KfK 2910
Mai 1980

# Die Report-Definitionssprache RDL und ihre Implementierung im Report-Generator FAREG

D. Stöckle, F. J. Polster
Institut für Datenverarbeitung in der Technik

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik

KfK 2910

# Die Report-Definitionssprache RDL

# und ihre Implementierung im Report-Generator FAREG

D. Stöckle

F.J. Polster

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

Zusammenfassung

Dieser Bericht beschreibt die Report-Definitionssprache, die für den Report-Generator FAREG entworfen wurde. Die Sprache stellt die Schnittstelle dar zwischen Report-Generator und dem Bediener, der den Report definiert. Sie ist problemorientiert und realisiert die wichtigsten Funktionen eines Report-Generators. Das Report-Programm ist in problembezogene Abschnitte entsprechend der Report-Struktur eingeteilt. Die einzelnen Anweisungen sind aus selbsterklärenden Kommandowörtern aufgebaut. Syntax und Semantik der Anweisungen werden ausführlich erläutert und an Programmbeispielen verdeutlicht. Zusätzlich werden Hinweise auf die Realisierung der Anweisungen und auf die Implementierung des Report-Generators FAREG gegeben.

# The Report-Definition Language RDL and its Implementation in the Report-Generator FAREG

Abstract

The report-definition language (RDL) is described which is implemented in the report-generator FAREG. This language serves as an interface between report-generator and the report-defining personnel. It is a problem-oriented descriptive language and implements the main functions of a report-generator. Each report-program is composed of special sections according to the structure of a report. To enhance readability the RDL-statements are formulated using self-explaining keywords. A complete description of syntax and semantics and some examples of RDL-programs are given. The implementation of the statements and of the report-generator FAREG is sketched.

Inhalt

Seite

## 1. Einleitung

Zur Beschreibung eines Reports sind die Standard-Programmiersprachen nur eingeschränkt tauglich, da sie im allgemeinen nicht über die speziellen Operatoren und Objekttypen verfügen. Deshalb werden die erforderlichen Funktionen in Spezialsprachen realisiert. Eine Einbettung der entsprechenden Sprachkonstrukte in eine Standard-Programmiersprache ist dann sinnvoll, wenn sich der Anwendungsbereich der Sprache über ein breites Spektrum von Funktionen erstreckt /GRID 76/.

Die Anwendung von Report-Generatoren konzentriert sich auf die Funktionen Datenauswahl, Datenverknüpfung, Datenaufbereitung und Ausgabesteuerung /POST 78/. Zur Definition eines Reports empfiehlt es sich, eine Spezialsprache zu verwenden. Eine solche Report-Definitionssprache (RDL = report-definition language) stellt eine wesentliche Komponente der Anwenderschnittstelle eines Report-Generators dar.

Dieser Bericht stellt zunächst die wichtigsten Sprachmerkmale heraus. Der Programmaufbau wird exemplarisch erläutert. Die Anweisungen werden in ihrer Funktion, Bedeutung und Syntax einzeln beschrieben. Zusätzlich werden Hinweise zur Realisierung, insbesondere zur Übersetzung, der Anweisungen gegeben. Einige Programmbeispiele veranschaulichen das Konzept der Sprache.

## 2. Sprachmerkmale

Report-Definitionssprachen (RDL) und Anfragesprachen zu Datenbanken weisen in der Aufgabenstellung Gemeinsamkeiten auf. Beide enthalten Operatoren zur Datenauswahl und -ausgabe. Der Unterschied zwischen beiden liegt darin, daß bei Report-Definitionssprachen die Datenmanipulation auf die Retrieval-Funktionen beschränkt und für die Ausgabe mehr Möglichkeiten vorhanden sind.

Programmiersprachen lassen sich einteilen in deskriptive Sprachen, die das Ziel beschreiben, und prozedurale Sprachen, die den Weg zum Ziel beschreiben. Die hier entworfene RDL hat deskriptiven Charakter mit zusätzlichen prozeduralen Sprachelementen. Eine eindeutige Klassifizierung ist wie bei Anfragesprachen /SHNB 78/ nicht möglich.

Der deskriptive Charakter der Sprache entspricht der Denkweise der Benutzer eines Report-Generators. Wie in /POST 78/ festgestellt, sind dies außer dem Systempersonal die sogenannten anspruchsvollen Laien, deren Denken sich mehr auf Anwendungsprobleme als auf Programmabläufe konzentriert. Sie sind an einer ergebnisorientierten Darstellung der Daten aus der Datenbank (Problemdaten) interessiert.

Deklarationen werden soweit möglich implizit erledigt. So werden z.B. Attributtypen und Recordtypen nach den entsprechenden Typen in der Datenbank bestimmt /PREC 78/. Dies trägt dazu bei, den Programmumfang einzuschränken und dem Benutzer in seiner Anwendersicht entgegenzukommen. Soweit möglich werden per Default Standardwerte gesetzt, so daß Komplexität und Umfang eines Programmes in einem angemessenen Verhältnis zum Anwendungsproblem stehen. Einfache Anwendungen lassen sich mit wenigen Anweisungen lösen (siehe Beispiele 1, 2 in Kap. 5), da viele Programmparameter mit Defaultwerten vorbesetzt sind. Je nach Anforderung genügt es für den Anwender, eine bestimmte Teilmenge von Anweisungen zu kennen.

Zur Steuerung des Programmablaufes im prozeduralen Teil sind Kontrollstrukturen vorgesehen /DIJE 72/. Somit kann auf eine Sprunganweisung verzichtet werden. Das Report-Programm wird dadurch übersichtlicher und das Programmieren weniger fehleranfällig. Zur Strukturierung trägt auch die Einteilung des Programms in Abschnitte bei. Die Lesbarkeit eines Programms wird durch selbsterklärende Wortsymbole unterstützt (keine mnemot. Abkürzungen).

Die Report-Definitionssprache setzt bei den Anwenderdaten ein relationales Datenmodell /CODE 70/ voraus. Die für die Anwendung wichtigsten Operationen auf Relationen (Projection, Join) sind in der Sprache realisiert.

Als Vorbilder beim Entwurf der Sprache dienten die Sprachen der Report-Generatoren

- SYSTEM 2000 /SYST 74/: Programmstruktur und Zuordnung von Programmabschnitten zu Report-Objekten
- Information Management System /CAGE 76/: einzelne Anweisungen
- QUERY /QUER 76/: Anweisungsparameter.

Weitere Anregungen, insbesondere für die Anweisungen zur Datenauswahl, gehen auf die Anfragesprachen SEQUEL /CHAD 76/ und TAMALAN /VANE 77/ zurück.

# 3. Programmaufbau und -übersetzung

## 3.1 Struktur des RDL-Programms

Jedes Report-Programm besteht aus einer Reihe von Programmabschnitten, die jeweils einen bestimmten Teil des Reports beschreiben oder eine inhaltlich zusammengehörende Verknüpfungsvorschrift enthalten. Es gibt optionale Programmabschnitte ([...]) und solche, die in jedem Programm unbedingt spezifiziert werden müssen.

```
[Initialisierung]
 Datenauswahl
[Report-Aufbereitung]
 Ausgabe
```

Die Gliederung des Programmabschnittes Report-Aufbereitung geht von der Grundstruktur eines Reports aus, der in Report-Kopf, -Rumpf und -Abschluß eingeteilt ist.

Bild 3-1 zeigt den prinzipiellen Aufbau eines Report-Programms. Die Abschnitte Initialisierung und Report-Aufbereitung sind optional.

### 3.1.1 Initialisierung

Hier können Report, Seitengröße und Variablen initialisiert werden. Variablenwerte können durch Wertzuweisung (4.5.1) oder Eingabe gesetzt werden (4.3.1). Für Report und Seite sind Defaultwerte vorgesehen. Der Abschnitt ist optional.

### 3.1.2 Datenauswahl

Durch die Anweisungen zur Datenauswahl (4.4) werden die Problemdaten für die Report-Aufbereitung bereitgestellt. Jedes Report-Programm muß mindestens eine entsprechende Anweisung (RETRIEVE 4.4.1) enthalten.

| Programmabschnitt | Anweisungsbeispiele ; Kommentar |
|---|---|
| Initialisierung | *REPORT ... ; Report-Deklaration*<br>*PAGE ... ; Seitengröße*<br>*READ ... ; Einlesen von Parametern*<br>*V1=0 ; Vorbesetzen von Variablen*<br>*V2=0 ;* |
| Datenauswahl | *RETRIEVE ...*<br>*JOIN ...* |
| Report-Aufbereitung | *WITH record DO ; Zuordnung des zu verarbeitenden Records* |
| a) Report-Kopf | *HEAD OF REPORT ; Report-Kopf*<br>*PAGENO ; Seitennummer*<br>*COLUMN ... ; Spaltenüberschrift*<br>⋮<br>*ENDHEAD* |
| b) Report-Rumpf | *; Report-Rumpf*<br>*A = B + C ; Datenverknüpfung*<br>⋮<br>*CREATE record ... ; Ausgabe-Record*<br>*SORT ... ; Sortierung*<br>⋮ |
| c) Report-Abschluß | *TAIL OF REPORT ; Report-Abschluß*<br>*SUM ALL ; Globalwerte*<br>*ENDTAIL* |
| | *ENDWITH* |
| Ausgabe | *PUT outrecord ON PRINTER ; Ausgabe d. Reports* |
| | *END ; Ende Report-Programm* |

Bild 3-1: Prinzipieller Aufbau eines Report-Programms.

### 3.1.3 Report-Aufbereitung

Zunächst muß bestimmt werden, welche der in 3.1.2 bereitgestellten Problemdaten für den Bericht aufbereitet werden sollen. Dies geschieht zumeist implizit. Eine explizite Zuordnung ist dann notwendig wenn

- mehrere Records (Typen) bei der Datenauswahl erzeugt wurden und
- ein anderer als der zuletzt im Programmteil referierte Record zugeordnet werden soll (4.6).

Dieser Abschnitt hat die folgenden drei Teilabschnitte:

a) Report-Kopf

Definition von Report- und Seiten-Kopf ist nur dann per Anweisung erforderlich, falls abweichend oder umfangreicher als durch Default vorgesehen (4.6.1). Es können insbesondere Überschriften definiert werden.

b) Report-Rumpf

Hier werden die Problemdaten aufbereitet und verknüpft. Im Gegensatz zu allen anderen Programmabschnitten werden die Anweisungen im Report-Rumpf für jede Ausprägung (Tupel) des zugeordneten Recordtyps einmal durchlaufen (Bild 3-2). Die Formatierung ist per Default geregelt, so daß dieser Teilabschnitt entfallen kann, falls keine Datenverknüpfung erforderlich ist.

c) Report-Abschluß

In diesem Abschnitt kann die Berechnung von report- oder seitenabhängigen Globalwerten spezifiziert werden. Dafür existieren keine Defaultwerte.

### 3.1.4 Ausgabe

Zur Steuerung der Ausgabe ist gegenwärtig eine Anweisung vorgesehen, die logische und physikalische Seitengröße gleichsetzt.

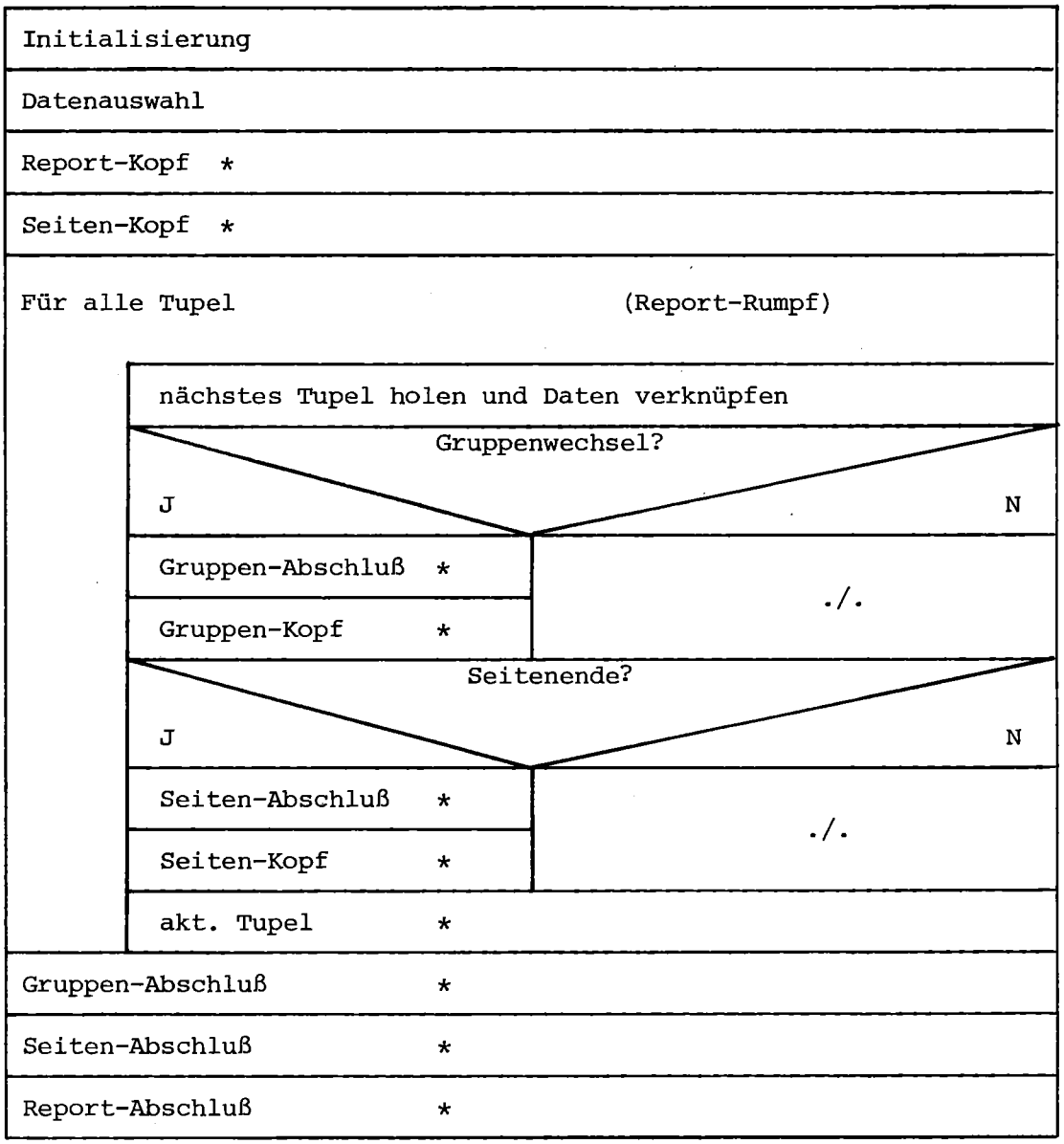



Bild 3-2: Vereinfachtes Ablaufschema eines Report-Programmes.

(*: Aufbereitung und Ausgabe).

## 3.2 Struktur der übersetzten RDL-Programme

Der vom RDL-Compiler erzeugte Code (Interpreter-Sprache) weist grundsätzlich die in Bild 3-3 gezeigte Struktur auf. Der starre Programmaufbau entspricht dem Charakter der Report-Definitionssprache, die eher deskriptiv als prozedural und auf die Beschreibung eines speziellen Produkts mit fester Struktur ausgerichtet ist.

Dargestellt werden die wichtigsten Schritte zur Erzeugung eines Reports. Es sind folgende Gesichtspunkte zu beachten:

- Die Ausführung bestimmter Schritte hängt vom jeweiligen RDL-Programm ab.
- In der Darstellung wird nicht unterschieden zwischen Default-Anweisungen und expliziten Programmbefehlen.
- Die Reihenfolge der Schritte im übersetzten Programm weicht von der Reihenfolge der entsprechenden Anweisungen im RDL-Programm ab. Um trotzdem eine Ein-Pass-Übersetzung zu ermöglichen, müssen die von der Sequenz abweichenden Anweisungsteile bei der Übersetzung in einem vom übrigen Interpretercode getrennten Bereich abgelegt werden.
- Defaultvereinbarungen für Anweisungsparameter und Programmteile; Defaultfälle werden vom Übersetzer erkannt und die entsprechenden Werte automatisch eingesetzt.

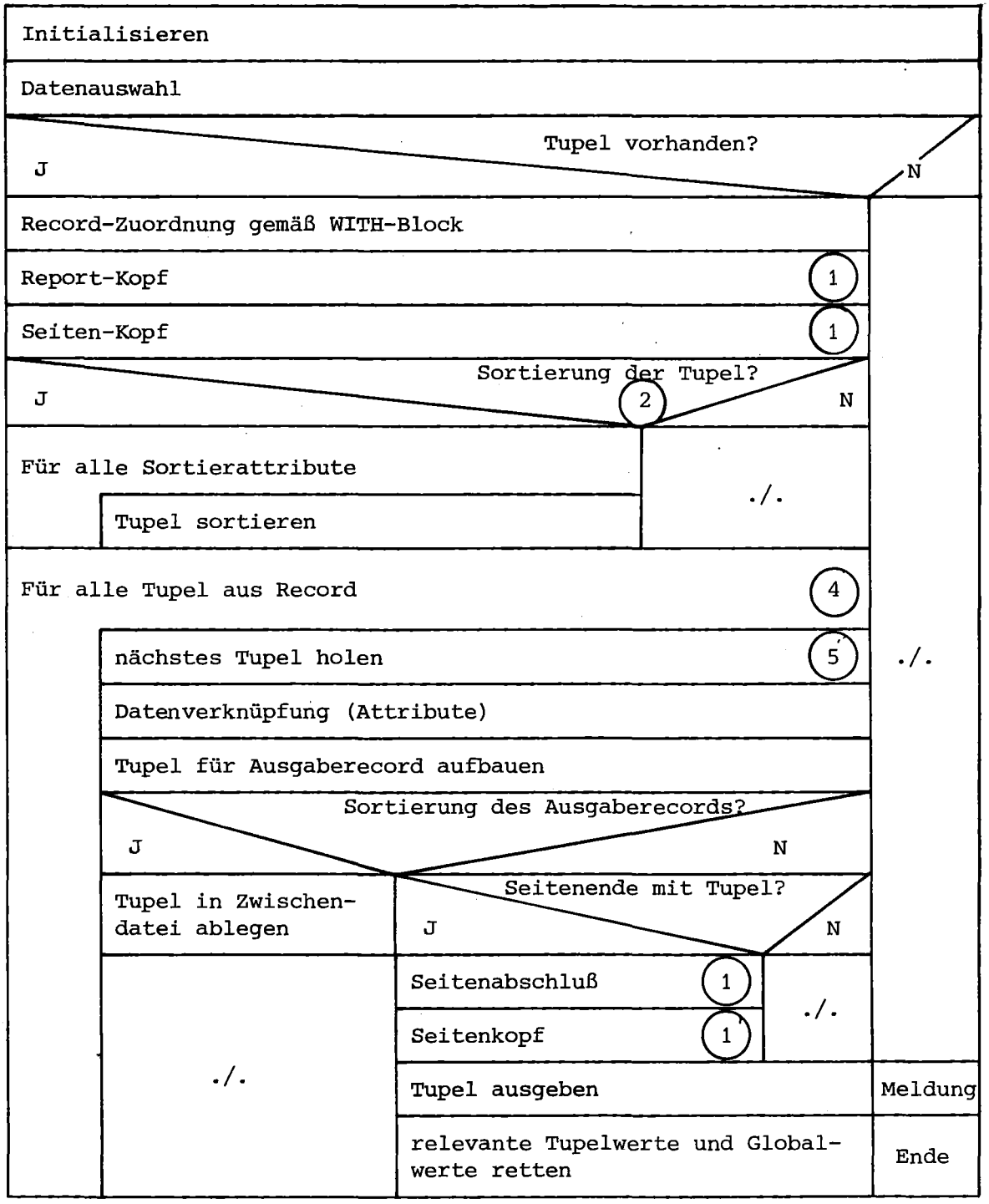



Bild 3-3a: Ablaufschema eines übersetzten RDL-Programms (Teil 1).

