Referenzhandbuch

OREGATOR

Version V3.01.031

Reportgenerator
für SQL-Datenbanken

OPAG Informatik AG
Fabrikmattenweg 11
CH-4144 Arlesheim

Tel. 0041 61 716 92 22 • Fax. 0041 61 716 92 29
E-Mail: mailbox@casymir.ch
http://www.casymir.ch/

20.4.1998

1 Inhalt

1 Inhalt....3

2 Installation....4

2.1 Installation....4
2.2 Befehlsbeschreibung....5
2.3 Tutorial....10

3 Definitionssprache....16

3.1 Globaler Aufbau....16
3.2 Darstellungsbeschreibung....17
3.3 Verarbeitungsbeschreibung....34
3.4 Anweisungen....38
3.5 Grafik....50

4 *Tips & Tricks*....58

4.1 Compute-Resultate....58
4.2 Hinweise zum Wert NULL....60

5 FrameMaker....61

5.1 MIF Musterdateien....61
5.2 Regeln für MIF Musterdateien....61
5.3 Erzeugung von MIF Dateien....62

6 Windows Export....63

6.1 Erzeugung von Export-Dateien....63

7 Anhang....64

7.1 Limits und Konstanten....64
7.2 Format der Antwortdatei....65
7.3 Bild-Konversionsregeln....66
7.4 PDFLib Konfiguration....68
7.5 HPDF Konfiguration....69
7.6 Cairo PostScript Konfiguration....70
7.7 Treiberspezifische Optionen (-k)....71
7.8 Versionen....73

8 Index....78

2 Installation

Im folgenden Kapitel wird die Installation des Reportgenerators Oregator beschrieben, sowie die Vorgehensweise beim Erstellen von Reportdefinitionen.

2.1 Installation

Wählen Sie einen geeigneten Platz für die Oregator-Installation, es werden etwa 2 MByte Speicherplatz benötigt. Das Installationsverzeichnis (./oregatorV3.01.031) wird automatisch erzeugt.

Wenn Sie unter Linux / Unix arbeiten, installieren Sie Oregator aus der Distributionsdatei mit dem folgenden Befehl:

```
$ tar xvf /dev/rfd0c
$ tar xvf ./oregatorV3.01.031.tar.gz
$ cd ./oregatorV3.01.031
```

Möchten Sie den Bildkonverter nutzen, so installieren Sie das Paket 'GraphicsMagick' oder 'ImageMagick'. Unter RHEL können die Tools direkt aus dem EPEL-Repository z.B. mit folgendem Befehl installiert werdein:

```
$ yum install GraphicsMagick
```

Installieren Sie den Reportgenerator (die Datei **`oregator`**) in einem Inhaltsverzeichnis auf das von allen Oregator-Anwendern im Netz zugegriffen werden kann. Es ist sinnvoll, das Verzeichnis **`/home/oregator`** mit Hilfe von NIS (auto.home) zu verwalten. Der Eintrag sollte etwa so aussehen:

```
oregator servername:/files/home/oregatorV3.01.031
```

Fordern Sie bei OPAG die Lizenz-Informationen via Fax oder e-mail an. Es wird die Host-Id und der Hostname benötigt, z.B.:

```
$ hostid
55000bd1
$ hostname
gammon
```

Legen Sie die Lizenzdatei mit Hilfe eines Texteditors in einem allgemein zugänglichen Verzeichnis an.

```
$ cat > oregator.lic
55000bd1 4slD3lLra6o djZR2OEPSAQ # gammon
```

```
<Ctrl-D>
```

Jeder Anwender von Oregator sollte im Unix-Environment zwei Variablen eintragen:

Die Variable **OREGATOR_PATH** bestimmt die Verzeichnisse in denen die Reportdefinitionen gesucht werden. Die einzelnen Verzeichnisse müssen durch einen Doppelpunkt getrennt werden. Das Oregator-Installationsverzeichnis muss im Suchpfad enthalten sein, sonst findet Oregator die Bilder, AFM- und MIF-Dateien nicht.

Die Variable **OREGATOR_LICENSE** bestimmt die Lizenzdatei:

```
set path = (/home/oregator $path)
setenv OREGATOR_PATH `/home/user/rpt:/home/oregator'
setenv OREGATOR_LICENSE `/home/oregator/oregator.lic'
```

Es ist von Vorteil, diese Werte ins **.cshrc** oder entsprechend ins **.profile** des Benutzers eingetragen.

2.2 Befehlsbeschreibung

Der Reportgenerator wird durch den Befehl **oregator** gestartet. Die Befehlssyntax ist wiefolgt:

oregator *[* **-options** *]* **report_file arguments...**

Beim Start von Oregator ohne Programmoption und ohne Argumente wird eine Liste der verfügbaren Programm-Optionen ausgegeben. Die Programm-Optionen sind wie folgt definiert:

Allgemeine Optionen

-d	**Debug-Flag**
-v	**generate vertical layout**
-i directory	**prepend dir to OREGATOR_PATH**

-d, -dd, -ddd
Mit dieser Option werden während des Formatierungslaufes Debug-Informationen mit gewünschtem Detaillierungsgrad auf stderr ausgegeben. Durch Angabe von mehreren Debug-Flags kann der Detaillierungsgrad erhöht werden.

-v
Beim Generieren der Report-Definition werden die Felder Vertikal statt Horizontal angeordnet.

-i directory
Das angegebene Verzeichnis wird dem **OREGATOR_PATH** vorangestellt. Die Option kann mehrfach auf der

Kommandozeile verwendet werden. Achtung: die Kommandozeile wird von links nach rechts abgearbeitet.

Verarbeitungoptionen

```
-r filename      Report definition file
-f fmt_name      select output format
-g filename      generate report pattern
-p pagenr        print page nr only
-a pagenr        start output on page nr
-b pagenr        end output on page nr
-c #copies       number of copies
-T rule_file     name of TIFF conversion rules
```

-r report_name
Mit dieser Option wird der Name der Report-Beschreibung angegeben. Die Report-Beschreibung wird in dem durch **OREGATOR_PATH** angegebenen Verzeichnis gesucht. Bei Weglassen der Option wird der Report " **default.ry** " eingelesen.

-f format_name
Mit dieser Option wird das Ausgabeformat gewählt. Default Ausgabeformat ist "**Default** ". Die Bezeichnungen der Formate können in der Report-Definition frei vergeben werden.

-g report_file
Bei Angabe dieser Option wird ein SQL-Statement versuchsweise im SQL-Server ausgeführt. Das Resultat der Abfrage wird analysiert und eine entsprechende Report-Definition für Horizontales Layout wird erzeugt (siehe unten).

-p pagenr
Es wird nur die Seite mit der Seitennummer pagenr ausgegeben.

-a pagenr
Der Ausdruck beginnt auf der Seite mit der Seitennummer pagenr.

-b pagenr
Der Ausdruck endet auf der Seite mit der Seitennummer pagenr.

-c #copies
Bestimmt die Anzahl der Kopien, die von jeder Ausgabeseite gedruckt werden.

-T rule_file
Mit dieser Option kann die TIFF-Konversionsregeldatei

eingestellt werden.

Ausgabeoptionen

```
-o filename       result output file
-e                produce EPSF output
-k config_str     set output driver specific options
```

-o output_file
Normalerweise wird die formatierte Ausgabe auf stdout geschrieben. Bei Angabe dieser Option kann die Ausgabe direkt in eine Datei geschrieben werden. Die Datei wird beim Öffnen der Ausgabe geleert.

-e
Beim Erzeugen von POSTSCRIPT Formaten wird anstelle von Standard Adobe PostScript ein Adobe EPSF File erzeugt.

-k
Diese Option setzt Treiberspezifische Optionen für die Ausgabeerzeugung. Der Konfigurationsstring hat das Format: key=val{,key=val}.

Sybase Optionen

```
-S server         SQL server name
-U username       SQL user name
-P passwd         SQL user passwd
-D database       SQL database
-C charset        SQL character set
-q                SQL quote arguments
```

-S server_name
Bestimmt den SQL-Server, zu dem die Verbindung aufgebaut werden soll. Bei Weglassen dieser Option wird der Wert der Environment-Variablen " **DSQUERY** " eingesetzt.

-U user_name
Bestimmt den Datenbankbenutzer beim Aufbau der Verbindung mit dem SQL-Server. Bei Weglassen dieser Option wird der Wert der Environment-Variablen "**USER** " eingesetzt.

-P password
Bestimmt das Datenbank-Benutzerpasswort beim Aufbau der Verbindung mit Sybase.

-D database_name
Bestimmt den Namen der Datenbank in der die SQL Abfrage ausgeführt werden soll. Bei Weglassen der Datenbankangabe wird die Abfrage in der Default-Datenbank des jeweiligen

Benutzers ausgeführt.

-C character_set
Bestimmt den gewünschten Client-Zeichensatz. Durch Angabe dieser Option wird ggf. eine serverseitige Zeichensatz-Translation eingeschaltet. Wird die Option weggelassen, so arbeitet OREGATOR mit dem Default-Zeichensatz, welcher in den Konfigurationsdaten der SQL Client-Library-Installation angegeben werden kann. Zulässige Werte sind z.B. „iso_1“ oder „utf-8“.

-q
Werden auf der Kommandozeile mehrere SQL-Argumente angegeben, so verkettet Oregator die Argumente. Zwischen den Argumenten wird ein Komma eingefügt. Bei Angabe der Quote-Option werden die hinzugefügten Argumente durch Apostroph Zeichen eingeschlossen (single quotes). Die Wirkung der Option kann mit der Debug-Option überprüft werden:

```
$ oregator -oo -d sp_help par1 par2
SQL = <exec sp_help par1,par2>
$ oregator -oo -d -q sp_help par1 "2a 2b"
SQL = <exec sp_help 'par1','2a 2b'>
```

Diese Option muss unbedingt angewendet werden, wenn die Parameter Leerzeichen enthalten.

SOLID Optionen

```
-R protocol       Protokoll Name (tcp)
-S hostname       Server Hostname (localhost)
-N portnum        Portnummer (1313)
-U username       SQL user name
-P passwd         SQL user passwd
-q                SQL quote arguments
```

-R protocol
Angabe des Protokolls für den Verbindungsaufbau mit der SOLID Datenbank.

-S hostname
Angabe des Server-hosts für den Verbindungsaufbau mit der SOLID Datenbank.

-N portnum
Angabe der Portnummer für den Verbindungsaufbau mit der SOLID Datenbank.

MySQL Optionen

```
-R protocol       Protokoll Name (MySQL)
-S hostname       Server Hostname (localhost)
-N portnum        Portnummer (1313)
-U username       SQL user name
-P passwd         SQL user passwd
-D database       SQL database
-q                SQL quote arguments
```

MSSQL Optionen

```
-R protocol       Protokoll Name (MSSQL)
-S server         SQL Servername
-U username       SQL user name
-P passwd         SQL user passwd
-C charset        SQL character set
-D database       SQL database
-q                SQL quote arguments
```

Dateneingabe Optionen

```
-I                read data from file
```

-I
Bei Angabe dieser Option wird anstelle des SQL-Servers eine Antwortdatei als Eingabekanal verwendet. Der Dateiname wird als Argument anstelle des SQL-Befehls eingesetzt. Die Antwortdatei kann mit der Option -O aus einem SQL-Befehl erzeugt werden. Die Antwortdatei kann mit einem Texteditor verändert werden. Das Format der Antwortdatei ist im Anhang beschrieben.

Datenausgabe Optionen

```
-O filename       save data to filename
```

-O response_file
Wird beim Ausführen einer SQL-Server Abfrage diese Option angegeben, so wird das strukturierte Abfrageresultat des SQL Servers in einer Antwortdatei abgelegt. Die Antwortdatei (Textdatei) kann später als Eingabekanal für die Formatierung (ohne SQL-Server) verwendet werden (siehe unten). Die Antwortdatei kann mit einem Texteditor verändert werden. Das Format der Antwortdatei ist im Anhang beschrieben.

Logging and journal options

```
-L                save log to SQL database
-l filename       save log to filename
-J                save journal to SQL database
-j filename       save journal to filename
-s                syslog error messages
```

-L
Logging in SQL Datenbank. Alle Fehler-, Debug- und Fortschrittsmeldungen werden in der SQL-Datenbank in der Tabelle **`OregatorLog`** protokolliert. Sollte die Tabelle nicht existieren, so wird sie automatisch angelegt.

-l filename
Logging in Antwortdatei. Alle Fehler-, Debug- und Fortschrittsmeldungen werden in der angegebenen Datei aufgezeichnet. Die Datei besitzt das Format einer Antwortdatei und kann von Oregator gelesen und weiterverarbeitet werden.

-J
Journalisierung in SQL Datenbank. Alle Datensätze welche mit den Befehlen OPEN, CLOSE und OUTPUT erzeugt werden, fügt Oregator in der SQL-Datenbank in die entsprechende Tabelle ein.

-j filename
Journalisierung in Antwortdatei. Alle Datensätze welche mit den Befehlen OPEN, CLOSE und OUTPUT erzeugt werden, legt Oregator in der angegebenen Antwortdatei ab. Die Datei besitzt das Format einer Antwortdatei und kann von Oregator gelesen und weiter verarbeitet werden.

-s
Schaltet die Protokollierung von Fehlermeldungen über den standard Syslog-Service (Facility: loca3, Priority: log_info) ein.

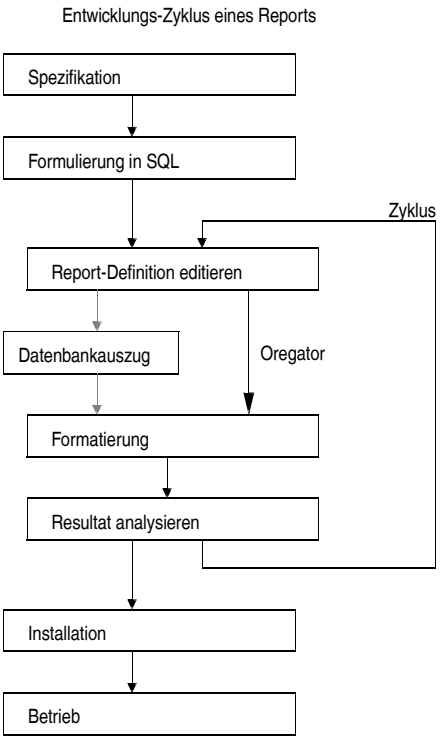
-W dbname
Zeichnet die I/O- und Time-Statistikdaten des SQL-Servers in der Tabelle SQLStats in der angegebenen Datenbank auf.

-w filename
Zeichnet die I/O- und Time-Statistikdaten des SQL-Servers in der genannten Datei im CSV-Format auf.

2.3 Tutorial

Um einen Report zu erstellen und das Aussehen in möglichst kurzer Zeit zu optimieren, sollte der Entwicklungs-Zyklus in möglichst kurzer Zeit abgewickelt werden können:

Oregator bietet Ihnen die Möglichkeit, das Resultat einer



Datenbankabfrage (die beliebig viel Rechenzeit beanspruchen kann) in einer Zwischendatei abzulegen. Die Formatierung und Darstellung des Reports kann aufgrund der zwischengespeicherten Daten in Sekundenbruchteilen erfolgen. Der typische Entwicklungs-Zyklus wird auf diese Weise wesentlich reduziert.

Der folgende Abschnitt beschreibt das Vorgehen bei der Reportdefinition anhand eines mitgelieferten Musterdatenbestandes. Voraussetzung für die Demonstration ist die Existenz einer Musterdatenbank. Die Musterdatenbank kann auf folgende Weise angelegt werden (**`sapw`** muss durch das SA-Passwort ersetzt werden):

```
$ isql -Usa -Psapw
> create database test
> go
<Ctrl-D>
$ cat crtestdb.sql | isql -Usa -Psapw
```

Als erstes wird eine Muster-Reportdefinition generiert:

```
$ oregator -Dtest -Usa -Psapw -gtest.r -oo \
"select * from Artikel"
```

Die Musterdefinition sieht etwa folgendermassen aus:

```
REPORT testr

FORMAT Default POSTSCRIPT (0,0,8.5,11) PORTRAIT;
FORMAT PDFHoch PDF (0,0,8.5,11) PORTRAIT;
FORMAT Text TEXT (0,0,8.5,11) CPI 15.0 LPI 10.0;

AREA Area ( 0.5, 0.5, 7.5, 10) VERTICAL;

BLOCK Block0 IN Area (,,,0.1)
  FIELD ArtNr (0,, 1.0,) ;
  FIELD Bezeichnung (*,, 4.0,) ;
  FIELD Preis (*,, 0.8,) ;
  FIELD BarCode (*,, 1.0,) ;
ENDBLOCK;

PROCESSING

BATCH
  RECORD
   NEW Block0;
   INPUT Block0;
   STROKE Block0;
  END;
ENDBATCH;

ENDREPORT
```

Die Muster-Definition lässt sich in diesem Fall direkt ausführen:

```
$ oregator -Dtest -Usa -Psapw -rtest.r \
      -oo "select * from Artikel"
$ pageview o &
```

Das Resultat ist PostScript und kann mit einem PostScript Viewer angeschaut werden. Mit Hilfe des folgenden Befehls kann das Resultat der SQL Query in der Antwortdatei **x** abgelegt werden:

```
$ oregator -Dtest -Usa -Psapw -rtest.r \
      -oo -Ox "select * from Artikel"
$ more x
```

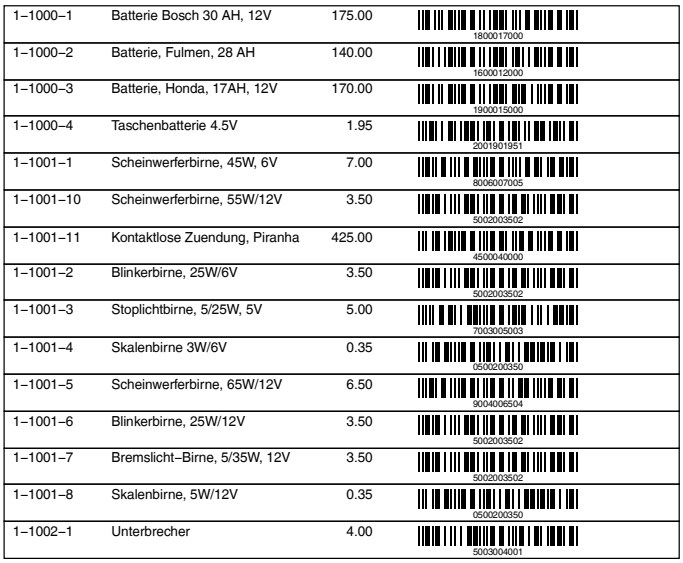
Um die Formatierung zu verfeinern, braucht nun der SQL Server nicht mehr mit der Query belastet zu werden. Die Formatierung mit Eingabe aus der Antwortdatei geschieht mit dem folgenden Befehl:

```
$ oregator -rtest.r -oo -I x
$ pageview o &
```

Ändern Sie nun die Blockdefinition in der Reportdefinition (**test.r**) mit Hilfe eines Texteditors folgendermassen ab:

```
BLOCK Block0 IN Area (,,,0.4)
  BOX B (0,0,,);
  FIELD ArtNr (0.1, 0.15,1.0,0.15) ;
  FIELD Bezeichnung (*,, 2.0,) ;
  FIELD Preis (*,, 0.8,) RIGHT;
  FIELD BarCode (4.6,0.05,2.0,0.3) BARCODE_25 0.01;
ENDBLOCK;
```

Nach dem Abspeichern der Reportdefinition wird die Formatierung (oben) nochmals durchgeführt. Das Ergebnis kann mit einem PostScript Viewer angeschaut werden. Es sieht etwa folgendermassen aus:



