
if (ii lt num_vars-1L) then cmd2      = cmd2 + ','      ;
TERMINATORS
if (ii eq num_vars-1L) then cmd2      = cmd2 + '}'
if (ii lt num_vars-1L) then convert_cmd = convert_cmd + ' & '

; Find position of all '|' characters, and create the necessary format
statement.
if (ii eq 0) then begin
    pos      = 0
    tab_pos = pos + 1
    formats = format_type
    format = '(2x,'      ; Initial format string (will be built upon)
endif else begin
    pos      = strpos( test_str, '|', pos+1)
    if ( pos ge 0 ) then tab_pos = [ [tab_pos], pos+1 ]
    formats = [ [formats], format_type ]
    format =
format+formats(ii-1)+strtrim(tab_pos(ii)-tab_pos(ii-1)-3,2)+ ',3X,'
endelse
endifor
format = format + format_type + strtrim(test_len-tab_pos(ii-1)-3,2) + ',2x)'

; Replicate the STRING data structure, and read the data, then map
dataS --> dataout
jnk  = execute(cmd1)      ; Execute cmd1 to create data
structure (dataS)
dataS = replicate( dataS, num_rows ) ; Create array of structures (dataS)

```

```
reads, datain, dataS, format=format    ; Read dataS string structure.
jnk  = execute(cmd2)                   ; Create a structure of arrays
(dataout)
jnk  = execute(convert_cmd)            ; Map dataS --> dataout

; Print MySQL-like end message.
;; print, strtrim(num_vars,2) + ' variables, each with ' + snum_rows + $
;; ' elements (' +
strtrim(string(systime(1)-start_time,format='(f29.2)'),2) + ' sec)'

return, dataout    ; Return the output data structure to the calling routine.

end
```