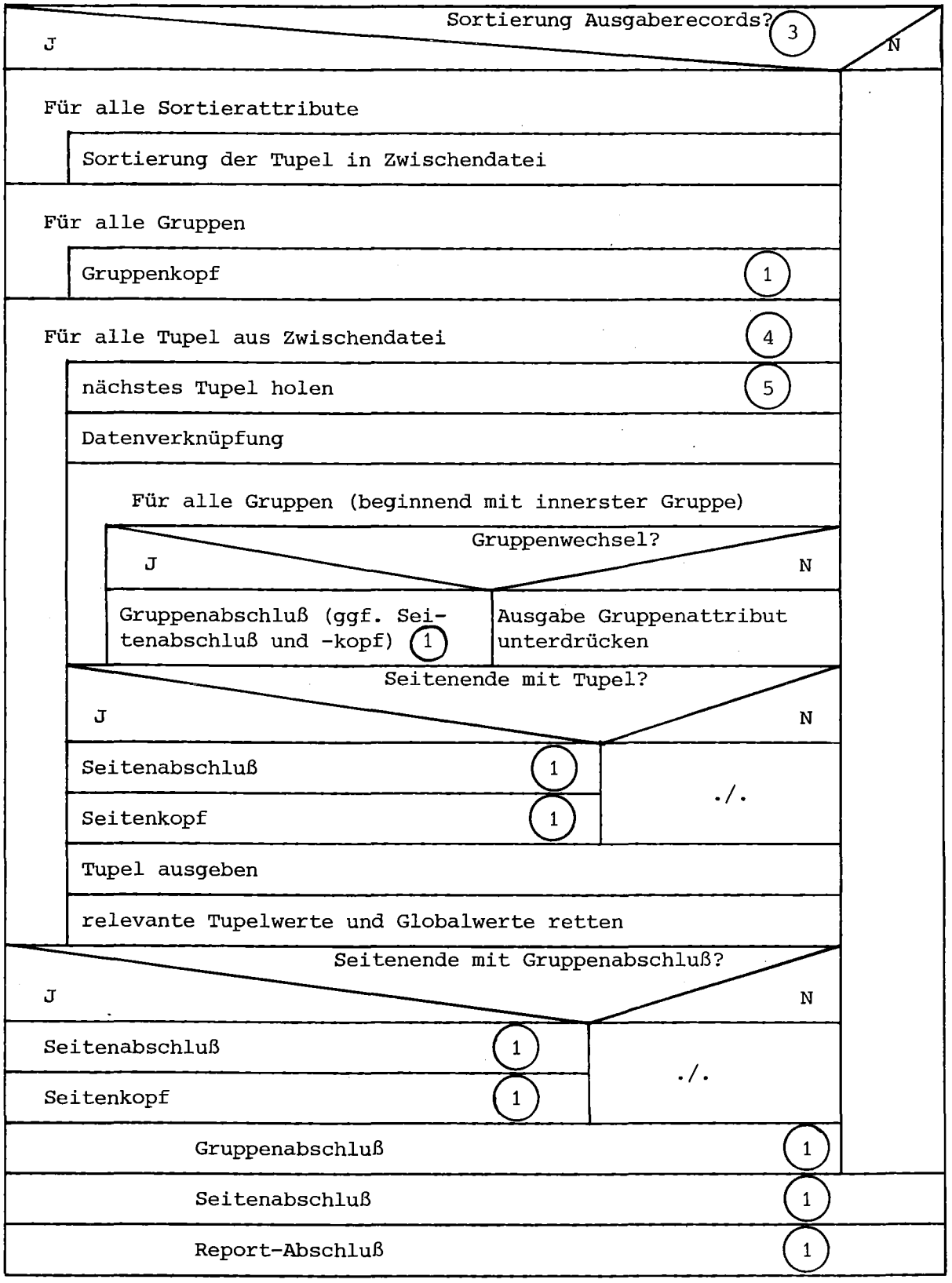



Bild 3-3b: Ablaufschema eines übersetzten RDL-Programms (Teil 2).

Erläuterungen zu Bild 3-3a und 3-3b:

(Die Ziffern weisen auf die entsprechenden Stellen im Ablaufschema (Bild 3-3a, 3-3b) hin).

(1) Die genannten Report-Strukturobjekte (Report, Seite, Gruppe) werden jeweils bearbeitet, also ggf. objektbezogene Datenverknüpfung, Globalwerte, Aufbereitung und Ausgabe von Daten.

(2), (3) Sortier-Anweisung kann sich beziehen auf:

- Report-Record (2) (durch Datenauswahl erzeugt)
- Ausgabe-Record (3) (im Report-Rumpf definiert).

Der Bezug wird durch die Stellung der SORT-Anweisung im RDL-Programm definiert (siehe 4.6.2c).

(4) Diese Schleifen enthalten die eigentliche Bearbeitung der einzelnen Tupel aus dem jeweiligen Record. Nur hier ist die Verwendung der SKIP-Anweisung sinnvoll; sie bewirkt einen Sprung an den Schleifenanfang, d.h., die auf SKIP folgenden Befehle bis zum Schleifenende werden übersprungen.

(5) Vorausgesetzt wird ein Datenbanksystem, das im 'one-tuple-at-a-time'-Verfahren die Daten übergibt.

## 3.3 Übersetzungsprinzip

Jedes (übersetzte) RDL-Programm wird bei Ausführung unterpretativ abgearbeitet /POLF 79/. Ein Compiler übersetzt zuvor den RDL-Code in eine dem Interpreter verständliche Sprache (IL = interpreter language) (Bild 3-4).

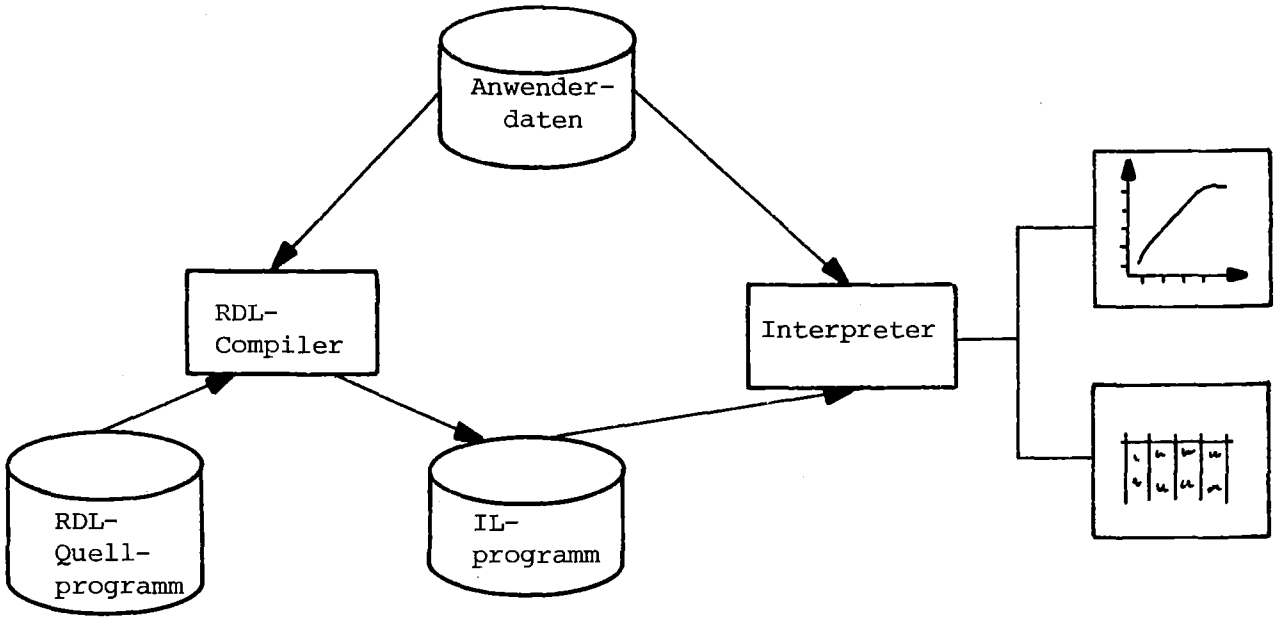



Bild 3-4: Bearbeitung eines RDL-Programms.

Einmal übersetzt und abgespeichert, kann ein RDL-Programm als IL-Code beliebig oft interpretiert werden.

Der RDL-Compiler hat folgende Merkmale:

- Ein-Pass-Übersetzung; dies wird unterstützt durch die Sprachkonzepte: geschlossene Kontrollstrukturen, einheitlicher Programmaufbau, Verzicht auf Sprunganweisungen.

- Erledigung möglichst vieler Aufgaben zur Übersetzungszeit; dazu gehören: Initialisierungen, Prüfung von Namen und Typen, Kontrolle der RDL-Programmstruktur.

## 4. Beschreibung der Anweisungen

Die Anweisungen werden einzeln beschrieben. Die Semantik wird verbal erläutert. Zur Beschreibung des Aufbaus werden die folgenden Symbole verwendet:

| *Symbol* | *Bedeutung (xxx, yyy: Teil einer Anweisung)* |
|---|---|
| [*xxx*] | *Option; der in [...] eingeschlossene Anweisungsteil ist optional, d.h. ist nicht zwingend vorgesehen.* |
| {*xxx*} | *Iteration; der in {...} eingeschlossene Anweisungsteil kann mehrfach wiederholt werden.* |
| \|*xxx*<br>*yyy*\| | *Alternative; entweder xxx oder yyy* |
| *ABCD...* | *Schlüsselwort der Anweisung; fest vorgegeben (Großbuchstaben)* |
| *abcd...* | *Meta-Variable (Kleinbuchstaben)* |
| < > | *zusammengesetzter Anweisungsteil; Verfeinerung anschließend* |

### 4.1 Objekte der Report-Definitionssprache

RDL kennt folgende Objekte, die implizit oder explizit in den Anweisungen als Operanden enthalten sind:

(1) Attribut bezeichnet die Werte eines Elementes in einer Relation der Datenbank. Die Kopie eines Wertes im Report-Programm wird ebenso mit Attribut bezeichnet. Bei Programmübersetzung wird der Attributtyp aus dem Datenbankschema gelesen und im RDL-Programm abgelegt.

(2) Variable: frei wählbarer Bezeichner eines elementaren Wertes, der im Report-Programm erzeugt wird. Die Deklaration erfolgt implizit bei der ersten Wertzuweisung.

(3) File bezeichnet eine Relation in der Datenbank; enthält im Report-Programm keinen Wert.

(4) Record: frei wählbarer Bezeichner für die Menge der im Report-Programm zu verarbeitenden Attribut- und Variablenwerte (Anwenderdaten).

Der Recordtyp setzt sich aus den Bezeichnern und Typen der im Record enthaltenen Attribute und Variablen zusammen. Die Deklaration erfolgt implizit bei der ersten Wertzuweisung.

(5) Tupel: Satz elementarer Werte eines Records. (Ausprägung eines Records).

(7) Konstante sind elementare Werte, die Variablen zugewiesen oder direkt in eine Ausgabezeile geschrieben werden können.

(8) Report-Strukturobjekte: Bericht (report), Seite (page), Gruppe (group), Zeile (row) sind in /POST 78/ definiert.

Ein Report erstreckt sich i.a. über mehrere Seiten. Report, Seite und Gruppe bestehen aus Kopf (head), Rumpf und Abschluß (tail). Der Report-Rumpf kann in Gruppen, das sind Mengen von Tupeln mit demselben Wert im Gruppenattribut, gegliedert werden.

Mit Spalte (column) wird die Menge der Werte eines Attributes oder einer Variablen im Report bezeichnet.

(9) Graphische Objekte:

- Graph, Bargraph beschreibt ein Achsenkreuz, die darzustellenden Objekte und ihre Darstellungsform
- Bild (picture) enthält ein oder mehrere Achsenkreuze (entspricht einer Seite).

Als Datentypen für Attribute und Variablen sind vorgesehen:

| | | |
|---|---|---|
| INT | : Integer; I*2 , I*4 | alle Deklarationen implizit. |
| REAL | : Real-Zahl; R*4 , R*8 | |
| STRING/length: | Zeichenkette/Anzahl der Zeichen | |

Numerische Konstante werden als Ziffernfolge angegeben; string-Konstante werden in '...' eingeschlossen.

Die Objekttypen sind nach der Deklaration den Objekten fest zugeordnet ('strong typing').

## 4.2 Initialisierungsanweisungen

Diese Anweisungen dienen zur Deklaration des Reporttyps und der Bestimmung der logischen Seitengröße. Daneben können Variablen Werte zugewiesen werden durch:

- Wertzuweisungen (elementare Datenverknüpfung 4.5.1). Die Zuweisung und Verknüpfung von Attributwerten ist bei der Initialisierung nicht möglich, da Attributwerte nur bei Report-Aufbereitung existieren.
- Benutzerkommunikation (READ-Anweisung 4.3.1).

### 4.2.1 Report-Deklaration

| *REPORT report-programmname [report-type]* |
|---|

Bedeutung:

Festlegung von Report-Programmbezeichnung und Report-Typ.

- report-programmname: Name zur Identifikation des Report-Programmes
- report-type: Typ des durch das Programm zu erzeugenden Reports;
  mögliche Werte:
  - ALPHA : alphanumerische Form (Defaultwert)
  - GRAPH : graphische Form
  - FILE : interne Form; Speicherung in einer Datei.

Typspezifische Anweisungen im Programm werden bei Unverträglichkeit mit dem Report-Typ überlesen.

### 4.2.2 Seiten-Initialisierung

| *PAGE* *pagelength, rowlength* |
|---|

Bedeutung:

Festlegung der logischen Seitengröße.
(Nur bei alphanumerischer Ausgabe)

- pagelength: Anzahl der Zeilen pro log. Seite (einschließlich Leerzeilen)
- rowlength: Anzahl der Zeichen pro Zeile, Seitenbreite (einschließlich Leerzeichen).

Default für Seitengröße: Standard-Druckerseite.

## 4.3 Ein-/Ausgabe-Anweisungen

### 4.3.1 Benutzerkommunikation

Anwendung:

Kommunikation mit dem Benutzer zur Laufzeit:

- Ausgabe von Meldungen
- Eingabe von Variablenwerten
  (Parameterversorgung)

*READ* [*FROM devicename*] {*variablename* *formatcode*}

Bedeutung:

Einlesen von Variablen- bzw. Parameterwerten

- devicename: Kennung des Eingabegerätes
  Default: Systemterminal
- variablename: implizite Deklaration einer Variablen;
  Typ wird durch zugeh. formatcode bestimmt.
- formatcode: Formatsymbole geben Typ und Länge der einzulesenden Größe an (vgl. 4.6.3).

*WRITE* [*ON devicename*] {[*constant*] [*variablename* *formatcode*]}

Bedeutung:

Ausgabe von Meldungen

- devicename: wie bei READ entsprechend.
- constant : bel. auszugebende Zeichen-Kette.
- variablename: Wert d. Variablen als Ausgabe
- formatcode: wie bei READ entsprechend.

### 4.3.2 Ausgabe des Reports

Zur Beschreibung der Report-Ausgabe auf das dafür vorgesehene Gerät dient folgende Anweisung:

```
PUT  { outrecordname | picturename }  ON  devicename  [{output-parameter}]
```

Bedeutung:

- outrecordname: Bezeichnung des im auszugebenden Report enthaltenen Records, der entweder im Report-Rumpf (durch CREATE) oder bei der Datenauswahl definiert wurde. Enthält ein Report verschiedene Ausgabe-Records, dann werden diese in der Anweisung aufgezählt. (Nur bei Report-Typ ALPHA und FILE)
- picturename: Bezeichnung des auszugebenden Bildes (4.6.2) (bei Report-Typ GRAPH).
- devicename: Bezeichnung des Ausgabegerätes
- output-parameter: geräteabhängige Parameter.

Diese Anweisung stellt die Zuordnung zwischen Report bzw. Teilen davon und Ausgabegerät her. Die Ausgabe selbst erfolgt jeweils bei Aufbereitung des entsprechenden Ausgabeobjekts (vgl. Bild 3-2).

## 4.4 Anweisungen zur Datenauswahl

Mit Hilfe dieser Anweisungen werden die Anwenderdaten aus der Datenbank für die Report-Erzeugung bereitgestellt. Die Anweisungen haben die allgemeine Form:

*recordname = {<operator> <operand-specification>}*

Bedeutung:

Wertzuweisung an die Recordvariable recordname /HUIM 75/.

- recordname : Bezeichnung des Report-Records, der das Ergebnis der Operation enthält.

  Report-Record wird implizit deklariert, indem im Datenbankschema auf die Attributdeskription der angesprochenen Relation zugegriffen wird.

- operator : Bezeichnung der Operation (Selektion und Extraktion, Zusammenfügen, Decodierung)

- operand-spec. : enthält Operanden (attribute, variable, constant file, record) und ggf. Erweiterung der Operation.

Realisierung:

Die Funktionen zur Datenauswahl werden zu Aufrufen an das Datenbanksystem FADABS umgesetzt. Dies geschieht über die normale Anwenderprogrammschnittstelle, die Datenmanipulationssprache FORDAM /POLF 78/. Die Funktionen zum Einrichten und Löschen von Dateien und zum Lesen von Relationen- und Attributdeskriptionen (Namen, Attributtypen) stehen an einer speziellen Schnittstelle zur Verfügung.

Einschränkungen:

Zur Vereinfachung der Implementierung sind in RDL-Programmen folgende Bedingungen einzuhalten:

(a) syntaktisch

Reihenfolge der Anweisungen im RDL-Programm wie folgt:

```
  {RETRIEVE ...}
 [{JOIN ...    }]
  [DECODE ...  ]
  [WITH ...    ]
```

(b) inhaltlich

| | |
|---|---|
| ⋮ | ⋮ |
| *a = JOIN r WITH s* | *a = JOIN r WITH s* |
| *b = JOIN a WITH t* | *b = DECODE IN a ...* |
| *c = JOIN b WITH u* | ⋮ |
| ⋮ | |

In aufeinanderfolgenden JOIN-Anweisungen wird bei jedem JOIN der zuvor erzeugte Record weiterverarbeitet; ebenso falls ein DECODE auf ein JOIN folgt.

### 4.4.1 Selektion und Extraktion

Selektion der Tupel, deren Attribute die Qualifikation erfüllen, aus einer bestimmten Datei und Extraktion von Attributen aus den Tupeln (Projection):

*recordname* = *RETRIEVE* | {*attributename*} | *FROM* *filename*
| *ALL* |

[*WHERE* < *qualification* >]

Bedeutung:

- Implizite Deklaration des Records recordname; der Recordtyp setzt sich aus den angegebenen Attributen zusammen (Attributtypen gemäß Datenbankschema).
- Selektion der Tupel, die der Qualifikation genügen, aus der Datei filename.

  <*qualification*> besteht aus Einzelvergleichen:

  *attributename* <*rop*> | *constant* / *variablename* |

  Einzelvergleiche können durch die logischen Operatoren AND/OR verknüpft werden (vgl. Syntaxbeschreibung). Zur Festlegung der Auswertungsreihenfolge von logischen Operatoren ist eine Klammerung möglich. (Auswertung ohne Klammern von links nach rechts).

  <*rop*> : Vergleichsoperator (= | < | ≤ | > | ≥ | ≠)

- Extraktion der durch attributename spezifizierten Attributwerte aus den selektierten Tupeln und Zuweisung an das Tupel des Records recordname.

  *ALL* : alle in der Datei filename enthaltenen Attribute werden in den Report-Record übernommen (ohne Extraktion).

Bemerkung:

Aus Gründen der Vereinfachung und Platzersparnis wurde auf eine getrennte Realisierung der Funktionen Selektion und Extraktion verzichtet und dafür nur eine Anweisung vorgesehen.

