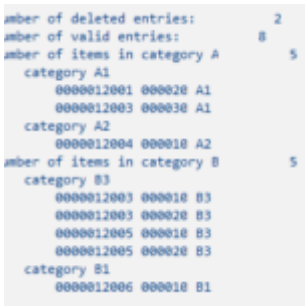

Loop At ITAB Group By

Mühsam ernährt sich das Eichhörnchen. Die heutige Nuss galt dem Befehlszusatz GROUP BY für den LOOP über eine interne Tabelle. Es gibt dankenswerter Weise in der SAP-Doku inzwischen viele Beispiele. Diese sind jedoch sehr abstrakt. Sie zeigen die Syntax, verdeutlichen aber nicht unbedingt, was damit möglich ist.

Ich präsentiere euch heute ein paar Möglichkeiten der Gruppierung, die hoffentlich die Funktionsweise deutlich machen.

Beispieldaten

im Folgenden verwende ich diese Struktur für meine Beispiele:

```
TYPES: BEGIN OF _doc,
         docnr TYPE n length 10,
         itmno TYPE n length 6,
         category type c length 2,
         del_flag type abap_bool,
       END OF _doc,
       _docs TYPE SORTED TABLE OF _doc WITH UNIQUE KEY docnr itmno.

DATA(documents) = VALUE _docs(
  ( docnr = 12001 itmno = 10 category = 'A1' del_flag = 'X' )
  ( docnr = 12001 itmno = 20 category = 'A1' del_flag = ' ' )
  ( docnr = 12002 itmno = 10 category = 'A2' del_flag = 'X' )
  ( docnr = 12003 itmno = 10 category = 'B3' del_flag = ' ' )
  ( docnr = 12003 itmno = 20 category = 'B1' del_flag = ' ' )  ).
```

Die Tabelle soll generelle Belege mit Positionsnummer, einem Positionstypen und einem Löschkennzeichen simulieren.

Generelle Funktionsweise

Der Befehlszusatz GROUP BY zum LOOP bietet die Möglichkeit der Gruppierung, ähnlich wie die Sprachelemente AT NEW oder AT CHANGE OF innerhalb eines LOOP. Allerdings bietet der GROUP BY Befehl noch einige weitere Möglichkeiten.

In der einfachen Variante kannst du ein Feld der internen Tabelle angeben, nach dem gruppiert werden soll.

```
LOOP AT documents INTO DATA(document)
  GROUP BY document-docnr INTO DATA(docgrp).
  WRITE: / docgrp.
ENDLOOP.
```

Das Ergebnis ist eine Liste aller eindeutigen Belegnummern:

```
0000012001
0000012002
0000012003
```

Gruppenelemente

Es ist nun eine Gruppe DOCGRP vorhanden. Auf die Zeilen dieser Gruppe kann mit Hilfe des Befehls LOOP AT GROUP zugegriffen werden:

```
  LOOP AT GROUP docgrp INTO DATA(docline).
    WRITE: / doc1line-docnr, docline-itmno.
  ENDLOOP.
```

So weit so gut...

Es gibt zwei Interessante Zusätze in einer Gruppe:

- GROUP INDEX
- GROUP SIZE

Diese können zusätzlich definiert werden. Im LOOP kann darauf zugegriffen werden.

GROUP INDEX

Der Zusatz GRIOUP INDEX liefert den aktuellen Index der (unsortierten) Gruppe.

GROUP SIZE

Der Zusatz GROUP SIZE liefert Informationen über die Anzahl der enthaltenen Gruppenelemente.

Beispiel

```
LOOP AT documents INTO DATA(document)
  GROUP BY
  ( doc   = document-docnr
    size  = GROUP SIZE
    index = GROUP INDEX )
  INTO DATA(docgrp).
  WRITE: / |elements in group {
    docgrp-index align = left } "{
    docgrp-doc }": {
    docgrp-size ALIGN = left } entries|.

  LOOP AT GROUP docgrp INTO DATA(doc).
    WRITE: / doc-docnr, doc-itmno.
  ENDLOOP.
ENDLOOP.
```

Ergebnis:

```
number of elements in group 1: 2 entries
0000012001 000010
elements in group 1 "0000012001": 2 entries
0000012001 000010
```

```
0000012001 000020
elements in group 2 "0000012002": 1 entries
0000012002 000010
elements in group 3 "0000012003": 2 entries
0000012003 000010
0000012003 000020
```

In diesem Fall müssen die Gruppenfelder, die nun in Klammern eingefasst werden müssen, selbst definiert werden.