1–1000–1	Batterie Bosch 30 AH, 12V	175.00	1800017000
1–1000–2	Batterie, Fulmen, 28 AH	140.00	1600012000
1–1000–3	Batterie, Honda, 17AH, 12V	170.00	1900015000
1–1000–4	Taschenbatterie 4.5V	1.95	2001901951
1–1001–1	Scheinwerferbirne, 45W, 6V	7.00	8006007005
1–1001–10	Scheinwerferbirne, 55W/12V	3.50	5002003502
1–1001–11	Kontaktlose Zuendung, Piranha	425.00	4500040000
1–1001–2	Blinkerbirne, 25W/6V	3.50	5002003502
1–1001–3	Stoplichtbirne, 5/25W, 5V	5.00	7003005003
1–1001–4	Skalenbirne 3W/6V	0.35	0500200350
1–1001–5	Scheinwerferbirne, 65W/12V	6.50	9004006504
1–1001–6	Blinkerbirne, 25W/12V	3.50	5002003502
1–1001–7	Bremslicht–Birne, 5/35W, 12V	3.50	5002003502
1–1001–8	Skalenbirne, 5W/12V	0.35	0500200350
1–1002–1	Unterbrecher	4.00	5003004001

Sie können dieselbe Reportdefinition für die Erzeugung eines Text-Reports verwenden. Der entsprechende Befehl ist:

```
$ oregator -fText -rtest.r -oo -I x
$ more o
```

Am besten wird der SQL-Code welcher die Daten erzeugt als Stored Procedure (SP) formuliert. Die folgende SP wurde bereits durch das Anlegen der Demonstrations-Datenbank vordefiniert:

```
create procedure testproc
 @ArtNr varchar(10)
as

select
  ArtNr, Bezeichnung, BarCode, Preis, Date=getdate()
from OregatorTest
where ArtNr like @ArtNr
compute sum(Preis)
return
```

Sie kann z.B. durch folgenden Befehl ausgeführt werden. Bitte beachten Sie die Quote-Option:

```
$ oregator -Dtest -Usa -Psapw -rtest.r -qoo \
 -Ox "testproc" "1-1002%"
```

Das Resultat der Abfrage wurde in der Abfragedatei x abgelegt und kann mit derselben Reportdefinition ausgeführt werden. Es werden lediglich nicht alle selektierten Felder dargestellt. Wir ergänzen das fehlende Feld:

```
BLOCK Block0 IN Area (,,,0.4)
  BOX B (0,0,,);
  FIELD ArtNr (0.1, 0.15,1.0,0.15) ;
  FIELD Bezeichnung (*,, 2.0,) ;
  FIELD Preis (*,, 0.8,) RIGHT;
  FIELD BarCode (4.6,0.05,2.0,0.3) BARCODE_25 0.01;
  FIELD Date(*,0.15,0.8,0.15) DATETIME '%e.%m.%y';
ENDBLOCK;
```

Nach der Formatierung erscheint das Datumsfeld. Die SP enthält ein **COMPUTE** statement. Aus der SP wissen wir, dass die Summe des Feldes Preis berechnet wird. Wir ergänzen nun die Berechnungsvorschrift des ersten **BATCH** es durch einen Compute-Footer:

```
BATCH
 RECORD
      NEW Block0;
      INPUT Block0;
      STROKE Block0;
 END;
 COMPUTE
  FOOTER
      NEW Block0;
      INPUT Block0.Preis;
      Block0.Bezeichnung = 'Total';
      STROKE Block0;
  END;
 END;
ENDBATCH;
```

Nach der Formatierung erscheint nach dem letzten Datensatz der Summensatz mit der Bezeichnung "Total".

Zuletzt kann nun die für die Demonstration angelegte Datenbank wieder entfernt werden.

```
$ isql -U sa -P sapw
> drop database test
> go
```

<Ctrl-D>

3 Definitionssprache

Im folgenden Kapitel wird die Syntax und Semantik der gesamten Reportbeschreibungssprache beschrieben. Die Reportbeschreibung wird mit Hilfe eines Texteditors erstellt und verändert. Die Reportdefinition wird im folgenden mit RD bezeichnet.

3.1 Globaler Aufbau

REPORT
ENDREPORT

Eine RD besteht stets aus zwei Teilen. Der erste Teil beschreibt die Darstellung der Informationen, der zweite Teil die Verarbeitung des Datenbankauszuges. Die beiden Teile der RD werden durch das Schlüsselwort **PROCESSING** getrennt.

```
REPORT name
      ...Darstellungsbeschreibung...
PROCESSING
      ...Verarbeitungsbeschreibung...
ENDREPORT
```

Normalerweise endet die Reportverarbeitung nach Erhalt des letzten, vom SQL-Server erhaltenen Datenbatches, mit der Verarbeitung vom betreffenden **BATCH FOOTER**. Alternativ kann zur Trennung der Teile auch das Schlüsselwort **PROCESSING_ALL_BATCHES** eingesetzt werden. Dies hat den Effekt, dass nach Erhalt und Verarbeitung des letzten Datenbatches vom SQL-Server zusätzlich auch noch sämtliche **BATCH HEADER** und **BATCH FOOTER** sequentiell bis zum Ende der Reportdefinition ausgeführt werden.

INCLUDE 'datei';

Eine RD kann auf mehrere Dateien verteilt werden. Durch Einfügen eines **INCLUDE** Befehles wird die angegebene Datei anstelle des **INCLUDE** Befehles in die RD eingesetzt. Die maximale Verschachtelungstiefe können Sie dem Anhang entnehmen.

Auf diese Weise ist es möglich und auch empfehlenswert, gemeinsame Teile ein einziges Mal in einer Datei zu definieren und in die verschiedenen RD mittels **INCLUDE** einzusetzen.

PREPEND 'prefix/';

Durch die Angabe eines Präfix mit Hilfe von PREPEND, wird die Arbeitsweise des INCLUDE-Befehls verändert. Wird ein nicht-leere Zeichenkette als Präfix angegeben, so versucht Oregator zweimal die Include-Datei auf dem OREGATOR_PATH zu lokalisieren. Zuerst indem der angegebene Präfix vor den Namen der Include-Datei gestellt wird, danach dasselbe nochmals ohne Präfix. Der PREPEND-Befehl gestattet auch die Angabe von Verzeichnissen, indem am Ende ein `/' gesetzt wird.

3.2 Darstellungsbeschreibung

Zu Beginn der Darstellungsbeschreibung können verschiedene Einstellungen eingetragen werden:

```
NAN_SYMBOL `NaN';
INF_SYMBOL `Inf';
```

Die beiden Einträge **NAN_SYMBOL** und **INF_SYMBOL** bestimmen den Text, welcher im Falle von numerischen Ausnahmen ausgegeben werden soll.

Wird die Ausgabe für eine ASCII **TEXT** Datei aufbereitet, so können mit Hilfe des folgenden Befehls ausgewählte Kontroll-Sequenzen initialisiert werden:

```
CONTROL (
`', `',          # Reportbeginn, Reportende
`','\0c',        # Seitenbeginn, Seitenende
`','\n',         # Zeilenbeginn, Zeilenende
...              # weitere Sequenzen nach Bedarf
);
```

Die verschiedenen Sequenzen werden vor/nach der genannten Einheit gesendet. Es sind folgende Kontroll-Sequenzen zulässig:

```
\n               Linefeed, \0a
\r               Carriage Return, \0d
\t               Tab, \09
\hh              Sonderzeichen, Hexadezimal
\\               Backslash
alles andere     wird 1:1 gesendet
```

Im Falle der Erzeugung von FrameMaker MIF Dateien kann der Befehl **MIF_TEMPLATE** zur Bestimmung der Musterdatei eingesetzt werden:

```
MIF_TEMPLATE `Template.mif';
```

Die Darstellungsbeschreibung besteht aus drei Teilen und kann für verschiedene Ausgabe-Formate formuliert werden. Im ersten Teil der Darstellungsbeschreibung werden die möglichen Ausgabeformate definiert:.

```
FNC1_SYMBOL `Á';  # default symbol is 0xC1
```

Das auf 0xC01 vordefinierte FNC1-Zeichen wird für die Generierung von GS1-Datamatrix Codes verwendet. Es kann mit der Deklaration abgeändert werden.

FORMAT

```
FORMAT Default
  POSTSCRIPT (0.1, 0.2, 8.5, 11.0) PORTRAIT;
FORMAT A4Quer
  POSTSCRIPT (0.2, 0.1, 11.0, 8.5) LANDSCAPE;
FORMAT A4Right
  POSTSCRIPT (0.2, 0.1, 11.0, 8.5) ROTATE LEFT;
FORMAT A4Left
  POSTSCRIPT (0.2, 0.1, 11.0, 8.5) ROTATE RIGHT;
FORMAT PDFHoch
  PDF (0.1, 0.2, 8.5, 11.0) PORTRAIT;
FORMAT PDFQuer
  PDF (0.2, 0.1, 11.0, 8.5) LANDSCAPE;
FORMAT PDFA4Left
  PDF (0.1, 0.2, 8.5, 11.0) ROTATE LEFT;
FORMAT PDFA4Right
  PDF (0.2, 0.1, 11.0, 8.5) ROTATE RIGHT;
FORMAT Lpr
  TEXT (0, 0, 11.0, 18.5) CPI 15.0 LPI 10.0;
FORMAT Frame
  FRAMEMAKER (0, 0,10.5,6.5) CPI 15.0 LPI 10.0;
FORMAT Word
  WINWORD (0,0,8.5,11);
FORMAT BanditHoch
  BANDIT (0,0,2,4) PORTRAIT;
FORMAT BanditQuer
  BANDIT (0,0,4,2) LANDSCAPE;
FORMAT TECQuer
  TEC (0, 0, 8.4, 4.2) LANDSCAPE;
FORMAT TECHoch
  TEC (0, 0, 8.4, 4.2) PORTRAIT;
FORMAT PCLQuer
  PCL ( 0, 0, 11.7, 8.3) LANDSCAPE;
FORMAT PCLHoch
  PCL ( 0, 0, 8.3, 11.7) PORTRAIT;
FORMAT CABQuer
```

```
    CAB (0, 0, 2.2, 4) LANDSCAPE;
  FORMAT CABHoch
    CAB (0, 0, 2.2, 4) PORTRAIT;
  FORMAT PNGHoch
    PNG (0.1, 0.2, 8.5, 11.0) PORTRAIT;
```

Beim Start von Oregator kann das gewünschte Ausgabeformat als Option angegeben werden. Bei Weglassen der Format-Option wird das Format mit der Bezeichnung **`Default`** ausgewählt.

Durch die Angabe von **`POSTSCRIPT`** wird Postscript Output erzeugt.

Bei **`TEXT`** -Formaten muss zusätzlich die Anzahl Zeichen pro Inch (**`CPI`**) und die Anzahl Zeilen pro Inch (LPI) angegeben werden. Alle Positions- und Längenangaben werden auf die angegebenen Werte gerundet.

Das Format **`FRAMEMAKER`** erzeugt ein FrameMaker MIF File als Ausgabedatei.

Das Format **`WINWORD`** erzeugt tabellarische Daten (CSV), die auf dem PC unter Windows (Word, Excel) importiert werden können.

Zur Erzeugung von **`PDF`** Ausgabeformat wird Anstelle von **`POSTSCRIPT`** einfach **`PDF`** angegeben.

Der **`PNG`**-Treiber generiert im Arbeitsverzeichnis für jede Ausgabeseite eine Bilddatei. Dabei wird anNamen der Ausgabedatei die Seitennummer in der Form «.0001.png» anhängt.

Es folgt die Grösse des Papiers **`(x,y,w,h)`** . Sämtliche Grössenangaben in Oregator werden in Inch (1 Inch = 2.54 cm) gemacht. Die Werte x,y sind Platzhalter für die Position der linken oberen Ecke. Eine Veränderung der Positionsangabe verschiebt den gesamten Ausdruck auf dem Papier (Page Offset). Die beiden Angaben w,h bezeichnen die Breite und die Höhe.

Durch die Angabe von **`BANDIT`** als Format wird in der Ausgabedatei das Steuerformat für Datamax Barcodedrucker erzeugt. Die Optionen **`PORTRAIT`** / **`LANDSCAPE`** bestimmen die Formatlage.

Die Format-Option **`PORTRAIT`** entspricht der Normallage. Durch Format-Option **`LANDSCAPE`** wird die Ausgabe bei in den Ausgabe-Treibern **`POSTSCRIPT`**,**`TEC`**,**`BANDIT`**,**`CAB`** um 90 Grad rotiert. Im **`PDF`** Treiber hat **`LANDSCAPE`** keinen Einfluss, da die

Blattrotation in PDF-Viewern durch die Dimensionen bestimmt wird. Möchte man die Ausgabe explizit rotieren, so können in den Treibern **POSTSCRIPT**, **PDF**, **PNG** die Format-Optionen **ROTATE LEFT** rsp. **ROTATE RIGHT** angegeben werden.

Beim FORMAT PDF kann nach der Rotationsangabe die PDF-Version als String in Klammern angegeben werden: VERSION(`PDF/A-1B')[12].

Verfübare Treiber

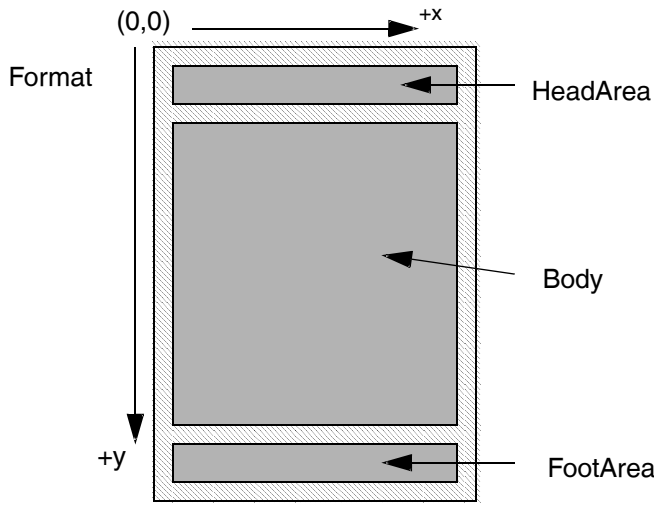
Oregator-Version	*UTF-8*	*ISO-8859-1*
POSTSCRIPT (Cairo)	X	
PDF (HARU)	X	
PNG	X	
TEXT	X	X
WINWORD (CSV)	X	X
FRAMEMAKER	X	X
POSTSCRIPT (Legacy)		X
PDF (PDFLib)		X
PCL		X
CAB (Etikettendrucker)		X
TEC (Etikettendrucker)		X
BANDIT (Datamax Etikettendrucker)		X

AREA

Nach der Format-Deklaration wird die Druckfläche global in mehrere Druckbereiche eingeteilt. Die Definition der

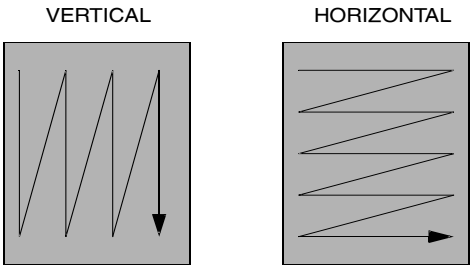
1 Die Angabe der VERSION ist heute nur im HPDF-Treiber unterstützt
2 Diese Option benötigt eine ICC Farbprofildefinition in der Datei afm/hpdf.icc. Siehe http://color.org

Druckbereiche kann entweder für alle Formate (keine Format-Angabe) oder für jedes Format einzeln geschehen.




```
AREA HeadArea FORMAT Default (0.6, 0.4, 10.5, 1.4);
AREA Body FORMAT Default (0.6, 1.85, 10.5, 5.55)
VERTICAL;
AREA FootArea FORMAT Default (0.6, 7.5, 10.5, 0.4);

AREA HeadArea FORMAT Lpr (0.1, 0.1, 10.5, 0.4);
AREA Body FORMAT Lpr (0.1, 0.7, 10.5, 16.6)
HORIZONTAL;
AREA FootArea FORMAT Lpr (0.1, 17.5, 10.5, 0.2);

AREA HeadArea FORMAT Frame (0.1, 0.1, 10.5, 0.4);
AREA Body FORMAT Frame (0.1, 0.7, 10.5, 16.6)
HORIZONTAL;
AREA FootArea FORMAT Frame (0.1, 17.5, 10.5, 0.2);
```

Ohne weitere Angabe wird ein Druckbereich statisch deklariert, d.h. der Inhalt des Druckbereiches befindet sich stets auf derselben Position. Durch die Angabe von **HORIZONTAL** oder **VERTICAL** wird der Druckbereich dynamisch deklariert. Der Inhalt eines dynamischen Druckbereiches wird vorzugsweise in der angegebenen Richtung mit Daten aufgefüllt und bewirkt einen Seitenvorschub, wenn der Druckbereich voll ist.



BLOCK

Im dritten Teil der Darstellungsbeschreibung befinden sich die Block-Beschreibungen. Ein Block fasst eine Menge von Objekten zusammen, welche mit Typ und Grösse versehen sind. Der Block selbst besitzt eine Grösse und Position. Die Angaben für die

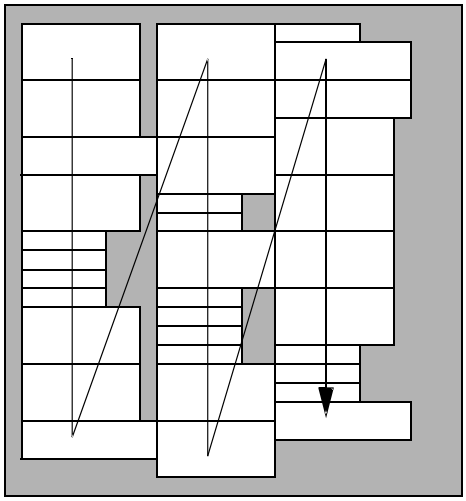
```
BLOCK Head FORMAT Default IN HeadArea (,,,0.65)
  ...Objektdefinitionen...
ENDBLOCK;

BLOCK Criteria FORMAT Default, Word IN HeadArea
(,1.15,,0.7)
  ...Objektdefinitionen...
ENDBLOCK;
```

Blockgrösse können weggelassen werden, es wird jeweils die entsprechende Grösse des zugehörigen Druckbereiches übernommen. Hinter dem Schlüsselwort **FORMAT** steht eine Liste von Formatbezeichnungen, für welche die Blockdeklaration gültig ist. Wird die **FORMAT** -Angabe weggelassen, so gilt die Blockdeklaration für alle Formate.

Beim Ausführen des Reports werden gemäss Verarbeitungsbeschreibung beliebig viele Instanzen eines Blockes mit Daten gefüllt und auf dem Papier plaziert.

Jeder Block wird bei der Deklaration einem Druckbereich zugeordnet. Falls es sich um einen statischen Druckbereich handelt, so erscheint jede neue Instanz eines Blockes an der selben Position. Ist der Druckbereich dynamisch, so wird jede neue Instanz gemäss Ausrichtung unterhalb oder rechts von der letzten generierten Instanz plaziert. Die Grösse des Blockes ist dabei für die Plazierung massgebend.