### 4.4.2 Zusammenfügen

Zusammenfügen der Tupel recordname 1, recordname 2 zu einem Tupel recordname bei Gleichheit in einem bestimmten Attribut (attributename 1 und ggf. attributename 2) /CODE 70/.

Voraussetzung: Attribute attributename 1 und 2 stimmen im Typ überein.

*recordname = JOIN recordname 1 WITH recordname 2*

*IN attributename 1 [attributename 2]*

Bedeutung:

- Deklaration des Records recordname; der Recordtyp wird gebildet aus den Typen der Records recordname 1 und recordname 2. (Bei identischen JOIN-Attributen, d.h. attributename 1 = attributename 2, wird nur attributename 1 in den resultierenden Record eingefügt.)
- Zusammenfügen der Tupel aus recordname 1 und recordname 2 zu einem Tupel und Zuweisung an das Tupel aus Record recordname; Bedingung:

  Wert (attributename 1) = Wert (attributename 2)

  wobei: attributename 1 aus recordname 1
  attributename 2 aus recordname 2

  (falls: attributename 1 = attributename 2
  dann: attributename 2 in der Anweisung optional)

  Tupel (recordname 1 oder recordname 2), die in attributename 1 bzw. attributename 2 keine gleichen Werte aufweisen, werden nicht zusammengefügt.

Bemerkung:

Für die JOIN-Funktion ist eine eigene Anweisung vorgesehen, um einen gewissen Grad an Transparenz zu gewährleisten. In SEQUEL z.B. ist diese Funktion in der Selektionsanweisung enthalten /CHAD 76/.

### 4.4.3 Decodieren

Durch das Decodieren wird ein interner Attributwert durch den externen Attributwert ersetzt anhand einer Decodiertabelle.

```
recordname = DECODE IN int.-recordname
             {int.-attributename TO ext.-attributename
             USING dec.-filename}
```

Bedeutung:

- Implizite Deklaration des Records recordname; der Recordtyp wird gebildet aus dem Typ des Records int.-recordname, wobei int.-attributename durch ext.-attributename (aus der Decodierdatei dec.-filename) ersetzt wird.
- Zuweisung der Attributwerte von int.-recordname an das Tupel des Records recordname. Das Attribut ext.-attributename erhält aus der Decodierdatei dec.-filename den Wert, der durch int.-attributename identifiziert wird.

Voraussetzung:

Decodierdatei enthält die Attribute int.-attributename und ext.-attributename mit eindeutigen Werten im Attribut int.-attributename.

## 4.5 Anweisungen zur Datenverknüpfung

Es sind nur Operatoren zur Verknüpfung numerischer Werte vorgesehen. Anweisungen zur Manipulation von Zeichenketten sind nicht in der RDL enthalten.

Laufzeitfehler, die sich aus Bereichsüberschreitungen ergeben, werden im Interpreterprogramm abgefangen.

### 4.5.1 Elementare Datenverknüpfung

Dabei werden neben Konstanten und Variablenwerten jeweils nur Attributwerte aus dem aktuellen Tupel des Report-Records verknüpft.

| *variablenname = arithmetic-expression* |
|---|

Wertzuweisung

Bedeutung:

- Die Variable variablename erhält den Wert des arithmetischen Ausdrucks arithmetic-expression nach dessen Auswertung.
- Die Syntax des arithmetischen Ausdrucks entspricht den FORTRAN-Regeln. Als Operatoren sind die Grundrechenarten und die in FORTRAN eingebauten Funktionen vorgesehen.

  Operanden: Variable, Attribute (nur im Abschnitt Report-Aufbereitung), Konstanten.

Realisierung:

(a) Typbestimmung des arithmetischen Ausdrucks (AA)

- falls alle Operanden vom selben Typ,

  dann: Typ (AA) = Typ (Operand)

- falls Operanden verschiedenen Typs (mixed mode),

  dann: Konversion nach der Hierarchie I*2 → I*4 → R*4 → R*8, so daß in einer Operation ein Operand der niedrigeren Hierarchiestufe in den Typ des Operanden mit der höheren Hierarchiestufe konvertiert wird (Konversionsoperatoren im Interpretercode).

(b) Typbestimmung bzw. -prüfung der Ergebnisvariablen

- falls Variable variablename nicht deklariert:

  Deklaration: Typ (Variable) = Typ (arithm. Ausdruck)

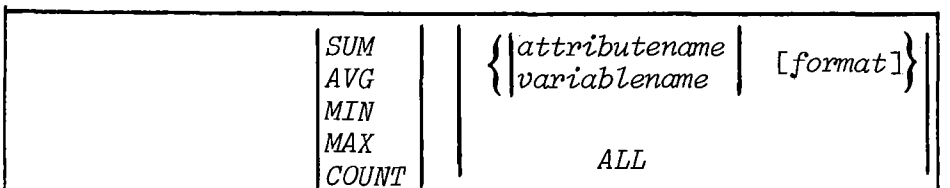
- falls Variable bereits deklariert: ggf. Typgleichheit zwischen Variable und arithm. Ausdruck herstellen (Konversionsoperatoren im Interpretercode), indem der Wert des arithm. Ausdrucks in den Typ der Ergebnisvariablen gewandelt wird.

(c) Auswertung des arithm. Ausdruckes und Ergebniszuweisung.

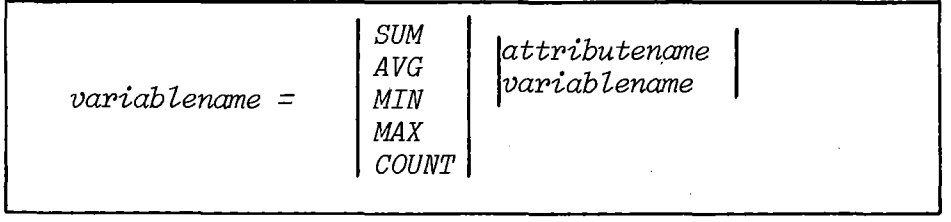
### 4.5.2 Globale Datenverknüpfung

Diese Anweisungen beziehen sich auf eine Gruppe, Seite oder den Report. Der Bezug wird implizit dadurch hergestellt, daß die Anweisung zur globalen Datenverknüpfung im Abschluß-Programmteil (TAIL ...) von Gruppe, Seite oder Report aufgeführt wird. Dort wird auch das Ergebnis bestimmt (Anweisungen zur globalen Datenverknüpfung nur in TAIL ... ENDTAIL zulässig).

(A)




(B)




Bedeutung:

Anweisung (A) enthält Verknüpfung und Ergebnisausgabe.

Anweisung (B) enthält nur Verknüpfung und Wertzuweisung (keine Ausgabe)

- Operatoren:

| | | | |
|---|---|---|---|
| SUM | : | Summe der Operanden | } Ausprägungen |
| AVG | : | Durchschnitt der Operanden | |
| MIN | : | Minimum der Operanden | |
| MAX | : | Maximum der Operanden | |
| COUNT | : | Anzahl an Operanden | |

- Operanden: Variable, Attribute des Ausgabedatensatzes;
  ALL bezeichnet alle Elemente des Ausgabedatensatzes

- format: formatcode und position (siehe 4.6.3). Nur erforderlich, falls Ausgabe des Ergebniswertes nicht im Standardformat erwünscht

- variablename: Ergebnisvariable; implizite Deklaration;
  Typ ergebnisabhängig (nur erforderlich, falls Ergebnis weiter verknüpft werden soll).

Realisierung:

(a) Typbestimmung des Resultats abhängig vom Operator:

- bei SUM, MIN, MAX: Typ des Operanden
- bei AVG : R*4, falls Typ (Operand) = I oder R*4
  R*8, falls Typ (Operand) = R*8
- bei COUNT : I*4

(b) Bestimmung der Ergebnisvariablen:

- bei Anweisung A: pro Operand implizite Deklaration einer Variablen mit Typ (Variable) = Typ (Resultat).
- bei Anweisung B: Deklaration der Variablen (falls noch nicht deklariert, sonst gemäß 4.5.1(b)): Typ (Variable) = Typ (Resultat).

(c) Ergebnisvariable mit (operatorabhängigen) Basiswerten jeweils zu Beginn des entsprechenden Strukturobjekts initialisieren (entspr. Interpretercode):

| Operator | Basiswert (typspezifisch) |
|---|---|
| SUM, COUNT | 0 |
| AVG | 0, K=0 (K = akt. Anzahl Records) |
| MIN | X } X= größte darstellbare Zahl |
| MAX | -X } |

(e) Formatierung (nur bei Anweisung A)

- Standardformat: (Default)
  - . pro Operator eine Zeile
  - . typabhängiger Standardformatcode
    (siehe 4.6.3), wobei:
    Länge (Ergebnisvariable) < Länge (Spalte des entspr. Operanden).
  - . Spaltenrichtige Positionierung der Ergebnisvariablen, d.h.:
    printposition = Beginn entspr. Spalte im Report-Ausgaberecord.

- explizites Format

  . Aufbau des Ausgabeformates entsprechend der Reihenfolge in der Anweisung

  . bei expliziter Positionierung keine Überprüfung auf Übereinstimmung mit Spaltendefinition.

(d) Pro Report-Record Berechnung der Ergebnisvariablen (im Report-Rumpf):

| Operator | Berechnungsvorschrift |
|---|---|
| SUM | v = v + operand |
| AVG | v = (v·k + operand)/(k+1)<br>k = k+1 (akt. Anzahl) |
| MIN | IF v > operand THEN v = operand |
| MAX | IF v < operand THEN v = operand |
| COUNT | v = v + 1 |

## 4.6 Anweisungen zur Datenaufbereitung

Es wird unterschieden zwischen Anweisungen

- zur Definition von Report- bzw. Seiten-Kopf und -Abschluß (Report-Rahmen)
- zur Definition des Report-Rumpfes.

Diese Anweisungen beziehen sich auf einen bestimmten Record, der per Datenauswahl erzeugt wurde. Um diesen Bezug herzustellen, steht die folgende Anweisung zur Verfügung: (Default: der zuletzt benannte Record)

```
WITH  recordname  DO
      {statement}
ENDWITH
```

Bedeutung:

Durch diese Anweisung wird ein Block definiert, in dem die Zuordnung von Daten zu Anweisungen einheitlich definiert ist.

- recordname: Bezeichnung eines Records;
  i.a. eine durch die Datenauswahl erzeugte Menge von Tupeln
- statement: alle Anweisungen zur Report-Aufbereitung, die sich auf Attribute aus dem Record recordname beziehen.

Anwendung:

Falls bei der Datenauswahl mehr als ein Record erzeugt wurde und sich eine Anweisungsfolge nicht auf den bis dahin zuletzt benannten Record bezieht, dann muß der Bezug explizit mit obiger Anweisung hergestellt werden (auch für Datenverknüpfung).

Realisierung:

(a) Prüfung, ob recordname bereits deklariert
(wenn nicht: Fehler).

(b) Zuordnung des Records herstellen.

(c) Falls keine WITH ...-Anweisung im RDL-Programm, d.h. einer Anweisung zur Datenauswahl folgt eine Anweisung ≠ WITH ...
(und ≠ RETRIEVE, JOIN, DECODE), dann: Zuordnung des zuletzt deklarierten Records.

### 4.6.1 Definition des Report-Rahmens

a) Kopf

Beschreibung des Report- bzw. Seiten-Kopfes anhand vorgegebener Optionen und/oder beliebiger Anweisungen. Pro Report oder Seite ist eine Kopfdefinition zugelassen.

```
{HEAD  OF   REPORT
            PAGE     [reportname [format]]

        [PAGENO [startno][format]]
        [COLUMNS |{columnheader [format]}|[LINES]]
                 | ALL       {[format]}  |

         [{statement}]}

ENDHEAD
```

Bedeutung:

- Die zum Seiten-Kopf (PAGE) angegebenen Optionen schließen den Report-Kopf (REPORT) ein. Anweisungen und Optionen zum Report-Kopf beziehen sich nur auf die erste Seite des Reports. ENDHEAD schließt die Kopf-Definition insgesamt ab (vgl. Syntax).
- Optionen:
  - reportname: Report-Name; beliebiger Text (max. 1 Zeile) als Überschrift. Default: report-programmname (siehe Deklaration 4.2)
  - format: legt printposition und formatcode der Optionen fest, falls abweichend vom Standardformat gewünscht (siehe 4.6.3)
  - PAGENO: Seitennummer; wird automatisch hochgezählt. Durch startno kann die Nummer der ersten Seite auf einen bel. Wert gesetzt werden (Default: 1)

    Default für PAGENO: keine Seitennummerierung

. COLUMNS: (Nur bei spaltenweiser Ausgabe und Report-Typ ALPHA).

Festlegung der Spaltenüberschrift columnheader (bel. Zeichenfolge). Mit der Format-Angabe wird die Spaltenbreite festgelegt. Falls keine FORMAT-Angabe, dann ist die Spaltenlänge gleich der Länge der Spaltenüberschrift. Wenn die Attribut- bzw. Variablennamen des Ausgabe-Records als Spaltenüberschrift dienen sollen, genügt die Angabe von ALL.

Die Option LINES erzeugt einen waagerechten Trennungsstrich nach der Überschrift und senkrechte Trennungsstriche zwischen den Spalten.

. statement: Anweisungen zur weiteren Beschreibung des Kopfes.

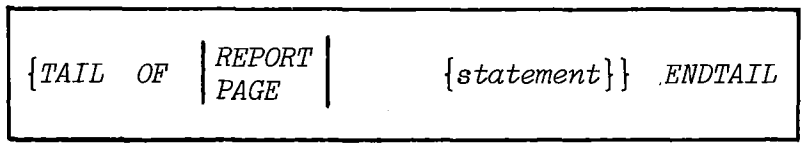
Durch die Definition von Spalten wird eine Spalteneinteilung festgelegt. Diese Einteilung bestimmt die Positionierung der Attribut- und Variablenwerte und der Globalwerte in den entsprechenden Ausgabezeilen (falls keine abweichende Positionierung verlangt).

Realisierung:

- Ausgabe des Report-Namens (Überschrift) aufbereiten
- falls Seiten-Numerierung: Seiten-Nr. initialisieren (nur zu Beginn der Bearbeitung)
- falls Spaltendefinition:
  . Spalteneinteilung bestimmen gemäß Formaten (Länge) der Überschriften. Falls keine Formate: gleichmäßige Aufteilung der Zeile.
  . Spaltenüberschriften aufbereiten
  . ggf. Trennungslinien einbauen (waagerechter Strich nach Spaltenüberschrift; senkrechte Striche auf Spaltengrenze).
- Ergebnisvariable für Globalwerte auf Basiswerte setzen (vgl. 4.5.2(c)).

b) Abschluß

Beschreibung des Report- bzw. Seiten-Abschlusses anhand beliebiger Anweisungen. Pro Report oder Seite ist eine Abschlußdefinition zugelassen.




ENDTAIL schließt die Abschlußdefinition insgesamt ab; steht also genau ein Mal im Programm, falls Abschlußdefinition erfolgt.

Als Anweisungen werden insbesondere solche zur globalen Datenverknüpfung (4.5.2) und Formatierung angewandt. Die Globalwerte werden automatisch gemäß Spalteneinteilung (durch HEAD ... COLUMNS oder CREATE record ...) in die Mitte der entsprechenden Spalte positioniert (falls nicht explizit abweichend definiert).

Realisierung:

- Übersetzer muß Länge des Abschlusses (Zeilenzahl) feststellen;
- falls gegeben: Ausgabe der Globalwerte aufbereiten;
  Positionierung gemäß Spalteneinteilung (falls nicht abweichend definiert);
- ggf. weitere Anweisungen;
- bei PAGE:
  - . Seitenvorschub
  - . Zeilenzähler zurücksetzen
  - . Seitenzähler erhöhen.

### 4.6.2 Definition des Report-Rumpfes

a) Abgrenzung des Programmabschnittes

Der Programmabschnitt zur Definition des Report-Rumpfes wird implizit abgegrenzt durch:

- ENDHEAD und TAIL OF ..., falls Kopf und Abschluß von Report oder Seite definiert.
- Datenauswahl und Reportausgabe, falls kein Kopf und Abschluß definiert; alle Anweisungen, die auf die Datenauswahl folgen, beziehen sich dann auf den Report-Rumpf (bis zur Ausgabe-Anweisung PUT ...).
- Datenauswahl und nächsten WITH-Block.

b) Definition des Ausgabeobjektes

Abhängig vom Report-Typ wird das Ausgabeobjekt definiert als Record, Row (alphanum. Ausgabe) oder Picture (grafische Ausgabe). Die CREATE-Anweisung dient zur Festlegung des entsprechenden Objekts.

Die Anweisung hat folgende allgemeine Form:

```
CREATE      <Ausgabeobjekt>      OF
            <Objektspezifikation>
ENDCREATE
```

b1) Zur Definition des Ausgabe-Datensatzes (Records) wird die folgende Version der Anweisung angewandt:

```
CREATE    recordname    OF
                  | attributename |
                  | variablename  |   [detailgrade]  [format]
ENDCREATE
```

Bedeutung:

Festlegung von Inhalt und Format eines Ausgabe-Datensatzes.

Die Anweisung CREATE ist im Report-Programm erst dann möglich, wenn alle im Ausgabedatensatz enthaltenen Variablen deklariert sind (durch Wertzuweisung).

- detailgrade:

  Zu jedem Objekt (Attribut, Variable) kann eine Detaillierungsstufe (Di; i=1, 2, 3, 4) angegeben werden. Diese steuert die Datenextraktion aus dem Record bei Report-Erzeugung:

  > Bestimmung eines DK bei Report-Erzeugung bedeutet:
  > alle Spalten mit Di (i ≤ K) erscheinen im
  > Report, während die Spalten mit Di (i > K)
  > entfallen.

  Default: D4 (bei Report-Def. und -Erzeugung).

- format:

  Ausgabeformat des Tupels; bei Report-Typ ALPHA: Positionierung der Ausgabeobjekte gemäß Spalteneinteilung (falls definiert) auf die Spaltenmitte. Abweichende Angaben in CREATE-Anweisung werden überlesen.

Falls im Report-Programm keine CREATE-Anweisung erfolgt, wird der implizit oder explizit (WITH ...) zugewiesene Record aus der Datenauswahl als Ausgabe-Datensatz angenommen, d.h., es werden nur Attributwerte ausgegeben (im Attributtyp-abhängigen Standardformat).

Pro WITH ... ENDWITH-Block ist max. eine CREATE record ...-Anweisung zugelassen.

Realisierung:

- Deklaration des Records recordname, bestehend aus den angegebenen Attributen und Variablen und deren Typen. Fehler falls:
  - . Record schon deklariert oder
  - . Attribut oder Variable nicht deklariert.

- Formatierung
  - . falls Spaltendefinition (vgl. 4.6.1a)) vorliegt: Anpassung der Formate an die Spaltengröße (auch bei Standardformaten per Default), so daß:
    - . Länge der Ausgabevariablen ≤ Spaltenlänge
    - . Beginn der Ausgabevariablen in der Zeile nicht vor Spaltenbeginn
    - . Positionierung auf die Spaltenmitte

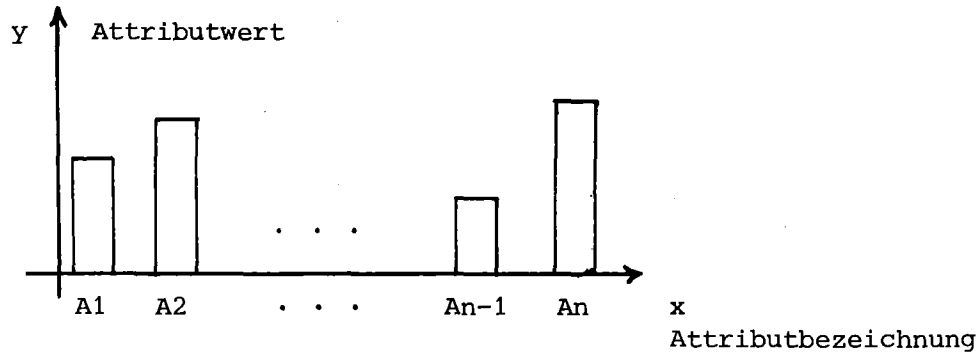
    Spaltendefinition geht vor Formatangabe!
  - . falls keine Spalten definiert: Spaltendefinition festlegen aus den impliziten oder expliziten Einzelformaten der Ausgabevariablen.

