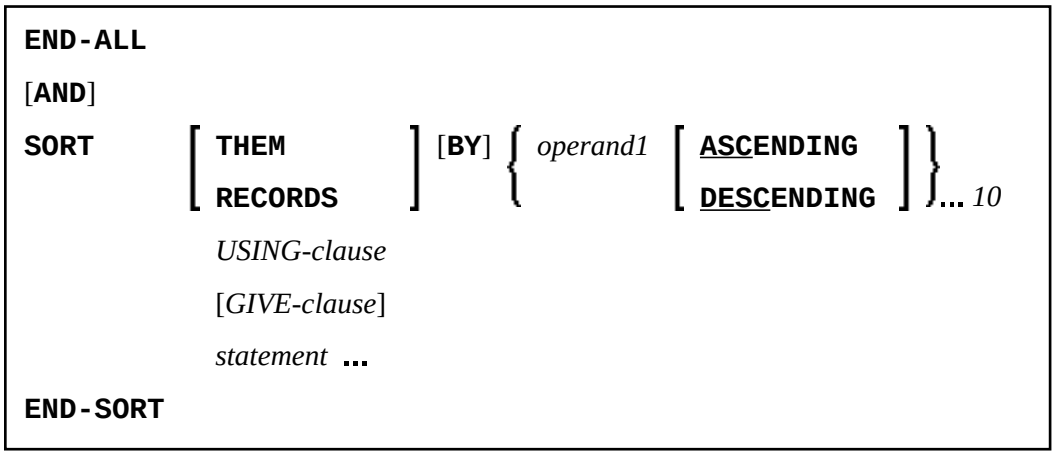
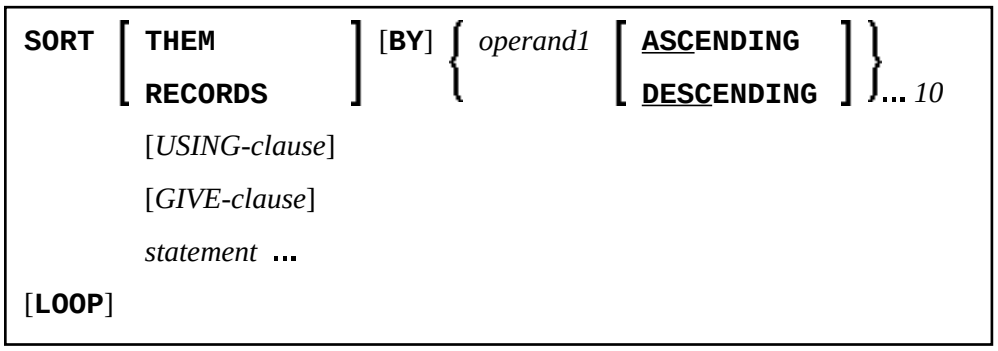
SORT

Structured Mode Syntax



* If a statement label is specified, it must be placed *before* the keyword SORT, but *after* END-ALL (and AND).

Reporting Mode Syntax



This chapter covers the following topics:

- Function
- Restrictions
- Syntax Description
- Three-Phase SORT Processing
- Example

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Related Statement: FIND with SORTED BY option

Belongs to Function Group: *Loop Execution*

Function

The SORT statement is used to perform a sort operation, sorting the records from all processing loops that are active when the SORT statement is executed.

Restrictions

- The SORT statement must be contained in the same object as the processing loops whose records it sorts.
- Nested SORT statements are not allowed.
- The total length of a record to be sorted must not exceed 10240 bytes.
- The number of sort criteria must not exceed 10.

Syntax Description

Operand Definition Table:

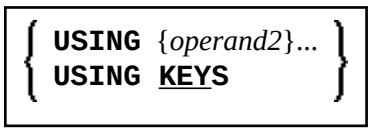
Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
operand1		S				A	N	P	I	F	B	D	T					no	no

Syntax Element Description:

Syntax Element	Description
END-ALL	**Closing All Currently Active Loops:** In structured mode, the SORT statement must be preceded by END-ALL, which serves to close all active processing loops. The SORT statement itself initiates a new processing loop, which must be closed with END-SORT. **Note:** For reporting mode: The SORT statement closes all active processing loops and initiates a new processing loop.
operand1	**Sort Criteria:** *operand1* represents the fields/variables to be used as the sort criteria. 1 to 10 database fields (descriptors and non-descriptors) and/or user-defined variables may be specified. A multiple-value field or a field contained within a periodic group may be used. A group or an array is not permitted.
ASCENDING DESCENDING	**Sort Sequence:** The default sort sequence is ascending. If you wish the values to be sorted in descending sequence, specify DESCENDING. ASCENDING/DESCENDING may be specified for each sort field.
USING	**USING Clause:** See *USING Clause* below.
GIVE	**GIVE Clause:** See *GIVE Clause* below.
END-SORT	**End of SORT Statement:** The Natural reserved word END-SORT must be used to end the SORT statement.