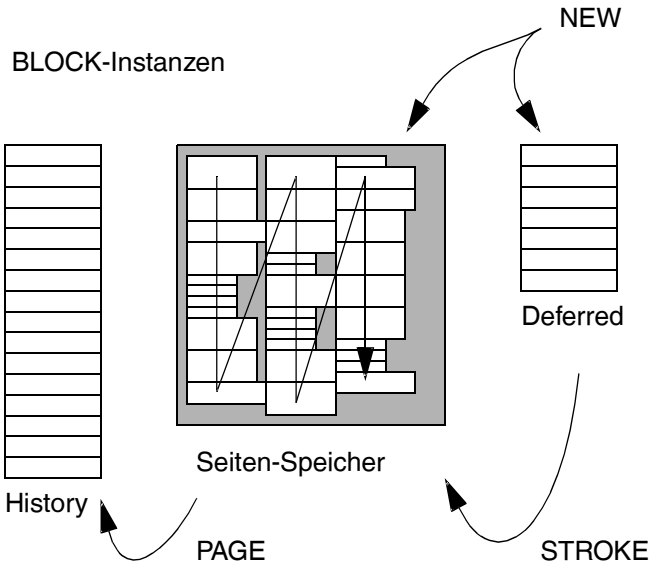
VERTICAL Positionierung von BLOCKs



Ein Block muss vor dem ersten Zugriff alloziiert werden. Dies geschieht mit der Anweisung **NEW** . Je nach Einstellung der Blockoptionen wird der Block direkt auf der aktuellen Seiten-Speicher oder im Deferred-Speicher angelegt.

Nach der Ausgabe der aktuellen Seite werden alle Felder die mit der Option **KEEP** versehen sind, in den History-Speicher übertragen. Die aktuelle Seite wird danach gelöscht.

Blöcke im Deferred-Speicher bleiben solange dort gespeichert, bis sie mit der Anweisung **STROKE** in den aktuellen Seitenspeicher übertragen werden.



Blockoptionen

Nach der Positionsangabe können verschiedene Optionen eingeschaltet werden.

Im Normalfall wird ein Block durch die Anweisung **NEW** See

Durch die Anweisung NEW wird eine neue Instanz eines Blocks erzeugt. Die neue Instanz steht nun zur Bearbeitung zur Verfügung. Bevor ein Feld mit einem Wert versehen werden kann oder ein Objekt gezeichnet wird, muss der entsprechende Block erzeugt werden. erzeugt und sofort auf dem Blatt an der nächsten freien Stelle positioniert. Wird ein Block in der Deklaration mit dem Schlüsselwort **DEFERRED** versehen, so wird der Block verzögert positioniert. Die Anweisung **NEW** erzeugt lediglich einen unsichtbaren, pendenten neuen Block. Dieser kann dann beliebig mit Feldinhalten versehen werden. Die Feldinhalte können auch wieder ausgelesen werden. Der Block bleibt über beliebig viele Seiten erhalten. Erst nach der Anweisung **STROKE** wird der Block positioniert und erscheint auf dem Papier. Bei **DEFERRED** Blöcken kann nur eine einzige verzögerte Instanz generiert werden. Versucht man weitere pendente Instanzen zu erzeugen, so wird lediglich die erste pendente Instanz geleert, d.h. allen Feldern des Blocks wir ein leerer Wert zugewiesen.

Durch die Angabe von **FIT_VERTICAL** oder **FIT_HORIZONTAL** wird die Grösse des Blocks im Moment der Positionierung in der entsprechenden Richtung an seinen Inhalt angepasst. Diese Optionen sind in der Regel nur in Kombination mit **DEFERRED** Positionierung interessant, darum wird automatisch. **DEFERRED** Mode eingeschaltet.

Um sehr lange **TEXT** Objekte umzubrechen, die nicht auf einer Ausgabeseite Platz finden, kann ein **BLOCK** gleichzeitig mit den Optionen **DEFERRED** , **FIT_VERTICAL** und **BREAK** versehen werden. Bei der Ausführung des **STROKE** Befehls, wird der Inhalt des **TEXT** Objektes in mehrere Stücke zerteilt. Der **BLOCK** wird vervielfältigt und jede Instanz des **BLOCKs** erhält ein Stück des Textes. Anschliessend werden die einzelnen Stücke auf der Seite platziert. Der Text wird immer dort getrennt, wo er einen Zeilenvorschub enthält (Newline). Das Trennzeichen (Newline) selbst wird entfernt.

Sollte der mit **BREAK** versehene **BLOCK** nicht nur ein **TEXT** Objekt sondern zusätzlich noch andere Objekte enthalten, so wird der Inhalt der anderen Objekte ab der 2. Instanz geleert.

Durch die Angabe von **INHERIT** wird ein Block mit Inhalt, der auf dem Blatt positioniert wurde, automatisch an alle weiteren Seiten vererbt. Die einzelnen Felder des Blocks werden beim Seitenumbruch stets neu erzeugt und `frisch' positioniert. Sind mehrere **INHERIT** -Blöcke vorhanden, so werden diese in der Reihenfolge vererbt, in der sie erzeugt wurden.

Ein Block kann mit der Option **NEED_VERTICAL** oder **NEED_HORIZONTAL** versehen werden:

```
BLOCK Fusszeile IN Body (,,,0.65)
  ...Objektdefinitionen...
ENDBLOCK;

BLOCK NormalZeile IN Body (,1.15,,0.1)
  NEED_VERTICAL Fusszeile
  ...Objektdefinitionen...
ENDBLOCK;
```

Auf diese Weise kann sichergestellt werden, dass beim Füllen von dynamisch gefüllten **AREA** s noch Platz für den angegebenen **BLOCK** reserviert bleibt, wenn ein Seitenumbruch ausgelöst wird. Der benötigte **BLOCK** muss vorher deklariert werden. Es ist auch eine transitive Verschachtelung der Benötigung möglich. Im **PAGE FOOTER** ist dann in jedem Fall noch genügend Platz vorhanden, um den benötigten **BLOCK** auf der alten Seite ausgeben zu können. Diese Option ist besonders wertvoll, wenn man z.B. unten auf jeder Seite eine Summen-Übertrags-Zeile programmieren möchte.

```
BLOCK UnterZeile IN Body (,1.15,,0.1) BOTTOM
  ...Objektdefinitionen...
ENDBLOCK;
```

Wird ein **BLOCK** mit der Option **BOTTOM** versehen, so rutscht der **BLOCK** bei der Positionierung in **VERTICAL** positionierten Blöcken stets ans untere Ende der **AREA** .

Objektdeklaration

Es werden verschiedene Typen von Objekten unterschieden. Bei der Objektdeklaration können die Objekte jeweils mit einer Position und Grösse versehen werden.

Die einzelnen Werte für Position oder Grösse können weggelassen werden, es wird jeweils der Wert des vorhergehenden Objektes übernommen. Für das erste Objekt werden als Defaultwert die Grössenparameter des Blockes gewählt.

Wird als Positionswert ein ***** (Stern) eingesetzt, so wird das Objekt neben bzw. unter dem vorhergehenden Objekt plaziert, verschoben um dessen Grösse / Höhe plus ein kleiner Zwischenraum (siehe Anhang).

Wird als Positionswert ein **-*** (Minus-Stern) eingesetzt, so wird das Objekt vor bzw. über dem vorhergehenden Objekt plaziert, verschoben um dessen Grösse / Höhe plus ein kleiner Abstand.

Alle Objekte werden mit einem Namen bezeichnet, welche

innerhalb des Blockes eindeutig sein muss. Die Objektnamen werden später bei der Verarbeitung eingesetzt, um die Objekte zu identifizieren.

In Klammern wird stets die Position (x,y,Breite,Höhe) des Objektes relativ zum Block angegeben.

LINE

Der einfachste Objekttyp ist eine Linie. Es handelt sich dabei um ein grafisches Objekt ohne berechenbaren Inhalt.

```
LINE ObjName (0,0,8.5,0);
LINE ObjName (1.4,0,0,2.5) BORDER 1;
LINE ObjName (1.4,0,0,2.5) BORDER 2 RGB(0.5,0.0,0.8);
LINE ObjName (1.4,0,0,2.5) BORDER RGB(0.5,0.0,0.8);
```

Mit der Angabe von **BORDER** kann die Strichdicke in ganzzahligen Schritten auf 1 Punkt (1/72 Inch) genau eingestellt werden. **BORDER 0** zeichnet eine Haarlinie. Wird die Angabe **BORDER** weggelassen, so wird **BORDER 1** angenommen. Die Angabe des **RGB** -Wertes bestimmt die Farbe der Linie.

BOX

Eine Box wird analog zur Linie definiert. Eine Box besteht aus 4 Umrisslinien und einem optionalen Füllmuster.

```
BOX ObjName (0,0,8.5,0);
BOX ObjName (1.4,,1.0,2.5) LIGHTGRAY BORDER 0;
BOX ObjName (*,,,) GRAY BORDER 2;
BOX ObjName (*,,,) DARKGRAY BORDER 3;
BOX ObjName (*,,,) RGB(1,0,0);
BOX ObjName (*,,,) RGB(1,0,0) BORDER 2 RGB(0,1,0);
```

Die Angabe des **RGB** -Wertes[3] vor **BORDER** bestimmt die Füllfarbe. Die Angabe des **RGB** -Wertes hinter dem **BORDER** bestimmt die Umrisslinien-Farbe.

IMAGE

Die folgende Deklaration definiert ein EPSF-Bild, welches in einer Datei abgelegt ist. Bilder werden nur auf dem PostScript Ausdruck sichtbar.

```
IMAGE ObjName (0,0,8.5,0);
IMAGE ObjName (0,0,8.5,0) `FileName';
IMAGE ObjName (0,0,8.5,0) DBIMAGE;
```

31. Angabe der Grauwerte: 0 = schwarz bis 9 = weiss

```
IMAGE ObjName (0,0,8.5,0) FGND;
```

Im ersten Beispiel kann der Bildname berechnet oder in einer Query selektiert werden. Im zweiten Beispiel ist der Bildname konstant. Im dritten Beispiel wird das Bild direkt aus der Datenbank selektiert.

Bitte achten Sie darauf, dass beim SYBASE SQL Server die Spaltenbreite bei **TEXT** und **IMAGE** Feldern per Default auf 32768 Bytes begrenzt ist. Diese Grenze kann mit Hilfe des SQL-Statements `set textsize' eingestellt werden.

Wenn es sich um eine nicht direkt unterstützte Bilddatei handelt, so kann die Datei anhand der Bild-Konversionsregeln automatisch in eine unterstütztes Datenformat kovertiert werden. Siehe Anhang [7.3 Bild-Konversionsregeln],

Bilder werden grundsätzlich in der Z-Ebene im Hintergrund plaziert. Hintergrundbilder werden von allen anderen Objekten abgedeckt. Die Option **FGND** kann bei PostScript- und PDF-Treibern dazu verwendet werden, ein Bild im Vordergrund zu plazieren.

Das Bild wird in das angegebene Rechteck proportional eingepasst. Die Bilder müssen im Unterverzeichnis **$OREGATOR_PATH/images** abgelegt werden.

FIELD

Felder sind Objekte, welche einen berechenbaren Wert enthalten. Der Wert ist entweder ein Text, eine Zahl oder ein Datum. Felder ohne Positionsangabe sind reine Berechnungsfelder, welche nicht auf dem Papier sichtbar sind.

```
FIELD FieldName;
FIELD FieldName (0,0,1.45,0.115);
FIELD FieldName (*,,1.45,) LEFT;
FIELD FieldName (*,,1.45,) RIGHT;
FIELD FieldName (*,,1.45,) CENTERED;
FIELD FieldName (*,,1.45,) UNIQ;
FIELD FieldName (*,,1.45,) FONT 'FontName';
FIELD FieldName (*,,1.45,) DATETIME 'DateFormat';
FIELD FieldName (*,,1.45,) FIXED num;
FIELD FieldName (*,,1.45,) KEEP;
FIELD FieldName (*,,1.45,) RGB(0,0.5,1);

FIELD FieldName (*,,1.45,) LEFT FONT 'Helvetica-Bold'
UNIQ KEEP;
```

Mit **RGB** wird die Zeichenfarbe eingestellt.

TEXT

Objekte vom Typ **TEXT** gestatten die Formatierung von fliessendem Text mit Zeilenumbruch.

```
TEXT FieldName (*,,1.45,2.3,20,0.115) Feldoptionen ;
```

Die **TEXT** Objekte verhalten sich bezüglich der Feldoptionen gleich wie **FIELD** Objekte. Die **TEXT** Objekte verfügen über zwei zusätzliche Parameter (am Ende der Klammerung). Der fünfte Parameter in der Klammer gibt die Anzahl der im Objekt darzustellenden Zeilen an. Der sechste Parameter bestimmt die Schriftgrösse. Auf diese Weise ist es möglich, den Zeilenabstand zu kontrollieren.

Feldoptionen

Nach der Positionsangabe kann eine Menge von Optionen eingeschaltet werden. Die Schlüsselwörter **LEFT** , **CENTERED** , **RIGHT** beziehen sich auf die Ausrichtung des Inhaltes innerhalb der angegebenen Feldgrösse.

Die Option **UNIQ** bewirkt, dass das Feld nur dann sichtbar wird, wenn sich der Feldinhalt gegenüber der letzten Instanz auf derselben Seite geändert hat. Die erste Instanz auf einer neuen Seite ist in jedem Fall sichtbar. Gleiche Werte werden auf derselben Seite unterdrückt. Das Feld behält aber trotzdem seinen berechenbaren Wert.

Durch die Angabe von **FONT** kann eine PostScript Schrift bezeichnet werden, welche für die Ausgabe des Feldinhaltes eingesetzt werden soll. Die Definition des Zeichensatzes gilt für alle weiteren Felddefinitionen, bis zur nächsten **FONT** -Angabe. Die Höhe der Schrift ergibt sich aus der Feldhöhe.

Tabelle 1 : PostScript Standard Fonts

Times-Roman *Times-Italic* **Times-Bold** ***Times-BoldItalic***
Helvetica *Helvetica-Oblique* **Helvetica-Bold** ***Helvetica-BoldOblique***

```
Courier
Courier-Oblique
Courier-Bold
Courier-BoldOblique
```

Für den Gebrauch eines PostScript Fonts muss eine entsprechende AFM-Datei im Verzeichnis **$OREGATOR_PATH/afm** vorhanden sein.

Die Angabe eines **DATETIME** bewirkt, dass die Ausgabe des Feldinhaltes als Datum/Zeitangabe formatiert wird. Die Zeichenkette **DateFormat** legt fest, wie das Resultat auszusehen hat. Es gilt die folgende Übersetzungstabelle:

Tabelle 2 : Datumsformatierung

%%	%
%a	Wochentag, Abkürzung
%A	Wochentag, voll
%b, %h	Monatsname, Abkürzung
%B	Monatsname, voll
%c	Datum und Zeit als %x %X
%C	Datum und Zeit in langem Format
%d	Monatstag (01-31)
%D	Datum als %m/%d/%y
%e	Monatstag (1 – 31)
%H	Stunde (00 – 23)
%I	Stunde (00 - 12)
%j	Tagesnummer im Jahr (001 – 366)
%k	Stunde (0 – 23)
%l	Stunde (0 – 12)
%m	Monat (01 - 12)
%M	Minute (00 – 59)
%p	AM / PM
%r	Zeit als %I:%M:%S %p
%R	Zeit als %H:%M
%S	Sekunde (00 – 59)
%T	Zeit als %H:%M:%S
%U	Wochennummer im Jahr (01 - 53), gemäss US Standard, Sonntag ist der erste Tag in der Woche

%w	Wochentag (0 - 6), Sonntag ist 0
%W	Wochennummer im Jahr (01 - 53), gemäss ISO-8601 Standard, Montag ist der erste Tag in der Woche[4]
%x	Datum, lokales Format
%X	Zeit, lokales Format
%y	Jahr (00 - 99)
%Y	Jahr inkl. Jahrhundert
%Z	Abkürzung der lokalen Zeitzone

Mit der Option **FIXED** kann die gewünschte Anzahl der Nachkommastellen eingestellt werden. **FIXED 0** unterdrückt alle Nachkommastellen. Der Feldinhalt wird nur als Nummer interpretiert und formatiert, falls er nicht mit einem Buchstaben beginnt. Zwischen den Tausendern wird ein Hochkomma eingefügt. Um das Hochkomma zu unterdrücken kann die Option **NOQUOTE** eingeschaltet werden. Bei Angabe der Option **COMMA** wird anstelle des Dezimalpunktes ein Dezimalkomma ausgegeben.

```
FIELD Percent (*,, 0.4,) RIGHT FIXED1 PICTURE `@ %';
```

Mit der Option **PICTURE** kann der Feldinhalt mit einem Präfix oder Suffix versehen werden. Anstelle des **@** Zeichens wird der aktuelle Feldinhalt eingesetzt, alle anderen Zeichen werden 1:1 übernommen.

```
FIELD BarCode (4.6,0.05,1.0,0.3) BARCODE_25 0.01;
```

Enthält ein Feld die Option **NUMPAGES**, so wird die verzögerte Seitenausgabe aktiviert, was eine deutliche Erhöhung des Speicherverbrauchs mit sich ziehen kann. Nachdem alle Seiten generiert wurden, wird der Feldinhalt unmittelbar direkt vor der Ausgabe, auf die Anzahl Seiten des generierten Dokumentes gesetzt. Aus diesem Grunde können mit dem Feldinhalt keine Verarbeitungen durchgeführt werden.

Wird ein Feld mit der Option **BARCODE_25** versehen, so wird der Feldinhalt als Barcode (2/5 Interleaved) dargestellt. Mit der folgenden Zahl kann die Strichdicke (in inch) eingestellt werden. Die Feldhöhe bestimmt die Höhe des Barcodes, die Breite des Barcodes ist eine Funktion der Strichdicke und der Länge der dargestellten Zahl. Es können nur ganzzahlige numerische Werte mit gerader Anzahl Ziffern als Barcode dargestellt

4 Die Wochennummer %W wird gemäss ISO-8601berechnet. Dabei kann der 01.01. des Jahres in die KW 52/53 des Vorjahres fallen. Damit die Jahresverschiebung korrekt dargestellt werden kann, werden die Zeitangaben durch den Einsatz der Formatierungsangabe %W für den Rest der Formatierung angepasst, dahingehend dass mit %W/%Y eine korrekte Ausgabe erfolgt. Nach dem %W darf nur noch %Y oder %y genutzt werden, andere Angaben wie %A,%a,%d liefern falsche Ausgabe.

werden (Limitation von 2/5 Interleaved).

Weiter Barcode-Optionen sind **BARCODE_39** (Code 30), **BARCODE_EAN** (EAN-13, bzw EAN-8), **BARCODE_128** (Code-128 / GS1-128). Ausschlaggebend für die Unterscheidung der EAN-Codes 8 und 13 ist die Anzahl Zeichen im Datenfeld. Wenn die Länge des Datenfeldes 12 bzw. 7 Zeichen ist, so wird automatisch eine Prüfziffer berechnet und ausgegeben. Um aus dem Code-128 einen GS1-128 zu machen, muss das Spezialzeichen `0xC1' (FNC1) vorangestellt werden. Dasselbe Zeichen ist als Trennzeichen bei variabler Feldlänge im GS1-128 einzusetzen.

BARCODE_QR[5] gestattet die Erzeugung von QR-Codes.

- Der Fehlerkorrektur-Level kann mit den Optionen **QR_ECLEVEL_L** (Default), **QR_ECLEVEL_M**, **QR_ECLEVEL_Q**, **QR_ECLEVEL_H** eingestellt werden.
- Der Modus mit den Optionen **QR_MODE_NUM, QR_MODE_AN, QR_MODE_8** (Default), **QR_MODE_KANJI,QR_MODE_STRUCTURE**, **QR_MODE_ECI**, **QR_MODE_FNC1FIRST**, **QR_MODE_FNC1SECOND,**
- Die Case-Sensitivity kann mit den Optionen **QR_CASE_SENSITIVE** (Default) und **QR_CASE_INSENSITIVE**. eingestellt werden.
- Die Version kann mit **QR_VERSION(V)** eingestellt werden, wobei V eine ganze Zahl von [0..40] ist. Default ist V 0 womit die Library die nötige Version automatisch bestimmt.
- Der Zeichensatz für die Codierung des Barcode-Inhaltes ist per Default ISO-8859-1. Er kann mit der Option **QR_CHARSET_UTF8** auf UTF-8 umgeschaltet werden.

