ABAP Part I

Lesson 08: Internal Tables - II

Lesson Objectives

After completing this lesson, participants will be able to know:

- To Determine the number of Rows of Internal table
- To use collect statement
- To use Append Sorted by in Internal Table
- Control break statements on Internal Table

Determining the Number of Rows in an Internal Table

To determine the number of rows in an internal table, use sy-tfill variable.

It is set by the describe table statement.

Syntax

The following is the syntax for the *describe table* statement.

- **describe table it [lines i]   [occurs j].**

where:

- it is the name of an internal table
- i and j are numeric variables

Determining the Number of Rows in an Internal Table

The describe statement fills the three system variables shown in table below

Variable	Value
Sy-tfill	Number of rows
Sy-tleng	Length of a row in bytes
Sy-toccu	Current value of the occurs clause

The following points apply:

- If the lines i addition is specified, the number of rows is placed in both sy-tfill and I
- If the occurs j addition is specified, the size of the occurs clause is placed in both sy-toccu and j

Creating Top 10 Lists Using the append sorted by

Imagine that you are asked to create a report of the top 10 sales representatives in your company.

Assuming you have a way of obtaining the total sales for each rep, you could do one of the following:

- Append all reps and their total sales into an internal table
- Sort them descending by sales
- Write out the first 10

This seems like a logical way to proceed.

However, using the append sorted by statement often can produce the same result and be twice as efficient

Creating Top 10 Lists Using the append sorted by

Syntax for the append sorted by Statement:

- **append [wa to] it sorted by c.**

where:

- it is the name of an internal table
- wa is a work area having the same structure as row of the internal table
- c is a component of it

Creating Top 10 Lists Using the append sorted by

The append sorted by statement takes a row from the work area and inserts it into the internal table at the point where it belongs in the sort order.

It has two unusual properties:

- The number of rows that can be appended is limited by the value on the occurs clause.
- For example, if the occurs is 10, a maximum of 10 rows can be appended to the internal table.
- This is the only situation where occurs limits the number of rows that can be added to an internal table
- It only sorts in descending order

The net effect is a "top n list," where n is the number on the occurs clause.

Creating Top 10 Lists Using the append sorted by

With each append, the system searches the existing table contents to determine where the new record fits.

The sort order is by c descending.

If there are fewer rows in the internal table than specified by n on the occurs clause, the row is inserted as per the sort order.

If there are n rows in the internal table, the row is inserted as per the sort order and the last row is discarded.

If the value in c already exists in the internal table, the new row is always appended after existing rows that have the same value.

Therefore, if occurs is 3 and row 3 contains ‘X’ in c, a new row having ‘X’ in c will not be appended

Demo

Program Append Sorted By

Filling an Internal Table Using collect

Using the collect statement, you can create totals with an internal table as you fill it.

Syntax for the collect Statement

The following is the syntax for the *collect* statement.

- **collect [wa into] it.**

- where:
  - it is an internal table
  - wa is a work area having the same structure as it

Demo

Program Collect

Control Break Processing

After you fill an internal table with data, you often need to write the data out.

This output will frequently contain summary information (such as totals) at the top or bottom of the report.

To do this, you can read the data into the internal table and then, within loop at, use the following statements:

- at first / endat
- at last / endat
- at new / endat
- at end of / endat
- sum
- on change of / endon

Using the at first and at last Statements

Use the at first and at last statements to perform processing during the first or last loop pass of an internal table.

Syntax for the at first and at last Statements

The following is the syntax for the *at first* and *at last* statements.

```
loop at it.
...
  at first.
    ...
    endat.
...
  at last.
    ...
    endat.
endloop
```

- where:
  - it is an internal table
  - … represents any number of lines of code (even zero)

Using the at first and at last Statements

Use at first for:

- Loop initialization processing
- Writing totals at the top of a report
- Write headings

Use at last for:

- Loop termination processing
- Writing totals at the bottom of a report
- Write footings

Demo

Program At First At Last

Using the at new and at end of Statements

Use the at new and at end of statements to detect a change in a column from one loop pass to the next.

Syntax for the at new and at end of Statements

The following is the syntax for the at new and at end of statements.

```
sort by c.
loop at it.
...
        at new c.
          ...
          endat.
...
        at end of c.
          ...
          endat.
endloop
```

- where:
  - it is an internal table
  - c is a component of it
  - … represents any number of lines of code (even zero)

Using at New

Each time the value of c changes, the lines of code between at new and endat are executed.

This block is also executed during the first loop pass or if any fields to the left of c change.

Between at and endat, the numeric fields to the right of c are set to zero.

The non-numeric fields are filled with asterisks (*).

If there are multiple occurrences of at new, they are all executed.

At end of behaves in a similar fashion

Using at New

A control level is the component named on a control break statement; it regulates the control break.

For example, in the following code snippet, f2 is a control level because it appears on the at new statement.

```
loop at it.
at new f2.
            " (some code here)
            endat.
endloop.
```

Using at end of

The lines of code between at end of and endat are executed:

- If the control level changes
- If any field prior to the control level changes
- If this is the last row of the table

Demo

Program At New At End

Using the sum Statement

Use the sum statement to calculate totals for the rows of a control level.

Syntax for the sum Statement

The following is the syntax for the *sum* statement.

```
at first / last / new / end of.
   …
   sum.
   …
   endat.
```

- where:
  - … represents any number of lines of code

Demo

Program - Sum

Using the on change of Statement

Another statement you can use to perform control break processing is on change of. It behaves in a manner similar to at new.

Syntax for the on change of Statement

The following is the syntax for the *on change of* statement.

```
on change of v1  [ or v2 ... ].
        ---
[else.
         ---]
        endon.
```

- where:
  - v1 and v2 are variable or field string names
  - ... indicates that any number of or conditions might follow
  - --- represents any number of lines of code

Demo

Program – On Change of

Obsolete Statements

Using HEADER LINE as Work Area.

Example:

```
  TYPES: BEGIN OF line,
      num TYPE i,
      sqr TYPE i,
END OF line.
DATA it_tab TYPE  TABLE OF line WITH UNIQUE KEY col1 WITH HEADER LINE.
DO 5 TIMES.
      It_tab-num = sy-index.
      It_tab-sqr = sy-index ** 2.
      INSERT TABLE it_tab.
   ENDDO.
It_tab-sqr = 100.
   MODIFY TABLE it_tab.
It_tab-num = 4.
   DELETE TABLE it_tab.
```

Summary

In this lesson, you have learnt:

- To determine the number of Rows in an Internal Table
- To Sort the Contents of an Internal Table
- Control break statements on Internal Table

Review Question

Question 1: ______variable is used to determine the number of rows in an internal table.

Question 2: _______ statement allows to create totals with an internal table as you fill it.