SOFTWARE

WEGA

WEGA-DATA Systemhandbuch

EAW *electronic*

P8000

Version 1.0 (2008-02-17)

Diese Dokumentation wurde von einem Kollektiv des

Kombinat
VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"

erarbeitet.

Nachdruck und jegliche Vervielfaeltigungen, auch auszugsweise, sind nur mit Genehmigung des Herausgebers zulaessig. Im Interessse einer staendigen Weiterentwicklung werden die Nutzer gebeten, dem Herausgeber Hinweise zur Verbesserung mitzuteilen.

Herausgeber:

Kombinat
VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"
Hoffmannstrasse 15-26
BERLIN
1193

Verantwortlicher Bearbeiter: Michael Rachow

WAE/03-0308-01
Ausgabe: 12/87

Aenderungen im Sinne des technischen Fortschritts vorbehalten.

Die vorliegende Dokumentation unterliegt nicht dem Aenderungsdienst.

Spezielle Hinweise zum aktuellen Stand der P8000-Softwarepakete befinden sich in README-Dateien auf den entsprechenden Vertriebsdisketten.

Inhalt:

1. EINLEITUNG

In diesem Kapitel soll das WEGA-DATA-Systemhandbuch vorgestellt werden. In ihm werden spezielle Informationen darueber gegeben, wie die verschiedenen Werkzeuge von WEGA-DATA zu verwenden sind. In den einzelnen Abschnitten werden Beispiele angefuehrt, wie WEGA-DATA verwendet werden kann. Im WEGA-DATA-Nutzerhandbuch sind zahlreiche Beispiele dafuer angefuehrt, wie WEGA-DATA-Werkzeuge im Zusammenhang mit der Entwicklung eines Anwendungsprojektes verwendet werden. Wer noch nie mit WEGA-DATA gearbeitet hat, sollte daher mit dem Studium dieses Nutzerhandbuches beginnen.

Das Systemhandbuch ist in 10 Kapitel gegliedert, die sich jeweils mit einem bestimmten Aspekt von WEGA-DATA beschaeftigen. Im ersten Kapitel soll das Handbuch vorgestellt und ein Ueberblick ueber die WEGA-DATA-Umgebung gegeben werden, d.h. darueber, welche Verzeichnisse und Dateien erstellt und welche Umgebunsvariablen vor der Verwendung von WEGA-DATA gesetzt werden muessen. Im Kapitel 2 wird MENUH, der Menue-Handler, beschrieben, der das primaere Nutzer-Interface fuer WEGA-DATA ist. In Kapitel 3 werden die Dienstprogramme der Datenbank vorgestellt, die zur Verwaltung der Struktur der Datenbank und der Datenbank selbst verwendet werden.

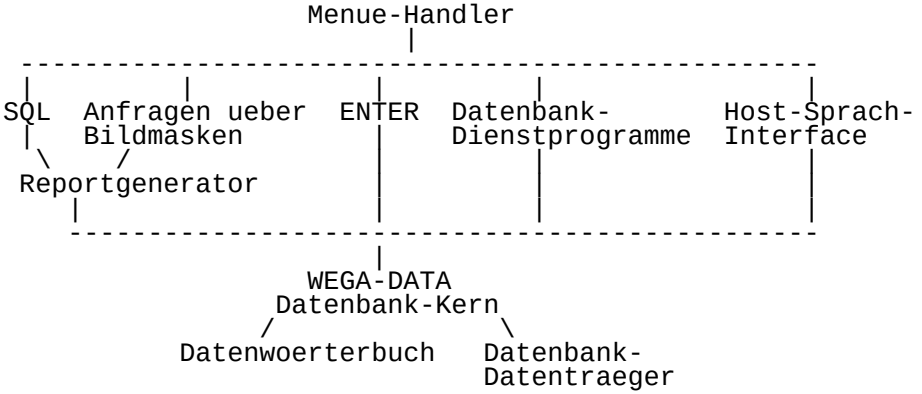
In Kapitel 4 wird SFORM behandelt, das Werkzeug zur Erstellung und Verwaltung von Bildmasken. Im naechsten Kapitel wird ENTER beschrieben, ein allseitig verwendbares Dateneingabeprogramm, das zum Betreiben von SFORM-Bildmasken verwendet werden kann. Ausserdem kann ENTER fuer Anfragen ueber Bildmasken an die Datenbank oder als Grundlage fuer nutzerspezifische Dateneingabe-Bildmasken dienen.

Nach Eingabe der Daten, koennen die in den Kapiteln 6 und 7 beschriebenen Programme SQL und RPT von WEGA-DATA verwendet werden, um die Daten abzufragen und in einem sinnvollen Format anzubieten. Neben SQL hat WEGA-DATA ein einfacheres, weniger leistungsfaehiges Anfrageinterface, mit dem Dateiauflistungen mit Summen und Zwischensummen erfolgen koennen. Dieses Programm ist der Listen-Prozessor und wird in Kapitel 8 beschrieben. Der Datenbank-Testtreiber, ein einfacher Datenbank-Editor, der in der Hauptsache fuer Programmierer gedacht ist, wird in Kapitel 9 beschrieben.

Kapitel 10 stellt die Werkzeuge vor, die von hoeheren Programmiersprachen aus genutzt werden koennen und mit denen nutzerspezifische Anwendungsfaelle geschrieben werden koennen. Waehrend sich das in Kapitel 10 beschriebene Host-Sprach-Interface auf C-Programme bezieht, beschreibt Anhang A das Interface fuer COBOL. Ausserdem koennen alle WEGA-Sprachen, die C-Funktionen aufrufen, wie z.B. Pascal und FORTRAN, mit dieser Schnittstelle arbeiten.

1.1 Die Umgebung von WEGA-DATA

In diesem Abschnitt und den folgenden Unterabschnitten soll die Betriebsumgebung von WEGA-DATA erlaeutert werden. Jedes Datenbank-Verwaltungssystem besteht aus mehr als einem Programm. WEGA-DATA verfuegt ueber mehr als 20 verschiedene Programme, die alle so integriert sind, dass der Anwender Datenbank-Systeme zum Speichern und Wiedergewinnen von Daten anlegen und modifizieren kann. Im folgenden wird der Aufbau des WEGA-DATA-Systems veranschaulicht:



Aufbau von WEGA-DATA

Das wichtigste Nutzerinterface fuer das WEGA-DATA-System ist der Menue-Handler MENUH. MENUH ermoglicht die Auswahl der gewuenschten Programme von WEGA-DATA und die Sicherheitsueberwachung, d.h., er kann den Zugriff auf bestimmte Programme verhindern. WEGA-DATA wird mit einer Anzahl integrierter Menues geliefert, die der Anwender fuer die Entwicklung seines Anwendungssystems nutzen kann. Es koennen aber auch eigene Menues angelegt werden. Die ausfuehrliche Beschreibung des Menue-Handlers erfolgt im Kapitel 2.

Die nichtprozedurale Anfrage- und Aktualisierungssprache von WEGA-DATA ist eine Implementation der 'Structured Query Language', SQL. Mit SQL kann man Informationen in die Datenbank interaktiv einfuegen, sie modifizieren, loeschen und abfragen. In Kapitel 6 werden die Eigenschaften von SQL beschrieben. WEGA-DATA ist eine Erweiterung von SQL. So kann man beispielsweise eine Bildmaske mit einem SQL-Anfragegskript verbinden und anschliessend die Ergebnisse der Anfrage mit dem Reportgenerator RPT formatieren. Dieses als 'SQL ueber Bildmasken' bezeichnete Interface wird in Abschnitt 6.2.6 beschrieben. Im Kapitel 4 wird das Erstellen von Bildmasken beschrieben und in Kapitel 7 wird RPT erlaeutert.

Zur Ausfuehrung einfacher Anfragen, die haeufig mehr als

die Haelfte der benoetigten Anfragen ausmachen, bietet WEGA-DATA die Moeglichkeit der 'Anfragen ueber Bildmasken'. Dabei kann man in eine Bildmaske Suchwerte einsetzen, die dann von der Bildmaske aus benutzt werden, um die dazugehoerigen Datensaetze zu suchen. Die Reporte koennen einzeln auf der Bildmaske betrachtet werden oder die Ergebnisse werden an einen der Reportgeneratoren von WDATA, RPT (leistungsfaehig) oder LST (leichte Handhabung), geschickt. Anfragen ueber Bildmasken werden in Abschnitt 5.2.2, RPT in Kapitel 7 und die LST-Kommandos zur Reportformatierung in Abschnitt 8.2 beschrieben. In Abschnitt 5.1 wird dann erlaeutert, wie der gesamte Prozess gekoppelt werden kann. In Kapitel 4 wird das Erstellen von Bildmasken beschrieben.

ENTER ermoeglicht die Dateneingabe ueber Bildsmasken. Die Beschreibung erfolgt in Kapitel 5. Mit ENTER kann man die Datenaufbereitung ueberwachen, einschliesslich der Typen-, Laengen-, Datums- und Zeitformate. Zur weiteren nutzerspezifischen Gestaltung koennen vom Nutzer eigene Funktionen hinzugefuegt werden. Wie die vorhergehenden Programme verwendet auch ENTER Bildmasken die, unter Verwendung der in Kapitel 4 beschriebenen SFORM-Dienstprogramme, erstellt und modifiziert werden koennen.

In Kapitel 3 werden die fuer das Anlegen und die Verwaltung einer Datenbank erforderlichen Dienstprogramme beschrieben. Hierzu gehoeren Programme fuer die Eingabe und Modifizierung des Aufbaus der Datenbank, zum Anlegen und Rekonfigurieren der Datenbank, zum Hinzufuegen und Loeschen von Indizes, zur Verwaltung der Sicherheit der Datenbank, zum Drucken von Statistiken, zum Schreiben und Lesen von Backup-Disketten, zur Wiederherstellung verstuemmelter Datenbanken und zum Laden von Daten aus regulaeren ASCII-Dateien. Man kann ausserdem festlegen, dass eine WEGA-DATA-Datenbank auf einem "raw"-Dateisystem gespeichert werden soll, wodurch die Leistungsfaehigkeit gegenueber der Verwendung einer normalen WEGA-Datei erheblich vergroessert wird.

Sollen Programme unter Verwendung des Host-Sprach-Interface' geschrieben werden, koennen Kapitel 10 (C-Sprach-Interface) und Anhang A (COBOL-Interface) herangezogen werden. In diesen Kapiteln werden mehr als 90 Funktionen beschrieben, mit denen man die Datenbank, die Bildmasken und Drucker zur Erzeugung nutzerspezifischer Anwendungsfaelle manipulieren kann. Mit dem Host-Sprach-Interface erreicht man eine wirklich hohe Leistungsfaehigkeit.

Die uebrigen Teile von WEGA-DATA sind systemintern und werden nicht extra beschrieben.

In den folgenden Unterabschnitten sollen WEGA-DATA-Dateien, der allgemeine Aufbau der Verzeichnisse, die Shell-Umgebungsvariablen und die Bedeutung der termcap, der Datenbasis, in der die Terminaleigenschaften enthalten sind, be-

schrieben werden. Obwohl man bei der Anwendung von WEGA-DATA kaum ueber Kenntnisse von WEGA, der Shell oder der Programmierung verfuegen muss, sind die in diesem Abschnitt angefuehrten Informationen ziemlich anspruchsvoll. Zum Verstaendnis muss man mit Verzeichnissen, der Shell und dem Kapitel ueber die termcap vertraut sein; man muss wissen, wie Terminals aus der Sicht der Programmierung funktionieren. In diesem Abschnitt werden Informationen zu verschiedenen Themen zusammengestellt, die an keiner anderen Stelle des Systemhandbuches beschrieben werden, die jedoch fuer die Anwendungen von WEGA-DATA sehr nuetzlich sind.

Da sich diese Informationen in gewisser Weise mit dem gesamten System beschaeftigen, werden viele Verweise auf andere Abschnitte des Systemhandbuches gegeben. Wenn der Leser noch nicht mit WEGA-DATA vertraut ist, sollte er sich nicht verunsichert fuehlen, wenn er noch nicht genau weiss, was in den jeweiligen Abschnitten beschrieben wird. Dennoch scheint es angeraten, dieses Kapitel des Handbuches zu Beginn zu lesen, um einen gewissen Ueberblick zu erlangen. Nachdem sich der Leser mit WEGA-DATA vertraut gemacht hat, sollte er dann dieses Kapitel erneut durcharbeiten; dann wird er die Anwendung der Informationen selbstaendig einordnen koennen.

1.1.1 Dateien

Bei WEGA-DATA werden Vereinbarungen fuer die Benennung von Dateien angewandt, um die verschiedenen, fuer den Anwendungsfall erforderlichen, Dateiarten identifizieren zu koennen. Dateinamen haben die Form name.erweiterung, wobei sowohl name als auch erweiterung eine variable Laenge haben. Mit dem Suffix wird der Dateityp angegeben, wie es den normalen Vereinbarungen zur Benennung von WEGA-Dateien entspricht. Es folgt eine Liste der Erweiterungen der WEGA-DATA-Dateinamen, ihre Bedeutung und die Beschreibung wichtiger Dateien dieses Typs. WEGA-DATA-Dateien werden normalerweise in einem bestimmten Verzeichnis gespeichert, das jeweils auch angefuehrt wird. Im naechsten Abschnitt wird dann der Aufbau der Verzeichnisse naeher erlaeutert.

.a Eine mit dem WEGA-Kommando ar(1) angelegte Archivdatei, die verschiebbare binaere Dateien in einem Format enthaelt, das zum Laden mit dem Kommando ld(1) geeignet ist. WEGA-DATA hat einige Systemarchive, die im Verzeichnis lib des Systems sind. Ausserdem kann der Anwender Archivdateien anlegen, die sich normalerweise im Unterverzeichnis src befinden.

.c Eine C-Quelldatei. Sie wird vom Anwender angelegt und im Unterverzeichnis src gespeichert.

.db Eine Datenbank-Datei. Fuer jeden Anwendungsfall gibt es zwei Datenbank-Dateien - das Datenwoerterbuch,

wdata.db, und die Datenbank-Datei fuer das jeweilige Datenbank-System, file.db. Der Nutzer kopiert bei der ersten Inbetriebnahme von WEGA-DATA sein eigenes wdata.db aus dem System-Prototyp wdata.db. file.db wird mit 'Create Data Base' (Abschnitt 3.2.1) angelegt. Die Datenbank-Dateien des Nutzers werden in seinem Verzeichnis bin gespeichert, waehrend sich das Prototyp-Datenwoerterbuch im Verzeichnis lib des Datenbank-Systems befindet.

.dbr Die "raw"-Datenbank-Datei namens file.dbr. Wird ebenfalls von 'Create Data Base' (Abschnitt 3.2.1) angelegt. Fuer jeden Anwendungsfall gibt es eine solche Datei; diese ist jedoch entweder durch Link mit file.db verbunden, naemlich wenn fuer die Datenbank eine normale WEGA-Datei verwendet wird, oder sie zeigt auf denselben Platten-Speicherbereich wie filed.db, wenn fuer die Datenbank ein Dateisystem verwendet wird. Es handelt sich hierbei um eine weitere Moeglichkeit des Zugriffs auf die Datenbank-Datei des Anwendungsfalles.

.h Eine include-Datei, die in Verbindung mit dem C-Quellkode verwendet wird. Solche Dateien befinden sich entweder im Verzeichnis def oder include. WEGA-DATA legt zwei Arten von .h-Dateien an. Die erste ist file.h, die bei 'Create Data Base' oder 'Reconfigure Data Base' entsteht (Abschnitt 3.2.1 und 3.2.2). Diese Datei enthaelt eine Liste von Datensatz- und Feldnamen, die von den auf die Datenbank zugreifenden C-Programmen verwendet werden. Die zweite Art der .h-Dateien sind die Dateien bildmaske.h, die entweder von 'Paint Screen' (Abschnitt 4.1) oder von 'Process Screen' (Abschnitt 4.4) angelegt werden. Diese Dateien enthalten eine Liste der Bildmasken-Feldnamen, die von den, die entsprechende Bildmaske benutzenden, C-Programmen verwendet werden.

.idx Eine B-Baum-Indexdatei. Diese Dateien sind binaere Dateien, die im Verzeichnis bin des Nutzer gespeichert sind. Sie werden von 'Index Maintenance' (Abschnitt 3.2.3) angelegt und werden von den Programmen aktualisiert, die Daten in ein indiziertes Feld speichern. Die Dateien btnnnn.idx (nnnn ist eine Zahl) sind wirkliche B-Baum-Dateien, waehrend die Datei field.idx ein Verzeichnis fuer die Indexdateien ist.

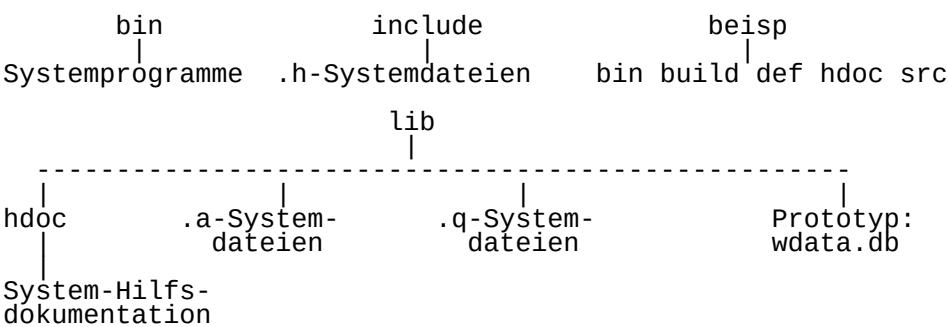
.ld Eine Shell-Kommandodatei, die das WEGA-Kommando ld(1) zum Anlegen von ausfuehrbaren Dateien verwendet. Diese werden normalerweise im Verzeichnis build des Nutzers gespeichert.

.o Eine verschiebbare binaere Datei, die sich aus einem Aufruf cc -c ergibt. Diese Dateien werden normalerweise in einer Archivdatei gespeichert.

.q Eine Bildmasken-Datei. Diese Dateien enthalten binaere Versionen von Bildmasken, die schnell waehrend der Laufzeit gelesen werden koennen. Sie werden entweder durch 'Paint Screen' (Abschnitt 4.1) oder durch 'Process Screen' (Abschnitt 4.4) erzeugt. System-Bildmasken von WEGA-DATA werden im Verzeichnis lib des Datenbank-Systems gespeichert, wohingegen die Bildmasken des Nutzers normalerweise in dessen Verzeichnis bin gespeichert werden.

1.1.2 Verzeichnisse

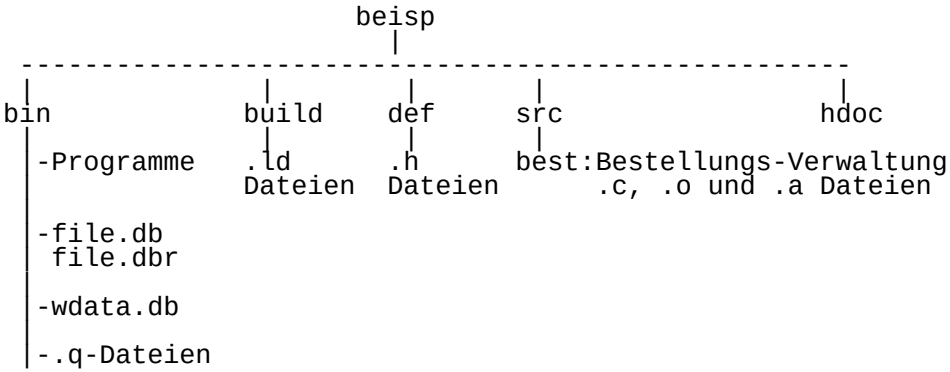
WEGA-DATA verwendet zwei Arten von Verzeichnissen. Es gibt einen Satz von Systemverzeichnissen, deren Inhalt nie modifiziert werden darf und die von allen Nutzern verwendet werden. Diese Verzeichnisse werden mit der gelieferten WEGA-DATA-Version bereitgestellt. Ausserdem gibt es fuer jeden Anwendungsfall einen Satz von speziellen Verzeichnissen, die die fuer diesen bestimmten Anwendungsfall spezifischen Dateien enthalten. Fuer beide Arten von Verzeichnissen koennen Shell-Umgebungsvariable gesetzt werden, so dass die Verzeichnisse an geeigneter Stelle angelegt werden koennen. Dazu werden naehere Ausfuehrungen im Abschnitt ueber Umgebungsvariable gemacht. Die Systemverzeichnisse von WEGA-DATA haben folgenden Aufbau:



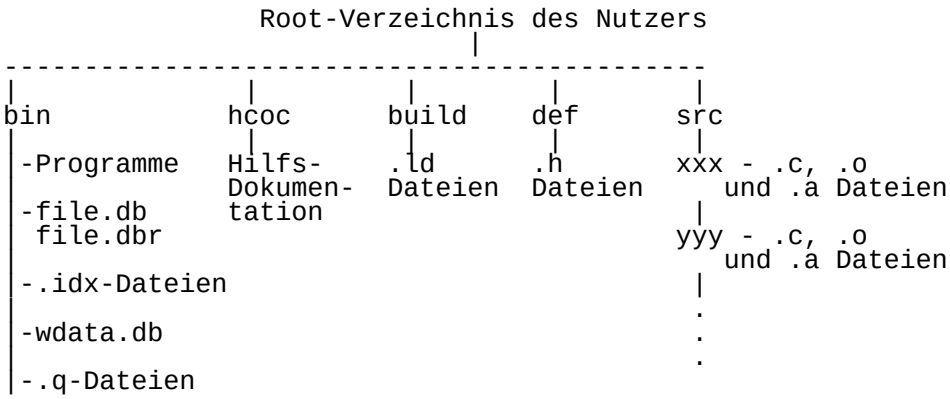
Diese Verzeichnisse befinden sich unter den Verzeichnissen, in denen WEGA-DATA installiert wurde. Nur die Verzeichnisse bin und lib werden waehrend der Laufzeit verwendet, die in include befindlichen Dateien werden nur bei der Kompilierung von Programmen verwenden. Das WEGA-DATA-Startprogramm namens wdata bildet eine Ausnahme. Dieses Programm wird von der Installationsprozedur in /usr/bin gebracht, so dass WEGA-DATA gestartet werden kann, ohne dass Veraenderungen an den Umgebungs- oder .profile-Dateien vorgenommen werden muessen.

Als Beispiel wird unten ein Verzeichnissystem fuer einen Anwendungsfall gezeigt. Dieses wurde entsprechend den Vereinbarungen zum Anlegen von WEGA-DATA-Verzeichnissen ange-

legt, die in Kapitel 3 erlaeutert werden.



Die typische Struktur eines Nutzer-Verzeichnissystems wuerde wie folgt aussehen. xxx und yyy sind die Namen von Unterverzeichnissen von src.



1.1.3 Umgebungsvariable

Die WEGA-Shell liefert eine geeignete Methode zum Setzen von Parametern, auf die von Shell-Skripten und Programmen aus zugegriffen werden kann. Diese als Umgebungsvariable bezeichneten Parameter koennen dann die Shell-Skripts und Programme anpassen, die auf sie zugreifen, und so zu der lokalen Umgebung einen Bezug herstellen. Umgebungsvariable treten in der Form name=wert auf, wobei name und wert beliebige Zeichenketten sind.

Ein gutes Beispiel fuer die Verwendung der Umgebungsvariablen ist die Art und Weise in der WEGA bestimmt, welche Verzeichnisse zum Auffinden von Programmen zu duchsuchen sind. Der Wert der Umgebungsvariablen namens PATH teilt der

Shell mit, welche Verzeichnisse auf der Suche nach einem bestimmten Programm zu durchsuchen sind. Der Standardwert fuer PATH ist :/bin:/usr/bin. Es soll angenommen werden, dass der Nutzer die WEGA-DATA-Programme in /z/wdata/bin bringen will. Dann muss dieses Verzeichnis in die Liste der durchsuchten Verzeichnisse aufgenommen werden. Dazu gibt man ein:

```
PATH=:/bin:/usr/bin:/z/wdata/bin
```

wenn davon ausgegangen wird, dass die Bourne-Shell verwendet wird. Oder man gibt ein:

```
PATH=$PATH:/z/wdata/bin
```

wobei $PATH bedeutet, dass der aktuelle Wert der Variablen PATH zu ersetzen ist. Soll dieser neue Suchpfad in Subshells wirksam werden, muss der Name exportiert werden. Das geschieht, indem eingegeben wird:

```
export PATH
```

Die WEGA-DATA-Installationsprozedur baut das Startprogramm (wdata) in /usr/bin auf, wobei alle "Stamm"-Umgebungsvariablen von WEGA-DATA vollstaendig gesetzt und exportiert sind (PATH, WDATA und BUDEV). Da /usr/bin Teil der vorgegebenen PATH-Liste ist, kann man wdata aus jedem beliebigen Verzeichnis heraus aufrufen und mit der Verwendung von WEGA-DATA ueber das eingebaute Menuesystem beginnen.

Wenn man von einem WEGA-DATA-Menue aus die Shell aufruft, um ein Programm ablaufen zu lassen, oder wenn man Programme ablaufen laesst, indem man sie aus einem Menue auswaehlt, braucht man die Grundumgebungsvariablen von WEGA-DATA nicht zu setzen, sie stehen allen WDATA "Tochter"-Prozessen zur Verfuegung, da sie exportiert wurden. Soll dagegen ausserhalb des Menuesystems ein von wdata verschiedenes WEGA-DATA-Programm verwendet werden, muessen die erforderlichen Umgebungsvariablen vom Nutzer selbst gesetzt und exportiert werden.

Im folgenden sind die WEGA-DATA-Programme aufgefuehrt, die der Nutzer moeglicherweise auf diese Weise ablaufen lassen moechte:

Shell-Kommandos von WEGA-DATA

Name	Abschnitt	Beschreibung
DBLOAD	3.5.5	Data Base Load. Gestattet das Laden von Daten aus einer ASCII-Datei in eine Datenbank.
LST	8	Listen-Prozessor. Gestattet die Erzeugung einfacher sortierter Dateiauflistungen mit Summen und

		Zwischensummen.
RPT	7	Reportgenerator. Gestattet die Erzeugung komplizierter Reporte unter Verwendung einer leistungsfaehigen, nichtprozeduralen Sprache.
SQL	6	Anfrage und Datenmanipulationssprache. Gestattet Anfragen an eine Datenbank-Datei und die Aktualisierung derselben, wobei eine leistungsfaehige, nichtprozedurale Sprache verwendet wird.
enter.ld	5.3.2.2	Shell-Kommandodatei. Gestattet die Erzeugung einer nutzerspezifischen Version von ENTER, in der die eigenen Host-Sprach-Funktionen des Nutzers enthalten sind.
ucc	10.1	C-Kompilierkommando von WDATA. Gestattet das Kompilieren von C-Programmen, die auf Datenbank-Datensaetze und -Feldnamen Bezug nehmen.
uld	10.1	Shell-Kommandodatei. Gestattet das Laden von Programmen, die das Host-Sprachen-Interface von WDATA verwenden.
uid	2.	Programm, mit dem die WEGA-Nutzer-ID des aktuellen Nutzers an die Standardausgabe ausgegeben wird.
rip	7.9.2	Dient zum Aufbau einer RPT-Eingabesektion. Vereinfacht das Schreiben von RPT-Skripten fuer ENTER-Bildmasken.
rpt.ld	7.4.6.2.4	Shell-Kommandodatei. Gestattet die Erarbeitung einer nutzerspezifischen Version von RPT, die Host-Sprach-Funktionen des Nutzers enthaelt.
runcobol.ld	Anhang A	Shell-Kommandodatei. Gestattet das Laden von COBOL-Programmen, die die Host-Sprach-Funktionen von WDATA verwenden.

In der Datei .profile des Nutzers koennen Umgebungsvariable gesetzt und exportiert werden, so dass sie bei Anmeldung durch Login automatisch gesetzt werden. Weitere Informationen dazu siehe WEGA-Handbuch, login(1) und profile(5).

Unten werden die Namen der WEGA-DATA-Umgebungsvariablen aufgelistet, die vom Nutzer gesetzt werden koennen, und es wird beschrieben, welche Wirkung sie haben. Alle Variablen muessen, damit sie wirksam werden koennen, exportiert werden.

PATH - Muss die Verzeichnisse enthalten, in denen sich die WEGA-DATA-Systemprogramme befinden. Ausserdem kann sie die Verzeichnisse enthalten, in denen die Anwendungsprogramme gespeichert sind, wenn diese nicht vom aktuellen Verzeichnis aus ausgefuehrt werden sollen. Der Standard-Suchpfad ist :/bin:/usr/bin.

WDATA - Pfad fuer das Systemverzeichnis lib von WEGA-DATA. Wird in dieser Variablen nicht das richtige Verzeichnis angegeben oder ist sie gleich Null, kann WEGA-DATA die .q-Dateien des Datenbank-Systems, die help-Dateien und die Systembibliotheken nicht finden.

TERM - Der Typ des Terminals laut Definition in der Datei termcap. Es gibt fuer diese Variable keinen Standardwert, so dass, soll ein korrekter Ablauf gewaehrleistet werden, diese Variable gesetzt werden muss.

BUDEV - Der vollstaendige Pfadname des Backup-Geraets des Systems, wobei es sich um ein Diskettenlaufwerk handelt. Es gibt keinen Standardwert, daher muss die Variable fuer alle Programme, die das Backup-Geraet verwenden, gesetzt werden. Sie wird normalerweise von der Installationsprozedur festgelegt.

TERMCAP - Der Pfadname der termcap-Datei. Standard ist /etc/termcap. Diese Variable braucht wahrscheinlich ueberhaupt nicht gesetzt werden.

DBPATH - Das Verzeichnis, in dem sich die System-Datendateien befinden. Dazu gehoeren die Dateien file.db, file.dbr, wdata.db und alle .idx-Dateien sowie alle .q-Dateien. Ausserdem erwartet WEGA-DATA, dass sich die Help-Dokumentationen des Nutzers im Verzeichnis $DBPATH/../hdoc befinden. Wird diese Variable gesetzt, muessen alle oben angefuehrten Dateien in den angegebenen Verzeichnissen abgelegt sein, da sie ansonsten nicht gefunden werden. Sind die Variablen DBPATH, PATH und WDATA richtig gesetzt, kann man den jeweiligen WEGA-DATA-Anwendungsfall von jedem Verzeichnis aus ausfuehren. Wird DBPATH nicht gesetzt, wird davon ausgegangen, dass sich alle oben angefuehrten Dateien im aktuellen Verzeichnis (oder, im Falle einer Help-Dokumentation, in ../hdoc) befinden, was der Normalfall ist. Achtung: Diese

Variable darf nicht gleich der leeren Zeichenkette sein!

EDIT - Der Name des bevorzugten Texteditors oder Wortprozessors. Standard ist der Editor vi, aber wenn dieser Editor nicht zur Verfuegung steht oder wenn ein anderer bevorzugt wird, kann eine entsprechende Aenderung vorgenommen werden. Der Editor wird von 'Enter Help Documentation' (Abschnitt 2.1.5) und von SQL (Abschnitt 6.3.3.3) zur Editierung von Anfragen verwendet.

SPOOLER - Der Name des System-Druckerspoolers. Standard ist der Standard-WEGA-Spooler namens lpr, der wahrscheinlich gar nicht geaendert werden braucht. Man kann jedoch bei Bedarf Optionen fuer den Spooler spezifizieren. Will man beispielsweise das von lpr normalerweise erzeugte Deckblatt unterdruecken, kann man SPOOLER folgendermassen setzen:

```
SPOOLER="lpr -nb"
```

Diese Umgebungsvariable muss gesetzt werden, um die Ausgabe, wie in den folgenden Beispielen angegeben, ueber eine Pipe oder mit Umlenken auszugeben:

```
cat dateiname | $SPOOLER
$SPOOLER < dateiname
$SPOOLER dateiname_1 dateiname_2
```

UUID - Der id-Kode (lt. Definition in 'Employee Maintenance') des aktuellen Nutzers. Diese Variable wird vom Menue-Handler gesetzt, so dass Nutzerprogramme darauf Bezug nehmen koennen.

UUACL - Das Zugriffsrecht des aktuellen WEGA-DATA-Nutzers fuer das aktuelle Programm (lt. Definition in 'Employee Maintenance'). Der Wert des Zugriffsrechtes ist eine Zeichenkette der Laenge 2 von "00" bis "15", mit der angegeben wird, welche Zugriffsrechte der Nutzer fuer das aktuelle Programm hat.

UNICAP - Der vollstaendige Pfadname der unicap-Datei (z.B. /usr/lib/meinunicap). Wenn UNICAP nicht gesetzt ist, ist der Standardspeicherplatz dieser Datei in $WDATA/unicap. Diese Datei wird von 'Paint Screen' (Abschnitt 4.1) verwendet, um die Terminaltastatur fuer die 'Paint Screen'-Kommandos zu konfigurieren. Mit WEGA-DATA wird auch eine Standard-unicap-Datei geliefert.

1.1.4 Die Datei termcap

Die Hardware-Funktionen der meisten Terminals koennen in der in /etc/termcap gespeicherten termcap-Datei beschrieben werden. Format und Verwendung dieser Datei werden im WEGA-Handbuch, termcap(5) beschrieben. Existiert keine solche Datei, kann die Beispieldatei, die mit der WEGA-DATA-Version im lib-Verzeichnis geliefert wird, herangezogen werden. WEGA-DATA verwendet diese Datei, um Informationen darueber zu erlangen, wie der Cursor positioniert wird, bestimmte Bildschirmmanipulationen ausgefuehrt und wie die ueber Tastatur erfolgten Eingaben zugeordnet werden.

WEGA-DATA funktioniert bereits mit nur vier der im WEGA-Handbuch beschriebenen Standard-Terminaleigenschaften. Ausserdem kann WEGA-DATA jedoch auch einige Eingaben verwenden, die dort nicht beschrieben sind. Diese Eingaben koennen verwendet werden, um die Ergebnisse von Tastatureingaben mit Parametern zu belegen, so dass die Verwendung der Eingabetasten am Terminal und die Ausgabecharakteristika des speziellen Terminaltyps nutzerspezifisch festgelegt werden. In der folgenden Tabelle werden alle WEGA-DATA termcap-Eintraege zusammengefasst und ihre Verwendung wird erklaert.

Terminalfunktionen (Standard)

Name	Typ	Auffuellen?	Beschreibung
cd	str	(P*)	Loeschen bis Ende Display
ce	str	(P)	Loeschen bis Zeilenende
cf	str		Loeschen der ungeschuetzten Felder
cl	str		Alles Loeschen
cm	str	(P)	Cursorbewegung
ps	str		Anfang des Schutzmodus'
pe	str		Ende des Schutzmodus'
so	str		Anfang des Hervorhebungsmodus', (meist Video invers)
se	str		Ende des Hervorhebungsmodus'
us	str		Anfang der Unterstreichung
ue	str		Ende der Unterstreichung
ru	str		Anfang Unterstreichung invers
re	str		Ende Unterstreichung invers
ws	str		Anfang des Schreibschutzes (halbe Intensitaet)
we	str		Ende des Schreibschutzes (wieder volle Intensitaet)

Tastaturzurodnung (nur bei WEGA-DATA)

Name	Typ	Auffuellen?	Beschreibung
Es	str		Anfang des Suchvorganges (mit ENTER Bildmaske) Standard ist CTRL/E
Ec	str		Loeschen der Eingabe (wird von

		ENTER verwendet) Standard: CTRL/Z
Ex	str	Beenden von ENTER, Standard: CTRL/X
Uf	str	Bewegung vorwaerts/nach unten in Bildmasken/Menues Standard:RETURN
Ub	str	Bewegung rueckwaerts/nach oben in Bildmasken/Menues Standard:CTRL/U

NOTWENDIGE TERMINALFUNKTIONEN

Name	Typ	Auffuellen?	Beschreibung
cd	str	(P*)	Loeschen bis Ende Display
ce	str	(P)	Loeschen bis Zeilenende
cl	str		Alles Loeschen
cm	str	(P)	Cursorbewegung

Die vier oben aufgefuehrten Charakteristika sind unbedingt fuer die Verwendung von WEGA-DATA erforderlich. Das Terminal muss mindestens diese vier Operationen ausfuehren koennen und sie muessen alle in der termcap-Datei fuer den verwendeten Terminaltyp eingetragen sein. Fuer die meisten Terminaltypen sind bereits die folgenden Operationen in termcap eingetragen: Cursorbewegung (cm), Alles Loeschen (cl) und Loeschen bis Zeilenende (ce). Fuer viele ist jedoch nicht Loeschen bis Ende Display (cd) eingetragen. Am einfachsten kann man ueberpruefen, ob diese Eigenschaft eingetragen ist, indem man 'Schema Maintenance' (Abschnitt 3.1) ausprobiert und nachsieht, ob noch etwas auf dem Bildschirm steht, wenn man von der Bildmaske fuer Datensatztypen zur Bildmaske fuer die Felder des Datensatztyps uebergeht. Bleibt noch etwas auf dem Bildschirm stehen, wird cd fuer das Terminal nicht ausgefuehrt und dieser Eintrag muss hinzugefuegt werden.

OPTIONELLE TERMINALFUNKTIONEN

Name	Typ	Auffuellen?	Beschreibung
cf	str		Loeschen der ungeschuetzten Felder
ps	str		Anfang des Schutzmodus'
pe	str		Ende des Schutzmodus'
ws	str		Anfang des Schreibschutzes (halbe Intensitaet)
we	str		Ende des Schreibschutzes (wieder volle Intensitaet)
so	str		Anfang des Hervorhebungsmodus', (meist Video invers)
se	str		Ende des Hervorhebungsmodus'
us	str		Anfang der Unterstreichung
ue	str		Ende der Unterstreichung
ru	str		Anfang Unterstreichung invers
re	str		Ende Unterstreichung invers

Unterstuetzt das Terminal die ersten fuenf der oben angefuehrten Funktionen, kann es Bildschirm-Prompter in geringer Intensitaet (im geschuetzten Modus) und Datenwerte in hoher Intensitaet (im ungeschuetzten Modus) anzeigen. Die Daten koennen mit einer einzigen Operation vom Bildschirm geloescht werden, indem alle ungeschuetzten Felder geloescht werden. Bei Terminals, die nicht ueber diese Eigenschaft verfuegen, werden Prompter und Daten in derselben Intensitaet angezeigt und jedes Datenfeld muss fuer sich geloescht werden. Diese fuenf Eintraege sind als eine Einheit zu behandeln, da man nicht einige dieser Charakteristika verwenden und andere weglassen kann.

Auch die folgenden sechs Ausgabecharakteristika sind nicht unbedingt erforderlich. Aber wenn das verwendete Terminal diese unterstuetzt, kann man Menuezeilen entweder in Negativanzeige oder im Unterstreichungsmodus anzeigen und die Dateneingabe-Prompter auf den SFORM-Bildmasken kann man entweder in Negativanzeige, im Unterstreichungsmodus oder in unterstrichener Negativanzeige anzeigen. Die Verwendung dieser Eigenschaften wird in Abschnitt 2.1.1 (Executable Maintenance), 4.1 (Paint Screen), 4.2 (Screen Entry) und 5.1 (ENTER Screen Registration) erlaeutert. Im allgemeinen wird die Negativanzeige fuer Prompter und Menuezeilen durch ~r und ~s, die Unterstreichung durch ~u bzw. ~v und die Unterstreichung der Negativanzeige durch ~w und ~x ein- bzw. ausgeschaltet.

TASTATURZUORDNUNG

Name	Typ	Auffuellen?	Beschreibung
Es	str		Anfang des Suchvorganges (mit ENTER Bildmaske) Standard ist CTRL/E
Ec	str		Loeschen der Eingabe (wird von ENTER verwendet) Standard: CTRL/Z
Ex	str		Beenden von ENTER, Standard: CTRL/X
Uf	str		Bewegung vorwaerts/nach unten in Bildmasken/Menues Standard:RETURN
Ub	str		Bewegung rueckwaerts/nach oben in Bildmasken/Menues Standard:CTRL/U

Mit den oben beschriebenen Eingaben kann man die von WEGA-DATA im Menue-Handler und den ENTER-Bildmasken verwendeten Standard-Steuerzeichen durch andere Steuerzeichen ersetzen. Damit kann man diese Steuerzeichen nutzerspezifisch veraendern. Die in der oben stehenden Tabelle beschriebenen Eintraege in die termcap-Datei muessen die Form name=^X haben, wobei name der aus zwei Zeichen bestehende, in der Tabelle eingetragene, Kode ist und ^X eine Steuerfunktion ist, die ausgeloest wird, indem CTRL und eine Buchstabentaste gedrueckt werden. Fuer diese Eingabe koennen mit Ausnahme von H,J und M alle Grossbuchstaben verwendet werden.

Wird Uf in die termcap-Datei eingetragen, wird damit be-

wirkt, dass eine andere Steuertaste die Funktionen ausfuehrt, die bei WEGA-DATA ansonsten RETURN uebernimmt. Das heisst, dass man zusaetzlich zum normalen RETURN eine weitere Taste hat, mit der zum "naechsten Feld" uebergegangen werden kann. Wird z.B. in die Terminalbeschreibung in der termcap Datei Uf=^P eingefuegt, hat das Druecken von CTRL/P bei der Verwendung von WEGA-DATA-Bildmasken und -Menues dieselbe Wirkung wie das Druecken der Taste RETURN.

Der Steuercode Ub wird verwendet, um der Steuertaste, die die Rueckkehr zum vorhergehenden Feld oder Prompter bewirkt, einen anderen Wert zuzuordnen. Standard ist CTRL/U. Mit dem Parameter Ub kann man jede beliebige Taste dafuer zuordnen.

Auch die mit dem Buchstaben E beginnenden Namen werden benutzt, um fuer ENTER Steuertasten festzulegen. Waehrend der Verwendung von ENTER wird dem Nutzer gemeldet, welche Tasten akzeptiert werden. Standards sind: Suchen: CTRL/E, Loeschen: CTRL/Z und Austritt aus dem Programm: CTRL/X. Diese Standards werden dann verwendet, wenn die Eintraege in der termcap-Datei nicht korrekt sind oder wenn fuer das verwendete Terminal keine anderen Eintraege in der termcap-Datei vorhanden sind.

AUFBAU

Es wird empfohlen, die termcap-Datei so anzulegen, dass sie nur Eingaben fuer die gerade fuer das System verwendeten Terminaltypen enthaelt. Werden noch andere Eintraege verwendet, sind die zu verwendenden an den Anfang der Datei zu bringen, da die Datei linear nach einem spezifischen Terminaltyp abgesucht wird. Der Suchvorgang wird erheblich beschleunigt, wenn der Umfang der zu bearbeitenden Daten reduziert wird.

Der Nutzer braucht nur die Umgebungsvariable TERM, wie in Abschnitt 1.1.3 beschrieben, auf den Typ des verwendeten Terminals und die Terminalgeschwindigkeit zu setzen. Bei Verwendung der Shell erfolgt dies folgendermassen:

```
TERM=xxx
export TERM
stty nnnn
```

wobei xxx der Kode des Terminaltyps in der termcap-Datei und nnnn die Baud-Rate ist, mit der gearbeitet wird (normalerweise 9600). Verlangt das Terminal bestimmte Verzoegerungen, um ordentlich funktionieren zu koennen, ist die Terminalgeschwindigkeit, wie oben gezeigt, explizit zu setzen. Die Geschwindigkeit extb wird von der termcap-Software nicht anerkannt, d.h. die entsprechenden Nullzeichen werden nicht ausgegeben. Beide Parameter koennen in der vom Nutzer verwendeten Datei .profile gesetzt werden, so dass der Nutzer sie nicht bei jedem Login neu zu setzen braucht.

2. DER MENUE-HANDLER - MENUH

Menue-Handler und Sicherheitssystem, MENUH, bieten dem Anwender eine vollstaendige, anwenderfreundliche Umgebung fuer das WEGA-DATA-System. Er wird in der Entwicklungsphase eines Projektes verwendet, wenn WEGA-DATA zur Schaffung einer Anwendung benutzt wird und wenn das abgeschlossene Anwendungssystem in Betrieb genommen wird. Somit werden sowohl die Anwendungsentwicklung als auch der Anwendungsbetrieb vom Menues aus betrieben.

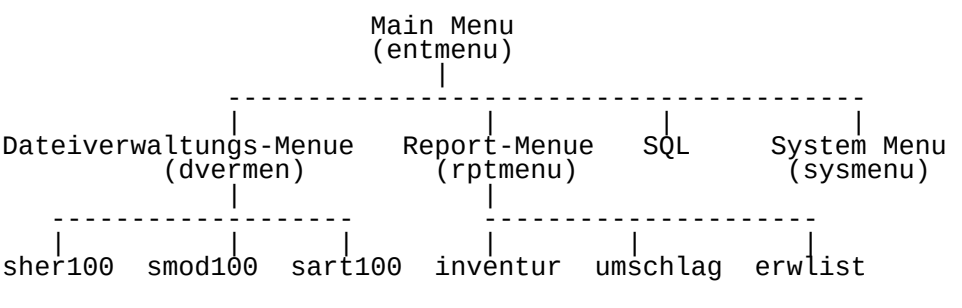
Alle Dialoge, die der Nutzer mit WEGA-DATA unterhaelt, werden ueber MENUH gesteuert. MENUH aktualisiert das Datenwoerterbuch, um sich ueber die Menues, Anwenderfunktionen, ENTER-Bildmasken und Nutzer eines Anwendersystems auf dem Laufenden zu halten. MENUH weiss auch, wo die Archive fuer die einzelnen Nutzerprogramme zu finden sind, welche Programme zusammen in einer einzigen ablauffaehigen Speicherabbild-Datei geladen werden und welche SFORM-Bildmaske angezeigt werden soll, wenn ein Programm seinen Ablauf beginnt.

Im urspruenglichen Datenwoerterbuch sind nur die WEGA-DATA-Programme und -Menues und eine Nutzer-Identifikation enthalten. MENUH wird waehrend der Entwicklung einer Anwendung zur Schaffung der Menuestruktur fuer dieses Systems verwendet.

MENUES

Ein Menue ist ein Satz von Auswahlmoeglichkeiten, der dem Nutzer angeboten wird, damit er eine der Optionen auswaehlen kann. Jede Auswahlmoeglichkeit stellt entweder eine Funktion, ein Programm oder ein anderes Menue mit Auswahlmoeglichkeiten dar, das vom Nutzer gewaehlt werden kann. MENUH gestattet die Schaffung und Verwaltung eines Satzes von hierarchischen Menuebaeumen. Hier ein Beispiel fuer einen Menue/Programm-Baum fuer eine Teilmenge eines Lagerbestandssystems eines Grosshandelslagers.

Beispielmenue - Grosshandelslager-Verwaltungssystem



Bei Eintritt ins oberste Menue werden folgende Optionen

angeboten:

```
[entmenu]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                               Main Menu

1. Dateiverwaltungs-Menue

2. Report-Menue

3. SQL - Query/DML Language

4. System Menu

SELECTION: _
```

In diesem Moment stehen vier Optionen offen. Wird 1 eingegeben, wird das Dateiverwaltungs-Menue folgendermassen angezeigt:

```
[dvermen]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                        Dateiverwaltungs-Menue

1. Hersteller Verwaltung

2. Modell Verwaltung

3. Artikel Verwaltung


SELECTION: _
```

Will man ins vorhergehende Menue zurueckkehren, wird CTRL/U eingegeben. Wenn es vorher kein Menue gab (d.h. wenn das vorliegende Menue das Ausgangsmenue ist), wird bei Druecken von CTRL/U die Steuerung an die Shell oder einen anderen Vater-Prozess zurueckgegeben. In diesem Fall wird durch CTRL/U erneut das 'Main Menu' angezeigt.