BARCODE_DMTX[6] gestattet die Erzeugung von Datamatrix-Codes.

- Das Datamatrix-Schema kann mit den Optionen **DMTX_SCHEME_ASCII**, **DMTX_SCHEME_C40**, **DMTX_SCHEME_TEXT** (Default), **DMTX_SCHEME_X12**, **DMTX_SCHEME_EDIFACT**, **DMTX_SCHEME_BASE256** eingestellt werden.

5 Der QR-Code ist nur verfügbar in der UTF-8 Version von Oregator. Die Feldoptionen werden in der ISO-8859-1 Version von Oregator ignoriert.

6 Der Datamatrix-Code ist nur verfügbar in der UTF-8 Version von Oregator. Die Feldoptionen werden in der ISO-8859-1 Version von Oregator ignoriert.

- Für die Generierung von GS1-Datamatrix Barcodes muss via Option **DMTX_GS1** die Interpretation des **FNC1_SYMBOL**'s eingeschaltet werden. Jedes **FNC1_SYMBOL** im Feldinhalt wird so in der Ausgabe durch die korrekte Repräsentation des FNC1-Symbols ersetzt. Damit aus dem Datamatrix tatsächlich ein GS1-Datamatrix wird, muss man trotzdem das FNC1-Zeichen im Feldinhalt voranstellen.

Allgemeine Barcode-Optionen:

```
FIELD BarCode (4.6,0.05,1.0,0.3) BARCODE_25 0.01
  BARCODE_RATIO 1:3 FONT ``;
```

Durch die Angabe der **BARCODE_RATIO s:b** kann das Strichdickenverhältnis schmal:breit kontrolliert werden. Eine **BARCODE_RATIO 0:0** setzt das Strichdickenverhältnis wieder auf den Defaultwert des betreffenden Codes zurück. Das Verhältnis lässt sich für alle 2D-Codes mit Ausnahme von **BARCODE_EAN** einstellen, bei letzterem wird nur die Angabe für schmal berücksichtigt und die anderen Strichbreiten daraus hergeleitet. Mit der Option **BARCODE_ROTATE (0-3)** kann der Barcode in 90-Grad Schritten (ccw) gedreht werden. Die Barcode-Legende in Klartext kann durch die Wahl eines leeren Font-Strings **FONT ``** unterdrückt werden.

Der Barcode kann analog zu einem normalen Feld linksbündig, zentriert oder rechtsbündig ausgerichtet werden. Eine Beschriftung des Barcodes in Klartext muss mit Hilfe eines zweiten Feldes geschehen. Der Inhalt des zweiten Feldes kann mit einer einfachen Zuweisung im verarbeitenden **BATCH** geschehen.

Mit Hilfe der beiden Optionen **CLIP_BLOCK** und **CLIP_AREA** können **LINE** und **BOX** Objekte an den Grenzen des umgebenden **BLOCK** oder **AREA** Strukturen abgeschnitten (geklippt) werden. Diese Optionen sind besonders interessant, wenn sich die Grösse und Position des Objekts erst im Laufe der Formatierung ergibt.

RGB(r,g,b)

Die **RGB** -Option kontrolliert die Ausgabefarbe in den Komponenten rot, grün und blau. Die angegebenen Werte müssen sich im Bereich [0..1] liegen. **RGB(1,1,1)** ist weiss und **RGB(0,0,0)** ist schwarz.

KEEP

In der Regel werden die Feldinhalte nach dem Abarbeiten einer

vollen Seite weggeworfen. Durch das Schlüsselwort **KEEP** wird das System dazu veranlasst, alle Instanzen eines Feldinhaltes während des gesamten Report-Ablaufes für Berechnungszwecke aufzubewahren.

NOTE

Durch die Angabe der Feldoption **NOTE** wird im PDF und PostScript-Ausgabeformat eine Annotation mit Feldname / Inhalt erzeugt. Das Feld darf zwar unsichtbar sein, es muss aber mit dem Befehl **STROKE** sichtbar gemacht werden, damit die Annotation ausgegeben wird.

HYPERLINK

Markiert ein **FIELD** als anklickbare Web-URL.

BOOKMARK n

Wird ein sichtbares Feld (**FIELD**) für die Ausgabe in PDF mit der Option **BOOKMARK n** versehen, so erzeugt Oregator jedesmal wenn der betreffende Block aufs Papier gebracht wird einen PDF Bookmark auf die betreffende Seite, mit dem Text des Feldinhaltes. Die Angabe n (0,1,2,3...) steuert die Verschachtelungsebene auf welcher der Bookmark angelegt wird. Ein **BOOKMARK 0** wird stets auf Dokumentenebene angelegt. Ein darauf folgender **BOOKMARK 1** erzeugt Untereinträge letzten Level 0 Bookmark. Das Bookmark-Feld muss sichbaren Text enthalten.

Im Folgenden sehen Sie ein Beispiel für eine vollständige, gültige Blockdefinition:

```
BLOCK Body FORMAT Text IN Body (,,,0.1)
  LINE LL (0,0.1,0,) BORDER 0;
  LINE Ll (1.05,,,) BORDER 0;
  LINE LR (10.5,,,) BORDER 0;

  FIELD Distance (0.1,0.1,0.6,0.1);
  INCLUDE 'flds.def'

ENDBLOCK

BLOCK Body FORMAT Default IN UpperBody (,,,0.13)
  LINE LL (0,0,0,) BORDER 0;
  LINE Ll (1.05,0,,) BORDER 0;
  LINE LR (10.5,,,) BORDER 0;

  FIELD Distance (0.1,0.1,0.6,0.115);
  INCLUDE 'flds.def'
```

```
ENDBLOCK;
```

Die eigentliche Felddefinition befindet sich in der Datei `flds.def':

```
FIELD PrID (,, 0.6,) FONT 'Times-Bold' UNIQ;
FIELD PL (*,, 0.2,) FONT 'Times-Roman' UNIQ;
FIELD CoID (*,, 0.2,) UNIQ;
FIELD RpDate (*,, 0.4,) RIGHT DATETIME '%e.%m.%y';
FIELD Description (*,, 2.0,);
FIELD Hours (*,, 0.4,) RIGHT FIXED 2;
FIELD HrCurr (*,, 0.2,) RIGHT UNIQ;
FIELD HrRate (*,, 0.4,) RIGHT;
FIELD Km (*,, 0.4,) RIGHT;
FIELD KmCurr (*,, 0.2,);
FIELD KmRate (*,, 0.4,) RIGHT;
FIELD ExKind (+0.5,, 0.2,);
FIELD ExCurr (*,, 0.2,);
FIELD ExAmount (*,, 0.4,) RIGHT;
FIELD Fees_C (*,, 0.6,) RIGHT;
FIELD Fees_S (*,, 0.6,) RIGHT;
FIELD Expenses (*,, 0.6,) RIGHT;
FIELD Bill (*,, 0.3,) RIGHT;
FIELD Vis (*,, 0.2,) RIGHT;
```

Die eigentliche Felddefinition ist in einer **INCLUDE** -Datei abgelegt, und braucht auf diese Weise nur einmal geschrieben zu werden. Die Propagation der Defaultwerte für Position und Feldgrösse zusammen mit dem * -Operator sorgen dafür, dass dieselbe Felddefinition für verschiedene Formate eingesetzt werden kann. Das Feld namens **Distance** wird lediglich dazu benötigt, die Schriftgrösse für alle Felder gleichzeitig festzulegen.

3.3 Verarbeitungsbeschreibung

Durch die Verarbeitungsbeschreibung wird bestimmt, wie der strukturierte Datenbankauszug in die vorher beschriebene Darstellung umzusetzen ist. Sie besteht aus einer strukturierten Menge von Prozeduren, welche sequentiell abgearbeitet werden. Die verschiedenen Prozeduren werden im Laufe des Reports zu unterschiedlichen Zeitpunkten aufgerufen. Die Ausführungsstruktur wird dabei unter anderem durch die zu formatierenden Daten beeinflusst (Seitenumbruch etc.).

REPORT

Mit dem Schlüsselwort **REPORT** werden Prozeduren definiert, welche ganz am Anfang/Ende des gesamten Datenauszuges

ausgeführt werden. Beide Prozeduren (**HEADER** und **FOOTER**) sind optional.

```
REPORT
 HEADER
      ...Anweisungen...
 END;
 FOOTER
      ...Anweisungen...
 END;
END;
```

PAGE

Mit dem Schlüsselwort **PAGE** werden Prozeduren definiert, welche jeweils bei Beginn einer neuen Seite (**HEADER**) oder direkt vor dem Auswurf einer neuen Seite (**FOOTER**) ausgeführt werden. Beide Prozeduren sind optional.

```
PAGE
 HEADER
      ...Anweisungen...
 END;
 FOOTER
      ...Anweisungen...
 END;
END;
```

BATCH

Im Anschluss an die beiden Prozeduren **REPORT** und **PAGE** sollte für jedes SELECT-Statement ein e **BATCH** -Prozedur definiert werden, welche die Verarbeitung des Resultates übernimmt.

```
BATCH
 HEADER
      ...Anweisungen...
 END;
 FOOTER
      ...Anweisungen...
 END;
 RECORD
      ...Anweisungen...
 END;
 COMPUTE
   HEADER
      ...Anweisungen...
   END;
```

```
      FOOTER
         ...Anweisungen...
      END;
   END;
  BREAK ON FieldList
      HEADER
         ...Anweisungen...
      END;
      FOOTER
         ......
      END;
   END;
  END;
```

Sämtliche Prozeduren der **BATCH** Deklaration sind wiederum optional.

Die Anweisungen in den **HEADER/FOOTER** Prozeduren werden zu Beginn / am Ende des SQL-SELECTs ausgeführt.

Jeder **BATCH** kann optional mit einem Namen versehen werden. Bei Einschalten der einfachen Debug-Option (-d) werden dann die Namen der ausgeführten **BATCH** es sichtbar.

```
BATCH Adr
      ... Statements ...
ENDBATCH;

BATCH Position
      ... Statements ...
ENDBATCH;
```

Die **BATCH** -Namen müssen eindeutig vergeben werden. Die Ausführung eines bestimmten Folge- **BATCH** es kann durch SQL-Kontrolle aufgrund des Namens bestimmt werden. Siehe hierzu die Beschreibung des **INPUT BATCH** Statements.

RECORD

Die Anweis ungen der **RECORD** Prozedur werden für jeden selektierten Datensatz ausgeführt.

COMPUTE

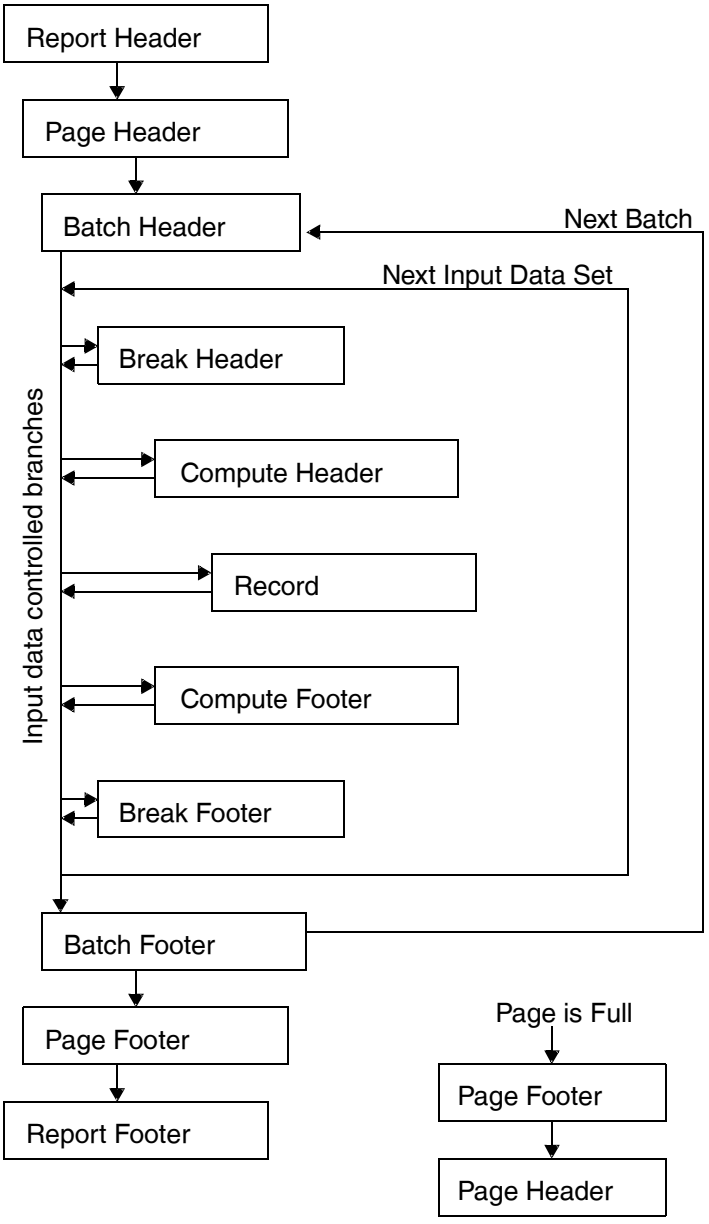
Die **COMPUTE** -Prozedur sollte nur definiert werden, wenn die zugehörige SQL-SELECT Anweisung eine COMPUTE-Klausel enthält. Für jede COMPUTE-Klausel sollte eine **COMPUTE** -Prozedur definiert werden. Die Anweisungen der **COMPUTE** -Prozedur werden ausgeführt, zu Beginn / am Ende der SQL-COMPUTE-Berechnung.

BREAK

Die **BREAK** -Prozedur kann definiert werden, um ORDER BY-Breaks zu erzeugen. Ein solches Break bezieht sich stets auf eine Liste von Feldern. Die **BREAK** -Prozedur wird ausgeführt bevor oder nachdem sich mindestens eines der Felder in der Liste ändert. Alle Felder in der Liste müssen in der Form BlockName.FeldName aufgeführt werden.

Das folgende Schema veranschaulicht, in welcher Reihenfolge die oben beschriebenen Prozeduren aufgerufen werden:

Schematischer Aufbau der Kontrollstruktur



Entscheidend dabei ist die Tatsache, dass die Reihenfolge des Aufrufs der verschiedenen Prozeduren in einem **BATCH** durch die Eingabedaten kontrolliert wird. Grundsätzlich wird ein

COMPUTE HEADER beispielsweise vor dem ersten **RECORD** aufgerufen, welcher im COMPUTE-Statement berücksichtigt wird. Der entsprechende **COMPUTE FOOTER** wird nach dem letzten Record aufgerufen, welcher am Compute beteiligt ist. Entsprechendes gilt für die Breaks.

Wenn immer eine Seite (durch alloziieren eines neuen Blocks) gefüllt wird, werden die beiden Prozeduren **PAGE FOOTER** und **PAGE HEADER** ausgeführt. Dies geschieht erst, wenn der neue Block nicht mehr auf der alten Seite Platz hat. Seine Plazierung wird auf die nächste Seite hinausgeschoben. Sollte der Block nirgends Platz finden, so wird eine Fehlermeldung erzeugt.

3.4 Anweisungen

NEW

Durch die Anweisung **NEW** wird eine neue Instanz eines Blocks erzeugt. Die neue Instanz steht nun zur Bearbeitung zur Verfügung. Bevor ein Feld mit einem Wert versehen werden kann oder ein Objekt gezeichnet wird, muss der entsprechende Block erzeugt werden.

```
NEW BlockName;
NEW Body;
NEW Head;
```

Sollte der Block einem dynamischen Bereich angehören, so wird der Block positioniert: die nächste freie Druckposition wird berechnet und der Block erscheint auf dem Papier. Findet der Block nicht mehr auf der laufenden Seite Platz, so wird automatisch ein Seitenumbruch erzeugt, bevor der Block plaziert wird (siehe **PAGE**).

Durch die Block-Option **DEFERRED** , kann die Positionierung von der Erzeugung getrennt werden See Durch die Angabe von FIT_VERTICAL oder FIT_HORIZONTAL wird die Grösse des Blocks im Moment der Positionierung in der entsprechenden Richtung an seinen Inhalt angepasst. Diese Optionen sind in der Regel nur in Kombination mit DEFERRED Positionierung interessant, darum wird automatisch. DEFERRED Mode eingeschaltet..

INPUT

Im nächsten Schritt können die Datenfelder eines Blocks mit Werten versehen werden. Durch die Anweisung **INPUT** wird der Block mit Daten aus der Datenquelle aufgefüllt.

```
INPUT BlockName;
INPUT BlockName.FeldName,...
INPUT Body.RpDate, Body.Hours, Body.Description, Head;
```

Die **INPUT** Anweisung wird gefolgt von einer Liste von Block- oder Feldnamen (durch Komma getrennt). Wird ein Block-Name angegeben, so versucht Oregator entsprechende Felder der Datenquelle (solche, die denselben Namen tragen wie die Felder des Blocks) mit Daten aufzufüllen. Felder, welche keinen entsprechenden Namen haben, werden ignoriert (bzw. nicht verändert).

Wird die **INPUT** Anweisung in einer **COMPUTE** Prozedur eingesetzt, so müssen alle aufzufüllenden Felder in der richtigen Reihenfolge eingesetzt werden. Block-Namen sind in diesem Falle nicht gestattet.

INPUT BATCH

Normalerweise werden die **BATCH** es in sequentieller Reihenfolge abgearbeitet, wie sie in der Report-Definition eingetragen sind. Für jedes SQL SELECT Statement wird der nächste **BATCH** ausgeführt. Die Anweisung **INPUT BATCH** gestattet dem Anwender eine flexiblere Kontrolle der Abarbeitungs-Reihenfolge aus SQL. Betrachten Sie hierzu den folgenden SQL-Abschnitt:

```
WHILE (select count(*) from #Adressen) > 0
BEGIN
      SELECT @Nr = min(AdrNr) from #Adressen

      SELECT BATCH='Adr'
      SELECT * from #Adresse where AdrNr = @Nr
      SELECT * from Position where AdrNr = @Nr

      DELETE #Adressen where AdrNr = @Nr
END
```

Dieser Code erzeugt eine (datengesteuerte) variable Anzahl von SELECTs. Die entsprechende Report-Definition könnte etwa wiefolgt aussehen.

```
BATCH             # Batch-Name einlesen
  RECORD
   INPUT BATCH;
  END:
ENDBATCH;

BATCH Adr
  RECORD
   NEW Adresse;
```

```
    INPUT Adresse;
    STROKE Adresse;
   END:
ENDBATCH;

BATCH Pos
  RECORD
    NEW Position;
    INPUT Position;
    STROKE Position;
   END:
ENDBATCH;

BATCH             # Batch-Name einlesen
  RECORD
    INPUT BATCH;
   END:
ENDBATCH;
```

Durch das **INPUT BATCH** Statement wird der Name des nächsten auszuführenden **BATCH** bestimmt. Per Default wird sequentiell abgearbeitet, wenn der Name des nächsten **BATCH** nicht gesetzt wird. Die Ausführung des Reports terminiert mit Fehlermeldung, wenn der selektierte Name nicht existiert.

STROKE

Ist der Block erzeugt worden und enthalten die Felder Werte, so können diese mit Hilfe der Anweisung **STROKE** auf dem Papier ausgegeben werden.

```
STROKE ObjektListe;
STROKE BlockName,...;
STROKE BlockName.FeldName,...
STROKE Body.RpDate, Body.Hours, Body.Description,
Head;
STROKE Head.Line, Head.Box,...
```

Die Objekte **LINE** , **BOX** und **IMAGE** werden immer gedruckt. Ein **FIELD** -Objekt hingegen wird nur gedruckt, wenn es einen Wert enthält und als druckbares Feld mit einer Positionsangabe deklariert wurde.

