Dynamic Internal table

Created by Anonymous, last modified by Sandra Rossi on Nov 03, 2009

Dynamic internal Table A Dynamic Internal Table is an internal table with variable number of rows and columns, which can be defined during run time only.

A dynamic internal table is not declared in the program as static.

Some of the benefits of Dynamic internal table are:

- Flexiblilty
- Extremely useful when the numbers of columns / fields are not known at the design time / compile time.
- Avoids redundancy

Some of the important attributes of an internal table which can be set dynamically are

- Column position
- Field name of internal table field
- Column width
- Reference field
- Reference table
- Data type of the fields
- Domain name of the fields
- Check table etc.

(For the remaining attributes please refer the ABAP structure LVC_S_FCAT) Prerequisites of dynamic internal table are

Knowledge of field symbols

Knowledge on data references

Field symbols are placeholders or symbolic names for other fields. They are similar to dereference pointers in C.

Field symbols allow you to:

Assign an alias to a data object

Adopt or change the type and size of a data object dynamically at runtime

Set the offset and length for a string variably at runtime

Set a pointer to a data object that you determine at runtime (dynamic ASSIGN)

Access components of a structure

The statement ASSIGN f to <fs> assigns the field f to field symbol <fs>. The field symbol <fs> then "points" to the contents of field f at runtime. This means that all changes to the contents of f are visible in <fs> and vice versa.

You declare the field symbol <fs> using the statement FIELD-SYMBOLS: <fs>.

Data references are pointers to data objects.

You can use data references to create data objects dynamically. You can also create references to existing data objects.

You can only dereference a data reference using a special assignment to a field symbol.

You can create a data reference variable by using:

DATA <dref> TYPE REF TO DATA.

To create a data object dynamically during a program, you need a data reference variable and the following statement:

CREATE DATA <dref> TYPE <type>|LIKE <obj>.

To access the contents of the data object to which a data reference is pointing, you must dereference it.

ASSIGN <dref>->* TO <FS>. Dynamic internal tables can be created using :

- The class CL_ALV_TABLE_CREATE and the method CREATE_DYNAMIC_TABLE.

  Export parameter: it_fieldcatalog

  Import parameter: ep_table

  Exceptions: generate_subpool_dir_full = 1

  Others = 2 **Structure for Dynamic Internal Table Creation**

- ABAP Structure LVC_S_FCAT

  This structure is used to maintain the attributes of each field of the Dynamic

  Internal Table such as the fieldname, column position, etc.

- ABAP Table type LVC_T_FCAT

  This table type has a line type of LVC_S_FCAT. The field attributes of all the

  Fields are maintained in this table Steps to create a Dynamic Internal Table

1. **Data Definitions**

Declare a structure of type lvc_s_fcat.

Declare an internal table of type lvc_t_fcat (The line type of this internal table is lvc_s_fcat).

Declare two data reference variables, one for the dynamic internal table (say dr1) and the other for the work area (say dr2)

Declare field symbols of type 'ref to data', 'any table' and of type 'any' (say fs1, fs2 and fs3 respectively). # Populate the internal table with fieldnames required for the dynamic internal table

a).Assign the field name, field type, field width, check table etc. to the structure.

b).Append the structure to the internal table # Assign Field-Symbol to dynamic internal table

i.e. Assign dr1 to <fs1> # Call the method CREATE_DYNAMIC_TABLE

a).Pass the internal table containing the field attributes as the export parameter.

b).Pass the field symbol of the dynamic internal table as the import parameter. # This creates the dynamic internal table

<fs1> now refers to dynamic internal table that we wanted to create at start. # Assign the data contents of <fs1> to a field-symbol <fs2> (dereferencing).

So <fs2> now points to the dynamic internal table. # Next step is to create a work area for our dynamic internal table.

Create data dr2 like line of <fs2>. # A field-symbol to access that work area

Assign dr2->* to <FS_2>. Drawbacks of a Dynamic Internal Table

- Programs with many dynamic internal tables are less readable and they are less secure, since error cannot be detected by syntax check, but only by the runtime system.
- Performance when a Dynamic Internal Table is used will not be as good as, when a Static internal table is used.