Wird, nach erfolgter Rueckkehr, 2 eingegeben, wird das Report-Menue aufgerufen. Wird nun eine 1 eingegeben, laeuft das Programm Lagerbestandsueberblick ab. Es generiert einen Ueberblick ueber die im Lager vorhandenen Artikel.

```
[rptmenu]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                             Report-Menue

1. Lagerbestandsueberblick

2. Artikelumschlags-Report

3. Lagereingangsdaten


SELECTION: _
```

Neben der Moeglichkeit, in oben beschriebener Weise die Menues durchzugehen, koennen Programme und Menues direkt abgearbeitet werden. Zu diesem Zweck wird jedem Menue und jedem Programm ein Name gegeben, der zur Kennzeichnung desselben dient. Der Name wird bei Ablauf des Menues oder Programms immer in der oberen linken Ecke des Bildschirms angezeigt. Dann koennen Programme und Menues einfach abgearbeitet werden, indem man statt der Nummer ihren Namen eingibt. Befindet man sich zum Beispiel im 'Main Menu', kann man den Lagerbestandsueberblick auch direkt durch seinen Namen (inventur) aufrufen. Erfahrene Nutzer koennen damit viel Zeit sparen.

Dem Nutzer werden nur Menues und Programme angeboten, die seinen Zugriffsrechten entsprechen. Natuerlich verhindert MENUH dann auch, dass Nutzer ueber den Namen auf Programme oder Menues zugreifen, fuer die sie kein Zugriffsrecht haben. Im naechsten Abschnitt soll mehr darueber gesagt werden.

NUTZER

Bei Eintritt ins WEGA-DATA-System, muss der Nutzer einen Nutzer-id und ein Passwort angeben. Dieses wird von MENUH benutzt, um zu bestimmen, welches Menue dem Nutzer als Eintrittsmenue angeboten wird und welche Optionen auf diesem Menue und auf den folgenden zulaessig sind. Ausserdem uebergibt MENUH bei Ablauf eines Programms diesem Programm die Zugriffsrechte des aktuellen Nutzers.

Jeder Nutzer besitzt fuer jedes Programm und Menue ein festgelegtes Zugriffsrecht. Entweder hat der Nutzer keinen Zugriff (Standard) oder er hat das Recht abzufragen, zu aendern, zu loeschen und/oder hinzuzufuegen. Die vier Moeglichkeiten des Zugriffs koennen beliebig miteinander kombiniert werden. Mit anderen Worten: ein Nutzer kann fuer ein Programm das Recht abzufragen und zu loeschen oder zu modifizieren und hinzuzufuegen oder nur abzufragen haben usw. Dieses Recht wird als eine Zahl weitergegeben, die vom Nutzerprogramm interpretiert werden muss, das dann nur die

angegebenen Operationen zulassen darf. Die Dienstprogrammfunktion priamd (siehe Abschnitt 10.3) ist zu diesem Zweck in der WEGA-DATA-Nutzerbibliothek enthalten.

Der Nutzer hat zusaetzlich zu seinen Zugriffsrechten ein Eintrittsmenue. Das ist das erste Menue, das der Nutzer nach dem Login in WEGA-DATA sieht. Das heisst, er sieht nur den Teil des Menuebaums, der fuer ihn zutrifft. Da Programme auf verschiedenen Menues erscheinen koennen, kann der Systemverwalter die Umgebung eines jeden Nutzers genau nach den Erfordernissen individuell gestalten.

Aus Gruenden der Zweckmaessigkeit wird jeder Nutzer einer Gruppe zugeordnet. Eine Gruppe besteht aus mehreren Nutzern mit aehnlichen Zugriffsrechten. Dadurch kann man einer Gruppe einen Standardsatz an Zugriffsrechten zuordnen und dann fuer die einzelnen Nutzer nur die Abweichungen davon eingeben. So werden zum Beispiel die Beschaeftigten, die die Daten eingeben (Datenverarbeitungs-Spezialisten) als Gruppe eingegeben, da sie im wesentlichen fuer die gleichen Programme Zugriff haben. Einem neuen Beschaeftigten wird man jedoch nicht gleich Zugriff zu den kritischeren Programmen geben. Ausserdem wird ein neuer Beschaeftigter fuer bestimmte Programme nur Abfragerechte haben. Der Verantwortliche hat dagegen Zugriff auf zwei zusaetzliche Programme und erhaelt fuer alle Programme das Recht des Loeschen.

Im WEGA-DATA-System spielt der Superuser eine besondere Rolle. Dieser hat automatisch Zugriff zu allen Menues und Programmen. Der Kode des Superusers, der Superuser-id, sein Passwort und Eintrittsmenue werden in 'System Parameter Maintenance' (siehe Abschnitt 2.1.7) eingegeben. Der id des Superusers ist standardmaessig als su festgelegt und das Eintrittsmenue als entmenu.

HILFSDOKUMENTATION - HELP

MENUH ermoeglicht fuer jedes Programm und jedes Menue eine Hilfsdokumentation. Diese wird durch 'Enter Help Documentation' eingegeben (Abschnitt 2.1.5). Wird der Prompter SELECTION: angezeigt, kann das Wort help eingegeben werden und die fuer die aktuelle Bildmaske zutreffende Hilfsdokumentation wird angezeigt. Es ist auch moeglich, help n einzugeben, wobei n eine gueltige Auswahlnummer ist. Dann wird die, dem ausgewaehlten Menue oder Programm zugeordnete, Hilfsdokumentation angezeigt. Es ist auch moeglich, help xxxx einzugeben, wobei xxxx ein Menue- oder Programmname ist. Dann wird die dem benannten Programm zugeordente Hilfsdokumentation angezeigt.

AUSFUEHRBARE DATEIEN

Eine ausfuehrbare Datei ist entweder ein WEGA-Shellskript oder eine Menge von einem oder mehreren kompilierten Programmen, die in eine ausfuehrbare Datei gelinkt wurden.

Handelt es sich bei der ausfuehrbaren Datei um eine Speicherabzugs-Datei (core image file), muss jedes Programm in dieser Datei einen Eintrittspunkt (d.h. eine C-Funktion) haben, dessen Name gleich dem Namen ist, unter dem es unter MENUH registriert ist. In einer solchen ausfuehrbaren Datei ist die Hauptroutine eine Funktion namens sysrecev. Die Funktion sysrecev holt die Argumente vom Menue-Handler, bestimmt den Terminaltyp, erfuellt andere Aufgaben zur Organisation des Programmablaufs und ruft dann das korrekte Programm auf. Indem verschiedene Programme in einen Speicherabzug geladen werden, koennen sowohl Platten- als auch Hauptspeicherplatz gespart werden.

Besteht die ausfuerbare Datei nur aus einem Nutzerprogramm, braucht sysrecev nicht verwendet werden. Der Menue-Handler fuehrt einfach das Programm des Nutzers aus und dieses Programm ist verantwortlich fuer die Schaffung seiner eigenen Umgebung.

PROGRAMME

Die Programme, die in jeder ausfuehrbaren Datei sind, werden mit WEGA-DATA registriert. Vom Nutzer ist der Prompter anzugeben, der auf jedem Menue, auf dem das Programm auftritt, erscheinen soll. Dieser Prompter wird auch auf dem Bildschirm oben angezeigt, wenn der Menue-Handler das Programm ausfuehrt. Optionell kann der Nutzer die SFORM-Bildmaske angeben, die angezeigt werden soll, bevor das Programm ablaeuft und das Verzeichnis angeben, in dem sich das Objektkode-Archiv des Programmes befindet. Letzteres wird von 'Program Loading' verwendet (siehe 2.1.6).

PROGRAMMABARBEITUNG UEBER DEN MENUE-HANDLER

Waehlt der Nutzer aus einem Menue ein Programm, indem er entweder den Namen oder die Nummer des Programms eingibt, sieht der Menue-Handler im Datenwoerterbuch nach, welche ausfuehrbare Datei dieses Programm enthaelt. Dann startet der Menue-Handler die ausfuehrbare Datei unter Verwendung des WEGA-Systemrufs execvp(2). Ist das Programm in einer ausfuehrbaren Datei enthalten, die sysrecev nicht verwendet, uebergibt der Menue-Handler der ausfuehrbaren Datei keine Parameter. Er startet sie einfach. Derartige ausfuehrbare Dateien koennen nur ein einziges Programm enthalten, es sei denn, der Nutzer hat auf eine andere Art programmiert, wie das Programm (mit anderen Moeglichkeiten als einer Parameterliste) bestimmen kann, welches Programm gestartet werden soll.

Ist das Programm jedoch in einer ausfuehrbaren Datei, die sysrecev verwendet, konstruiert der Menue-Handler eine Parameterliste, die folgende Elemente enthaelt:

```
ENAME T index zugriffs-ebene sprach-kode bm-name
```

Die Parameter haben folgende Bedeutung:

Parameter	Verwendung	Verweis
ENAME	Name der ausfuehrbaren Datei	Executable Maintenance Abschnitt 2.1.1
T	Wird nicht verwendet, ist historisch bedingt vorhanden	Konstanter Wert
index	Einspungnummer in der ausfuehrbaren Datei des zu startenden Programms	Executable Maintenance Abschnitt 2.1.1
zugriffs-ebene	Ganze Zahl zwischen 0 und 15, die festlegt, ob der entspr. Nutzer Hinzufuegunge-, Abfrage-, Aenderungs- oder Loeschrechte hat.	Group Maintenance Abschnitt 2.1.3 oder Employee Maintenance Abschnitt 2.1.4
sprach-kode	Systemsprach-Kode	System Parameter Maintenance, Abschn. 2.1.7
bm-name	Name der diesem Programm zugeordneten Bildmaske	Executable Maintenance Abschnitt 2.1.1

Wir wollen kurz einen einfachen Beispielaufruf betrachten.

```
ENAME T 0 15 EN sher100
```

Das bedeutet, dass die Standard-Systemsprache (Englisch) unveraendert beibehalten wird. Hat man sich als Superuser von WDATA angemeldet und waehlt man aus einem Menue ename, wird es gestartet. Wenn ename abzulaufen beginnt, laeuft zuerst die Funktion sysrecev ab, da dies die Funktion main ist. Sie ordnet den Parametern folgendermassen globale Variable zu:

Parameter	Variable
T	nicht gespeichert
index	nicht gespeichert
acess-level	int logacl
language-code	char langtp[2]
screen-name	char *_savscr

Dann benutzt sie index zur Ausfuehrung der entsprechenden Nutzerfunktion, die als Programm-Einsprungstelle definiert

wurde. Das geschieht, indem eine von 'Program Loading' (Abschnitt 2.1.6) erarbeitete Tabelle von Pointern auf Funktionen verwendet wird. Der Name der Tabelle ist menucall. Sie ist in der Quelldatei prgtab.c definiert. Der Quellkode fuer unser Beispiel koennte so aussehen:

```
int her100(), art100();
int ( menucall[])() = {
                        her100,
                        art100
                       };
```

Aus dieser Tabellendefinition ist zu ersehen, dass her100 die Einsprungadresse 0 und art100 die Einsprungstelle 1 hat.

VERWENDUNG VON WEGA-DATA ALS SHELL

Wenn man will, kann man WEGA-DATA als Shell fuer WEGA-Nutzer verwenden. Dann sieht der Nutzer nach Login in WEGA, die Login-Bildmaske von WEGA-DATA anstelle des Shell-Prompters. Man kann aber auch den Login-Bildschirm von WEGA-DATA voellig umgehen, so dass der Nutzer nach dem Login in WEGA sein erstes WEGA-DATA-Menue sieht. Um das zu tun, muss man jedoch mit dem Format und der Verwendung der WEGA Passwort- und Profile-Dateien vertraut sein. Diese werden im WEGA-Handbuch unter login(1), passwd(1), passwod(5) und profile(5) beschrieben. Man muss auch mit den in Abschnitt 1.1.3 erlaeuterten Umgebungsvariablen von WEGA-DATA vertraut sein.

Der Menue-Handler besteht aus einem "startup"-Shellskript und aus drei Programmen, die alle in diesem Abschnitt beschriebenen Programme ausfuehren. Das startup-Shellskript heisst wdata. Diesen Namen gibt man auch unter der Shell ein, wenn die Login-Bildmaske von WEGA-DATA angezeigt werden soll und mit der Arbeit an einer WEGA-DATA-Anwendung begonnen werden soll. Dieses Shell-Skript setzt die entsprechenden Umgebungsvariablen, exportiert sie und startet das "eigentliche" WEGA-DATA - eine ausfuehrbare Datei im Verzeichnis bin von WEGA-DATA - das die Menues anzeigt und die Zugriffsrechte ueberwacht. Die ausfuehrbare Datei MENUH enthaelt alle Programme, die in Abschnitt 2.1 beschrieben sind, waehrend die ausfuehrbare Datei RMENU die in Abschnitt 2.2 beschriebenen Programme enthaelt.

Damit WEGA-DATA als die "Shell" fuer einen Nutzer verwendet werden kann, muss der entsprechende Eintrag in der WEGA-Passwort-Datei erfolgen und eine Datei .profile angelegt werden, um die entsprechende Umgebung zu schaffen und WEGA-DATA ausfuehren zu koennen. Die Passwort-Datei verfuegt ueber einen Eintrag, der das Home-Verzeichnis fuer den Nutzer angibt. Wird die Umgebungsvariable DBPATH nicht verwendet, muss dieses Verzeichnis das Verzeichnis bin des Anwenders sein. (siehe Abschnitt 1.1.2). Will sich zum Beispiel WEGA-Nutzer Hugo, der die Nutzer-id 10 hat, anmel-

den und das Einarbeitungsbeispiel als seine Shell benutzen, und ist angenommen die Verzeichnisstruktur gleich der in Abschnitt 1.1.2 beschriebenen und das Einarbeitungsbeispiel befindet sich im Dateisystem /z, dann waere der entsprechende Eintrag in die Passwort-Datei folgender:

```
hugo::10:10:/z/wdata/beisp/bin:
```

Es ist zu beachten, dass die Shellspezifikation offen geblieben ist, so dass die Standardshell die Datei .profile im Home-Verzeichnis abarbeiten wird. Dann muesste die Datei .profile folgende Kommandos enthalten (unter der Annahme, dass der Nutzer ein VT100-kompatibles Terminal hat).

```
TERM=PV
export TERM
wdata
kill -9 0
```

Es ist zu beachten, dass die Shell-Kommandodatei wdata die korrekten Werte einfuehrte und die Umgebungsvariablen PATH, WDATA und BUDEV exportierte. Das letzte Kommando in profile ist das Kommando kill -9 0, das den Nutzer wieder in die Login-Bildmaske von WEGA bringt, wenn er aus WEGA-DATA austritt. Wird diese Datei .profile im Verzeichnis bin des Nutzers abgelegt, werden alle Nutzer, fuer die dieses Verzeichnis ihr Home-Verzeichnis ist, die Login-Bildmaske von WEGA-DATA fuer diese Anwendung sehen. Sie koennen dann ihre jeweilige Nutzer-id fuer WDATA eingeben und mit der Arbeit an dieser Anwendung beginnen.

Man kann aber auch Vereinbarungen treffen, dass, anstelle der Login-Bildmaske von WDATA, bei Anmeldung des Nutzers das Eintrittsmenue dieses Anwendungssystems angezeigt wird. Das kann erreicht werden, indem man mittels 'Employee Maintenance' (Abschnitt 2.1.4) die WEGA-Nutzer-id's auch als WDATA-Nutzer-ID's festlegt. Im oben angefuehrten Beispiel wuerde das bedeuten, dass Hugos Nutzer-id unter WEGA-DATA 10 waere und dass das auch seine Nutzer-id unter WEGA waere. Dann muss man nur die Zeile aendern, die WEGA-DATA startet, um die Identifikation des aktuellen Nutzers zu uebergeben. WEGA-DATA bekommt diese von dem Shell-Kommando uid, das diese auf die Standardausgabe ausgibt. Das heisst, die Datei .profile sieht folgendermassen aus:

```
TERM=PV
export TERM
PATH=/z/wdata/beisp:$PATH
export PATH
wdata 'uid'
kill -9 0
```

Alle Nutzer, die das Verzeichnis beisp als Home-Verzeichnis nutzen wollen und eine Nutzer-id haben, die gleich ihrer WEGA-Nutzer-id ist, koennen nun gleich nach der Anmeldung in WEGA-DATA auf dem Bildschirm ihr WEGA-DATA Eintrittsme-

nue sehen. Nutzer ohne gueltige Nutzer-id fuer WEGA-DATA erhalten keine Genehmigung fuer Login.

EDITIEREN VON DATEIEN

Hinter dem Prompter 'SELECTION:' kann man das Kommando edit eingeben, wodurch das schnelle Editieren einer Datei ermoeglicht wird. Gibt man nur edit ein, wird man noch nach dem Dateinamen gefragt. Bei Eingabe von edit dateiname wird der aktuelle Editor (siehe Umgebungsvariable EDIT) mit diesem Namen gestartet. Drueckt man als Antwort auf die Frage nach dem Dateinamen einfach RETURN, wird der Editor ohne Dateinamen gestartet. Bei Druecken von CTRL/U kehrt der Nutzer unmittelbar ins aktuelle Menue zurueck. Bei Austritt aus dem Editor wird erneut das aktuelle Menue angezeigt.

2.1 Bildmasken des Menue-Handlers

2.1.1 Verwaltung von ausfuehrbaren Dateien

Dieses Programm gestattet dem Nutzer das Hinzufuegen, Modifizieren und Loeschen von ablauffaehigen Dateien. Es gestattet dem Nutzer auch, alle in einer gegebenen ablauffaehigen Speicherabzugsdatei enthaltenen Programme zu spezifizieren.

```
[execmnt]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                       Executable Maintenance

EXECUTABLE'S NAME: ''''''''
USES SYSRECEV ? '
PROGRAMS:
amd NAME      HEADING                    SCREEN    DIRECTORY

  0 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  1 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  3 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  4 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  5 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  6 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  7 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  8 """""""""  """""""""""""""""""""""""  """""""""  """""""""
  9 """""""""  """""""""""""""""""""""""  """""""""  """""""""

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE a
```

```
''''''''  Eingabebereich fuer ausfuehrbare Dateien
"""""""""  Eingabebereich fuer Programme
```

Mit dieser Bildmaske kann man ausfuehrbare Dateien und Pro-

gramme innerhalb dieser ausfuerbaren Dateien erfragen, hinzufuegen, aendern und loeschen. Eine ausfuehrbare Datei kann mehrere Einsprungstellen (entry points) enthalten, von denen jede ein anderes Programm starten kann. Das trifft allerdings normalerweise nur auf Nutzerprogramme zu und nicht auf Kommandodateien. Es folgt eine Erklaerung aller Prompter.

[I]INQUIRE, [A]DD, [M]ODIFY, [D]ELETE

Mit diesem Prompter kann man einen Betriebsmodus fuer die ausfuehrbare Datei waehlen. Die verschiedenen Modi haben die folgende Bedeutung:

i - Abfragemodus; zeigt eine ausfuehrbare Datei, die bereits mit dem Menue-Handler, MENUH, registriert ist.

a - Ergaenzungsmodus; gestattet die Registrierung einer neuen ausfuehrbaren Datei in MENUH.

m - Aenderungsmodus; gestattet die Aenderung von HEADING, SCREEN und DIRECTORY fuer Programme innerhalb einer existierenden ausfuehrbaren Datei oder gestattet das Hinzufuegen und Loeschen von Programmen.

d - Loeschmodus; gestattet das Loeschen einer existierenden ausfuehrbaren Datei und aller dazu gehoerigen Programme.

EXECUTABLE'S NAME:
Eine aus 8 Zeichen bestehende Zeichenkette, die der Name einer WEGA-Datei ist, die eine ausfuehrbare Datei ist. Dieser Dateiname wird an die Shell uebergeben, wenn ein Programm innerhalb der ausfuehrbaren Datei vom Nutzer gewaehlt wird.

USES SYSRECEV ?
Ein Y bedeutet, dass die ausfuehrbare Datei mit der "Haupt"-WEGA-Routine namens sysrecev geladen wurde, die die von MENUH uebergebenen Argumente interpretiert (Standard-Argumentliste). Trifft nur auf Nutzerprogramme zu. Bei Shell-Skripten oder bei ausfuehrbaren Dateien, die nur ein Programm enthalten und es nicht brauchen, verweist ein N darauf, dass sysrecev nicht verwendet wird.

amd
Diese Spalte wird zur Eingabe eines Kommandos verwendet, das fuer die Ausfuehrung einer Operation am aktuellen Programm verwendet wird. Vermittels CTRL/U und RETURN kann man den Cursor im mehrzeiligen Dateneingabebereich auf dem Bildschirm nach unten bzw. oben bewegen. Die Cursorstellung markiert das aktuelle Programm. Hier die gueltigen Kommandos:

a - Der ausfuehrbaren Datei soll ein neues Programm

hinzugefuegt werden. Diese Antwort ist nur auf der ersten leeren Zeile des mehrzeiligen Bereiches gueltig. Das heisst, dass man vor Hinzufuegen eines neuen Programms den Cursor in die erste Leerzeile bringen muss.

m - HEADING, SCREEN oder DIRECTORY wird fuer das aktuelle Programm modifiziert. Soll NAME modifiziert werden, ist die Zeile zu loeschen und dann wieder einzufuegen.

d - Das aktuelle Programm soll geloescht werden.

q - Der Cursor wird aus dem mehrzeiligen Dateneingabebereich in den Prompter 'USES SYSRECEV ?' gebracht.

NAME

Ein aus 8 Zeichen bestehender Name des Programms innerhalb der ausfuehrbaren Datei. Es handelt sich hierbei um den Namen, der in Beantwortung des Menue-Prompters SELECTION: zur Ausfuehrung des Programms eingegeben wird. Wenn die ausfuehrbare Datei ein Nutzerprogramm ist und der Name ein Eintrittspunkt in die ausfuehrbare Datei ist, muss dieser Name mit dem Namen in der Programmiersprache (C, FORTRAN, PASCAL, usw.) des Nutzerprogramms uebereinstimmen.

HEADING

Eine aus 36 Zeichen bestehende Zeichenkette, die auf der dritten Zeile des Bildschirms angezeigt wird, wenn das Programm ablaeuft bzw. auf den Menues, wenn dieses Programm eine Option ist. Es kann sich um alphanumerische Zeichen oder um Umschaltzeichen handeln, die bewirken, dass der Prompter in einem speziellen Modus angezeigt wird, sofern die termcap-Tabelle fuer das aktuelle Terminal diesen Modus zulaesst (siehe Abschnitt 1.1.4). Das Umschaltzeichen Zeichen ist die Tilde "~". Die folgenden speziellen Modi sind zulaessig:

~r - Anfang Video invers
~s - Ende Video invers
~u - Anfang der Unterstreichung
~v - Ende der Untersteichung
~w - Anfang Video invers unterstrichen
~x - Ende Video invers unterstrichen

Beginnt man eine Ueberschrift in einem speziellen Anzeigemodus, muss dieser auch beendet werden, da sonst ein unkorrektes Bild entsteht. Der spezielle Modus wird nur in Menues angezeigt und nicht auf der dritten Zeile, wenn das Programm ablaeuft.

SCREEN

Der Name der SFORM-Bildmaske, die von sysrecev angezeigt wird, bevor das Nutzerprogramm gestartet wird. Es

handelt sich hierbei um einen optionalen Eintrag.

DIRECTORY
Bei Nutzerprogrammen handelt es sich hierbei um den Namen des WEGA-Verzeichnisses, in dem sich der Quellkode des Programms befindet. Das Shell-Skript, das die WEGA-DATA-Programme laedt (uld), geht ebenfalls davon aus, dass das die Programme enthaltende Archiv sich in diesem Verzeichnis befindet. Der relative Pfad ../src/ wird dem hier eingegebenen Verzeichnisnamen vorangestellt. Wird kein Eintrag vorgenommen, verwendet uld die ersten 3 Zeichen des Programmnamens als Namen des Verzeichnisses. Ist z.B. der Programmname best100, sucht uld in ../src/bes nach dem Archiv, in dem best100 enthalten ist.

2.1.2 Verwaltung der Menues

Dieses Programm erlaubt es, Menues zu erstellen, zu modifizieren oder aus dem Datenwoerterbuch zu loeschen. Nach Aufruf von 'Menu Maintenance' erscheint folgende Bildmaske:

```
[menumnt]                        WDATA SYSTEM
                              24 JUL 1986 - 15:25
                               Menu Maintenance

NAME: ######## HEADING: ##################################
amd  LINE  MENU/PROG  M/P   PROMPT
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
 "     ""  """""""""   "    """""""""""""""""""""""""""""""
[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

######## Eingabebereich fuer Menues
""""""""" Eingabebereich fuer Menuezeilen

Mit dieser Bildmaske kann man Menues und Menuezeilen abfragen, hinzufuegen, aendern und loeschen. Es ist ein Dateneingabebereich fuer Menuezeilen vorhanden. Hier nun eine Erklaerung der einzelnen auf dem Bildschirm erscheinenden

Prompter und Spaltenueberschriften.

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE:

Mit diesem Prompter kann man den Betriebsmodus fuer das Programm waehlen. Die verschiedenen Modi haben folgende Bedeutung:

i - Abfragemodus, ermoeglicht die Betrachtung eines existierenden Menues.

a - Ergaenzungsmodus, ermoeglicht die Erstellung neuer Menues.

m - Aenderungsmodus, ermoeglicht Hinzufuegen, Aendern und Loeschen von Zeilen in einem existierenden Menue.

d - Loeschmodus, ermoeglicht das Loeschen vollstaendiger Menues.

NAME

Ein aus 8 Zeichen bestehender Menuename. Der Name darf fuer andere Menues, Bildmasken und Programme nicht noch einmal verwendet werden.

HEADING

Ist eine aus 34 Zeichen bestehende Zeichenkette, die auf der dritten Zeile des auf dem Schirm angezeigten Menues erscheint oder die in anderen Menues erscheint, wenn dort dieses Menue zur Auswahl angeboten wird. Man kann Text oder Umschaltzeichen eingeben, wodurch bestimmt wird, das die Ueberschrift in einem bestimmten Modus angezeigt werden soll, sofern der termcap-Eintrag fuer das aktuelle Terminal diesen Modus zulaesst (siehe Abschnitt 1.1.4). Das Umschaltzeichen ist die Tilde (~). Die folgenden Modi sind zulaessig:

~r - Video invers (Anfang)
~s - Video invers (Ende)
~u - Anfang der Unterstreichung
~v - Ende der Unterstreichung
~w - Anfang Video invers unterstrichen
~x - Ende Video invers unterstrichen

Schaltet man fuer eine Ueberschrift einen bestimmten Anzeigemodus ein, muss man diesen am Schluss der Ueberschrift wieder beenden. Sonst erscheint die Bildmaske nicht richtig auf dem Bildschirm. Der Modus erscheint nur auf den Menues, er erscheint nicht in der dritten Zeile, wenn das Menue selbst aktiv ist.

amd

Diese Spalte wird zur Eingabe eines Befehls verwendet, der zur Ausfuehrung einer Operation auf der aktuellen Menuezeile dient. Durch Druecken von CTRL/U und RETURN kann man den Cursor im mehrzeiligen Dateneingabebereich

des Bildschirms nach unten und oben bewegen. Die Cursorstellung markiert die aktuelle Menuezeile. Die gueltigen Befehle sind:

a - Hinzufuegen einer neuen Zeile im Menue. Ist nur gueltig in der ersten freien Zeile des mehrzeiligen Bereiches. Das heisst, dass man vor Hinzufuegen einer neuen Menuezeile den Cursor zunaechst in die erste freie Zeile bringen muss.

m - Zum Modifizieren von LINE oder MENU/PROG fuer die aktuelle Menuezeile.

d - Loeschen der aktuellen Menuezeile.

q - Der Dateneingabebereich wird verlassen und der Cursor wird auf den Prompter HEADING: gesetzt.

LINE
Die Stelle im Menue, an der die aktuelle Menuezeile erscheinen soll. Es kann eine beliebige Zahl von 1 bis 16 eingegeben werden und die Menuezeilen werden entsprechend auf dem Schirm verschoben.

MENU/PROG
Name des Menues, des Programms, der ENTER- oder SQL-Bildmaske, die bei Wahl dieser Menuezeile aktiviert werden.

M/P
Dieses Feld ist nur zur Anzeige vorgesehen. In ihm wird der Typ der obigen Eingabe angezeigt. Es gibt folgende Moeglichkeiten:

M - Die aktuelle Menuezeile startet ein Menue
P - die aktuelle Menuezeile startet ein Programm
E - die akt. Menuezeile startet eine ENTER-Bildmaske
S - die aktuelle Menuezeile startet eine SQL-Bildmaske

PROMPT
Dieses Feld ist nur zur Anzeige vorgesehen. In ihm wird angegeben, welcher Text im Menue fuer die aktuelle Menuezeile erscheinen soll. Fuer Menues wird der Prompter in 'Menu Maintenance' (dieses Programm) eingegeben, fuer Programme in 'Executable Maintenance' (execmnt), fuer ENTER-Bildmasken wird er in 'ENTER Screen Registration' (entmnt) eingegeben und fuer SQL-Bildmasken in 'SQL Screen Registration' (ssqlmnt).

2.1.3 Verwaltung von Gruppen

Unter 'SELECTION:' wird 3 eingetragen, um 'Group Maintenance' aufzurufen. Dieses Programm gestattet die Aenderung, Hinzufuegung und das Loeschen von Gruppenzugriffsrechten im Datenwoerterbuch. Diese Rechte bestimmen,

welche Programme und Menues die Gruppe benutzen darf und auf welcher Ebene von der Gruppe eine Aktualisierung vorgenommen werden darf. Die Bildmaske fuer dieses Programm sieht folgendermassen aus:

```
[grpmnt]                        WDATA SYSTEM
                              24 JUL 1986 - 15:25
                               Group Maintenance

GROUP ID: ####  NAME: ##############################

SYSTEM ENTRY PT: #######  - #

ACCESS LEVELS:
LN MENU/PROG M/P IN AD MO DE LN MENU/PROG M/P IN AD MO DE
'' '''''''' '  '  '  '  '  '' '''''''' '  '  '  '  '

"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "
"" """""""" "  "  "  "  "  "" """""""" "  "  "  "  "

[I]NQUIRE, [A]DD,  [M]ODIFY, [D]ELETE: _
```

```
######## Eingabebereich fuer DVS
'''''''' Eingabebereich fuer Zugriffsrechte
"""""""" Darstellungsbereich fuer Zugriffsrechte
```

Mit dieser Bildmaske kann man Gruppen von Beschaeftigten und Zugriffsrechte abfragen, ergaenzen, aendern und loeschen. Es ist ein Dateneingabebereich fuer Gruppen von Beschaeftigten, ein Dateneingabebereich fuer Zugriffsrechte und ein Aufrufbereich fuer Zugriffsrechte vorhanden. Hier eine Erklaerung der auf dem Bildschirm erscheinenden einzelnen Prompter und Spaltenueberschriften:

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE:

Mit diesem Prompter kann man einen Modus fuer den Bereich der Dateneingabe fuer Gruppen von Beschaeftigten waehlen. Die einzelnen Modi haben folgende Bedeutung:

i - Abfragemodus, ermoeglicht die Sichtung der Zugriffsrechte einer vorhandenen Gruppe von Beschaeftigten.

a - Ergaenzungsmodus, ermoeglicht das Hinzufuegen neuer Gruppen von Beschaeftigten und deren Zugriffsrechte.

m - Aenderungsmodus, ermoeglicht Ergaenzung, Aenderung und

Loeschen von Zugriffsrechten fuer eine existierende Gruppe von Beschaeftigten.

d - Loeschmodus, ermoeglicht das Loeschen einer existierenden Gruppe von Beschaeftigten mit deren Zugriffsrechten

GROUP ID
Ein aus 4 Zeichen bestehender Kode zur Kennung der Gruppe von Beschaeftigten.

NAME
Ein aus 30 Zeichen bestehender Name fuer die Gruppe von Beschaeftigten. Dieser Name wird nur zur Kennung und Dokumentation verwendet.

SYSTEM ENTRY PT
Der Name des Eintrittsmenues, -programms, der ENTER- oder SQL-Bildmaske, den die Beschaeftigten dieser Gruppe beim Login in den Menue-Handler sehen. Wird nur ein Programm, eine ENTER- oder SQL-Bildmaske angegeben, koennen die Beschaeftigten nur diese eine Funktion ausfuehren.

[N]ext page, [P]rev page, [A]dd line oder number:

Prompter fuer Seitenaufrufbereich; wird angezeigt nachdem der Prompter 'SYSTEM ENTRY PT' beantwortet wurde. Mit diesem Prompter kann man den Modus fuer die Seitenbank waehlen. Die einzelnen Auswahlmoeglichkeiten haben folgende Bedeutung:

n - Die naechste Seite der Zugriffsrechte wird angezeigt.

p - Die vorhergehende Seite der Zugriffsrechte wird angezeigt.

a - Gestattet das Hinzufuegen neuer Zugriffsrechte, indem der Cursor auf den Eingabebereich fuer Zugriffsrechte gesetzt wird.

1-999 - Es wird die Seite angezeigt, die das angegebene Zugriffsrecht enthaelt und der Cursor wird auf dieses Zugriffsrecht gesetzt, so dass geaendert oder geloescht werden kann, wie im folgenden beschrieben.

Spalte links von LN
Dieser Bereich wird zur Eingabe eines Befehls verwendet, durch den eine Operation am aktuellen Zugriffsrecht ausgefuehrt wird. Stimmt mit der mit CMD ueberschriebenen Spalte auf dem 'Schema Entry Screen' ueberein. Die Zeile ist jedoch so voll, dass fuer die Bezeichnung dieser Spalte kein Platz mehr war. Durch Druecken von CTRL/U und RETURN kann der Cursor in dieser Spalte auf und abbewegt werden. Die Cursorstellung kennzeichnet die aktuelle Zeile. Die gueltigen Befehle sind:

m - Aendern von INQ, ADD, MOD oder DEL fuer das aktuelle Zugriffsrecht, wenn dieses fuer ein Programm, eine ENTER- oder eine SQL-Bildmaske erteilt wurde.

d - Loeschen des aktuellen Zugriffsrechts.

q - Der Promter fuer den Seitenaufruf erscheint erneut unten auf dem Schirm.

LN
Ist eine vom System zugewiesene Zeilennummer. Wird zur Bezugnahme auf das Zugriffsrecht verwendet, falls dieses geaendert werden soll.

MENU/PROG
Der Name eines Menues, eines Programms, einer ENTER- oder SQL-Bildmaske, zu dem die Beschaeftigten dieser Gruppe Zugriff haben.

M/P
Es handelt sich um ein Feld, das nur zur Anzeige verwendet wird. Hier wird angegeben, worauf sich die Eingabe in der obenstehenden Spalte bezog. Es gibt folgende Moeglichkeiten:

M - Das aktuelle Zugriffsrecht ist fuer ein Menue.
P - Das aktuelle Zugriffsrecht ist fuer ein Programm.
E - Das aktuelle Zugriffsrecht bezieht sich auf eine ENTER-Bildmaske.
S - Das aktuelle Zugriffsrecht bezieht sich auf eine SQL-Bildmaske.

INQ
Ein in dieser Spalte stehendes Y bedeutet, dass die Angehoerigen dieser Gruppe den Abrufmodus fuer das Programm, die ENTER- oder SQL-Bildmaske verwenden koennen. Ein N bedeutet, dass dies nicht der Fall ist.

ADD
Ein in dieser Spalte stehendes Y bedeutet, dass die Angehoerigen dieser Gruppe den Ergaenzungsmodus fuer das Programm, die ENTER- oder SQL-Bildmaske verwenden koennen. Ein N bedeutet, dass dies nicht der Fall ist.

MOD
Ein in dieser Spalte stehendes Y bedeutet, dass die Angehoerigen dieser Gruppe den Aenderungsmodus fuer das Programm, die ENTER- oder SQL-Bildmaske verwenden koennen. Ein N bedeutet, dass dies nicht der Fall ist.

DEL
Ein in dieser Spalte stehendes Y bedeutet, dass die Mitglieder dieser Gruppe den Loeschmodus fuer das Programms, die ENTER- oder SQL-Bildmaske verwenden koennen. Ein N bedeutet, dass dies nicht der Fall ist.

2.1.4 Verwaltung von Beschaeftigten

Mit diesem Programm kann man die Beschaeftigten einzeln verwalten. Es koennen neue Beschaeftigte hinzugefuegt werden, bereits eingetragene geaendert oder geloescht werden. Ausserdem kann man eine Liste verwalten, auf der die Unterschiede zwischen den Zugriffsrechten der Beschaeftigten und denen der Gruppe, der sie angehoeren, aufgefuehrt sind. Wird fuer einen Beschaeftigten angegeben, dass er Zugriff auf ein Programm oder ein Menue hat, besitzt diese Angabe gegenueber der fuer die Gruppe den Vorrang. Die Bildmaske fuer dieses Programm sieht folgendermassen aus:

```
[empmnt]                        WDATA SYSTEM
                             24 JUL 1986 - 15:25
                            Employee Maintenance

LOGIN ID: #### NAME: ################################
GROUP ID: #### NAME: ################################
PASSWORD: ######## SYSTEM ENTRY PT: ########  - #

ACCESS LEVELS DIFFERENT FROM GROUP:
 LN   MENU/PROG   M/P   ACCESS?   INQ  ADD  MOD  DEL
 ''   ''''''''    '     '         '    '    '    '

 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "
 ""   """""""""    "     "         "    "    "    "

[I]NQUIRE,  [A]DD, [M]ODIFY, [D]ELETE: a
```

```
######## Eingabebereich fuer Gruppenmitglieder
'''''''' Eingabebereich fuer Zugriffsrechte
""""""""" Anzeigebereich fuer Zugriffsrechte
```

Mit dieser Bildmaske kann man Beschaeftigte und Zugriffsrechte abfragen, hinzufuegen, aendern und loeschen. Ausserdem kann man Unterschiede zwischen den Zugriffsrechten der Gruppe, zu der der Beschaeftigte gehoert, und den Zugriffsrechten des Einzelnen festlegen. Mit anderen Worten: Die Rechte der Gruppe legen fuer den Beschaeftigten die grundsaetzlichen Zugriffsrechte fest, aber unter Verwendung dieses Programms kann man fuer den Einzelnen Abweichungen festlegen.

Es ist ein Dateneingabebereich fuer Beschaeftigte, ein Dateneingabebereich fuer Zugriffsrechte und ein Datenaufbereitungsgebiet fuer Zugriffsrechte vorhanden. Im folgenden

werden die Prompter und die Spaltenueberschriften erklaert:

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE:

Mit diesem Prompter kann der Betriebsmodus fuer den Dateneingabebereich fuer Beschaeftigte auf der Bildmaske gewaehlt werden. Die verschiedenen Modi haben folgende Bedeutung:

i - Abfragemodus, ermoeglicht Einsicht in die Zugriffsrechte eines Beschaeftigten.

a - Ergaenzungsmodus, ermoeglicht das Hinzufuegen neuer Beschaeftigter und zugehoeriger Zugriffsrechte.

m - Aenderungsmodus, ermoeglicht Hinzufuegen, Aendern und Loeschen von Zugriffsrechten der Beschaeftigten.

d - Loeschmodus, ermoeglicht das Loeschen eines Beschaeftigten und aller ihm zugeordneten Zugriffsrechte.

LOGIN ID
Ein aus vier Zeichen bestehender Kode zur Identifizierung des Beschaeftigten. Es ist der gleiche Kode, der im LOGIN-Prompter erscheint, wenn die Anmeldung zur Arbeit mit WEGA-DATA erfolgt.

NAME
Eine aus 30 Zeichen bestehende Zeichenkette, die fuer den Namen des Beschaeftigten steht. Dient nur der Dokumentation und Identifizierung.

GROUP
Ein aus 4 Zeichen bestehender Kode zur Kennung der Gruppe, zu der der Beschaeftigte gehoert.

NAME
Ein Feld, das ausschliesslich der Anzeige dient und den Namen der Beschaeftigtengruppe zeigt.

PASSWORD
Ein aus 8 Zeichen bestehendes Passwort, das der Beschaeftigte hinter dem Prompter PASSWORD: eintragen muss, bevor er WEGA-DATA benutzen kann.

SYSTEM ENTRY PT
Der Name des Eintrittsmenues, -programms, der ENTER- oder SQL-Bildmaske, das der Beschaeftigte beim Anmelden (LOGIN) sieht. Ist fuer die Gruppe standardmaessig vorgegeben, kann aber bei Bedarf fuer den Einzelnen geaendert werden.

[N]ext page, [P]rev page, [A]dd line, or number:

Prompter fuer Seitenaufrufbereich. Wird nach Beantwortung des Prompters 'SYSTEM ENTRY PT' angezeigt. Mit dem Prompter

kann man den Betriebsmodus fuer den Seitenaufrufbereich waehlen. Die einzelnen Auswahlmoeglichkeiten haben folgende Bedeutung:

n - Die naechste Seite der Zugriffsrechte wird angezeigt.

p - Die vorhergehende Seite der Zugriffsrechte wird angezeigt.

a - Gestattet das Hinzufuegen neuer Zugriffsrechte, indem der Cursor auf den Eingabebereich fuer Zugriffsrechte gestellt wird.

1-999 - Die Seite wird angezeigt, die das angegebene Zugriffsrecht enthaelt und der Cursor wird auf dieses Zugriffsrecht gestellt, so dass es, wie im folgenden beschrieben, geaendert oder geloescht werden kann.

Spalte links von LN
Dieser Bereich wird zur Eingabe eines Befehls verwendet, durch den eine Operation am aktuellen Zugriffsrecht ausgefuehrt wird. Wird in gleicher Weise verwendet, wie die entsprechende Spalte im 'Group Maintenance Screen'. Durch Druecken von CTRL/U und RETURN kann der Cursor in dieser Spalte auf und ab bewegt werden. Die Cursorstellung kennzeichnet die aktuelle Zeile. Die gueltigen Befehle sind:

m - Aendern von ACCESS, INQ, ADD, MOD, oder DEL im aktuellen Zugriffsrecht.

d - Loeschen des aktuellen Zugriffsrechts.

q - Erneute Anzeige des Seitenaufrufprompters unten auf der Seite.

LN
Ist eine vom System zugewiesene Zeilennummer. Wird zur Bezugnahme auf das Zugriffsrecht verwendet, falls dieses geaendert werden soll.

MENU/PROG
Der Name eines Menues, eines Programms, einer ENTER- oder SQL-Bildmaske, zu dem der Beschaeftigte ein anderes Zugriffsrecht besitzt als die anderen der Gruppe.

M/P
Es handelt sich um ein Feld, das nur zur Anzeige verwendet wird. Hier wird angegeben, welcher Art die Eingabe in der oben stehenden Spalte war. Es gibt folgende Moeglichkeiten:

M - Das aktuelle Zugriffsrecht ist fuer ein Menue.
P - Das aktuelle Zugriffsrecht ist fuer ein Programm.
E - Das aktuelle Zugriffsrecht ist fuer eine ENTER-Bildmaske.

S - Das aktuelle Zugriffsrecht ist fuer eine SQL-Bildmaske.

ACCESS?
Ein in dieser Spalte stehendes Y bedeutet, dass der betreffende Beschaeftigte Zugriff zum angegebenen Menue, Programm, zur ENTER- oder SQL-Bildmaske haben wird. N bedeutet, dass dies nicht der Fall ist.

INQ
Ein in dieser Spalte stehendes Y bedeutet, dass der betreffende Beschaeftigte den Abfragemodus fuer das Programm, die ENTER- oder SQL-Bildmaske verwenden kann. N bedeutet, dass dies nicht der Fall ist.

ADD
Ein in dieser Spalte stehendes Y bedeutet, dass der Beschaeftigte den Ergaenzungsmodus fuer das Programm, die ENTER- oder SQL-Bildmaske verwenden kann. N bedeutet, dass dies nicht der Fall ist.

MOD
Ein in dieser Spalte stehendes Y bedeutet, dass der betreffende Beschaeftigte den Aenderungsmodus fuer das Programm, die ENTER- oder SQL-Bildmaske verwenden kann. N bedeutet, dass dies nicht der Fall ist.

DEL
Ein in dieser Spalte stehendes Y bedeutet, dass der betreffende Beschaeftigte den Loeschmodus fuer das Programm, die ENTER- oder SQL-Bildmaske verwenden kann. N bedeutet, dass dies nicht der Fall ist.

2.1.5 Erstellen von Hilfsdokumentationen

Unter Verwendung von 'Enter Help Documentation', enthdoc, kann man zur Erlaeuterung beliebiger Menues, Programme, ENTER- oder SQL-Bildmasken eine Hilfsdokumentation eingeben. Die Textdatei wird mit dem vom Nutzer bevorzugten WEGA-Texteditor eingegeben. Standardeditor ist der Editor vi; man kann jedoch unter Verwendung der Umgebungsvariablen EDIT auch einen anderen Editor verwenden. Weitere Informationenen siehe Abschnitt 1.1.3, Umgebungsvariable. Durch dieses Programm eingegebene Textdateien werden auf dem Bildschirm durch help programmname, help menue oder help nummer angezeigt.

Nachdem diese Option ueber Menue ausgewaehlt wurde, wird nach dem Namen des Menues oder des Programms gefragt, dem die Datei zugeordnet werden soll. Existiert die Datei bereits, wird sie zur Modifizierung eroeffnet. Andernfalls wird eine neue Datei angelegt. Fuer den Zugriff auf die Datei werden die Standard-Editorkommandos verwendet. Nach Abschluss der Editierungsarbeiten wird die Datei auf uebliche Art gespeichert.

Die Textdateien, die mit diesem Programm bearbeitet werden, werden in ../hdoc gespeichert (oder in $DBPATH/../hdoc falls DBPATH gesetzt wurde - siehe Abschnitt 1.1.3, Umgebungsvariable). Damit man Hilfsdokumentationen erstellen kann, muss dieses Verzeichnis existieren. Die System-Dokumentation fuer Programme und Menues von WEGA-DATA ist unter dem Bibliotheks-Verzeichnis des Systems in lib/hdoc (oder in $WDATA/hdoc gespeichert, wenn WDATA gesetzt wurde - siehe Abschnitt 1.1.3, Umgebungsvariable). Das Programm meldet sich wie folgt auf dem Bildschirm:

```
[enthdoc]                    WDATA SYSTEM
                        24 JUL 1986 - 15:25
                    Enter Help Documentation