PAGE

Die folgende Anweisung bewirkt einen Seitenumbruch:

```
PAGE;
```

Durch das Auslösen eines Seitenumbruches wird zunächst der

PAGE FOOTER ausgeführt, die Seite wird gedruckt und auf der neuen Seite wird der **PAGE HEADER** ausgeführt.

ABORT

Manchmal entscheiden erst die via SQL selektierten Daten, ob eine Seite ausgegeben werden soll.

Mit dem Schlüsselwort **ABORT** kann die Verarbeitung der aktuellen Seite abgebrochen werden, ohne dass die Seite ausgegeben wird. Es wird ein Seitenumbruch erzeugt, die Seitennummer wird aber nicht inkrementiert. Der **PAGE FOOTER** wird nicht mehr ausgeführt.

CLEAR_PAGE

Mit dem Schlüssel wort **CLEAR_PAGE** wird die aktuelle Seite geleert. Es werden ALLE Objekte (inkl. Header, Footer, Inherit, etc) aus der aktuellen Seite entfernt. Es bleibt eine vollständig leere Seite übrig.

Möchte man pro Datensatz eine eigene Seite drucken, so kann nach jedem **RECORD** mit Hilfe von **PAGE** elegant eine Seite ausgegeben werden. Nach dem letzten Datensatz wird dann jedoch eine leere Seite ausgedruckt. Diese Leerseite enthält lediglich den **PAGE HEADER** . Die Leerseite kann, wie das folgende Beispiel zeigt, im **BATCH FOOTER** mit **CLEAR_PAGE** entfernt werden. Die vollständig leere Seite wird auf dem Drucker nicht ausgegeben.

```
BATCH
  RECORD
   NEW Rechnung;
   INPUT Rechnung;
   STROKE Rechnung;
   PAGE;
  END;
  FOOTER
   CLEAR_PAGE;
  END;
END;
```

Mit Hilfe von **CLEAR_PAGE** können unter anderem alle **INHERIT BLOCK** s durch folgende Anweisungsfolge entfernt werden:

```
CLEAR_PAGE;
ABORT;
```

Die vollständig leere Seite wird dann durch das **ABORT** Statement frisch mit dem **PAGE HEADER** initialisiert.

LOG

Ausgabe einer Meldung im Log (stderr):

```
LOG(aexpr);
```

DEBUG

Mit der folgenden Anweisung kann der Debug-Level in der Reportdefinition verändert werden. Diese Anweisung kann nur im Kopfbereich der Darstellungsbeschreibung vor den FORMAT-Angaben angegeben werden. Siehe -d Option im Abschnitt Allgemeine Optionen.

```
DEBUG Integer;
```

CHARSET

Die Anweisung legt den Zeichensatz für String-Konstanten in der Reportdefinition fest. Diese Anweisung kann nur im Kopfbereich der Darstellungsbeschreibung vor den FORMAT-Angaben angegeben werden. Default ISO-8859-1.

```
CHARSET 'utf-8';
```

Zuweisung

Die interessanteste Anweisung ist die Zuweisung (**=**). Auf der linken Seite steht ein Feld welches einem Block angehört. Auf der rechten Seite steht ein arithmetischer Ausdruck.

```
BlockName.FeldName = aexp;
```

Eine andere Form der Zuweisung gestattet es, den Feldnamen zu berechnen oder zu selektieren:

```
ASSIGN(`BlockName.FeldName', aexp );
ASSIGN( aexp , aexp );
```

Artithmetische Ausdrücke

Ein arithmetischer Ausdruck (***aexp***) besteht aus zwei Operanden, die durch eine arithmetische Operation verknüpft werden. Die folgenden Operationen sind definiert:

A + B	**Addition**
A - B	**Subtraktion**
A & B	**String - Verkettung**
A * B	**Multiplikation**
A / B	**Division**

Als Operanden sind die folgenden Grundelemente erlaubt:

BlockName.FeldName
Datenfeld (letzte Instanz)

BlockName.FeldName [PAGENR, Index]
Datenfeld (bestimmte Instanz)

BlockName.FeldName [SeitenNummer, Index]
Datenfeld (bestimmte Instanz)

`String'	Zeichenkette
Integer	Ganze Zahl
Float	Fliesskommazahl
PAGENR	Aktuelle Seitennummer

DATETIME(`DateFormat')
Aktuelles Datum, siehe unten

COUNT(BlockName.FeldName)
Anzahl Instanzen auf aktueller Seite

SUM(BlockName.FeldName)
Summe aller Instanzen dieser Seite

ABS(aexp)
Mathematischer Absolutwert

ROUND(aexp, aexp)
Rundet den ersten Ausdruck auf
die im zweiten Ausdruck angegebene Genauigkeit

LZERO(aexp, aexp)
Füllt den ersten Ausdruck mit führenden Nullen auf, bis auf die im zweiten Ausdruck angegebene maximale Länge

STRSTR(String, Muster) String-Suche:
Wenn das Muster im String vorkommt, so wird die Position(> 0)zurückgegeben. Kommt das Muster nicht im String vor, so wird 0 zurückgegeben.

STRSUB(String, Muster, Ersatz) String-Replacement:
Wenn das Muster im String vorkommt, so wird das erste

Vorkommnis von links durch den Ersatz ersetzt.

LTRIM(str)
schneidet links Leerzeichen ab

RTRIM(str)
schneidet rechts Leerzeichen ab

LENGTH(str)
Länge der Zeichenkette

SUBSTR(str, idx, len) Substring:
str: Zeichenkette
idx: Startposition(0=erstes Zeichen)
len: max. Länge des Resultats

ELEMENT(str, seps, idx) Extrahiere Listenelement:
str: separierte Liste, z.B. CSV
seps: Separatoren(mehrere möglich)
idx: Listenindex(1=erstes Element)

NEWLINE
Gibt einen Zeilenumbruch als Zeichenkette zurück.

EXISTS(`Block.Feld')
Gibt 1 zurück wenn das Objekt EXISTS(`Block.Feld') deklariert wurde, 0 sonst.

Block.Feld
Inhalt des Feldes

FETCH(`Block.Feld')
Inhalt des Feldes, wobei Feldname berechnet werden kann.

FOLD(`Block.Feld')
Formatierter Inhalt des Feldes. Dabei wird die Zahlenformatierung FIXCOMMA, FIXED, NOQUOTE, COMMA und PICTURE des Ursprungsfeldes angewandt, auch wenn dieses unsichtbar ist.

CONTROL(Integer)
Drucker-Steuersequenz, funktioniert nur bei Text-Output

EANPZ(String)	EAN-Prüfziffer
FORMAT	Aktueller Formatname (siehe -f option)

REPORT	Aktueller Reportname
(aexp)	Arithmetischer Ausdruck in Klammer

Die Funktion **CONTROL** gibt eine nummerierte Steuersequenz zurück See Die verschiedenen Sequenzen werden vor/nach der genannten Einheit gesendet. Es sind folgende Kontroll-Sequenzen zulässig:. Die Sequenzen sind von 0 an aufsteigend nummeriert. Sequenz Nr. 6 ist die erste frei verfügbare Sequenz.

Die Funktion **DATETIME** gibt das aktuelle Datum/Uhrzeit gemäss Spezifikation im ***DateFormat*** zurück. Für die Format-Spezifikation, See Datumsformatierung.

Mit einer Zuweisung können auch **CONTROL** -Sequenzen umgesetzt werden:

```
BLOCK Ctrl DEFERRED
  FIELD BoR;
  FIELD EoR;
  FIELD BoP;
  FIELD EoP;
  FIELD BoL;
  FIELD EoL;
ENDBLOCK;

BATCH SetupControl # Selektion der Escapesequenzen
  HEADER
   NEW Ctrl;
  END;
  RECORD
  INPUT Ctrl;
   CONTROL(0) = Ctrl.BoR;
   CONTROL(1) = Ctrl.EoR;
   CONTROL(2) = Ctrl.BoP;
   CONTROL(3) = Ctrl.EoP;
   CONTROL(4) = Ctrl.BoL;
   CONTROL(5) = Ctrl.EoL;
  END;
ENDBATCH;
```

Auf diese Weise können **CONTROL** -Sequenzen aus der Datenbank selektiert werden. Dabei ist zu beachten, dass die **CONTROL** -Sequenzen erst bei der Ausgabe der aktuellen Seite ausgelesen werden, und so die gesamte laufende Seite mit den neuen **CONTROL** -Sequenzen erzeugt wird. Die Sequenz **CONTROL(0)** kann auf diese Weise nicht gesetzt werden, da diese direkt beim Start von Oregator ausgegeben wird.

Instanzen

Das System unterhält zwei Speicher für Datenfelder. In den ersten Speicher (Primärspeicher) werden alle erzeugten Datenfelder der aktuellen Seite eingetragen. Nach dem Druck der aktuellen Seite werden die Datenfelder welche mit der Option **KEEP** deklariert wurden in den zweiten Speicher (Sekundärspeicher) übertragen. Der gesamte Inhalt des Primärspeichers wird weggeworfen.

In der Regel erscheint ein Datenfeld mehrfach auf derselben Seite. Auf diese Weise entstehen mehrere Instanzen des Datenfeldes auf derselben Seite. Ein Zugriff der Form ***BlockName. FeldName*** gibt die letzte Instanz auf der aktuellen Seite zurück. Durch Angabe von eckigen Klammern können die Instanzen indiziert werden:

BlockName.FeldName
BlockName.FeldName*[* **Index** *]*
BlockName.FeldName*[* **SeitenNummer, Index** *]*

BlockName.FeldName*[* **-** *]*
BlockName.FeldName*[* **1,** *]*
BlockName.FeldName*[* **-,** *]*
BlockName.FeldName*[* **,1** *]*

Der ***Index*** selektiert die angesprochene Instanz eines Objekts auf einer Seite. Als ***Index*** kann eine ganze Zahl (>0) eingesetzt werden. Wird der Wert **-** (Minuszeichen) eingesetzt, so wird die zweitletzte Instanz eingesetzt. Der ***Index*** kann auch weggelassen werden, dann wird die letzte Instanz eingesetzt. Die Angabe von Index 0 setzt ebenfalls die letzte Instanz ein.

Als ***SeitenNummer*** kann eine ganze Zahl (>0) eingesetzt werden. Die Numerierung der Seiten beginnt mit dem Wert 1. Wird die ***SeitenNummer*** leer gelassen, so wird die aktuelle Seite gewählt. Wird als ***SeitenNummer*** der Wert **-** (Minuszeichen) eingesetzt, so wird die vorhergehende Seite (aktuelle Seite - 1) gewählt.

Tabelle 3: Indexierung von Feldern

Seite\Index	**[Seite,1..n]**	**[Seite,]**	**[Seite,-]**
[1..n,Index]	Best. Seite Best. Instanz	Best. Seite Letzte Instanz	Best. Seite Vorletzte Instanz
[PAGENR,Index]	Aktuelle Seite Best. Instanz	Aktuelle Seite Letzte Instanz	Aktuelle Seite Vorletzte

			Instanz
[-,Index]	Vorletzte Seite Best. Instanz	Vorletzte Seite Letzte Instanz	Vorletzte Seite Vorletzte Instanz
[*]	Irgendeine Seite Letzte Instanz überhaupt		

IF

ACHTUNG: Auf die Instanzen einer vorhergehenden Seite kann nur zugegriffen werden, wenn das entsprechende Feld mit der **KEEP** Option versehen wurde.

Für die beschränkte Kontrolle der Ausführung stehen die folgenden Struktur en zur Verfügung:

```
IF ( Logischer Ausdruck ) THEN
... Anweisungen ...
ENDIF;

IF ( Logischer Ausdruck ) THEN
... Anweisungen ...
ELSE
... Anweisungen ...
ENDIF;
```

Ein ***Logischer Ausdruck*** besteht entweder aus einer Vergleichsoperation mit zwei arithmetischen Ausdrücken:

```
A > B           A ist grösser als B
A < B           A ist kleiner als B
A = B           A ist gleich B
A != B          A ist verschieden von B
A &= B          Vergleich von Zeichenketten
```

oder einem logischen Operator mit 1-2 geklammerten Ausdrücken:

```
NOT( Logischer Ausdruck )
( Logischer Ausdruck ) AND ( Logischer Ausdruck )
( Logischer Ausdruck ) OR ( Logischer Ausdruck )
```

Das System rechnet grundsätzlich mit Zeichenketten von begrenzter Länge (siehe Anhang). Man beachte, dass beim Zeichenkettenvergleich ein spezieller Operator eingesetzt werden muss, da sonst der Zahlenmässige Wert der Zeichenketten verglichen wird. Die folgenden Beispiele illustrieren diesen Sachverhalt:

```
IF (`abc' = `def') THEN
  ... Falsch: immer erfüllt ...
ENDIF;

IF (123 = `123xyz') THEN
  ... Falsch: auch immer erfüllt ...
ENDIF;

IF (`abc' &= `def') THEN
  ... korrekter Zeichenketten-Vergleich ...
ENDIF;
```

Insbesondere ist zu beachten, dass der Test ob ein Feld den Wert **NULL** enthält, in jedem Falle als Zeichenkettenvergleich zu implementieren ist, da der Wert einer leeren oder ungültigen Zeichenkette 0 ist.

```
IF (A &= `') THEN ...   ; Test auf NULL Wert
```

Für die Auswertung von arithmetischen und logischen gilt die folgende Tabelle für die Auflösung der Rechengenauigkeit:

Tabelle 4: Auflösung der Rechengenauigkeit

Ausdruck	Genauigkeit	Bemerkung
A + B	double float	Volle Auflösung
A – B	double float	Volle Auflösung
A / B	double float	Volle Auflösung
A * B	double float	Volle Auflösung
SUM(A)	double float	Volle Auflösung
ROUND(A, prec)	double float	Volle Auflösung
A = B	integer	Nur Vorkommastellen werden berücksichtigt
A != B	integer	Nur Vorkommastellen werden berücksichtigt
A > B	Integer	Nur Vorkommastellen werden berücksichtigt
A < B	integer	Nur Vorkommastellen werden berücksichtigt
COUNT(A)	integer	
A &= B	string	Zeichenkettenvergleich
A & B	string	Zeichenverkettung
LZERO(A)	string	Zeichenkettenoperation

Möchte man einen Fliesskommavergleich mit bestimmter

Rechengenaugkeit implementieren, so zeigen die folgenden Beispiele, wie vorgegangen werden muss. Die Variable prec wird dabei auf die gewünschte Genauigkeit initialisiert, (z.B. auf 0.001):

```
IF ((B - A) / prec = 0) THEN ... ; ungefähr gleich
IF ((B - A) / prec < 0) THEN ... ; B < A
IF ((B - A) / prec > 0) THEN ... ; A < B
```

OPEN, CLOSE, OUTPUT

Die Befehle **OPEN** , **CLOSE** und **OUTPUT** dienen der Datenausgabe (Journalisierung) während des Report-Ablaufs. Die Befehle werden nur dann ausgeführt, wenn die Journalisierungs-Option (-j -J) beim Start von Oregator angegeben wird.

Mit dem Schlüsselwort **OPEN** wird der Ausgabekanal eröffnet, mit dem Schlüsselwort **CLOSE** kann er wieder geschlossen werden. Dazwischen können Ausgaben mit dem Befehl **OUTPUT** erfolgen.

```
OPEN TabellenName, Block.Feld1, Block.Feld2...
   ... Ausgabe ...
CLOSE
```

Als **TabellenName** kann ein beliebiger arithmetischer Ausdruck eingesetzt werden. Sollte die Tabelle nicht existieren, so wird Sie von Oregator automatisch angelegt. Es folgt eine Liste von Feldern vom Typ **FIELD** oder **TEXT** . Durch die Feldliste wird der Tabellenaufbau beschrieben. Die Feldoptionen sind massgebend, für den Datentyp der erzeugten Tabelle. **FIXCOMMA 0** erzeugt eine Integer-Spalte. **FIXCOMMA >0** erzeuft eine Real-Spalte. **DATETIME(`%D %r')** erzeugt eine Datums-Spalte. Ansonsten wird eine Zeichen-Spalte erzeugt. Die Angabe der Feldbreite (,,w,) bestimmt die Spaltenbreite (auf Basis von 15 CPI).

```
OUTPUT ArithExpr, ArithExpr, ...
```

Durch den **OUTPUT** Befehl wird ein Journalisierungs-Datensatz ausgegeben. Es müssen genauso viele arithmetische Ausdrücke angegeben werden, wie beim **OPEN** angegeben wurden.

Es ist empfehlenswert, für die Journalisierungs-Ausgaben einen **DEFERRED** -Block anzulegen, der die notwendigen Felder enthält.

3.5 Grafik

GRAPHIC

Um eine Grafik aufzubereiten definieren wir einen **BLOCK** der mit der Option **GRAPHIC** versehen wird:

```
BLOCK GfxBlock IN UpperBody (0,0,10,6) GRAPHIC
   ... Felddefinitionen ...
ENDBLOCK;
```

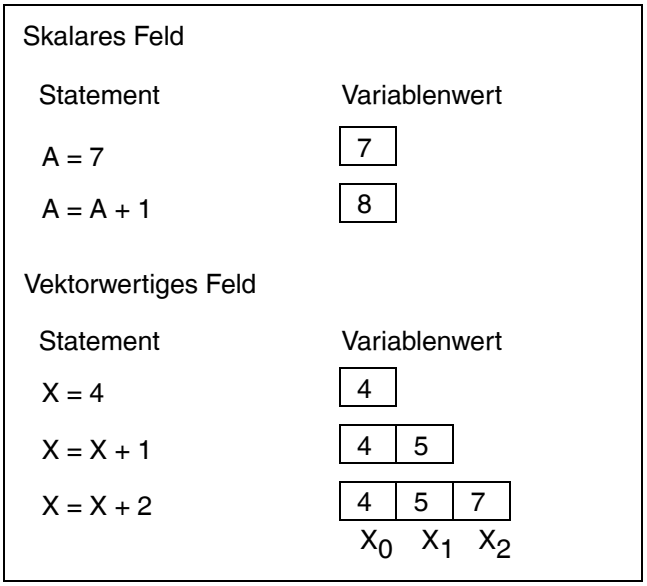
Der Grafik- **BLOCK** verhält sich in Bezug auf die Positionierung wie ein normaler **BLOCK** . Auf einer Seite können somit beliebig viele Grafiken positioniert werden.

Der Grafikblock kann nun auf gewohnte Weise mit Objekten versehen werden. In der Regel verhalten sich die Objekte wie gewohnt. Eine Ausnahme sind die Felder mit der folgenden Bezeichnung:

```
BLOCK GfxBlock IN UpperBody (0,0,10,6) GRAPHIC
  FIELD X (1.5,1, 8.0,4.0);
      Wertevektoren
  FIELD Y0;
  ... bis ...
  FIELD Y9;

  FIELD Style;
      Darstellungsstil

  FIELD XLine;                    X-
Achsenbeschriftung
  FIELD XLabel (,5.2,,0.125) CENTERED;
  FIELD XValue;

  FIELD YLine;                    Y-
Achsenbeschriftung
  FIELD YLabel (0.1,,1.3,0.125) RIGHT;
  FIELD YValue;
ENDBLOCK;
```

Vektoren

Diese Felder sind, wenn Sie in einem **GRAPHIC BLOCK** eingesetzt werden, keine skalaren Felder sondern vektorwertige Felder. Vektorwertige Felder (Vektoren) können beliebig viele Werte enthalten. Nach der Erzeugung des Blocks enthalten alle vektorwertigen Felder keinen Wert (DIM(Vektor) = 0).



Wir bezeichnen die Dimension des Vektors v im folgenden mit DIM(v). Durch jede Zuweisung oder jedes **INPUT** Statement erhöht sich die Dimension eines vektorwertigen Feldes um 1. Der neue Wert wird am Ende des Vektors angehängt.