Sample Program example: how to create a dynamic internal table

```
TYPE-POOLS : abap.
FIELD-SYMBOLS: <dyn_table> TYPE STANDARD TABLE,
             <dyn_wa>,
             <dyn_field>.
DATA: dy_table TYPE REF TO data,
    dy_line  TYPE REF TO data,
    xfc TYPE lvc_s_fcat,
    ifc TYPE lvc_t_fcat.
SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME.
PARAMETERS: p_table(30) TYPE c DEFAULT 'T001'.
SELECTION-SCREEN END OF BLOCK b1.

START-OF-SELECTION.
  PERFORM get_structure.
  PERFORM create_dynamic_itab.
```

```
*******Creates a dynamic internal table*********
  PERFORM get_data.
  PERFORM write_out.
*&---------------------------------------------------------------------*
*&      Form  get_structure
*&---------------------------------------------------------------------*
FORM get_structure.
  DATA : idetails TYPE abap_compdescr_tab,
       xdetails TYPE abap_compdescr.
  DATA : ref_table_des TYPE REF TO cl_abap_structdescr.
* Get the structure of the table.
  ref_table_des ?=
      cl_abap_typedescr=>describe_by_name( p_table ).
  idetails[] = ref_table_des->components[].
  LOOP AT idetails INTO xdetails.
    CLEAR xfc.
    xfc-fieldname = xdetails-name .
* Correction by Paul Robert Oct 28, 2009 17:04
*    xfc-datatype = xdetails-type_kind.
    CASE xdetails-type_kind.
      WHEN 'C'.
        xfc-datatype = 'CHAR'.
      WHEN 'N'.
        xfc-datatype = 'NUMC'.
      WHEN 'D'.
        xfc-datatype = 'DATE'.
      WHEN 'P'.
        xfc-datatype = 'PACK'.
      WHEN OTHERS.
        xfc-datatype = xdetails-type_kind.
    ENDCASE.
    xfc-inttype = xdetails-type_kind.
    xfc-intlen = xdetails-length.
    xfc-decimals = xdetails-decimals.
    APPEND xfc TO ifc.
  ENDLOOP.
```

```
ENDFORM.                    "get_structure
*&---------------------------------------------------------------------*
*&      Form  create_dynamic_itab
*&---------------------------------------------------------------------*
FORM create_dynamic_itab.
* Create dynamic internal table and assign to FS
  CALL METHOD cl_alv_table_create=>create_dynamic_table
    EXPORTING
      it_fieldcatalog  = ifc
      i_length_in_byte = 'X' "added by Paul Robert Oct 28, 2009 17:04
    IMPORTING
      ep_table         = dy_table.
  ASSIGN dy_table->* TO <dyn_table>.
* Create dynamic work area and assign to FS
  CREATE DATA dy_line LIKE LINE OF <dyn_table>.
  ASSIGN dy_line->* TO <dyn_wa>.
ENDFORM.                    "create_dynamic_itab

*&---------------------------------------------------------------------*
*&      Form  get_data
*&---------------------------------------------------------------------*
FORM get_data.
* Select Data from table.
  SELECT * INTO TABLE <dyn_table>
             FROM (p_table).
ENDFORM.                    "get_data

*&---------------------------------------------------------------------*
*&      Form  write_out
*&---------------------------------------------------------------------*
FORM write_out.
  LOOP AT <dyn_table> INTO <dyn_wa>.
    DO.
      ASSIGN COMPONENT  sy-index
         OF STRUCTURE <dyn_wa> TO <dyn_field>.
      IF sy-subrc <> 0.
        EXIT.
```

```
      ENDIF.
      IF sy-index = 1.
        WRITE:/ <dyn_field>.
      ELSE.
        WRITE: <dyn_field>.
      ENDIF.
    ENDDO.
  ENDLOOP.
ENDFORM.                    "write_out
```