MENU/PROGRAM: _
```

MENU/PROGRAM
: Der Name eines Menues, eines Programms, einer ENTER- oder SQL-Bildmaske. enthdoc startet den Editor vi oder den durch die Umgebungsvariable EDIT bestimmten Editor mit dem Dateinamen ../hdoc/xxxxxxxx oder $DBPATH/../hdoc/xxxxxxxx falls DBPATH definiert ist, wobei xxxxxxxx der Name des Menues, Programms, der ENTER- oder SQL-Bildmaske ist.

Nach Beendigung der Arbeit mit dem Editor wird der oben stehende Prompter erneut angezeigt und man kann den Namen eines naechsten Menues oder Programms, einer naechsten ENTER- oder SQL-Bildmaske eingeben. Durch Druecken von CTRL/U kehrt man ins vorangegangene Menue zurueck.

2.1.6 Laden von Programmen

Der Menue-Handler kann bei einem Anwendungssystem zur Steuerung des Ladens ablauffaehiger Dateien verwendet werden. Das ist moeglich, weil jedes Programm und jede ablauffaehige Datei unter Verwendung von 'Executable Maintenance' in MENUH gespeichert werden kann. Dieser Prozess wird in Kapitel 2. allgemein und Abschnitt 2.1.1 erlaeutert. Es ist zu beachten, dass ein Programm, wenn es unter Verwendung dieser Option geladen werden soll, nicht in einem Menue erscheinen muss.

Das Laden eines Programms kann auf drei Arten geschehen. Moeglich sind "automatisch", "halbautomatisch" und "per Hand". Mit diesem Begriffen wird charakterisiert, wieviel Arbeit der Nutzer ausserhalb des Menue-Handlers zu leisten hat, bevor ein Programm geladen wird. Beim automatischen Laden folgt man den Vereinbarungen bzgl. Benennung von Dateien und Verzeichnisstruktur, so dass MENUH eine Kommandodatei zum Laden erzeugen, die Reihenfolge von Program-

men innerhalb der ablauffaehigen Datei bestimmen und die Kommandodatei abarbeiten kann. Beim halbautomatischen Laden kann der Nutzer seine eigene Kommandodatei zum Laden anlegen, aber er ueberlaesst MENUH das Ordnen der Programme und die Abarbeitung der Kommandodatei. Im manuellen Modus kann der Nutzer nach eigenem Gutduenken vorgehen.

Der Vorteil des automatischen und halbautomatischen Ladens besteht darin, dass MENUH alle Programme und ausfuehrbaren Dateien in einem System kennt. Der 'Executable Listing Report' liefert dann ein vollstaendiges Verzeichnis des gesamten Anwendungssystems, wodurch die Ueberschaubarkeit erleichtert wird. Ausserdem wird damit eine einheitliche Verzeichnisstruktur im Entwicklungsbereich durchgesetzt, was die Verwaltung von Programmen und die Einarbeitung neu am Projekt arbeitender Programmierer erleichtert.

Das Laden von Programmen wird folgendermassen durchgefuehrt. Unter der MENUH-Bildmaske wird 'Program Loading' ausgewaehlt und dann der Name des zu ladenden Programmes eingegeben. Dieser Name muss mit dem Namen uebereinstimmen, mit dem das Programm unter 'Executable Maintenance' eingegeben wurde. Es ist zu beachten, dass dieser Name nicht der Name der ablauffaehigen Datei selbst ist. Wenn die Kommandodatei vollstaendig ist, kehrt der Cursor zu dem Prompter zurueck und es kann eine neue ablauffaehige Datei geladen werden.

Wenn jemand anders etwas ins gleiche Verzeichnis laedt, wartet das Programm, bis dieser Ladevorgang beendet ist und setzt erst dann seine Arbeit fort. Damit soll vermieden werden, dass eine Verwechslung bzgl. der Datei a.out entsteht, die WEGA als Standardausgabedatei des Ladens verwendet. Dies geschieht durch Anlegen einer Sperrdatei chen, muss diese Datei vor einem erneuten Laden geloescht werden. Das geschieht aehnlich wie bei der Sperrdatei des Spoolers von WEGA, /usr/spool/lpd/lock. Es ist zu beachten, dass dies nicht zutrifft, wenn der Nutzer der Superuser ist, da man dann eine Datei immer anlegen kann, auch wenn diese bereits existiert.

Das Programm bestimmt die ausfuehrbare Datei, die das gewuenschte Programm enthaelt, indem es im Datenwoerterbuch nachsieht, den Namen der ausfuehrbaren Datei in Kleinbuchstaben umsetzt und dann .ld anhaengt. Dadurch entsteht der Name der Lade-Kommandodatei.

Man kann die Kommandodatei entweder selbst anlegen (halbautomatisch) oder diese Aufgabe dem Programm 'Program Loading' ueberlassen (automatisches Laden). Wird die Datei nicht gefunden, erzeugt das Programm eine Kommandodatei, die diese Aufgabe erfuellt. Folgende Vereinbarungen werden bei der Erzeugung der Lade-Kommando-Datei verwendet:

1. Die Kommandodatei uld im Verzeichnis fuer ausfuehrbare Dateien von WEGA-DATA wird verwendet. $1 ist der Name

der ausfuehrbaren Datei. uld wird genauer unter Abschnitt 10.1, Kompilieren und Laden von Programmen beschrieben.

2. Wenn die ausfuehrbare Datei sysrecev als ihre Hauptfunktion verwendet, was der Fall sein muss, wenn sie mehr als nur ein einzelnes Programm enthaelt, wird prgtab.o als $2 verwendet. Diese Datei wird automatisch vom Programm erzeugt, indem es die unter 'Executable Maintenance' eingegebene Struktur der ausfuehrbaren Datei untersucht. Die Datei enthaelt eine Tabelle von Pointern, die auf die Funktionen zeigen, die Bestandteil der ausfuehrbaren Datei sind, und zwar fuer jede Funktion je einen Pointer. Der Name der Tabelle ist menucall. In der Einleitung wurde kurz ein Beispiel angegeben.

   Die Datei wird automatisch kompiliert und es entsteht eine .o-Datei und nach dem Laden werden sowohl .c- als auch .o-Dateien geloescht.

3. Fuer jedes Programm in einer ausfuehrbaren Datei wird angenommen, dass es ein Archiv gibt, das die Funktionen fuer jedes Programm enthaelt. Es wird angenommen, dass sich die Archive in src-Unterverzeichnissen befinden, die entsprechend der in Abschnitt 2.1.1 gegebenen Beschreibung benannt sind. Alle Archive im src-Unterverzeichnis werden einbezogen, aber nuetzlicherweise legt man nur ein Archiv pro Verzeichnis an.

4. Schliesslich ist noch das Archiv ../src/util/util.a in der Kommandodatei enthalten. Bei der Entwicklung von Software-Systemen werden normalerweise Funktionen so geschrieben, dass sie nicht nur in einem Programm verwendet werden koennen. Diese Funktionen werden alle im Archiv util gesammelt, so dass nur eine Kopie des Quellkodes vorhanden ist.

Wird 'Program Loading' gestartet, erscheint die folgende Bildmaske:

```
[lfilegen]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                            Program Loading

PROGRAM NAME: _
```

Unter 'PROGRAM NAME:' wird der Name eines Programms, das, wie vorhin erlaeutert, unter 'Executable Maintenance' registriert wurde, eingegeben. Die das Programm enthaltende ausfuehrbare Datei wird geladen und der Cursor kehrt zum Prompter zurueck, so dass die Eingabe eines weiteren Programmnamens moeglich ist.

2.1.7 Verwaltung der Systemparameter

Das Programm enthaelt einige Grundparameter fuer das Menuesystem. Mit diesem Programm koennen Informationen ueber Superuser, Systemtitel und die Mnemoniks fuer die Monate des Jahres geaendert werden. Da es zur Aenderung der Superuser-Identifikation und des Passwortes verwendet werden kann, sollte seine Verwendung mittels 'Employee Maintenance' eingeschraenkt werden (siehe Abschnitt 2.1.4). Im Menue MENUH wird 'System Parameter Maintenance' (parmnt) gewaehlt, um das Programm zu starten. Der Bildschirm sieht dann folgendermassen aus:

```
[parmnt]                    WDATA SYSTEM
                        24 JUL 1986 - 15:25
                    System Parameter Maintenance

SUPER USER ID  : su

PASSWORD       : su

ENTRY POINT    : entmenu

SYSTEM HAEDING : WDATA SYSTEM

LANGUAGE       : EN

MONTH MNEMONICS: JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC

BLOCKS / VOLUME: 179
```

Die verschiedenen Eintraege, die auf dem Bildschirm angezeigt sind, sind alle Systemparameter, die geaendert werden koennen. Die Eintraege haben folgende Bedeutung:

SUPER USER ID
: Ist die Login-Identifikation des Superusers. Kann eine beliebige, aus acht Zeichen bestehende Zeichenkette sein.

PASSWORD
: Ist das Passwort des Superusers. Eine beliebige, aus 8 Zeichen bestehende, Zeichenkette ist zulaessig.

ENTRY POINT
: Der Name des Menues, das der Superuser nach seinem Login sieht.

SYSTEM HEADING
: Eine aus 50 Zeichen bestehende Zeichenkette, die auf allen Bildmasken und Menues des Datenbanksystems auf der ersten Zeile steht.

LANGUAGE
Ist ein aus zwei Zeichen bestehender Kode, der von MENUH an die Programme gegeben wird. sysrecev bringt den Wert in eine globale Variable (char langtp[2]), wodurch Nutzerprogramme Zugriff haben. Dieser Parameter wird gegenwaertig nicht verwendet.

MONTH MNEMONICS
Ist eine aus 36 Zeichen bestehende Zeichenkette, die fuer jeden Monat des Jahres je 3 Zeichen enthaelt. Diese Zeichenkette wird zur Anzeige der zweiten Zeile auf den Bildmasken und Menues im Datenbanksystem verwendet.

BLOCKS/VOLUME
Ist eine ganze Zahl, die die Dienste 'Write Data Base Backup', 'Read Data Base Backup' und 'Reconfigure Data Base' darueber informiert, wieviele 4k-(4096-Byte)-Bloecke auf einen Datentraeger des Backup-Geraetes des Systems passen. Als Backup-Medium des Systems wird normalerweise die Floppy-Disk verwendet.

Der Cursor steht zu Beginn auf dem Promter 'SUPER USER ID:'. RETURN bewegt den Cursor abwaerts und CTRL/U nach oben. Die Eingabe von CTRL/U hinter dem ersten Prompter bewirkt die Rueckkehr zum vorangegangenen Menue. Das Aendern von Parametern geschieht einfach durch die Eingabe der neuen Werte gefolgt von RETURN.

2.2 Reporte des Menue-Handlers

In diesem Abschnitt werden die Standardreporte des Datenwoerterbuchs beschrieben, die von WEGA-DATA bereitgestellt werden. Da das Datenwoerterbuch selbst eine WEGA-DATA-Datenbank ist, koennen natuerlich Anfragen an dieses Datenwoerterbuch gerichtet werden, die alle gewuenschten Informationen bringen. Dieser Satz von Reporten wird aber wahrscheinlich die meisten Beduerfnisse befriedigen, die fuer die Auflistung von Informationen aus dem Datenwoerterbuch bestehen.

Aus dem 'System Menu' heraus wird 'MENUH Report Menu' aufgerufen (11 oder rptmenu). Danach hat man die im folgenden beschriebenen Optionen.

2.2.1 Listen der ausfuehrbaren Dateien

Dieses Programm liefert einen Report aller im Datenwoerterbuch vorhandenen ausfuehrbaren Dateien. Ausserdem werden alle in den ausfuehrbaren Dateien enthaltenen Programme aufgelistet. Die Auflistung erfolgt, wenn das entsprechende Programm im 'Report Menu' gewaehlt wird (1 oder lexec). Es gibt keine Optionen. Der Report ist 93 Zeichen breit, aber wurde hier aus Platzgruenden in der Breite reduziert.

DATE : 07/24/86 TIME: 17:11:21 PAGE: 1

SCHEMA REPORTS
Executable Listing

EXECUTABLE	SYSFLAG	PROGRAM	HEADING	IDX	DIRECTORY SCREEN
BUDB	N	budb	Write Data Base Backup	0	
CDSF	N	cdsf	Create Default Screen Form	0	
DBSTATS	N	dbstats	Data Base Statistics	0	
ENTER		her	Hersteller Maintenance	0	her
		modell	Modell Maintenance	0	modell
		sher100	Hersteller Verwaltung	0	sher100
		smod100	Modell Verwaltung	0	smod100
		art	Artikel Maintenance	0	art
		sart100	Artikel Verwaltung	0	sher100
		best	Bestellung Maintenance	0	best
ENTMNT	Y	entmnt	ENTER Screen Registration	0	
FSEC	Y	fldsec	Field Security Maintenance	0	
HTS	N	hts	Hash Table Statistics	0	
IDXMNT	N	idxmnt	Index Maintenance	0	
INVENTUR	N	inventur	Lagerbestandsueberblick	0	
LST	N	lst	Listing Processor	0	
MENUH	Y	execmnt	Executable Maintenance	0	
		menumnt	Menu Maintenance	1	
		grpmnt	Group Maintenance	2	
		empmnt	Employee Maintenance	3	
		enthdoc	Enter Help Documentation	5	
		lfilegen	Program Loading	4	
		parmnt	System Parameter Maintenance	6	
PAINT	Y	paint	Paint Screen	0	
PROCPASS	Y	procpass	Process Field Passwords	0	
REDB	N	redb	Read Data Base Backup	0	
REKEY	N	rekey	Hash Table Maintenance	0	
REPOINT	N	repoint	Explicit Relationship Maintenance	0	
RMENU	Y	lexec	Executable Listing	0	
		lmenu	Menu Listing	1	
		lgrp	Group Listing	2	
		lemp	Employee Listing	3	
		prthdoc	Print Help Documentation	4	

```
DATE : 07/24/86  TIME: 17:11:21                          PAGE:   2
                    SCHEMA REPORTS
                  Executable Listing
          SYSFLAG                                        DIRECTORY
EXECUTABLE    PROGRAM HEADING                      IDX SCREEN

  SCHENT    Y schent  Schema Maintenance           0
  SCHLST    Y schlst  Schema Listing               0
  SCOM      Y scom    Reconfigure Data Base        0
              crdb    Create Data Base             1
  SFORM     Y sfmaint Screen Entry                 0
              sfproc  Process Screen               1
              sfsamp  Test Screen                  2
              sfrep   Screen Reports               3
              sfrestr Restore Screen               4
              sfslist List Screens                 5
  SH        Y sh      WEGA Shell                   0
  SQL       N sql     SQL - Query/DML Language     0
  SSQL        erwlist Lagereingangsdaten           0 erwlist
  SSQLMNT   Y ssqlmnt SQL Screen Registration      0
  SYS920    N sys920  Data Base Test Driver        0
  UMSCHLAG  N umschlag Artikelumschlags-Report     0
  VOLMNT    Y volmnt  Volume Maintenance           0
  beet100   N beet100                              0
  best100   Y best100 Bestellungs-Verwaltung       0 sbest100
                                                       best

TOTAL EXECUTABLES:   30   PROGRAMS:   52
```

2.2.2 Listen der Menues

Dieses Programm erzeugt eine Liste aller im Datenwoerterbuch vorhandenen Menues. Die Menues werden in alphabetischer Reihenfolge ihrer Namen aufgelistet und dahinter stehen jeweils die im Menue enthaltenen Auswahlmoeglichkeiten. Der Report wird erzeugt, wenn er aus dem 'Report Menu' (2 oder lmenu) heraus aufgerufen wird. Es gibt keine Optionen. Der Report ist 79 Spalten breit und wurde hier aus drucktechnischen Gruenden in der Breite reduziert.

DATE : 07/24/86 TIME : 17:11:35 PAGE: 1

SCHEMA REPORTS
Menu Listing

MENU		HEADING / PROMPT	MENU/PROG	M/P
dbmenu		Data Base Maintenance Menu		
	1	Field Security Maintenence	fldsec	P
	2	Process Field Passwords	procpass	P
	3	Index Maintenance	idxmnt	P
	4	Volume Maintenance	volmnt	P
	5	Hash Table Maintenance	rekey	P
	6	Explicit Relationship Maintenance	repoint	P
	7	Data Base Statistics	dbstats	P
	8	Hash Table Statistics	hts	P
dvermen		Dateiverwaltungs-Menue		
	1	Hersteller Verwaltung	sher100	E
	2	Modell Verwaltung	smod100	E
	3	Artikel Verwaltung	sart100	E
entmenu		Main Menu		
	1	Dateiverwaltungs-Menue	dvermen	M
	2	Report-Menue	rptmenu	M
	3	SQL - Query/DML Language	sql	P
	4	System Menu	sysmenu	M
rptmen		MENUH Report Menu		
	1	Executable Listing	lexec	P
	2	Menu Listing	lmenu	P
	3	Group Listing	lgrp	P
	4	Employee Listing	lemp	P
	5	Print Help Documentation	prthdoc	P
rptmenu		Report-Menue		
	1	Lagerbestandsueberblick	inventur	P
	2	Artikelumschlags-Report	umschlag	P
	3	Lagereingangsdaten	erwlist	S
scrmen		MENUH Screen Menu		
	1	Executable Maintenance	execmnt	P
	2	Menu Maintenance	menumnt	P
	3	Group Maintenance	grpmnt	P
	4	Employee Maintenance	empmnt	P
	5	Enter Help Documentation	enthdoc	P
	6	Program Loading	lfilegen	P
	7	System Parameter Maintenance	parmnt	P
sfmenu		SFORM Menu		
	1	Paint Screen	paint	P
	2	Screen Entry	sfmaint	P
	3	Test Screen	sfsamp	P
	4	Process Screen	sfproc	P
	5	Screen Reports	sfrep	P
	6	Restore Screen	sfrestr	P

```
DATE : 07/24/86  TIME : 17:11:35                    PAGE:   2
                    SCHEMA REPORTS
                     Menu Listing

MENU      HEADING / PROMPT                          MENU/PROG M/P

       7  List Screens                              sfslist    P
       8  Create Default Screen Form                cdsf       P

sysmenu   System Menu
       1  Schema Maintenance                        schent     P
       2  Schema Listing                            schlst     P
       3  Create Data Base                          crdb       P
       4  SFORM Menu                                sfmenu     M
       5  ENTER Screen Registration                 entmnt     P
       6  SQL - Query/DML Language                  sql        P
       7  SQL Screen Registration                   ssqlmnt    P
       8  Listing Processor                         lst        P
       9  Data Base Test Driver                     sys920     P
      10  MENUH Screen Menu                         scrmen     M
      11  MENUH Report Menu                         rptmen     M
      12  Reconfigure Data Base                     scom       P
      13  Write Data Base Backup                    budb       P
      14  Read Data Base Backup                     redb       P
      15  Data Base Maintenance Menu                dbmenu     M

TOTAL MENUS:   8
```

2.2.3 Listen der Gruppen

Mit diesem Programm wird eine Liste aller im Datenwoerterbuch enthaltenen Gruppen von Beschaeftigten erzeugt. Der Report wird nach Gruppen-Identifikatoren geordnet in alphabetischer Reihenfolge erzeugt. Hinter der Gruppe steht die Liste der Programme, auf die die Gruppe Zugriff hat, und die jeweiligen Zugriffsrechte fuer alle Programme. Es gibt keine Optionen. Der Report ist 79 Spalten breit und wurde hier aus drucktechnischen Gruenden in der Breite reduziert. Aufruf: 3 oder lgrp.

```
DATE : 07/24/86  TIME : 17:19:15                    PAGE:   1
                    SCHEMA REPORTS
                  Group Access Listing

                                          ACCESS PRIVILEGES
ID   NAME              ENTRY PT  MENU/PROG  INQ ADD MOD DEL

DVS  DV Spzialisten    dvermen   sher100  E  Y   Y   Y
                                 smod100  E  Y   Y   Y
                                 sart100  E  Y   Y   Y
```

2.2.4 Listen der Beschaeftigten

Mit diesem Programm wird eine Liste aller im Datenwoerterbuch eingetragenen Beschaeftigten erzeugt. Der Report ist in alphabetischer Reihenfolge nach Login-Identifikatoren der Beschaeftigten sortiert. Hinter den Beschaeftigten sind die Programme aufgefuehrt, fuer die der Beschaeftigte Zugriffsrechte hat, die von denen der Gruppe abweichen. Der Report wird erstellt, indem unter dem 'Report Menu' eine 4 (oder lemp) eingegeben wird. Es gibt keine Optionen. Der Report ist 132 Zeichen breit und wurde hier aus drucktechnischen Gruenden in der Breite reduziert.

```
DATE : 07/24/86  TIME : 17:19:25                          PAGE:
                        SCHEMA REPORTS
                   Employee Access Listing

                                    SPECIAL ACCESS PRIVILEGES
ID   PASSWORD NAME      GROUP ENTRY PT MENU/PROG ACC I A M D

haha zug      Harry     DVS   entmenu  sher100 E  Y  Y Y Y Y
               Hirsch                  smod100 E  Y  Y Y Y Y
                                       sart100 E  Y  Y Y Y Y
                                       sql     P  Y
                                       inventurP  Y
                                       umschlagP  Y
                                       erwlist S  Y
                                       dvermen M  Y
                                       rptmenu M  Y

ort  statd    Jens-Uwe DVS   dvermen
```

2.2.5 Listen einer Hilfsdokumentation

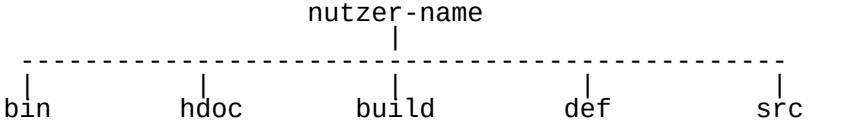
Mit diesem Programm wird die zu einem Menue oder Programm gehoerige Hilfsdokumentation ausgegeben. Aufruf durch Eingabe von 5 oder prthdoc). Der Nutzer wird gebeten, ein Menue oder Programm auszuwaehlen und dann wird die dazugehoerige Hilfsdokumentation ausgegeben. Ein Hilfsdokumentations-Ausdruck koennte beispielsweise folgendermassen aussehen:

```
                    Lagerbestandsuebersicht

  Dieser Report listet alle Artikel im Lager, gruppiert
  nach Bezeichnung und Hersteller. Die Artikel jeder
  Gruppe werden gezaehlt und der durchschnittliche
  Grosshandelspreis fuer sie berechnet.
```

3. VERWALTUNG EINER WEGA-DATA-DATENBANK

In diesem Abschnitt soll beschrieben werden, wie eine WEGA-DATA-Datenbank erstellt und verwaltet wird. Die wichtigsten Operationen, die hier erlaeutert werden, sind: Verwaltung der Datenbank-Struktur (Schema), Verwaltung der Datenbank selbst (ihre Erstellung, Rekonfiguration, sekundaere Indextabellen und physische Anordnung), Schaffung einer sicheren Umgebung, Gewinnung von Statistiken ueber die Nutzung der Datenbank und der Dienstprogramme.

Das Schema sollte in einer Verzeichnis-Struktur geschaffen werden, die fuer die Programmentwicklung geeignet ist, die beginnt, sobald der Grundstein fuer die Datenbank gelegt ist. Das folgende Bild illustriert, wie eine Verzeichnis-Struktur fuer die Entwicklung eines Anwendungssystems aussehen koennte. Die Anordnung muss nicht genauso aussehen, es handelt sich hierbei lediglich um ein uebliches und nuetzliches Vorgehen.



Das Verzeichnis bin enthaelt das Datenwoerterbuch, die ausfuehrbaren Dateien, die Datenbasis selbst und andere Hilfsdateien, die waehrend der Laufzeit benoetigt werden. Das Verzeichnis hdoc enthaelt Hilfsdokumentationen fuer die in bin befindlichen Programme. Das Verzeichnis build enthaelt die Kommandodateien, die zur Erstellung der ausfuehrbaren Dateien erforderlich sind. Das Verzeichnis def enthaelt die Dateien, die im Programm-Quellkode mit Hilfe von include-Anweisungen verwendet werden. Diese Dateien enthalten Datendefinitionen und Parameter, die innerhalb des gesamten Systems gueltig sind. Das Verzeichnis src enthaelt den Quellkode fuer Programme. Bei grossen Anwendungssystemen sollte das src-Verzeichnis selbst Unterverzeichnisse enthalten.

3.1 Verwaltung der Struktur der Datenbank

In diesem Abschnitt wird die Verwendung des interaktiven Schema-Eingabeprogramms beschrieben. Dieses Programm kann verwendet werden, um ein vollstaendig neues Schema einzugeben oder um das Schema einer existierenden Datenbank zu modifizieren. Das Schema wird im WEGA-DATA-Datenwoerterbuch verwaltet und zwar in derselben WEGA-DATA-Datenbank-Datei, die alle Systemmenues, Bildmasken, Programme, Beschaeftigten und Zugriffsrechte enthaelt. Damit widerspiegeln sich die in diesem Programm vorgenommenen Veraenderungen so lange nicht in der Anwenderdatenbasis, bis das Rekonfigurationsprogramm abgearbeitet wurde.

Vom Schema-Eingabeprogramm werden zwei Bildmasken verwendet. Mit der ersten Bildmaske koennen allgemeine Informationen ueber den Aufbau eines Datensatztyps der Datenbank festgelegt werden. Die zweite Bildmaske dient der Definition der Felder in den jeweiligen Datensatztypen. Das Programm geht von einer zur anderen Bildmaske ueber.

Bei der Vornahme von Eintragungen erzwingt das Programm 'Schema Maintenance' die Regeln des Schemaaufbaus interaktiv. Fehler werden entweder sofort gemeldet oder gar nicht erst zugelassen, indem der Cursor Charakteristika, die nicht zulaessig sind, ueberspringt. Wenn ein bestimmtes Merkmal eingetragen werden soll und das ist nicht moeglich, muss eins der anderen Merkmale des Feldes geaendert werden, um eine gueltige Eintragung vorzunehmen.

Obwohl das Rekonfigurationsprogramm letztendlich fast alle Arten von Aenderungen am Schema zulassen soll, gelten zur Zeit bestimmte Einschraenkungen.

1. Schluessel muessen immer das erste Feld im Datensatztyp sein. Das Programm erstellt automatisch das Schluesselfeld an der Spitze der Felderliste.

2. Ein Schluesselfeld kann nicht modifiziert (d.h. Laenge, Typ oder eine Veraenderung, durch die das Schluesselfeld kein Schluessel mehr ist) oder geloescht werden, wenn andere Felder in anderen Datensaetzen explizit Bezug auf dieses Feld nehmen. Wird ein Schluesselfeld, auf das nicht Bezug genommen wird, modifiziert muss nach Rekonfiguration der Datenbank 'Hash Table Maintenance' (Abschnitt 3.5.3) ablaufen, sonst ein Zugriff auf den dadurch betroffenen Datensatztyp nicht ueber dessen Primaerschluessel moeglich ist.

3. Felder koennen von einem Datensatztyp in einen anderen Datensatztyp gebracht werden, aber die Daten werden bei der Umsetzung nicht kopiert.

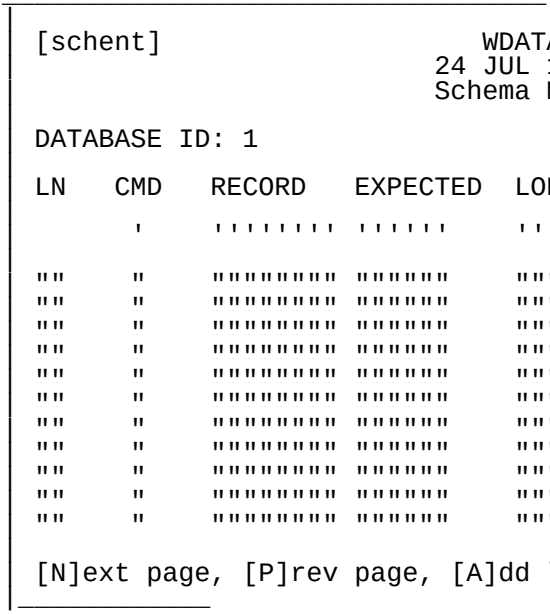
4. Obgleich ein Feld verlaengert oder gekuerzt werden kann, kann man seinen Typ nicht aendern. So kann z.B. AMOUNT 4 zu AMOUNT 10 werden, aber es kann nicht ein Feld vom Typ STRING 8 werden.

Soll das Programm zur Eingabe von Datensatztypen verwendet werden, wird aus dem Systemmenue heraus 'Schema Maintenance' gewaehlt. In den folgenden Abschnitten wird beschrieben, wie die Bildmasken verwendet werden koennen und wie die fertige Databasis-Struktur aufgelistet werden kann.

3.1.1 Verwaltung der Datensatztypen

Mit dieser Bildschirmmaske koennen Datensatztypen in der Struktur der Datenbank hinzugefuegt, geaendert und ge-

loescht werden.



'''''''' Eingabebereich fuer Datensatztyp
"""""""" Darstellungsbereich fuer Datensatztypen

Der Aufbau der Bildmaske ist typisch fuer viele im WEGA-DATA-System verwendete Bildmasken. Oben befindet sich der Raum fuer Dateneingaben, in dem neue Eintragungen vorgenommen werden koennen. Darunter steht ein Bereich zur Verfuegung, in dem existierende Eintragungen modifiziert werden koennen. Existierende Eintragungen werden seitenweise mit je 11 Eintraegen pro Seite auf der Bildmaske angezeigt.

Die Spalten mit der Ueberschrift

LN CMD RECORD EXPECTED LONG NAME DESCRIPTION

sind dafuer vorgesehen, dass unmittelbar darunter die Dateneingabe erfolgt. Nun folgt die Erklaerung des auf dem Schirm erscheinenden Prompters:

[N]ext page, [P]rev page, [A]dd line, oder number:

Es handelt sich um den Prompter fuer die Seitenbank. Die Seitenbank enthaelt alle Seiten mit Definitionen von Datensatztypen. Dieser Prompter gestattet Ihnen die Auswahl des Operationsmodus' fuer die Seitenbank. Die Auswahlmoeglichkeiten haben folgende Bedeutungen:

n - Die naechste Seite der Datensatztypdefinitionen er-

scheint auf dem Bildschirm.

p - Die vorhergehende Seite der Datensatztypdefinitionen erscheint auf dem Bildschirm.

a - Durch Positionieren des Cursors auf den Bereich zur Eingabe von Datensatztypen wird Ihnen das Hinzufuegen von neuen Datensatztypdefinitionen ermoeglicht.

1-256 Auf dem Bildschirm erscheint die Seite, die die angegebene Datensatztypdefinition enthaelt, der Cursor wird auf diesen Datensatztyp eingestellt, so dass Ihnen Aendern und Loeschen ermoeglicht wird. Sie koennen dazu die Erklaerungen verwenden, die in der Beschreibung der Spalte CMD angegeben sind.

LN

Es handelt sich um eine von WEGA-DATA vergebene Zeilennummer. Durch diese kann man auf die Zeile zugreifen, falls man diese aendern oder umstellen moechte.

CMD

Dieser Raum wird fuer die Befehlseingabe zur Ausfuehrung von Operationen auf der aktuellen Zeile benutzt. In der mit der Ueberschrift CMD versehenen Spalte kann man, durch die Befehle CTRL/U und RETURN, den Cursor nach oben und unten bewegen. Durch die Cursorstellung wird die aktuelle Zeile festgelegt.
Es folgen die gueltigen Befehle:

f - Modifizieren der Felder des aktuellen Datensatztyps durch Aufruf der zweiten Seite der Bildmaske.

m - Modifizieren der Felder EXPECTED, LONG NAME oder DESCRIPTION des aktuellen Datensatztyps.

d - Loeschen des aktuellen Datensatztyps.

1-256 Umnumerierung des aktuellen Datensatztyps entsprechend der Nummer und Umordnung der auf dem Bildschirm angezeigten Datensatztypen.

q - Erneute Anzeige des Seitenprompters unten auf dem Bildschirm.

RECORD

Name des Datensatztyps, bestehend aus 8 Zeichen. Er darf nur einmal vergeben werden, darf nur aus (Klein- und Gross-) Buchstaben, Zahlen und dem Unterstreichungszeichen (_) bestehen und muss mit einem Buchstaben beginnen. Dieser Name wird in der gesamten Systemstruktur zur Bezugnahme auf den Datensatztyp verwendet.

EXPECTED

Die zu erwartende Anzahl der Datensaetze dieses Typs.

LONG NAME
Es handelt sich hierbei um einen ausfuehrlicheren Namen bestehend aus max. 16 Zeichen, der von ENTER und dem Datenbasis-Testtreiber zur Anzeige von Fehlermeldungen verwendet wird. Der Name darf nur aus (Gross- und Klein-) Buchstaben, Zahlen und der Unterstreichung (_) bestehen und muss mit einem Buchstaben beginnen.

DESCRIPTION
Es handelt sich hierbei um ein Kommentarfeld, in das eine Bezeichnung des Datensatztyps eingetragen werden kann.

Zum Hinzufuegen neuer Datensatztypen wird 'a' eingegeben. Damit wird der auf dem Schirm unten stehende Prompter beantwortet. Der Prompter wird geloescht und der Cursor bewegt sich zu dem direkt unter der Spalteneberschrift RECORD befindlichen Dateneingabebereich. Man kann den Cursor innerhalb des Dateneingabebereiches unter den Spaltenueberschriften durch die Taste RETURN vorwaerts (von links nach rechts) bewegen. Durch Druecken der Tastenkombination CTRL/U wird der Cursor zurueckbewegt (von rechts nach links). Steht der Cursor auf der ersten Ueberschrift RECORD und wird CTRL/U gedrueckt, verlassen Sie den Ergaenzungsmodus und der Seitenprompter erscheint wieder unten auf dem Bildschirm.

3.1.2 Feldverwaltung

In diesem Abschnitt soll die Verwendung der Bildmaske zur Verwaltung der Felder von Datensatztypen (der zweiten Bildmaske des Programms 'Schema Maintenance') erklaert werden. Mit dieser Bildmaske koennen Felder definiert, modifiziert oder geloescht werden.

```
[schent]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                          Schema Maintenance

RECORD: her

LN CMD FIELD  KEY REF   TYPE  LEN  LONG NAME  COMB. FIELD
    '  ''''''  '  ''''' ''''' '''  '''''''''  ''''''''

""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""
""  "  """""""  "  """""" """""" """  """""""""  """""""""

[N]ext page, [P]rev page, [A]dd line, or number: a_
```

```
'''''''' Eingabebereich fuer Felder
"""""""" Darstellungsbereich fuer Felder
```

Diese Seite der Bildmaske wird in aehnlicher Weise wie die erste Seite behandelt. Sie hat einen Dateneingabebereich und einen Darstellungsbereich. Die Bildmasken-Prompter haben folgende Bedeutung:

[N]ext page, [P]rev Page, [A]dd line oder number:

Prompter fuer die Seitenbank. Mit diesem Prompter koennen Sie den Operationsmodus fuer die Seitenbank auswaehlen. Die Auswahlmoeglichkeiten haben folgende Bedeutung:

n - Die naechste Seite mit Feldern fuer den aktuellen Datensatztyp erscheint auf dem Bildschirm.

p - Die vorhergehende Seite mit Feldern fuer den aktuellen Datensatztyp erscheint auf dem Bildschirm.

a - Durch Positionierung des Cursors auf den Bereich fuer Dateneingabe koennen neue Felder hinzugefuegt werden.

1-256 Auf dem Bildschirm erscheint die Seite mit dem angegebenen Feld. Der Cursor zeigt auf dieses Feld, so dass Ihnen Modifizierungen und Loeschen ermoeglicht werden. Naehere Erklaerungen sind in der Beschreibung der Spalte CMD zu finden.

RECORD:
Dieser Prompter dient nur der Anzeige. Es zeigt an, fuer welchen Datensatztyp Sie Felder hinzufuegen oder aendern.

LN
Es handelt sich hierbei um eine vom System vergebene Zeilennummer. Durch diese kann man auf die entsprechende Zeile zugreifen, falls man diese modifizieren oder umstellen moechte.

CMD
Dieser Bereich wird fuer die Befehlseingabe zur Ausfuehrung von Operationen auf der aktuellen Zeile verwendet. Innerhalb der mit der Ueberschrift CMD versehenen Spalte kann man durch Druecken der Tasten CTRL/U und RETURN den Cursor in der aktuellen Seite nach oben und unten bewegen. Durch die Cursorstellung ist die aktuelle Zeile festgelegt.
Es folgen die gueltigen Befehle:

m - Modifizieren der Spalten KEY, REF, TYPE, LEN, LONG NAME oder COMB. FIELD im aktuellen Feld.

d - Loeschen des aktuellen Feldes.

1-256 Umnumerierung des aktuellen Feldes gemaess Nummer und Umordnen der angezeigten Felder.

q - Erneute Anzeige des Prompters unten auf dem Bildschirm.

FIELD
Nur einmal auftretender, aus 8 Schriftzeichen bestehender Feldname. Darf nur aus Buchstaben (Gross- und Kleinbuchstaben), Zahlen und der Unterstreichung (_) bestehen und muss mit einem Buchstaben beginnen. Dieser Name wird in der gesamten Systemstruktur zur Bezugnahme auf das Feld verwendet.

KEY
Steht in der Spalte ein Sternchen, so ist das aktuelle Feld der Primaerschluessel des Datensatzes.

REF
Der Name des Primaerschluessels eines anderen Datensatztyps. Dieser wird zur Schaffung der bereits oben besprochenen logischen (expliziten) Beziehungen verwendet.

TYPE
Ist der Datentyp des Feldes. Nur das erste Schriftzeichen des Typs muss eingegeben werden (Gross- oder Kleinbuchstabe). Hier die gueltigen Datentypen:

n - NUMERIC
f - FLOAT

s - STRING
d - DATE
t - TIME
a - AMOUNT
c - COMB

LEN

Die festgelegte Laenge, des auf den Bildmasken und den Listen erscheinenden Feldes. Bei NUMERIC- und STRING-Feldern ist LEN genau die angezeigte Laenge. Bei FLOAT-Feldern hat LEN das Format nnd, wobei nn die Gesamtzahl der angezeigten Positionen, einschliesslich Dezimalpunkt und d die Anzahl der rechts vom Dezimalpunkt stehenden Ziffern ist. Hier ein Beispiel: Ein FLOAT-Feld mit der Laenge LEN 123 besitzt das folgende Format

nnnnnnnn.nnn

Ein FLOAT-Feld mit der Laenge LEN 100 hat demzufolge das Format

nnnnnnnnnn

Beachten Sie, dass hierbei kein Dezimalpunkt auftritt, so dass das Format in diesem Fall zehn Ziffern hat. Bei AMOUNT-Feldern gibt LEN die Anzahl der links vom Dezimalpunkt stehenden Ziffern an, wobei man von der Annahme ausgeht, dass rechts vom Dezimalpunkt zwei Ziffern stehen. Hier ein Beispiel: Ein AMOUNT-Feld mit der Lange LEN 6 sieht auf Bildschirmen und Listen folgendermassen aus:

nnnnnn.nn

Die gesamte Anzeigelaenge von AMOUNT-Feldern ist LEN+3. Die Maximallaengen der verschiedenen Feldtypen sind folgende:

NUMERIC - 9
FLOAT - 179
STRING - 256
DATE - standardmaessig 8, keine Eingabe
TIME - standardmaessig 5, keine Eingabe
AMOUNT - 11
COMB - wird vom System berechnet, keine Eingabe

LONG NAME

Ein ausfuehrlicherer Name fuer das Feld, bestehend aus 16 Zeichen. Er darf nur aus (Gross- und Klein-) Buchstaben, Zahlen und der Unterstreichung (_) bestehen und muss mit einem Buchstaben beginnen. Dieser Name wird von SQL bei der Ausfuehrung von Anfragen verwendet. Es ist nicht erforderlich, dass dieser Name in der gesamten Datenbasis nur einmal auftritt, wie das beim Feldnamen der Fall sein muss. Er darf aber innerhalb dieses Datensatztyps nur einmal auftreten.

COMB. FIELD
Der Name des kombinierten Feldes (d.h. eines Felds vom Typ COMB) im aktuellen Datensatztyp, zu dem das aktuelle Feld gehoeren wird.

Zum Hinzufuegen neuer Felder in den aktuellen Datensatztyp wird zur Auswahl des Ergaenzungsmodus' die Taste A gedrueckt. Damit wird der auf dem Schirm unten stehende Prompter beantwortet. Der Prompter wird geloescht und der Cursor bewegt sich zu dem direkt unter der Spaltenueberschrift FIELD befindlichen Dateneingabebereich. Aehnlich wie bei der vorangegangenen Bildmaske, kann man den Cursor durch Druecken der Taste RETURN vorwaerts und vermittels der Tastenkombination CTRL/U zurueckbewegen. Steht der Cursor auf der ersten Ueberschrift FIELD und wird CTRL/U gedrueckt, verlassen Sie den Ergaenzungsmodus und der Prompter erscheint unten auf dem Bildschirm.

3.1.3 Listen des Schemas

Durch Aufruf von 2 ('Schema Listing') im 'System Menu' kann das gesamte aktuelle Schema ausgedruckt werden. Alle Datensatztypen, Felder und Beziehungen werden gedruckt und es werden alphabetische Querverweise von Feldern und Groessenangaben ausgedruckt. Dieses Programm hat keine Optionen. Der Report ist 79 Zeichen breit und wurde hier aus drucktechnischen Gruenden in der Breite reduziert.

DATE : 07/24/86 TIME: 11:29:19 PAGE: 1

SCHEMA REPORTS
Schema Listing

RECORD/FIELD	REF	TYPE	LEN	LONG NAME
her	10			hersteller
*henr		NUMERIC	4	nummer
hename		STRING	35	name
hestr		STRING	30	strasse
hort		STRING	20	ort
heplz		NUMERIC	4	postleitzahl
hetelex		NUMERIC	7	telex
modell	50			modell
*mosch		COMB		mod_schluessel
monr		NUMERIC	7	mod_nummer
mohenr	henr	NUMERIC	4	hersteller_num
mobez		STRING	30	bezeichnung
art	100			artikel
*sernr		NUMERIC	9	seriennummer
artmod	mosch	COMB		obermodell
arterda		DATE		erwerbsdatum
artgros		AMOUNT	5	grosshand_preis
artbest	benr	NUMERIC	9	bestellnummer
arinprei		AMOUNT	5	ind_abgabepreis
kunde	10			kunde
*kunr		NUMERIC	5	kundennummer
kuname		STRING	30	name
kustr		STRING	30	strasse
kort		STRING	20	ort
kuplz		NUMERIC	4	postleitzahl
kutelex		NUMERIC	7	telex
kuruf		NUMERIC	7	ruf
best	100			bestellung
*benr		NUMERIC	9	best_nummer
bedat		DATE		best_datum
bekun	kunr	NUMERIC	5	kundennummer

DATE : 07/24/86 TIME: 11:29:53 PAGE: 2

SCHEMA REPORTS
Record List

RECORD	NUMBER	EXPECTED	LENGTH
art	3	100	15
best	5	100	8
her	1	10	50
kunde	4	10	50
modell	2	50	21

DATE : 07/24/86 TIME: 11:29:59 PAGE: 3

SCHEMA REPORTS
Field List

FIELD	NUMBER	RECORD	TYPE	LENGTH
arinprei	18	art	AMOUNT	30
artbest	17	art	NUMERIC	9
arterda	12	art	DATE	
artgros	13	art	AMOUNT	5
artmod	9	art	COMB	6
artmod_mohenr	11	art	NUMERIC	4
artmod_monr	10	art	NUMERIC	7
bedat	27	best	DATE	
bekun	28	best	NUMERIC	5
benr	26	best	NUMERIC	9
hename	2	her	STRING	35
henr	1	her	NUMERIC	4
heplz	15	her	NUMERIC	4
hestr	3	her	STRING	30
hetelex	16	her	NUMERIC	7
hort	14	her	STRING	20
kort	22	kunde	STRING	20
kuname	7	kunde	STRING	30
kunr	19	kunde	NUMERIC	5
kuplz	23	kunde	NUMERIC	4
kuruf	25	kunde	NUMERIC	7
kustr	21	kunde	STRING	30
kutelex	24	kunde	NUMERIC	7
mobez	7	modell	STRING	30
mohenr	6	modell	NUMERIC	4
monr	5	modell	NUMERIC	7
mosch	4	modell	COMB	6
sernr	8	art	NUMERIC	9

DATE : 07/24/86 TIME: 11:30:07 PAGE: 4

SCHEMA REPORTS
Record Relationships

RECORD	RECORD	FIELD
art	best	artbest
art	modell	artmod
art	her	artmod_mohenr
best	kunde	bekun
modell	her	mohenr

DATE : 07/24/86 TIME: 11:30:07 PAGE: 5

SCHEMA REPORTS
Record Change List

DATE : 07/24/86 TIME: 11:30:07 PAGE: 6

SCHEMA REPORTS
Field Change List

3.2 Verwaltung der Datenbank

In diesem Abschnitt sollen die Werkzeuge beschrieben werden, die benutzt werden, um die physische Struktur der Datenbasis zu erstellen und zu verwalten. Es sollen folgende Programme besprochen werden: Erstellen einer Datenbank, Rekonfiguration einer Datenbank, Verwaltung des Sekundaerindex und Verwaltung des Datentraegers. Diese Dienstprogramme werden im normalen Verlauf der Arbeit mit einer WEGA-DATA-Datenbank verwendet.

3.2.1 Erstellen einer Datenbank

Dieses Dienstprogramm wird zum Erstellen einer Ausgangs-Datenbank entsprechend der Schema-Beschreibung des Datenwoerterbuchs benutzt. Der Prozess besteht aus: Zusammenstellen des Schema, Bestimmung des Typs der Datei, in der sich die Datenbank befinden soll und Schreiben der Ausgangsdatei. All diese Schritte werden vom Programm selbsttaetig ausgefuehrt. Da 'Create Data Base' eine leere Datenbank-Datei anlegt, sind bestimmte Vorsichtsmassnahmen einzuhalten:

1. Dieses WEGA-DATA-Anwendungssystem ist von keinem weiteren Anwender zu benutzen, wenn 'Create Data Base' arbeitet. Das ist vor Aufruf des Programms zu sichern.

2. Die "alte" Datenbasis darf keine speicherungswuerdigen Daten enthalten, da sie nach Beendigung von 'Create Data Base' leer sein wird. Sind wertvolle Daten vorhanden, sind diese unter Verwendung von SQL (Abschnitt 6.2.4) in eine ASCII-Datei auszulagern und dann nach Ablauf des Programms wieder zurueckzuladen (Abschnitt 6.2.5).

3. "Raw"-Datenbasen koennen nicht in einem gemounteten Dateisystem angelegt werden. Daher muss das Dateisystem zuerst dismountet werden.

Im allgemeinen koennen wahrend der Zusammenstellung des Schemas Fehler gemacht werden. Dazu gehoert auch die Ueberschreitung des fuer Datensatztyp- und Feldnamen zur Verfuegung stehenden Speicherplatzes, Stack-Ueberlauf waehrend der Felderzeugung, Ausfall der Hardware u.a. Das Datenwoerterbuch wird in eine Backup-Datei kopiert, bevor der Prozess beginnt, so dass bei Auftreten eines zum Absturz fuehrenden Fehlers das Datenwoerterbuch aus der Backup-Datei wieder abgerufen werden kann.

Eine Datenbasis-Datei kann entweder in einer normalen WEGA-Datei oder in einer speziellen WEGA-Datei abgelegt werden. Man kann dies durch Setzen der entsprechenden Parameter unter Verwendung von 'Volume Maintenance' (Abschnitt 3.2.4) festlegen. Standard ist eine normale WEGA-Datei im aktuellen Verzeichnis. Der Name der Datei wird durch das Programm festgelegt und ist file.db. Wird eine normale WEGA-Datei

verwendet, besteht ein "link" zur Datei file.dbr. Andernfalls zeigt file.dbr auf das Raw-Device, auf dem sich die Datei befindet, d.h. file.dbr hat nur dann eine reale Bedeutung, wenn eine spezielle Datei verwendet wird.

Es ist darauf zu achten, dass file.db und file.dbr immer auf dieselbe WEGA-Datei zeigen. Normalerweise ist das kein Problem. Einige WEGA Dienstprogramme, wie z.B. cp (copy), durch das von einem Verzeichnis in ein anderes kopiert wird, legen jedoch zwei verschiedene Dateien von miteinander verknuepften Dateien an. Um zu ueberpruefen, ob das vielleicht passiert ist, wird ls -i file.db file.dbr durchgefuehrt und es wird ueberprueft, ob beide Dateien dieselben Inode-Nummern haben. Ist das nicht der Fall, wird file.dbr geloescht und beide Dateien werden unter Verwendung des Kommandos ln file.db file.dbr wieder miteinander verbunden.

Ueberschreitet die zu erwartende Groesse der Datendatei ein Megabyte, kann man erheblich Zeit sparen und den Systemdurchsatz wesentlich erhoehen, wenn man die Datenbank als spezielle WEGA-Datei (special file) anlegt. Dadurch koennen Programme den E/A-Verkehr mit einem Raw-Device vornehmen und damit die Indizierung und Pufferung der WEGA-Datei umgehen. Anleitung siehe 'Volume Maintenance' (Abschnitt 3.2.4).

Zur Erstellung einer leeren Datenbank-Datei wird aus dem WEGA-DATA-Systemmenue heraus '3' aufgerufen. Wenn bereits eine Datenbank existiert, so wird sie so verkuerzt, dass sie keine Datensaetze mehr enthaelt.