Mathematisch könnte man den Sachverhalt folgendermassen formulieren: Voraussetzung ist DIM(X) = 0. Durch die Zuweisung

```
NEW GfxBlock;
GfxBlock.X = 33;
```

wird X *0* =33 und DIM(X) = 1 gesetzt. Durch erneute Zuweisung

```
GfxBlock.X = 17;
```

wird X *1* = 17 und DIM(X) = 2 gesetzt. X *0* behält dabei seinen Wert. Dieser Vorgang lässt sich beliebig fortsetzen.

Mit Hilfe eines SQL-Statements kann man einen Vektor auf einfache Weise füllen. Die folgende Tabelle:

```
select X=Value from #Table
```

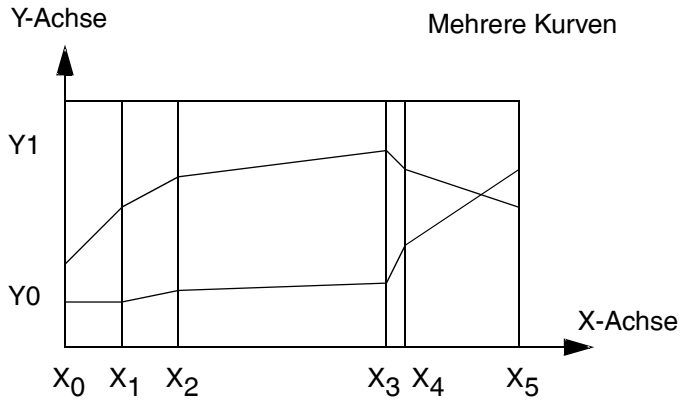
kann folgendermassen in den Vektor **X** eingelesen werden.

```
RECORD
  INPUT X;
END;
```

Die Vektoren haben in der Grafik folgende Bedeutung:

X, Y0, Y1, ...Y9

Die Vektoren **X, Y0, Y1, ... Y9** enthalten die darzustellenden Daten. Der Vektor **X** bestimmt den Wert der X-Achse. Für jede darzustellende Punkt kann ein zugehöriger Y-Wert im Vektor **Y0, Y1, ... Y9** abgelegt werden. Auf diese Weise können bis zu 10 Kurven in einer Grafik gleichzeitig dargestellt werden.



Die folgende Randbedingung muss erfüllt sein: DIM(**X**) = DIM(**Y0**) = DIM(**Y1**) = ... = DIM(**Y9**).

Die Positionsangabe des **X** -Vektors bestimmt dabei die Lage des eigentlichen Grafikbereiches innerhalb des **GRAPHIC BLOCK** s.

Xvalue, Xlabel, XLine

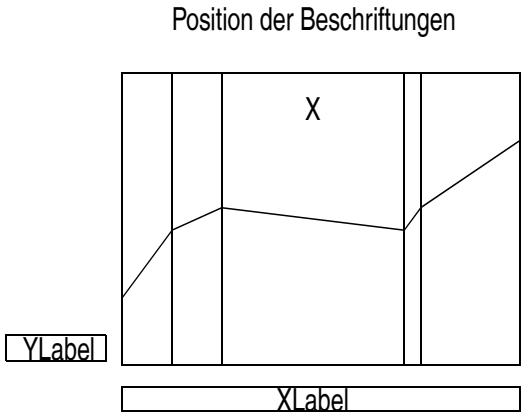
Die Vektoren **XValue** , **XLabel** und **XLine** bestimmen die Beschriftung der X-Achse in der Grafik. **XValue** enthält die X-Koordinate der Beschriftung. **XLabel** enthält den Text der Beschriftung. Ist der Text leer, so wird keine Beschriftung ausgegeben. **XLine** bestimmt den Grauwert [0.0 ... 1.0] der gezeichneten Linie (0 = weiss, 1=schwarz). Durch den Wertebereich von **XValue** [min(**XValue**) ... max(**XValue**)] wird die horizontale Skalierung der Grafik bestimmt. Die folgende Randbedingung muss erfüllt sein: DIM(**XValue**) = DIM(**XLabel**) = DIM(**XLine**) und DIM(**XValue**) > 1.

Die Positionsangabe des **XLabel** Vektors bestimmt dabei die Lage der X-Achsen Beschriftung innerhalb des **GRAPHIC BLOCK** s. Die Formatierungsangabe (**LEFT, CENTERED** etc.) des **XLabel** Vektors bestimmt die Darstellungsweise der Beschriftung relativ zum Referenzpunkt.

Yvalue, Ylabel, YLine

Die Felder **YValue** , **YLabel** und **YLine** bestimmen analog die Beschriftung und Skalierung der Y-Achse. Auch hier gilt die Randbedingung: DIM(**YValue**) = DIM(**YLabel**) = DIM(**YLine**) und DIM(**YValue**) > 1.

Die Positionsangabe des **YLabel** Vektors bestimmt dabei die Lage der Y-Achsen Beschriftung innerhalb des **GRAPHIC BLOCK** s. Die Formatierungsangabe (**LEFT, CENTERED** etc.) des **YLabel** Vektors bestimmt die Darstellungsweise der Beschriftung.



Style

Mit Hilfe des Vektors **Style** kann bestimmt werden, wie die Werte als Grafik dargestellt werden sollen. Für jede Kurve **Y0, Y1, ... Y9** muss in der Initialisierung des Blocks eine Zuweisung gemacht werden.

```
BLOCK Gfx (,,,) GRAPHIC
  FIELD X (1,1,4,3);
  FIELD Y0;
  FIELD Y1;
  FIELD Y2;
ENDBLOCK;

BATCH
  HEADER
   NEW Gfx;
   Gfx.Style = 5;       Darstellung von Y0
   Gfx.Style = 9;       Darstellung von Y1
   Gfx.Style = 25;      Darstellung von Y2
  END;
END;
```

Die folgenden Werte für Style sind zulässig:

Tabelle 5: Graphic Styles

Style	Beschreibung
0 ... 9	Einfache ausgezogene Kurve, Angabe des Grauwertes der Linie
10 ... 19	Ausgezogene Linien mit Füllung bis zum Nullpunkt, Angabe des Grauwertes der Füllung
20 ... 29	Kurve wird als Balken in einem Balkendiagramm dargestellt, Angabe des Grauwertes der Füllung

Bei der Darstellung als Balkendiagramm wird der letzte Wert ignoriert, damit Balken- und Liniendiagramme übereinandergelegt werden können. Das Balkendiagramm geht bei der Bereichseinteilung von äquidistanten X-Werten aus.

Grafikbeispiel

Das folgende Beispiel soll die Erzeugung von Grafiken veranschaulichen:

Die Datenbank-Anfrage liefert im wesentlichen drei Resultate, je eine SQL Anweisung für die X- die Y-Koordinate und für die darzustellenden Werte.

Die graphisch darzustellenden Werte werden zunächst in temporären Tabellen aufbereitet:

```
create table #XAchsis
(  XValue int null,
   XLabel varchar(20) null,
   XLine float
)
create table #YAchsis
(  YValue money null,
   YLabel varchar(20) null
   YLine float
)
create table #Points
(  X int null,
   Y0 real null,
   Y1 real null,
   Y2 money null,
   Y3 money null,
   Y4 money null
)
```

Die temporären Tabellen werden applikationsabhängig mit

Daten gefüllt. Die Resultate werden aus diesen temporären Teilen selektiert.

```
/* X and Y Achsis */
select * from #XAchsis order by XValue
select * from #YAchsis order by YValue

/* the points/bars */
select X, Y0, Y1, Y2, Y3, Y4 from #Points order by X
```

Die Ausgabe der Datenbank in Tabellenform dargestellt:

Tabelle 6: Beschriftung der X-Achse

XValue	XLabel	XLine
1	01.94	0.500000
2	02.94	0.500000
3	03.94	0.500000
4	04.94	0.500000
5	05.94	0.500000
6	05.94	0.500000
7	05.94	0.500000

Tabelle 7: Beschriftung der Y-Achse

YValue	YLabel	YLine
-20000.00	-20000	1.000000
0.00	0	1.000000
70,000.00	70000	1.000000

Tabelle 8: Selektion der Wertevektoren

X	Y0	Y1	Y2	Y3	Y4
1	-190000.00	-160000.00	500000.00	150000.00	0.00
2	-200000.00	-150000.00	600000.00	250000.00	150000.00
3	-180000.00	-140000.00	500000.00	180000.00	400000.00
4	-100000.00	-150000.00	300000.00	50000.00	580000.00
5	-120000.00	-140000.00	250000.00	-10000.00	630000.00
6	-0.00	-120000.00	0.00	-12000.00	620000.00
7	0.00	0.00	0.00	0.00	608000.00

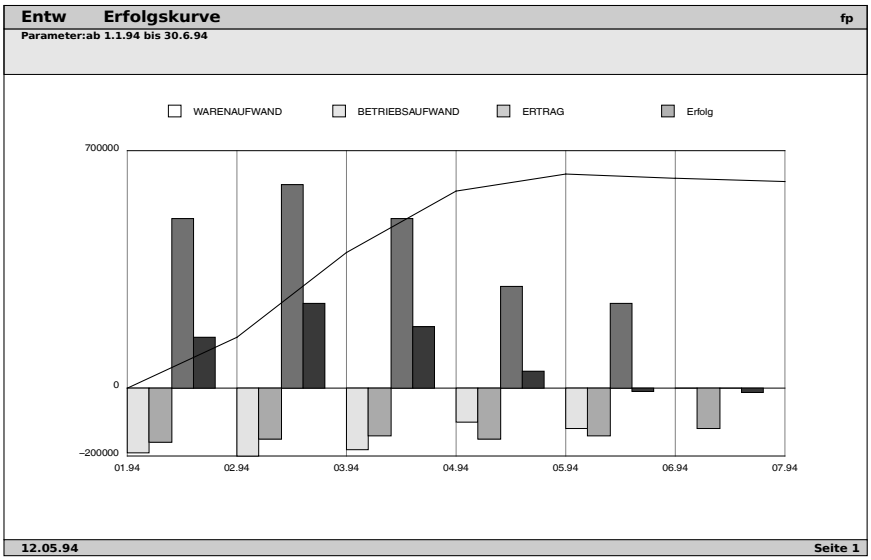
Die vollständige Reportbeschreibung sieht etwa so aus:

```
BLOCK Gfx IN UpperBody (0,0,10,6) GRAPHIC
  BOX B0 (2, 0.4,0.15,0.15) BORDER 0 ;
  FIELD T0 (2.3,0.525,2,0.125) ;
  BOX B1 (4, 0.4,0.15,0.15) LIGHTGRAY BORDER 0 ;
  FIELD T1 (4.3,0.525,2,0.125) ;
```

```
  BOX B2 (6, 0.4,0.15,0.15) GRAY BORDER 0 ;
  FIELD T2 (6.3,0.525,2,0.125) ;
  BOX B3 (8, 0.4,0.15,0.15) DARKGRAY BORDER 0 ;
  FIELD T3 (8.3,0.525,2,0.125) ;

  FIELD X (1.5,1, 8.0,4.0);
  FIELD Y0;
  FIELD Y1;
  FIELD Y2;
  FIELD Y3;
  FIELD Y4;

  FIELD Style;
  FIELD XLine;
  FIELD XLabel (,5.2,,0.125) CENTERED;
  FIELD XValue;

  FIELD YLine;
  FIELD YLabel (0.1,,1.3,0.125) RIGHT;
  FIELD YValue;
ENDBLOCK;

PROCESSING

BATCH
 HEADER
  NEW Gfx;
  Gfx.T0 = 'WARENAUFWAND';
  Gfx.T1 = 'BETRIEBSAUFWAND';
  Gfx.T2 = 'ERTRAG';
  Gfx.T3 = 'Erfolg';

  Gfx.Style = 28;
  Gfx.Style = 26;
  Gfx.Style = 24;
  Gfx.Style = 22;
  Gfx.Style = 0;
 END;
 RECORD
  INPUT Gfx;
 END;
ENDBATCH;

BATCH
 RECORD
  INPUT Gfx;
 END;
ENDBATCH;

BATCH
 RECORD
```

```
  INPUT Gfx;
 END;
ENDBATCH;
```

Diese Reportbeschreibung steht als Beispiel zur Verfügung und kann für ähnliche Graphiken nahezu unmodifiziert verwendet werden. Das Resultat sieht etwa wie folgt aus:



4 Tips & Tricks

4.1 Compute-Resultate

Da jede COMPUTE Aggregation im SQL die berechneten Summen stets hinter den massgebenden Daten liefert, ist es auf den ersten Blick nicht so einfach, die berechneten Summen schon im Datenbereich zu verwenden. z.B. um prozentuale Anteile der Datensätze an den COMPUTE Summen auszugeben.

Die im folgenden beschriebene Vorgehensweise zeigt auf, wie man es anpacken kann.

Doppelte Selektion im SQL

Selektieren Sie im SQL den betreffenden SQL-Batch inkl. COMPUTE Klauseln zweimal, direkt hintereinander. Das erste SELECT dient dazu, die COMPUTE-Resultate einzusammeln. Mit dem zweiten SELECT werden die Daten ausgegeben, wobei man die COMPUTE-Resultate in der Verarbeitung berücksichtigen kann.

```
SELECT Gruppe, Posten, SaldoAP, SaldoVP
from #temp
order by Gruppe, Posten
compute sum(SaldoAP), sum(SaldoVP) by Gruppe
compute sum(SaldoAP), sum(SaldoVP)

SELECT Gruppe, Posten, SaldoAP, SaldoVP
from #temp
order by Gruppe, Posten
compute sum(SaldoAP), sum(SaldoVP) by Gruppe
compute sum(SaldoAP), sum(SaldoVP)
```

Eine Blockdefinition für jeden COMPUTE-Level anlegen

Der Block keine sichtbaren Felder und die Dimensionen (0,0,0,0) damit er die Darstellung nicht verändert. Die Felder sind mit der Option KEEP versehen, damit alle Instanzen im Primär- und Sekundärspeicher aufbewahrt werden.

```
BLOCK ComputeResult1
 IN UpperBody (0,0,0,0) # Level1 Computes
  FIELD SaldoAP KEEP;
  FIELD SaldoVP KEEP;
ENDBLOCK;

BLOCK ComputeResult2
```

```
 IN UpperBody (0,0,0,0) # Level2 Computes
  FIELD SaldoAP KEEP;
  FIELD SaldoVP KEEP;
ENDBLOCK;
```

COMPUTE Resultate aufsammeln

Im Ersten BATCH werden die COMPUTE-Resultat pro Level alle auf der ersten Seite abgespeichert.

```
BATCH
  COMPUTE # by Gruppe
    FOOTER
      NEW ComputeResult1;
      INPUT
        ComputeResult1.SaldoAP,
        ComputeResult1.SaldoVP;
    END;
  END;

  COMPUTE # by ALL
    FOOTER
      NEW ComputeResult2;
      INPUT
        ComputeResult2.SaldoAP,
        ComputeResult2.SaldoVP;
    END;
  END;
ENDBATCH;
```

COMPUTE Resultate auswerten

Im Zweiten BATCH wird mit den COMPUTE-Summen gerechnet.

Dazu sind zunächst Zähler-Variablen pro Level anzulegen.

```
BLOCK Var DEFERRED
  FIELD ComputeCounter1; # Level1 Compute Index
  FIELD ComputeCounter2; # Level2 Compute Index
ENDBLOCK;

NEW Var;
Var.ComputeCounter1 = 0;
Var.ComputeCounter2 = 0;
```

Die Counter werden im HEADER jedes COMPUTE-Levels inkrementiert. Im RECORD werden die Summen verwendet.

```
BATCH
  RECORD
```

```
    ...
    Body.DiffPP = Body.SaldoAP
      / ComputeResult1.SaldoAP[1,Var.ComputeCounter1]
      * 100);
    Body.DiffPK = Body.SaldoVP
      / ComputeResult2.SaldoVP[1,Var.ComputeCounter2]
      * 100);
  END
  …
  COMPUTE
    HEADER
      Var.ComputeCounter1 = Var.ComputeCounter1 + 1;
    END;
  END;
  COMPUTE
    HEADER
      Var.ComputeCounter2 = Var.ComputeCounter2 + 1;
    END;
  END;
  …
ENDBATCH;
```

4.2 Hinweise zum Wert NULL

In Anlehnung der Semantik von ANSI SQL ist der Umgang mit NULL-Werten wiefolgt implementiert:

Falls ein Term in einem arithmetischen Ausdruck NULL ist, so ist das gesamte Resultat des arithmetischen Ausdrucks NULL.

Falls ein Term in einem logischen Ausdruck NULL ist, so ist das Resultat des logische Ausdrucks NULL (undefiniert), was einem logisch FALSE entspricht.

Das Resultat der logischen Funktion NOT(NULL) ist ebenfalls NULL.

Ein leerer STRING wird nicht als NULL dargestellt, sondern als Zeichenkette der Länge 0. Der Wert NULL kann z.B. mit Hilfe der Funktionen LTRIM, RTRIM erzeugt werden.

```
A = '';  # nicht NULL
A = LTRIM('');  # NULL
A = RTRIM('');  # NULL
```

5 FrameMaker

5.1 MIF Musterdateien

In der Report-Definition muss durch die Angabe von **MIF_TEMPLATE** eine geeignete Musterdatei ausgewählt werden:

```
MIF_TEMPLATE 'TemplateName.mif';
```

Bei Weglassen dieser Angabe, wird eine Musterdatei namens Default.mif eingesetzt. Die MIF Musterdateien befinden sich im Verzeichnis:

```
$OREGATOR_PATH/mif
```

Eine MIF Musterdatei kann mit FrameMaker (Version 3.x) erzeugt werden. Beim Abspeichern der Datei (`Speichern unter...') ist die Option MIF anzugeben.

5.2 Regeln für MIF Musterdateien

Eine MIF Musterdatei muss folgenden Anforderungen genügen:

- Eine MIF Musterdatei darf nur Master- und Referenzseiten enthalten.

  Achtung: Beim Laden der MIF Datei wird von FrameMaker automatisch eine Bodypage erzeugt! Aus diesem Grunde ist es ratsam, das Template auch als editierbares Dokument zu sichern.

- Pro **AREA** muss auf der rechten Master-Page (**Right**) ein Textrechteck definiert sein. Der dazugehörende Textfluss muss denselben Namen haben, wie die **AREA** .

- Pro **BLOCK** muss im Absatzkatalog ein Absatzformat definiert werden, dass denselben Namen trägt wie der entsprechende **BLOCK** .

- Ein weiteres Absatzformat mit dem Namen **Null** ist zu definieren. Das Absatzformat muss die minimale Höhe (1 pt) aufweisen. Sämtliche Textflüsse auf der Master-Page müssen im Absatzformat mit **Null** versehen werden.

- Die Master-Page kann mit weiteren Textflüssen oder Graphiken versehen werden.
- EPSF-Grafiken sind als Referenz zu importieren. Sie dürfen nicht ins Dokument kopiert werden.

5.3 Erzeugung von MIF Dateien

Bei der Erzeugung von MIF Dateien wird für jede neue Seite der Ausgabe eine Kopie der Master-Page (Right) angelegt.

Ein **BLOCK** auf dem Ausdruck wird in den entsprechenden Textfluss eingesetzt, der denselben Namen wie die **AREA** des **BLOCKs** trägt. Das Absatzformat wird auf den Namen des **BLOCKs** gesetzt.

Innerhalb eines **BLOCKs** werden alle Felder in denselben Absatz generiert. Zwischen den Feldern wird entweder ein Tabulator (Tab) oder ein Zeilenvorschub (HardReturn) eingesetzt. Der Tabulator wird dann eingesetzt, wenn die Felder vertikal auf derselben Position liegen. Andernfalls wird ein HardReturn eingesetzt.