USING Clause

The USING clause indicates the fields which are to be written to intermediate sort storage. It is required in structured mode and optional in reporting mode. However, it is strongly recommended to also use it in reporting mode so as to reduce memory requirements.



Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand2		S	A			A	N	P	I	F	B	D	T	L	C			no	no

Syntax Element Description:

Syntax Element	Description
`USING` *operand2*	**Additional Fields:** You can specify additional fields that are to be written to the intermediate sort storage - in addition to the sort key fields (as specified with *operand1*).
`USING KEYS`	**Sort Key Fields Only:** Only the sort key fields, as specified with *operand1*, will be written to intermediate sort storage.

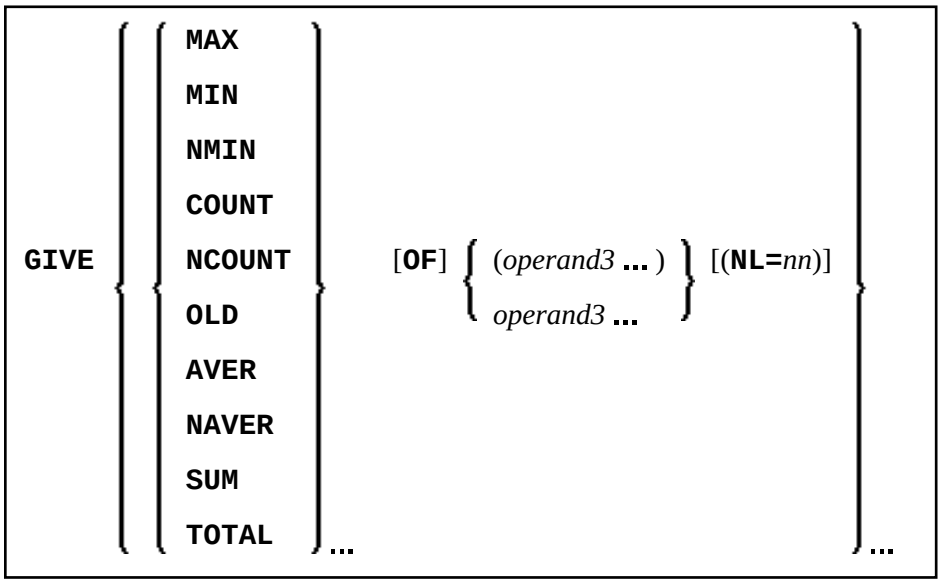
In Reporting Mode: If you omit the `USING` clause, all database fields of processing loops initiated before the `SORT` statement, as well as all user-defined variables defined before the `SORT` statement, will be written to intermediate sort storage.

If, after sort execution, a reference is made to a field which was not written to the sort intermediate storage, the value for the field will be the last value of the field before the sort.

GIVE Clause

The `GIVE` clause is used to specify Natural system functions (such as `MAX`, `MIN`) that are to be evaluated in the first phase of the `SORT` statement. These system functions may be referenced in the third phase (see *SORT Statement Processing*).

A reference to a system function after the `SORT` statement must be preceded by an asterisk, for example, `*AVER(SALARY)`.



Operand Definition Table:

Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
operand3		S	A			*											yes	no

* depends on function

Syntax Element Description:

Syntax Element	Description
MAX | MIN | NMIN | COUNT | NCOUNT | OLD | AVER | NAVER | SUM | TOTAL	**System Functions:** For details on the individual system functions, see the *System Functions* documentation.
operand3	**Field Name:** *operand3* is the field name.
(NL=*nn*)	**Preventing Arithmetic Overflows:** This option applies to the system functions AVER, NAVER, SUM and TOTAL only. It will be ignored for any other system function. This option may be used to prevent an arithmetic overflow during the evaluation of system functions; it is described under *Arithmetic Overflows in AVER, NAVER, SUM or TOTAL* in the *System Functions* documentation.

Three-Phase SORT Processing

A program containing a SORT statement is executed in three phases.

1st Phase - Selecting the Records to be Sorted

The statements before the SORT statement are executed. Data as described in the USING clause will be written to intermediate sort storage.

In reporting mode, any variables to be used as accumulators following the sort must not be defined before the SORT statement. In structured mode, they must not be included in the USING clause. Fields written to intermediate sort storage cannot be used as accumulators because they are read back with each individual record during the 3rd processing phase. Consequently, the accumulation function would not produce the desired result because with each record the field would be overwritten with the value for that individual record.

The number of records written to intermediate storage is determined by the number of processing loops and the number of records processed per loop. One record on the internal intermediate storage is created each time the SORT statement is encountered in a processing loop. In the case of nested loops, a record is only written to intermediate storage if the inner loop is executed. If in the example below a record is to be written to intermediate storage even if no records are found for the inner (FIND) loop, the FIND

statement must contain an IF NO RECORDS FOUND clause.

```
READ ...
  ...
  FIND ...
...
END-ALL
SORT ...
  DISPLAY ...
END-SORT
...
```

2nd Phase - Sorting the Records

The records are sorted.