```
[crdb]                          WDATA SYSTEM
                              24 JUL 1986 - 15:25
                               Create Data Base


This program creates a new, empty data base. If you
have information in an existing data base, it will
be lost. No one else should be using the data base
while this program is running.



PROCEED? y_
```

Zuerst erscheint zur Bestaetigung auf dem Schirm der Prompter PROCEED?. 'Y' startet den Prozess; durch Druecken von CTRL/U oder n wird diese Funktion abgebrochen. Die Ausschrift weist uns darauf hin, dass eine bereits vorhandene Datenbasis verloren gehen wuerde und dass niemand sonst die Datenbasis benutzen soll, waehrend dieses Programm (crdb) laeuft.

Im weiteren Verlauf wird das Programm die unten abgebildeten Prompter auf dem Schirm anzeigen:

```
[crdb]                          WDATA SYSTEM
                              24 JUL 1986 - 15:25
                               Create Data Base

** Phase I   **

** Phase II  **

** Phase III **


Process complete. Back up the database.  ->-> _
```

Als Antwort auf den letzten Prompter beenden Sie mit RETURN und das Systemmenue wird wieder erscheinen.

3.2.2 Rekonfiguration der Datenbank

Eine WEGA-DATA-Datenbasis kann rekonfiguriert werden, um es zu ermoeglichen, dass Datensatztypen, Felder innerhalb von Datensatztypen und Beziehungen zwischen Datensatztypen hinzugefuegt oder geloescht werden koennen. Felder koennen auch nach Bedarf verlaengert oder verkuerzt werden und entsprechend evtl. neu auftretenden Beduerfnissen kann die Anzahl von Datensaetzen aller Typen vergroessert oder verkleinert werden. Die notwendigen Beschraenkungen sind in 3.1, 'Schema Maintenance', aufgefuehrt. Das Datenwoerterbuch enthaelt die fuer den Umstrukturierungsprozess erforderlichen Informationen.

'Reconfigure Data Base' formatiert die Datenbasis-Datei um und aktualisiert das Datenwoerterbuch. Waehrend der Umformatierung duerfen keine weiteren Anwender dieses WEGA-DATA-Anwendungssystem benutzten. D.h., es darf auch nicht angefragt oder aktualisiert werden und es duerfen auch keine neuen Datensaetze eingegeben oder das Datenwoerterbuch geaendert werden. Vor dem Starten dieses Programms ist zu sichern, dass niemand diese Anwendung benutzt.

Das Rekonfigurationsprogramm erfuellt seine Funktion schrittweise. Zuerst werden alle Aktualisierungen an der Datenbasis gesperrt und dann wird eine Kopie des Datenwoerterbuchs angelegt, so dass dieses bei Auftreten eines Fehlers, durch den alle Daten verloren gehen (z.B. Lese/Schreib-Fehler auf Platte), wieder im Originalzustand abgerufen werden kann. Dann wird das Schema kompiliert und es werden Tabellen erzeugt, um die Datensaetze vom alten ins neue Format umzusetzen. Die Datenbasis wird in eine temporaere Datei geschrieben und dann in ihr neues Format

zurueckgelesen.

Wurde die erwartete Anzahl der Datensaetze verringert oder vergroessert, hat man Gelegenheit, den Hash-Table-Index neu aufzubauen. Es ist jedoch zu beachten, dass bei Aenderung des internen Datentyps (d.h. verlaengern, kuerzen, Typaenderung) von Schluesselfeldern durch 'Schema Maintenance', nach Beendigung der Rekonfiguration das Programm 'Hash Table Maintenance' laufen muss (siehe 3.5.3). Wird das unterlassen, ist kein Zugriff auf den betreffenden Datensatztyp ueber dessen Primaerschluessel moeglich.

Wird ein existierendes normales Feld so geaendert, dass eine explizite Beziehung aufgebaut wird, erstellt 'Reconfigure Data Base' nicht die erforderlichen Pointer-Listen. Man muss das folgende SQL-Skript auf die Datenbank anwenden, um die Pointer zu setzen.

```
update datsatyp
set feld = feld /
```

wobei feld der Name des geaenderten Feldes und datsatyp der Name des dieses Feld enthaltenden Datensatztyps ist.

Vor Rekonfigurieren einer existierenden Datendatei ist zu sichern, dass ein aktuelles Backup der Datenbank vorhanden ist. Anleitung dazu siehe Abschnitt 3.5.1, 'Write Data Base Backup'. Dann wird im System-Menue 'Reconfigure Data Base' (scom) gewaehlt.

```
[scom]                          WDATA SYSTEM
                              24 JUL 1986 - 15:25
                            Reconfigure Data Base

This program reformats the data base to reflect any
changes in structure you have made using Schema
Maintenance. You should make a backup copy of the
data base on diskettes or tape with Write Data Base
Backup (budb) before using this program. In addition,
no one else should be using the data base while this
program is running.


PROCEED? Y_
```

Der Text bedeutet: Dieses Programm formatiert die Datenbasis neu, um den Veraenderungen, die unter Verwendung von Schema Maintenance gemacht wurden, gerecht zu werden. Es ist eine Backup-Kopie der Datenbasis auf Disketten oder Band vermittels Write Data Base Backup (budb) anzulegen, bevor dieses Programm verwendet wird. Waehrend des Programmablaufs darf die Datenbasis nicht anderweitig verwendet werden.

Zum Starten des Prozesses wird der Prompter mit Y beantwortet. Die Eingabe von N oder CTRL/U fuehrt dazu, dass der Prozess gestoppt wird und ins Menue zurueckgegangen wird. Anschliessend werden folgende Meldungen angezeigt:

```
[scom]                          WDATA SYSTEM
                          24 JUL 1986 - 15:25
                        Reconfigure Data Base

** Phase I   **

** Phase II  **


Nominal hash table loading factor:  50%
New load factor without rebuilding: 52%


REBUILD HASH TABLE? _
```

Das Schema ist bereits kompiliert worden und die eigentliche Rekonfiguration kann beginnen. Hat man die Gesamtanzahl der Datensaetze in der Datenbasis vergroessert, hat man nun die Gelegenheit, die Groesse der Hash-Tabelle zu vergroessern. Es kann also 'Y' oder 'N' eingegeben werden. Auch auf den dann erscheinenden Prompter "REBUILD HASH INDEX?", muss mit 'y' oder 'n' geantwortet werden.

Als naechstes wird der folgende Prompter angezeigt:

```
                                WDATA SYSTEM
                          24 JUL 1986 - 15:25
                        Reconfigure Data Base

** Phase I   **

** Phase II  **

** Phase III **


Nominal hash table loading factor:  50%
New load factor without rebuilding: 52%


Use diskette/tape as the temporary file? _
```

Mit Beantwortung dieser Frage kann man entscheiden, ob die Rekonfiguration Diskette oder Band als temporaeres Speichermedium zur Hilfe nehmen soll. Wenn die Datenbasis-Datei zu umfangreich ist, als dass sie auf der Platte des Nutzers kopiert werden koennte, muss mit Y geantwortet werden.

Nach Abschluss des Prozesses werden Sie erneut gebeten, wie bereits vorher, ein Backup anzulegen, indem 'Write Data Base Backup' (budb) eingegeben wird. Es ist darauf zu achten, dass nicht dieselbe(n) Diskette(n) bzw. dasselbe Band wie zu Beginn dieses Abschnitts verwendet wird. Die Datenbasis-Datei ist rekonfiguriert worden und das Datenwoerterbuch ist aktualisiert worden, so dass der neue Aufbau des Schemas bereits widergespiegelt wird.

Existierende Programme brauchen nicht erneut kompiliert oder geladen zu werden, da die Struktur des Schemas dynamisch gebunden ist. Jedoch muessen Bildmasken (.q-Dateien) erneut verarbeitet (Process Screen) werden, wenn sie diese veraenderten Felder enthalten.

3.2.3 Index-Verwaltung

Vor der Verwendung des Programms ist zu sichern, dass keine weiteren Anwender das/die Feld(er) aktualisieren, fuer die Indexe hinzugefuegt oder geloescht werden sollen. Man geht ins 'Data Base Maintenance Menu' (15) und waehlt 'Index Maintenance' (3), wenn ein Sekundaer-(B-Baum)-Index fuer ein Feld in die Datenbank eingefuegt werden soll. Dann erscheint folgendes Bild auf dem Schirm:

```
[idxmnt]                    WDATA SYSTEM
                        24 JUL 1986 - 15:25
                          Index Maintenance

[A]DD OR [D]ROP INDEX: _
FIELD NAME           :

[A]SCENDING OR [D]ESCENDING:
```

Die Prompter auf dem Bildschirm haben folgende Bedeutung:

[A]DD OR [D]ROP INDEX:

Wenn diese Bildmaske erscheint, befindet sich der Cursor hinter diesem Prompter. Die zwei gueltigen Auswahlmoeglichkeiten und ihre Bedeutung sind:

a - ein B-Baum-Index soll fuer ein Datenbasis-Feld hinzugefuegt werden.

d - der B-Baum-Index eines Datenbasis-Feldes soll entfernt werden.

FIELD NAME:
Der Name des existierenden Datenbasis-Feldes, fuer das ein B-Baum-Index hinzugefuegt oder entfernt werden soll, wird eingegeben. Das Feld darf nicht den Typ COMB

haben. Nach Eingabe des Feldnamens erscheint zur Bestaetigung eine Meldung.

[A]SCENDING OR [D]ESCENDING:
Diese Frage wird nur gestellt, wenn ein Index hinzugefuegt werden soll.

a - Gibt an, dass die Eintragungen im Index in fallender Ordnung vorgenommen werden.

d - Gibt an, dass die Eintragungen im Index in steigender Ordnung vorgenommen werden.

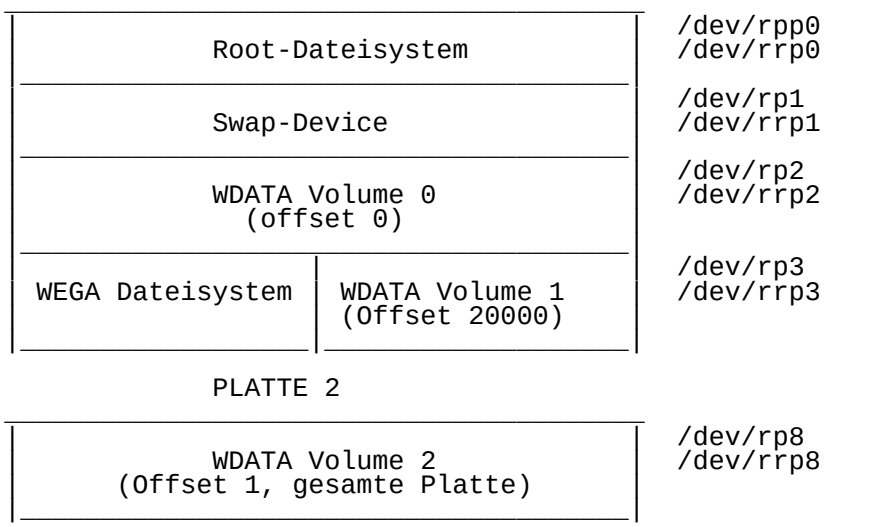
Befindet sich der Cursor hinter dem Prompter '[A]DD OR [D]ROP INDEX:', gelangt man mittels CTRL/U in den Menue-Handler. Andernfalls wird der Cursor mit CTRL/U zum vorhergehenden Prompter gebracht. Waehrend der Konstruktion eines B-Baums wird eine Zahl angezeigt, die angibt, wieviel Felder (in Zehnergruppen) aufgetreten und verarbeitet wurden.

3.2.4 Datentraeger-Verwaltung

WEGA-DATA ermoeglicht die Speicherung einer Datenbasis in einem oder mehreren zusammenhaengenden Erweiterungen des Plattenspeicherbereiches (raw disk). In der Terminologie von WEGA werden diese Plattenspeicherbereiche normalerweise als Dateisysteme bezeichnet. Normalerweise werden Dateisysteme unter Verwendung von mkfs initialisiert und in der WEGA Dateihierarchie als Verzeichnisse gemounted. Statt als WEGA-Dateisystem benutzt zu werden, kann ein zusammenhaengender Plattenspeicherbereich als als Teil einer WEGA-DATA-Datenbank verwendet werden. Daurch wird die Leistungsfaehigkeit der Systeme erhoeht, die WEGA-DATA verwenden. Werden keine Plattenspeicherbereiche angegeben, wird die Datanbank im aktuellen Verzeichnis als eine normale WEGA-Datei angelegt. Es ist zu beachten, das Raw-Datenbanken nicht in einem gemounteten Dateisystem angelegt werden koennen.

Das Programm 'Volume Maintenance' wird zur Spezifizierung der Plattenspeicherbereiche verwendet, die fuer die Datenbasis verwendet werden. Sie werden als Volumes bezeichnet. Das Programm 'Create Data Base' registriert die angegebenen Datentraeger und initialisiert sie zur Nutzung. Volume Null enthaelt immer die Datenbasis-Parameter und die Hash-Tabelle. Es muss immer ein Offset gleich Null haben, gerechnet vom Beginn des angegebenen Dateisystems. Andere Volumes koennen Offsets ungleich Null haben.

In der folgenden Uebersicht wird gezeigt, wie ein aus zwei Platten bestehendes WEGA-System unter Verwendung von WEGA-DATA konfiguriert werden kann.

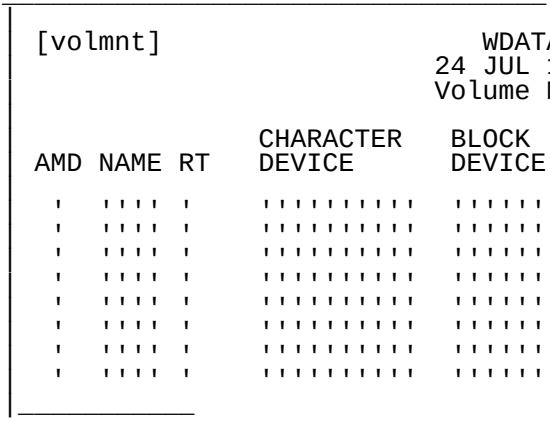


Das Geraet rp0 enthaelt das Root-Dateisystem. Geraet rp1 enthaelt den Swap-Bereich. Geraet rp2 enthaelt das Root-Volume der WEGA-DATA-Datenbasis. WEGA-DATA verwendet sowohl die blockweise Eingabe (/dev/rp2) als auch die zeichenweise Eingabe (/dev/rrp2). Der erste Teil von Geraet rp3 enthaelt ein WEGA-Dateisystem. Es wurde durch Aufruf von mkfs mit der entsprechenden Blockanzahl erstellt. Der Rest des Geraetes enthaelt das zweite Volume der WEGA-DATA-Datenbank. Die gesamte zweite Platte (/dev/rp8) enthaelt den Rest der WEGA-DATA-Datenbank. In einem gegebenen Computersystem kann es eine beliebige Anzahl von WEGA-DATA-Datenbanken geben. Einige davon koennen in Raw-Devices sein und der Rest in normalen WEGA-Dateien. Aufgabe des Nutzers ist es, unzulaessige Konflikte in der Zuordnung von Geraeten zu verhindern.

'Volume Maintenance' wird verwendet, um die Geraete zu spezifizieren, die fuer die WEGA-DATA-Datenbank verwendet werden sollen. Wenn keine Geraete angegeben sind, verwendet 'Create Data Base' eine normale Datei. Verbindungen (links) zu dieser Datei werden durch die Datei(-Namen) file.db und file.dbr hergestellt. Werden unter Verwendung von 'Volume Maintenance' spezielle Geraete angegeben, legt 'Create Data Base' (auch Reconfigure) file.db und file.dbr als spezielle Dateien an. file.db ist das Geraet fuer blockweise E/A fuer das Root-Geraet, waehrend file.dbr der zeichenweisen E/A fuer das Root-Geraet dient.

Zum Spezifizieren der Datentraeger fuer die Datenbank wird in 'Data Base Maintenance Menu' das Programm 'Volume Maintenance' (4) gewaehlt. Folgendes Bild erscheint:

.CP 22



Jede der auf dem Bildschirm angezeigten Zeilen steht fuer ein Volume der Datenbank. Leerzeilen werden ignoriert und werden nicht angezeigt, wenn die aktuelle Datentraeger spaeter erneut angezeigt werden. Genau eine Zeile muss in der Spalte RT ein x enthalten, wodurch angegeben wird, dass es sich dabei um das Root-Device handelt. Das Root-Device muss ein Offset von null Bloecken haben.

Soll ein neuer Datentraeger hinzugefuegt werden, wird der Cursor in eine leere Zeile gebracht, und in der Spalte AMD wird a eingegeben.

Soll ein existierender Datentraeger modifiziert werden, wird der Cursor in die Spalte AMD dieses Datentraegers gebracht, und es wird m eingegeben.

Soll ein existierender Datentraeger geloescht werden, wird der Curosr in die Spalte AMD dieses Volumens gebracht und es wird d eingegeben.

Befindet sich der Cursor in der Spalte AMD, wird er durch RETURN auf der Seite nach unten und durch CTRL/U nach oben bewegt. Befindet sich der Cursor in der letzten Zeile und wird RETURN gedrueckt, geht er in die erste Zeile. Befindet sich der Cursor in der ersten Zeile und wird CTRL/U gedrueckt, wird die Programmarbeit beendet. Wird eine Zeile hinzugefuegt oder modifiziert, wird der Cursor mittels RETURN nach rechts und mittels CTRL/U nach links bewegt. Befindet sich der Cursor in der letzten Spalte, geht er mittels RETURN in die erste zu modifizierende Spalte auf der Zeile. Befindet sich der Cursor in der ersten zu modifizierenden Spalte, bewirkt CTRL/U, dass er in die Spalte AMD auf der naechsten Zeile geht.

Hier die Erklaerung der auf der Bildmaske erscheinenden Prompter:

AMD
Befindet sich der Cursor in dieser Spalte, kann der Nutzer einen neuen Datentraeger hinzufuegen, indem er a eingibt (nur wenn die Zeile leer ist). Er kann den Datentraeger in dieser Zeile durch m modifizieren, oder durch Eingabe von d loeschen. Soll in die naechste Zeile gegangen werden, wird RETURN gedrueckt. Soll in die vorhergehende Zeile gegangen werden, wird CTRL/U gedrueckt. Befindet sich der Cursor in der ersten Zeile wird durch Druecken von CTRL/U das Programm beendet. Befindet sich der Cursor in der letzten Zeile, geht er durch Druecken von RETURN in die erste Zeile.

NAME
Ist ein aus vier Zeichen bestehender Name fuer den Datentraeger. Wird in der Hauptsache zur Anzeige und fuer Fehlermeldungen verwendet. Der Name darf nur einmal auftreten.

RT
Ein in dieser Spalte eingegebenes x gibt an, dass es sich um das Root-Device handelt. Das Root-Device enthaelt die Hash-Tabelle und die Datenbank.

CHARACTER DEVICE
Ist ein Name fuer zeichenweise E/A fuer den Datentraeger, z.B. /dev/rp2.

OFFSET
Ist der Offset (die Anzahl der 512-Byte-Bloecke), gerechnet vom Anfang des Geraets.

LENGTH
Ist die Laenge (die Anzahl der 512-Byte-Bloecke) des Datentraegers.

3.3 Datenschutz

In diesem Abschnitt soll beschrieben werden, wie man den Zugriff zu einer WEGA-DATA Datenbank kontrollieren kann. WEGA-DATA ermoeglicht verschiedene Sicherheitsebenen, die in Verbindung mit den Datei-Schutzvorkehrungen von WEGA verwendet werden koennen, wodurch eine sehr sichere Umgebung geschaffen wird.

Auf der hoechsten Ebene bietet WEGA-DATA Zugriffsbeschraenkungen fuer Programme und Menues, so dass nichtautorisierte Nutzer keine Programme oder Menues abarbeiten koennen, die bestimmten Zugriffsbeschraenkungen unterliegen. Dann verhindern die Zugriffsbeschraenkungen auf Feldebene, dass nichtautorisierte Nutzer auf einzelne Datenbank-Felder zugreifen koennen. Schliesslich existieren noch die Datei-Schutzvorkehrungen von WEGA, die verhindern, dass die Nutzer den Inhalt der Datenbank einfach auf den Bildschirm oder Drucker ausgeben. Im folgenden soll erklaert werden,

wie diese Eigenschaften genutzt werden koennen.

3.3.1 Datenschutz auf Feldebene

Der Datenschutz auf Feldebene wird durch die Verwendung der Bildmaske 'Field Security Maintenance' implementiert. Auf dieser Bildmaske koennen fuer jedes Datenbank-Feld einzelne Lese- und Schreibpassworte angegeben werden. Ist diese Spezifikation erst einmal eingegeben worden, kann das Feld nur gelesen oder beschrieben werden, wenn das entsprechende Passwort eingegeben wird.

Mit diesem Programm kann man also Lese- und Schreib-Passworte festlegen, die fuer ein einzelnes Datenbank-Feld bestimmt sind. Nachdem die Passworte mit diesem Programm festgelegt oder geaendert wurden, muessen sie verschluesselt werden ('Process Field Passwords', Abschnitt 3.3.2), so dass eine binaere Datei entsteht, die die Passwort-Informationen enthaelt. Danach ist das Passwort vom Nutzer einzugeben, wenn er Zugriff auf das Feld wuenscht. (siehe unlock, Abschnitt 6.3.3.3, 8.1.2.7 und 8.2.2.8 und accsfld, Abschnitt 10.3). Wird ein Passwort festgelegt, das nur das Lesen gestattet (read-only-Passwort) und wird kein Passwort zum Schreiben angegeben, erhaelt der Nutzer Lese- und Schreibrechte, wenn er dieses Passwort liefert. Sind beide Passworte spezifiziert, ermoeglicht das Schreib-Passwort das Lesen und Schreiben und das Lese-Passwort nur das Lesen.

Felder erhalten nicht nur Passworte fuer das Lesen und Schreiben, sie werden ausserdem einer Datenschutz-Gruppe zugeordnet. Liefert ein Nutzer das Passwort fuer ein Feld der Gruppe, wird die gesamte Gruppe je nach Passwort entweder zum Lesen oder Schreiben geoeffnet. Wird die Gruppe nicht spezifiziert, bildet das Feld fuer sich allein eine Gruppe.

Soll ein Nutzerprogramm Datensaetze in eine Datei einfuegen oder Datensaetze aus einer Datei loeschen, muss es Schreibrechte fuer alle in der Datei befindlichen Felder haben.

Aus 'Data Base Maintenance Menu' heraus ist 'Field Security Maintenance' (1, fldsec) zu waehlen, wenn man Datenschutz fuer einzelne Felder festgelegen will. Auf dem Bildschirm wird angezeigt:

```
[fldsec]                    WDATA SYSTEM
                        24 JUL 1986 - 15:25
                    Field Security Maintenance

FIELD NAME:

READ ONLY PASSWORD:
READ ONLY GROUP   :

WRITE PASSWORD    :
WRITE GROUP       :

[A]DD, [I]NQUIRE, [M]ODIFY, [D]ELETE _
```

Diese Bildmaske arbeitet wie eine ENTER-Bildmaske (siehe Abschnitt 5.2). Es folgt die Erklaerung der auf dem Schirm erscheinenden Prompter:

```
[A]DD, [I]NQUIRE, [M]ODIFY, [D]ELETE
```

Hinter diesem Prompter wird der Betriebsmodus des Programms festgelegt. Zum Hinzufuegen wird a, fuer Abfragen wird i, zum Modifizieren m und zum Loeschen d eingegeben. Dieser Prompter kann je nach Zugriffsrechten des aktuellen Nutzers verschieden sein.

FIELD NAME
: Name des Feldes, dessen Passwort-Attribute zu aendern sind.

READ ONLY PASSWORD
: Ist das Passwort, das dem Nutzer Lesezugriff auf das Feld gestattet. Wird kein Schreib-Passwort festgelegt, raeumt dieses Passwort dem Nutzer Lese- und Schreibrechte fuer das Feld ein.

READ ONLY GROUP
: Die hier eingegebene Zeichenkette bezeichnet eine Gruppe von Feldern. Alle Felder, fuer die an dieser Stelle die gleiche Zeichenkette eingegeben wurde, werden als zur selben Gruppe gehoerig betrachtet. Liefert ein Nutzer das Read-Only-Passwort fuer ein Feld in der Feldgruppe, ist es so, als wuerde er die Passwoerter fuer alle Felder in dieser Gruppe liefern, d.h., er hat mit einem Passwort Zugrif auf alle Felder dieser Gruppe.

WRITE PASSWORD
: Wird dieses Passwort angegeben, muss der Nutzer dieses Passwort eingeben, wenn er fuer das Feld das Schreibrecht erhalten will. Wird das Schreib-Passwort angegeben, kann das Feld gelesen und geschrieben werden.

WRITE GROUP
Die hier eingegebene Zeichenkette bezeichnet eine Gruppe von Feldern. Alle Felder, fuer die an dieser Stelle die gleiche Zeichenkette eingegeben wurde, werden als zur selben Gruppe gehoerig betrachtet. Liefert ein Nutzer das Schreib-Passwort fuer ein Feld in der Feldgruppe, ist es so, als wuerde er die Passwoerter fuer alle Felder in dieser Gruppe liefern, d.h., er hat mit einem Passwort Zugrif auf alle Felder dieser Gruppe.

3.3.2 Verschluesseln der Feld-Passworte

Datenschutz auf Feldebenen wird implementiert, indem unter Verwendung von 'Field Security Maintenance' Passworte festgelegt werden und dann verarbeitet (verschluesselt) werden, so dass eine binaere Datei entsteht, die von anderen WEGA-DATA-Funktionen gelesen werden kann. Das verlaeuft aehnlich wie die Eingabe der Struktur einer Bildmaske unter Verwendung von 'Screen Entry' und ihrer Verarbeitung unter Verwendung von 'Process Screen'. Dieses Programm nun verarbeitet die Passwort-Spezifikationen und erzeugt die binaere Passwort-Datei namens upasswd.

Unter 'Data Base Maintenance Menu' wird 'Process Field Passwords' (procpass) gewaehlt, wenn der aktuelle Satz von Passworten auf Feldebene verarbeitet werden soll. Der folgende Bildschirminhalt wird angezeigt:

```
[procpass]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                       Process Field Passwords


PROCEED? _
```

Wird Y eingegeben, wird das Programm fortgesetzt und erstellt im aktuellen Verzeichnis die Datei upasswd (oder in $DBPATH, falls diese Umgebungsvariable gesetzt ist - siehe Abschnitt 1.1.3). Wird N oder CTRL/U eingegeben, kehrt das Programm sofort zum Menue-Handler zurueck, ohne weitere Verarbeitungen vorzunehmen. Wird RETURN eingegeben, hat das keine Wirkung.

3.3.3 Verwendung der Zugriffsbeschraenkungen von WEGA

Die in den vorhergehenden Abschnitten beschriebenen Massnahmen koennen nur so sicher sein, wie die WEGA-Umgebung, in der sie angewandt werden. Koennen Nutzer Programme ablaufen lassen, die die Sicherheitsmassnahmen umgehen, oder koennen sie den Inhalt von Datenbank-Dateien direkt auslagern, erhalten sie so Zugriff zu Daten, auf die sie kein

Zugriffsrecht haben. In diesem Abschnitt soll daher erklaert werden, wie man unter Verwendung von WEGA-DATA eine sichere Umgebung schaffen kann.

Zuerst sollen mit dem Menue-Handler Zugriffsrechte erteilt werden. Das WEGA-DATA Superuser-Passwort darf nur vom am hoechsten priviligierten Nutzer verwendet werden und geheim zu halten. In Abschnitt 2.1.7 wurde beschrieben, wie die Nutzer-Identifikation und das Passwort fuer den Superuser modifiziert werden. Es wird empfohlen, dass die verschiedenen Datanbank-Funktionen, wie 'Schema Maintenance', 'Write Data Base Backup', 'Read Data Base Backup' und 'Reconfigure Data Base' den id's anderer Nutzer zugeordnet werden. Damit wird die Zahl der Personen moeglichst klein gehalten, die im Besitz von Superuser-id und -Passwort sein muessen. Natuerlich muss gut ueberlegt werden, wer Zugriff zu 'Employee Maintenance' und 'Group Maintenance' erhaelt.

'Executable Maintenance' ist ein kritischer Bereich. Da die WEGA-Zugriffsbeschraenkungen so festgelegt sind, dass Programme nur ueber WEGA-DATA ablaufen koennen, ist es wichtig, die in diesen Programmen verwendeten Verfahren zu verifizieren, bevor sie registriert werden. So kann beispielsweise ein Beschaeftigter zwar ein Leserecht fuer eine Pruefdatei haben, aber das bedeutet nicht notwendigerweise, dass er auch ein Programm zum Drucken dieser Datei registrieren kann.

Um nichtautorisierten Programmen den Zugriff auf die Datenbank zu verwehren, sollte eine spezielle WEGA Nutzer-id festgelegt werden. Dann ist diese Nutzer-id als Eigentuemer aller Systemdateien fuer ein Anwendungsbeispiel festzulegen, d.h. fuer Systemdateien wie wdata.db, file.db, file.dbr und upasswd. Diese Dateien sind nur fuer den Eigner les- und schreibbar zu machen. Dann ist das Programm wdata in das Eigentuemerverzeichnis dieser Nutzer-id zu bringen (zusammen mit allen anderen ausfuehrbaren Dateien von WEGA-DATA), und das "set user id bit" ist zu setzen (siehe chmod(1) im WEGA-Handbuch).

3.4 Statistiken

In diesem Abschnitt werden die Statistiken ueber die Datenbank beschrieben, die zur Ueberwachung von Groesse und Charakteristika der verschiedenen Datenbank-Parameter bereitsteht. Ueber folgendes wird eine Statistik gefuehrt: Dateigroesse, Prozentsatz des genutzten Speicherraums, Dichte der Hash-Tabelle und Konflikt-Informationen.

3.4.1 Statistiken ueber die Datenbank

Dieses Programm gibt einen Satz von statistischen Informationen ueber verschiedene Datenbank-Parameter aus. Aus 'Data Base Maintenance Menu' heraus wird 'Data Base

Statistics' gewaehlt, woduch das Programm gestartet wird. Es gibt keine Optionen und die Ausgabe wird an den Drucker gesendet. Der Report ist 79 Zeichen breit und wurde hier aus drucktechnischen Gruenden in der Breite reduziert.

DATE : 07/24/86 TIME : 11:13:07 PAGE: 1
SCHEMA REPORTS
DATA BASE STATISTICS

RECORD	EXPECTED	ACTUAL	PERCENT
her	12	6	50.0
modell	54	22	40.7
art	102	14	13.7
kunde	12	3	25.0
best	102	4	3.9
b	12	0	0.0

HASH TABLE 8.3% FULL
DATA BASE SIZE 33 BLOCKS

Es folgt die Erlaeuterung der Ueberschriften:

RECORD
Name des Datensatztyps.

EXPECTED
Die etwaige Anzahl der erwarteten Datensaetze. Aufgrund des Platzzuweisungsalgorithmus' ist diese Anzahl nicht genau gleich der in 'Schema Maintenance' eingegebenen erwarteten Anzahl.

ACTUAL
Die tatsaechliche Anzahl der in der Datei vorhandenen Datensaetze.

PERCENT
Der Prozentsatz des in der Datei bereits belegten Platzes. Eine gegebene Datei kann bis zu 166% voll sein, bevor durch Rekonfiguration mehr Platz zugeordnet werden muss.

HASH TABLE
Gefuellter Platz in der Hash-Tabelle. 50% ist der optimale Ladefaktor. Sind mehr als 70% des freien Platzes gefuellt, wird die Leistungsfaehigkeit von acckey und addrec verringert.

DATA BASE SIZE
Die maximale potentielle Groesse der Datenbank in 512-Byte-Bloecken.

3.4.2 Statistik ueber die Hash-Tabelle

Dieses Programm gibt statistische Angaben ueber verschiedene Parameter der Hash-Tabelle aus. Aus 'Data Base Maintenance Menu' wird 'Hash Table Statistics' ausgewaehlt, um das Programm zu starten. Es gibt keine Optionen und die Ausgabe wird an den Bildschirm gesendet. Ist man mit der Betrachtung des Bildschirms fertig, wird RETURN gedrueckt. Es folgt das Beispiel einer solchen Statistik:

```
[hts]                          WDATA SYSTEM
                            24 JUL 1986 - 15:25
                           Hash Table Statistics


                    HASH TABLE STATISTICS

 Total Entries              49
 Per Cent Filled            15.00
 Average Chain Length        1.96
 Longest Chain Length       7
 Starting Address           3528

                         Number of Chains of Length...
  2    3    4    5    6    7    8    9    10    >10
 4     1    1    1    1    1    0    0     0    0

 ->-> _
```

Die auf dem Bildschirm erscheinenden Eintraege haben folgende Bedeutung:

Total Entries
: Die Gesamtanzahl der belegten Plaetze in der Hash-Tabelle.

Per Cent Filled
: Prozentzahl der gefuellten Eintraege. Sind 50% der freien Plaetze gefuellt, hat die Tabelle ihre optimale Dichte erreicht. Sind 70% oder mehr gefuellt, wird die Leistungsfaehigkeit von acckey und addrec verringert.

Average Chain Length
: Die durchschnittliche Anzahl der aufeinanderfolgenden gefuellten freien Plaetze, die vor einem leeren freien Platz sind. Die Haelfte dieser Zahl gibt einen groben Ueberblick, wie oft geprueft werden muss, bevor man einen Datensatz findet. Im oben angefuehrten Beispiel wird jeder Datensatz "sofort" gefunden. Diese Zahl sollte kleiner als 6 sein.

Longest Chain Length
: Die laengste Folge von aufeinanderfolgenden gefuellten leeren Plaetzen. Hierdurch wird ein grober Ueberschlag

fuer die Zugriffszeit, die im unguenstigsten Fall benoetigt wird, vorgenommen.

Starting Address
Die Adresse des ersten leeren Platzes der laengsten Kette, falls man etwas in der Hash-Tabelle untersuchen moechte.

Number of Chains of Length...
Die Anzahl der Folgen aufeinanderfolgender gefuellter leerer Plaetze angegebener Laenge. Daurch erhaelt man eigentlich einen Ueberblick ueber die Guete des Hash-Verfahrens.

3.5 Dienstprogramme

In diesem Abschnitt werden die in WEGA-DATA zur Verfuegung stehenden Dienstprogramme beschrieben. Es gibt 4 verschiedene Programme, die die Bedienung des Anwendungssystems so unterstuetzen, dass die Datenbank immer wiederhergestellt werden kann. Ausserdem gibt es ein Programm, mit dem man grosse Datenmengen in eine Datenbank laden kann. Es hat sich in der Praxis gezeigt, dass es erforderlich ist, taeglich ein Backup der Datenbank anzulegen, insbesondere dann, wenn diese staendig aktualisiert wird. Wenn dann die Hardware versagt und die Konsistenz des Dateisystems nicht mehr gegeben ist oder wenn auf der Platte ein fehlerhafter Block auftritt, kann die Datenbank-Datei mit einem ziemlich aktuellen Stand wiederhergestellt werden. Dazu vorgesehen sind die Programme 'Write Data Base Backup' und 'Read Data Base Backup'.

Tritt ein Software- oder ein wiedergutzumachender Hardwarefehler auf, und entsteht der Verdacht, dass die Integritaet der Datenbank in Mitleidenschaft gezogen wurde, stehen Dienstprogramme zur Verfuegung, die kritische Teile der Datei wieder aufbauen koennen. Die Hash-Tabelle kann unter Verwendung von 'Hash Table Maintenance' wiederaufgebaut werden. Erhaelt man ploetzlich ueber den Schluessel keinen Zugriff mehr auf einen Datensatz oder eine ganze Gruppe von Datensaetzen, laesst man dieses Programm ablaufen, um das Problem zu loesen. Wenn Anfragen oder Programme, die explizite Beziehungen verwenden, keine uebereinstimmenden Werte liefern, koennen diese unter Verwendung von 'Explicit Relationship Maintenance' wiederhergestellt werden. Versagen alle beschriebenen Methoden, kann man mit SQL die Daten auslagern (Abschnitt 6.2.4), das Programm 'Create Data Base' (Abschnitte 3.2.1) ablaufen lassen, und dann die Daten unter Verwendung von 'Data Base Load' (Abschnitt 3.5.5 und 6.2.5) wieder laden.

3.5.1 Erstellen eines Backups der Datenbank

Diese Funktion erstellt ein Backup der Datanbank-Datei und

des Datenwoerterbuchs im aktuellen Verzeichnis. Das Programm findet heraus, wo sich die Datenbank-Datei befindet, ob es sich dabei um eine normale WEGA-Datei handelt oder ob sie auf mehrere Raw-Devices verteilt ist. Es berechnet die Anzahl der zu schreibenden Bloecke , indem es die Groesse der Datenbank aus der Datei selbst liest und kann das Backup bei grossen Datenbanken auch ueber mehr als einen Datentraeger ausdehnen. Das Programm bestimmt den Namen des Backup-Geraetes aus der Umgebungsvariablen BUDEV.

Vor dem Starten dieses Programms ist zu sichern, dass keine weiteren Nutzer die Datenbank verwenden. Nur so kann eine einwandfreie Kopie der Datenbank angefertigt werden. Um das Programm zu starten wird im 'System Menu' die 13, 'Write Data Base Backup', gewaehlt. Folgendes Bild erscheint auf dem Schirm:

```
[budb]                          WDATA SYSTEM
                              24 JUL 1986 - 15:25
                           Write Data Base Backup

This program copies the data base and data dictionary
to diskettes or tape. No one else should be using the
data base while this program is running.


PROCEED? _
```

Diese Mitteilung besagt: Dieses Programm kopiert die Datenbank und das Datenwoerterbuch auf Disketten oder Band. Waehrend des Ablaufs dieses Programms darf die Datenbank nicht anderweitig benutzt werden.

Vor Beantwortung des oben gezeigten Prompters ist zu sichern, dass das Backup-Medium (Diskette) beschrieben werden kann und dass es ordnungsgemaess im Laufwerk installiert ist. Es ist zu beachten, dass Disketten vor ihrer Benutzung zu formatieren sind. Ist das gesichert, kann hinter dem Prompter PROCEED? nun Y eingegeben werden. Danach wird RETURN gedrueckt. (Wird hinter dem Prompt PROCEED? ein N eingegeben oder CTRL/U gedrueckt, wird der Prozess abgebrochen und ins Menue zurueckgekehrt). Das Programm hat mehrere Prompter, die waehrend seiner Arbeit auftreten koennen:

```
  Mount first diskette/tape  ->->
```

Der erste Datentraeger des Backup-Mediums muss bereit sein. Handelt es sich dabei um Disketten wird das Backup i.a. mehrere Disketten benoetigen. Durch Druecken von RETURN wird die Backup-Datei angelegt.

```
  Backup device is off line or not write enabled-Try again?
```

Das Backup-Geraet ist nicht eingeschaltet, ist nicht angeschlossen, die Diskette oder das Band sind schreibgeschuetzt oder die WEGA Nutzer-id des Bedieners hat kein Lese/Schreib-Recht fuer das Backup-Geraet.

Soll der Versuch wiederholt werden, muss das Problem beseitigt und dann mit Y oder y geantwortet werden. Das Backup wird geschrieben. Soll das Programm abgebrochen werden, wird CTRL/U, N oder n eingegeben.

Mount next diskette/tape ->->

Das physische Ende des aktuellen Backup-Datentraegers wurde erreicht und es muessen noch mehr Datentraeger beschrieben werden. Zum Erhalt einer gueltigen Kopie der Datenbank muessen alle Datentraeger beschrieben werden.

Es wird eine neue Diskette oder ein neues Band eingelegt. Auf der Tastatur kann eine beliebige Taste gedrueckt werden, um den naechsten Datentraeger zu beschreiben.

The backup device environment variable(BUDEV)is not set->->

Diese Umgebungsvariable muss den Namen des Backup-Geraets enthalten. Das geschieht normalerweise im Verlauf der Installierung.

Durch Druecken einer beliebeigen Taste wird das Programm beendet.

Can't open data dictionary ->->

Das Datenwoerterbuch konnte nicht zum Lesen geoeffnet werden.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Es ist zu pruefen, ob die verwendete WEGA Nutzer-id das Leserecht fuer das Datenwoerterbuch hat.

Can't open data base file - BUDB ->->

Die Datenbank-Datei konnte nicht zum Lesen geoeffnet werden.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Es ist zu pruefen, ob die verwendete WEGA Nutzer-id das Leserecht fuer die Datenbank-Datei hat.

Can't open data base volume ->->

Eines der Raw-Devices fuer die Datenbank (spezifiziert in 'Volume Maintenance') konnte nicht zum Lesen geoeffnet werden.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Es ist zu pruefen, ob die verwendete WEGA Nutzer-

id das Leserecht fuer alle Datenbank-Datentraeger hat und dass die Eintraege der speziellen Datei fuer jeden Datentraeger intakt sind.

Read error - BUDB ->->

Beim Lesen der Datenbank oder des Datenwoerterbuchs trat ein Fehler auf.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Dieser Fehler tritt meist nur dann auf, wenn ein anderer Nutzer die Datenbank aktualisiert, waehrend das Backup angelegt wird. Es ist zu sichern, dass keine anderen Anwender die Datenbank verwenden und der Backup-Vorgang wird erneut gestartet. Der Fehler koennte auch auftreten (obgleich diese Moeglichkeit nur selten auftritt), wenn in der Datenbank ein Plattenblock fehlerhaft geworden ist und das Betriebssystem diesen nicht mehr lesen kann. Ist das der Fall, muss die Platte neu formatiert werden, um den schlechten Block auszuschliessen und die Datenbank muss aus dem vorhergehenden Backup wieder aufgebaut werden.

Write error - BUDB ->->

Beim Beschreiben des Backup-Mediums trat ein Fehler auf. Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck.

Wahrscheinlich ist der Datentraeger, bei dem der Fehler auftrat, fehlerhaft. Der Backup-Vorgang ist zu wiederholen, diesmal mit einem einwandfreien Datentraeger.

Backup complete ->->

Das Backup wurde erfolgreich erstellt. Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck.

3.5.2 Lesen eines Backups der Datenbank

Dieses Programm liest die von 'Write Data Base Backup' erstellten Backup-Disketten, um die Datenbank- und Datenwoerterbuch-Dateien wiederherzustellen. Den Namen des Backup-Geraets erhaelt es von der Umgebungsvariablen BUDEV. Vor Beginn des Wiederherstellungsprozesses ist zu sichern, dass die Datenbank von keinen weiteren Anwendern genutzt wird.

Aus dem 'System Menu' heraus wird 'Read Data Base Backup' gewaehlt, um das Programm zu starten. Auf dem Schirm erscheint folgendes Bild:

```
[redb]                          WDATA SYSTEM
                              24 JUL 1986 - 15:25
                            Read Data Base Backup

This program restores a copy of the data base and
data dictionary from diskettes or tape. No one
else should be using the data base while this
program is running.


PROCEED? _
```

Die erste Backup-Diskette oder erste Backup-Band sind einzulegen und es ist zu ueberpruefen, ob das Backup-Geraet on-line und betriebsbereit ist. Mit Y startet man den Wiederherstellungsprozess. Soll das Programm jetzt beendet und ins Menue zurueckgegangen werden, wird CTRL/U, N oder n gedrueckt. Nach Starten des Programms kann man das Programm an verschiedenen Punkten verlassen. Wurden jedoch vor Austritt ein oder mehrere Datentraeger (aber nicht alle) eingelesen, ist die Datenbank ungueltig und vor Benutzen der Datenbank muss ein anderes Backup vollstaendig eingelesen werden.

Waehrend der Arbeit verwendet das Programm die unten erlaeuterten Prompter:

Mount first diskette/tape ->->

Der erste Datentraeger des Backup-Mediums muss bereit sein. Handelt es sich dabei um Disketten wird das Backup i.a. aus mehreren Disketten bestehen. Durch Druecken von RETURN wird der erste Backup-Datentraeger eingelesen.

Backup device is off line - Try again?

Das Backup-Geraet ist nicht eingeschaltet oder ist nicht angeschlossen.

Soll der Versuch wiederholt werden, muss das Problem beseitigt und mit Y oder y geantwortet werden. Der Backup-Datentraeger wird gelesen. Soll das Programm abgebrochen werden, wird CTRL/U, N oder n eingegeben.

This is not the first diskette/tape - Try again?

Die Backup-Datentraeger sind intern durchnumeriert, um zu sichern, dass sie in der richtigen Reihenfolge gelesen werden. Der jetzt eingelegt Datentraeger ist aber nicht der erste.

Soll der Versuch wiederholt werden, muss die erste Diskette eingelegt und Y oder y eingegeben werden. Dann wird der

Datentraeger eingelesen. Soll das Programm beendet werden, wird CTRL/U, N oder n eingegeben.

Mount next diskette/tape ->->

Das physische Ende des aktuellen Backup-Datentraegers wurde erreicht und es muessen noch mehr Datentraeger eingelesen werden. Zum Erhalt einer gueltigen Kopie der Datenbank muessen alle Datentraeger in der richtigen Reihenfolge eingelesen werden.

Es wird die naechste Diskette oder das naechste Band eingelegt. Auf der Tastatur kann eine beliebige Taste gedrueckt werden, um den naechsten Datentraeger zu lesen.

Diskettes/tapes are out of sequence - Try again?

Die im Laufwerk befindliche Diskette (bzw. Band) ist nicht die nachfolgende der gerade eingelesenen.

Soll der Versuch wiederholt werden, ist die richtige Diskette (Band) einzulegen und dann mit Y oder y zu antworten. Der eingelegte Datentraeger wird gelesen. Soll aus dem Programm ausgetreten werden, wird CTRL/U, N oder n eingegeben. Wird an diesem Punkt ausgetreten, ist die Datenbank nicht gueltig, und dieses oder ein anderes Backup vollstaendig eingelesen werden.

Time stamps don't match - Try again?

Die Nummer des eingelegten Datentraegers stimmt, aber er gehoert zu einem Backup, das zu einem anderen Zeitpunkt angefertigt wurde.

Soll der Versuch wiederholt werden, ist die richtige Diskette (Band) einzulegen und dann mit Y oder y zu antworten. Der eingelegte Datentraeger wird gelesen. Soll aus dem Programm ausgetreten werden, wird CTRL/U, N oder n eingegeben. Wird an diesem Punkt ausgetreten, ist die Datenbank nicht gueltig, und dieses oder ein anderes Backup vollstaendig eingelesen werden.

This is not a backup diskette/tape - Try again?

Backup-Datentraeger sind mit einem speziellen Header versehen, so dass das Programm weiss, ob wirklich ein Backup-Medium eingelegt wurde. Beim vorliegenden Datentraeger handelt es sich nicht um ein Backup-Medium.

Soll der Versuch wiederholt werden, muss die erste Diskette eingelegt und Y oder y eingegeben werden. Dann wird der Datentraeger eingelesen. Soll das Programm beendet werden, wird CTRL/U, N oder n eingegeben.

The backup device environment variable(BUDEV)is not set->->

Diese Umgebungsvariable muss den Namen des Backup-Geraets enthalten. Das geschieht normalerweise im Verlauf der Installierung.

Durch Druecken einer beliebeigen Taste wird das Programm beendet.

Error writing data dictionary ->->

Das Datenwoerterbuch konnte nicht wiederhergestellt werden. (Auch die Datenbank-Datei wurde nicht wiederhergestellt). Datenbank und Datenwoerterbuch sind nun ungueltig.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Zur Wiedergewinnung muss dieses Backup neu oder ein anderes eingelesen werden. Es ist zu ueberpruefen, ob auf der Platte genuegend Speicherplatz fuer die Dateien vorhanden ist.

Can't create data dictionary ->->

Das Datenwoerterbuch konnte nicht wiederhergestellt werden. (Auch die Datenbank-Datei wurde nicht wiederhergestellt). Datenbank und Datenwoerterbuch sind nun ungueltig.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Zur Wiederherstellung muss das Backup entweder von einem WEGA-Nutzer, der Dateien im Datenbank-Verzeichnis anlegen darf, eingelesen werden oder aber der Schutz des Verzeichnisses ist so zu veraendern, dass der jetzige Nutzer Dateien anlegen darf.

Can't open data base volume ->->

Eines der Raw-Devices fuer die Datenbank (spezifiziert in 'Volume Maintenance') konnte nicht zum Lesen geoeffnet werden. Datenbank und -woerterbuch sind jetzt ungueltig.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Es ist zu pruefen, ob eine der beiden folgenden Moeglichkeiten vorliegt. Die verwendete WEGA Nutzer-id hat nicht das Schreibrecht fuer das Dateisystem (Volume) oder die Eintraege der speziellen Datei sind geloescht oder zerstoert.

Tape read error - REDB ->->

Beim Lesen des Backup-Mediums trat ein Fehler auf. Datenbasis und -woerterbuch sind nicht mehr gueltig.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Vermutlich ist das Backup fehlerhaft. Man kann es erneut versuchen einzulesen, aber wahrscheinlich muss ein anders Backup verwendet verden.

Write error - REDB ->->

Beim Beschreiben der Platte trat ein Fehler auf. Datenbasis und -woerterbuch sind nicht mehr gueltig.

Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck. Wahrscheinlich ist der Speicherplatz auf der Platte zu klein. Dann muss mehr Speicherplatz bereitgestellt und das Einlesen erneut versucht werden.

Restore complete ->->

Der Wiederherstellungslauf wahr erfolgreich. Durch Druecken einer beliebigen Taste kehrt man ins Menue zurueck.

3.5.3 Verwaltung der Hash-Tabelle

Mit diesem Programm wird die Hash-Tabelle von Grund auf neu aufgebaut. Dieses Programm ist dann zu verwenden, wenn man nicht mehr durch den Primaerschluessel auf einen Datensatztyp zugreifen kann, wenn man ihn aber durch Absuchen der Datei oder durch Verwendung eines Sekundaerschluessels finden kann. Das kann passieren, wenn man den internen Datentyp eines Schluesselfeldes unter Verwendung von 'Schema Maintenance' aendert und dann die Datenbank rekonfiguriert. Ein Beispiel fuer die Aenderung des internen Datentyps ist: Aendern eines Feldes NUMERIC 3 auf NUMERIC 9 (Es wird von kurz auf lang geaendert).

'Hash Table Maintenance' bringt zunaechst den Hash-Tabellen-Index auf Null und liest dann die Datenbank durch, wobei fuer jeden gefundenen Schluessel ein Eintrag vorgenommen wird. Es gibt weder Optioen noch eine Ausgabe.

Das Programm wird gestartet, indem im 'Data Base Maintenance Menu' 'Hash Table Maintenance' gewaehlt wird. Auf dem Bildschirm erscheint:

```
[rekey]                        WDATA SYSTEM
                          24 JUL 1986 - 15:25
                       Hash Table Maintenance

This program rebuilds the data base hash table index.
No one else should be using the data base while this
program is running.


