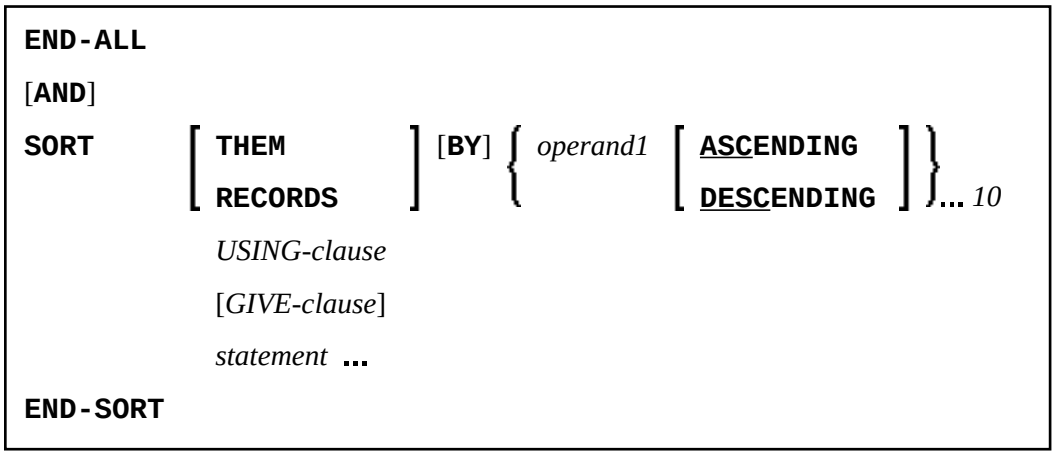
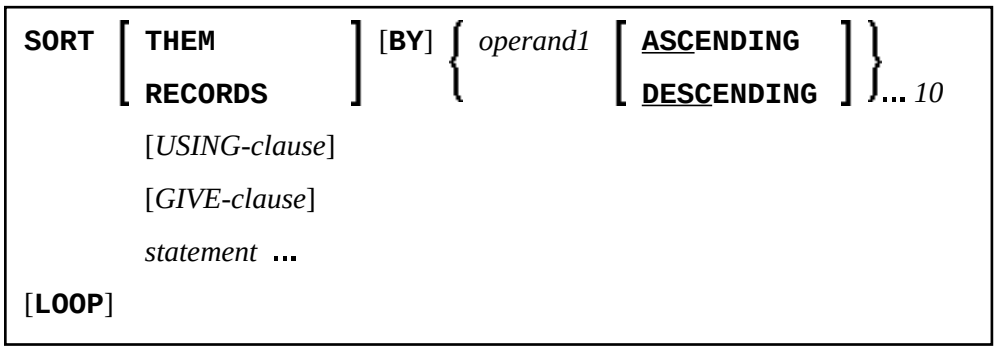
SORT

Structured Mode-Syntax



* Wenn ein Statement-Label angegeben wird, muss es vor dem Schlüsselwort SORT, aber *nach* END–ALL (und AND) stehen.

Reporting Mode-Syntax



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Einschränkungen
- Syntax-Beschreibung
- Phasen der SORT-Verarbeitung
- Beispiel

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Verwandtes Statement: FIND mit SORTED BY-Option

Gehört zur Funktionsgruppe: *Schleifenverarbeitung*

Funktion

Das Statement SORT dient dazu, eine Sortieroperation durchzuführen und die Datensätze aus allen Verarbeitungsschleifen, die zum Zeitpunkt der SORT-Ausführung aktiv sind, zu sortieren.

Einschränkungen

- Das SORT-Statement muss im selben Objekt stehen wie die Verarbeitungsschleifen, deren Datensätze es sortiert.
- Geschachtelte SORT-Statements sind nicht erlaubt.
- Die Gesamtlänge eines zu sortierenden Datensatzes darf 10240 Bytes nicht überschreiten.
- Die Anzahl der Sortierkriterien darf 10 nicht überschreiten.

Syntax-Beschreibung

Operanden-Definitionstabelle:

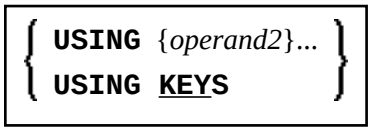
Operand	Mögliche Struktur					Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1		S				A	N	P	I	F	B	D	T					nein	nein

Syntax-Element-Beschreibung:

END-ALL	Im Structured Mode müssen Sie vor dem SORT-Statement das Statement END–ALL angeben; damit werden alle noch aktiven Verarbeitungsschleifen beendet. Das SORT-Statement initiiert seinerseits eine neue Verarbeitungsschleife, welche mit END–SORT geschlossen werden muss. **Anmerkung:** Im Reporting Mode beendet das SORT-Statement alle noch aktiven Verarbeitungsschleifen und initiiert eine neue Verarbeitungsschleife.
operand1	**Sortierkriterien:** *operand1* sind die Felder/Variablen, die als Sortierkriterium dienen. 1 bis 10 Felder dürfen angegeben werden. Hierbei kann es sich um Datenbankfelder (Deskriptoren oder Nicht-Deskriptoren) und/oder Benutzervariablen handeln. Multiple Felder oder Felder aus einer Periodengruppe können ebenfalls verwendet werden; eine Feldgruppe oder ein Array kann nicht verwendet werden.
ASCENDING	**Sortierreihenfolge:** Wird nichts anderes angegeben, so gilt ASCENDING, d.h. die Werte werden in aufsteigender Reihenfolge sortiert. Möchten Sie die Werte in absteigender Reihenfolge sortiert haben, geben Sie das Schlüsselwort DESCENDING an. Das Schlüsselwort ASCENDING bzw. DESCENDING kann für jedes Sortierfeld getrennt angegeben werden.
DESCENDING	
USING	**USING-Klausel:** Siehe *USING-Klausel* weiter unten.
GIVE	**GIVE-Klausel:** Siehe *GIVE-Klausel* weiter unten.
END-SORT	Das für Natural reservierte Wort END–SORT muss zum Beenden des SORT-Startements benutzt werden.

USING-Klausel

In der USING-Klausel geben Sie die Felder an, die in den Sortier-Zwischenspeicher geschrieben werden sollen. Die USING-Klausel ist im Structured Mode unbedingt erforderlich, im Reporting Mode nicht; allerdings wird dringend empfohlen, sie auch im Reporting Mode zu verwenden, um Speicherplatz zu sparen.



Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand2		S	A			A	N	P	I	F	B	D	T	L	C			nein	nein

Syntax-Element-Beschreibung:

USING *operand2*	Mit *`USING operand2`* können Sie weitere Felder angeben, die — zusätzlich zu den (als *operand1* angegebenen) Sortierfeldern — in den Sortier-Zwischenspeicher geschrieben werden sollen.
USING KEYS	Wenn Sie `USING KEYS` angeben, werden nur die als *operand1* angegebenen Sortierfelder in den Sortier-Zwischenspeicher geschrieben.

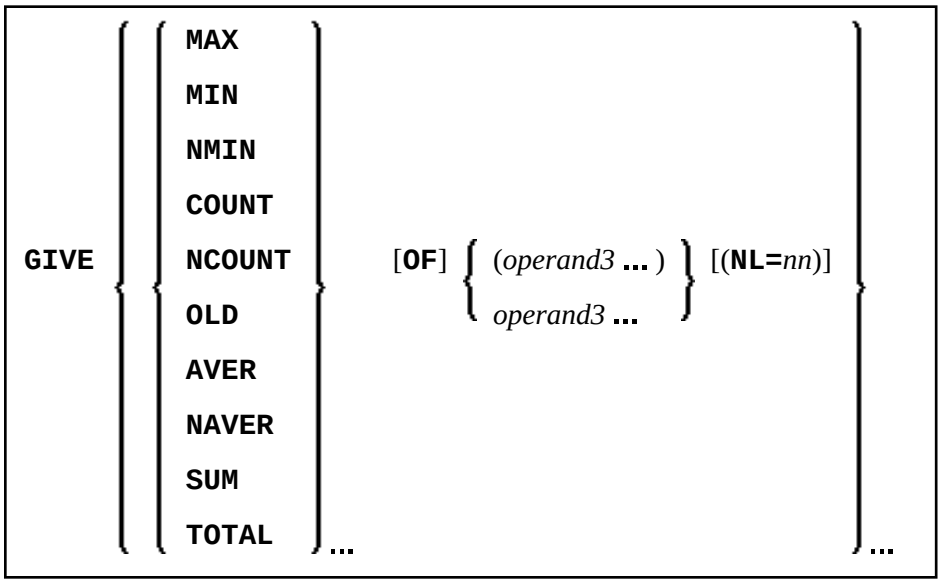
Im Reporting Mode:

Verwenden Sie keine `USING`-Klausel, so werden alle Datenbankfelder aus vor dem `SORT`-Statement initiierten Verarbeitungsschleifen sowie alle vor dem `SORT`-Statement definierten Benutzervariablen in den Sortier-Zwischenspeicher geschrieben.

Wird nach Ausführung des `SORT`-Statements ein Feld referenziert, das nicht in den Sortier-Zwischenspeicher geschrieben wurde, so ist der Wert des Feldes der, den es vor der SORT-Operation hatte.

GIVE-Klausel

Die `GIVE`-Klausel dient dazu, Natural-Systemfunktionen (`MAX`, `MIN` usw.) anzugeben, die in der ersten Phase des Sortiervorgangs ausgewertet werden und dann in der dritten Phase referenziert werden können (siehe Abschnitt *Phasen der SORT-Verarbeitung*). Wird nach dem `SORT`-Statement eine Systemfunktion referenziert, muss ihrem Namen ein Stern vorangestellt werden; Beispiel: `*AVER(SALARY)`.



Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate										Referenzierung erlaubt	Dynam. Definition
operand3		S	A			*										ja	nein

* je nach Funktion

Syntax-Element-Beschreibung:

`MAX` | `MIN` | `NMIN` | `COUNT` | `NCOUNT` | `OLD` | `AVER` | `NAVER` | `SUM` | `TOTAL`	Näheres zu den einzelnen Systemfunktionen finden Sie in der *Systemfunktionen*-Dokumentation.
operand3	*operand3* ist der Feldname.
`(NL=`*nn*`)`	Diese Option gilt nur für `AVER`, `NAVER`, `SUM` und `TOTAL`; für alle anderen Systemfunktionen wird sie ignoriert. Diese Option kann dazu verwendet werden, einen arithmetischen Überlauf bei der Auswertung von Systemfunktionen zu vermeiden; sie ist unter *Arithmetischer Überlauf bei AVER, NAVER, SUM oder TOTAL* in der *Systemfunktionen*-Dokumentation beschrieben.

Phasen der SORT-Verarbeitung

Ein Programm, das ein `SORT`-Statement enthält, wird in drei Phasen ausgeführt:

1. Phase — Auswählen der zu sortierenden Datensätze

Die Statements vor dem `SORT`-Statement werden ausgeführt. Die in der `USING`-Klausel angegebenen Daten werden in den Sortier-Zwischenspeicher geschrieben.

Im Reporting Mode dürfen Variablen, die nach dem Sortieren als Akkumulatoren verwendet werden, nicht vor dem SORT-Statement definiert werden.

Im Structured Mode dürfen sie nicht in der `USING`-Klausel angegeben werden.

In den Sortier-Zwischenspeicher geschriebene Felder können als Akkumulatoren nicht verwendet werden, weil sie in der dritten Phase mit jedem einzelnen Datensatz zurückgeschrieben werden. Folglich hätte die Akkumulationsfunktion nicht das gewünschte Ergebnis, da das Feld bei jedem Datensatz mit dem Wert des jeweiligen Datensatzes überschrieben würde.

Die Anzahl der in den Sortier-Zwischenspeicher geschriebenen Datensätze ergibt sich aus der Anzahl der Verarbeitungsschleifen und der Anzahl der verarbeiteten Datensätze pro Schleife. Jedesmal wenn das `SORT`-Statement in einer Verarbeitungsschleife ausgeführt wird, wird im internen Sortier-Zwischenspeicher ein Datensatz angelegt.

Bei geschachtelten Schleifen wird ein Datensatz nur in den Sortier-Zwischenspeicher geschrieben, wenn die innere Schleife ausgeführt wird. Sollen im folgenden Beispiel Datensätze in den Sortier-Zwischenspeicher geschrieben werden, auch wenn in der inneren (`FIND`-)Schleife keine gefunden werden, so muss das `FIND`-Statement eine `IF NO RECORDS FOUND`-Klausel enthalten.

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

2. Phase — Sortieren der Datensätze

Die Datensätze werden sortiert.

3. Phase — Weiterverarbeitung der sortierten Datensätze

Die Datensätze aus dem Sortier-Zwischenspeicher werden in der angegebenen Sortierfolge mit den auf das SORT-Statement folgenden Statements weiterverarbeitet. Werden Datenbankfelder nach dem SORT-Statement referenziert, so muss dies über ein Statement-Label oder durch Angabe der entsprechenden Sourcecode- Zeilennummer erfolgen.

Beispiel

- Beispiel 1 — SORT
- Beispiel 2 — SORT

Beispiel 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
```

Ausgabe des Programms 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
```

Das obige Beispiel wird wie folgt verarbeitet:

Phase 1:

- Von der EMPLOYEES-Datei werden Datensätze mit CITY = BOSTON gelesen.
- Die ersten beiden Ausprägungen des SALARY-Feldes werden in der Variablen #TOTAL−SALARY addiert.
- Es werden nur Datensätze weiterverarbeitet, bei denen der Wert von #TOTAL−SALARY größer als 0 ist.
- Die Sätze werden in den Sortier-Zwischenspeicher geschrieben. Die Datenbank-Arrays SALARY (die ersten beiden Ausprägungen) und CURR−CODE (erste Ausprägung), das Datenbankfeld PERSONNEL−ID und die Benutzervariable #TOTAL−SALARY werden in den Zwischenspeicher geschrieben.
- Der Durchschnittswert von #TOTAL−SALARY wird errechnet.

Phase 2:

- Die Datensätze werden sortiert.

Phase 3:

- Der sortierte Zwischenspeicherinhalt wird gelesen.
- Bei der Ausführung des AT START OF DATA-Blocks wird der Durchschnittswert von #TOTAL–SALARY angezeigt.
- Der Wert von #TOTAL–SALARY wird zu #TOTAL–TOTAL hinzuaddiert; es werden die Felder PERSONNEL–ID, SALARY (1), SALARY (2), #AVER–PERCENT und #TOTAL–SALARY angezeigt.
- Bei der Ausführung des AT END OF DATA-Blocks wird die Variable #TOTAL–TOTAL ausgegeben.

Äquivalentes Reporting-Mode-Beispiel: SRTEX1R.

Beispiel 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
```

Ausgabe des Programms SRTEX2S:

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

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