3rd Phase - Processing the Sorted Records

The statements after the SORT statement are executed for all records on the intermediate storage in the specified sorting sequence. Database fields to be referenced after a SORT statement must be correctly referenced using the appropriate statement label or reference number.

Example

- Example 1 - SORT
- Example 2 - SORT

Example 1 - SORT

```
** Example 'SRTEX1S': SORT (structured mode)
************************************************************************
DEFINE DATA LOCAL
1 EMPL-VIEW VIEW OF EMPLOYEES
  2 CITY
  2 SALARY      (1:2)
  2 PERSONNEL-ID
  2 CURR-CODE   (1:2)
*
1 #AVG          (P11)
1 #TOTAL-TOTAL  (P11)
1 #TOTAL-SALARY (P11)
1 #AVER-PERCENT (N3.2)
END-DEFINE
*
LIMIT 3
FIND EMPL-VIEW WITH CITY = 'BOSTON'
  COMPUTE #TOTAL-SALARY = SALARY (1) + SALARY (2)
  ACCEPT IF #TOTAL-SALARY GT 0
  /*
END-ALL
AND
SORT BY PERSONNEL-ID USING #TOTAL-SALARY SALARY(*) CURR-CODE(1)
     GIVE AVER(#TOTAL-SALARY)
  /*
  AT START OF DATA
    WRITE NOTITLE '*' (40)
         'AVG CUMULATIVE SALARY:' *AVER (#TOTAL-SALARY) /
```

```
    MOVE *AVER (#TOTAL-SALARY) TO #AVG
  END-START
  COMPUTE ROUNDED #AVER-PERCENT = #TOTAL-SALARY / #AVG * 100
  ADD #TOTAL-SALARY TO #TOTAL-TOTAL
  /*
  DISPLAY NOTITLE PERSONNEL-ID SALARY (1) SALARY (2)
          #TOTAL-SALARY CURR-CODE (1)
          'PERCENT/OF/AVER' #AVER-PERCENT
  AT END OF DATA
    WRITE / '*' (40) 'TOTAL SALARIES PAID: ' #TOTAL-TOTAL
  END-ENDDATA
END-SORT
*
END
```

Output of Program SRTEX1S:

```
PERSONNEL   ANNUAL     ANNUAL   #TOTAL-SALARY CURRENCY PERCENT
     ID       SALARY     SALARY                    CODE     OF
                                                          AVER
  --------- ---------- ---------- ------------- -------- -------

  ************************************ AVG CUMULATIVE SALARY:        41900

  20007000       16000      15200        31200  USD        74.00
  20019200       18000      17100        35100  USD        83.00
  20020000       30500      28900        59400  USD       141.00

  ************************************ TOTAL SALARIES PAID:        125700
```

The previous example is executed as follows:

First Phase:

- Records with `CITY=BOSTON` are selected from the `EMPLOYEES` file.
- The first 2 occurrences of `SALARY` are accumulated in the field `#TOTAL-SALARY`.
- Only records with `#TOTAL-SALARY` greater than 0 are accepted.
- The records are written to the sort intermediate storage. The database arrays `SALARY` (first 2 occurrences) and `CURR-CODE` (first occurrence), the database field `PERSONNEL-ID`, and the user-defined variable `#TOTAL-SALARY` are written to the intermediate storage.
- The average of `#TOTAL-SALARY` is evaluated.

Second Phase:

- The records are sorted.

Third Phase:

- The sorted intermediate storage is read.
- At the at-start-of-data condition, the average of `#TOTAL-SALARY` is displayed.

- #TOTAL-SALARY is added to #TOTAL-TOTAL and the fields PERSONNEL-ID, SALARY(1), SALARY(2), #AVER-PERCENT and #TOTAL-SALARY are displayed.
- At the end-of-data condition, the variable #TOTAL-TOTAL is written.

Equivalent reporting-mode example: SRTEX1R.

Example 2 - SORT

```
** Example 'SRTEX2': SORT
************************************************************************
DEFINE DATA LOCAL
1 VEHIC-VIEW VIEW OF VEHICLES
  2 MAKE
  2 YEAR
END-DEFINE
*
LIMIT 10
*
READ VEHIC-VIEW
END-ALL
SORT BY MAKE YEAR USING KEY
  DISPLAY NOTITLE (AL=15) MAKE (IS=ON) YEAR
  AT BREAK OF MAKE
    WRITE '-' (20)
  END-BREAK
END-SORT
END
```

Output of Program SRTEX2S:

```
     MAKE       YEAR
--------------- -----

FIAT             1980
                 1982
                 1984
--------------------
PEUGEOT          1980
                 1982
                 1985
--------------------
RENAULT          1980
                 1980
                 1982
                 1982
--------------------
```