PROCEED? _
```

Auch bei der Arbeit dieses Programms duerfen keine weiteren Anwender die Datenbank benutzten.

Das Programm wird durch Eingabe von Y oder y gestartet. CTRL/U, N oder n gestattet die Rueckkehr ins Menue. Nach Beendigung seiner Arbeit zeigt das Programm die Meldung:

```
  Program complete. ->->
```

Durch Druecken einer beliebigen Taste kehrt man in den Menue-Handler zurueck.

3.5.4 Verwaltung expliziter Beziehungen

Mit diesem Programm werden in der Datenbank die expliziten Beziehungen von Grund auf neu aufgebaut. Dieses Programm wird dann vewendet, wenn die Funktionen, die auf die expliziten Beziehungen bezug nehmen, (unisort, makeset, setsize, nextrec, prevrec) zum Absturz fuehren oder fehlerhafte Ergebnisse liefern.

Zunaechst setzt 'Explicit Relationship Maintenance' die existierenden Pointer auf Null, dann liest es die Datenbasis durch und linked die Datensaetze, wo immer erforderlich, neu. Das ist moeglich, weil eine bestimmte Menge redundanter Daten gespeichert ist, die angeben, welche expliziten Beziehungen bestehen sollten. Es gibt keine Optionen. Das Programm wird gestartet, indem aus 'Data Base Maintenance Menu' heraus 'Explicit Relationship Maintenance' (repoint) gewaehlt wird. Es erscheint folgendes Bild auf dem Schirm:

```
[repoint]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                    Explicit Relationship Maintenance

This program rebuilds all the explicit relationship pointer
chains in the data base. No one else should be using the
data base while this program is running.


PROCEED? _
```

Das Programm wird durch Eingabe von Y oder y gestartet. CTRL/U, N oder n gestattet die Rueckkehr ins Menue. Nach Beendigung seiner Arbeit zeigt das Programm die Meldung:

```
  Program complete. ->->
```

Durch Druecken einer beliebigen Taste kehrt man in den Menue-Handler zurueck.

Wurde das Programm erfolgreich abgeschlossen, erfolgt keine weitere Ausgabe. War es nicht in der Lage, einige der Datensaetze neu zu linken, erscheint stattdessen folgende Meldung:

```
  -- WARNING --
  All the records were not relinked.
  You must manually relink them, following
```

the instructions given in the file "repoint.err".

Diese Meldung bedeutet, dass repoint nicht in der Lage war, fuer einige Datensaetze die Pointer zu reparieren. Diese Datensaetze konnten nicht repariert werden, weil die Pointer-Werte, durch die sie mit ihren zugehoerigen Referenz-Dateien verbunden sind, nicht gueltig waren und Datenwerte in den Feldern, auf die Bezug genommen wurde, nicht in den Referenz-Dateien vorhanden waren (d.h., ein Aufruf von pfield fuer diese Werte, wuerde -3 zurueckgeben). Das bedeutet normalerweise, dass der Datensatz durch einen Fehler in einem Programm oder durch Versagen der Hardware zerstoert wurde. Es gibt aber auch noch eine andere Moeglichkeit. Man erhaelt naemlich denselben Wert, wenn niemals ein gueltiger Datenwert in einem expliziten Beziehungsfeld gespeichert war, d.h. man erhaelt diese Meldung, wenn repoint in einer Datenbank ablaeuft, in der einige explizite Beziehungsfelder niemals gesetzt wurden.

Die Datei repoint.err, die im aktuellen Verzeichnis angelegt wird, enthaelt eine Liste von Datensaetzen, die nicht bearbeitet werden konnten. Darin sind folgende Informationen enthalten:

Record Type

Ist der Name des Datensatztyps mit dem fehlerhaften Pointer (in der expliziten Beziehung des Tochter-Datensatztyps).

Location

Diese Nummer ist der relative Platz, auf dem sich der bestimmte Datensatz innerhalb der Datei befindet. Datensatztyp und Platz koennen von der Funktion setloc in 'Data Base Test Driver' zum Aufffinden des Datensatzes verwendet werden. Das Kommando waere folgendes:

:setloc datensatztyp platz

Related Record

Ist der Name des zum fehlerhaften in Beziehung stehenden Datensatztyps (in der Beziehung der Vater-Datensatztyp).

Reference Field

Ist der Name des expliziten Beziehungs-Feldes im Tochter-Datensatztyp. Es handelt sich hierbei um das Feld, das aktualisiert werden muss, um den Datensatz wieder mit der Liste der "Vaeter" linken zu koennen. Nach Aktualisierung des Datensatzes, kann man 'Data Base Test Driver' verwenden (Erklaerung siehe Location).

3.5.5 Laden der Datenbank

Das Programm 'Data Base Load' gestattet das Laden neuer

Datensaetze und das Aktualisieren existierender Datensaetze in einer WEGA-Datenbank unter Verwendung einer normalen ASCII-Datei. Enthaelt die ASCII-Datei neue Schluesselwerte, werden die Daten eingefuegt. Enthaelt die Datei existierende Schluesselwerte, werden die Felder in den existierenden Datensaetzen mit den in der ASCII-Datei enthaltenen Informationen aktualisiert. Die ASCII-Datei kann auf verschiedene Weise angelegt werden: durch Auslagern von Daten aus einem existierenden WEGA-Anwendungsbeispiel unter Verwendung der von dieser Anwendung gelieferten Werkzeuge, durch Verwendung eines Editors zur Eingabe von Daten, durch Verwendung von SQL (Abschnitt 6) zum Auslagern von Daten aus einer existierenden WEGA-Datenbank und so weiter. Die einzige Bedingung ist, dass das Format der Datei so angelegt ist, wie es fuer das Ladeprogramm erforderlich ist. Hat man eine Datei in einem anderen Format, kann man eines der WEGA-Textverarbeitungsprogramme wie sed(1) oder awk(1) verwenden, um die Datei in das erforderliche Format zu bringen.

Da man beim Anlegen der ASCII Datei-wahrscheinlich mit der Shell arbeiten wird, ist dieses Programm so angelegt, dass es nicht unter dem Menue-Handler, sondern unter der Shell benutzt wird. Damit kann man die Eigenschaften von WEGA zur Umlenkung der E/A und der Pipes fuer das Laden der Daten benutzen. Es gibt jedoch fuer dieses Programm auch ein SQL-Interface, mit dem das Laden und Auslagern von Daten innerhalb einer WEGA-DATA-Anwendung wesentlich erleichtert wird. In Abschnitt 6.2.4 sind Informationen, wie man Daten von SQL aus in WEGA-Dateien auslagern kann und in Abschnitt 6.2.5 wird beschrieben, wie die Daten in die Datenbank zurueckgeladen werden koennen.

Will man das Programm 'Data Base Load' von der Shell aus benutzen, ist zu sichern, dass die Umgebingsvariablen PATH und WDATA korrekt gesetzt sind. Weitere Informationen siehe Abschnitt 1.1.3, Umgebungsvariable. 'Data Base Load' erwartet, wenn es mit einer Shell-Kommando-Zeile aufgerufen wird, 4 Parameter.

```
DBLOAD [-n] [-u] datenbank datsatyp datei spez-datei
```

Die Parameter werden folgendermassen verwendet:

Parameter	Verwendung
-n	Ergaenzungsmodus; stimmt ein Datensatzschluessel in datei mit einem Datensatzschluessel in der Datenbank ueberein, wird eine Warnung ausgegeben, die auf den doppelten Datensatzschluessel hinweist. Der Datensatz in der Datenbank wird nicht aktualisiert.
-u	Aenderungsmodus; das erste Feld einer jeden Zeile in datei muss eine Datensatzposition enthalten. Der Datensatz, der sich in der Datenba-

sis auf dieser Position befindet, wird mit den Werten der restlichen Felder aktualisiert. (siehe setloc, Abschnitt 10)

Standardmaessig (wenn weder -n noch -u angegeben sind) wird ein neuer Datensatz hinzugefuegt, wenn der aktuelle Schluessel in datei noch nicht existiert oder wenn doch, der mit diesem Schluessel existierende Datensatz aktualisiert.

datenbank Name der zu ladenden Datenbank-Datei. Ist entweder file.db, wenn Daten in die Datenbank geladen werden oder wdata.db, wenn Daten in das Datenwoerterbuch geladen werden.

datsatyp Name des Datensatztyps, in den die Daten geladen werden.

datei Name der ASCII-Eingabedatei, die die zu ladenden Daten enthaelt. Ist dieser ein Bindestrich (-), wird die Standardeingabe benutzt, d.h. DBLOAD kann am Ende einer Pipe(line) benutzt werden.

spez-datei Der Name einer anderen ASCII-Datei, die die Liste der Felder enthaelt, die innerhalb der Datensatztypen zu aktualisieren sind, und zwar in der Reihenfolge, in der sie in der Eingabedatei aufgelistet sind.

Die Spezifikationsdatei beschreibt das Format der zu ladenden Eingabedatei. Die Datei besteht aus einer einzigen Zeile, in der ein Eintrag fuer jedes zu aktualisierende Feld ist. Nicht alle Felder des Datensatztyps muessen aufgelistet werden, nur diejenigen, die aktualisiert werden sollen. Die einzige Bedingung ist, dass, falls der zu aktualisierende Datensatztyp einen Schluessel hat, dieser Schluessel eins der Felder sein muss. Die Felder muessen auch nicht in einer bestimmten Reihenfolge auftreten - die Reihenfolge innerhalb der Spezifikationsdatei muss nur mit der Reihenfolge in der Eingabedatei uebereinstimmen.

Die Spezifikationsdatei hat folgendes Format:

```
feld[sep]feld[sep]feld...[sep]
```

wobei feld der Name (Kurzname oder ausfuehrlicher Name) des Datenbank-Feldes ist und [sep] ein aus einem Zeichen bestehender Separator, der das Feldende markiert. Felder vom Typ COMB sind nicht zu verwenden, sondern sind sowohl innerhalb der Spezifikations- als auch innerhalb der Eingabedatei in ihre Bestandteile aufzuspalten.

Der Separator darf kein Buchstabe, keine Ziffer und kein Unterstreichungszeichen (_) sein, da diese in Feldnamen verwendet werden. Er muss auch so gewaehlt werden, dass es in keinem STRING-Feld auftritt, da sonst das Programm

durcheinander kommt. Gut eignet sich als Trennsymbol z.B. ein senkrechter Strich (|). Es ist zu beachten, dass das letzte Zeichen in einer Spezifikationsdatei der Separator sein muss. Man muss zur Trennung von Feldern nicht immer denselben Separator verwenden, aber der in der Spezifikationsdatei verwendete muss darselbe sein, wie der in der Eingabedatei verwendete.

Die Eingabedatei hat das in der Spezifikationsdatei beschriebene Format. Jeder Datensatz der Eingabedatei besteht aus einer Anzahl von Feldern, die durch Separatoren getrennt sind und mit Newline abgeschlossen sind. Alle ueberfluessigen Leerzeichen im Feld werden ignoriert, waehrend STRING-Felder, die kuerzer sind, als bei ihrer (Datensatztyp-)Definition festgelegt, rechts mit Leerzeichen aufgefuellt werden. In den folgenden als Beispiel verwendeten Eingabe- und Spezifikationsdateien wird der Schluessel des Datensatzes nicht als erstes, sondern als letztes Feld aufgelistet, um ein Beispiel fuer das freie Format der Dateien zu geben:

Eingabedatei
```
Buchhandlung "Samuel Butler"|Urgasse 12|Erewhon|1820|
                                                843255|446990|1
Theo Retisch|Wunschallee 1102|Irgendwo|1111|0|566331|2
1000 Grosse Dinge|Am Steilhang 1|Bad Berg|1984|449277|
                                                    653575|3
```

Spezifikationsdatei
```
kuname|kustr|kort|kuplz|kutelex|kuruf|kundennummer
```

Im folgenden sind die Fehlermeldungen aufgefuehrt, die bei Verwendung von 'Data Base Load' auftreten koennen, ebenso wie vorgeschlagene Massnahmen zur Behebung der Fehler. Weitere Fehlermeldungen sind in Abschnitt 10.2 aufgefuehrt.

```
usage:DBLOAD [-n|-u]<database><record type><file><specfile>
```

Es wurde nicht die korrekte Anzahl Parameter an DBLOAD geliefert. Versuch muss unter Verwendung der 4 aufgelisteten Parameter wiederholt werden. Wird eine Option verwendet, muss es sich dabei um -n oder -u handeln.

```
  Invalid format for field ffff, record nnnn
```

Das sich im Datensatz Nummer nnnn in der Eingabedatei befindliche Feld namens ffff hat ein falsches Format. Suchen Sie ein ungueltiges Datum, eine falsche Zeitangabe oder einen numerischen Wert mit enthaltenen Buchstaben. Falls jeder Datensatz dasselbe Problem aufweist, passen wahrscheinlich Spezifikationsdatei und Eingabedatei nicht zusammen.

Record nnnn is a duplicate key

Der in der Eingabedatei befindliche Datensatz mit der Nummer nnnn hat einen Schluesselwert, der bereits in der Datenbank existiert. Das ist kein Fehler, wenn man keine neuen Datensaetze einfuegt, sondern nur existierende aktualisiert.

Invalid value for field ffff, record nnnn

Das Feld namens ffff hat eine explizite Beziehung zu einem anderen Feld in der Datenbank und der Wert, der fuer ffff in der Eingabedatei im Datensatz mit der Nummer nnnn steht, stimmt nicht mit der Bezugsdatei ueberein. Entweder muss ein Datensatz in die Bezugsdatei eingefuegt werden, der dem Wert des Feldes entspricht, oder der Wert muss geaendert werden, so dass er einem existierenden (Bezugs-)Datensatz entspricht.

Invalid record type: rrrr

Der Datensatztyp namens rrrr existiert nicht in der Datenbank, die geladen werden soll. Entweder wird die falsche Datenbank-Datei verwendet oder ein falscher Datensatztyp.

Can't open specification file: ffff

Die Spezifikationsdatei kann nicht eroeffnet werden. Es ist zu ueberpruefen, ob der Nutzer das Recht hat, die Datei zum Lesen zu oeffnen.

Invalid field name: ffff

Das Feld namens ffff existiert nicht in der Datenbank. Es wurde wahrscheinlich in der Spezifikationsdatei falsch geschrieben.

Field ffff is not in record type rrrr

Obwohl das Feld namens ffff in der Datenbank an sich existiert, ist es nicht im Datensatztyp rrrr. Entweder handelt es sich um den falschen Feldnamen oder um den falschen Datensatztyp.

No more Space for record rrrr

In file.db ist kein Platz mehr, um zusaetzliche Datensaetze vom Typ rrrr zu speichern. Das geschieht, wenn die Datei 66% mehr Datensaetze erhaelt, als beim Entwurf vorgesehen waren. Entweder muessen einige existierende Datensaetze vom Typ rrrr geloescht werden oder 'Schema Maintenance' muss verwendet werden, um die erwartete Anzahl von Datensaetzen vom angegebenen Typ zu erhoehen und dann die Datenbank zu rekonfigurieren.

Can't open file ffff

Die Datei namens ffff kann nicht eroeffnet oder gefunden werden. Es ist zu pruefen, ob der Nutzer fuer die Datei Leserecht besitzt, und ob die Datei existiert. Es ist zu pruefen, ob der Name der Datei korrekt geschrieben wurde.

There is insufficient memory for record buffer

Im zugewiesenen Datensatz-Puffer ist nicht genug Platz, um die aktuellen Eingabe-Datensaetze einzulesen. Die Groesse des Datensatzes muss verringert werden.

Load interrupt, nnnn records added, xxxx records updated

Mit dieser Meldung wird man davon in Kenntnis gesetzt, dass der Daten-Ladeprozess unterbrochen wurde und es wird angegeben, wieviele Datensaetze zur Zeit der Unterbrechung aktualisiert bzw. hinzugefuegt worden waren.

unable to open database file

Es wurde ein falscher Datenbank-Dateiname verwendet. Es ist zu pruefen, ob die korrekte Schreibweise gewaehlt wurde, ob das Leserecht in Ordnung ist und ob man sich im richtigen Verzeichnis befindet. Der Dateiname muss sein: file.db oder wdata.db

4. SFORM - EIN WERKZEUG ZUR ERSTELLUNG VON BILDMASKEN

Eine Bildmaske liefert die grundsaetzlichen Mittel fuer die Eingabe und Untersuchung von Daten in der Datenbank. Das SFORM-Paket von WEGA-DATA ermoeglicht einen leichten und schnellen Aufbau und das Aendern von Bildmasken. 'Paint Screen', der Bildmasken-Editor von SFORM ist das wichtigste Werkzeug fuer Aufbau und Aendern von Bildmasken.

Eine Bildmaske kann auf drei verschiedene Arten verwendet werden. Erstens kann es mit dem vielseitig verwendbaren Dateneingabeprogramm ENTER fuer einfache Dateneingaben und Anfragen verwendet werden. Im Kapitel 5 werden die Anforderungen besprochen, die fuer die Verwendung von ENTER an den Aufbau der Bildmasken gestellt werden. Zweitens kann eine Bildmaske mit einem SQL-Skript verbunden werden, so dass Parameter eingegeben werden koennen, die in dem Skript vor dessen Abarbeitung substituiert werden. In Abschnitt 6.2.6 wird beschrieben, wie SFORM-Bildmasken zusammen mit SQL (SQL ueber Bildmasken) verwendet werden koennen. Drittens gibt es zur vollstaendigen Steuerung der Verarbeitung der Bildmasken einen Satz von Funktionen, der durch Kundenprogramme (z.B. C-Programme) aufgerufen werden kann, um Bildmasken anzuzeigen oder zu loeschen und Daten zu akzeptieren oder anzuzeigen. Weitere Informationen siehe Programmiersprachen-Interface, Abschnitt 10.

Bevor wir zur Erstellung und Modifizierung von Bildmasken kommen, sollen die Elemente betrachtet werden, aus denen eine Bildmaske besteht. Hier eine typische Bildmaske, die zum Aktualisieren und zur Abfrage einer einfacher Bestellungsdatei benutzt werden koennte:

```
[best]                          WDATA SYSTEM
                            24 JUL 1986 - 15:25
                          Bestellung Maintenance

best nummer :
best datum  :
kundennummer:


[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

Eine Bildschirmmaske besteht aus drei Arten von Elementen. Erstens aus dem Namen der Bildmaske, der im Beispiel best ist. Es ist erforderlich, dass dieser Name nur einmal auftritt, damit eindeutig darauf bezug genommen werden kann. Die statischen, unveraenderbaren, "dekorativen" Elemente, die den eigentlichen Aufbau der Maske ausmachen, werden als Prompter bezeichnet. Im Beispiel waeren das 'best nummer :', 'best datum :' und 'kundennummer'. Zusaetzlich koennten noch erlaeuternde Bemerkungen oder Ueberschriften auftreten. Prompter dienen dazu, das dritte

Element der Masken, die Bildmasken-Felder zu kennzeichnen. Bildmasken-Felder kann man sich als "Fenster" vorstellen, durch die man in die Datenbank-Datei hineinschauen und die Werte der Datenbank-Felder fuer den aktuellen Datensatz betrachten kann. Im Beispiel kann man mit der Bildmaske zur Verwaltung von Bestellungen die Datenbankfelder mit der Nummer des Kunden, der Nummer der Bestellung und dem Datum der Bestellung "betrachten".

Jedes Bildmasken-Feld besteht aus seinem Namen und einigen der folgenden Elemente:

1. Dem Datenbank-Feld zur Eingabe oder Anzeige auf dem Bildmaske. Dasselbe Datenbank-Feld kann mehr als einmal unter verschiedenen Bildmasken-Feldnamen auftreten. Das ist ein optionelles Charakteristikum des Bildmasken-Feldes.

2. Typ und Anzeigelaenge des anzuzeigenden Feldes. Ist ein Datenbank-Feld nicht angegeben, koennen Typ- und Laengendefinitionen zur Festlegung einer Editierungsmaske fuer das gegebene Bildmasken-Feld genutzt werden. Anderenfalls werden Typ und Laenge durch das Datenbank-Feld festgelegt. Das ist ein optionelles Charakteristikum des Bildmasken-Feldes.

3. Die xy-Koordinaten, die angeben, wo das Feld auf dem Schirm eingegeben und angezeigt wird.

4. Der dem Bildmasken-Feld zugeordnete Prompter. Es handelt sich hierbei um eine Text-Zeichenkette, die ausgegeben wird, wenn die Bildmaske auf einem Terminal angezeigt wird. Es handelt sich um ein optionelles Charakteristikum eines Bildmasken-Feldes.

5. Die xy-Koordinaten des Prompters. Bildmasken-Felder werden entsprechend der xy-Koordinaten ihrer Prompter numeriert.

Es ist zu beachten, dass sowohl das Datenbank-Feld als auch der Prompter optionelle Elemente sind, dass jedoch, sollten beide fehlen, das Bildmasken-Feld voellig "unsichtbar", und letztendlich nutzlos ist.

Da die gesamten Bildmaskeninformationen getrennt von den Programmen gespeichert werden, koennen Bildmasken einfach modifiziert werden, ohne dass Programm-Quellkode modifiziert oder neu kompiliert werden muss. Ausserdem ist es moeglich, dass sich mehrere Programme ein und dieselbe Bildmaske teilen. Ausserdem sind die Programme in einem gewissen Grad unabhaengig vom den Display-Attributen und koennen ohne Bezugnahme auf xy-Koordinaten geschrieben werden. Auch Hardware-Charakteristika verschiedener Terminals koennen im Kundenprogramm vernachlaessigt werden, wodurch es moeglich wird, dasselbe Programm fuer verschiedene Terminaltyps zu verwenden. Diese Eigenschaften fuehren

dazu, dass die Programme leichter zu schreiben, zu verstehen und zu verwalten sind.

Nach der Definition einer Bildmaske, wird SFORM zum Anlegen von zwei Dateien verwendet, die von Programmen (in einer "normalen" Programmiersprache) verwendet werden, die auf die Bildmaske Bezug nehmen wollen. Die erste Datei namens bm_name.h muss in den Quellkode von Funktionen, die namentlich Bezug auf Bildmaskenfelder nehmen wollen, durch include-Anweisungen eingefuegt werden. Sie enthaelt eine Liste von Definitionen zu den Bildmasken-Feldern, die den Programmen gestattet, auf Bildmasken-Felder ueber ihren Namen, anstatt ueber ihre Nummer Bezug zu nehmen. Die Nummern der Bildmasken-Felder werden logisch fortlaufend von oben nach unten und von links nach rechts entsprechend den xy-Koordinaten des Prompters zugeordnet. In der zweiten Datei, namens bm_name.q, ist eine binaere Darstellung des Bildmasken-Formats und der Feldnamen enthalten. Diese Datei wird waehrend der Laufzeit zur tatsaechlichen Anzeige der Bildmaske verwendet.

Am einfachsten kann man eine Bildmasken unter Verwendung des Programms 'Create Default Screen Form' erstellen, das selbstaendig einfache Bildmasken entwirft. Hat man erst einmal eine Standardmaske oder will man die Maske nach eigenen Vorstellungen erzeugen, kann man 'Paint Screen' oder 'Screen Entry' verwenden. 'Paint Screen' ist ein Editor, mit dem man den gesamten Bildschirm bearbeiten kann, den Cursor auf dem Bildschirm bewegen, Prompter und Bildmasken-Felder nach individuellen Wuenschen eintragen kann. Es koennen auch spezielle Anzeigemodi eingegeben werden, z.B. Negativanzeige und Unterstreichung. Die vom Programm verwendeten Kommandos sind aehnlich denen, die von anderen Texteditoren verwendet werden. Es ist auch moeglich, dass man die Kommandotasten den eigenen Beduerfnissen entsprechend programmiert.

Das Programm 'Screen Entry' verlangt die Angabe der xy-Koordinaten der Prompter und Bildmasken-Felder und wurde aus Kompatibilitaetsgruenden aufgenommen. Bildmasken, die mit 'Paint Screen' und 'Screen Entry' erzeugt wurden, sind identisch, so dass man mit beiden Programmen dieselben Bildmasken erstellen und modifizieren kann.

Will man die SFORM-Werkzeuge verwenden, waehlt man das 'System Menu' und dann das 'SFORM Menu' (4). Es wird ein Menue mit folgenden Optionen angezeigt:

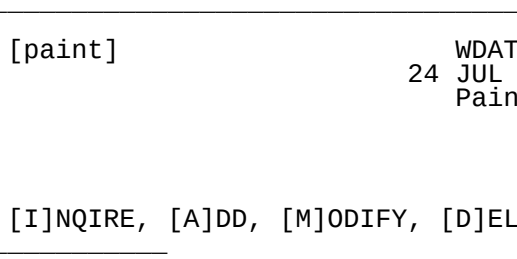
```
[sfmenu]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                              SFORM Menu

1. Paint Screen

2. Screen Entry

3. Test Screen

4. Process Reports

5. Screen Reports

6. Restore Screen

7. List Screens

8. Create Default Screen Form

SELECTION: _
```

In den folgenden Abschnitten sollen diese Optionen erlaeutert werden.

4.1 'Paint Screen'

Das Paint-Screen-Programm, PAINT, ist ein Editorprogramm zur Erstellung, Aenderung und Anzeige von SFORM-Bildmasken. Durch PAINT werden Bildmasken unter Verwendung des WEGA-DATA-Datenwoerterbuchs gespeichert und der Zugriff zu ihnen ermoeglicht. Nachdem Sie einen Maskennamen angegeben haben, koennen Sie den Cursor auf dem Bildschirm bewegen und damit Prompter und Bildschirmfelder ergaenzen, aendern und loeschen. Die Standardkommandos sind in Abschnitt 4.1.1 beschrieben. Auf Wunsch koennen diese Kommandos geaendert werden (siehe Abschnitt 4.1.2, Angepasste Paint-Kommandos).

Das Programm wird aktiviert, indem im SFORM-Menue 'Paint Screen', 1, gewaehlt wird. Danach erscheint folgende Darstellung auf dem Schirm:



Mit dieser Bildmaske wird Ihnen die Abfrage, das Hinzufuegen, Aendern und Loeschen von Bildmasken, Promptern und Bildmasken-Feldern ermoeglicht. Im Editierbereich, den freigelassenen Zeilen 3 bis 22, koennen Sie Ihre eigene Bildmaske gestalten. Der Ausgangspromter, der Modus-Prompter, wird folgendermassen verwendet:

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE:

Mit diesem Prompter koennen Sie einen Betriebsmodus fuer den Editierbereich waehlen. Durch Druecken von CTRL/U kehren Sie in den Menuehandler zurueck. Die verschiedenen Modi haben folgende Bedeutung:

i - Abfragemodus; mit dem man die Struktur einer existierenden Bildmaske sehen, aber nicht aendern kann.

a - Ergaenzungsmodus. Damit kann man neue Bildmasken dem Datenwoerterbuch hinzufuegen und dann Prompter und Bildschirmfelder hinzufuegen, aendern und ergaenzen.

m - Aenderungsmodus. Mit diesem kann man Prompter und Bildmaskenfelder existierender Bildmasken ergaenzen, aendern und loeschen.

d - Loeschmodus. Mit diesem kann man existierende Bildmasken mit allen dazugehoerigen Promptern und Bildmaskenfeldern aus dem Datenwoerterbuch loeschen. Mit Loeschen einer Maske werden alle Verweise auf diese in den Menues, 'ENTER Screen Registrations' und 'SQL Screen Registrations' gestrichen.

Waehlt man eine der angefuehrten Moeglichkeiten, erscheint der aktuelle Modus in der oberen linken Ecke des Bildschirms und der Modus-Promter wird durch den Prompter SCREEN: ersetzt. Dort muessen Sie den Namen der Bildmaske eingeben, den Sie bearbeiten wollen. Bei Druecken von CTRL/U wird der aktuelle Modus geloescht und der Modus-Promter erscheint erneut auf dem Schirm.

Befinden Sie sich im Abfragemodus, erscheint die Bildmaske auf dem Schirm. Druecken Sie RETURN waehrend Sie sich im Abfragemodus befinden, verschwindet das auf dem Schirm befindliche Bild und es erscheint der Prompter SCREEN. Sind

Sie im Loeschmodus, wird die Bestaetigung fuer den Loeschbefehl abgefordert. Sie koennen entweder y (Schirm loeschen) oder n (Schirm nicht loeschen) druecken. (Druecken von CTRL/U bedeutet Schirm nicht loeschen.)

Ergaenzungs- und Aenderungsmodus arbeiten nach Eingabe des Bildmaskennames in gleicher Weise. Der einzige Unterschied besteht darin, dass im Ergaenzungsmodus dem Datenwoerterbuch eine neue Bildmaske hinzugefuegt wird, deren Name ueber die Tastatur eingegeben wurde. In beiden Faellen wird der Cursor auf Zeile 3 gesetzt und wartet auf ein PAINT-Kommando. Nach Beendigung der Aufbereitungsarbeit koennen Sie durch Druecken von q die aktuelle Bildschirmdarstellung vom Schirm abnehmen. Danach wird folgender Prompter auf dem Schirm erscheinen:

```
[S]ave, [D]on't save, [R]esume editing _
```

Bei Eingabe von s, werden die von Ihnen vorgenommenen Veraenderungen im Datenwoerterbuch gespeichert und die Bildmaske wird so verarbeitet, dass aktualisierte .q- und .h-Dateien entstehen. Bei Eingabe von d werden die von Ihnen vorgenommenen Veraenderungen nicht gespeichert, so dass die Bildschirmdarstellung die gleiche wie vor Beginn des Editierens ist. Bei Eingabe von r (oder CTRL/U) koennen Sie das Editieren der Bildmaske fortsetzen.

Aufgrund der von PAINT zur Zerlegung der Bildmaske in Elemente des Datenwoerterbuchs verwendeten Algorithmen, koennen kleine Unterschiede zwischen den von 'Screen Entry' und 'Process Screen' erzeugten .q- und .h-Dateien auftreten (Abschnitt 4.2 und 4.4). Es ist also am besten, wenn man fuer die Erstellung und Modifizierung von Bildmasken entweder nur PAINT oder nur 'Screen Entry' verwendet.

4.1.1 Befehle fuer das Editieren der Bildmaske

PAINT-Bildmaskeneditier-Befehle koennen in vier verschiedene Gruppen eingeteilt werden: Befehle fuer Cursorbewegungen, Befehle zum Editieren von Promptern, Befehle zum Editieren von Bildmasken-Feldern und verschiedene andere Befehle. Hier eine Zusammenfassung aller PAINT-Befehle:

Befehle fuer Cursorbewegungen

Cursorbewegung	Taste(n)
Cursor nach links	Taste 'Pfeil links' Backtab h CTRL/H
Cursor nach rechts	Taste 'Pfeil rechts' Leertaste l

Cursor nach oben	Taste 'Pfeil hoch' k
Cursor nach unten	Taste 'Pfeil runter' j CTRL/J
Naechste Zeile	RETURN
Naechstes Wort	w
Voriges Wort	b
In Ausgangsstellung (Zeile 3, Spalte 0)	H
Cursor ans Ende (Zeile 22, Spalte 0)	L
Cursor in xy-Koordinate	gxy
Spalte 0 der aktuellen Zeile	0
Erstes Schriftzeichen der akt. Zeile	^
Letztes Schriftzeichen der akt. Zeile	$

Befehle zur Aufbereitung von Promptern

Vorgang	Taste
Eintritt Ergaenzungsmodus	a
Eintritt Einfuegemodus	i
Eintritt Ersetzungsmodus	R
Beenden a-, i-, R-, Video-Modi	ESC
Zeile transferieren	t
Zeichen loeschen	x
Zeile hier eroeffnen	O
Zeile darunter eroeffnen	o
Video normal	n
Video invers	[
Unterstreichung	]

Video invers und unterstreichen +

Befehle zum Aufbereiten von Bildmasken-Feldern

Vorgang	Taste
Feld hinzufuegen	A
Feld veraendern	M
Feld loeschen	D
Feld transferieren	T

Verschiedene Kommandos

Vorgang	Taste
Hilfe	?
Cursorstellungsanzeige ein/aus	c
Beenden PAINT-Modus	q
Erneute Darstellung des Schirms	CTRL/R
Rueckkehr zum Menue-Handler	CTRL/X

Normalerweise beginnt man Editierarbeiten mit den Befehlen zur Cursorpositionierung, um den Cursor an die gewuenschte Stelle zu bringen. Veraenderungen werden dann durch Editierkommandos vorgenommen. Geben Sie einen Befehl ein, der fuer die aktuelle Cursorstellung nicht gueltig ist, ertoent die Hupe der Tastatur. Der Arbeitsgang wird mit dem Speichern der von Ihnen vorgenommenen Veraenderungen abgeschlossen. Da beim Abspeichern eine umfangreiche Aktualisierung der Datenbank erforderlich ist, sollten Sie damit warten, bis Sie die Editierung insgesamt abgeschlossen haben. Hier unterscheidet sich WEGA-DATA von Textaufbereitungsprogrammen, wo haeufiger abgespeichert wird. In den naechsten Abschnitten werden diese Kommandos ausfuehrlich beschrieben.

4.1.1.1 Befehle fuer Cursorbewegungen

Mit diesen Kommandos koennen Sie den Cursor frei in dem editierbaren Bereich auf dem Bildschirm bewegen. In einigen Faellen haben unterschiedliche Tasten den gleichen Effekt.

Befindet sich der Cursor in einem Teil eines Bildmaskenfeldes, wird in der rechten oberen Ecke des Bildschirms, direkt unter dem Namen der bearbeiteten Bildmaske der Name

des Bildmaskenfeldes angezeigt.

Der Cursor wird durch folgende Kommandos bewegt:

Taste: 'Pfeil links' backspace h CTRL/H	Der Cursor wird um eine Spalte nach links bewegt. Man kann auch gleichzeitig mehr als eine Spalte nach links gehen, indem man vor dem Kommando eine Zahl eingibt. Diese Zahl wird als Wiederholungszaehler bezeichnet. Gibt man beispielsweise 10h ein, geht der Cursor 10 Spalten nach links. Befindet sich der Cursor in der Spalte 0, wird das Kommando ignoriert.
Taste: 'Pfeil rechts' Leertaste l	Der Cursor wird um eine Spalte nach rechts bewegt. Man kann den Cursor auch um mehr als eine Spalte bewegen, indem man vor dem Kommando eine Zahl eingibt. Wird z.B. 10l eingegeben, wird der Cursor um 10 Spalten nach rechts bewegt. Befindet sich der Cursor in Spalte 79, wird das Kommando ignoriert.
Taste: 'Pfeil hoch' k	Der Cursor wird um eine Zeile nach oben bewegt. Man kann auch um mehr als eine Zeile gleichzeitig nach oben gehen, indem man vor dem Kommando eine Zahl eingibt. Zum Beispiel wird der Cursor durch Eingabe von 10k um 10 Zeilen nach oben bewegt. Befindet sich der Cursor auf Zeile 3, wird das Kommando ignoriert.
Taste: 'Pfeil runter' j CTRL/J	Der Cursor wird in der gleichen Spalte um eine Zeile nach unten bewegt. Man kann gleichzeitig mehr als eine Zeile nach unten gehen, indem man eine Zahl vor dem Kommando eingibt. So geht der Cursor 10 Zeilen nach unten, wenn 10j eingegeben wird. Befindet sich der Cursor auf Zeile 22, wird das Kommando ignoriert.
RETURN	Der Cursor wird in die erste freie Spalte der naechsten Zeile gebracht.
w	Der Cursor wird auf das erste Zeichen des naechsten Wortes positioniert. Ein Wort ist definiert, als eine von Blanks begrenzte Folge von Zeichen.
b	Der Cursor wird auf das erste Zeichen des vorangehenden Wortes positioniert.
H	Der Cusor wird in die linke obere Ecke des editierbaren Bereiches (Zeile 3, Spalte 0) gebracht (home).
L	Der Cursor wird in die linke untere Ecke des

editierbaren Bereiches (Zeile 22, Spalte 0) gebracht.

gxy Der Cursor wird auf die angegebene Position innerhalb des editierbaren Bereiches gebracht. Nach Druecken von g geht der Cursor in die Zeilen/Spaltenanzeige , wo ein Koordinatenpaar (Koordinaten reichen von 3 bis 22 fuer Zeilen und von 0 bis 79 fuer Spalten) eingegeben werden kann. Wird nach Eingabe der Spaltennummer RETURN gedrueckt, geht der Cursor in die angegebene Position. Durch Druecken von CTRL/U wird das Kommando abgebrochen.

0 Der Cursor wird auf Spalte 0 der aktuellen Zeile positioniert.

^ Der Cursor wird auf das erste druckbare Zeichen der aktuellen Zeile positioniert.

$ Der Cursor wird auf das letzte druckbare Zeichen der aktuellen Zeile positioniert.

4.1.1.2 Kommandos zum Editieren von Promptern

Mit diesen Kommandos kann man Prompter in Bildmasken hinzufuegen, modifizieren und loeschen. Prompter sind feste Textzeichenketten, die den Grundaufbau des Bildschirms ausmachen. Sofern das Terminal es zulaesst, koennen Prompter normal, invers, unterstrichen und unterstrichen invers angezeigt werden.

Soll neuer Text auf die Bildmaske gebracht werden, muss vom Kommandomodus (command mode) zum Texteingabemodus (input mode) uebergegangen werden. Den Eingabemodus kann man in Abhaengigkeit davon, ob der Text vor oder hinter der aktuellen Cursorposition eingefuegt werden soll, auf zwei verschiedene Weisen aktivieren. Ist man erst einmal in diesem Modus, erscheinen alle druckbaren Zeichen, die eingegeben werden, auf dem Bildschirm (Zeichen, die nicht druckbar sind, werden ignoriert) und existierende Zeichen werden nach rechts gerueckt. Will man dann wieder Kommandos zur Bewegung des Cursors verwenden, muss man in den Kommandomodus zurueckkehren.

Neben Kommando- und Eingabemodus gibt es noch den Ersetzungsmodus (replace mode), unter dem die eingegebenen Zeichen existierende Zeichen ersetzen. In Zeile zwei wird auf dem Bildschirm immer ein Prompter angezeigt, dem zu entnehmen ist, ob man im Kommando-, Eingabe- oder Ersetzungsmodus ist. Will man die Verwendung von PAINT erlernen, ist es sehr wichtig, dass man die Unterschiede zwischen diesen Modi kennt.

PAINT behandelt jede Zeile auf dem Bildschirm als eine selbstaendige, unabhaengige Einheit. Das merkt man daran ganz deutlich, dass man eine neue Zeile eingefuegen kann, ohne den Eingabemodus zu benutzen und dass man im Eingabemodus nur auf der aktuellen Zeile schreiben kann, aber nicht in eine neue Zeile uebergehen kann.

Die Kommandos zum Editieren der Prompter sind folgende:

a Uebergehen in den Eingabemodus hinter der aktuellen Cursorposition. Bei der Eingabe von Zeichen werden die rechts vom Cursor bereits vorhandenen Bildmaskenfelder und Prompter nach rechts geschoben. Zeichen von Promptern, die sich in Spalte 79 befinden, werden ueber den Bildschirmrand hinausgeschoben und gehen verloren. Befindet sich jedoch ein Bildmaskenfeld in Spalte 79, koennen keine neuen Zeichen eingegeben werden (so dass das Feld nicht ueber den Rand geschoben wird und verloren geht). Mit den Tasten Backspace, Linkspfeil und CTRL/H kann man in diesem Mode eingegebene Zeichen loeschen.

i Uebergehen in den Eingabemodus auf der aktuellen Cursorposition. Ansonsten ist das Kommando gleich dem vorhergehen.

R Uebergang in den Ersetzungsmodus. Die eingegebenen druckbaren Zeichen ersetzen die bereits auf der Bildmaske befindlichen Zeichen. Bereits vorhandene Zeichen werden nicht wie im Eingabemodus nach rechts gerueckt, sondern ersetzt (ueberschrieben). Man kann den Ersetzungsmodus nicht zum Ueberschreiben von Bildmaskenfeldern verwenden. In diesem Modus kann man den Cursor mit den weiter oben beschriebenen Tasten auf der Bildmaske bewegen.

ESC Der Eingabe, Ersetzungs- oder Verschienumgsmodus, sowie die Anzeigeattribute (Video invers, Unterstreichung, Video invers unterstrichen) werden aufgebhoben in es wird in den Kommando-Modus zurueckgekehrt. War man bereits vorher im Kommandomodus, wird durch diese Taste der Kommando-Wiederholungszaehler auf 0 gesetzt. Der Wiederholungszaehler ist eine vor einem Kommando eingegebene Zahl, die dazu fuehrt, dass das Kommando genau so oft wiederholt wird, wie durch diese Zahl angegeben. Damit koennen die Kommandos Zeichen loeschen, Cursor nach links, Cursor nach rechts, Cursor nach oben und Cursor nach unten wiederholt werden.

x Das Zeichen auf der aktuellen Cursorposition wird geloescht. Rechts vom geloeschten Zeichen stehende Zeichen werden nach links verschoben. Befindet sich der Cursor in einem Bildmasken-Feld, ist das Kommando ungueltig. Durch Verwendung des Wiederholungszaehlers kann gleichzeitig mehr als ein Zeichen geloescht werden. Wird beispielsweise 5x eingegeben, werden begin-

nend mit dem durch den Cursor markierten Zeichen nach rechts 5 Zeichen geloescht.

O Eroeffnet eine Leerzeile direkt auf der aktuellen Zeile und rueckt den Rest der Zeilen auf dem Bildschirm nach unten. Die unterste Zeile wird ueber den Bildschirmrand hinausgeschoben und PAINT bleibt im Kommandomodus. Befindet sich ein Bildmasken-Feld in der untersten Zeile, kann man mit diesem Kommando keine Zeile einfuegen. Dann muss zuerst das Bildmasken-Feld von der untersten Zeile geloescht werden.

o Eroeffnet direkt unter der aktuellen Zeile eine Leerzeile. Ansonsten ist dieses Kommando identisch mit Kommando O.

d Die aktuelle Zeile wird geloescht und die darunter stehenden Zeilen um eine Zeile nach oben verschoben. Enthaelt die aktuelle Zeile Bildmasken-Felder, muessen diese erst mit dem Kommando zum Loeschen von Bildmasken-Feldern (Abschnitt 4.1.1.3) geloescht werden.

t Die aktuelle Zeile wird auf eine andere Position verschoben. Die Zeile darf sowohl Prompter als auch Bildmaskenfelder enthalten. Nachdem Eingeben dieses Kommandos wird der Cursor in die gewuenschte Position gebracht und dann wird erneut die "Kommando"-Taste (t) gedrueckt. Damit wird die Zeile in ihrer vorherigen Position geloescht und wird in die neue Position gebracht. Um den Cursor zu bewegen, kann ein beliebige Taste fuer Kommandos zur Cursorbewegung verwendet werden (z.B. Linkspfeil-Taste). Mit ESC kann dieses Kommando jederzeit abgebrochen werden.

Mit den folgenden vier Kommandos kann man verschiedene Anzeigeattribute fuer die Zeile angeben, in der sich der Cursor befindet. Alle funktionieren folgendermassen: Zuerst wird der Cursor an das eine Ende des Bereiches gebracht, fuer den das Anzeigeattribut geaendert werden soll. Dann wird das gewunschte Kommandozeichen eingegeben. Dadurch erscheint ein "Markierungszeichen". Dann wird der Cursor ans andere Ende des Bereiches gebracht, was links oder rechts vom Markierungszeichen sein kann. Man kann ein beliebiges Kommando zur Cusorbewegung verwenden, solange dadurch nicht die Zeile verlassen wird. Dann wird dasselbe Kommandozeichen nocheinmal eingegeben. Der zwischen dem Markierungszeichen und dem Cursor liegende Bereich (einschliesslich Cursorstellung) erscheint im gewuenschten Anzeigemodus. Mit ESC kann man jederzeit diese Kommandos abbrechen. Es ist zu beachten, dass die entsprechenden Eintreaege in der Datei termcap definiert sein muessen (siehe Abschnitt 1.1.4).

n Das Kommando Video normal fuehrt dazu, dass der Anzeigemodus im markierten Bereich in normaler Intensitaet erfolgt und helle Zeichen auf dunklem Hintergrund.

[Das Kommando Video invers fuehrt dazu, dass der Anzeigemodus im markierten Bereich Negativanzeige ist, d.h. dunkle Zeichen auf hellem Hintergrund

] Alle im markierten Bereich befindlichen Zeichen werden unterstrichen.

+ Das Kommando Video invers unterstrichen fuehrt dazu, dass alle im markierten Bereich befindlichen Zeichen in unterstrichener Negativanzeige erscheinen.