- ggf. Spaltendefinition erweitern um Detaillierungsgrad, d.h. pro Spalte Bestimmung des zugehörigen Detaillierungsgrades.

b2) Bei graphischer Ausgabe wird als Ausgabeobjekt das Bild definiert.

Ein Bild besteht aus:

- einem oder mehreren Achsenkreuzen mit jeweils einer oder mehreren Kurven ('Graphen') oder
- einem Achsenkreuz mit Balkendarstellung ('Bargraph') mehrerer Attribute (Ai) auf der x-Achse: (Voraussetzung: Ai alle vom selben Typ!).



Anweisung für **Kurvendarstellung**:

```
CREATE  picturename  OF
{GRAPH | attributename | IN domain SCRIBE 'constant' ON XAXIS [AS | DATE | ]
                       | variablename |                          | TIME |
                                                                 | DEC  |
       {{| attributen. | IN domain SCRIBE 'constant' ON YAXIS [AS | DATE | ]}}
         | variablen.   |                                       | TIME |
                                                                 | DEC  |
ENDCREATE
```

Anweisung für **Balkendarstellung**:

```
CREATE  picturename  OF
     BARGRAPH {| attributename |} ON XAXIS
              | variablename   |
              domain SCRIBE 'constant' ON YAXIS [AS | DATE | ]
                                                | TIME |
                                                | DEC  |
ENDCREATE
```

Bedeutung:

- picturename: Bezeichnung des Bildes zur Identifikation bei der Ausgabe (PUT-Anweisung 4.3.2)
- GRAPH bzw. BARGRAPH definiert genau ein Achsenkreuz (x- und y-Achse) mit Kurven- bzw. Balkendarstellung (Darstellungsart).
- ... XAXIS bzw. ... YAXIS: Achsenzuordnung; angegebener Anweisungsteil gilt für x- bzw. y-Achse
- attribute-, variablename: Bezeichnung des Objekts, das auf der entsprechenden Achse erscheinen soll (nur numerische Typen möglich)
- domain: Wertebereich des Objekts auf der Achse; besteht aus

  . untere Grenze (ug)
  . obere Grenze (og) ; ug < og .

  Die Werte können konstant oder variabel (durch Angabe einer Report-Variablen) sein.
- SCRIBE 'constant': Achsenbeschriftung mit Zeichen-Kette 'constant'; i.a. Dimension. Der Objektname wird automatisch angegeben.
- DATE, TIME, DEC: Skalierungstyp; legt Skalierung (Datum, Uhrzeit, dezimal) auf der entsprechenden Achse fest. Default: dezimal.

Einschränkungen:

- Bei Balkendarstellung: pro Bild ein Achsenkreuz und pro Attribut ein Wert darstellbar
- Bei Kurvendarstellung: pro Bild maximal 4 Achsenkreuze (d.h. pro CREATE-Anweisung maximal 4 GRAPH ...Anweisungsteile). Anzahl y-Achsen und Anzahl Kurven abhängig von Anzahl Achsenkreuze:

| Anzahl Achsenkreuze/Bild | Anzahl y-Achsen/Achsenkreuz | Anzahl Kurven/Achsenkreuz |
|---|---|---|
| 1 | 2 | 4 |
| 2, 3, 4 | 1 | 1 |

- Eineindeutige Beziehung zwischen Achsenbeschriftung, Wertebereich und Achse.

Realisierung:

- Deklaration des Records picturename bestehend aus den Attributen und Variablen und deren Typen.
- Prüfung der Datenbeschreibungsparameter und Ablegen in die Datei 'Datenbeschreibung'.

  Datenbeschreibungsparameter sind: Darstellungsart, Zahl der Achsenkreuze, Attribut- bzw. Variablenname, Achsenzuordnung, Wertebereich, Achsenbeschriftung, Skalierungstyp.
- ggf. Ablegen des Report-Kopfes in die Datei 'Überschrift'.

b3) Zur Definition einer einzelnen Ausgabezeile, die unabhängig von Report-Records erstellt werden soll, dient die folgende Anweisung:

```
CREATE     ROW      OF
    {[variablename]  [format]}
ENDCREATE
```

Bedeutung:

Definition einer Ausgabezeile bestehend aus:

- Variablenwerten, die nicht in einem Record enthalten sind
- Konstanten, die in der format-Option definiert werden.

Anwendung:

Diese Anweisung dient im Gegensatz zu 'CREATE record' und 'CREATE picture'

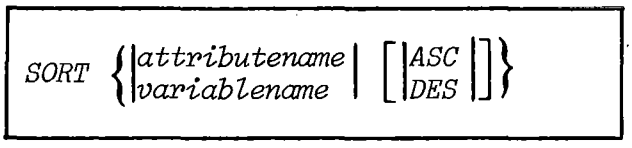
zur Ausgabe von Einzelwerten, die nicht in einem Report-Record enthalten sind, und steht insbesondere im Kopf und Abschluß von Report, Seite und Gruppe.

Realisierung:

- Erzeugung einer Ausgabezeile aus Variablenwerten und Formatbeschreibung.
- Falls Variable nicht deklariert: Fehler

c) Sortierung

Sortierung der Tupel eines Report-Records nach dem Sortierattribut in auf- oder absteigender Weise.




Bedeutung:

Sortiert werden die Tupel des Records, der im Report-Programm zuletzt vor der SORT-Anweisung deklariert wurde. Das ist entweder ein Record aus der Datenauswahl oder ein durch CREATE ... erzeugter Ausgaberecord. Die Position der SORT-Anweisung im RDL-Programm stellt den Bezug her.

- attributename, variablename:
  Sortierattribut; Typ beliebig; mehrere Sortierattribute nacheinander im Ausgabe-Record möglich
- ASC: aufsteigende Sortierreihenfolge
- DES: absteigende Sortierreihenfolge.

Realisierung:

pro Sortierattribut:

- falls Record aus Datenauswahl: Sortieraufruf an Datenbanksystem
- falls Ausgaberecord: Definition einer Zwischendatei und Record in Zwischendatei ablegen. Dann Sortieraufruf an Datenbank.

d) Gruppierung

Einteilung der Ausgabedatensätze in Gruppen mit gleichen Werten im Gruppenattribut. Eine Sortierung nach dem Gruppenattribut ist automatisch (aufsteigend) eingeschlossen. Gruppierung nur bei Report-Typ ALPHA.

Unterschiede zur Sortierung:

- Gruppe ist als zusammengesetztes Objekt definiert und damit sind gruppenbezogene Anweisungen am Anfang und Ende der Gruppe möglich, z.B.: Berechnung gruppenbezogener Globalwerte
- Wert des Gruppenattributes wird nur in der ersten Zeile jeder Gruppe ausgegeben und sonst unterdrückt.
- Gruppierung bezieht sich immer auf einen Ausgaberecord (Record entweder durch CREATE... explizit oder durch Datenauswahl implizit als Ausgaberecord definiert; vgl. 4.6.2 b1)).

```
GROUP  |attributename|
       |variablename |

       [[HEAD   {statement}]
        [TAIL   {statement}]   ENDGROUP]
```

Bedeutung:

- attributename, variablename:
  Gruppenattribut; Tupel des Ausgabe-Records mit demselben Wert im Gruppenattribut bilden eine Gruppe. Eine Gruppenschachtelung (mehrere GROUP-Anweisungen nacheinander) ergibt sich aus der Reihenfolge der Gruppenattribute im Ausgabe-Record.

- HEAD: Kennzeichnung der Anweisungsfolge, die zu Beginn jeder Gruppe (mit entspr. Gruppenattribut) ausgeführt werden soll.

- TAIL: Kennzeichnung der Anweisungsfolge, die am Ende jeder Gruppe (mit entspr. Gruppenattribut) ausgeführt werden soll.

Realisierung:

- Definition des Objekts Gruppe mit dem angegebenen Gruppenattribut zum aktuellen Ausgaberecord.
- Deklaration einer internen Vergleichsvariablen für das Gruppenattribut (mit entsprechendem Typ).
- Gruppenschachtelung gemäß Reihenfolge der Gruppenattribute im Ausgabe-Record.
- Länge (Zeilenzahl) Gruppenkopf und -abschluß feststellen; Kopf bzw. Abschluß immer auf eine Seite.
- Für jedes Tupel des Ausgaberecords
  . prüfen auf Gruppenwechsel, d.h.:
  Wert Vergleichsvariable = Gruppenattributwert.

  bei Gruppenwechsel:
  Gruppenabschluß- und -Kopf-Anweisungen durchführen.

falls kein Gruppenwechsel:

Gruppenattributwert nicht ausgeben.

. Vergleichsvariable mit Gruppenattributwert besetzen.

4.6.3 Formatierung

(nur bei Report-Typ ALPHA)

a) Format der Ausgabeelemente (format)

Festlegung von Druckposition und Ausgabeformat für Attribute und Variable durch:

| *FORMAT* [*printposition*], [*formatcode*] [*NEWROW*[*n*]] [*NEWPAGE*[*n*]] |
|---|

Format-Anweisung ist nur innerhalb anderer Anweisungen anzuwenden.

Bedeutung:

- printposition: Stelle in der Ausgabezeile, an der das Ausgabeelement beginnen soll. Dabei bedeutet:

  n $\hat{=}$ absolute Stelle, d.h. n Stellen vom Beginn der Zeile an (1≤n≤Zeilenlänge).

  $\pm$n $\hat{=}$ relative Stelle, d.h. um $\pm$ n Stellen gegenüber aktueller Stelle verschoben.

- formatcode: Format von Attribut- oder Variablenwerten wird durch folgende Symbole gekennzeichnet; die Formatcodes sind jeweils typabhängig:

  . I n : Ziffernfolge der Länge n (Integer)

. R n.m : Real-Zahl mit n Stellen vor und m Stellen nach Dezimalpunkt; führende Nullen werden unterdrückt

. E n.m : Real-Zahl in Exponenten-Schreibweise; n Stellen vor und m Stellen nach Dezimalpunkt. positiver und neg. Exponent möglich

. A n : Zeichenkette der Länge n

Zur Darstellung von Konstanten sind die folgenden Codes vorgesehen:

. X n : n Leerzeichen

. '...' : Konstante (string)

Editiermasken bestehen aus entweder I- oder R- oder E- oder A-Codes, die mit X oder '...' gemischt sind. Damit läßt sich z.B. die Integerzahl 25627 mit der Maske I3'DM'I2'PF' ausgeben als: 256DM27PF. Die Formatcodes einer Editiermaske beziehen sich auf ein Ausgabeobjekt.

- Standardformate (Defaultwerte):

. Formatcodes sind abhängig vom Typ des Ausgabeobjekts:

| Objekttyp | Standardformat |
|---|---|
| Integer | I5 |
| Real | E1.4 |
| String | Al (l = Länge) |

. Positionierung ist abhängig von der betreffenden Anweisung:

bei Report-Record-bezogenen Anweisungen richtet sich die Positionierung ggf. gemäß definierter Spalteneinteilung (vgl. 4.6.1, 4.6.2b1));

bei anderen Anweisungen erfolgt die Positionierung anschließend an letztes Ausgabeobjekt.

- NEWROW, NEWPAGE: siehe b)

Realisierung:

Formatabhängige Umwandlung der numerischen Ausgabeobjekte in Zeichenketten.

b) Zeilen- und Seitensteuerung

| *NEWROW[n]* | *(n = 1, 2, 3, ...)* |
|---|---|

Bedeutung:

- Beginn einer neuen Zeile (Zeilenvorschub)
- bei n>1: Einfügen von n-1 Leerzeilen
- Default: n=1

Zeilenvorschub implizit wenn: Zeilenlänge = log. Seitenbreite

| *NEWPAGE[n]* | *(n = 1, 2, 3, ...)* |
|---|---|

Bedeutung:

- Beginn einer neuen logischen Seite (Seitenvorschub)
- bei n>1: Einfügen von n-1 Leerseiten
- Default: n=1

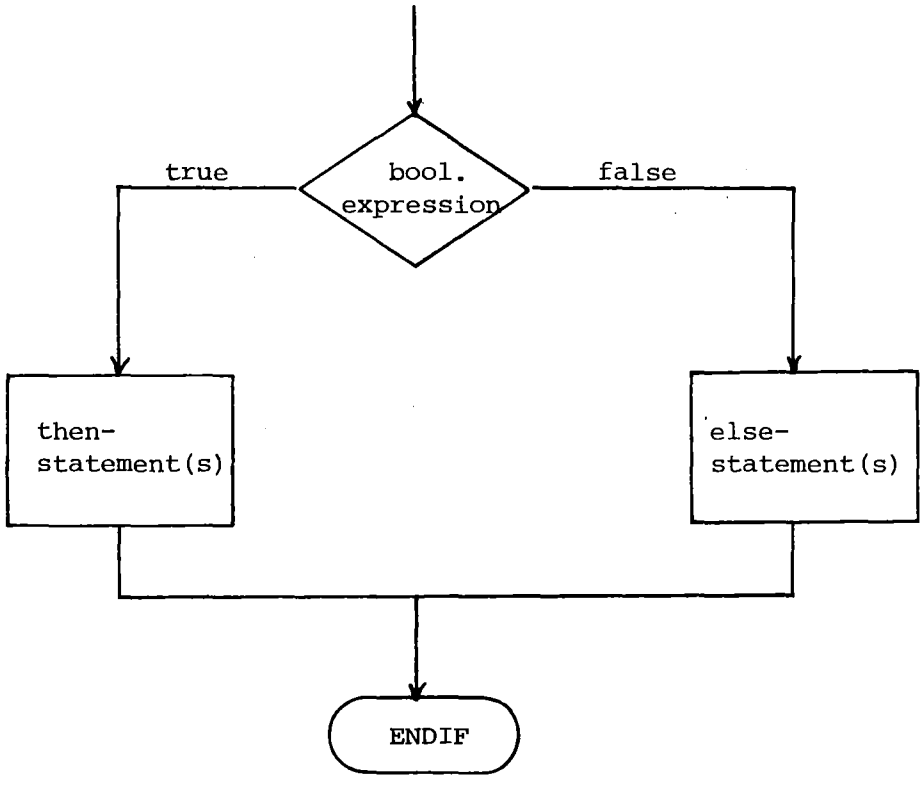
Seitenvorschub implizit wenn: Zahl der Ausgabezeilen = Zahl der Zeilen einer log. Seite (Seitenlänge).

## 4.7 Anweisungen zur Steuerung des Programmablaufes

### 4.7.1 Verzweigung

```
IF   bool.expression   THEN  {thenstatement}
                       [ELSE {elsestatement}]
                       ENDIF
```

Bedeutung:



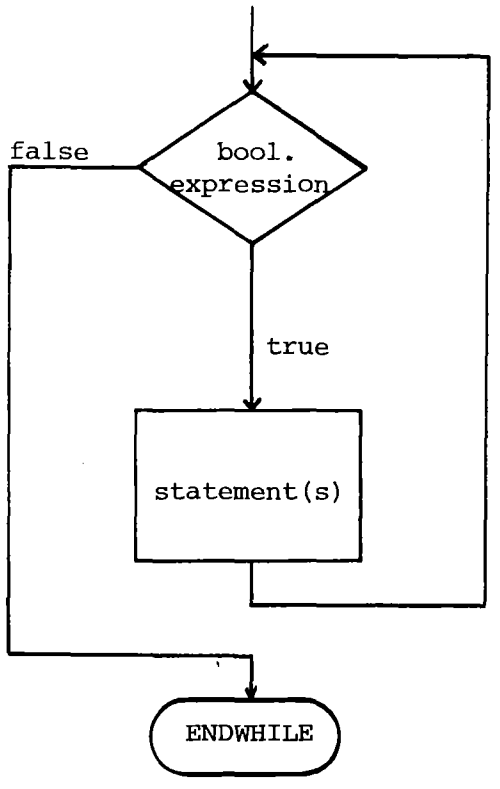

- bool.expression: logischer Ausdruck zusammengesetzt aus Vergleichsoperatoren (=, ≤, <, ≥, >, ≠) und logischen Operatoren (OR, AND) einschl. Klammerung. Als Operanden sind möglich: Attribute, Variable, Konstanten.

- Schachtelung mehrerer IF-Anweisungen möglich.

### 4.7.2 Iteration

*WHILE bool.expression DO {statement} ENDWHILE*

Bedeutung:



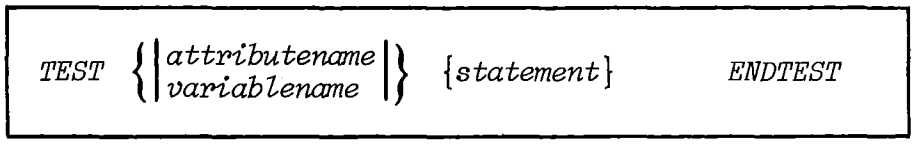

- Schachtelung von 'WHILE'-Anweisungen möglich.

Die durch diese Anweisung definierte Schleife bezieht sich auf eine Ausprägung eines Report-Records (Tupel). Die gesamten Schleifendurchläufe beziehen sich auf ein einziges Tupel.

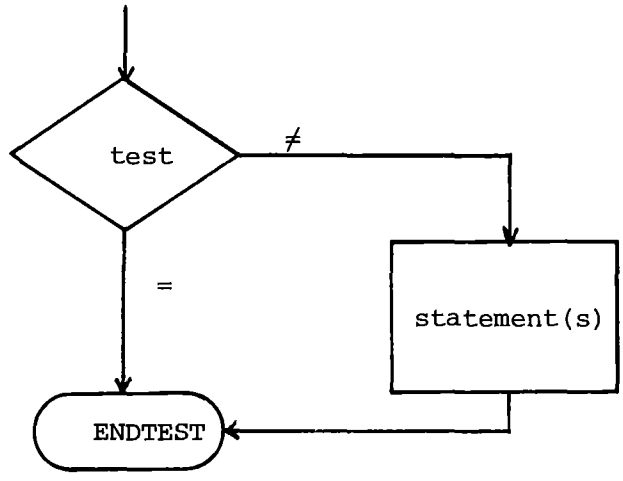
Die iterative Abarbeitung der Anweisungen im Report-Rumpf für die Gesamtheit der Records erfolgt implizit und kann nicht durch Anweisungen beeinflußt werden.

### 4.7.3 Änderungstest (changes-test /CAGE 76/)

Ausführung einer Anweisungsfolge, wenn sich der Wert einer Variablen oder eines Attributes aus dem aktuellen Tupel gegenüber dem vorhergehenden Tupel geändert hat.




Bedeutung:




- Vergleich der Attribut- bzw. Variablenwerte aus dem aktuellen Tupel mit den entsprechenden Werten aus dem vorhergehenden Tupel.
- Einzelvergleiche werden implizit mit ODER verknüpft. Bei Ungleichheit, d.h. mindestens ein Wertepaar ist ungleich, wird die zwischen 'TEST' und 'ENDTEST' eingeschlossene Anweisungsfolge ausgeführt.

Realisierung:

- zu jedem Vergleichswert (Attribut, Variable) Deklaration einer internen Variablen (typabhängig), die jeweils den entsprechenden Wert aus dem letzten Tupel enthält.
- falls Wert im aktuellen Tupel ≠ Wert Vergleichsvariable:
  - . Ausführung der Anweisungsfolge
  - . Vergleichsvariable erhält den aktuellen Wert.