Die Feld-Positionierungsangaben in der Report-Definition dienen nur der Bestimmung des Feldseparators. Die Grössenangaben für **BLOCKs** werden für die Erzeugung des Seitenumbruches verwendet. Stellen Sie bitte die Grössen von **BLOCK** , **AREA** und **FORMAT** etwa auf die Grössen der entsprechenden FrameMaker Objekte in der MIF Musterdatei ein.

Alle Felder vom Type **BOX** , **LINE** , **BARCODE** oder **IMAGE** werden ignoriert.

6 Windows Export

6.1 Erzeugung von Export-Dateien

In der Report-Definition muss ein Format vom Typ **WINWORD** definiert werden.

Innerhalb eines **BLOCKs** werden alle Felder in denselben Absatz generiert. Zwischen den Feldern wird entweder ein Tabulator (Tab) oder ein Zeilenvorschub (CR/LF) eingesetzt. Der Tabulator wird dann eingesetzt, wenn die Felder vertikal auf derselben Position liegen. Andernfalls wird ein HardReturn eingesetzt.

Alle Felder vom Typ **BOX** , **LINE** , **BARCODE** oder **IMAGE** werden ignoriert.

Die erzeugte Datei baut auf dem ISO-Latin-1 Zeichensatz auf und kann direkt in Windows-Applikationen importiert werden:

- Bei Erzeugung von Serienbrief-Steuersatzdateien für WinWord müssen alle Datenfelder auf derselben Zeile liegen und durch einen Tabulator Tab getrennt sein. Die erste Zeile muss die Feldbezeichnungen enthalten.

- Bei der Erzeugung von Excel Importdateien kann einfach eine Tabelle erzeugt werden, in der die einzelnen Spalten durch einen Tabulator und die Zeilen durch ein CR/LF getrennt sind.

7 Anhang

7.1 Limits und Konstanten

Version V3.01.031

Maximale Länge eines Bezeichners	32
Maximale Länge des String-Puffers	2048
Maximale Anzahl Objekte	3000
Maximale Anzahl Bezeichner:	3000
Arithmetic Stack Tiefe	100
Maximale Anzahl Batches	50
Maximale Anzahl Spalten pro select	100
Maximale Anzahl Computes pro Batch	10
Maximale Anzahl Breaks pro Batch	10
Maximale Include-Verschachtelungstiefe:	10
Maximale Länge von String-Konstanten	256
Anzahl signifikanter Stellen bei Arithmetik	16
Maximale Anzahl CONTROL-Sequenzen	20
Horizontaler Abstand (*) zwischen Spalten	0.1 inch
Vertikaler Abstand (*) zwischen Spalten	0.01 inch
Default Zeichensatz	ISO-Latin-1 / Helvetica
Default Datumsformat	%e.%m.%y %H:%M
Default Lizenzdatei	oregator.lic
Default Bild-Konversionsregeln	oregator.inf
Cairo Bild-Konversionsregeln	oregator_cairo.inf

Default Reportdefinition	default.ry
Default 0/0-Symbol	NaN
Default Unendlich-Symbol	Inf
Default MIF Musterdatei	Default.mif
Environment: Installationspfad	OREGATOR_PATH
Environment: Lizenzdatei	OREGATOR_LICENSE
Environment: Temp-Verzeichnis	OREGATOR_TMP
Bildverzeichnis	$OREGATOR_PATH/images
AFM-Dateien	$OREGATOR_PATH/afm
MIF-Musterdateien	$OREGATOR_PATH/mif

7.2 Format der Antwortdatei

Die Antwortdatei ist eine Textdatei und kann mit einem geeigneten Texteditor verändert werden. Die Zeilen können beliebig lange werden, da pro Zeile ein Datensatz abgelegt wird.

Es folgt eine formale Syntaxbeschreibung für die ***AntwortDatei*** . Terminale Symbole sind in **`Anführungszeichen'** notiert. Nicht-Terminale Symbole in ***Kursivschrift*** .

AntwortDatei = ***Head*** { ***Batch*** } ***Foot***
KommentarZeile = **`/'** { ***Char*** } ***NL***
Head = ***Database DBVersion Exportversion Libversion***
Database = **`DATABASE'** ***Name Integer Integer NL***
Name = ***Identifier***
DBVersion = **`DBVERSION'** ***VersionNr NL***
Exportversion = **`EXPORTVERSION'** ***VersionNr NL***
Libversion = **`LIBVERSION'** ***VersionNr NL***
VersionNr = ***Digit Dot Digit*** [***Dot Digit***]
Foot = **`ENDDATABASE'**

Batch = ***TableHead*** { ***Column*** } ***TableData*** { ***Row*** } ***TableFoot***
TableHead = **`TABLE'** ***TableName TableDescription NL***
TableName = ***Identifier***
TableDescription = ***String***
TableData = **`DATA'** ***NL***

```
TableFoot = `ENDTABLE' NL

Column = `COLUMN' Name Size Format Ref TypeString NL
Size = String
Format = String
Ref = `NULL'
TypeString = Quote TypeDesc { Bar TypeDesc } Quote
TypeDesc = `Integer' | `String' | `Real' | `Datetime'

Row = RegularRow | ComputeRow
RegularRow = `ROW' { Field } NL
ComputeRow = `COMPUTE' ComputeLevel { Field } NL
ComputeLevel = Integer
Field = Integer | Real | String | DateString

DateString = Quote Month Day Year Time Quote
Month = `Jan' | `Feb' | `Mar' | `Apr' | `May' | `Jun'
| `Jul' | `Aug' | `Sep' | `Oct' | `Nov' | `Dec'
Day = Integer
Year = Integer
Time = Integer `:' Integer `:' Integer `:' Integer ( `PM' |
`AM' )

Identifier = Letter { Letter | Digit }
Integer = (`+' | `-'] Digit { Digit }
Real = Digit [ Dot { Digit }] [ `e' Integer ]
String = Quote { Char } Quote
Letter = `a'-'z' | `A'-'Z'
Digit = `0' - `9'
Dot = `.'
NL = `\n'
Quote = `\''
Char = `\0x01' - `\0xff'
Bar = `|'
```

7.3 Bild-Konversionsregeln

Wird ein Bild mit Hilfe eines **IMAGE** Objektes eingesetzt, so kontrolliert Oregator bei der Ausgabe des Bildes die Signatur der Bilddaten (magic). Folgende Bildtypen können verarbeitet werden:

Tabelle 9: Bildformate

Ausgabeformat	Bilddatei-Formate
PostScript	EPS, TIFF
Cairo PostScript	JPEG, PNG, GIF
PDF	JPEG, TIFF, GIF
HPDF	JPEG, PNG, GIF

Bandit	BMP

Handelt es sich um ein Bild, so wird bei unbekanntem Dateiformat automatisch eine Konversionsprozedur gestartet, die das Bild in ein geeignetes Bildformat umsetzt werden. Die Konversionsregeln sind in einer Textdatei abgelegt, und beschreiben, wie dieser Konversionsvorgang exakt geschieht. Auf diese Weise kann die Qualität der Umsetzung frei konfiguriert werden, See -T rule_file Mit dieser Option kann die Bild-Konversionsregeldatei eingestellt werden.

Es ist darauf zu achten, dass für die Konversion genügend Temporärer Speicherplatz vorhanden ist, See Environment: Temp-Verzeichnis OREGATOR_TMP.

Die Konversion des Bildes geschieht in rekursiven Schritten. Zuerst wird das Bild analysiert, dann entsprechende Konversionsprozess gestartet. Das Resultat kann wieder analysiert und konvertiert werden, solange bis ein druckreifes Resultat vorliegt oder der Vorgang abgebrochen wird.

Die Konversionsregeln bestehen aus einer Liste von Anweisungen der folgenden Form:

```
ANALYSIS
  Analysebefehl $0
RESPONSE
  Mehrere Antwortzeilen, die alle in der Ausgabe
  des Analysebefehls exakt vorkommen müssen
  ...
CONVERSION
  Liste von Konversionsbefehlen
  ...
  REMOVE $n...
  PROCESS $n | ABORT | ANALYZE $n
END
```

Anstelle der symbolischen Platzhalter **$0** .. **$9** werden eindeutige Temporär-Dateinamen eingesetzt. **$0** enthält den Namen der Ursprungsdatei. Die Befehle und Antwortzeilen müssen jeweils mit dem Zeichen TAB (0x09) eingeleitet werden.

Alle Zeilen, die mit dem Zeichen `**#** ' eingeleitet werden, sind Kommentarzeilen und werden ignoriert.

Die erzeugten Zwischendateien sollten mit dem Befehl **REMOVE** wieder gelöscht werden. Der Befehl **REMOVE** arbeitet analog wie das UNIX Kommando `rm -f'. Es werden aber nur die

temporären Dateien entfernt.

Die **CONVERSION** Befehlsliste endet mit einem der drei Schlüsselwörter **PROCESS** , **ABORT** oder **ANALYZE** . Beim Einsatz von **PROCESS** wird die angegebene Datei als EPSF-File in den Report eingesetzt. Bei **ABORT** wird die Verarbeitung des Bildes abgebrochen, das Bild bleibt leer in der Ausgabe. Bei **ANALYZE** wird die Analyse mit der angegebenen Datei als Original erneut gestartet.

Es wird immer nur die erste zutreffende Konversionsregel ausgeführt. Trifft keine der Regeln auf das Bild zu, so wird die Konversion abgebrochen und es erscheint kein Bild auf dem Ausdruck.

Der Fortschritt der Konversion kann durch die mit Hilfe der Debug-Option verfolgt werden.

7.4 PDFLib Konfiguration

Zur Konfiguration der PDF-Generierung stehen folgende Möglichkeiten zur Verfügung:

pdflib.upr - PDFlib Resource Configuration

Wird die Datei wird auf dem AFM-Suchpfad gefunden, so wird Sie auch eingelesen und verarbeitet. Der Pfad zur gefundenen Datei wird auch gleich als Prefix fur alle in der Datei enthaltenen Angaben vorkonfiguriert. Details zu den Angaben in der Datei findet man im PDFlib Handbuch Abschnitt 3.3.6. Die Datei kontrolliert folgende Einstellungen:

- Ort und Name der AFM-Dateien zu den PostScript-Fonts
- Ort und Name der PFM-Dateien zu den Fonts
- Ort und Name der Outline-Fonts für Embedding
- Ort und Name der Encodingdateien für Custom Encoding

pdflib.ini - PDFlib Font Configuration

Wird diese Datei auf dem AFM-Suchpfad gefunden, so wird sie auch eingelesen und verarbeitet. Die Datei enthält weitergehende Informationen zur Fontverarbeitung. Diese Textdatei, die mit einem einfachen Editor erstellt werden kann, enthält folgende Informationen:

- Alle Zeilen die mit einem #-Zeichen beginnen werden als Kommentarzeilen igonriert

- Font-Substitution: Globaler Ersatz eines Fonts durch einen anderen Font:

```
FontSubstitution   Helvetica   Helvetica-Bold
FontSubstitution   Courier     MyFont
```

- Font-Encoding: Definiert ein vom Standard abweichendes Encoding fur einen bestimmten Font. Das Encoding muss in der Datei pdflib.upr definiert worden sein.

```
FontEncoding       MyFont      MyEncoding
```

- Font-Embedding: Definiert welche Fonts in die generierte PDF-Datei eingebettet werden sollen. Normalerweise werden keine Fonts eingebettet, damit stehen normalerweise nur die 14 Standard-Fonts zur Verfügung:

```
FontEmbedding      MyFont
```

7.5 HPDF Konfiguration

Die HPDF Library wird in der Oregator UTF-8 Version für die Erzeugung von PDF-Dateien eingesetzt.

hpdf.ini - HPDF Treiberkonfiguration (Oregator UTF-8 Version)

Wird diese Datei auf dem AFM-Suchpfad gefunden, so wird sie auch eingelesen und verarbeitet. Die Date enthält weitergehende Informationen zur Fontverarbeitung. Diese Textdatei, die mit einem einfachen Editor erstellt werden kann, enthält folgende Informationen:

- Alle Zeilen die mit einem #-Zeichen beginnen werden als Kommentarzeilen ignoriert

- Font-Substitution: Globaler Ersatz eines Fonts durch einen anderen Font:

```
FontSubstitution   OCR-B-CH   PrecisionID OCR B1
FontSubstitution   Courier    MyFont
```

- Font-Embedding: Definiert welche Fonts in die generierte PDF-Datei eingebettet werden sollen. Angegeben wird der Name einer TrueType Font-Datei, die sich ebenfalls im AFM-Suchpfad befinden muss.

```
FontEmbedding  PrecitionID OCR B1  PrecisionID OCR
B1.ttf
```

- WritePermission: Per Default sind alle generierten PDF-

Dokumente schreibgeschützt, wobei das Anbringen von Annotationen (Notizen) im PDF gestattet ist. Möchte man den Schreibschutz für einen Report entfernen, so kann für alle betreffenden Reports ein Eintrag in hpdf.ini gemacht werden:

```
WritePermission   ReportName
WritePermission   ReportName2
```

7.6 Cairo PostScript Konfiguration

Der Cairo PostScript-Treiber wird in der UTF-8 Version für die Generierung von PostScript Dateien genutzt.

cairo_ps.ini - Cairo Treiberkonfiguration (Oregator UTF-8 Version)

Wird diese Datei auf dem AFM-Suchpfad gefunden, so wird sie auch eingelesen und verarbeitet. Die Datei enthält weitergehende Informationen zur Fontverarbeitung.

Diese Textdatei, die mit einem einfachen Editor erstellt werden kann, enthält folgende Informationen[7]:

- Alle Zeilen die mit einem #-Zeichen beginnen werden als Kommentarzeilen ignoriert

- Font-Substitution: Globaler Ersatz eines Fonts durch einen anderen Font:

```
FontSubstitution  Helvetica   LiberationSans
FontSubstitution  Helvetica-Bold    LiberationSans Bold
FontSubstitution  Helvetica-Oblique LiberationSans Oblique
FontSubstitution  Helvetica-BoldOblique   LiberationSans Bold Oblique
FontSubstitution  Times-Roman LiberationSerif Regular Roman
FontSubstitution  Times-Bold  LiberationSerif Bold Roman
FontSubstitution  Times-Italic      LiberationSerif Regular Oblique
FontSubstitution  Times-BoldItalic  LiberationSerif Bold Oblique
FontSubstitution  Courier     LiberationMono Regular Roman
FontSubstitution  Courier-Bold      LiberationMono Bold Roman
FontSubstitution  Courier-Oblique   LiberationMono Regular Oblique
FontSubstitution  Courier-BoldOblique     LiberationMono Bold Oblique
```

Vor dem Laden der Treiberkonfigurationsdatei, wird der Cairo PostScript-Treiber mit obenstehende Font-Substitutionstabelle initialisiert. Die Defaults können durch Einträge in der Treiberkonfigurationsdatei überschrieben werden. Nicht jedes System hat die PostScript Standardschriften vorinstalliert. Die effektiv zur Anwendung kommenden Fonts werden mit Hilfe von

7 Die Konfigurationsdatei kann bis zu 3 Spalten haben, die durch das Trennzeichen TAB separiert werden

FontConfig[8] ermittelt.

- PostScriptLevel: Steuert den PostScript-Level der Ausgabe. Unterstützt sind Levels 2, 3:

```
PostScriptLevel   3
```

7.7 Treiberspezifische Optionen (-k)

FORMAT BANDIT

HEAT=10
Dieser Befehl ändert die Einschaltzeit des Druckelements. Factory-Default ist 10. Eine Erhöhung des Parameters bewirkt eine grössere Hitze beim Druck.
Wertebereich: 00-30.

COFFSET=0000
Diese Option setzt den horizontalen Offset in 1/100 inch. Default: 0000.

ROFFSET=0000
Diese Option setzt den vertikalen Offset in 1/100 inch. Default: 0000

fSPEED=E
Diese Option setzt die Präsentationsgeschwindigkeit. Wertebereich: A-G. A=1.0 inch/s, B=1.5 inch/s, C=2.0 inch/s, D=2.5 inch/s, E=3.0 inch/s, F=3.5 inch/s, G=4.0 inch/s. Defaultwert = SSPEED

PSPEED=E
Druckgeschwindigkeit, Wertebereich wie fSPEED

pSPEED=E
Backfeed-Speed, Wertebereich wie fSPEED

SSPEED=E
Vorschubgeschwindigkeit im nicht-druckbaren Bereich, Wertebereich wie fSPEED

FORMAT TEXT

TXT_STRIP=0
Schaltet den Zeilenabschnitt aus(=0) oder ein (=1). Defaultwert ist 1, d.h. Whitespace am Ende der Ausgabezeilen wird entfernt. Schaltet man den Zeilenabschnitt aus, so wird der Whitespace nicht mehr entfernt und alle Zeilen werden in der vollen Länge ausgegeben.

8 Fontconfig is a library for configuring and customizing font access. https://www.fontconfig.org/

FORMAT CAB

CAB_LABEL_DISTANCE=2

Definiert den Abstand zwischen den Etiketten in mm. Defaultwert: 2mm

CAB_PEEL_OFF=10

Wenn der Drucker mit einer Peel-Off Einrichtung ausgerüstet ist, kann mit Hilfe dieser Option der Peel-Off Modus im Drucker aktiviert werden. Mit der Zahl wird das Displacement in mm angegeben. Ein höherer Wert führt dazu dass die Etikette beim präsentieren weiter nach aussen geschoben wird. Defaultwert: 0 = kein Peel-Off Modus.

CAB_ROTATE=1

Rotiert den Ausdruck um 180 Grad. Defaultwert: 0 = keine Rotation.

CAB_FONT_INDEX=Courier@99;Helvetica@97;Symbol@98

Die Fonts können beim CAB Treiber mittels dieser Option eingestellt werden. Wobei die Zahl dem Slot der Font entspricht, welche dem Download.exe Tool angegeben wurde. Die Fonts können bei neueren Druckermodellen auch direkt via FTP (Binary Mode!) auf den Drucker ins Verzeichnis /card/fonts geladen werden. In der Option werden die Original-Oregator-Fontnamen angegeben. Der Dateiname (leider nur 8.3 Zeichen möglich) wird durch wegstrippen aller Sonderzeichen gebildet. Heist der Font im Oregator z.B. Pefa-Bold so wird eine Fontdatei namens `pefabold.ttf' gesucht.

Die Font-Indexnummern müssen so gewählt werden, dass sie nicht mit den internen Fontindices des Druckers kollidieren. Dazu konsultiert man am besten erst die interne Fontliste des Druckers via Web-Frontend oder Panel-Ausdruck. Note: Optional installierte Fonts sind auf der internen Fontliste des Druckers NIE ersichtlich.

Angegeben werden die Oregator-Fontnamen.

FORMAT PCL

PCL_MEDIASOURCE=PCL_DEFAULT
Definiert die Papierquelle. Zulässige Werte sind: PCL_DEFAULT, PCL_AUTO, PCL_MANUAL, PCL_MULTIPURPOSE, PCL_UPPER, PCL_LOWER, PCL_ENVELOPE, PCL_THIRD

TEC_PAPER_SENSOR=0
Steuert den Etiketten-Sensor:
0 = No Sensor
1 = Reflective Sensor
2 = Transmissive Sensor (w/ normal labels)
3 = Transmissive Sensor (w/ preprinted labels)
4 = Reflective Sensor (w/ manual threshold value)

TEC_ISSUE_MODE=C
Batch/Strip Modus
C = Batch Modus
D = Strip Modus (with back feed, the strip sensor if valid)
E = Strip Modus (with back feed, the strip sensor is ignored, the applicator supports this modus)

TEC_ISSUE_SPEED=3
Ausdruckgeschwindigkeit
3 = 3 inch/s
4 = 4 inch/s
5 = 5 inch/s
8 = 8 inch/s

TEC_RIBBON=0
Farbband-Typ:
0 = Kein Farbband
1 = Farbband mit Sparmodus
2 = Farbband ohne Sparmodus

TEC_MODEL=B-452
Druckermodell, muss beim Einsatz von Download-Fonts beim Drucker Modell B-452 gesetzt werden.