Einige Terminaltypen, wie auch das P8000-Terminal, erfordern fuer Ein- und Ausschalten der Video-Attribute eine Zeichenposition. Diese "Attributzeichen", die auf dem Bildschirm wie normale Leerzeichen aussehen, muessen anders behandelt werden, als normale Leerzeichen. PAINT gestattet nicht, dass sie ueber den Rand des Bildschirms hinaus geschoben, geloescht oder ersetzt werden, da sich damit die Anzeigeattribute des Bereiches veraendern wuerden. Stattdessen werden sie geloescht, indem die Anzeigeattribute auf "normal" gesetzt werden.

Diese Terminals koennen auch nicht zwei Negativanzeigebereiche oder zwei unterstrichene Bereiche anzeigen, die nur durch ein einziges "Normalanzeige"-Zeichen getrennt sind. Zwischen Bereichen desselben Typs muessen mindestens zwei Leerzeichen stehen. Sollen also Bildmasken an dieser Art Terminals verendet werden, muss man zwischen den mit verschiedenen Anzeigeattributen dargestellten Bereichen genuegend freien Platz lassen.

4.1.1.3 Kommandos zum Editieren von Bildmasken-Feldern

Mit diesen Kommandos kann man in einer Bildmaske Bildmaskenfelder hinzufuegen, modifizieren und loeschen. Bildmaskenfelder sind die "Fenster", durch die man in die Datenbasis schauen und Datenwerte fuer den aktuellen Datensatz sehen kann. Sie bestehen aus dem Namen des Bildmaskenfeldes, einem Datenbankfeld, einem Typ, einer Laenge und den Koordinaten. Man gibt die ersten zwei Datenelemente an und PAINT stellt die restlichen bereit.

Sollen Bildmaskenfelder hinzugefuegt oder geaendert werden, muss man vom Kommandomodus in den Modus zum Editieren von Bildmaskenfeldern uebergeben. Befindet man sich in diesem Modus, werden die letzten zwei Zeilen auf dem Bildschirm durch Prompter ersetzt, in die man die erforderlichen Informationen eingeben kann. Bei erneutem Eintritt in den Kommandomodus wird die Bildmaske wieder komplett angezeigt.

Bildmasken-Felder duerfen sich nicht gegenseitig und auch nicht mit Prompter ueberschneiden. Wuerden beim Hinzufuegen oder Verschieben von Feldern Ueberlappungen entstehen, wuerde das Hinzufuegen oder Verschieben verhindert. Das Anzeigeformat sieht fuer die verschiedenen Typen von Bild-

maskenfeldern folgendermassen aus:

NUMERIC	NNNNNN
AMOUNT	AAAA.AA
FLOAT	FFFFFF.FFF
DATE	MM/DD/YY
TIME	HH:MM
STRING	SSSSSS

Die Anzahl der fuer das Feld angezeigten Zeichen entspricht der Laenge des Datenbank-Feldes. Ist in der Datei termcap die Unterstreichung der Anzeige definiert, werden die Bildmaskenfelder unterstrichen. Bei Terminals, die den termcap-Eintrag ug#1 erfordern, werden Bildmaskenfelder nicht unterstrichen.

Die Kommandos zum Editieren von Bildmaskenfeldern sind folgende:

A Damit geht man vom Kommandomodus in den Modus zum Editieren von Bildmaskenfeldern ueber. Soll ein neues Bildmaskenfeld hinzugefuegt werden, muss sich der Cursor ueber einem freien Bereich befinden. Die letzten zwei Zeilen auf dem Bildschirm werden geloescht und die unter dem naechsten Kommando erklaerten Prompter werden angezeigt.

M Damit geht man vom Kommandomodus in den Modus zum Editieren von Bildmaskenfeldern ueber. Der Cursor muss sich in einem existierenden Bildmaskenfeld befinden. Die letzten zwei Zeilen auf dem Bildschirm werden geloescht und es erscheinen folgende Prompter:

```
SCREEN FIELD: _              TYPE:              LENGTH:
DATA BASE FIELD:
```

Die Prompter werden folgendermassen verwendet:

SCREEN FIELD
Nur einmal vorkommender Bildmaskenfeldname bestehend aus 8 Schriftzeichen. Dieser Name wird verwendet, wenn von Ihnen geschriebene Programme auf das Bildmaskenfeld Bezug nehmen. Daher darf kein anderes Datenbankfeld den gleichen Namen haben. Man kann jedoch bei verschiedenen Bildmasken die gleichen Bildmaskenfeldnamen verwenden. Die Anzahl der Bildmaskenfelder haengt von der verfuegbaren Speicherkapazitaet ab.

DATA BASE FIELD
Name des Datenbankfeldes zur Anzeige auf diesem Bildmaskenfeld. PAINT sucht den Namen im Datenwoerterbuch und ueberprueft ihn auf Gueltigkeit. Felder vom COMB-Typ koennen nicht direkt benutzt werden. Die Komponenten des Feldes sind getrennt einzugeben. Wird das Da-

tenbankfeld frei gelassen, muessen Sie die naechsten zwei Prompter (TYPE und LENGTH) ausfuellen. Der Datenbankfeldname wird durch Druecken der Leertaste geloescht.

TYPE
Der Typ des Datenbankfeldes (NUMERIC, STRING, DATE, TIME, AMOUNT oder FLOAT) wird von PAINT aus dem Datenwoerterbuch uebernommen, wenn ein Datenbankfeldname festgelegt wurde. Sie koennen daran keine Veraenderungen vornehmen.

LENGTH
Die auf dem Schirm angezeigte Laenge des Datenbankfeldes. Wird von PAINT aus dem Datenwoerterbuch uebernommen, wenn ein Datenbankfeldname angegeben wurde. Sie koennen daran keine Veraenderungen vornehmen.

Die Taste RETURN kann verwendet werden, um einen Prompter zu ueberspringen. Durch CTRL/U kann man zurueckgehen. Wird CTRL/U gedrueckt, waehrend man im Prompter DATA BASE FIELD ist, kehren Sie in den Befehlsmodus zurueck, die zwei letzten Zeilen erscheinen wieder auf dem Schirm und der Schirm wird durch Anzeige aller vorgenommenen Veraenderungen aktualisiert. Wurde kein Datenbankfeld festgelegt (Ergaenzungs- und Aenderungsmodus), wird der Cursor bei Druecken der RETURN-Taste auf die Prompter TYPE und LENGTH eingestellt. Diese sind auszufuellen.

D Das aktuelle Bildmaskenfeld wird geloescht. Das aktuelle Bildmaskenfeld ist das Feld, dessen Name in der rechten oberen Ecke des Bildschirms angezeigt wird. Es ist deshalb das aktuelle Feld, weil sich der Cursor in diesem Feldes befindet.

T Das aktuelle Bildmaskenfeld wird auf eine andere Position gebracht. Nach Aufruf des Kommandos stellt man den Cursor in die gewuenschte Position und drueckt dann erneut die "Kommando"-Taste (T). Zur Bewegung des Cursors kann dabei eins der Kommandos fuer die Cursorbewegung verwendet werden. Passt das Feld nicht an die neue Position, erhaelt man die Moeglichkeit, den Cursor woandershin zu bringen. Mit ESC kann dieses Kommando jederzeit abgebrochen werden.

4.1.1.4 Verschiedene Kommandos

Die in diesem Abschnitt angefuehrten Kommandos dienen der Ausfuehrung bestimmter Funktionen, die sich nicht direkt auf die Erstellung und Editierung von Bildmasken beziehen. Diese Kommandos sind folgende:

? Das Kommando help zeigt eine Zusammenfassung der zur Verfuegung stehenden Kommandos an. Es gibt zwei Bildschirme voll mit Hilfsinformatioenen. Auf dem

ersten befinden sich die Kommandos zur Editierung von Promptern, zur Editierung von Bildmaskenfeldern und verschiedene andere Kommandos. Auf dem zweiten befinden sich die Kommandos zur Bewegung des Cursors. Will man den zweiten Bildschirm sehen, drueckt man einfach RETURN. Wird hinter dem unten auf dem Schirm erscheinenden Prompter stattdessen n eingegeben, erscheint der zweite Bildschirm nicht. Sieht man den zweiten "Teil" der Hilfsinformation, kehrt man mittels RETURN in die Bildmaske zurueck, an deren Editierung man gerade gearbeitet hat.

c Mit diesem Kommando kann man die in Zeile 2 erfolgende Zeilen- bzw. Spaltenanzeige fuer die Cursorposition ein- bzw. ausschalten.

CTRL/R Der editierbare Bereich des Bildschirm wird geloescht und die Bildmaske entsprechend den vorhandenen Informationen wieder aufgebaut. Dieses Kommando wird verwendet, wenn der Aufbau des Schirm durch Meldungen oder dergleichen zerstoert wurde.

q Dieses Kommando wird im Kommandomodus verwendet, um die Editierung der aktuellen Bildmaske abzuschliessen. Bei Aufruf dieses Kommandos erscheint folgendes Prompter:

[S]ave, [D]on't save, [R]esume editing

Bei Eingabe von s, werden die von Ihnen vorgenommenen Veraenderungen im Datenwoerterbuch gespeichert und die Bildmaske wird so verarbeitet, dass aktualisierte .q- und .h-Dateien entstehen. Bei Eingabe von d werden die von Ihnen vorgenommenen Veraenderungen nicht gespeichert, so dass die Bildschirmdarstellung die gleiche wie vor Beginn des Editierens ist. Bei Eingabe von r (oder CTRL/U) koennen Sie das Editieren der Bildmaske fortsetzen.

Aufgrund der von PAINT zur Zerlegung der Bildmaske in Elemente des Datenwoerterbuchs verwendeten Algorithmen, koennen kleine Unterschiede zwischen den von 'Screen Entry' und 'Process Screen' erzeugten .q- und .h-Dateien auftreten (Abschnitt 4.2 und 4.4). Es ist also am besten, wenn man fuer die Erstellung und Modifizierung von Bildmasken entweder nur PAINT oder nur 'Screen Entry' verwendet.

CTRL/X Dieses Kommando (Exit) kann jederzeit verwendet werden, wenn man unmittelbar in den Menue-Handler zurueckkehren moechte, ohne erst die normale Folge der Prompter zu durchlaufen. Alle seit dem letzten Speichern vorgenommenen Aenderungen gehen verloren.

4.1.2 Angepasste PAINT-Kommandos

In der Standardversion von PAINT werden alle Kommandos durch Druecken der in den vorhergehenden Abschnitten beschriebenen Tasten aufgerufen. Mit dem Tastendruck wird ein Zeichen gesendet, das PAINT als einem bestimmten Kommando zugehoerig erkennt. Man kann jedes Kommando so aendern, dass es von einem anderen Zeichen aufgerufen wird, indem man die Textdatei editiert, auf die sich PAINT zur Erlangung dieser Informationen bezieht. Die Datei heisst unicap und in ihr sind die Beziehungen zwischen den PAINT-Kommandos und den Zeichen, die diese Kommandos aufrufen, enthalten. Standardmaessig befindet sich unicap im Verzeichnis $WDATA. Unter Verwendung der Umgebungsvariablen UNICAP kann man diese Position aendern (siehe Abschnitt 1.1.3).

In Abschnitt 4.1.2.1 werden die auf die individuellen Beduerfnisse anpassbaren Kommandos aufgefuehrt, in Abschnitt 4.1.2.2 werden die Imaginaer-Zeichen eingefuehrt, die PAINT versteht (im Gegensatz zu den "realen" Zeichenfolgen, die tatsaechlich vom Terminal ausgesendet werden, wenn eine Taste gedrueckt wird). In Abschnitt 4.1.2.3 wird beschrieben, wie PAINT die Datei unicap verwendet.

4.1.2.1 Namen von PAINT-Kommandos

Alle PAINT-Kommandos koennen selbst gewaehlten Kommandozeichen zugeordnet werden. Ein Beispiel, wie Kommandos und Zeichen einander zugeordnet werden, gibt die folgende Zeile in der Datei unicap, durch die das Zeichen festgelegt wird, mit dem das Einfuegen auf Cursorposition aufgerufen wird:

```
ins = i
```

Der interne Name des Kommandos steht links - ins. Der Name des Zeichens, das das Kommando aufruft, steht rechts - i. Moechte man aber, dass dieses Kommando ausgefuehrt wird, wenn die Taste e gedrueckt wird, braucht man nur e anstelle von i einzusetzen.

Es folgt die Liste der internen Kommandonamen, die von PAINT in der Datei unicap verwendet werden:

PAINT-Kommandos

Interner Name	Kommando	Standardzeichen
addfld	Add screen field	A
app	Append input mode	a
bot	Last line	L
col0	Column 0	0
cpdisp	Toggle cursor position display	c
delch	Delete character	x
delfld	Delete screen field	D
delln	Delete line	d

down	Cursor down	j
endln	End of line	$
esc	Escape	<ESC>
exit	Exit to the menuhandler	CTRL/X
goto	Go to x,y	g
help	Help	?
home	Home cursor	H
ins	Insert input mode	i
left	Cursor left	h
modfld	Modify screen field	M
nrmlon	Normal video	n
nxtln	Next Line	RETURN
nxtw	Next word	w
oplna	Open line at	O
oplnb	Open line below	o
prvw	Previous word	b
quit	Quit PAINT mode	q
redraw	Redraw screen	CTRL/R
rep	Replace mode	R
right	Cursor right	l
rvon	Reverse video	[
rvulon	Reverse underline video	+
stln	Beginning of line	^
trnfld	Transfer screen field	T
trnld	Transfer line	t
ulon	Underline video	]
up	Cursor up	k

4.1.2.2 Imaginaer-Zeichen

Verschiedene Terminals senden verschiedene Folgen von Oktalkodes, wenn ihre Tasten gedrueckt werden. Der ASCII-Zeichensatz ist standardisiert, so sendet z.B. jedes Terminal beim Druecken von 'A' den Oktalcode '101'. Fuer andere Tasten wie z.B. Rechtspfeil, Linkspfeil, Home Cursor, F1 (Funktionstaste 1) usw. gibt es keine Standards. Moechte man also einige dieser Tasten an verschiedenen Terminals verwenden, muss PAINT davon in Kenntnis gesetzt werden, wie es erkennen kann, welche nichtstandardisierte Taste gedrueckt wurde.

Das geschieht, indem man die von den "realen" Tasten jedes Terminals ausgesendeten Zeichenfolgen in eine standardisierte interne Form uebersetzt, die PAINT erkennen kann. Die standardisierte interne Form einer Zeichenfolge wird als imaginaeres Zeichen, imaginary character, bezeichnet, da diese Form nicht wirklich existiert. Fuer jede nichtstandardisierte "reale" Zeichenfolge gibt es ein imaginaeres Zeichen, das PAINT erkennen kann. PAINT betrachtet jedes imaginaere Zeichen als einen einzelnen, einmaligen Oktalcode.

PAINT hat jedem Imaginaer-Zeichen einen Namen zugeordnet, der in der Datei unicap verzeichnet ist. In unicap gibt es eine Sektion fuer jeden Terminaltyp (in Uebereinstimmung

mit den termcap Eintraegen fuer Terminals, siehe Abschnitt 1.1.4), die PAINT mitteilt, welche Zeichenfolge zu erwarten ist, wenn die jeweiligen Tasten gedrueckt werden und welches Imaginaer-Zeichen welcher Folge zugeordnet ist. Die Folge kann auch aus einem einzelnen Zeichen bestehen. In Abschnitt 4.1.2.3 wird erlaeutert, wie Kommandos und Imaginaer-Zeichen einander zugeordnet sind.

Hier Name und Beschreibung der imaginaeren Zeichen:

Name	Beschreibung
del_char	Dieses Zeichen wird der von der Taste 'delete character' erzeugten Folge zugeordnet.
del_line	Dieses Zeichen wird der von der Taste 'delete line' erzeugten Folge zugeordnet.
down_arrow	Dieses Zeichen wird wird der Folge zugeordnet, die durch die Taste mit Pfeil nach unten erzeugt wird.
f0 f1 . . . f20	Diese Zeichen werden den von den Funktionstasten erzeugten Folgen zugeordnet. Es gibt einundzwanzig imaginaere Funktionstasten-Zeichen mit Namen von f0 bis f20.
help	Dieses Zeichen wird der Folge zugeordnet, die durch die Taste mit der Aufschrift HELP erzeugt wird. (Nur wenige Terminals haben eine solche Taste.)
home	Dieses Zeichen wird der Folge zugeordnet, die durch die Taste mit der Aufschrift HOME oder HOME CURSOR erzeugt wird.
ins_char	Dieses Zeichen wird der Folge zugeordnet, die durch die Taste 'insert charakter' erzeugt wird.
ins_line	Dieses Zeichen wird der Folge zugeordnet, die durch die Taste 'insert line' erzeugt wird.
left_arrow	Dieses Zeichen wird der Folge zugeordnet, die durch die Taste mit Linkspfeil erzeugt wird.
next_pg	Dieses Zeichen wird der Folge zugeordnet, die durch die Taste 'next page' erzeugt wird.
prev_pg	Dieses Zeichen wird der Folge zugeordnet, die durch die Taste 'previous page' erzeugt wird.

right_arrow Dieses Zeichen wird der Folge zugeordnet, die durch die Taste mit Rechtspfeil erzeugt wird.

up_arrow Dieses Zeichen wird der Folge zugeordnet, die durch die Taste mit Pfeil nach oben erzeugt wird.

4.1.2.3 Zuordnen von Zeichen zu Kommandos

In den vorhergehenden drei Abschnitten wurde die Datei unicap, der PAINT-Kommandosatz und Imaginaer-Zeichen vorgestellt. In diesem Abschnitt soll das Format der Datei unicap erklaert werden und wie die Zeichen den Kommandos zugeordnet werden.

Grundelement der Datei unicap ist eine Zeile. Zeilen haben folgendes Format:

```
scluessel_wort = zeichenketten_liste kommentar
```

Gueltige schluessel_worte sind die von PAINT anerkannten. Die zeichenketten_liste kann eine der vielen unten beschriebenen Formen annehmen. Die kommentare sind optionell. Die Kommandonamen und die Namen der Imaginaer-Zeichen, die in den vorhergehenden zwei Abschnitten beschrieben wurden, sind alles Schluesselworte. Die uebrigen Schluesselworte und ihre Bedeutung werden in diesem Abschnitt beschrieben.

Eine zeichenketten_liste besteht entweder aus einer einzelnen Zeichenkette oder aus einer Liste von Zeichenketten, die durch senkrechte Striche (|) voneinander getrennt sind. Einige Schluesselworte lassen nur eine einzelne Zeichenkette zu, was in der Beschreibung des Schluesselwortes vermerkt ist. Jede Zeichenkette besteht aus Zeichen aus einer oder mehreren der fuenf Zeichenklassen - druckbare Zeichen, Escape-Zeichen, Steuerzeichen, Oktalzeichen und imaginaere Zeichen. Eine Zeichenkette darf ausser den unten erklaerten keine Leerzeichen oder Tabs enthalten. Dadurch koennen Leerzeichen und Tabs genutzt werden, um die Lesbarkeit der zeichenketten_liste zu verbessern.

Die druckbaren Zeichen sind alle druckbaren ASCII-Zeichen (Gross- und Kleinbuchstaben, Ziffern und Sonderzeichen) mit Ausnahme von Leerzeichen, "Dach" (^), Backslash (\) und "kleiner als" (<). Diese Zeichen werden durch ihre normale gedruckte Form dargestellt. Dach, Rueckstrich und kleiner als haben eine besondere Bedeutung, wenn sie durch ihre normale Druckform dargestellt werden (wie spaeter gezeigt werden soll), somit muessen sie als Escape-Zeichen dargestellt werden. Mit den Escape-Zeichen kann man einige der nichtdruckbaren Sonderzeichen - Dach, Rueckstrich und kleiner als darstellen. Escape-Zeichen werden folgendermassen dargestellt:

Escape-Zeichen

Darstellung	Bedeutung
\E	Escape
\n	New line
\r	Return
\t	Tab
\b	Backspace
\f	Form feed
\^	Dach
\\	Backslash
\<	kleiner als

Mit den Steuerzeichen kann man alle moeglichen Funktionstasten darstellen. Diese werden folgendermassen dargestellt:

Steuerzeichen

Darstellung	Bedeutung
^X	CTRL X, wobei X ein beliebi- Buchstabe von A-Z ist
^[	CTRL [
^]	CTRL]
^\	CTRL \
^^	CTRL ^
^_	CTRL _

Mit den Oktalzeichen kann man nichtdruckbare Zeichen darstellen, die nicht auf andere Weise darstellbar sind, oder druckbare Zeichen, die eine Sonderbedeutung haben, wenn sie als solche dargestellt werden. Oktalzeichen bestehen aus einem Backslash und drei darauf folgenden Oktalziffern (Oktalziffern reichen von 0 bis 7). \000 ist kein gueltiges Oktalzeichen. Das Leerzeichen muss als ein Oktalzeichen dargestellt werden, da "normale" Leerzeichen in der Datei unicap ignoriert werden. Es wird oktal als \040 dargestellt.

Imaginaer-Zeichen wurden in Abschnitt 4.1.2.2 beschrieben. Kurz zur Wiederholung: Mit Imaginaer-Zeichen kann man die nichtstandardisierten Zeichenfolgen darstellen, die ausgesendet werden, wenn Terminaltasten wie "Rechtspfeil" gedrueckt werden. Die Imaginaer-Zeichen werden durch ihre Namen dargestellt.

Der im folgenden gezeigte Teil der Standard unicap-Datei, der zur Verwendung mit PAINT vorgesehen ist, soll diese Beziehungen der Begriffe untereinander demonstrieren.

```
SECTION = system   general system characteristics
ndelaypad = 1        # pad chars to ignore in no delay mode

SECTION = vt100         vt100
left_arrow = \E[A           left arrow
right_arrow = \E[B          right arrow
up_arrow = \E[C             up arrow
down_arrow = \E[D           down arrow

SECTION = 912           televideo 912
left_arrow = \b             left arrow
right_arrow = \^L           right arrow
up_arrow = ^K               up arrow
down_arrow = ^J             down arrow

SECTION = paint                 paint info
help = ?                              help command
left   = < left_arrow > | \b | h      cursor left
right  = < right_arrow > | l | \040   cursor right
up     = < up_arrow > | k             cursor up
down   = < down_arrow > | j | ^J      cursor down
escape = \E                           escape
stln   = \^                           start of line
nxtln  = \n | \r                      next line
home   = H                            cursor home
```

Die Zeilen sind in Sektionen gegliedert, wobei jede dieser Sektionen eine Gruppe verwandter Charakteristika beschreibt. Eine Sektion beginnt mit der Zeile SECTION = schluessel_wort. Sektionen koennen in beliebiger Reihenfolge auftreten. Hier die Spezifikationen, fuer die fuer PAINT erforderlichen unicap-Sektionen: Worte im Fettdruck sind aktuelle Schluesselworte, waehrend unterstrichene Worte formale Schluesselworte darstellen, bei denen man ein aktuelles Schluesselwort aus der entsprechenden Klasse einsetzen muss.

SECTION = system

Mit dieser Sektion werden Gesamt-Systemcharakteristika beschrieben. Im Moment gibt es nur dieses eine gueltige Schluesselwort fuer diese Sektion.

ndelaypad = n

Dieses Schluesselwort beschreibt die relative Geschwindigkeit zwischen der Baud-Rate und dem verwendeten Prozessor. n ist eine positive ganze Zahl. Fuer schnelle Baud-Raten (im Verhaeltnis zur Prozessor-Geschwindigkeit) muss diese klein sein und fuer langsame Baudraten ist sie gross. Diese Zahl gibt an, wieviele "no delay"-Lesungen auszufuehren sind, um jedes Zeichen vom Terminal zu erhalten. Wenn n korrekt gesetzt ist, erkennt PAINT die von Funktionstasten, Pfeiltasten usw. kommenden Zeichenfolgen so, als waeren sie von einem einzigen Tastendruck gekommen. Wenn die Zahl n zu klein ist,

denkt PAINT, dass der Nutzer die Zeichen einzeln eingegeben hat und kann daher diese Imaginaer-Zeichen nicht richtig erkennen. Bei 9600 Baud ist n normalerweise 0 und bei 110 Baud ist n normalerweise etwa 4.

SECTION = terminal_typ

Es kann so viele Sektionen terminal_typ geben, wie gewuenscht werden und zwar fuer jeden Typ eine. Diese Sektion besteht aus Zeilen, die die von den "nichtstandardisierten" Tasten ausgesendeten Zeichenfolgen den Imaginaer-Zeichen zuordnen. Will man keine dieser Tasten verwenden, kann die Sektion ausgelassen werden. Der aktuelle terminal_typ muss dem Wert der Umgebungsvariablen TERM entsprechen (siehe Abschnitt 1.1.3). Ist zum Beispiel in der aktuellen Umgebung TERM=PV, muss man eine SECTION = PV haben, die fuer die VT100-Terminals die Zuordnung von realen zu Imaginaer-Zeichen beschreibt.

imaginaerzeichen_name = zeichenkette

Die Namen der Imaginaer-Zeichen werden aus den in Abschnitt 4.1.2.2 aufgelisteten ausgewaehlt. Jede zeichenkette ist eine Darstellung einer Zeichenfolge, die von der Taste auf dem Terminal ausgesendet wird, die diesem Zeichennamen gleichgesetzt werden soll. Jedem Imaginaer-Zeichen kann nur eine Folge zugeordnet werden.

SECTION = paint

In dieser Sektion ist fuer jedes in Abschnitt 4.1.2.1 beschriebene Kommando die Zuordnung zwischen Zeichen und Kommandos enthalten, ebenso wie verschiedene fuer ganz PAINT gueltige Parameter.

PAINT_kommando = zeichenketten_liste

Die PAINT_kommandos werden aus den in Abschnitt 4.1.2.1 aufgelisteten Kommandos gewaehlt. Die zugeordnete zeichenketten_liste definiert das Zeichen, das das Kommando aufruft. Wird ein bestimmtes Kommando nicht definiert, kann man die entsprechende Funktion nicht ausfuehren. Die Namen von Imaginaer-Zeichen, die als Teile einer zeichenketten_liste erscheinen, muessen in winklige Klammern < und > eingeschlossen werden. Bei der Zuordnung von Kommandos und Zeichen gibt es einige Einschraenkungen.

1. Ein Kommando kann von mehr als einem Zeichen aufgerufen werden, aber ein Zeichen kann nur einem einzigen Kommando zugeordnet werden.

2. Da Ziffern fuer die Angabe der Kommandowiederholungen, (Wiederholungszaehler), verwendet werden, koennen sie nicht zum Aufruf von Kommandos verwendet werden. Eine Ausnahme bildet die Ziffer 0,

die nur dem Kommando 'column 0' zugeordnet werden kann.

3. Dem Kommando escape koennen nur nichtdruckbare Zeichen zugeordnet werden. Wird dieses Kommando einem druckbaren Zeichen zugeordnet, kann man dieses Zeichen nicht in eine Bildmaske eintragen, da seine Eingabe zum Beenden des Eingabemodus' fuehrt.

ignoremiss = TRUE
Ohne diese Zeile gibt PAINT fuer jedes Kommando, das keiner Taste zugeordnet ist, eine Meldung aus. Ist diese Zeile vorhanden, werden die Meldungen unterdrueckt. Kommandos, die keiner Taste zugeordnet sind, koennen nicht ausgefuehrt werden; eine Ausnahme bilden die Kommandos escape und quit, die standardmaessig ESC und q sind.

mrkchar = zeichen
Mit diesem Schluesselwort kann man das Standardzeichen 'marker character' aendern, das zur Markierung der ursprunglichen Cursorstellung bei verschiedenen Kommandos (add field, transfer field, transfer line, reverse video usw.) verwendet wird. zeichen ist ein einzelnes druckbares Zeichen.

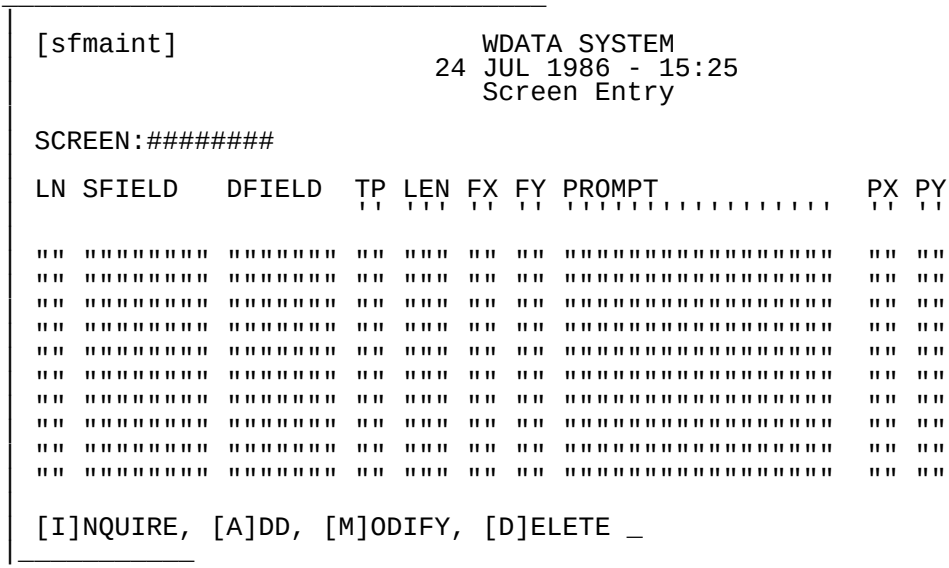
Es gibt ein Kommando, mit dem man die Syntax der Datei unicap pruefen kann, das vom Shell aus aufgerufen werden kann. Das Kommando hat folgende Form:

```
chkunicap [-p] datei_name
```

Der datei_name ist der Name der verwendeten Datei unicap. Das optionelle -p druckt nach den Meldungen die Datei. Es meldet Fehler im Format und in der Darstellung der Zeichen, prueft jedoch nicht die Semantik. Diese wird von PAINT geprueft und gemeldet, bevor PAINT startet. Siehe oben, Schluesselwort ignoremiss.

4.2 Eingabe von Bildmasken - 'Screen Entry'

In diesem Abschnitt wird beschrieben, wie Bildmasken aufgerufen, geaendert oder geloescht werden koennen. Das Programm zeigt gleichzeitig eine ganze Seite von Bildmaskenfeldern an. Man kann zur naechsten oder vorhergehenden Seite uebergehen, bestimmte Bildmaskenfelder zur Modifizierung oder zum Loeschen auswaehlen, neue Bildmaskenfelder hinzufuegen oder ganze Bildmasken loeschen. Das Programm wird gestartet, indem man in 'SFORM Menu' die 2, 'Screen Entry', waehlt:




```
########  Eingabebereich fuer Bildmasken
''''''''  Eingabebereich fuer Bildmasken-Felder
"""""""""  Darstellungsbereich fuer Bildmasken-Felder
```

Mit dieser Bildmaske kann man Bildmasken und Bildmaskenfelder abfragen, hinzufuegen, modifizieren und loeschen. Es gibt einen Dateneingabebereich fuer Bildmasken, einen Dateneingabebereich fuer Bildmaskenfelder und einen Darstellungsbereich fuer Bildmaskenfelder. Im folgenden sollen die auf dem Bildschirm erscheinenden Prompter und die Spaltenueberschriften erklaert werden.

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE:

Mit diesem Prompter kann man den Betriebsmodus fuer das Programm waehlen. Die verschiedenen Modi haben folgende Bedeutung:

i - Abfragemodus, ermoeglicht die Betrachtung vorhandener Bildmaskenfelder einer existierenden Bildmaske.

a - Ergaenzungsmodus, ermoeglicht die Erstellung neuer Bildmasken mit ihren Bildmaskenfeldern.

m - Aenderungsmodus, ermoeglicht Hinzufuegen, Aendern und Loeschen von Feldern in einer existierenden Bildmaske.

d - Loeschmodus, ermoeglicht das Loeschen vollstaendiger Bildmasken (mit allen ihren Bildmaskenfeldern). Durch das Loeschen werden auch alle Referenzen in Menues und bei Registrierungen (ENTER, SQL) dieser Bildmasken

geloescht.

SCREEN
Ein aus 8 Zeichen bestehender Name der Bildmaske. Der Name wird im gesamten System zur Bezugnahme auf die Bildmaske verwendet.

[N]ext page, [P]prev page, [A]dd line oder number:

Prompter fuer Seitenaufrufbereich; wird angezeigt nachdem der Prompter 'SCREEN' beantwortet wurde. Mit diesem Prompter kann man den Modus fuer den Seitenbereich waehlen. Die einzelnen Auswahlmoeglichkeiten haben folgende Bedeutung:

n - Die naechste Seite der Bildmaskenfelder wird angezeigt.

p - Die vorhergehende Seite der Bildmaskenfeleder wird angezeigt.

a - Gestattet das Hinzufuegen neuer Bildmaskenfelder, indem der Cursor auf den Eingabebereich fuer Felder gesetzt wird.

1-99 - Es wird die Seite angezeigt, die das angegebene Bildmaskenfeld enthaelt und der Cursor wird auf dieses Feld gesetzt, so dass geaendert oder geloescht werden kann, wie im folgenden beschrieben.

LN
Ist eine vom System zugewiesene Zeilennummer. Wird zur Bezugnahme auf das Bildmaskenfeld verwendet, falls dieses geaendert werden soll.

Spalte zwischen LN und SFIELD
Dieser Bereich wird zur Eingabe eines Befehls verwendet, durch den eine Operation am aktuellen Bildmaskenfeld recht ausgefuehrt wird. Stimmt mit der mit CMD ueberschriebenen Spalte auf dem "Schema Entry Screen" ueberein. Die Zeile ist jedoch so voll, dass fuer die Bezeichnung dieser Spalte kein Platz mehr war. Durch Druecken von CTRL/U und RETURN kann der Cursor in dieser Spalte auf und abbewegt werden. Die Cursorstellung kennzeichnet die aktuelle Zeile. Die gueltigen Befehle sind:

m - Aendern von DFIELD, TP, LEN, FX, FY, PROMT, PX oder PY fuer das aktuelle Bildmaskenfewld.

d - Loeschen des aktuellen Felds.

q - Der Promter fuer den Seitenaufruf erscheint erneut unten auf dem Schirm.

SFIELD
Ein einmalig auftretender aus 8 Zeichen bestehender Bildmaskenfeldname. Dieser Name wird verwendet, um in

Programmen, die vom Nutzer geschrieben werden, auf das Bildmaskenfeld Bezug zu nehmen. Daher sollte der Name nicht mit Datenbank-Feldnamen uebereinstimmen. Verschiedene Bildmasken koennen jedoch die gleichen Bildmaskenfeldnamen verwenden. Man kann so viele Bildmaskenfelder haben, wie Puffer und Bildschirm zulassen.

DFIELD
Name des Datenbank-Feldes, der fuer dieses Bildmaskenfeld angezeigt wird. Wird diese Spalte freigelassen, muessen die naechsten beiden Spalten (TP und LEN) ausgefuellt werden. SFORM sucht den Namen im Datenwoerterbuch, um ihn auf seine Gueltigkeit zu ueberpruefen. Felder vom Typ COMB koennen nicht direkt mit SFORM benutzt werden; die Komponenten des Feldes sind getrennt einzugeben.

TP
Der Typ des Datenbank-Feldes. Wird von SFORM durch Informationen aus dem Datenwoerterbuch ausgefuellt, wenn der Name eines Datenbank-Feldes angegeben wurde.

LEN
Die Anzeigelaenge des Datenbank-Feldes. Wird ebenfalls von SFORM durch Informationen aus dem Datenwoerterbuch ausgefuellt, wenn der Name eines Datenbank-Feldes angegeben wurde.

FX
Die x-Koordinate (Spalte), in der das Datenbank-Feld auf der Bildmaske erscheint. Es muss eine Zahl zwischen 0 und 79 sein.

FY
Die y-Koordinate (Zeile) des Datenbank-Feldes. Es muss eine Zahl zwischen 0 und 23 sein.

PROMPT
Auf den Bildschirm auszugebender Text, der erlaeutert, was fuer Daten eingegeben und/oder angezeigt werden. Man kann Text oder eine Escape-Sequenz eingeben, wodurch bestimmt wird, das der Prompter in einem bestimmten Modus angezeigt werden soll, sofern der termcap-Eintrag fuer das aktuelle Terminal diesen Modus zulaesst (siehe Abschnitt 1.1.4). Das Umschalt-Zeichen ist die Tilde (~). Die folgenden Modi sind zulaessig:

~r - Video invers (Anfang)
~s - Video invers (Ende)
~u - Anfang der Unterstreichung
~v - Ende der Unterstreichung
~w - Anfang Video invers unterstrichen
~x - Ende Video invers unterstrichen

Schaltet man fuer einen Prompter einen bestimmten Anzeigemodus ein, muss man diesen am Ende des Prompters

wieder beenden. Sonst erscheint die Bildmaske nicht richtig auf dem Bildschirm.

PX
Die x-Koordinate (Spalte) des Prompters auf der Bildmaske. Es muss eine Zahl zwischen 0 und 79 sein.

PY
Die y-Koordinate (Zeile) des Prompters. Es muss eine Zahl zwischen 0 und 23 sein.

4.3 Pruefen von Bildmasken

Mit dieser Option kann man eine Bildmaske testen, ohne dass man erst ein Programm schreiben muss, durch das sie angezeigt wird. Man kann damit leicht die Position der Prompter und Daten-Eingabestellen ueberpruefen. Das Programm wird gestartet, indem man im SFORM-Menue 'Test Screen' waehlt. Folgendes Bild erscheint auf dem Schirm:

```
[sfsamp]                    WDATA SYSTEM
                       24 JUL 1986 - 15:25
                            Test Screen


SCREEN: _
```

Es ist nur ein Prompter, SCREEN:, zu beantworten. Man kann den Namen einer Bildmaske eingeben und diese wird angezeigt. Mittels CTRL/U kehrt man in den Menue-Handler zurueck, RETURN wird ignoriert.

4.4 Verarbeitung von Bildmasken - Process Screen

Mit dieser Option werden die zwei in Abschnitt 4 beschriebenen Dateien zur Verwendung durch das Kundenprogramm angelegt, bzw. aktualisiert. Die eine Datei (.h) wird von Programmen einbezogen, die sich auf Namen von Bildmaskenfeldern beziehen. Die andere (.q) wird waehrend der Laufzeit durch die Funktion loadscr gelesen, um die Bildmaske anzuzeigen und die von anderen SFORM-Funktionen benoetigten, im Speicher befindlichen Tabellen, zu laden. Das Programm wird gestartet, indem in 'SFORM Menu' 'Process Screen' ausgewaehlt wird. Es erscheint folgendes Bild auf dem Schirm:

```
[sfproc]                    WDATA SYSTEM
                       24 JUL 1986 - 15:25
                          Process Screen

```

Dieses Programm hat keine Optionen. Nach Beendigung dieses Programms (selbstaendige Rueckkehr zum Menue-Handler) verfuegt man ueber gueltige .q- und .h-Dateien fuer alle Bildmasken.

4.5 Reporte der Bildmasken

Waehlen Sie im SFORM-Menue 5, 'Screen Reports', Sie erhalten damit eine Auflistung der eingegebenen Bildmasken. Die Auflistung kann als Dokumentation fuer das System verwendet werden. Der Report ist 120 Zeichen breit. Die folgende Bildmaske erscheint:

```
[sfrep]                          WDATA SYSTEM
                              24 JUL 1986 - 15:25
                                Screen Reports

SCREEN  _

LISTING [Y or N]:
PRINT SCREEN:
```

Die Prompter haben folgende Bedeutung:

SCREEN
: Name einer zu druckenden, existierenden Bildmaske. Alle Bildmasken im Datenwoerterbuch koennen jedoch auch gedruckt werden, indem dieser Prompter mit ALL beantwortet wird.

LISTING [Y oder N]:
: Y bedeutet den Druck eines tabellarischen Reports, in dem die Attribute aller Bildmaskenfelder aufgefuehrt sind. N bedeutet, dass dieser Teil des Reports nicht gedruckt wird.

PRINT SCREEN:
: Y bewirkt, dass die Abbildung der Anzeige so ausgedruckt wird, wie sie auf einem Bildschirm erscheinen wuerde. N bedeutet, dass die Abbildung nicht gedruckt wird.

4.6 Wiederherstellen von Bildmasken

Diese Option wird verwendet, um Bildmasken wiederzugewinnen, die aus Versehen aus dem Datenwoerterbuch geloescht wurden oder um alle Prompter in einer Bildmaske umzuordnen. Sind viele Veraenderungen vorgenommen worden und die Prompter sind in Unordnung geraten, wird die Bildmaske (trotzdem) verarbeitet (Abschnitt 4.4), dann geloescht (Abschnitt 4.2) und dann wird dieses Programm zur Wiedergewinnung verwendet. Die Prompter werden in der Reihenfolge ihrer

Anzeige umgeordnet. Das Programm liest die .q-Datei in das aktuelle Verzeichnis und fuegt die Bildmaske wieder in das Datenwoerterbuch ein. Das Programm wird gestartet, indem in 'SFORM Menu' 'Restore Screen' gewaehlt wird. Es wird folgender Bildschirm angezeigt:

```
[sfrestr]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                            Restore Screen

SCREEN: _
```

Hinter dem Prompter SCREEN: wird der Name einer Bildmaske eingegeben, und diese Bildmaske wird entsprechend der .q-Datei wiederhergestellt. Danach wird der Bildschirminhalt geloescht und man kann einen neuen Bildmaskennamen eingeben. Mittels CTRL/U kehrt man in den Menue-Handler zurueck, RETURN wird ignoriert.

4.7 Liste der Bildmasken

Dieses Programm listet auf dem Terminal die Namen aller im Datenwoerterbuch befindlichen Bildmasken auf. Es gibt keine Optionen oder Prompter, die beantwortet werden muessten. Das Programm wird gestartet, indem in 'SFORM Menu' 'Screen List' gewaehlt wird. Es folgt eine typische Anzeige:

```
[sfslist]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                             List Screen

her
modell
art
sher100
smod100
sart100
erwlist
best
sbest100

 ->-> _
```

Das Programm zeigt die Namen an und wartet vor Rueckkehr in den Menue-Handler auf das Druecken der Taste RETURN.

4.8 Erstellen einer Standardbildmaske

Unter Verwendung dieses Programms kann eine Bildmaske, die auf Felder in einem bestimmten Datensatztyp Bezug nimmt,

mit einem Schritt erstellt (Abschnitt 4.1, 4.2), verarbeitet (Abschnitt 4.4) und mit ENTER registriert werden (Abschnitt 5.1). Das Programm wird gestartet, indem man in 'SFORM Menu' 'Create Default Screen Form' waehlt. Folgendes Bild wird auf dem Schirm angezeigt:

```
[cdsf]                          WDATA SYSTEM
                            24 JUL 1986 - 15:25
                        Create Default Screen Form

RECORD: _
```

Hinter dem Prompter RECORD: muss man den Namen eines existierenden Datensatztyps eintragen, damit wird eine Standard-Bildmaske fuer diesen Datensatztyp generiert. Nachdem die Standard-Bildmaske erstellt, verarbeitet und mit ENTER registriert wurde, wird der Bildschirminhalt geloescht und man kann den Namen eines neuen Datensatztyps eingeben. Mittels CTRL/U kehrt man zum Menue-Handler zurueck, RETURN wird ignoriert.

Wird ein Datensatztyp angegeben, dessen Felder nicht in die Standard-Bildmaske passen, wird die Meldung

```
  There is not enough room on the screen for all of the
                                                    fields ->->
```

angezeigt. Wird hinter dem Prompter

```
  Continue?
```

ein Y eingegeben, enthaelt die Standard-Bildmaske nur die hineinpassenden Felder. Wird dieser Prompter mit N beantwortet, kehrt man zum Prompter RECORD: zurueck. Die Standard-Bildmaske wird nicht erstellt.

Sollte die entsprechende Bildmaske schon existieren, erscheint:

```
  Screen already exists ->->
```

Die letzte Meldung, die erscheint, bevor man den Namen eines weiteren Datensatztyps eingeben kann, lautet:

```
  The screen can now be used for data entry ->->
```

Um die Bildmaske zur Dateneingabe zu nutzen, wird hinter dem Prompter SELECTION:, der auf jedem Menue unten erscheint, der Name der Bildmaske eingegeben.

Da die Bildmaske mit ENTER registriert wurde, kann sie in die Liste der Optionen eines bestimmten Menues aufgenommen werden. Das kann geschehen, indem man das Programm 'Menu Maintenance' verwendet, das im 'MENUH Screen Menu' zu fin-

den ist (siehe Abschnitt 2.1.2).

Unter Verwendung von 'Screen Entry' (siehe Abschnitt 4.2) kann eine Standard-Bildmaske modifiziert oder geloescht werden. Man kann auch unter Verwendung von 'ENTER Screen Registration' (Abschnitt 5.1) die Ueberschrift aendern. Wird eine Bildmaske geloescht, wird auch ihre Registrierung in ENTER geloescht. War sie als Option in einem Menue registriert, wird diese Eintragung ebenfalls geloescht.

Im allgemeinen werden die Felder in der Reihenfolge ihres Auftretens in 'SCHEMA REPORTS' auf der Bildmaske von oben nach unten angeordnet. Die Felder werden untereinander angeordnet und rechts entsprechend dem laengsten Feldnamen beendet. Die Ueberschrift der Bildmaske ist der vollstaendige oder regulaere (kurze) Name des Basis-Datensatztyps gefolgt von dem Wort 'Maintenance'. Alle Unterstreichungszeichen werden in Leerzeichen geaendert und die Anfangsbuchstaben gross geschrieben.

Die Prompter werden in einer, zwei oder drei Spalten ausgerichtet. Der Prompter eines Feldes ist sein vollstaendiger Name (siehe Uebersicht), oder falls dieser nicht angegeben ist, wird der regulaere (kurze) Name des Feldes verwendet. In beiden Faellen werden alle Unterstreichungszeichen durch Leerzeichen ersetzt. Ein Datenbank-Feld hat einen impliziten Namen, wenn es ein Bezugsfeld ist. Der implizite Name besteht aus dem regulaeren Namen des Feldes und dem Namen des Feldes, auf das es Bezug nimmt. Auch das Feld, auf das Bezug genommen wird, kann auf ein anderes Feld Bezug nehmen, was dazu fuehrt, dass auch der Name dieses Feldes im impliziten Namen erscheint.

Die Elemente eines COMB-Feldes erhalten gesonderte Prompter. Auch wenn ein Feld auf ein COMB-Feld Bezug nimmt, werden gesonderte Prompter fuer jedes der Elementarfelder erzeugt, auf die Bezug genommen wird.