### 4.7.4 Überspringen von Tupeln

| *SKIP[n]* | *(n = 1, 2, 3, ...)* |
|---|---|

Bedeutung:

Das aktuelle Tupel wird nicht weiter bearbeitet und nicht ausgegeben. Die n-1 folgenden Tupel werden ebenfalls von der Bearbeitung und Ausgabe ausgenommen.

Default: n=1

### 4.7.5 Programmende

| *END* |
|---|

Damit wird das statische Ende eines Programmes gekennzeichnet.

| *STOP* |
|---|

Diese Anweisung legt das dynamische Ende des Programmes fest.

### 4.7.6 Unterprogramm-Aufruf

Zum Aufruf benutzereigener FORTRAN-Unterprogramme steht folgende Anweisung zur Verfügung:

| *CALL subroutinename [(parameterlist)]* |
|---|

Es gelten die in FORTRAN üblichen Vereinbarungen.

## 5. Programmbeispiele

Die folgenden Beispiele sind nach Komplexität geordnet und beziehen sich auf folgende Relationen (Attribute):

plist (name, alter, gehalt)
tellist (name, titel, telnr)
raumlist (raumnr, name, raumgröße).

### 5.1 Liste alle Tupel der raumlist auf Schnelldrucker (LP).

RDL-Programm:

```
rtupel  =  RETRIEVE ALL FROM  raumlist; Datenauswahl

PUT     rtupel  ON  LP ;  Ausgabe
STOP
END
```

### 5.2 Liste raumnr, name, titel geordnet nach titel der Bewohner.

RDL-Programm:

```
rtupel  =  RETRIEVE  raumnr, name FROM raumlist; Datenauswahl

ttuple  =  RETRIEVE  name, titel FROM tellist

etuple  =  JOIN  rtuple WITH ttuple  IN name

SORT  titel                                   ; Report-Rumpf

PUT    etuple  ON  LP                         ; Ausgabe
STOP
END
```

5.3 a) Liste name und Verhältnis gehalt/alter aller Diplom-Informatiker (titel = dipl-inform), falls Verhältnis >100.

```
ptuple  =  RETRIEVE  name, alter, gehalt
                     FROM plist                    ; Datenauswahl
ttuple  =  RETRIEVE  name, titel  FROM  tellist
                     WHERE titel = dipl-inform
etuple  =  JOIN ttuple WITH ptuple IN name
                                                   ; Report-Rumpf
  verhaelt  =  gehalt/alter                        ; Berechnung Verhältnis
  IF  verhaelt  >100  THEN
    CREATE   outrecord  OF  name, verhaelt         ; Definition
    ENDCREATE                                      ;     Ausgabedatensatz
    ELSE SKIP
  ENDIF
PUT     outrecord  ON  LP
STOP ;  Programmende
END
```

5.3 b) zusätzlich: Spaltenüberschriften ('Mitarbeiter' und 'Gehaltsindex') und Berechnung der durchschnittlichen Verhältniszahl. (Ziffern wie in a))

```
:  1
HEAD  OF REPORT COLUMNS  'mitarbeiter',  'gehaltsindex'
ENDHEAD
:  2
TAIL   OF  REPORT
    AVG  verhaelt; Berechnung Durchschnittsverhaeltnis
ENDTAIL
 PUT ...
 :
```

5.4 Liste titel, name, alter, gehalt und jahreseinkommen mit folgenden Randbedingungen:

- Seitengröße 20x80
- Spaltenüberschriften (auf jeder Seite): 'akad. Grad', 'mitarbeiter', 'alter', 'gehalt', 'jahreseinkommen'. Spaltentrennung durch Linien
- Seitennumerierung
- Gruppierung nach titel; pro Gruppe: Max. Gehalt
- Berechnung des Jahreseinkommens für jeden Mitarbeiter
- Berechnung des Durchschnittsalters aller Mitarbeiter
- jede 5. Zeile als Leerzeile

```
PAGE  20,80                          ; Initialisierung  Seitengröße
I=0                                  ; Initialisierung  Zeilenzähler
ptuple  =  RETRIEVE ALL FROM plist   ; Datenauswahl
ttuple  =  RETRIEVE name, titel FROM tellist
etuple  =  JOIN ptuple WITH ttuple IN name
HEAD OF PAGE                         ; Seiten-Kopf
    PAGENO
    COLUMNS  'akad. Grad', 'mitarbeiter', 'alter', 'gehalt',
          'jahreseinkommen'  LINES;  Spaltenüberschrift
ENDHEAD
                                     ; Report-Rumpf
    I = I+1                          ; Zeilenzähler weiterschalten
    jahrein = gehalt * 13            ; Berechnung Jahreseinkommen
    IF I=5  THEN  NEWROW
            I=0   ENDIF              ; Leerzeile einfügen
    CREATE  outrec  OF  titel, name, alter, gehalt, jahresein-
                                kommen; Definition  Ausgabe-Record
    ENDCREATE
    GROUP  titel                     ; Gruppierung nach Titel
          TAIL MAX gehalt            ; Gruppen-Abschluß
    ENDGROUP
TAIL OF REPORT                       ; Report-Abschluß
        AVG alter                    ; Berechnung Durchschnittsalter
ENDTAIL
PUT   outrec  ON  LP                 ; Ausgabe auf Schnelldrucker
STOP
END
```

5.5 Grafische Ausgabe von alter (x-Achse)
und gehalt (y-Achse) der Dipl.-Informatiker.

```
ptuple  =  RETRIEVE  name, alter, gehalt
                  FROM plist
ttuple  =  RETRIEVE  name, titel  FROM  tellist
                  WHERE  titel = dipl-inform
etuple  =  JOIN  ttuple  WITH  ptuple  IN  name
               ; Auswahl der Dipl.-Informatiker

                                          ;  Report-Rumpf

  CREATE  picture  OF                     ;  Def. Bild
    GRAPH  alter  IN  20,65               ;  Wertebereich
             SCRIBE 'jahre'               ;  Achsenbeschriftung
             ON  XAXIS
           gehalt  IN  2,10               ;  Wertebereich gehalt
             SCRIBE 'tdm'                 ;  Achsenbeschriftung
             ON  YAXIS

  ENDCREATE

PUT  picture  ON  DP                      ;  Ausgabe auf Display
STOP
END
```

## 6. Tabellarische Übersicht der Anweisungen

Die folgenden Tabellen geben eine alphabetische Übersicht der Anweisungen, geordnet nach den wichtigsten Wortsymbolen der Anweisungen. Zu jeder Anweisung wird die mögliche Programmumgebung angegeben, d.h. in welchem Programmabschnitt (vgl. Bild 3-1) die Anweisung zugelassen ist. Die Bedeutung der Anweisung wird stichwortartig umrissen, und es folgt ein Verweis auf die ausführliche Beschreibung.

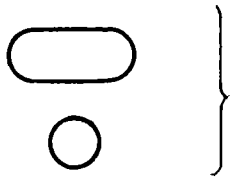
| Wortsymbol/ Anweisung | Programmabschnitt | Bedeutung | Beschreibung Kap. |
|---|---|---|---|
| AVG | Report-Aufbereitung (b, c) | Globalwert Durchschnitt | 4.5.2 |
| CALL | alle | Unterprogramm-Aufruf | 4.7.6 |
| COLUMNS | Report-Aufbereitung (a) | Definition Spaltenüberschrift | 4.6.1 |
| COUNT | Report-Aufbereitung (b, c) | Globalwert Anzahl | 4.5.2 |
| CREATE | Report-Aufbereitung | Definition Ausgabeobjekt | 4.6.2 |
| DECODE | Datenauswahl | Decodieren | 4.4.3 |
| END | - | statisches Programmende | 4.7.5 |
| format | Report-Aufbereitung | Formatierung | 4.6.3 |
| GRAPH | Report-Aufbereitung (b) | Definition Achsenkreuz | 4.6.2 |
| GROUP | Report-Aufbereitung (b) | Gruppierung von Tupeln | 4.6.2 |
| HEAD | Report-Aufbereitung (a, b) | Kennung Report-, Seiten-, Gruppen-Kopf | 4.6.1<br>4.6.2 |
| IF... THEN... | alle | bedingte Verzweigung | 4.7.1 |
| JOIN | Datenauswahl | Zusammenfügen von Records | 4.4.2 |
| LINE | Report-Aufbereitung | Trennlinien, Unterstreichen | 4.6.3 |
| MAX, MIN | Report-Aufbereitung (b, c) | Globalwert Maximum, Minimum | 4.5.2 |
| NEWPAGE | Report-Aufbereitung | Seitenvorschub | 4.6.3 |
| NEWROW | Report-Aufbereitung | Zeilenvorschub | 4.6.3 |
| PAGE | Initialisierung | Seitendeklaration | 4.2.2 |

| Wortsymbol/ Anweisung | Programmabschnitt | Bedeutung | Beschreibung Kap. |
|---|---|---|---|
| PAGENO | Report-Aufbereitung (a) | Definition Seiten-Nr. | 4.6.1 |
| PUT | Ausgabe | Report-Ausgabe | 4.3.2 |
| READ | Initialisierung | Einlesen | 4.3.1 |
| REPORT | Initialisierung | Report-Deklaration | 4.2.1 |
| RETRIEVE | Datenauswahl | Selektion und Extraktion | 4.4.1 |
| SKIP | Report-Aufbereitung (b) | Überspringen von Tupeln | 4.7.4 |
| SORT | Report-Aufbereitung (b) | Sortieren von Tupeln | 4.6.2. |
| STOP | - | dynamisches Ende | 4.7.5 |
| SUM | Report-Aufbereitung (b, c) | Globalwert Summe | 4.5.2 |
| TAIL | Report-Aufbereitung (b, c) | Kennung Report-, Seiten-, Gruppen-Abschluß | 4.6.1, 4.6.2 |
| TEST | Report-Aufbereitung (b) | Änderungstest | 4.7.3 |
| Wertzuweisung | alle | Wertzuweisung | 4.5.1 |
| WITH... DO | Report-Aufbereitung | Record-Zuordnung | 4.6 |
| WHILE | alle | Schleife | 4.7.2 |
| WRITE | alle | Ausgabe von Meldungen | 4.3.1 |

# 7. RDL-Syntax

## 7.1 Zeichenerklärung

### a) Symbole



Terminalsymbole; feste Symbole der Anweisung

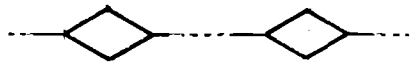
Terminalsymbol; Variable einfacher Symbole

Metavariable

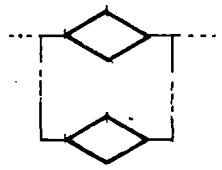
Beginn, Ende einer Anweisung

### b) Strukturen

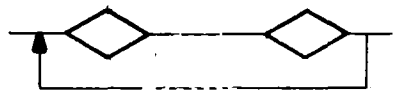
(◇= beliebiges Symbol oder Struktur)



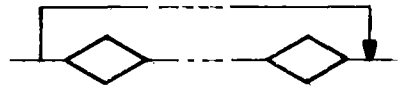
Sequenz



Alternative

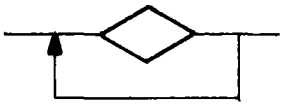


Iteration

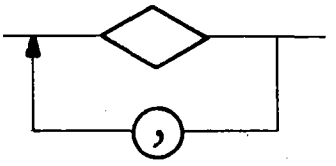


Optionalität

Für Iterationen gilt grundsätzlich:

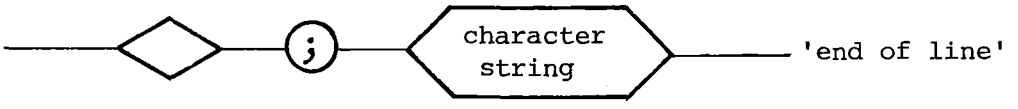


abgekürzte Schreibweise für:




d.h. die einzelnen Schleifen der Iteration werden durch Komma getrennt.

c) Kommentare



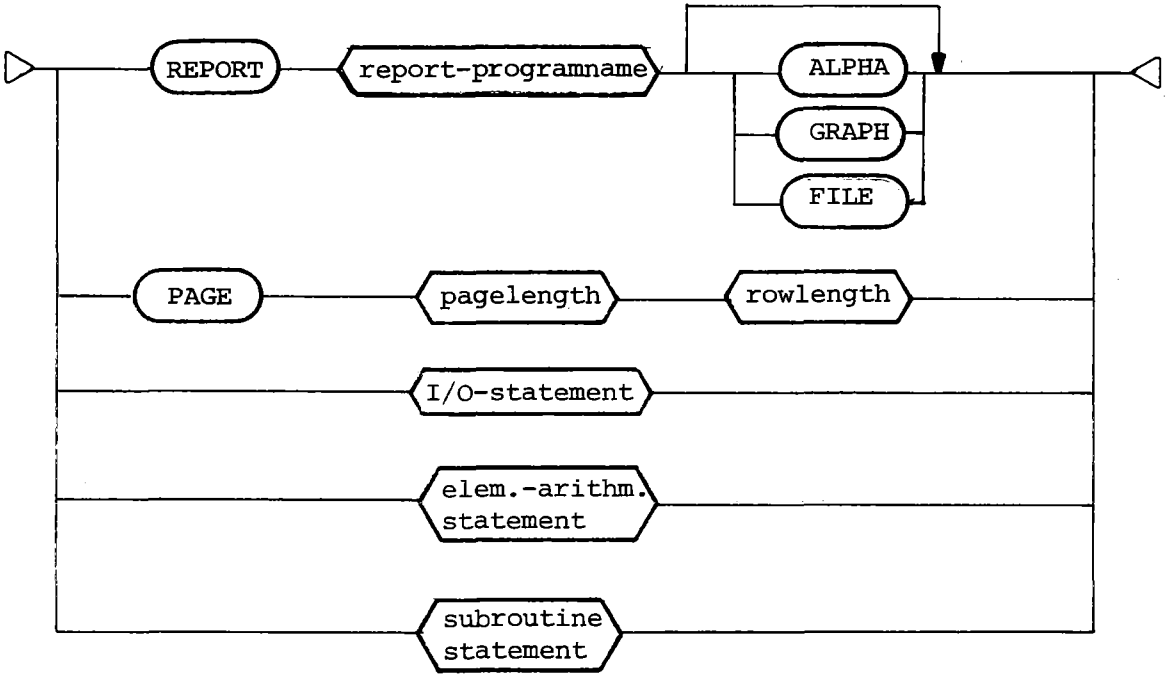

Eine Zeichenkette zwischen Strichpunkt und Zeilenende wird als Kommentar interpretiert und bei Übersetzung überlesen. Ein Kommentar kann innerhalb einer Anweisung stehen, die sich über mehr als eine Zeile erstreckt.

## 7.2 Syntax-Diagramme

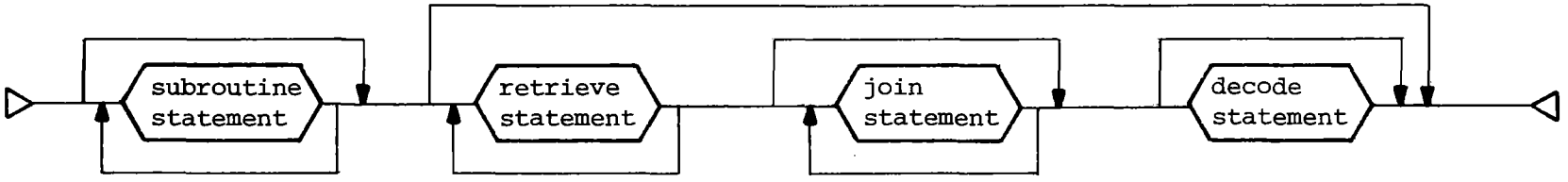
Die einzelnen Ableitungen sind durchnumeriert. Ein alphabetisches Verzeichnis (Index, 7.3) erleichtert das Auffinden.

1 rdl-program:

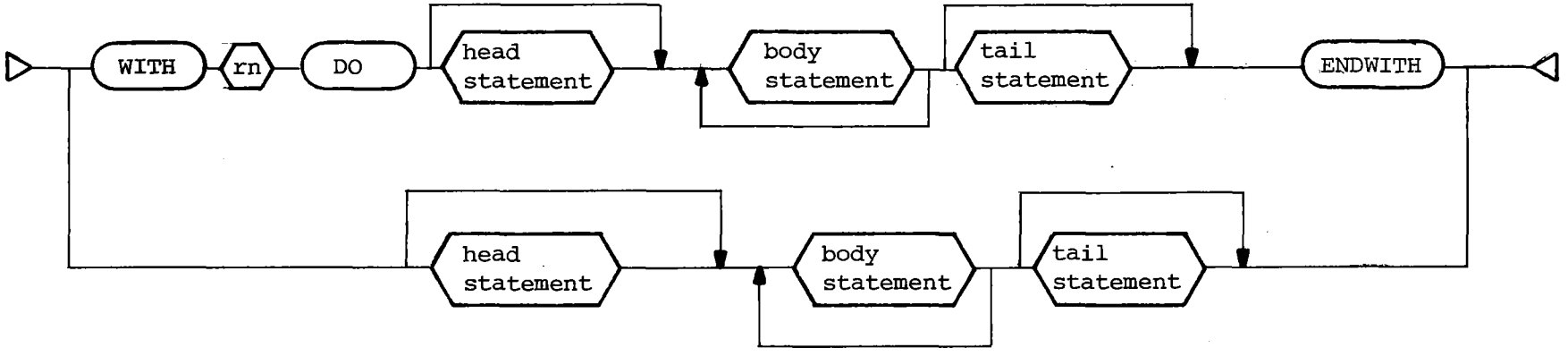

2 initialization statement:



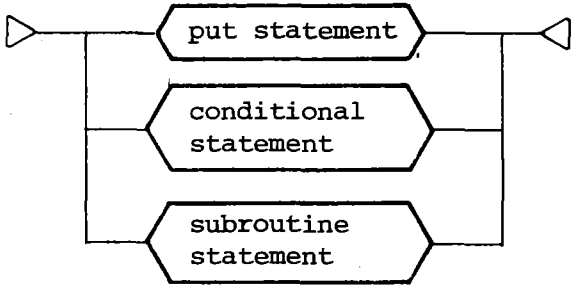
3 data-retrieval statement:



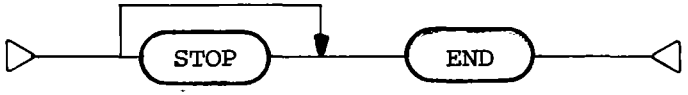

4 report-creation statement:



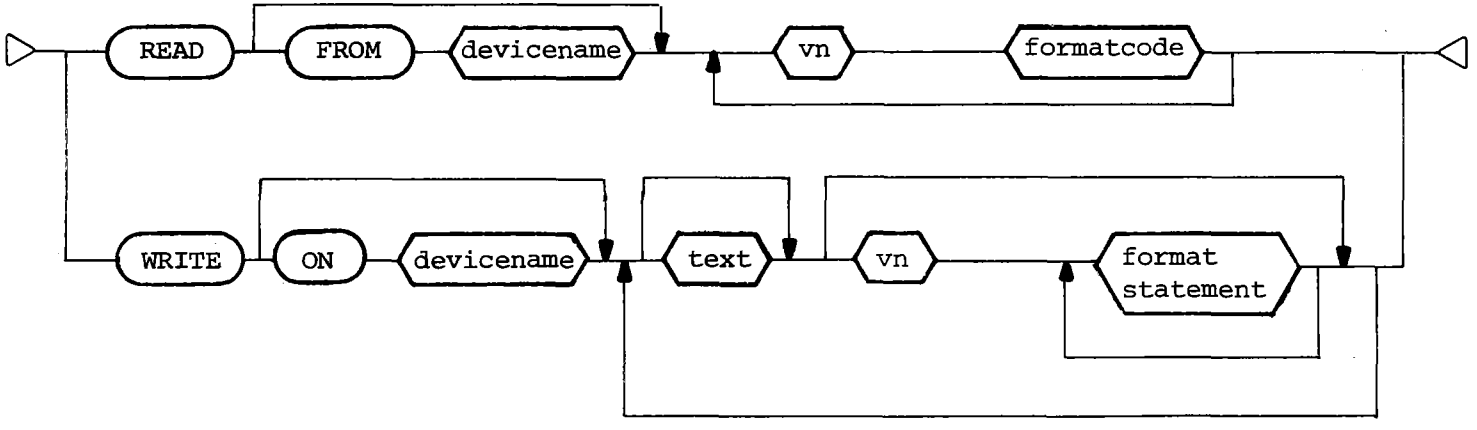
5 report-output statement:



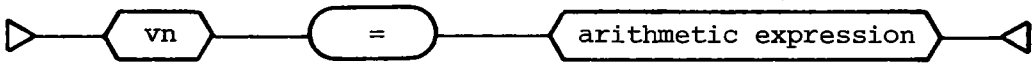

6 end statement:



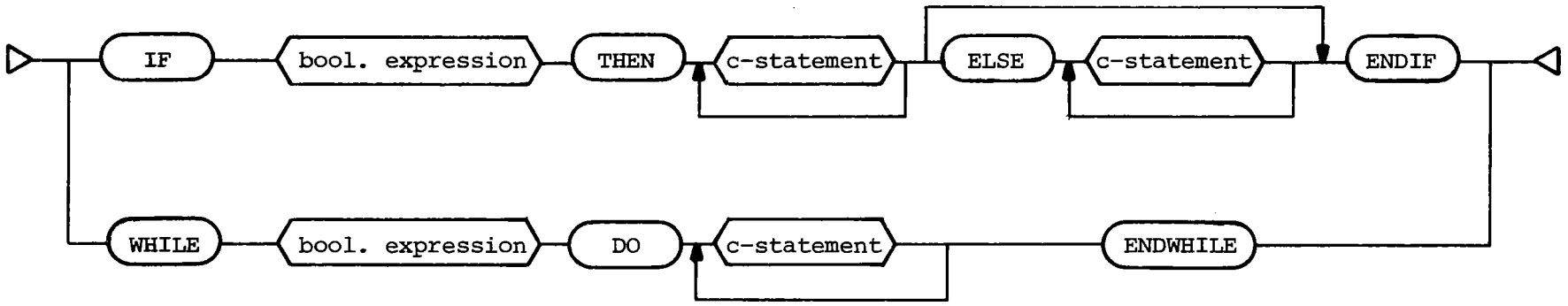

7 I/O-statement:



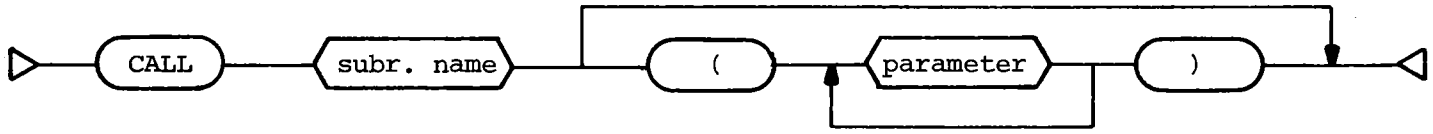
8 elem.-arithm. statement:



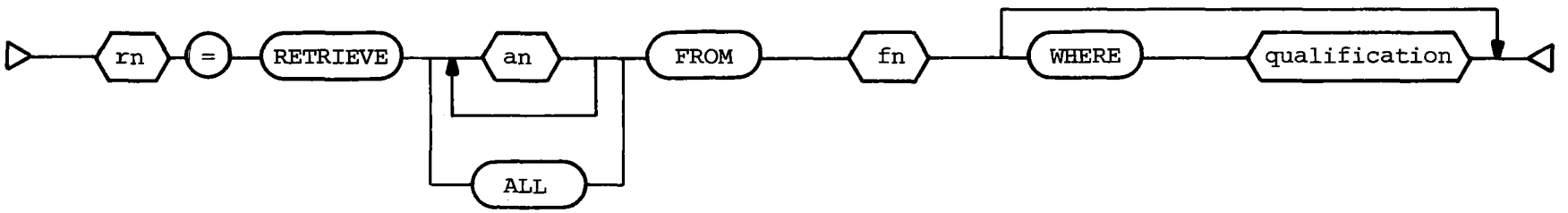

9 conditional statement:



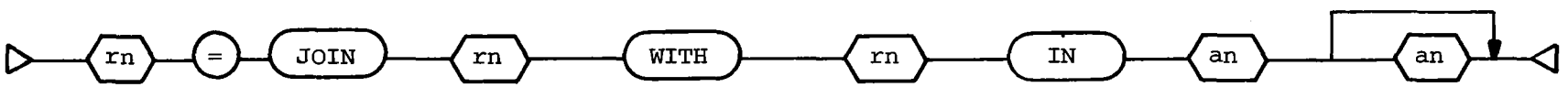

10 subroutine statement:



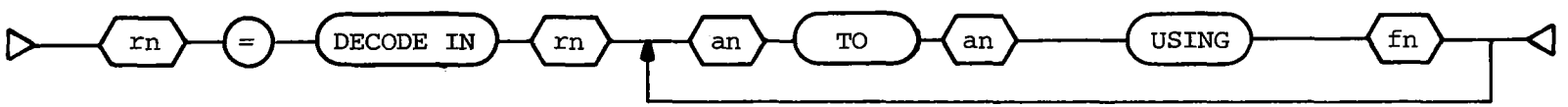
11 retrieve statement:



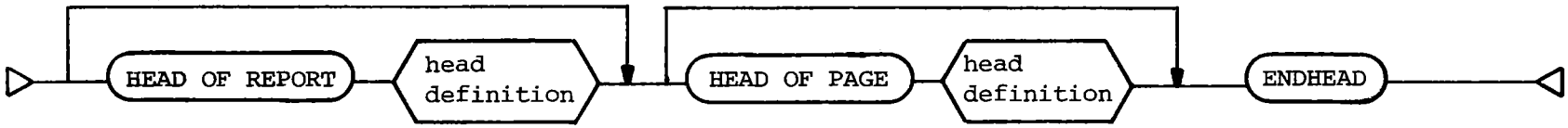

12 join statement:



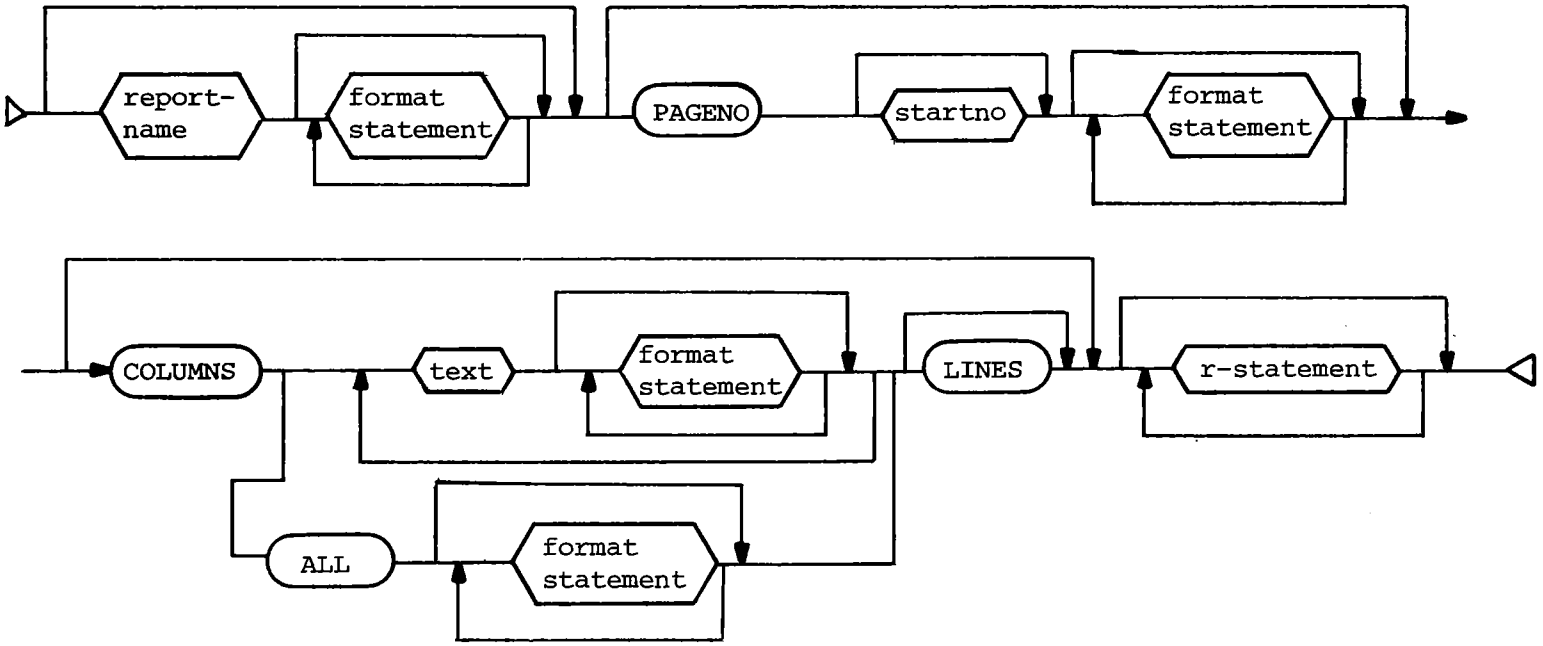

13 decode statement:



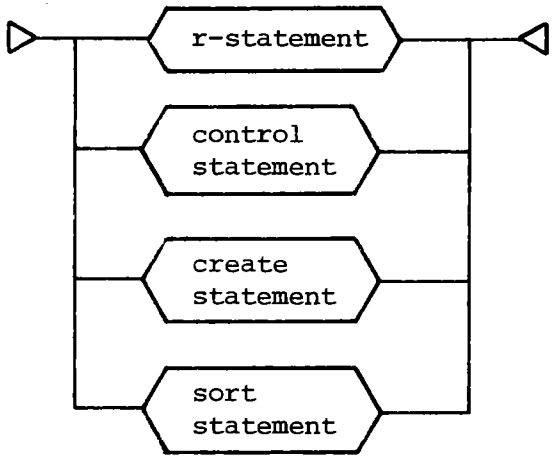
14 head statement:



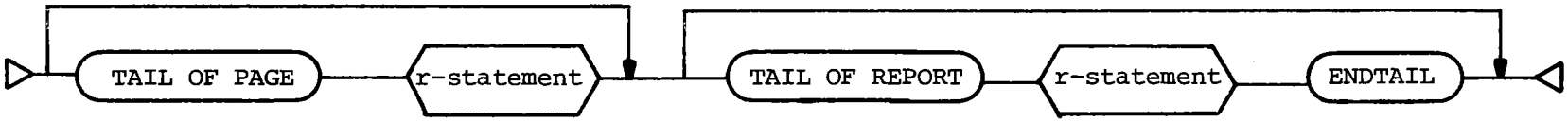

15 head definition:



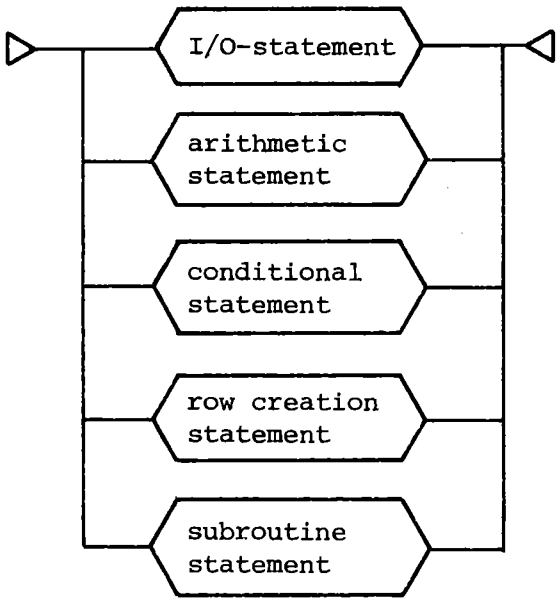
16 body statement:



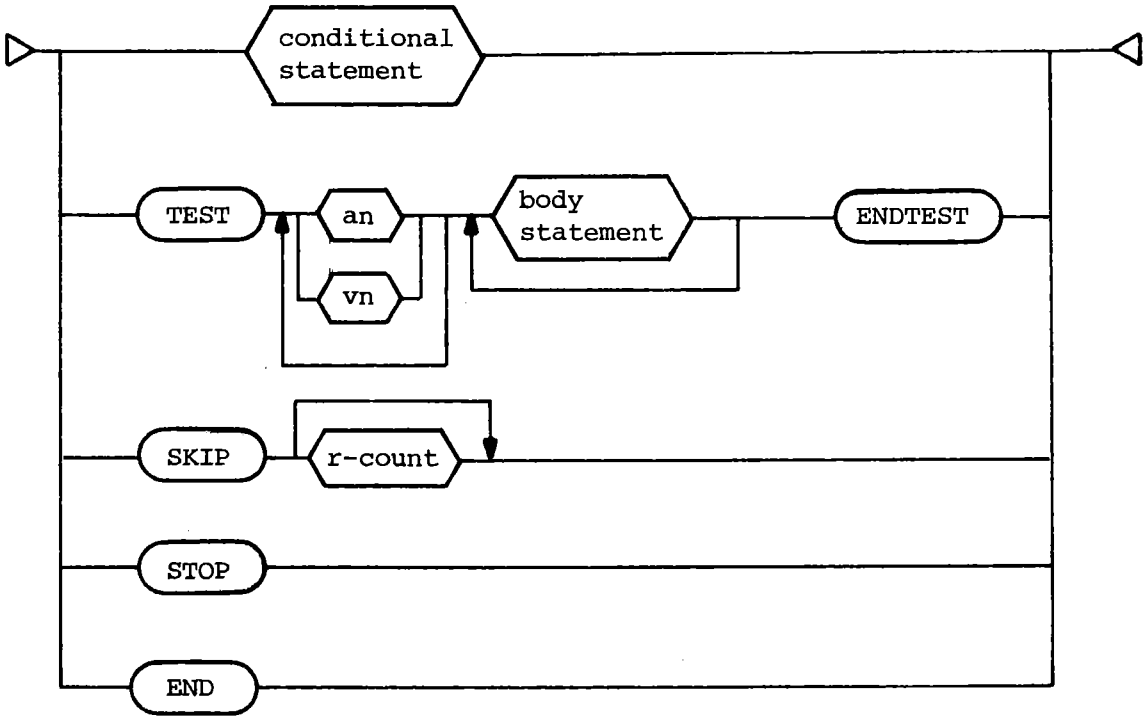

17 tail statement:



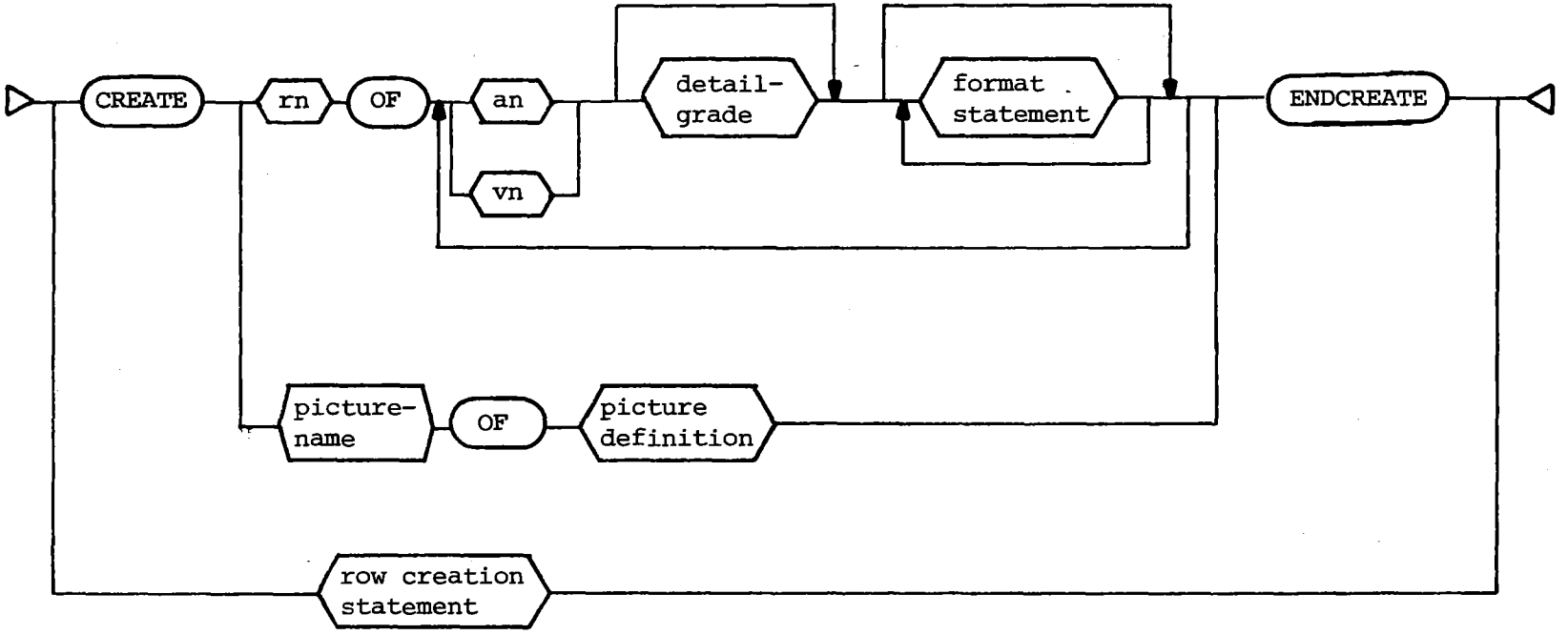
18 r-statement:



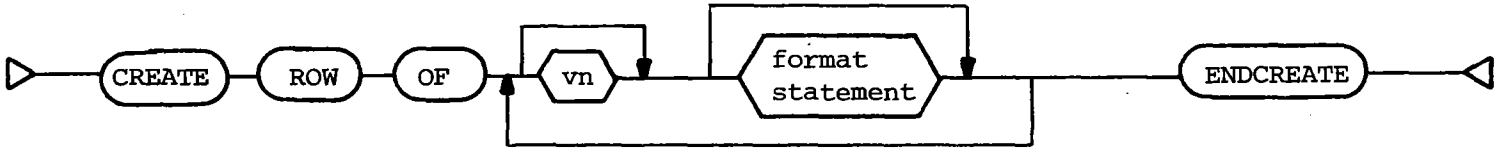

19 control statement:



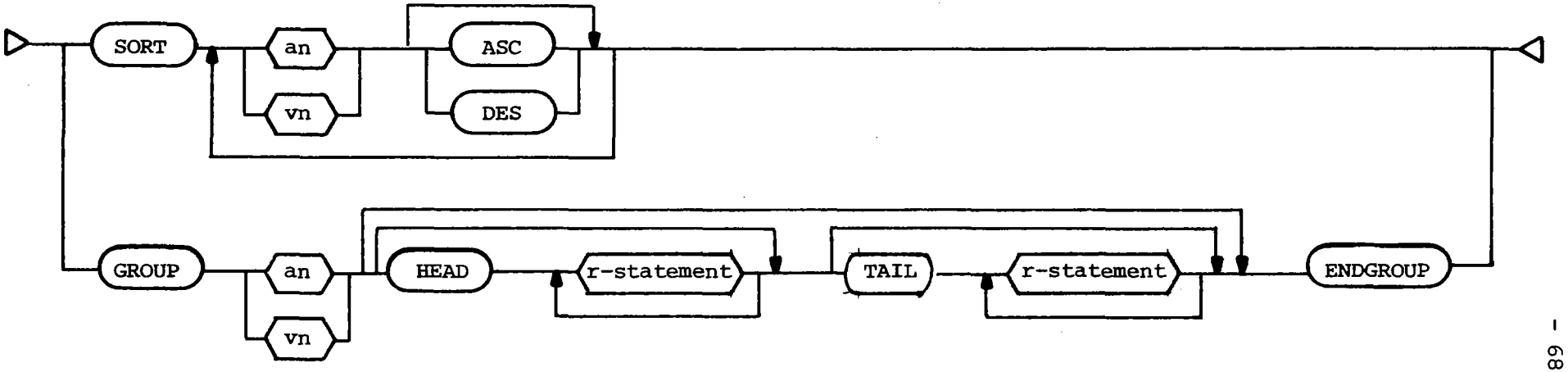
20 create statement:




20a row creation statement:

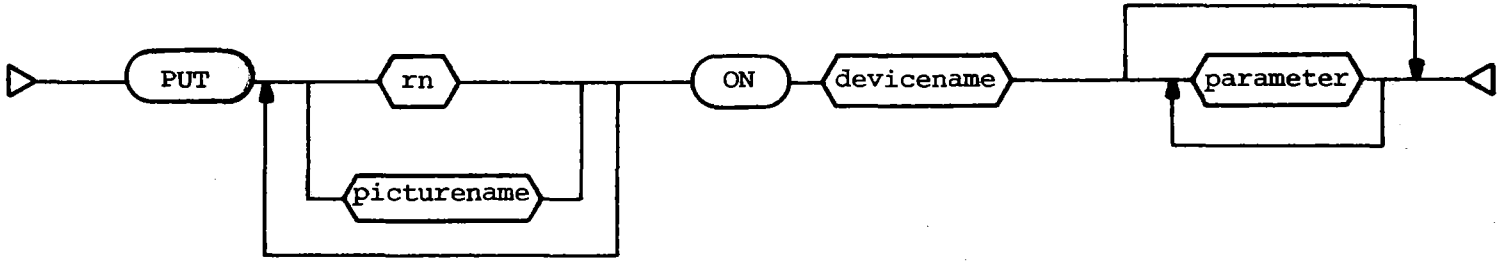


21 sort statement:

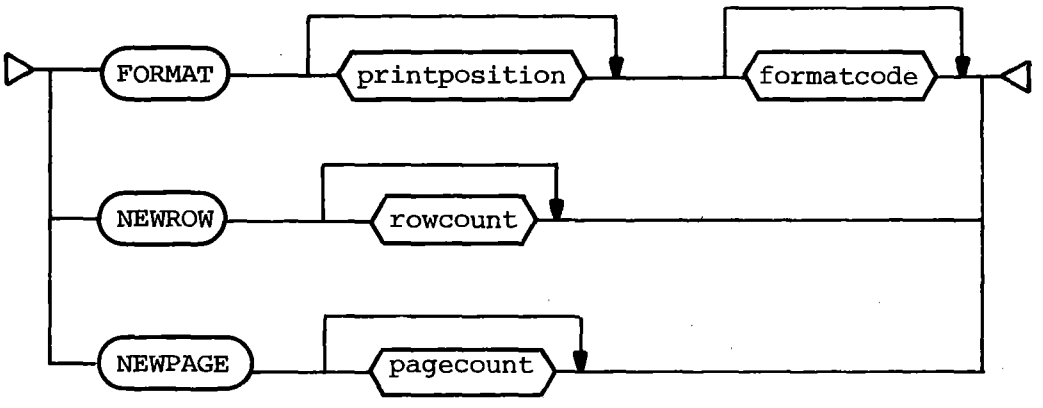




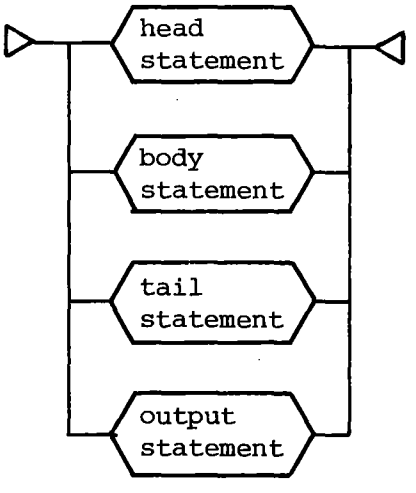

22 put statement:



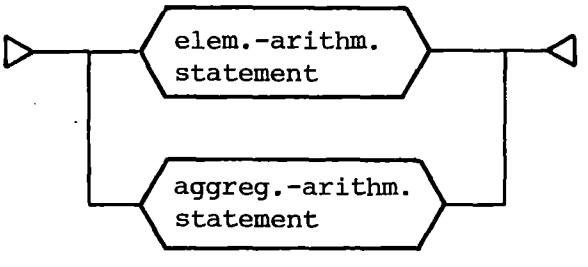
23 format statement:



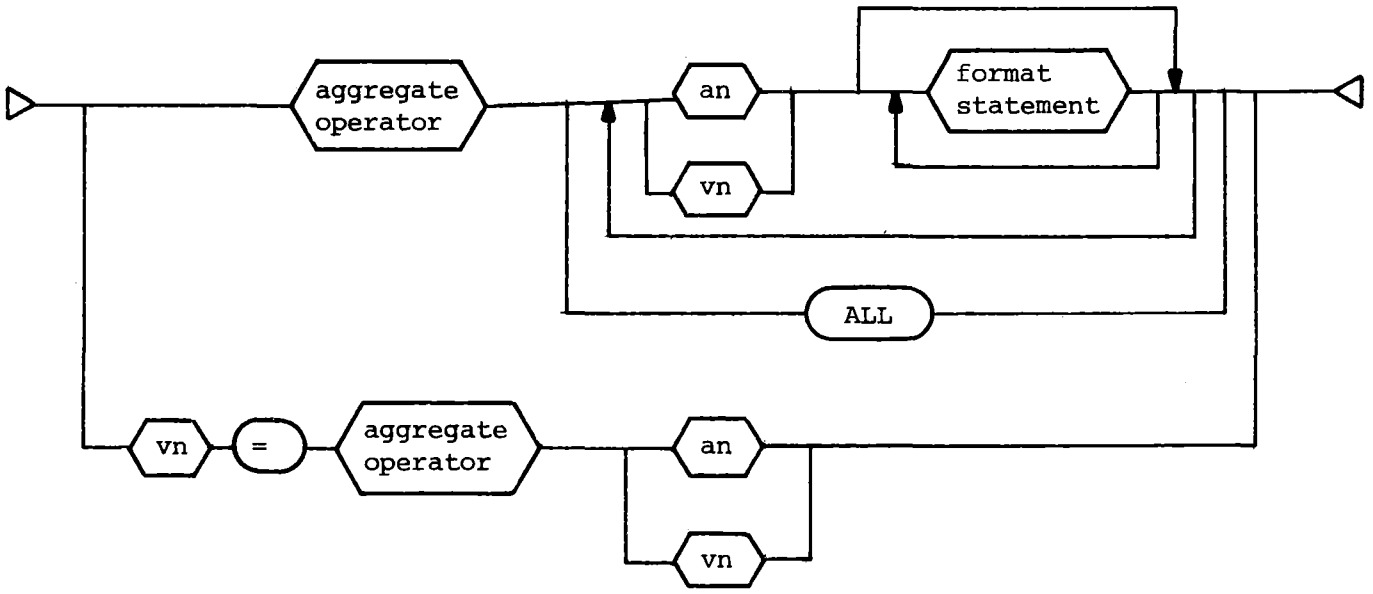

24 c-statement:



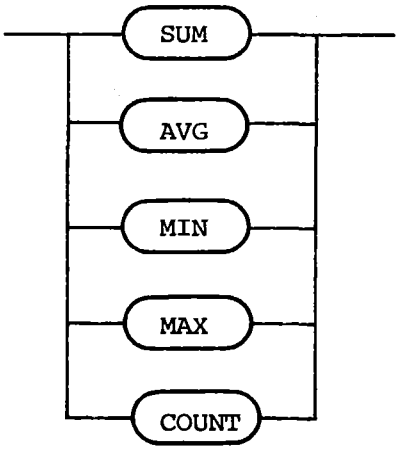

25 arithmetic statement:



26 aggreg.-arithm. statement:




26a aggregate operator




27 fn:

28 an:

29 vn:

31 function name:

32 rn:

33 reportprogramname:

34 subr.name:

35 picturename:


36 length:

37 pagelength:

38 rowlength:

39 startno:

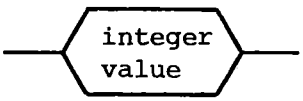
40 r-count:

41 rowcount:

42 pagecount:

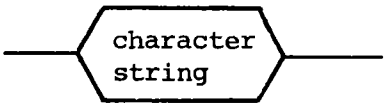
43 printbegin:

44 printend:

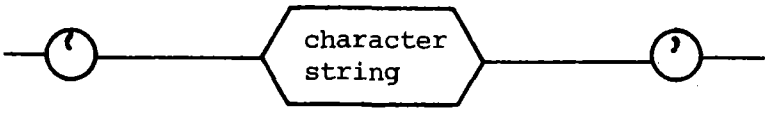



45 devicename:

46 report-name:

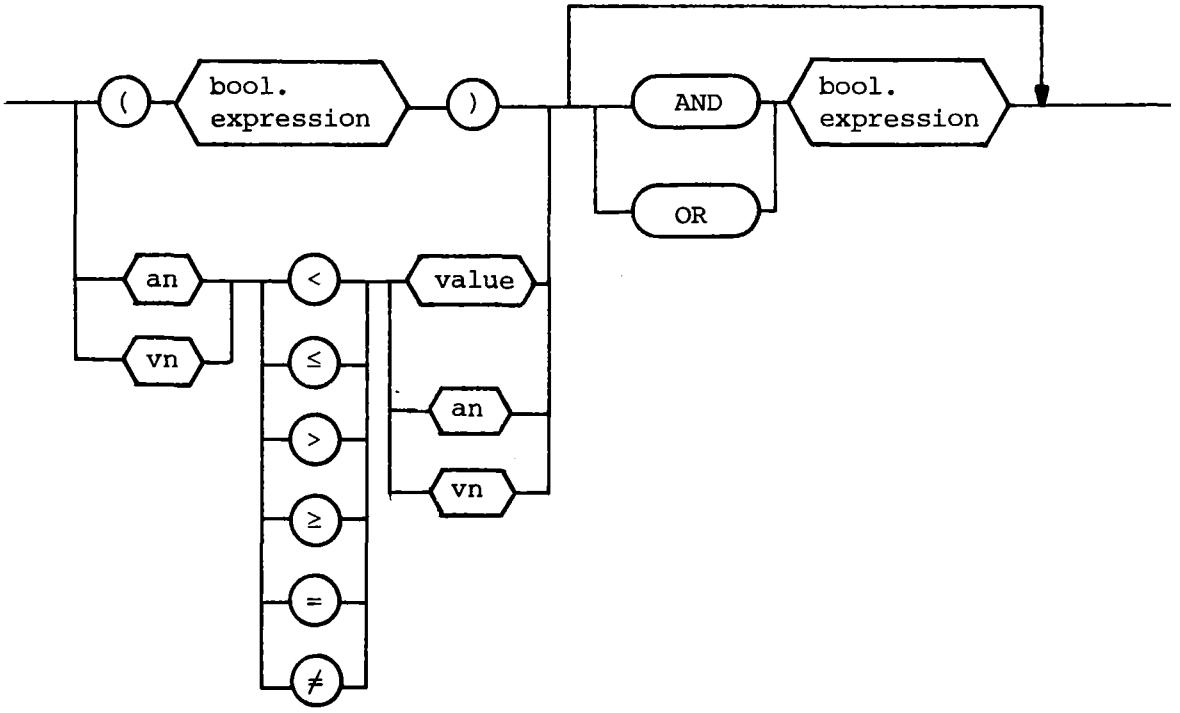



47 text:

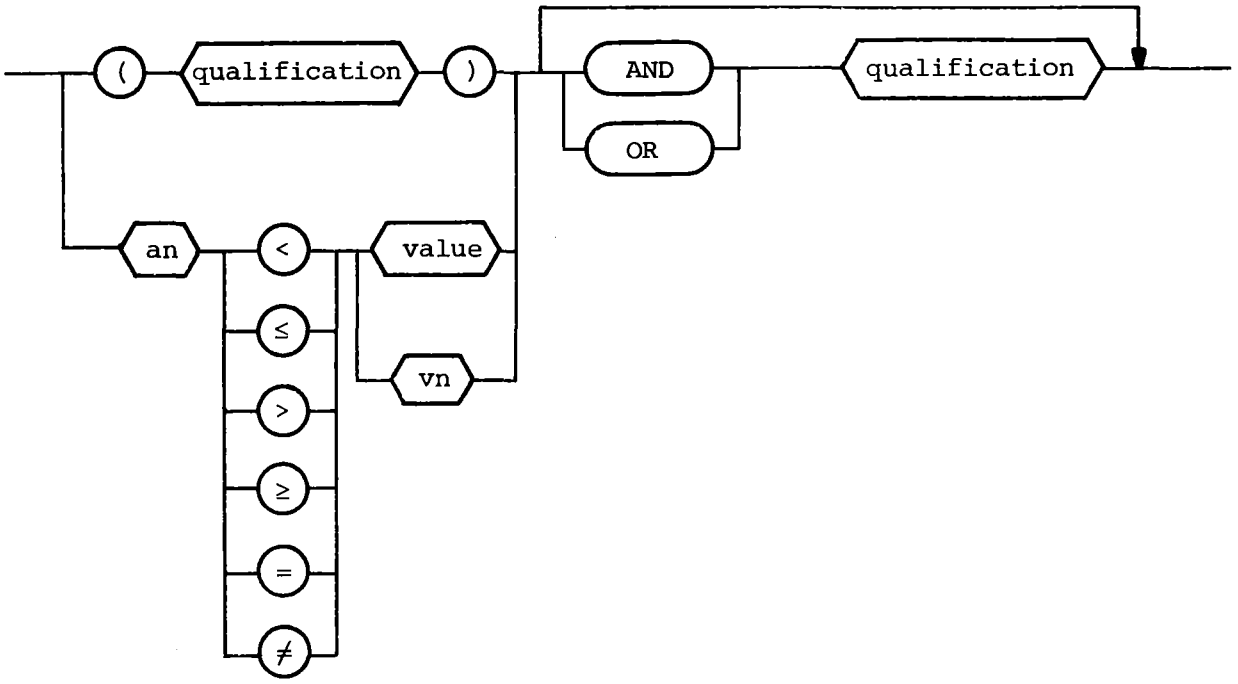



48 parameter:

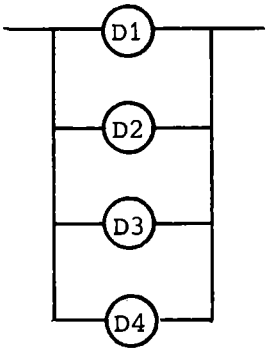

49 bool. expression:



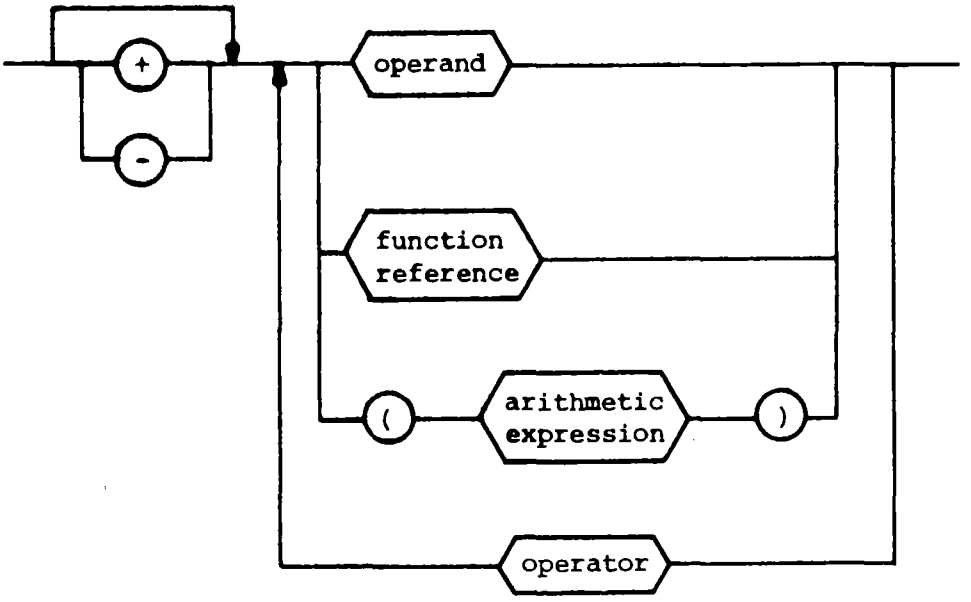
50 qualification:



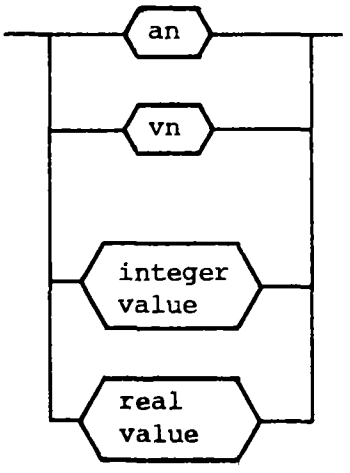

51 detailgrade:



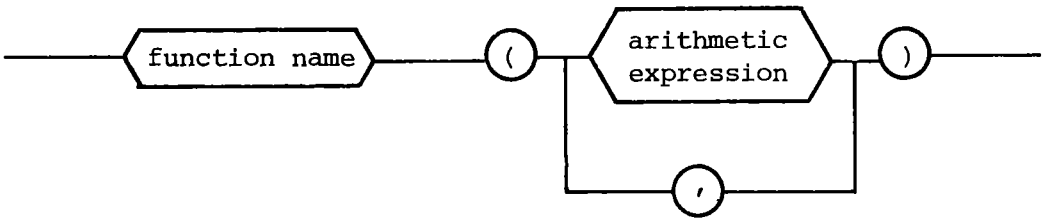
52 **arithmetic expression**:



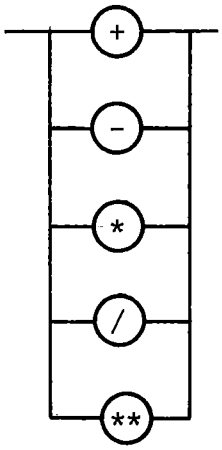

53 **operand**:



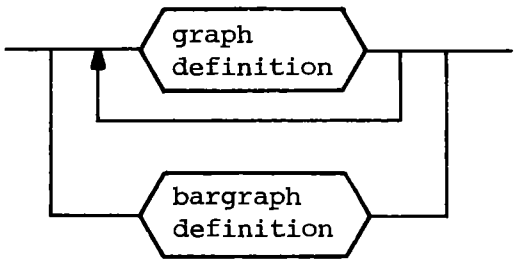
54 function reference:



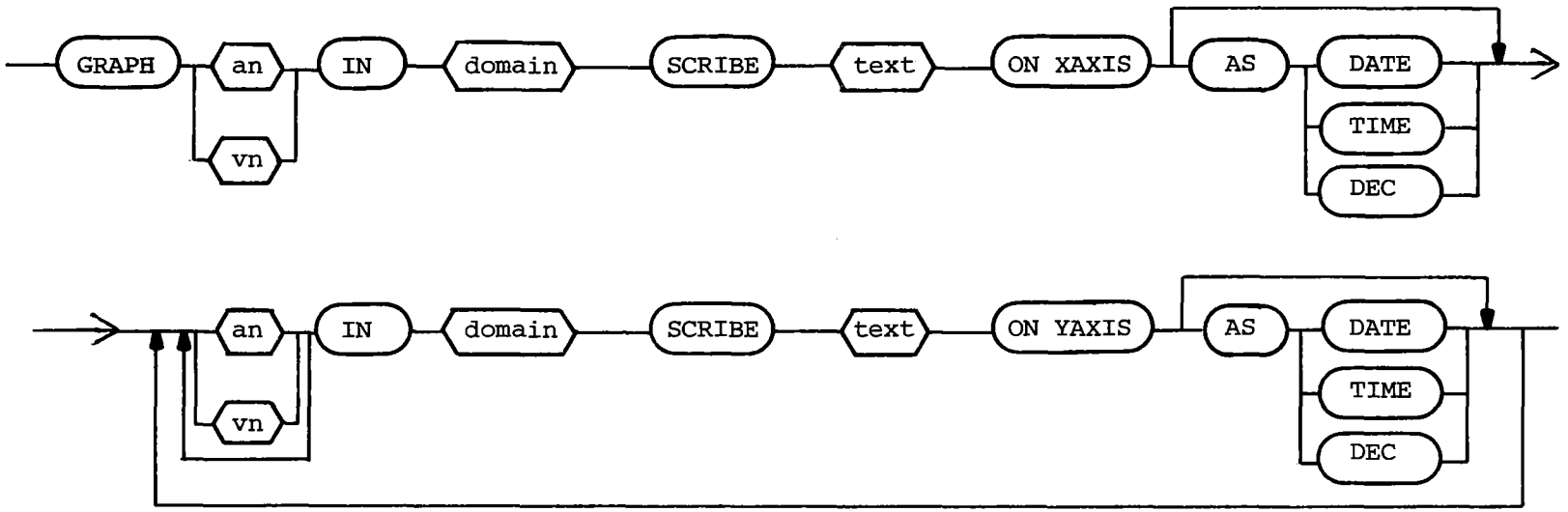

55 operator:



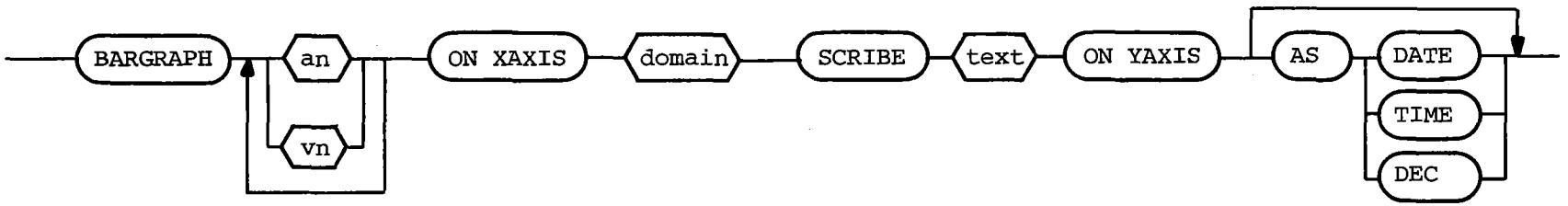

56 picture definition:



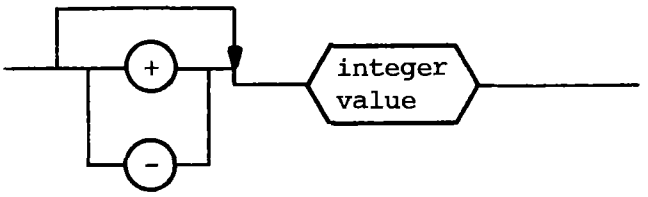
56a graph definition:



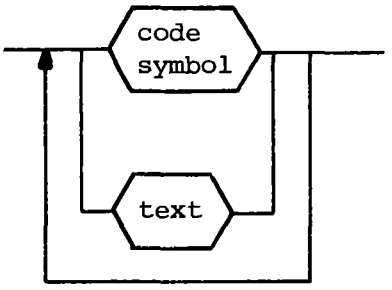

56b bargraph definition:



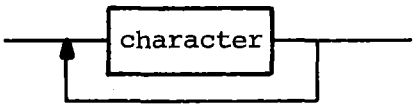
57 printposition:



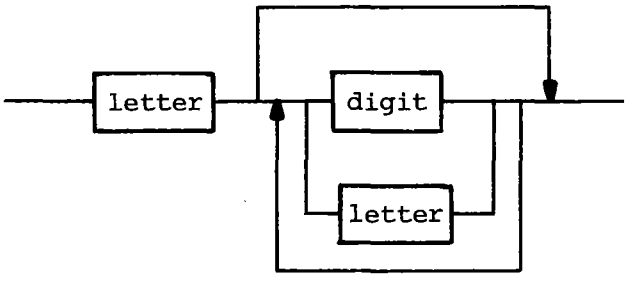

58 formatcode:




59 linecharacter

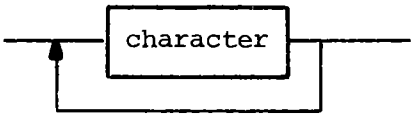



60 name:

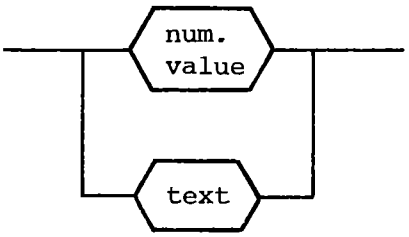



61 integer value:

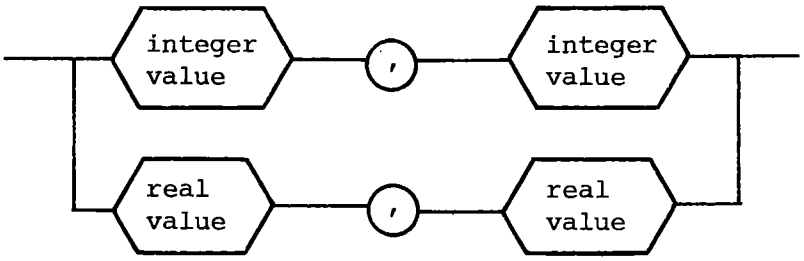
62 character string:



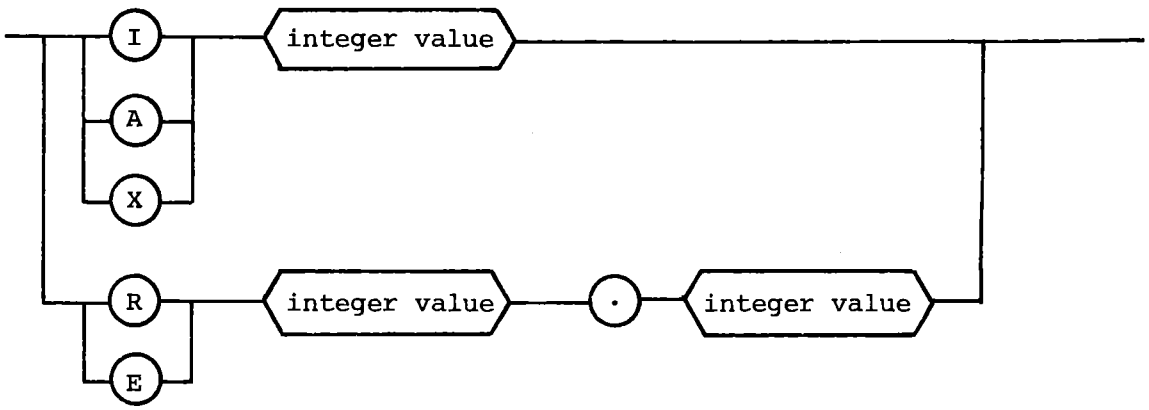

63 value:



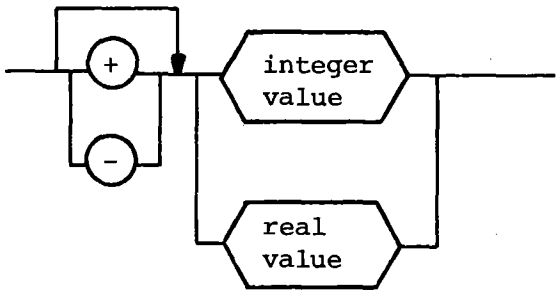

64 domain:



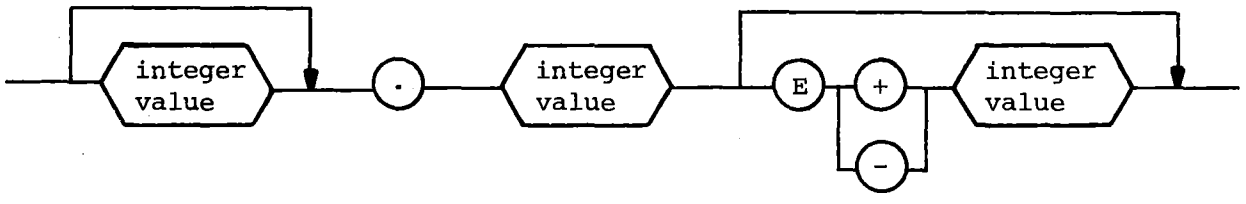

65 code symbol:



66 num. value:




67 real value:



## 7.3 Index

Abkürzungen:

| | |
|---|---|
| fn | filename |
| an | attributename |
| vn | variablename |
| rn | recordname |

## 8. Literatur

/CAGE 76/ Cagley, E.M.; et al.: Information Management System - Reference Manual. Microfiche PB-258915, Mathematics and Computation Lab. (EDM), Washington D.C. (Oct. 1976), pp. 103-121.

/CHAD 76/ Chamberlin, D.D.; et al.: SEQUEL2: A Unified Approach to Data Definition, Manipulation and Control. IBM Journal of Research and Development, Vol. 20 (1976), pp. 560-575.

/CODE 70/ Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, Vol. 13 (1970) No. 6, pp. 377-387.

/DIJE 72/ Dijkstra, E.W.: Notes on Structured Programming. In: Dahl, O.-J.; Dijkstra, E.W.; Hoare, C.A.R.: Structured Programming. London: Academic Press, 1972, pp. 1-82.

/GRID 76/ Gries, D.: Some Comments on Programming Language Design. In: Programmiersprachen, 4. Fachtagung der GI, Erlangen, März 1976, Schneider, H.-J.; Nagl, M. (eds.). Informatik-Fachberichte, Vol. 1, 1976, Springer Verlag, Berlin, pp. 235-252.

/HUIM 75/ Huits, M.H.H.: Requirements for Languages in Data Base Systems. In: Douqué, B.C.M.; Nijssen, G.M. (eds.): Data Base Description. Amsterdam, North-Holland Publishing Co., 1975, pp. 85-108.

/POLF 78/ Polster, F.J.: FADABS: Ein Datenbanksystem für den Siemens Prozeßrechner 330. In: Voges, U.: Tagungsbericht der 9. Jahrestagung des Siemens Prozeßrechner-Anwenderkreises I, Kernforschungszentrum Karlsruhe, 5.-7. April 1978. KfK-Bericht 2642, Kernforschungszentrum Karlsruhe, S. 227-244.

/POLF 79/ Polster, F.J.: MIAP: Ein Modul zur interpretativen Ausführung von Programmen. In: Abend, P.; Pangritz, R. (Hrsg.): Tagungsbericht des Siemens-Prozeßrechner-Anwenderkreises I. HMI Berlin, 9.-11. Mai 1979, S. 123-133.

/POST 78/ Polster, F.J.; Stöckle, D.: Grundlagen und Funktionen von Report-Generatoren. KfK-Bericht 2644, Juni 1978, Kernforschungszentrum Karlsruhe.

/PREC 78/ Prenner, C.J.; Rowe, L.A.: Programming Languages for Relational Data Base Systems. In: Gosh, S.P.; Liu, L.Y. (eds.): Proceedings of the National Computer Conference, Anaheim, June 5-8, 1978. Montvale, AFIPS Press, Vol. 47, 1978, pp. 849-855.

/QUER 76/ QUERY - HP3000 Series II Computer System, Reference Manual. Prod. No. 32216A, Hewlett Packard, Santa Clara, California (1976).

/SHNB 78/ Shneiderman, B.: Improving the Human Factors Aspect of Database Interactions. ACM Transactions on Database Systems, Vol. 3 (1978) No. 4, pp. 417-439.

/SYST 74/ SYSTEM 2000 - Report Writer Feature.
MRI Systems Corporation, Austin/Texas (1974).

/VANE 77/ Vandijck, E.: Towards a More Familiar Relational Retrieval Language. Information Systems, Vol. 2 (1977) No. 4, pp. 159-169.

## ANHANG: Implementierung des Report-Generators FAREG

Es soll im folgenden die Realisierung und Arbeitsweise des Report-Generators (FAREG) skizziert werden, dabei wird hauptsächlich auf die Programmstruktur eingegangen, eine detaillierte Programmbeschreibung ist nicht beabsichtigt. Diese wird mit der Programmierung erstellt.

Zur Realisierung wird als zugrundeliegendes Datenbanksystem FADABS benutzt; an dieser Stelle sei jedoch ausdrücklich darauf hingewiesen, daß RDL selbst unabhängig vom benutzten Datenbanksystem ist, dies gilt ebenso für weite Teile des Programmcodes.

Der Report-Generator selbst besteht aus drei, für das Betriebssystem selbständigen Prozessen; es sind dies:

- ein Generatormodul
- der RDL-Übersetzer
- ein Interpretermodul.

Diese Prozesse benötigen als Systemkataloge zwei FADABS Relationen:

RPGOBJ, ASR

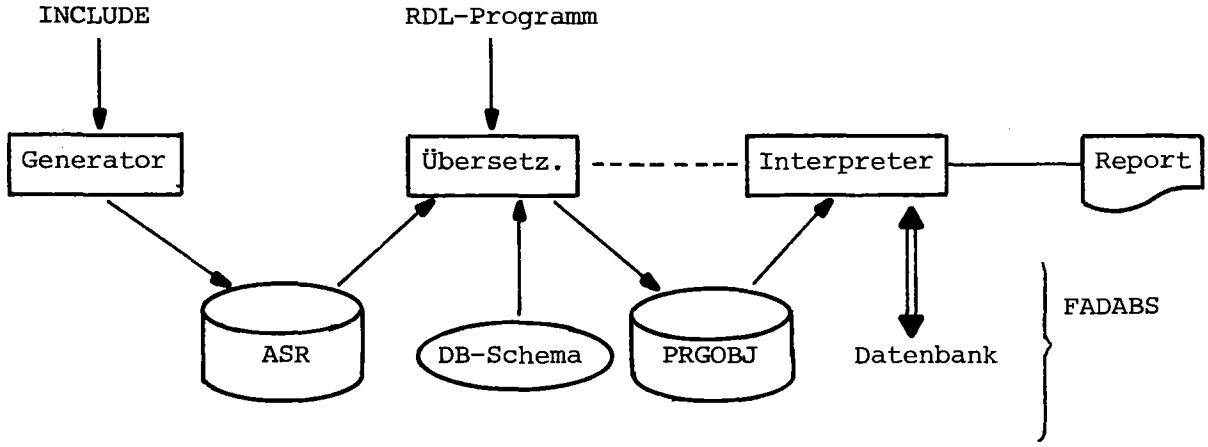


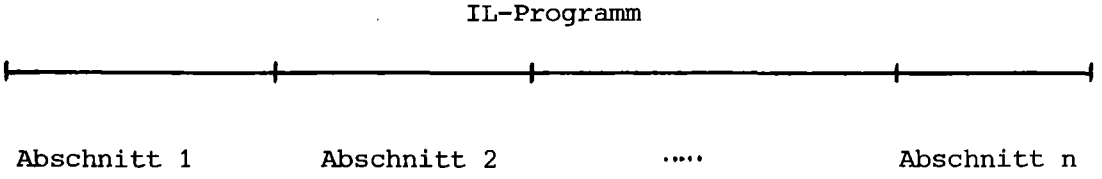

Bild A1: Architektur des Report-Generators.

Die Bestandteile von FAREG sollen nun kurz beschrieben werden.

Der Übersetzer

Wie man Bild A1 entnehmen kann, ist es Aufgabe des Übersetzers, die RDL-Programme in eine interne Form zu übersetzen. Die Ausgabe ist ein IL-Programm (IL = interpreter language) und wird in der Relation RPGOBJ abgelegt. Die Codeerzeugung und die Struktur der IL-Programme wurde im wesentlichen bereits in /POLF 79/ dargestellt.

Zu dieser Übersetzung und zur Erzeugung des IL-Programms benötigt der Übersetzer beschreibende Informationen über die im RDL-Programm angesprochenen FADABS-Relationen und die internen Namen der in CALL-Aufrufen referierten anwendungsspezifischen Routinen: es wird also lesend nur auf das Datenbank-Schema und die Relation ASR zugegriffen, hier findet noch kein Datenretrieval für den Report statt. Das erzeugte IL-Programm wird in die Relation RPGOBJ eingefügt. Um die Länge der IL-Programme nicht willkürlich beschränken zu müssen, werden diese in Abschnitte fester Länge eingeteilt und diese Abschnitte von 1 beginnend durchnumeriert




Die Tupel der Relation PRGOBJ enthalten je einen solchen Abschnitt (Attribut CODE), ergänzt um seine Nummer (Attribut LINENR) und den Report-Namen (Attribut REPORT).

RPGOBJ hat also folgende Struktur:

| Attribut | Attributtyp |
|---|---|
| REPORT | A*6 |
| LINENR | I*2 |
| CODE | A*400 |

Der Interpreter

Aufgabe des Interpreters ist die Ausführung der vom Übersetzer erzeugten IL-Programme in RPGOBJ.

Der eigentliche Interpreterteil und die Interpretersprache IL sind detailliert in /POLF 79/ beschrieben.

Zur Erzeugung eines Reports liest der Interpreter zunächst das IL-Programm aus RPGOBJ ('Hole alle Tupel von RPGOBJ mit REPORT = report-name'); dieser Vorgang wird 'LADEN' genannt. Nach dem Laden erfolgt die Interpretation des IL-Programms, also die Report-Erzeugung. Hierzu greift der Interpreter lesend (Retrieval der Problemdaten) und i.a. auch schreibend - inclusive Ändern des Datenbank-Schemas - (Ablegen von Zwischenergebnissen, einrichten/löschen von Hilfsrelationen etc.) auf die Datenbank zu.

Der Generatormodul

Der Generatormodul wird vor allem zum Einbringen von anwendungsspezifischen Routinen benötigt. Wie ebenfalls in /POLF 79/ beschrieben, ist mit INCLUDE-Karten anzugeben, welche Unterprogramme man in RDL-Programmen zu referieren in Lage sein möchte. Aus diesen Angaben erzeugt der Generator für jedes Unterprogramm eine Zahl als int. Namen und legt diesen mit den anderen Angaben in der Relation ASR (application specific subroutines) ab. Weiter werden in diesem Generierlauf die zwei FORTRAN-Unterprogramme ASSUB, ASFUNC des Interpreters erzeugt. Diese sind dann zu compilieren und mit diesen der Interpreter zu binden und laden. Im Gegensatz zu Interpreter liegt der Übersetzer als Programm vollständig vor und ist nicht erst zu generieren.

Die Relation ASR:

| Attribute | Attributtyp | |
|---|---|---|
| NAME | A* 6 | Unterprogrammname |
| INAME | I* 2 | interner Name |
| TYPE | A* 2 | Unterprogrammtyp |
| LIB | A*10 | Bibliothek |
| TYP1 | A* 2 | Typ des i-ten formalen Parameters |
| TYP2 | A* 2 | |
| TYP3 | A* 2 | |
| TYP4 | A* 2 | |
| TYP5 | A* 2 | |
| TYP6 | A* 2 | |
| TYP7 | A* 2 | |
| TYP8 | A* 2 | |
| TYP9 | A* 2 | |

Unterprogrammtypen:

S : subroutine

I2: INTEGER*2
I4: INTEGER
R4: REAL
R8: DOUBLE PRECISION

} - function