TEC_TTFONT01=FontName
Definiert den Fontnamen der in die Bank 01 geladen wurde.
Wertebereich Bank: 01-24

7.8 Versionen

Der folgende Abschnitt beschreibt Unterschiede und Neuerungen zwischen den verschiedenen Oregator-Versionen.

Version V3.01.005 / 26.12.16

- Bug fix: version number fixed (V3)
- New: HPDF driver: Support for QR-Code and Datamatrix PR#210859
- Bug fix: PDF: invalid barcode font size != 0

Version V3.01.006 / 29.09.17

- New: CAIRO PostScript driver (UTF-8 Version)
- Cairo Driver: use image converter rule file REPO_IMG_CONF_CAIRO
- Cairo Driver: read fontmap from CAIRO_PS_CONFIG_FILE
- Cairo Driver: read PostScriptLevel from CAIRO_PS_CONFIG_FILE
- Cairo Driver: added Generator,Title,Subject comments in output
- PDF,PostScript Drivers: new IMAGE option FGND, foreground stacking order
- Die direkten Etikettendrucker-Treiber (BANDIT, CAB, TEC) werden in der UTF-8 Version nicht mehr länger unterstützt. Setzen Sie zukünftig geeignete PostScript- oder PDF-Druckertreiber ein.

Version V3.01.007 / 23.12.17

- New: CAIRO PNG driver

Version V3.01.008 / 01.04.18

- New: FNC1_SYMBOL string constant
- New: Datamatrix DMTX_GS1 flag

Version V3.01.009 / 16.09.18

- Bug fix: ISO-8601 week number
- Bug fix: fixed format option LANDSCAPE for drivers Cairo-PS (UTF-8), HPDF (UTF-8), PNG (UTF-8), PDFLib (ISO), 224642,220106
- New: format options: ROTATE LEFT, ROTATE RIGHT for drivers: Cairo-PS (UTF-8), HPDF (UTF-8), PNG (UTF-8), PostScript (ISO), PDFLib (ISO) 224642,220106

Version V3.01.010 / 17.11.18

- Bug fix: week number 224640
- fixed struct tm initialisation with tm_isdst=-1 (use timezone information)
  use local copy of struct tm to ensure no side effects when using %W, only tm_year will be modified for later use of %Y
- Bug fix: -c option disabled in Cairo-PS (UTF-8), PNG (UTF-8) drivers 225486,225525

Version V3.01.011 / 15.12.18

- Bug fix: sql_args_list: command buffer overflow 225633
- Bug fix: (UTF-8) illegal characters in DATETIME('%B') 225633

Version V3.01.012 / 25.05.19

- New: support for QR_VERSION(V) 228879
- New: HPDF driver: support for PDF VERSION(STRING) in FORMAT declaration 220229

Version V3.01.013 / 07.07.19

- Bug fix: HPDF driver: fixed fontconfig V2.12.1 size pattern 229030

Version V3.01.014 / 14.09.19

- New: arithmetic function FOLD(Block.Field) returns formatted field contents (number format, PICTURE)

Version V3.01.015 / 19.09.19

- New: support for Block.Field[aexp] and Block.Field[aexp,aexp]

Version V3.01.016 / 12.09.20

- New: support for **QR_CHARSET_UTF8**, Default remains ISO-8859-1

Version V3.01.017 / 11.10.20

- check thousands separator on startup with logged warning 238547

Version V3.01.018 / 19.12.20

- libpng updated to version 1.6.37 239716
- HPDF driver updated to latest version

Version V3.01.019 / 26.12.20

- Bug fix: SEGV when using FOLD() on unintialized field 240284
- field option NUMPAGES 239777,239336
- New: logical operators AND, OR, NOT
- Improvement: update libqrencode to v4.1.1

Version V3.01.020 / 23.01.21

- Bug fix: QR-Code diagonally flipped 300356

Version V3.01.021 / 11.02.21

- Cleanup: removed obsolete PCL5 driver

Version V3.01.022 / 31.03.21

- Bug fix: SEGV in opgU_error() 302506

Version V3.01.023 / 28.08.21

- Bug fix: NUMPAGES counter fixed 302701

Version V3.01.024 / 26.09.21

- Bug fix: NUMPAGES counter fixed for empty output pages 302043
- Bug fix: ELEMENT function doesn't return empty elements 306039
- Bug fix: ELEMENT function collapses subsequent separators 306039

Version V3.01.025 / 26.05.22

- Bug fix: ELEMENT function with multibyte delimiter 311543

Version V3.01.026 / 25.06.22

- Bug fix: cross compilation issue for MinGW fixed

Version V3.01.027 / 01.08.23

- Bug fix: alternative PROCESSING keyword: PROCESSING_ALL_BATCHES 318698
- Bug fix: prevent possible buffer overrun when setting FNC1_SYMBOL
- Bug fix: prevent possible buffer overrun when parsing analyzer configuration
- Bug fix: prevent possible buffer overrun when debug printing SQL output
- Bug fix: prevent possible buffer overrun when parsing locale definition
- Bug fix: initialize graphics info buffer with 0

Version V3.01.028 / 15.10.23

- Bug fix: DATETIME format %U fixed 320619
- enable datetime parser debug output starting from level 4 (insted of 1)

Version V3.01.029 / 04.11.23

- added debug output for compute level emulation 320396

Version V3.01.030 / 25.11.23

- added version to application name in SQL connect 0320868

Version V3.01.031 / 08.06.24

- Bug fix: lex warning: duplicate definition of symbols PAGE and LZERO
- New: field option **HYPERLINK** 325374

8 Index

Index

A.....
ABORT.....41f., 67f.
ABS.....43
Abstand.....25, 64, 72
Achse.....52f., 55
Addition.....43
aexp.....42f., 45, 75
afm.....20, 29, 65
AFM.....5, 29, 65, 68ff.
AFM-Dateien.....65, 68
AFM-Suchpfad.....68ff.
ANALYSIS.....67
ANALYZE.....67f.
AND.....47, 75
Antwortdatei.....3, 9f., 12, 65
AntwortDatei.....65
Area.....12ff.
AREA.....12, 20f., 25, 32, 61f.
Artithmetische Ausdrücke...42
ASSIGN.....42
Auflösung.....48
Ausgabe.....7, 17ff., 23, 28ff., 32f., 42, 45, 49, 55, 62, 66ff., 71
Ausgabeformat...6, 19, 33, 66
B.....
Balkendiagramm.....54
Bandit.....67
BANDIT.....18ff., 71, 74
BARCODE.....63
BARCODE_128.....31
BARCODE_25.....13f., 30, 32
BARCODE_39.....31
BARCODE_DMTX.....31
BARCODE_EAN.....31f.
BARCODE_QR.....31
BARCODE_RATIO.....32
BARCODE_ROTATE.....32
Batch.....39f., 58, 64f., 73
BATCH.....12, 14, 16, 32, **35,** 36f., 39ff., 45, 53, 56, 59
Beschriftung.....32, 52f., 55
Bild-Konversionsregeln 27, 66
Block....22ff., 26, 33, 38ff., 42, 44, 49, 58, 75
BLOCK 12ff., 22, 24f., 32f., 41, 45, 50, 52f., 55, 58f., 61f.
BMP.....67
Bookmark.....33
BOOKMARK.....33
BORDER.....26, 33, 55f.
BOTTOM.....25
Box.....26, 40
BOX.13f., **26,** 32, 40, 55f., 62f.
Break.....37
BREAK.....24, 36, **37**
C.....
CAB.....19f., 72, 74
Cairo.....64, 66, 70
Cairo PostScript Konfiguration.....70
cairo_ps.ini.....70
CENTERED.27f., 50, 52f., 56
character set.....7, 9
character_set.....8
charset.....7, 9
CHARSET.....42
CLEAR_PAGE.....41f.
CLIP_AREA.....32
CLIP_BLOCK.....32
CLOSE.....10, 49
Code-128.....31
COMMA.....30, 44
compute.....13, 58, 76
Compute.....3, 14, 38, 58f.
COMPUTE. .14, 35f., 38f., 58ff., 66
CONTROL.....17, 44f., 64
conversion.....6
CONVERSION.....67f.
copies.....6
count.....39
COUNT.....43, 48
CPI.....12, 18f., 49
CSV.....10, 19f., 44
D.....
DARKGRAY.....26, 56
database.....7, 9
Datamatrix-Codes.....31
Datamax.....19f.
Date.....13f., 68f.
DateFormat...27, 29, 43, 45
Datei.....70
Datenbank 7f., 10, 13f., 27, 45, 54f.
datetime.....76
Datetime.....66
DATETIME. 14, 27, 29, 34, 43, 45, 49, 75f.
Datum.....27, 29f., 43, 45
Datumsformatierung. 29, 45
dbname.....10
debug.....76
Debug...5, 8, 10, 36, 42, 68
DEBUG.....42
Debug-Flags.....5
default.....6, 18, 65
Default...6ff., 12, 18f., 21f., 27, 31, 33, 40, 42, 61, 64f., 69, 71, 75
DEFAULT.....72
Default.....18
Deferred.....23
DEFERRED24, 38, 45, 49, 59
Deferred-Speicher.....23
Dezimalkomma.....30
Dimension.....51
Division.....43
DMTX_GS1.....32, 74
DMTX_SCHEME_ASCII.31
DMTX_SCHEME_BASE25

6.....31
DMTX_SCHEME_C40.....31
DMTX_SCHEME_EDIFACT 31
DMTX_SCHEME_TEXT.....31
DMTX_SCHEME_X12.....31
Druckbereich.....21f.
Druckbereiches.....21f.
DSQUERY.....7
dynamisch.....21f., 25
E.....
EAN-13.....31
EAN-8.....31
EAN-Prüfziffer.....44
EANPZ.....44
eingebettet.....69
Element.....44
ELEMENT.....44, 76
Embedding.....68f.
Encoding.....68f.
Encodingdateien.....68
EPS.....66
EPSF.....7, 26, 62, 68
Ersatz.....43f., 68ff.
Excel.....19, 63
EXISTS.....44
F.....
Fehler.....10
Feldoptionen.....**28,** 31, 49
FETCH.....44
FGND.....27, 74
field.....75, 77
Field.....66, 75
FIELD. 12ff., **27,** 28, 30, 32ff., 40, 45, 49f., 53, 55f., 58f.
filename.....9f.
FIT_HORIZONTAL.....24, 38
FIT_VERTICAL.....24, 38
fixed.....73ff.
FIXED.....27, 30, 34, 44
Fliesskommavergleich.....48
FNC1.....31f.
FNC1_SYMBOL.....18, 32, 74
FOLD.....44, 75
font.....70, 73
Font.....32, 68ff., 72
FONT.....27f., 32, 34, 72
Font-Embedding.....69
Font-Encoding.....69
Font-Substitution.....68ff.
Font-Substitutionstabelle 70
FontEmbedding.....69
FontEncoding.....69
fonts.....72
Fonts.....28f., 68ff., 72f.
FontSubstitution.....69f.
FontSubstitution.....70
Fontverarbeitung.....68ff.
Footer.....14, 41
FOOTER. 14, 16, 25, 35f., 38, 41, 59
format.....6, 74ff.
Format 3, 7, 9f., 19ff., 29f., 45, 63, 65f.
FORMAT 12, **18,** 19ff., 33, 42, 44, 62, 71ff., 75
Formatname.....44
Fortschrittsmeldungen...10
FrameMaker 3, 17, 19, 61f.
FRAMEMAKER.....18ff.
G.....
Generieren.....5
GIF.....66
Grafik.....3, 50ff.
graphic.....2
Graphic.....54
GRAPHIC.....50, 52f., 55
GRAY.....26, 56
GS1-128.....31
H.....
Header.....41
HEADER 16, 35f., 38, 41f., 45, 53, 56, 59f.
History-Speicher.....23
Hochkomma.....30
Horizontal.....5
HORIZONTAL...21, 24, 38
Host-Id.....4
hostname.....8f.
hpdf.....20, 69f.
HPDF....3, 20, 66, 69, 73ff.
HPDF Konfiguration.....69
hpdf.ini.....69f.
HYPERLINK.....33, 77
I.....
image.....74
IMAGE **26,** 27, 40, 62f., 66, 74
images.....27, 65
Include.....17, 64
INCLUDE.....16f., 33f.
INF_SYMBOL.....17
Inherit.....41
INHERIT.....24, 41
INPUT 12, 14, 36, 38ff., 45, 51, 56f., 59
INPUT BATCH.....36, 39f.
Installationsverzeichnis...4f.
Instanz. .22, 24, 28, 38, 43, 46f.
ISO-8859-1.....42
J.....
Jahrhundert.....30
Journalisierung.....10, 49
JPEG.....66
K.....
KEEP. .23, 27, **32,** 33, 46f., 58f.
Konversionsregeldatei 6, 67
Konversionsregeln....3, 27, 64, 66f.
Kopien.....6
L.....
LANDSCAPE.....18f., 74
LEFT...18, 20, 27f., 52f., 74
LENGTH.....44
LIGHTGRAY.....26, 55
Line.....40
LINE.....**26,** 32f., 40, 62f.
Linie.....26, 52, 54
Listenelement.....44
Lizenz.....4
LOG.....42
Logging.....10
LPI.....12, 18f.
LTRIM.....44, 60
LZERO.....43, 48, 77
M.....
MIF File.....19
MIF_TEMPLATE....17f., 61
Monatsname.....29
Monatstag.....29
MSSQL.....9
Multiplikation.....43
Muster.....11f., 43
MySQL.....9
N.....
Nachkommastellen.....30
NAN_SYMBOL.....17

NEED_HORIZONTAL.....24
NEED_VERTICAL.....24f.
new.....74
New.....73ff., 77
NEW….12, 14, 23f., 38ff., 45, 51, 53, 56, 59
Newline.....24
NEWLINE.....44
NOQUOTE.....30, 44
NOT.....47, 75
Note.....72
NOTE.....**33**
null.....54
Null.....61
NULL.....3, 48, 60, 66
NUMPAGES.....30, 75
O.....
Objektdeklaration.....25
OPEN.....10, 49
OR.....47, 75
order by.....55, 58
ORDER BY.....37
oregator.....4f., 8, 11ff., 64
Oregator. 4f., 8, 10f., 17, 19f., 31, 33, 39, 45, 49, 66, 69f., 72f.
OREGATOR. 1, 5f., 8, 17, 27, 29, 61, 65, 67
OREGATOR_LICENSE. 5, 65
OREGATOR_PATH…5f., 17, 27, 29, 61, 65
oregator.lic.....4f., 64
Outline-Fonts.....68
output.....6f., 74, 76
Output.....19, 44
OUTPUT.....10, 49
output_file.....7
P.....
page.....6
Page.....19, 61f.
PAGE……25, **35,** 38, 40ff., 77
PAGE FOOTER…..25, 38, 41
pagenr.....6
PAGENR.....43, 46
pageview.....12
passwd.....7ff.
password.....7
Passwort.....11
PCL.....18, 20, 72
PDF 12, 18ff., 27, 33, 66, 68f., 73ff.
PDF.....18
PDF/A-1B.....20
PDFlib Font Configuration 68
PDFlib Resource Configuration.....68
pdflib.ini.....68
pdflib.upr.....68f.
pendente.....24
PFM-Dateien.....68
PICTURE.....30, 44, 75
PNG.....19f., 66, 74
points.....55
Points.....54f.
portnum.....8f.
PORTRAIT.....12, 18f.
Positionierung..24f., 38, 50
Postscript.....19
PostScript 2f., 7, 12f., 26ff., 33, 66, 68, 70f., 74
POSTSCRIPT. .7, 12, 18ff.
PostScriptLevel.....71
Präfix.....17, 30
prepend.....5
PREPEND.....17
PROCESS.....67f.
PROCESSING 12, **16,** 56, 76
PROCESSING_ALL_BATCHES.....**16,** 76
protocol.....8f.
Protokoll.....8f.
Q.....
QR_CASE_INSENSITIVE 31
QR_CASE_SENSITIVE 31
QR_CHARSET_UTF8. 31, 75
QR_ECLEVEL_H.....31
QR_ECLEVEL_L.....31
QR_ECLEVEL_M.....31
QR_ECLEVEL_Q.....31
QR_MODE_8.....31
QR_MODE_AN.....31
QR_MODE_ECI.....31
QR_MODE_FNC1FIRST 31
QR_MODE_FNC1SECOND.....31
QR_MODE_KANJI.....31
QR_MODE_NUM.....31
QR_MODE_STRUCTURE 31
QR_VERSION.....31
QR-Codes.....31
Qualität.....67
R.....
Rechengenauigkeit.....48
Record.....38
RECORD….12, 14, 35, **36,** 38ff., 45, 51, 56, 59
REMOVE.....67
report.....5f.
Report…5f., 10, 33, 39, 49, 61ff., 68, 70
REPORT.12, 16, **34,** 35, 45
Reportname.....45
ReportName.....70
response.....9
RESPONSE.....67
RGB.....26f., 32
Right.....61f.
RIGHT….13f., 18, 20, 27f., 30, 34, 50, 56, 74
ROTATE LEFT…18, 20, 74
ROTATE RIGHT.18, 20, 74
ROUND.....43, 48
RTRIM.....44, 60
rule_file.....6
S.....
Schriftgrösse.....28, 34
Seite….6, 23ff., 28, 33, 35, 38, 41ff., 45ff., 50, 59, 62
Seiten-Speicher.....23
Seitenvorschub.....21
Sekunde.....29
select…6, 11ff., 39, 51, 55, 64
SELECT.....35f., 39, 58
Sequenz.....45
server.....7, 9
Skalierung.....52f.
SOLID.....8
Speicherplatz.....4, 67
Standard-Fonts.....69
statisch.....21
stdout.....7
Stored Procedure.....13
Strichdicke.....26, 30

string.....48, 74
String.....20, 42ff., 64ff.
STRING.....60, 75
String.....43
STROKE.12, 14, 23f., 33, 40f.
STRSTR.....43
STRSUB.....43
Stunde.....29
Style.....50, 53f., 56
Substitution.....68ff.
SUBSTR.....44
Subtraktion.....43
Suffix.....30
sum.....13, 58
SUM.....43, 48
Summen-Übertrags-Zeile…25
Sybase.....2, 7
SYBASE.....27
T.....
Tab.....17, 62f.
TAB.....67, 70
Tagesnummer.....29
TEC.....18ff., 73f.
Text. 12f., 17, 24, 27f., 33, 44, 52
TEXT 12, 17ff., 24, 27, **28,** 31, 49, 71
TIFF.....6, 66
Treiberspezifische Optionen.....7
U.....
UNIQ.....27f., 34
user.....5, 7ff.
USER.....7
username.....7ff.
utf-8.....42
V.....
Vektoren.....50ff.
Vektorwertige Felder…..**50**
Verkettung.....43
VERSION.....20
vertical.....5
VERTICAL.12, 21, 24f., 38
W.....
WHILE.....39
Windows.....3, 19, 63
WinWord.....63
WINWORD.....18ff., 63
Wochennummer.....29f.
Wochentag.....29f.
Word.....18f., 22
Word.....18
X.....
Xvalue, Xlabel, XLine.….52
Y.....
Y0, Y1, ... Y9.....52f.
Yvalue, Ylabel, YLine.….53
Z.....
Zeit.....10, 29f.
Zuweisung. .32, 42, 45, 51, 53
Zwischendatei.....11
Zwischenraum.....25
BLOCK.....33
.....
.cshrc.....5
.profile.....5
`.....
`'.....17, 32, 48