Es soll beispielsweise das Erstellen einer Standard-Bildmaske fuer den im Nutzerhandbuch angefuehrten Datensatztyp kunde erstellt werden. Hier die Felder des Datensatztyps kunde:

RECORD/FIELD	REF	TYPE	LEN	LONG NAME
kunde	10			kunde
*kunr		NUMERIC	5	kundennummer
kuname		STRING	30	name
kustr		STRING	30	strasse
kort		STRING	20	ort
kuplz		NUMERIC	4	postleitzahl
kutelex		NUMERIC	7	telex
kuruf		NUMERIC	7	ruf

In 'SFORM Menu' wird 'Create Default Screen Form' gewaehlt und hinter dem Prompter RECORD: wird der Datensatztypname

kunde eingegeben. Waehrend die Bildmaske erstellt, verarbeitet und mit ENTER registriert wird, erscheinen Verarbeitungsmeldungen. Nach Beendigung des Prozesses, muss der Bildschirm folgendermassen aussehen:

```
[cdsf]                          WDATA SYSTEM
                            24 JUL 1986 - 15:25
                        Create Default Screen Form

RECORD: kunde

The screen will be named kunde

Created
Processed
Registered with ENTER

The screen can now be used for data entry.  ->-> _
```

Zur Rueckkehr ins Menue wird CTRL/U gedrueckt und hinter dem Prompter SELECTION: wird kunde eingegeben. Es erscheint folgende Bildmaske, die zur Eingabe von Daten bereit steht:

```
[kunde]                         WDATA SYSTEM
                            24 JUL 1986 - 15:25
                             Kunde Maintenance

kundennummer:
name        :
strasse     :
ort         :
postleitzahl:
telex       :
ruf         :

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

5. Dateneingabe und Anfragen mit ENTER

ENTER ist ein vielseitiges Programm, mit dem man Bildmasken zur Dateneingabe und Abfrage entwickeln kann, ohne selbst ein Programm schreiben zu muessen. ENTER gestattet ein effizientes und nutzerfreundliches Hinzufuegen von Daten zu einer Datenbank und liefert ein wirksamenes und einfach zu verwendendes Abfrage-Interface. Dadurch kann man gewuenschte Daten suchen und anzeigen, ohne eine Abfrage schreiben zu muessen. Sind die Daten gefunden, kann man sie aktualisieren, loeschen oder in einem Report zusammenfassen. ENTER ermoeglicht das, indem es "hinter" den unter Verwendung von SFORM entwickelten Bildmasken (Abschnitt 4) ablaeuft. Man braucht nur unter Verwendung der SFORM-Werkzeuge die Bild-

maske anzugeben und ENTER erledigt den Rest.

In den folgenden Abschnitten wird erlaeutert, wie man mit ENTER eine SFORM-Bildmaske registriert (und damit eine ENTER-Bildmaske anlegt) und wie man die Bildmaske "bedient", wenn sie erst einmal registriert wurde. ENTER-Bildmasken sind automatisch an den Menue-Handler angeschlossen, so dass die Sicherung ueber Zugriffsrechte auf sie angewandt wird. Die Selektionsdatei, die durch eine Anfrage ueber ENTER erzeugt wird, kann von ENTER selbst, von RPT (siehe Abschnitt 5.2.3 und 7.9.2), von LST (siehe Abschnitt 8.2) oder von einem in einer Programmiersprache, wie z.B. in C, geschriebenen Kundenprogramm benutzt werden (siehe Abschnitt 10).

Wenn eine Bildmaske zur Verwendung durch ENTER entwickelt wird, ist es wichtig, die Regeln zu verstehen, die ENTER benutzt. Diese Regeln entscheiden ueber die Reihenfolge, in der die Felder auf der Bildmaske akzeptiert und auf Gueltigkeit geprueft werden, und ob ENTER die Bildmaske ueberhaupt bedienen kann. Es ist naemlich moeglich mit SFORM Bildmasken zu definieren, die nicht von ENTER bedient werden koennen. In diesem Fall kann ein Kundenprogramm zum Ansteuern des Bildschirms verwendet werden.

Eine ENTER-Bildmaske behandelt einen einzigen Arbeits-Datensatztyp, der derjenige ist, der hinzugefuegt, geloescht, geaendert oder abgefragt werden soll. Felder anderer Datensatztypen koennen auf der Bildmaske erscheinen, um "Einblicke" in die Datenbank zu gewaehren. Diese sekundaeren Felder koennen nur angezeigt werden. Soll ENTER ordnungsgemaess ablaufen, muessen explizite Beziehungen zwischen den Arbeits-Datensatztypen und den sekundaeren Datensatztypen bestehen. Im folgenden werden die Grundregeln fuer Bildmasken aufgestellt, die mit ENTER bedient werden sollen:

1. Nur der angegebene Arbeits-Datensatztyp wird von einer ENTER-Bildmaske manipuliert. Der Arbeits-Datensatztyp wird mit "ENTER Screen Registration" festgelegt, siehe Abschnitt 5.1. Felder anderer Datensatztypen koennen nur angezeigt werden. Nur Arbeits-Datensaetze koennen hinzugefuegt oder geloescht werden. Der Inhalt von Feldern anderer Datensaetze wird nur angezeigt, wenn ein vorher auf der Bildmaske angezeigtes Feld ein explizites Bezugsfeld vom Arbeits-Datensatz zu diesem anderen Datensatztyp war. So wird z.B. im Nutzerhandbuch der Name des Herstellers nur deshalb auf der Modell-Bildmaske angezeigt, weil vorher die Kennzahl des Herstellers, mohenr bzw. hersteller_num, angezeigt wurde. hersteller_num ist eine explizite Beziehung zwischen Modell und Hersteller.

2. Steht ein Kombinationsfeld (Typ COMB) in explizitem Bezug zu einem anderen Datensatz, hohlt ENTER zuerst alle Elemente der Kombination, bevor es versucht, das Kombinationsfeld in der Datenbank zu speichern. Stehen auch

Teile der Kombination in einer Beziehung, werden sie nacheinander auf Gueltigkeit untersucht, aber nicht gespeichert. Sind nur Teile eines Kombinationsfeldes auf dem Bildschirm, wird jeder Teil einzeln gespeichert. Aufgrund dieser Regel ist es normalerweise besser, wenn man alle Teile eines Kombinationsfeldes gemeinsam auf den Bildschirm bringt, so dass der Cursor nicht immer von einer Position springt.

3. Beim Hinzufuegen (ADD) muss als erstes der Primaerschluessel eingegeben werden, bevor ein anderes Feld eingegeben werden kann. Handelt es sich bei dem Schluessel um ein Kombinationsfeld, muessen alle Teile gehohlt und auf Gueltigkeit untersucht werden. Es ist am besten, wenn man den Primaerschluessel auf der Bildmaske oben platziert.

4. Bei allen Bildmaskenfeldern muessen die Datenbankfeld-Koordinaten FX und FY und die Prompter-Koordinaten PX und PY ausgefuellt sein, auch wenn kein Prompter vorhanden ist. ENTER benutzt die FX-FY-Koordinaten, wenn es Daten vom Bildschirm der Terminals loescht, die keine geschuetzten Felder haben. Die PX-PY-Koordinaten werden verwendet, um die Felder auf der Bildmaske zu ordnen und zu bestimmen, wie sich der Cursor von einem Feld zum naechsten bewegt. Wird FX-FY auf Null gelassen, wird beim Loeschen die erste Zeile des Bildschirms geloescht. Wird PX-PY auf Null gelassen, bewegt sich der Cursor sprunghaft von einem Feld zum naechsten. ENTER ignoriert Bildmasken-Felder, die sich nicht auf Datenbank-Felder beziehen.

5. ENTER nutzt die letzten drei Zeilen auf dem Bildschirm zur Anzeige von Informationen, Promptern und Fehlermeldungen. Daher sind die Zeilen 21, 22 und 23 nicht fuer die Anzeige nutzereigener Daten zu verwenden, da diese ueberschrieben werden!

5.1 Registrierung von Bildmasken mit ENTER

Bevor ENTER eine SFORM-Bildmaske bedienen kann, muss man ENTER zunaechst ueber diese Bildmaske informieren und darueber, wie diese verwendet werden soll.

Der Registrierungsprozess macht das, indem er einer bestimmten Bildmaske einen Arbeits-Datensatztyp, einen Titel und eine optionelle Liste von Reporten zuordnet. Es ist erforderlich, den Arbeits-Datensatztyp anzugeben, da Felder aus verschiedenen Datensatztypen auf der Bildmaske sein koennen. Das Ergebnis dieses Prozesses ist ein Eintrag im Datenwoerterbuch, der als ENTER-Bildmaske bezeichnet wird. Eine SFORM-Bildmaske kann auch mit SQL (Abschnitt 6.2.6) registriert werden, aber nicht mit beiden. Will man wissen, ob eine existierende Bildmaske mit ENTER oder SQL registriert wurde, muss ein 'Executable Listing' (Abschnitt

2.2.1) gedruckt werden.

'ENTER Screen Registration' ermoeglicht auch die Zuordnung einer Anzahl von Reporten zum Bildschirm. Jeder Report ist eine Kombination aus Formatierungsprogramm, Parametern, Titel und einem Satz der zulaessigen Reportausgabe-Ziele. Diese Reporte koennen dann unter Verwendung von 'Report Options Screen' ablaufen (siehe Abschnitt 5.2.3).

Ist eine ENTER-Bildmaske erst einmal registriert, kann sie in ein Menue gebracht werden (siehe Abschnitt 2.1.2). Achtung: Wird eine ENTER-Bildmaske aus dem Datenwoerterbuch geloescht, werden alle Verweise auf diese (d.h. alle Menueeintraege, die sich auf sie beziehen) geloescht. Mit 'Menu Listing' (Abschnitt 2.2.2) erhaelt man eine Auflistung aller Menues im System.

Das Programm wird gestartet, indem man in 'System Menu' 'ENTER Screen Registration' waehlt. Man kann immer nur eine SFORM-Bildmaske gleichzeitig mit ENTER registrieren. Auf dem Bildschirm erscheint:

```
[entmnt]                    WDATA SYSTEM
                        24 JUL 1987 - 15:25
                    ENTER Screen Registration

SCREEN NAME:
TARGET RECORD:                HEADING:

FORMATTING PROGRAMS:
      NAME       PARAMETERS   TITLE                OUTPUT
                -
                -
                -
CMD _____________________________________________________

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

Die Prompter haben folgende Bedeutung:

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE:

Mit diesem Prompter kann man einen Betriebsmodus fuer das Programm waehlen. Die verschiedenen Modi haben folgende Bedeutung:

i - Abfragemodus, damit kann man eine Bildmaske betrachten, die bereits mit ENTER abgelegt wurde.

a - Ergaenzungsmodus, damit kann man eine Bildmaske mit ENTER ablegen.

m - Aenderungsmodus, damit kann man TARGET RECORD, HEADING und Formatierungsprogramme fuer eine bereits

mit ENTER abgelegte Bildmaske modifizieren.

d - Loeschmodus, damit kann man eine durch ENTER registrierte Bildmaske entfernen. Wird eine abgelegte Bildmaske geloescht, bleibt die urspruengliche, von SFORM erzeugte Bildmaske unberuehrt (Loeschen von SFORM-Bildmasken siehe Abschnitt 4.2). ACHTUNG: Bei Loeschen einer ENTER-Bildmaske unter Verwendung von "ENTER Screen Registration" werden alle Verweise darauf (d.h. Menueauflistungen) entfernt. Man erhaelt eine Aufstellung aller im System vorhandenen Menues, indem man das "Menu Listing"-Programm verwendet (Abschnitt 2.2.2).

SCREEN NAME
Der Name der abzulegenden Bildmaske.

TARGET RECORD
Der Name des Datensatztyps, der von der Bildmaske beibehalten wird (d.h. der hinzugefuegte, modifizierte usw.). Es ist erforderlich, den Arbeits-Datensatztyp anzugeben, da auf der Bildmaske viele verschiedene Datensaetze erscheinen koennen.

HEADING
Eine Zeichenkette von bis zu 34 Zeichen, die auf der dritten Bildschirmzeile erscheint, wenn ENTER arbeitet. Dieser Kopf wird ebenfalls auf allen Menues angezeigt, auf denen sich die Bildmaske befindet. Man kann druckbare Zeichen oder nichtdruckbare Zeichen eingeben, die angeben, dass die Ueberschrift in einem bestimmten Modus auf dem Bildschirm erscheinen soll, vorausgesetzt, dass die termcap-Tabelle fuer das aktuelle Terminal diesen Modus zulaesst (siehe Abschnitt 1.1.4). Das Umschaltzeichen ist die Tilde (~). Folgende Modi sind zugelassen:

~r - Anfang Video invers
~s - Ende Video invers
~u - Unterstreichung, Anfang
~v - Unterstreichung, Ende

Setzt man einen bestimmten Darstellungsmodus fuer eine Ueberschrift, muss man diesen nach der Ueberschrift immer beenden.

FORMATTING PROGRAMS
Im Bereich unter diesem Prompter kann man die sich auf diese ENTER-Bildmaske beziehenden Reporterzeugungsprozesse aktualisieren und darstellen. Die Kombination aus dem NAME, den PARAMETERS, dem TITLE und dem OUTPUT eines Formatierungsprogrammes wird als Report bezeichnet.

NAME
Der erste Prompter im Aktualisierungsbereich. Es akzeptiert einen aus acht Zeichen bestehenden Namen fuer das

auszufuehrende Formatierungsprogramm, das zur Erzeugung der gewuenschten Ausgabe benutzt wird. Jeder abgelegten Bildmaske koennen maximal 8 verschiedene Formatierungsprogramme zugeordnet werden.

Bei ENTER-Screen-Registration kann das Formatierungsprogramm LST, der Reportgenerator RPT, ein vom Nutzer erstelltes Formatierungsprogramm, das nicht zu WEGA-DATA gehoert, oder eine Leerstelle sein (LST und RPT sind in Grossbuchstaben zu schreiben). Ein vom Nutzer erstelltes Formatierungsprogramm wird als "Nutzerprogramm" bezeichnet.

Alle nach WEGA-DATA erlaubten, aber von RPT und LST abweichenden Namen, sind nicht gueltig und werden nicht akzeptiert. Wird ein Leerzeichen eingegeben, wird kein Formatierungsprogramm benutzt und als Report erscheinen die Felder in einfacher tabellarischer Form auf dem "Formular". Deshalb kann eine Ausgabe, die sich aus der Eingabe eines Leerzeichens ergibt, dazu fuehren, dass Zeilen ueberschrieben werden.

Dieses Feld wird so editiert, dass nur die vor einem Leerzeichen eingegebenen Zeichen verbleiben. Die Zeichenkette wird durch RETURN linksbuendig ausgerichtet, vorangestellte Leerstellen und zusaetzliche Zeichen werden geloescht.

PARAMETERS

Stellt vier Zeichenketten bereit, die jeweils aus 14 Zeichen bestehen. Diese werden als Parameter dem Formatierungsprogramm uebergeben. Es ist Aufgabe des Formatierungsprogramms, die Parameter zu interpretieren. Die Parameter werden vertikal im Anschluss an jeden Bindestrich (-) eingegeben. Die Anzahl der zulaessigen Parameter wird durch den verwendeten Typ des Formatierungsprogramms bestimmt.

Formatierungsprogramm		erlaubte PARAMETERS-Felderanzahl
RPT	-	1 (Der Name eines RPT-Skripts)
LST	-	1 (Der Name eines LST-Skripts)
Leerzeichen	-	0
Nutzerprogramm	-	4

Eine PARAMETERS-Eingabe muss ein Wort sein, in dem keine Leerstellen sein duerfen und dem auch keine Leerstellen vorangestellt sein duerfen. Wenn jedoch NAME ein Nutzerprogramm ist, dann werden die Parameter nicht editiert und es kann mehr als ein echter Parameter in das PARAMETER-Feld eingebracht werden, indem Leerstellen gelassen werden (vorausgesetzt, die Gesamtlaenge ueberschreitet nicht 14 Zeichen). Das Nutzerprogramm wird gestartet, als waere folgender Befehl unter der Shell eingegeben worden:

programm sel_datei p1 p2 p3 p4

Dabei ist sel_datei der Name der Auswahldatei, die durch die vom Nutzer ergehende Anfrage-durch-Bildmaske aufgebaut wurde und p1-p4 sind die vier Parameter-Zeichenketten. sel_datei kann unter Verwendung einiger im WEGA-DATA-Systemhandbuch, Abschnitt 10, beschriebenen Funktionen manipuliert werden (z. opensf, closesf, frstsel, nextsel, prevsel).

TITLE

Eine Kennzeile fuer den Report, die erzeugt wird, wenn man diese Option ueber 'ENTER Report Options Screen' (Abschnitt 5.2.3) waehlt. Dieses Feld erscheint mit seiner Selektionsnummer auf dem 'Report Options Screen'.

OUTPUT

Legt die zulaessigen Ausgabeeinheiten fuer Reportausgabe und optionellen Debug-Modus fest. Die Ausgabe kann pro Durchlauf auf ein oder mehrere Ziele geleitet werden. Es werden bis zu vier Zeichen akzeptiert. Diese sind:

S - bewirkt Ausgabe auf Bildschirm (Standard)
P - bewirkt Ausgabe auf Drucker
F - bewirkt Ausgabe in eine Datei
N - ermoeglicht den Debug-Modus ("Ein" ist Standard)

Die Eingabe von Kleinbuchstaben ist moeglich. Diese werden in Grossbuchstaben konvertiert. Doppeltes Auftreten eines Zeichens wird ignoriert.

Wird der Report im Debug-Modus erstellt, werden alle Fehlermeldungen, die verwendeten Skripts und die Laufzeitergebnisse angezeigt. Bei aktivem Debug-Modus koennen die verwendeten aktuellen Skripts verifiziert werden. Syntaxfehler werden angezeigt, um die Korrektur von Programmen und Skriptfehlern zu erleichtern.

CMD

Der restliche Bildschirm kann fuer die Datenanzeige verwendet werden. Hier wird die erste Zeile jedes registrierten Reports in der Reihenfolge angezeigt, in der diese auf dem 'ENTER Report Options Screen' erscheinen. Einzelne Reporte koennen im Aktualisierungsbereich hinzugefuegt und veraendert werden. Die Reporte werden jedoch aus diesem Bereich selbst geloescht. Unter CMD werden Kommandos fuer eine Zeile eingegeben.

q - Reporteingabe beenden und Datenanzeigebereich verlassen. Kann fuer jede beliebige Zeile eingegeben werden.

d - Loeschen des aktuellen Reports aus dieser ENTER-Bildmaske.

a - Hinzufuegen eines neuen auf diese ENTER-Bildmaske bezogenen Reports. a kann nur in die Leerzeile am Ende des Datenanzeigebereiches eingegeben werden.

- Eine Zahl, die zwischen 1 und x liegt, und die eine Zeilennummer darstellt (x ist die Anzahl der in dieser Bildmaske registrierten Reporte). Das aktuelle Objekt wird zur angegebenen Zeilennummer gebracht.

m - Gestattet die Modifizierung aller zum aktuellen Report gehoerenden Felder. ACHTUNG: Wird NAME auf ein Leerzeichen geaendert, werden alle bis dahin existierenden Parameterfelder geloescht. Wird NAME auf ein von WEGA-DATA geliefertes Formatierungsprogramm geaendert, werden alle Parameterfelder mit Ausnahme des ersten geloescht.

5.2 Verwendung von ENTER-Bildmasken

Eine ENTER-Bildmaske hat 4 Betriebsmodi - Abfrage-, Ergaenzungs-, Aenderungs- und Loeschmodus. In Abhaengigkeit von den in 'Group Maintenance' und 'Employee Maintenance' festgelegten Zugriffsrechten kann ein Modus oder koennen mehrere Modi unzulaessig sein. Im Ergaenzungsmodus muss man zuerst das Primaerschluessel-Feld ausfuellen. Besteht der Schluessel aus mehreren Feldern, muessen alle Felder auf dem Bildschirm erscheinen. Nachdem der Primaerschluessel eingegeben wurde, kann man jedes Feld auf der Bildmaske modifizieren.

In den anderen drei Modi kann man Datensaetze suchen, indem man den/die gewuenschten Prompter auf dem Bildschirm ausfuellt. ENTER sucht die dazugehoerigen Datensaetze. Der Satz von Datensaetzen, der der Abfrage entspricht, kann dann entsprechend den Anforderungen datensatzweise untersucht, aktualisiert oder geloescht werden. Die Datensaetze koennen aber auch unter Verwendung eines Reportskripts formatiert und die Ergebnisse auf den Bildschirm, den Drucker oder in eine Datei ausgegeben werden. In den folgenden Abschnitten werden die zwei Hauptarten der Verwendung von ENTER beschrieben - Dateneingabe und Anfragen ueber (ENTER-)Bildmasken.

5.2.1 Dateneingabe mit ENTER

Unter Dateneingabe versteht man i.a. den Prozess des Hinzufuegens neuer Daten in die Datenbank oder die Modifizierung existierender Daten. In diesem Abschnitt soll beschrieben werden, wie Ergaenzungsmodus und Aenderungsmodus benutzt werden, nachdem man einen bestimmten zu modifizierenden Datensatz lokalisiert hat. Im naechsten Abschnitt soll beschrieben werden, wie man Datensaetze lokalisiert. Die Verwendung von ENTER kann man wahrscheinlich am besten

an Hand von Beispielen verstehen. Daher ist es empfehlenswert, dass der Leser die im WEGA-DATA-Nutzerhandbuch angefuehrten Beispiele zur Verwendung der einzelnen ENTER-Bildmasken durcharbeitet.

Generell arbeitet eine ENTER-Bildmaske folgendermassen: Im Ergaenzungsmodus geht der Cursor zuerst ins Schluesselfeld. Der Schluessel muss eingegeben werden, bevor der Cursor auf dem Bildschirm weiter nach unten bewegt werden kann. Ist der Schluessel ein Kombiniationsfeld, das aus mehreren Teilen besteht, muessen alle Teile eingegeben werden. Nach Angabe des Schluessels wird der Datensatz sofort in die Datenbank eingefuegt und gesperrt, so dass andere Nutzer an anderen Terminals ihn nicht aendern oder loeschen koennen.

Im Aenderungsmodus geht der Cursor auch zuerst ins Schluesselfeld. Hier kann man jedoch 'Anfragen ueber Bildmasken' verwenden, um die gewuenschten Datensaetze zu lokalisieren, d.h. beliebige Felder zur Suche ausfuellen. Benutzt gerade jemand anders an einem anderen Terminal denselben Datensatz (d.h. wird dieser dort modifiziert oder geloescht), wird eine Fehlermeldung angezeigt, wenn man versucht, ihn zu modifizieren, und die Aktualisierung des Datensatzes wird nicht zugelassen. Erst nachdem der andere Nutzer seine Arbeit an dem Datensatz abgeschlossen hat, kann man erneut einen Versuch unternehmen. Nach Lokalisieren des gewuenschten Datensatzes verhaelt sich der Bildschirm genau wie im Ergaenzungsmodus.

Wurde ein bestimmter Datensatz hinzugefuegt oder gefunden, geht der Cursor auf den ersten auf dem Bildschirm stehenden Prompter. Von dort kann man vermittels RETURN den Cursor auf dem Schirm nach unten und vermittels CTRL/U auf dem Schirm nach oben bewegen. Bei Erreichen des letzten Prompters springt der Cursor wieder ganz nach oben. Befindet sich der Cursor im ersten Prompter und wird CTRL/U gedrueckt, werden die Daten in Vorbereitung der naechsten Operation auf dem Schirm geloescht. Wird nun nochmals CTRL/U gedrueckt, erscheint erneut der Betriebsmodus-Prompter ([I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE) unten auf dem Schirm. Die Eingabe von CTRL/X bewirkt an jeder Stelle eine Rueckkehr zum Menue-Handler.

Sollen die in einem Prompter angezeigten Daten geaendert werden, wird so lange RETURN gedrueckt, bis sich der Cursor auf diesem Prompter befindet. Dann werden die neuen Daten eingegeben und mit RETURN abgeschlossen. Werden die neuen Daten akzeptiert, werden sie in der Datenbank gespeichert und der Cursor geht auf den naechsten Prompter. Werden sie nicht akzeptiert, erscheint eine Fehlermeldung unten auf dem Schirm und die Datenbank wird nicht aktualisiert. Zur Bestaetigung der Fehlermeldung muss RETURN gedrueckt und ein erneuter Versuch unternommen werden.

Hat man mit der Eingabe von Daten begonnen und man aendert dann seine Absicht, kann man vermittels CTRL/U die Eingabe

ohne Aenderung der Datenbank loeschen. Zur Aenderung von Daten in einem sekundaeren Datensatz muss eine andere ENTER-Bildmaske definiert werden, die diesen sekundaeren Datensatztyp zur Basis hat oder man benutzt ein Nutzerprogramm, um diese Bildmaske zu bedienen.

5.2.2 Anfragen ueber Bildmasken

Ene Anfrage ueber Bildmaske ist ein Prozess, mit dem ein Datensatz oder eine Menge von Datensaetzen gesucht wird, der/die ueberprueft (d.h. abgefragt), modifiziert oder geloescht werden soll. In diesem Abschnitt wird beschrieben, wie man den/die zu bearbeitenden Datensaetze lokalisieren kann. Im vorhergehenden Abschnitt wird die Eingabe von Daten beschrieben. Wie die Dateneingabe, kann man auch die 'Anfragen uber Bildmasken' am besten im Zusammenhang mit einem konkreten Beispiel erklaeren. Deshalb wird empfohlen, das im WEGA-DATA-Nutzerhandbuch angefuehrte Beispiel durchzuarbeiten.

Anfragen ueber Bildmasken arbeiten folgendermassen. Hat man hinter dem auf dem Bildschirm unten angezeigten Prompter den Betriebsmodus gewaehlt (I, A, M oder D), wird folgender Prompter angezeigt:

```
Begin search [CTRL E],Clear field [CTRL Z],Exit [CTRL X]
```

Es handelt sich hierbei um die Standard-Steuertastenwerte fuer diese Funktionen. Man kann diese veraendern, indem man die Datei termcap gemaess der in Abschnitt 1.1.4 erfolgten Erklaerung editiert. Diese Steuerzeichen werden benutzt, um folgende Funktionen auszufuehren:

CTRL/E - Beginnt die Suche mit den eingegebenen Optionen. Man kann auch, um die Suche zu starten, einfach RETURN druecken, bis man zum letzten Bildmaskenfeld gelangt, um die Suche zu starten.

CTRL/Z - Damit wird die letzte Auswahlspezifikation geloescht. Hat man beispielsweise einen Feldwert fuer Erwerbsdatum eingegeben und gelangt dann zu der Auffassung, dass das Datum eigentlich gar keine Rolle spielt, stellt man den Cursor auf den Prompter Erwerbsdatum und drueckt CTRL/Z. Die Spezifikation wird dann geloescht und Erwerbsdatum wird in der Anfrage nicht verwendet.

CTRL/X - Verlaesst die aktuelle ENTER-Bildmaske und sorgt fuer Rueckkehr in den Menue-Handler. Im Gegensatz zu den zwei anderen CTRL-Eingaben, kann CTRL/X immer dann verwendet werden, wenn ein normales Zeichen auch akzeptiert wuerde. Damit kann man schnell eine ENTER-Bildmaske entfernen.

Die einfachste Anfrage besteht darin, dass man einen be-

stimmten Datensatz ueber seienen Primaerschluessel sucht. Da es sich hier um eine Zugriff ueber Primaerschluessel handelt, wird der Datensatz sofort gefunden, ohne dass sich ein Suchen erforderlich macht. Nun kann RETURN gedrueckt werden, wodurch man in ein beliebiges gewuenschtes Feld auf dem Bildschirm gelangen und den Datensatz durch die Eingabe neuer Daten aktualisieren kann.

Ausser der Anfrage ueber den Primaerschluessel, bietet ENTER zwei Moeglichkeiten der Anfrage - genaue Uebereinstimmung, d.h. es wird genau das eingegeben, was die ausgewaehlten Datensaetze enthalten sollen; und teilweise Uebereinstimmung, d.h. es werden vom Nutzer Sonderzeichen eingegeben, die waehrend des Zuordnungsprozesses eine unterschiedliche Bedeutung haben. Beispiele sind Zahlen- und Datenbereiche (d.h. Zahlen von 1 bis 100 oder die Daten vom 3/1/86 bis 3/31/86) oder die Uebereinstimmung von Teil-Zeichenketten (d.h. alle Zeichenketten, die die Zeichenkette "zange" enthalten).

Um eine genaue Uebereinstimmung anzugeben, werden einfach die auf dem Bildschirm befindlichen Felder genau mit der Information ausgefuellt, die man sehen moechte. Zur Angabe einer teilweisen Uebereinstimmung von Zeichenkettenfeldern wird der gleiche Satz von Sonderzeichen verwendet, der von SQL und dem Listen-Prozessor verwendet wird. Die Sonderzeichen und ihre Bedeutung sind:

? - Das variable Zeichen. Das Fragezeichen entspricht einem beliebigen einzelnem Zeichen.

* - Die variable Zeichenkette. Der Stern entspricht einer beliebigen Zeichenkette beliebiger Laenge, einschliesslich Zeichenketten der Laenge Null.

[...] - Das eingeschraenkt variable Zeichen. Die drei Punkte entsprechen einem Satz von Zeichen, die eine Zeichenklasse festlegen. Die Zeichenklasse entspricht einem beliebigen einzelnen Zeichen, das der Klasse angehoert. Zeichenbereiche koennen angegeben werden, indem zwei Zeichen durch einen Bindestrich "-" getrennt werden. Alle Buchstaben koennen z.B. durch [A-Za-z] dargestellt werden.

Teilweise Uebereinstimmung von Feldern, die keine Zeichenketten sind, koennen durch den folgenden Satz von Ausdruecken konstruiert werden. Im folgenden beziehen sich f1 und f2 auf die vom Nutzer gelieferten Feldwerte.

> f1 - Alle Felder mit Werten, die groesser als der eingegebene Wert sind, werden zugeordnet.

< f1 - Alle Felder mit Werten, die kleiner als der eingegebene Wert sind, werden zugeordnet.

!f1 - Alle Felder, die nicht dem eingegebenen Wert ent-

sprechen, werden zugeordnet.

f1-f2 - Alle Felder, die mit den eingegebenen Werten ueber-einstimmen oder die zwischen den eingegebenen Werten liegen, werden zugeordnet.

!f1-f2 - Alle Felder, die echt ausserhalb des angegebenen Bereichs liegen, werden zugeordnet.

Eine beliebige Anzahl der auf dem Schirm befindlichen Felder kann in der oben beschriebenen Weise ausgefuellt werden. Dann werden alle Angaben zur Erzeugung einer Anfrage durch 'UND' miteinander verknuepft. Wurde eine Gruppe von Datensaetzen "gefunden", wird der erste Datensatz auf der Bildmaske, zusammen mit einem Prompter angezeigt. Der Prompter unten auf dem Bildschirm zeigt, welche Operationen gerade gueltig sind. In Abhaengigkeit davon, welches der aktuelle Modus ist (Abfrage-, Aenderungs- oder Loeschmodus) und ob der Bildmaske ein Reportformat zugeordnet ist oder nicht, sind 6 verschiedene Operationen moeglich. Hier die Erklaerung aller Moeglichkeiten.

[N]EXT, [P]REVIOUS, [M]ODIFY, [D]ELETE, [R]EPORT, [S]TOP

[N]EXT, [P]REVIOUS und [S]TOP werden fuer alle Modi angezeigt. [M]ODIFY und [D]ELETE werden nur im Aenderungs- bzw. Loeschmodus angezeigt. [R]EPORT wird im Abfragemodus angezeigt, vorausgesetzt, dass der ENTER-Bildmaske ein Formatierungsporgramm fuer Reporte zugeordnet ist.

Die verschiedenen Antwortmoeglichkeiten haben folgende Bedeutung:

n - Zeigt den naechsten Datensatz innerhalb der Menge der abgerufenen Datensaetze an. RETURN hat dieselbe Wirkung.

p - Zeigt den vorhergehenden Datensatz innerhalb der Menge der abgerufenen Datensaetze an.

m - Der Cursor wird in das erste Bildmaskenfeld gebracht, um eine Aenderung der Daten im aktuellen Datensatz zu ermoeglichen.

d - Loescht den aktuellen Datensatz aus der Datenbank.

r - Nimmt die aktuelle Menge der abgerufenen Datensaetze und formatiert ihn zur Ausgabe entsprechend des zugeordneten Reportskriptes. Einer Bildmaske koennen bis zu 8 verschiedene Report-Skripts zugeordnet sein. Die zur Verfuegung stehenden Reporte werden auf dem ENTER-'Report Options Screen' angezeigt, wo man den entsprechenden Report auswaehlen kann und bestimmen kann, wohin die Ausgabe erfolgen soll.

s - Die aktuelle Menge der abgerufenen Datensaetze wird

geloescht und die Bildmaske wird von Daten befreit, so dass eine neue Anfrage erfolgen kann. CTRL/U hat dieselbe Wirkung.

searched: nnnnnn

Zeigt, wieviele Datensaetze zur Beantwortung der Anfrage durchsucht werden.

selected: nnnnnn

Zeigt, wieviele Datensaetze ausgewaehlt wurden.

current: nnnnnn

Zeigt, welcher Datensatz innerhalb der ausgewaehlten Menge gerade auf dem Bildschirm angezeigt wird. Diese Zahl liegt zwischen 1 und der Anzahl der ausgewaehlten Datensaetze.

5.2.3 ENTER-Reporte

Wurde eine Menge von Datensaetzen durch Anfragen ueber Bildmasken unter ENTER (Abschnitt 5.2.2) ausgewaehlt, d.h., wird die entsprechende Bildmaske im Anfragemodus betrieben ([I]NQUIRE), kann man mit dem ENTER 'Report Options Screen' einen Report waehlen und angeben, wohin dessen Ausgabe erfolgen soll. Die Reporte, die zur Verfuegung stehen, und die Ausgabeziele werden mit 'ENTER Screen Registration' (Abschnitt 5.1) festgelegt. Verfuegt eine Bildmaske ueber zugeordnete Reporte, ist der 'Report Options Screen' eine Option dieser Bildmaske ([R]EPORT), wenn sie unter dem Anfragemodus betrieben wird.

Hier ein Beispiel fuer einen 'Report Options Screen' von ENTER:

```
[sart100]                    WDATA SYSTEM
[I]NQUIRE                 24 JUL 1986 - 15:25
                          Artikel Verwaltung

   REPORT              TO:     SCREEN PRINT FILE FILENAME

1.Artikelliste-art300                  [ ]

2.Artikelbestellungsliste-art320  [ ] [ ]


REPORT #: _
```

Die Prompter werden folgendermassen verwendet:

REPORT #
Jede der aufgelisteten Optionsnummern kann eingegeben

werden. Durch Druecken von RETURN geht man standard-maessig in das erste Datenelement des Reports. Von diesem Prompter aus kann man auch mit den Kommandos sh oder edit in die WEGA-Shell oder einen Texteditor gelangen. Diese Moeglichkeit kann genutzt werden, wenn man Formatierungsskripts mit einem Editor modifizieren muss oder wenn man sich eine Ausgabe ansehen moechte, die auf eine Datei umgeleitet wurde.

REPORT
Der ausfuehrliche Titel des Reports, der erzeugt wird, wenn diese Option gewaehlt wird. Diese Zeichenkette ist in der Bildmaske 'ENTER Screen Registration' unter TITLE aufgefuehrt.

TO
Die Ausgabe kann geleitet werden auf:

SCREEN	Bildschirm des Terminals
PRINT	Drucker
FILE	Eine Datei namens FILENAME

[] Wird an dieser Stelle x eingegeben, soll die Ausgabe des aktuellen Reports auf dieses Ziel erfolgen. Die Angabe aller zur Verfuegung stehenden Optionen ist moeglich.

Wurde die Option Ausgabe an eine Datei, FILE, gewaehlt, muss hinter diesem Prompter der Name der Ausgabedatei angegeben werden. Existiert dieser Dateiname bereits, erscheint die Meldung

File already exists, destroy it?

Wird auf diese Meldung mit Y geantwortet, wird die Datei ueberschrieben, wird mit N geantwortet, kann man einen neuen Dateinamen eingeben.

Nachdem man mit 'REPORT #:' einen gueltigen Report gewaehlt hat, wird die entsprechende Zeile invers dargestellt. Der Cursor geht zum Prompter mit dem ersten zulaessigen Ausgabeziel. Nachdem man alle gewuenschten Ausgabeziele markiert hat, kehrt man mittels CTRL/U oder RETURN zum ersten Ausgabeprompter zurueck. Es wird CTRL/U eingegeben und der Prompter zum Verarbeitungsmodus (siehe unten) wird angezeigt. Kann der Report nur auf ein gueltiges Ausgabeziel ausgegeben werden, wird diese Option automatisch angefordert und der Prompter mit dem Verarbeitungsmodus erscheint:

Proceed in [F]oreground,[B]ackground,[D]ebug or [C]ancel?

Mit diesem neuen Prompter wird gesteuert, wie der Report verarbeitet wird. Die Option [D]ebug erscheint nicht, wenn mit 'ENTER Screen Registration' der Debug-Modus ausgeschaltet wurde. [B]ackground ist nicht zulaessig, wenn SCREEN

als Ausgabeziel gewaehlt wurde. Folgende Moeglichkeiten gibt es also:

f - Der Report laeuft ab, waehrend der Nutzer wartet.

b - Der Report wird gestartet und der Nutzer kann waehrenddessen seine Arbeit fortsetzen (Report im Hintergrund).

d - Der Report laeuft im Vordergrund-Modus und die verwendeten Skripts, die Laufzeit-Ergebnisse und Fehlermeldungen werden angezeigt. Die Verwendung des Debug-Modes ist nuetzlich, da dadurch die verwendeten Skripts mit den tatsaechlichen Parameterwerten an der entsprechenden Stelle angezeigt werden.

c - Die Reportanfrage wird geloescht und zum Prompter 'Report #:' zurueckgegangen.

Waehrend der Reporterstellung tritt natuerlich eine Verzoegerung auf. Wenn ein Report mehr als einen Bildschirminhalt ausfuellt, kann man ihn durch Druecken von RETURN durchblaettern:

```
Display next page? [RETURN] continues, [n] terminates
```

Mit RETURN kann der naechste Bildschirm mit Daten angezeigt werden, mit n wird der Prozess beendet. Ist der Report abgeschlossen, erscheint der Prompter:

```
Complete. Please enter RETURN to continue ->->
```

Durch Eingabe von RETURN gelangen man wieder zur Report-Auswahl.

Weitere Meldungen des 'Report Options Screen' sind:

```
<EMPTY>
```

Die Ausgabedatei enthaelt keine Daten.

```
Unable to create output file, please reenter filename
```

Waehrend des Prozesses konnte die Datei mit dem angegebenen Dateinamen (FILENAME) nicht angelegt werden (Ursache ??).

5.3 Anpassung von ENTER

Von sich aus bietet ENTER Moeglichkeiten zur Editierung im Zusammenhang mit gueltigen Datenbank-Feldformaten und expliziten Beziehungen. Aber gelegentlich ist es erforderlich, eine spezielle Datenaufbereitung oder eine zusaetzliche Verarbeitung der Datenbank vorzunehmen. Beispielsweise: Aufbereitung von Wertebereichen, Vergleich von Daten mit einem Satz gueltiger Zeichenketten oder Werten, Vergleich mit anderen Datenbank-Feldern, Zuordnung von Stan-

dardwerten (z.B. fortlaufende Schluesselnummer oder aktuelles Datum) oder kurzer Abstecher in ein Bildmaskenfeld, das auf dem Inhalt eines anderen Datenbank-Feldes beruht. Man kann auch von Grund auf ein eigenes Bildmasken-Programm schreiben, indem man dazu das Host-Sprachinterface verwendet. In diesem Fall muesste man allerdings im Umgang mit dem Host-Sprachinterface geuebt sein, und selbst dann koennte wahrscheinlich die entstehende Bildmaske nicht alle Moeglichkeiten von Anfragen ueber Bildmasken erfuellen, die ENTER erfuellen kann.

Aus diesem Grund beinhaltet WEGA-DATA das ENTER-Archiv in der Datei lib/enter.a. Diese Version von ENTER enthaelt verschiedene Host-Sprach-"Verbindungen" die man verwenden kann, um in einer Version von ENTER, Editierungs- und Verarbeitungsfunktionen einzubeziehen, die fuer den jeweiligen Anwendungsfall zutreffen. Damit wird die Entwicklung einer individuellen Bildmaske erleichtert - man muss das Host-Sprachinterface nicht so genau kennen und die grundsaetzliche Programmlogik ist bereits implementiert.

Eine solche fuer den individuellen Anwendungsfall zugeschnittene Version von ENTER kann, im Rahmen des Kodelimits des verwendeten Rechners, die spezielle Verarbeitung vieler verschiedener Bildmasken umfassen und kann dennoch in ueblicher Weise in Verbindung mit allen anderen ENTER-Bildmasken benutzt werden. Reicht der Platz dann immer noch nicht fuer alle Bildmasken aus, kann man sogar mehrere ausfuehrbare ENTER-Dateien, allerdings mit verschiedenen Namen, erstellen. In diesem Abschnitt soll eine theoretische Erklaerung der Host-Sprach-Verbindungen gegeben werden und es sollen Beispiele fuer individuell gestaltete ENTER-Bildmasken angefuehrt werden und es wird erlaeutert, wie man diese erstellen kann.

5.3.1 Arbeitsprinzipien

Es gibt fuenf verschiedene Arten der Nutzerfunktionen, die in eine nutzerspezifische Version von ENTER einbezogen werden koennen. Es handelt sich um folgende Funktionen: Bildmasken-Initialisierungsfunktion, Bildmaskenfeld-Vorverarbeitungsfunktion, Eingabefunktion, Nachverarbeitungsfunktionen und Bildmasken-Abschlussfunktion. Diese Funktionen koennen sich auf globale Variable beziehen, die den aktuellen Bildmaskennamen und den Verarbeitungsmodus (Hinzufuegen, Abfragen, Aendern oder Loeschen) von ENTER angeben.

Vom Nutzer definierte Tabellen teilen ENTER mit, welche Funktionen fuer die jeweiligen nutzerspezifischen Bildmasken zu verwenden sind. Bildmasken, die nicht in der Tabelle erscheinen, werden nach Standard verarbeitet. Man kann, innerhalb des Kode-Limits der verwendeten Hardware, fuer beliebig viele verschiedene Bildmasken den Kode mit einbeziehen.

Die Funktionen zum Initialisieren und Beenden der Arbeit mit der Bildmaske werden von der ENTER-Hauptroutine aufgerufen, waehrend die Funktionen Pre- und Postprocessing von der Bildmaskenfeld-Eingaberoutine von ENTER aufgerufen werden. Diese Routine heisst e_inbuf und erhaelt im ADD- und MODIFY-Modus alle Bildmaskenfeld-Datenelemente vom Nutzer. Die Nutzer Funktions-Verbindungen sind in diese Funktion eingebaut, die im wesentlichen eine spezielle Version von inbuf ist, die zum Aufruf von Nutzerfunktionen modifiziert wurde. Befindet sich ENTER im Modus 'Anfragen ueber Bildmasken', der zum Erhalt der zu bearbeitenden Menge von Datensaetzen zu Beginn der Modi MODIFY, INQUIRE und DELETE verwendet wird, kann man nicht in die Arbeit eingreifen.

Hohlt e_inbuf vom Nutzer ein Bildmaskenfeld-Datenelement, gestattet diese Routine dem Nutzer an drei Punkten, eine Funktion auszufuehren. Diese Punkte sind: bevor der Cursor sich zum Erhalt der Eingabe bewegt (Preprocessing-Funktion), waehrend sich der Cursor bei der Eingabe bewegt (Eingabe-Funktion) und nach erfolgter Eingabe (Postprocessing-Funktion). Wird eine dieser Funktionen angegeben, wird diese unabhaengig von der Reaktion des Nutzers immer aufgerufen. Die einzige Ausnahme waere, wenn die Preprocessing-Funktion einen Wert ungleich Null zurueckgibt. Dann wuerde eine sofortige Rueckkehr erfolgen. Man kann fuer alle Bildmaskenfelder die Funktionen Preprocessing, Eingabe und Postprocessing beliebig miteinander kombinieren.

Unter Verwendung von Initialisierungstabellen mit den Namen der Bildmasken, Bildmaskenfelder und Funktionen, wird angegeben, welche Funktion an welchem Punkt aufgerufen werden soll. Die Tabellendefinitionen sind in der Include-Datei edits.h. In dieser Datei sind auch externe Verweise auf Variable enthalten, die den aktuellen Betriebsmodus, scrmode, und den Namen der aktuellen Bildmaske, _savscr, angeben.

In der ersten Tabelle, SFTABLE, ist eine Liste der Bildmaskenfelder enthalten, denen Funktionen zugeordnet sind. Die Tabelle hat folgendes Format:

```
struct sftabel {
  int ssfld,                  /* the screen field number */
  (*prefunc)(),               /* pre processing function */
  (*inpfunc)().                        /* input function */
  (*postfunc)()             /* post processing function */
            };
#define SFTABLE struct sftable
```

Man kann beliebig viele Tabellen dieser Art haben, und sie koennen beliebige Namen haben, wobei je eine Tabelle und ein Name fuer jede auf angepasste Bildmaske verwendet wird. Jeder Tabelleneintrag entspricht einem Bildmaskenfeld und enthaelt die Nummer des Bildmaskenfeldes und die Adresse

der Preprocessing-, Input- und Postprocessing-Funktion.

Die zweite Tabelle, SCRTABLE informiert ENTER darueber, welche Tabelle SFTABLE fuer eine bestimmte Bildmaske zu verwenden ist. Diese Tabelle hat folgendes Aussehen:

```
struct scrtable {
  char *sname;                    /* the screen form name */
  int (*inifunc)(),            /* initialization function */
  (*trmfunc)();                   /* termination function */
  SFTABLE *scrdf;                 /* processing functions */
                };
#define SCRTABLE struct scrtable
```

Diese Tabelle gibt es nur einmal und sie muss mit dem Namen screentab definiert werden. ENTER wird mit externer Bezugnahme auf diesen Namen kompiliert und die Tabelle bleibt undefiniert, wenn man beim Erstellen der Tabellen diesen Namen nicht verwendet. Jeder Tabelleneintrag enthaelt den Namen einer Bildmaske, die Adresse der Initialisierungs- und Beendigungs-Funktion der Maske und die Adresse der SFTABLE, die die Funktionen enthaelt, die diese Bildmaske nutzerspezifisch macht. Man muss den Bildmaskennamen und die SFTABLE-Adresse angeben, die Adressen der Funktionen kann man jedoch mit 0 angeben, wenn man sie nicht verwenden will. Der letzte Eintrag in dieser Tabelle muss an jedem freien Platz eine 0 haben.

Wenn ENTER beginnt, durchsucht es screentab nach nutzerspezifischen Funktionen fuer die aktuelle Bildmaske. Das geschieht, indem ENTER den Namen der aktuellen Bildmaske mit den in screentab verzeichneten Namen vergleicht. Ist der Name der aktuellen Bildmaske in der screentab enthalten, verwendet ENTER die angegebene SFTABLE, um festzustellen, welche Nutzer-Funktionen auszufuehren sind. Es folgen einige Versionen der ENTER Steuerroutine und e_inbuf in Pseudokode, durch die angegeben wird, wo und wie die Nutzer-Funktionen aufgerufen werden. Danach erfolgt eine Beschreibung der eingebauten Nutzer-Funktionen:

```
enter ()
{
  display screen ();
  if (user initialization function exists for this screen)
    user initialization function ();
  while (operational mode = priamd (1)) |= CTRL u)
  {
    switch (operational mode)
    {
    case ADD:
      add mode ();
    case MODIFY:
      modify mode ();
    case INQUIRE:
      inquire mode ();
    case DELETE:
      delete mode ();
    }
  }
  if (user termination function exists for this screen)
    user termination function ();
}


e_inbuf (screen field, buf)
{
  while (1)
  {
    if (preprocessing function exists for screen field)
    {
 error status = preprocessing function (screen field, buf);
      if (error status |= 0)
        return (error status);
    }
    if (input function exists for screen field)
      error status = input function (screen field, buf);
    else
  error status = system input function (screen field, buf);
    if (post processing function exists for screen field)
    {
if(postprocessing function(screenfield,buf,errorstatus)==0)
        return (error status);
    }
    else
      return (error status);
  }
}
```

INIFUNC

NAME
inifunc - Initialisierung der Bildmaske

SYNTAX
inifunc ()

BESCHREIBUNG
Diese Funktion wird von ENTER aufgerufen, nachdem die Bildmaske angezeigt wurde, aber noch bevor der Prompter [I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE angezeigt wird. Da beim Starten von WEGA-DATA ENTER nach Standard ausgefuehrt wird und da ENTER mit einer Pipe mit dem Menue-Handler verbunden ist, bleiben waehrend der Aufrufe alle vom Nutzer eingelesenen Tabellen oder aktuellen Datensaetze unveraendert. Das ist aber eigentlich nur dann von Bedeutung, wenn man unter Verwendung von sbrk(2) oder aehnlichen Systemaufrufen mehr Speicherplatz erhalten moechte. Das braucht nur einmal zu geschehen und nicht bei jedem Aufruf einer neuen Bildmaske. Wird die Nutzer-Version von ENTER anders bezeichnet, trifft dies nicht zu, da das Programm fuer jede Bildmaske, die sich von den anderen unterscheidet, ablaeuft.