No labels

6 Comments

Unknown User (ivix04a)

Hi,

I have a problem with the method

```
  CALL METHOD cl_alv_table_create=>create_dynamic_table
    EXPORTING
*     i_style_table            = space
      it_fieldcatalog          = p_t_fcat
    IMPORTING
      ep_table                 = t_tabla_dyn
    EXCEPTIONS
      generate_subpool_dir_full = 1
      OTHERS                    = 2.
```

My routine is called several times into the program, and in some moment the exception is raising

GENERATE_SUBPOOL_DIR_FULL At Most 36 Subroutine Pools Can Be Generated Temporarily

Can yuo help me ? i dont need to save the subroutine pool generated and i can't find the way to free them

Thanks!

Former Member

Two Issues with the code as it applies to a Unicode environment.

The table created will have character fields doubled in length from the original fields

To correct the code, the "datatype" field has to expand on the value to match what is expected later.

```
case xdetails-type_kind.
  when 'C'.
xfc-datatype = 'CHAR'.
  when 'N'.
xfc-datatype = 'NUMC'.
  when 'D'.
xfc-datatype = 'DATE'.
  when 'P'.
xfc-datatype = 'PACK'.
WHEN OTHERS.
xfc-datatype = xdetails-type_kind.
endcase.
```

More data types may need to change, the above meets my current requirement.

The call method for creating the dynamic_table also needs one more parameter.

the i_length_in_byte parameter needs to be set to 'X'.

```
CALL METHOD cl_alv_table_create=>create_dynamic_table
  EXPORTING
    it_fieldcatalog = gta_ifc
    i_length_in_byte = 'X'
  IMPORTING
    ep_table        = gta_dy_table.
```

Sandra Rossi

Paul, I have reproduced the error, and corrected it as per your suggestion.

Thanks a lot.

sandra

Eduardo Puricelli

Hi I want to export table to excel, I use WS_DOWNLOAD and works fine but I need the field corresponding header of all fields to know what fields is it.

thanks!

Anonymous

Hi,

I had modified above code according to my requiremens and had length problem. I will explain my case to prevent other users do the same mistake.

I had to create dynamic internal table but I didn't know the structure before runtime. Then, I filled table ifc manually instead of using form "get_structure". Then I called "cl_alv_table_create=>create_dynamic_table" to create internal table dynamically. This time all fields are created with half of the original size. I removed "i_length_in_bytes" parameters since I gave the exact lengths when I fill ifc manually. If you fill ifc with method "cl_abap_typedescr=>describe_by_name" it fills lengths in bytes, but if you fill it manually then you probably won't use lengths in bytes.

Abdul.

Unknown User (tnc2pbd)

Hi,
I am still a beginner in ABAP and especially to dynamic programming, but I think we can create the dynamic table in much easier way, does the approach below have any disadvantage compared to the code in the example?

```
FIELD-SYMBOLS: <dyn_table> TYPE STANDARD TABLE,
<dyn_wa>,
<dyn_field>.
DATA: dy_table TYPE REF TO data,
dy_line TYPE REF TO data.

SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME.
PARAMETERS: p_table(30) TYPE c DEFAULT 'T001'.
SELECTION-SCREEN END OF BLOCK b1.

START-OF-SELECTION.

CREATE DATA dy_table TYPE TABLE OF (p_table).

ASSIGN dy_table->* TO <dyn_table>.

CREATE DATA dy_line LIKE LINE OF <dyn_table>.

ASSIGN dy_line->* TO <dyn_wa>.

SELECT * INTO TABLE <dyn_table> FROM (p_table).

LOOP AT <dyn_table> ASSIGNING <dyn_wa>.
DO.
ASSIGN COMPONENT sy-index
OF STRUCTURE <dyn_wa> TO <dyn_field>.
IF sy-subrc <> 0.
EXIT.
ENDIF.
IF sy-index = 1.
WRITE:/ <dyn_field>.
ELSE.
WRITE: <dyn_field>.
ENDIF.
ENDDO.
ENDLOOP.
```