Dynamische Gruppen

Jetzt kommt der spannende Teil, der einen großen Vorteil gegenüber der alten AT NEW Gruppenstufenverarbeitung hat. Die Gruppen können dynamisch anhand der Feldwerte definiert werden. Im Folgenden Beispiel fasse ich alle Einträge mit einem Löschkennzeichen in der Gruppe „deleted“ zusammen. Alle anderen Einträge kommen in die Gruppe „valid“.

```
LOOP AT documents
INTO DATA(doc)
GROUP BY (
    del_group = COND string( WHEN doc-del_flag = space THEN 'valid' ELSE
'deleted' )
    size  = GROUP SIZE
    index = GROUP INDEX  )
  INTO DATA(docgrp).
  WRITE: / 'number of', docgrp-del_group, 'entries:', docgrp-size.
ENDLOOP.
```

Ergebnis:

```
number of deleted entries:          2
number of valid entries:          3
```

Ich füge also je nachdem, ob das Löschhkennzeichen gesetzt ist oder nicht, eine andere Gruppenbezeichnung ein. Das ist besonders dann interessant, wenn man mehrere unterschiedliche Elemente gruppieren möchte. Beispielsweise alle Positionstypen, die mit A oder B beginnen:

```
LOOP AT documents
```

```
INTO DATA(doc)
GROUP BY (
    cat   = doc-category(1)
    size  = GROUP SIZE
    index = GROUP INDEX  )
  INTO DATA(docgrp).

  WRITE: / 'number of items in category',
    docgrp-cat LEFT-JUSTIFIED NO-GAP,
    docgrp-size.

ENDLOOP.
```

Ergebnis:

```
number of items in category A         3
number of items in category B         2
```

Ebenso könnte man nach Bestellungen gruppieren, die einen „geringen“ oder einen „höheren“ Bestellwert haben. Oder ich kann Aufträge direkt nach Aufträgen mit A-, B- oder C-Kunden gruppieren, indem ich für die Ermittlung der Kundenklassifizierung eine Methode in der GROUP BY Klausel verwende.

Gruppen gruppieren

Die definierten Gruppen können ebenfalls weiter gruppiert werden. Im folgenden Beispiel gruppiere ich erst nach der *Positionstypengruppe* (A oder B) aus dem obigen Beispiel. In dieser Gruppe gruppiere ich dann noch einmal nach dem eigentlichem Positionstyp (A1, A2, …). zudem berücksichtige ich in der WHERE-Bedingung nur die Positionen ohne Löschkennzeichen. Für dieses Beispiel habe ich die Datenbasis etwas erweitert…

```
LOOP AT documents
INTO DATA(doc2)
GROUP BY (
    cat   = doc2-category(1)
    size  = GROUP SIZE
    index = GROUP INDEX  )
  INTO DATA(docgrp2).

  WRITE: /1 'number of items in category',
    docgrp2-cat LEFT-JUSTIFIED NO-GAP,
```

```
    docgrp2-size.

  LOOP AT GROUP docgrp2 INTO DATA(docline2)
  WHERE del_flag = space GROUP BY ( category = docline2-category ) INTO
DATA(grpcat).
    write: /5 'category', grpcat-category.
    LOOP AT GROUP grpcat INTO DATA(cat).
      WRITE: /9 cat-docnr, cat-itmno, cat-category.
    ENDLOOP.
  ENDLOOP.

ENDLOOP.
```

Ergebnis:

```
number of items in category A         5
    category A1
        0000012001 000020 A1
        0000012003 000030 A1
    category A2
        0000012004 000010 A2
number of items in category B         5
    category B3
        0000012003 000010 B3
        0000012003 000020 B3
        0000012005 000010 B3
        0000012005 000020 B3
    category B1
        0000012006 000010 B1
```

Einschränkungen

Bei meinem heutigen Ausflug in die Gruppenstufen bin ich über folgende Einschränkungen gestolpert:

1. Sortierung nur nach Gruppenstufenfeldern möglich, aber nicht nach GROUP SIZE
2. Kein WHERE über Gruppenstufen möglich

Sortierung

Ich wollte gerne die Gruppen nach der Anzahl der Elemente sortieren. Das ist leider nicht möglich.

WHERE über Gruppenstufen

Es ist anscheinend nicht möglich, die erzeugten Gruppen direkt über eine WHERE-Bedingung einzuschränken. In den meisten Fällen kann man es sicherlich über eine geschickte WHERE-Bedingung über die Tabelle abbilden (zum Beispiel WHERE category(1) = ‚A'). Allerdings wäre eine Einschränkung über die Gruppen selbst eventuell auch hilfreich. Zum Beispiel könnte die Bedingung für die Gruppe DEL_GROUP, die ich mit *valid* und *deleted* definiert hatte, etwas komplizierter und aufgrund einen Methodenaufrufes nicht ersichtlich sein. Ich würde dann trotzdem nur über die Einträge mit *valid* verarbeiten wollen. Das geht natürlich mit CHECK innerhalb des LOOP, eine WHERE-Bedingung wäre jedoch eleganter.