Diese Funktion wird ueblicherweise zum Oeffnen oder Anlegen von temporaeren Dateien, von Zugriffsparameter-Datensaetzen in der Datenbank oder zum Stellen von Fragen des Nutzers verwendet. Der Name der spezifischen Initialiserungsfunktion wird fuer jede Bildmaske in der Tabelle SCRTABLE namens screentab abgelegt.

RUECKGABEWERT
Der Statuskode wird nicht von ENTER ueberprueft.

INPFUNC

NAME
inpfunc - Eingabeverarbeitung fuer ein Bildmasken-Feld

SYNTAX
```
inpfunc (sfeld, buf)
int sfeld;
char *buf;
```

BESCHREIBUNG
Wird diese Funktion angegeben, wird sie anstelle der Standard-ENTER-Eingaberoutine aufgerufen. Die dieser Funktion uebergebenen Parameter sind die Nummer des Bildmaskenfeldes und die Adresse des Puffers, in den die Daten abgelegt werden sollen. Ueblicherweise wird diese Funktion fuer die Zuweisung von Standardwerten fuer Schluesselfelder (z.B. eine laufende Nummer) verwendet, wenn der Nutzer RETURN eingibt. Oder sie wird verwendet, um das Eingabefeld zu maskieren (z.B. bei einer Telefonnummer das Setzen von runden Klammern um die Vorwahlnummer und das Setzen eines Bindestriches innerhalb der Nummer). Der Name jeder spezifischen Funktion wird in eine SFTABLE eingetragen, die vom Nutzer erstellt wird.

Auch die Postprocessing-Funktion kann verwendet werden, um Standardwerte zuzuweisen, jedoch nicht fuer Felder, die keine Schluesselfelder sind, weil die Standard-ENTER-Eingaberoutine Schluesselfelder als Felder behandelt, in die eine Eingabe erfolgen muss, und weil ENTER dort RETURN nicht zulaesst.

RUECKGABEWERT
- 0 - in naechstes Feld gehen, Daten wurden eingegeben
- -2 - in vorhergehendes Feld zurueckgehen, es wurden keine Daten eingegeben
- -3 - in naechstes Feld uebergehen, es wurden keine Daten eingegeben

POSTFUNC

NAME
postfunc - Postprocessing fuer ein Bildmaskenfeld

SYNTAX
```
postfunc (sfeld, buf, ier)
int sfeld, ier;
char *buf;
```

BESCHREIBUNG
Wird diese Funktion benutzt, wird sie aufgerufen, nachdem das angegebene Bildmaskenfeld eingegeben wurde. Die Funktion wird immer aufgerufen, auch wenn der Nutzer nur RETURN oder CTRL/U eingibt. Die Parameter sind die Nummer des Bildmaskenfeldes, die Adresse des Puffers, der vom Nutzer eingegebene Daten enthaelt, und der von der Input-Funktion zurueckgegebene Statuskode. Ueblicherweise wird diese Funktion verwendet, um eine zusaetzliche Editierung des vom Nutzer eingegebenen Wertes vorzunehmen. Gibt diese Funktion 0 zurueck, kehrt e_inbuf mit dem Satuskode der Input-Funktion zu ENTER zurueck. Andernfalls beginnt e_inbuf wieder mit der Preprocessing-Funktion.

Man kann diese Funktion auch verwenden, um eine zusaetzliche Verarbeitung der Datenbank vorzunehmen, wie z.B. das Einfuegen oder Loeschen von Datensaetzen in andere(n) Dateien oder die Aktualisierung von anderen Datenbankfeldern oder Austritt in ein anderes Nutzerprogramm. Der Name jeder spezifischen Postprocessing-Funktion wird in einer vom Nutzer angegebenen SFTABLE-Struktur eingetragen.

RUECKGABEWERT
0 - es kann fortgesetzt werden
<>0 - Der Eingabeprozess fuer dieses Bildmasken-Feld ist noch einmal von vorn zu beginnen.

PREFUNC

NAME
prefunc - Preprocessing fuer ein Bildmasken-Feld

SYNTAX
```
prefunc (sfeld, buf)
int sfeld;
char *buf;
```

BESCHREIBUNG
Wurde diese Funktion angegeben, wird sie von ENTER aufgerufen, bevor das angegebene Bildmaskenfeld eingegeben wird. Die Funktion wird immer aufgerufen, bevor das Bildmaskenfeld eingegeben wird. Die Parameter sind die Nummer des Bildmaskenfeldes und die Adresse eines Puffers, in dem man Daten ablegen kann. Ueblicherweise kann diese Funktion verwendet werden, wenn man auf der Grundlage beliebiger Kriterien die Eingabe des aktuellen Feldes ganz ueberspringen will, oder wenn man ein Feld berechnen und anzeigen will. Der Name jeder spezifischen Preprocessing-Funktion wird in eine vom Nutzer erstellte SFTABLE-Struktur eingetragen.

RUECKGABEWERT
0 - weiter mit Eingabe-Verarbeitung
-2 - es wird ins vorhergehende Feld zurueckgegangen, ohne dass das aktuelle Feld eingegeben wird
-3 - es wird ins naechste Feld gegangen, ohne dass das aktuelle Feld eingegeben wird

TRMFUNC

NAME
trmfunc - Abschluss der Arbeit mit der Bildmaske

SYNTAX
```
trmfunc ()
```

BESCHREIBUNG
Diese Funktion wird von ENTER aufgerufen, wenn der Nutzer hinter dem Prompter [I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE ein CTRL/U eingegeben hat, oder nachdem der Nutzer die ENTER-Bildmaske durch Eingabe von CTRL/X "abgebrochen" hat. Ueblicherweise wird diese Funktion verwendet, um temporaere Dateien abzuschliessen oder ein Link aufzuheben, um eine zusaetzlicher Editierung vorzunehmen oder die Datenbank zu verarbeiten. Der Name der spezifischen Funktion wird fuer jede Bildmaske in der Tabelle SCRTABLE namens screentab abgelegt.

RUECKGABEWERT
Der Statuskode wird nicht von ENTER ueberprueft.

5.3.2 Beispiele fuer angepasste ENTER-Programme

In diesem Abschnitt soll mit einigen Beispielen gezeigt werden, wie die in den vorhergehenden Abschnitten beschriebenen nutzerspezifischen Funktionen verwendet werden koennen. Der Quellkode fuer diese Beispiele und die sich ergebende Nutzerversion von ENTER ist auf den WEGA-DATA-Auslieferungsdisketten enthalten.

Es soll das im Nutzerhandbuch angefuehrte Schema mit Herstellern, Modellen, Artikeln, Kunden und Bestellungen verwendet werden.

RECORD/FIELD	REF	TYPE	LEN	LONG NAME
her	10			hersteller
*henr		NUMERIC	4	nummer
hename		STRING	35	name
hestr		STRING	30	strasse
hort		STRING	20	ort
heplz		NUMERIC	4	postleitzahl
hetelex		NUMERIC	7	telex
modell	50			modell
*mosch		COMB		mod_schluessel
monr		NUMERIC	7	mod_nummer
mohenr	henr	NUMERIC	4	hersteller_num
mobez		STRING	30	bezeichnung
art	100			artikel
*sernr		NUMERIC	9	seriennummer
artmod	mosch	COMB		obermodell
arterda		DATE		erwerbsdatum
artgros		AMOUNT	5	grosshand_preis
artbest	benr	NUMERIC	9	bestellnummer
arinprei		AMOUNT	5	ind_abgabepreis
kunde	10			kunde
*kunr		NUMERIC	5	kundennummer
kuname		STRING	30	name
kustr		STRING	30	strasse
kort		STRING	20	ort
kuplz		NUMERIC	4	postleitzahl
kutelex		NUMERIC	7	telex
kuruf		NUMERIC	7	ruf
best	100			bestellung
*benr		NUMERIC	9	best_nummer
bedat		DATE		best_datum
bekun	kunr	NUMERIC	5	kundennummer

Angenommen, es sollen zwei ENTER-Bildmasken nutzerspezifisch verwendet werden, einmal die ENTER-Bildmaske fuer die Kunden-Datensaetze und zum anderen die fuer die Artikel-Datensaetze. Die Grund-Bildmasken, die verwendet werden, wurden unter Verwendung der in 'SFORM Menu' aufgefuehrten

Option 'Create Default Screen Form' generiert (siehe Abschnitt 4.8). Die zwei Masken sehen folgendermassen aus:

```
[kunde]                          WDATA SYSTEM
                              24 JUL 1986 - 15:25
                               Kunde Maintenance

kundennummer:
name        :
strasse     :
ort         :
postleitzahl:
telex       :
ruf         :

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

```
[sfmaint]                        WDATA SYSTEM
[I]NQUIRE                     24 JUL 1986 - 15:25
                                 Screen Entry

SCREEN: kunde

LN SFIELD   DFIELD  TP LEN FX FY PROMPT             PX PY

 1 kunde01  kunr     N 5   15 4  kundennummer:      1  4
 2 kunde02  kuname   S 30  15 5  name         :      1  5
 3 kunde03  kustr    S 30  15 6  strasse      :      1  6
 4 kunde04  kort     S 20  15 7  ort          :      1  7
 5 kunde05  kuplz    N 4   15 8  postleitzahl:      1  8
 6 kunde06  kutelex  N 7   15 9  telex        :      1  9
 7 kunde07  kuruf    N 7   15 10 ruf          :      1  10

 ->-> _
```

```
[art]                            WDATA SYSTEM
                              24 JUL 1986 - 15:25
                              Artikel Maintenance

seriennummer   :
artmod mohenr  :
artmod monr    :
erwerbsdatum   :
grosshand preis:

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

```
[sfmaint]                    WDATA SYSTEM
[I]NQUIRE                 24 JUL 1986 - 15:25
                             Screen Entry

SCREEN: art

LN SFIELD   DFIELD   TP LEN FX FY PROMPT              PX PY

 1 art01    sernr    N 9   18 4  seriennummer   :    1  4
 2 art02    artmod_m N 4   18 5  artmod mohenr  :    1  5
 3 art03    artmod_m N 7   18 6  artmod monr    :    1  6
 4 art04    arterda  D     18 7  erwerbsdatum   :    1  7
 5 art05    artgros  A 5   18 8  grosshand preis:    1  8

 ->-> _
```

5.3.2.1 Angepasste ENTER-Funktionen

In diesem Abschnitt werden fuer unsere zwei als Beispiel verwendeten Bildmasken die nutzerspezifischen Funktionen beschrieben und es wird auf einige Details hingewiesen, die nicht sofort auf der Hand liegen. Fuer die Kunde-Bildmaske soll der naechste sequentielle Schluesselwert zugeordnet werden, wenn der Nutzer im Ergaenzungs-Modus (ADD) nur RETURN fuer den Schluesselwert eingibt. Ausserdem soll in den Modi ADD und MODIFY die Postleitzahl auf ihre Gueltigkeit geprueft werden. Dazu ist fuer kunde01 eine Input-Funktion und fuer kunde05 eine Postprocessing-Funktion erforderlich.

Auch fuer die Artikel-Bildmaske soll der naechste Schluesselwert zugeordnet werden, wenn der Nutzer nur RETURN eingibt. Das erfordert fuer das Bildmaskenfeld art01 eine Input-Funktion. Fuer das Erwerbsdatum soll im Ergaenzungsmodus (ADD) als Standardwert das Tagesdatum eingetragen werden, falls der Nutzer nicht selbst das Datum eingibt. Ausserdem soll das Datum so editiert werden, dass es kleiner oder gleich dem Sytemdatum ist, wenn der Nutzer es eingibt. Das erfordert fuer das Bildmaskenfeld art04 eine Postprocessing-Funktion.

Der erste Schritt bei der nutzerspezifischen Gestaltung der Bildmaske ist die Erstellung der Funktionstabellen, so dass ENTER weiss, welche Funktionen aufzurufen sind. Es sind zwei SFTABLE-Strukturen erforderlich - eine fuer die Bildmaske kunde und eine fuer die Bildmaske art. Dann benoetigen wir die Standardtabelle SCRTABLE screentab, um die Bildmasken ihrer zugehoerigen SFTABLE-Struktur zuzuordnen. Hier die erforderlichen Definitionen der Funktionstabellen:

```
#include "../../include/edits.h"
int gkunkey(), ed_state ();
#include "../../def/kunde.h"

SFTABLE kundedits[] = {
/* screen field */   /* pre */   /* input */   /* post */
kunde01,             0,          gkunkey,      0,
kunde05,             0,          0,            ed_state,
-1,                  0,          0,            0 };
int ed_date(),gartkey(),ed_intf(),cksal(),ed_pamt(),
                                                 initeds();
#include "../../def/item.h"

SFTABLE artedits[] = {
/* screen field */   /* pre */   /* input */   /* post */
art01,               0,          gartkey,      ed_intf,
art04,               0,          0,            ed_date,
-1,                  0,          0,            0 };

SCRTABLE screentab[] = {
/* screen field */   /* pre */   /* input */   /* post */
"kunde",             initeds,    0,            kundedits,
"art",               initeds,    0,            artedits,
0,                   0,          0,            0 };
```

Man kann beliebige Namen fuer die Funktionen und SFTABLE-Strukturen waehlen. Die einzige Bedingung ist, dass die SCRTABLE als screentab benannt wird.

Wir wollen uns nun einigen Editierungs- und Verarbeitungsfunktionen zuwenden: Die erste ist die Initialisierungsfunktion initeds. In diesem Fall wird dieselbe Initialisierungsfunktion fuer beide Bildmasken verwendet, man kann jedoch, falls man das wuenscht, auch fuer jede Maske eine andere Funktion verwenden. Diese Funktion braucht nichts weiter zu tun, als auf den Datensatz max zuzugreifen, da einige der anderen Editierungsfunktionen Felder aus diesem Datensatz verwenden muessen. Es folgt der Quelltext dieser Funktion:

```
$include "../../def/file.h"
initeds ()
{
  if (acckey (max, "A"))
  {
    prtmsg (0,23,"Kein Zugriff auf Parameter-Datensatz");
    prtmsg (0,23,"Abbruch !");
    exit ();
  }
}
```

Die naechste Funktion ist die Eingaberoutine, mit der man die Kundennummer erhaelt. Gibt der Nutzer eine Nummer ein, wird diese Nummer fuer den Schluessel verwendet. Wird dage-

gen ein RETURN eingegeben, ordnet die Funktion einen fortlaufenden Schluesselwert zu. Die naechste laufende Nummer wird aus dem Feld mkunlfd von max gewonnen. Die Zahl im Datensatz max ist der kleinste Schluesselwert, der zur Verfuegung steht. Bei mehreren Nutzern ist es moeglich, dass, wenn zwei gleichzeitig neue Kunden hinzufuegen wollen, sie die gleiche Kundennummer erhalten. In diesem Fall wird einer der beiden Nutzer bedient, waehrend die andere die Meldung 'Record already exists' erhaelt, da die Funktion addrec sichert, dass nicht gleichzeitig zwei verschiedene Datensaetze (des gleichen Datensatztyps) mit demselben Schluessel existieren. In diesem Fall muss der Nutzer einfach den Versuch wiederholen.

Die Verwendung der globalen Variablen scrmode ist zu beachten. Auf diese Variable wird extern verwiesen in edits.h, und sie gibt den aktuellen Betriebsmodus von ENTER an, d.h. ob dieser ADD, INQUIRE, MODIFY oder DELETE ist. In edits.h werden auch Symbole definiert, die man verwenden kann, um den aktuellen Wert von scrmode zu testen.
Es folgt der Quelltext dieser Funktion:

```
$include "../../def/file.h"
#include "../../def/art.h"
#include "../../../include/edits.h"
short gotckey = 0;
gkunkey (sfield, buf)
int sfield;
char *buf;
{
  int ier;
  long ltemp, lrec;
  if (scrmode |= ADD)
    return inbuf (sfield, buf);
  if ((ier = inbuf(sfield, buf)) == 0)
  {
    gotckey = 1;
    return (0);
  }
  else
  {
    if (ier == -3)
    {
      if (gotckey)
        return (-3);
      gfield (mkunlfd, &ltemp);
      loc (kunde, &lrec);
      while (acckey (kunde, &ltemp) == 0)
        ltemp++; /* Finden naechste verfuegbare Kundennr */
      setloc (kunde, lrec);
      pfield (mkunlfd, &ltemp);
      outbuf (sfield, &ltemp);
      *(long *)buf = ltemp;
      gotckey = 1;
      return (0);
    }
    else
    {
      gotckey = 0;                    /* Kode war CTRL/U */
      return (ier);
    }
  }
}
```

Ist Hinzufuegen nicht der aktuelle Betriebsmodus, gibt die Routine fuer das Bildmaskenfeld einfach den Wert von inbuf zurueck. Da wir aber wissen, dass wir im ADD-Modus sind, wird inbuf benutzt, um den Wert vom Nutzer zu erhalten. Gibt der Nutzer etwas ein, wird ein globales Flag gesetzt um anzuzeigen, dass der Kundenschluessel bestimmt wurde und es wird Status 0 zurueckgegeben. Wurde RETURN eingegeben (-3), muss der naechste laufende Schluessel generiert werden. Wurde der Schluesselwert bereits bestimmt, geben wir einfach -3 zurueck. Das geschieht, wenn der Nutzer auf der Bildmaske ganz nach oben geht, nachdem er den Datensatz hinzugefuegt hat, und wenn er dann noch einmal RETURN drueckt. In diesem Fall soll kein weiterer Schluesselwert

generiert werden.

Zur Generierung des Schluessels wird wir zuerst der Schluesselkandidat aus dem Feld mkunlfd des Datensatzes max geholt und dann wird die Adresse des aktuellen Datensatzes gespeichert, falls einer existiert. Das geschieht, weil ENTER annehmen koennte, dass es etwas aktuelles verarbeiten soll und gesichert werden muss, dass alles unveraendert bleibt. Dann wird versucht, auf mit diesem Wert Datensaetze zuzugreifen und dieser wird inkrementiert, bis ein Datensatz gefunden wird, der nicht existiert. Dann wird der alte Datensatz wieder zum aktuellen gemacht und der Schluesselwert im Datensatz max gespeichert.

Schliesslich wird der Schluesselwert auf dem Bildschirm angezeigt, der generierte Schluessel im Puffer gespeichert, als haette der Nutzer ihn eingegeben, das Flag wird gesetzt, dass anzeigt, dass der Schluessel bestimmt wurde und es wird ein 0 Status zurueckgegeben, was bedeutet, dass eine Eingabe erfolgte. ENTER legt den Wert im Puffer in den Record ab, so als haette der Nutzer ihn eingegeben.

Wenn der Nutzer CTRL/U (-2) eingegeben hat, muss das Flag, dass anzeigt , das ein Schluessel bestimmt wird geloescht und der Status an ENTER zurueckgeben werden. Dann loescht ENTER den Bildschirm und gestattet dem Nutzer das Hinzufuegen eines weiteren Datensatzes.

Die Pruefung der Gueltigkeit des Zustandskodes ist leichter. Die Postprocessing-Funktion muss nur den vom Nutzer eingegebenen Wert betrachten und ihn mit der Tabelle der gueltigen Kodes vergleichen. In der in diesem Beispielprogramm verwendeten Tabelle sind nicht alle Kodes enthalten, aber sie reicht sicher zum Verstaendnis. Stimmt die Eingabe des Nutzers mit einem der gueltigen Werte ueberein, gibt die Routine einen Status 0 zurueck. Andernfalls zeigt sie eine Fehlermeldung an, zeigt erneut den aktuellen Wert des Feldes an und gibt den Status -1 zurueck. Das fuehrt dazu, dass e_inbuf zurueckgeht und erneut den Prompter anzeigt, mit dem der Nutzer zur Eingabe aufgefordert wird. Es ist zu beachten, dass noch immer das Feld editiert werden soll, wenn der Nutzer RETURN (Statuskode -3) eingegeben hat. Dazu muss das Feld aus der Datenbasis geholt werden, damit sein aktueller Wert festgestellt werden kann. Bei dieser Editierung wird ein Wert Null (leer oder Null) als richtig betrachtet. Es soll daran erinnert werden, dass STRING-Felder in der Datenbasis vollstaendig auf Null gesetzt werden, wenn der Datensatz hinzugefuegt wird.

```
$include "../../def/file.h"
char *statecds[] = ("  ","CA","OR","NY","HI","NH","WA",0);
ed_state (sfield, buf, ier)
int sfield, ier;
char *buf;
{
  char xstate[3];
  if (ier == -3)
  {
    cfill (0, xstate, 3);
    gfield (cstate, xstate);
    if (valstr (xstate, statecds) == 1 || xstate[0] == 0)
    return (0);
  }
  if (ier == -2)            /* kein Editieren bei CTRL/U */
    return (0);
  buf[2] = 0;                 /* Null beendet den Puffer */
  if (valstr (buf, statecds) == 1)
    return   (0);   /* gueltiger Zustandskode eingegeben */
  else
  {
    prtmsg (1, 23, "Falscher Zustandskode");
    output (sfield);   /* erneut aktuellen Wert anzeigen */
    return (-1);
  }
}
```

Die Funktion gartkey fuer die Bildmaske art fast identisch mit der fuer die Bildmaske kunde. Deshalb soll sie hier nicht besprochen werden. Die Routine zur Editierung von Daten ed_date, ist ein Beispiel dafuer, wie eine Post-processing-Funktion fuer die Zurordnung eines Standardwertes benutzt wird. Es folgt der Quellkode dieser Funktion:

```
$include "../../def/file.h"
#include "../../../include/edits.h"
ed_date (sfld, buf, ier)
int sfld, ier;
char *buf;
{
  short jday;
  if (ier |= 0)
    gfield (iad, &jday);
  else
    jday = *(short *)buf;
  if (ier == -3 && scrmode == ADD && jday == -32768)
  {
    jday = curdate ();
    outbuf (sfld, &jday);
    pfield (iad, &jday);
    return (0);
  }
  if (ier |= -2 && jday > curdate ())
  {
    prtmsg (1, 23, "heutiges oder frueheres Datum");
    output (sfld);
    return (-1);
  }
  return (0);
}
```

Gibt der Nutzer bei Verwendung dieser Funktion Daten ein, werden diese in einer temporaeren Variablen abgelegt, andernfalls wird der Wert aus der Datenbasis geholt. Hat der Nutzer einen RETURN eingegeben (Statuskode -3) und ist ENTER im ADD-Modus und ist das Datum Null (es soll daran erinnert werden, dass ein Nulldatum -32768 ist), wird das heutige Datum als Standardwert zugeordnet. Es ist zu beachten, dass dieses in der Datenbasis zu speichern ist, da der RETURN-Wert (Statuskode -3) an ENTER zurueckgegeben wird. ENTER speichert den Puffer nur in der Datenbasis, wenn es von der Eingaberoutine einen Status 0 zurueckbekommt.

Hat der Nutzer RETURN oder ein gueltiges Datum eingegeben, wird geprueft, ob dieses kleiner oder gleich dem heutigen Datum ist. Ist das nicht der Fall, wird eine Fehlermeldung ausgegeben, der aktuelle Wert des Datums wird erneut angezeigt und ein Status -1 zurueckgegeben. Das fuehrt dazu, dass erneut per Prompter nach dem Feld gefragt wird.

Der Nutzer kann den Bildschirm durch Druecken von CTRL/U loeschen, wenn er sich auf der Bildmaske im ersten Bildmaskenfeld, art01, befindet. Eine Postprocessing-Funktion kann fuer dieses Bildmaskenfeld benutzt werden, um ein letztes Mal auf Abhaengigkeiten zwischen den Feldern zu pruefen, wenn der Nutzer CTRL/U drueckt. Die Funktion wird ed_intf genannt und hat folgenden Quellkode:

```
$include "../../def/file.h"
#include "../../def/art.h"
extern short gotikey;
ed_intf (sfield, buf, ier)
int sfield, ier;
char *buf;
{
  long xipamt;
  if (ier == -2)
  {
    gotikey = 0;
  }
  return (0);
}
```

Diese Funktion ist ganz einfach. Es ist zu beachten, dass das Flag, das die Schluesselsuche anzeigt, dieser Routine geloescht wird und nicht von gartkey. Das ist deshalb der Fall, weil der Schluessel erneut angefordert werden kann, wenn die Postprocessing-Funktion einen Status ungleich Null liefert, und dann muss bekannt sein, dass der Schluessel bereits bestimmt wurde.

5.3.2.2 Laden von ENTER

In diesem Abschnitt soll erklaert werden, wie man Funktionen mit ENTER laedt, wenn sie erst einmal geschrieben wurden. Benennung der Funktionen siehe Abschnitt 5.3.1 und Beispiele fuer Funktionen siehe Abschnitt 5.3.2.

Auf den Installationsdisketten von WEGA-DATA ist auch eine Kommandodatei namens enter.ld. Sie befindet sich im Verzeichnis bin und enthaelt die Kommandos, die erforderlich sind, um unter Verwendung des ENTER-Archivs im Verzeichnis lib des Systems ENTER zu laden. Sie akzeptiert ein einziges Argument - den Pfadnamen des Archivs mit den nutzerspezifischen Funktionen.

Bevor man enter.ld ablaufen lassen kann, muss man die Umgebungsvariablen PATH und WDATA setzen, die auf die Verzeichnisse bin und lib des WEGA-DATA-Systems zeigen. Vollstaendige Informationen zu Umgebungsvariablen von WEGA-DATA siehe Abschnitt 1.1. Der naechste Schritt bestuende darin, die Funktionen zu kompilieren und in ein Archiv abzulegen. Um dies fuer das Uebungsbeispiel zu machen, wird das Verzeichnis auf beisp/src/sent geaendert und die Kommandos im Verzeichnis werden unter Verwendung des Kommandos ucc kompiliert:

```
ucc -c -O *.c
```

Dann wird das Archiv angelegt und die .o-Dateien entfernt unter Verwendung der folgenden Kommandos:

```
ar crv *.a *.o
```

```
rm *.o
```

Das Archiv sfedit.a existiert bereits in diesem Verzeichnis. Die Reihenfolge der Funktionen im Archiv ist sehr wichtig. Die Definitionen der Funktionstabellen muessen in der ersten .o Datei im Archiv sein, und in den Definitionen der Funktionstabelle muss Bezug genommen werden auf alle Funktionen im Archiv. Somit werden im Beispiel alle Funktionstabellen in der Quelldatei initeds.c definiert, und initeds.o ist die erste Funktion im Archiv.

Ist das Archiv nicht richtig geordnet, werden die Funktionen nicht richtig geladen und am Ende des Ladeprozesses treten undefinierte Symbole auf. Fuer das nutzerspezifische ENTER kann man versuchen, das Kommando

```
ar crv xxxx.a `lorder *.o | tsort`
```

zu benutzen, das im WEGA-Handbuch unter lorder (1) erklaert ist und das dem Ordnen des Archivs dient. Es hat sich aber herausgestellt, dass dieses Kommando nicht immer funktioniert. Entstehen noch immer undefinierte Symbole, muss man das Archiv manuell ordnen, indem man die .o Dateien, die die undefinierten Symbole enthalten, im Archiv an den Anfang bringt.

Nun wird das Verzeichnis auf ../../bin geaendert und durch Eingabe von

```
enter.ld ../src/sent/sfedit.a
```

wird ENTER geladen.

Damit entsteht die nutzerspezifische Version von ENTER im aktuellen Verzeichnis, dem lokalen Verzeichnis bin (von beisp). Startet nun der Menue-Handler eine ENTER-Bildmaske aus, startet er die nutzerspezifische Version im lokalen Verzeichnis. Wenn man will, kann man nun die kundenspezifische Version im Verzeichnis bin des WEGA-DATA-Systems installieren und sie kann von allen verwendet werden. In Abschnitt 5.3.2.3 wird beschrieben, wie man der nutzerspezifischen Version von ENTER einen anderen Namen geben und es ermoeglichen kann, dass sie aus den Menues heraus gewaehlt werden kann.

Moechte man einen anderen Verzeichnis- oder Archivnamen fuer die eigene Version von ENTER verwenden, braucht man nur enter.ld den korrekten Pfadnamen fuer das nutzerspezifische Funktionsarchiv zu geben. Man vewendet einfach den Pfadnamen, der der eigenen nutzerspezifischen Archivdatei entspricht. Wird der Pfadname zum Archiv nicht angegeben, erscheint die Meldung:

```
The customizing archive parameter is missing.
```

Existiert das Archiv nicht, oder kann es nicht gelesen

werden, erscheint die Meldung:

```
  enter.ld: Unable to read xxxx.
```

wobei xxxx der Pfadname des Archivs ist.

Es folgt der Quelltext der Shell-Kommandodatei enter.ld:

```
if test "$WDATA"; then
    if test "$1"; then
        if test -r $1; then
            uld ENTER \
                $WDATA/enter.a \
                $WDATA/fprg.a \
                $1
        else
            echo $0: Unable to read $1.
        fi
    else
        echo "The customizing archive parameter is missing"
    fi
else
    echo "The WDATA environment variable is not set."
fi
```

5.3.2.3 Mehrere ENTER-Programme

In diesem Abschnitt wird beschrieben, wie man mehrere Kopien der ENTER-Programme erhalten kann, wobei alle verschiedene Namen haben und wie man diese aus dem Menue-Handler waehlen kann. In Abschnitt 5.3.2.2 wird beschrieben, wie man diese ausfuehrbaren Dateien erstellen kann. Sie sind noetig, wenn man nicht nur ein einziges, da evtl. sehr grosses, ENTER-Programm haben moechte.

Die Loesung besteht darin, dass man das Standard-ENTER mit den Standard-Bildmasken verwendet und dann die nutzerspezifischen Bildmasken mehrere Saetze aufteilt. Dann laedt man die verschiedene Versionen von ENTER, eine Version pro Satz und gibt diesen unterschiedliche Namen. Damit erhaelt man mehrere kleinere ausfuehrbare Dateien, die jeweils eine nutzerspezifische Version von ENTER darstellen. Diese nutzerspezifischen ENTERs arbeiten genauso so, wie das Standard-ENTER. Mit der einen Ausnahme, dass sie nicht in einer Pipe laufen und daher immer explizit aufgerufen werden muessen, wenn sie ausgefuehrt werden sollen.

Sollen diese nutzerspezifischen ENTERs vom Menu-Handler aufgerufen werden, muessen sie registriert werden (siehe Abschnitt 2.1.1). Der Menue-Handler muss jedoch ENTER als einen Sonderfall behandeln, da ENTER seine Parameter nicht in der gleichen Weise interpretiert, wie "normale" ausfuehrbare Dateien und da das Standard-ENTER ueber eine Pipe mit dem Menue-Handler verbunden ist. Sollen also die nutzerspezifischen ENTERs ordentlich funktionieren, muss man

wissen, welche Parameter der Menuehandler an "normale" Programme uebergibt und mit welchen Parametern ENTER aufgerufen werden muss. Weiss man das erst einmal, kann man ein Shellskript schreiben, das die notwendige Uebersetzung vornimmt und das jeweilige nutzerspezifische ENTER korrekt aufruft. Dann wird das Shellskript im Menue-Handler registriert.

In Abschnitt 2 wurde ausfuehrlich erklaert, wie der Menue-Handler normalerweise eine ausfuehrbare Datei aufruft. Hier soll nur eine Zusammenfassung gegeben werden. Angenommen, man hat ein normales Executable namens EXEC, das mehrere Programme enthaelt und sysrecev verwendet. Der Menuehandler wuerde dann ein Programm mit diesem Executable starten, indem er folgende Zeile an die Shell uebergibt:

```
EXEC T index zugriffs-ebene sprach-kode bm-name
```

Die Parameter haben folgende Bedeutung:

Parameter	Verwendung	Verweis
T	Wird nicht verwendet ist nur traditionell vorhanden	Konstanter Wert
index	Einspungnummer in der ausfuehrbaren Datei des zu startenden Programms	Executable Maintenance Abschnitt 2.1.1
zugriffs-ebene	Ganze Zahl zwischen 0 und 15, die festlegt, ob der entspr. Nutzer Hinzufuegunge-, Abfrage-, Aenderungs- oder Loeschrechte hat.	Group Maintenance Abschnitt 2.1.3 oder Employee Maintenance Abschnitt 2.1.4
sprach-kode	Systemsprach-Kode	System Parameter Maintenance, Abschn. 2.1.7
bm-name	Name der diesem Programm zugeordneten Bildmaske	Executable Maintenance Abschnitt 2.1.1

ENTER verwendet alle diese Parameter. Der einzige Unterschied besteht darin, dass der Parameter index nicht als prgtab-Tabellenindex verwendet wird. Stattdessen ist dieser Parameter die Nummer des Arbeits-Datensatztyps fuer die Bildmaske, der ein Bindestrich (-) vorangestellt ist. Diese Nummer findet man in Abschnitt 3.1.3, Schema Listing, im Teil des Datensatzes mit der Ueberschrift 'Record List' in der Spalte mit der Ueberschrift NUMBER. Man kann die Datensatznummer auch in der Datei file.h finden. Dann braucht man nur noch den Parameter index in die richtige Nummer zu

uebersetzen und das ENTER-Programm aufrufen, das den nutzerspezifischen Kode fuer diese Bildmaske enthaelt.

Soll beispielsweise das Standard-ENTER fuer normale ENTER-Bildmasken beibehalten werden und getrennte ausfuehrbare Dateien fuer die in den vorhergehenden Abschnitten entwickelten nutzerspezifischen Bildmasken vorhanden sein, bestuende der erste Schritt in der Umbenennung des nutzerspezifischen ENTERs. Das koennte folgendermassen geschehen:

```
mv ENTER MENT
```

Der naechste Schritt bestuende darin, eine neue ausfuehrbare Datei im Menue-Handler zu registrieren, das zwei Programme enthaelt - eins fuer die nutzerspezifische Bildmaske kunde und eins fuer die nutzerspezifische Bildmaske art. Nach Registrierung dieser neuen ausfuehrbaren Datei sieht die Bildmaske 'Executable Maintenance' folgendermassen aus:

```
[execmnt]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                       Executable Maintenance

EXECUTABLE'S NAME: MENTER
USES SYSRECEV ? Y
PROGRAMS:
amd NAME      HEADING                  SCREEN    DIRECTORY

  0 kund      Kunde Maintenance        kunde
  1 arti      Artikel Maintenance      art
  3
  4
  5
  6
  7
  8
  9

[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

Es ist unbedingt erforderlich, hinter 'USES SYSRECEV?' ein Y einzugeben, da die Parameter sonst nicht uebergeben werden. Nun soll das Shell-Skript MENTER (Name der ausfuehrbaren Datei) geschrieben werden, das die Standardparameter vom Menue-Handler nimmt und den Parameter index von der prgtab-Tabellenindexnummer in die Arbeits-Datensatztypnummer fuer die entsprechende Bildmaske uebersetzt. Sieht man mit 'Schema Listing' nach, sieht man, dass kunde Datensatztyp 4 und art Datensatztyp 3 ist. Aus der Bildmaske 'Executable Maintenance' kann man entnehmen, dass der Index fuer kunde 0 ist und der Index fuer art 1. Damit kann das Shell-Skript geschrieben werden:

```
case $2 in
```

```
    0) MENT T -4 $3 $4 $5;;
    1) MENT T -3 $3 $4 $5;;
  esac
```

Nun muss nur noch der Modus der ausfuehrbaren Datei MENTER festgelegt werden und man ist fertig:

```
chmod +x MENTER
```

Man kann nun die Programme kund und arti in beliebige Menues bringen, oder sie direkt von der Menuezeile SELECTION: aus ausfuehren, indem man einfach ihre Namen eingibt. In gleicher Weise kann man mit einer beliebigen Anzahl verschiedener nutzerspezifischer ENTER verfahren.

Die einzige Einschraenkung bei diesen nutzerspezifischen ENTERs ist, dass man nach erfolgter Abfrage keine Reporte erstellen lassen kann. Da diese ENTERs nicht als ENTER-Bildmasken, sondern unter Verwendung von 'Executable Maintenance' registriert werden muessen, hat man keine Moeglichkeit, diesem Programm einen Report zuzuordnen.

6. DIE ABBILDUNGSORIENTIERTE SPRACHE SQL

In diesem Abschnitt sollen einige Eigenschaften der wichtigsten Anfragesprache - der Structured Query Language (SQL) - vorgestellt werden. SQL ist eine auf englische Schluesselworte orientierte Anfragesprache grosser Leistungsfaehigkeit und Dynamik. Vielfaeltige Versuche haben zur Entwicklung einer Sprache gefuehrt, die auch fuer Personen, die keine Programmierer sind, leicht zu erlernen ist und die dennoch auch den Anspruechen von Datenverarbeitungsfachleuten genuegt. In den in diesem Abschnitt angefuehrten Beispielen koennen nicht alle Moeglichkeiten, die die Sprache bietet, gezeigt werden, da die hier als Beispiel verwendete Datenbank eines Lagers nicht die Beschreibung aller Moeglichkeiten bietet.

6.1 Moeglichkeiten fuer Anfragen von SQL

Eine SQL-Anfrage besteht aus 'Klauseln", vor denen jeweils ein Schluesselwort steht. Diese Schluesselworte haben fuer SQL eine bestimmte Bedeutung und duerfen daher nicht als Namen fuer Datenbasen oder Felder verwendet werden. Schluesselworte werden, wenn sie im Nutzerhandbuch verwendet werden, fett gedruckt, z.B. select. Einige der Klauseln muessen und andere koennen verwendet werden. Hier die Klauseln, die verwendet werden muessen:

select eine Liste von Feldnamen

from eine Liste von Datensatztypen

Hier die Klauseln, die verwendet werden koennen:

where eine Bedingung (eine wahre/falsche Aussage)

group by eine Liste von Feldnamen

having eine Gruppenbedingung (eine wahre/falsche Aussage)

order by eine Liste von Feldnamen

into eine WEGA-Datei

Die bei der Auswahl und Berechnung zu verwendenen Datensatztypen und -felder werden identifiziert, indem man die regulaeren Datensatznamen und die langen Feldnamen benutzt. Die vollstaendigen Feldnamen muessen nicht wie die regulaeren Feldnamen innerhalb der gesamten Datenbank einmalig auftreten. Gibt man verschiedenen Feldern denselben langen Namen und moechte man beide Felder in ein und derselben Abfrage verwenden, muss man jeden vollstaendigen Feldnamen zusammen mit dem Namen des Datensatztyps angeben. Das wird in Abschnitt 6.1.11.1 demonstriert. Es folgen Regeln fuer gueltige lange Feldnamen (die im folgenden einfach als Feldnamen oder nur als Felder bezeichnet).

1. Feldnamen muessen mit einem Buchstaben beginnen und duerfen nur Buchstaben (sowohl Klein- als auch Grossbuchstaben sind zulaessig), Ziffern oder das Unterstreichungszeichen (_) enthalten.

2. Feldnamen duerfen keine Leerstellen enthalten. SQL koennte sonst nicht unterscheiden, ob es sich um zwei getrennte Felder oder um ein einziges Feld mit einer Leerstelle handelt.

3. Innerhalb eines Datensatzyps darf ein Feldname nur einmal auftreten.

SQL gestattet, dass Abfrage-Anweisungen in freiem Format eingegeben werden. Das heisst, dass man zusaetzliche Leerzeichen und RETURNs nach eigenem Gutduenken verwenden kann, um die Lesbarkeit einer Abfrage-Anweisung zu verbessern, ohne dabei ihre Bedeutung zu veraendern. Bei allen in den folgenden Abschnitten angefuehrten Abfragen wird diese Moeglichkeit genutzt.

Eine Abfrage wird durch einen Schraegstrich (/) beendet, danach beginnt SQL mit der Auswertung der eingegebenen Abfrage.

Wird eine gueltige Abfrage eingegeben, wird durch die Meldung

recognized query!

mitgeteilt, dass die Abfrage verarbeitet wird. Soll SQL beendet werden, wird end eingegeben, und man kehrt in den Menue-Handler zurueck.

Versteht SQL die Abfrage nicht, wird eine Fehlermeldung angezeigt, die auf die Art des Fehlers verweist. In Abschnitt 6.3.2 wird eine Zusammenfassung der Fehlermeldungen und Massnahmen zur Korrektur der Fehler angefuehrt. In den Abschnitten 6.3.1 und 6.3.2 sind Informationen darueber enthalten, wie man gespeicherte Abfragen editiert, erneut speichert und erneut startet.

In den folgenden Abschnitten werden die einzelnen Ausdruecke gezeigt, und moegliche Abfragen werden erlaeutert. Wegen des Umfangs und der Leistungsfaehigkeit der Sprache ist es nicht moeglich, alle Arten von Abfragen unter Verwendung von Beispielen zu zeigen. Aus diesem Grund wird die formale Grammatik von SQL in Abschnitt 6.3.3 dargelegt. In diesem Abschnitt sind alle gueltigen Abfragen verzeichnet.

6.1.1 Hilfsgrade

Die SQL-Sprache von WEGA-DATA bietet drei verschiedene Hilfsgrade, die das Erlernen und die Verwendung von SQL erleichtern. Hilfe wird gegeben fuer die allgemeine Syntax von SQL, fuer spezielle Schluesselworte und fuer die gueltigen Namen von Datensatztypen und Feldnamen der in Benutzung befindlichen Datenbank.

Fuer den ersten Hilfegrad wird einfach hinter dem SQL-Prompter help eingegeben. Daraufhin erscheint eine Liste der gueltigen Schluesselworte von SQL.

```
sql> help
```

Um Informationen ueber ein bestimmtes Schluesselwort zu erhalten, muss man help gefolgt von dem Namen dieses Schluesselworts eingeben. Moechten Sie z.B. mehr ueber select herausfinden, wird help select eingegeben:

```
sql> help select
```

Es erscheint eine Erklaerung, wie select zu benutzen ist.

Ausserdem kann man Informationen ueber die in der aktuellen Datenbank definierten Datensatztypen und Felder erhalten. Um eine Liste der gueltigen Datensatztypnamen zu erhalten, muss man

```
sql> records
```

eingeben. Es erscheint eine Liste der Datensatztypnamen, die man mit der Klausel from verwenden kann (Beispiel):

```
her      modell    art       kunde     best
```

Um die Feldnamen eines bestimmten Datensatztyps herauszufinden, wird in aehnlicher Weise fields und dann der Name des gewuenschten Datensatztyps eingegeben. Sollen z.B. die

Felder im Datensatztyp her aufgelistet werden, wird hinter dem SQL-Prompter

```
sql> fields her
```

eingegeben. Es erscheint eine Liste der Feldnamen fuer diesen Datensatztyp.

```
sql> fields her
NAME                                    TYPE    LENGTH

nummer                                  INTEGER 4
name                                    STRING  35
strasse                                 STRING  30
ort                                     STRING  20
postleitzahl                            INTEGER 4
telex                                   LONG    7
sql> _
```

Es ist zu beachten, dass die Datensatztypnamen die regulaeren Namen sind, waehrend die Feldnamen die langen Namen sind.

6.1.2 Auswahl von Datensaetzen - select-from

Die einfachste Art der SQL-Abfrage enthaelt sowohl eine Klausel select als auch eine Klausel from. Die select-Klausel listet die auszugebenden Felder auf, waehrend die from-Klausel angibt, aus welchem Datensatztyp (oder -typen) die Felder kommen sollen. Im folgenden Teil des Abschnitts 6 werden die Begriffe Datei und Datensatztyp synonym verwendet. Die ausgewaehlten Felder muessen in den hinter from angegebenen Datensatztypen enthalten sein.

BEISPIEL: Fuer jeden Datensatz in der Datei her sind alle Felder auszuwaehlen. Damit wird der gesamte Inhalt der Datei her in unserer Uebungsdatenbank aufgelistet. '*' ist eine Abkuerzung fuer alle (hier: alle Felder).

```
sql> select *
sql> from modell/
recognized query!

mod_nummer|hersteller_num|bezeichnung
-----------------------------------------------------------
      8700|           700|Melkhocker
      8701|           700|Schrankwand
     23000|           701|Rohrzange
     23010|           701|Heckenschere
     23020|           701|Wasserstrahlbiegezange
     23030|           701|Schraubendreher
     23040|           701|Ofenrohr
       880|           704|Buch
       881|           704|Zeitschrift
       882|           704|Dreckfuehler
```

```
         1|           709|Verteiler
         2|           709|Stromschleife
         3|           709|Sicherung
         4|           709|Stromkreis
         5|           709|Armleuchter
        64|           711|Taschenrechner
       800|           711|Grossrechner
      1715|           711|PC
      1000|           711|Weiche Ware
      8000|           711|Bigbug
         5|           712|Kugellager

mod_nummer|hersteller_num|bezeichnung
-----------------------------------------------------------
         8|           712|Matratze
sql> _
```

Sind die Tabellen, wie in diesem Fall, fuer einen Bildschirm zu lang, koennen Sie mit den Tastenkombinationen CTRL/S und CTRL/Q das Auflisten stoppen und weiterlaufen lassen.

Wird der Stern (*) bei der Auswahl benutzt, erscheinen die Felder in einer vom System bestimmten Reihenfolge. Fordert man die Felder explizit an, kann man die Reihenfolge selbst festlegen.

Oft ist ein Datensatztyp laenger (Summe der Laengen seiner Felder) als die Bildschirmbreite. Um die Ausgaben zwar unschoen, aber ueberhaupt vollstaendig lesen zu koennen, muss ihr Terminal die 'wrap around'-Option eingschaltet haben.

Man kann also nicht nur alle Felder auflisten, sondern es kann auch angegeben werden, welche speziellen Felder aufzulisten sind.

BEISPIEL: Fuer alle Hersteller sind Herstellernummer, Name und Strasse aufzulisten. Dabei wird die angegebene Reihenfolge verwendet.

```
sql> select nummer, name, strasse
sql> from her/
recognized query!

nummer|name                                  |strasse
-------------------------------------------------------------
   700|Kombinat "Moebel"                     |Holzweg 24
   701|PGH "Metall"                          |Eisengasse 8
   704|VEB "Druck"                           |Matersteg 33
   709|VEB "Elektro"                         |Am Strom 9
   711|Kombinat "Computer"                   |Sackgasse 1
   712|VEB "Waelzlager"                      |Am Ring 2
sql> _
```

6.1.3 Engere Auswahl - where

Da man kaum jemals den gesamten Inhalt einer Datei auflisten moechte, kann man die Klausel where verwenden, mit der man Auswahlkriterien angeben kann. Mit der where-Klausel kann man ein Feld mit einer Konstanten, einem Ausdruck oder den Ergebnissen einer anderen select-Klausel vergleichen. In Abschnitt 6.1.9 sollen diese verschachtelten Anfragen detaillierter beschrieben werden. Die where-Klausel kann auch einen komplexen Booleschen Ausdruck enthalten, der aus Auswahlkriterien besteht, die durch die Operatoren and und or miteinander verbunden sind. Durch die Verwendung eckiger Klammern kann die Prioritaet festgelegt werden. In den folgenden drei Abschnitten soll die Verwendung der Klausel where demonstriert werden.

6.1.3.1 Logische Ausdruecke und Operatoren

BEISPIEL: Fuer alle Hersteller, deren Herstellernummer groesser als 701 ist, sind Herstellernummer, Name und Strasse aufzulisten.

```
sql> select nummer, name, strasse
sql> from her
sql> where nummer > 701/
recognized query!

nummer|name                                |strasse
------------------------------------------------------------
   704|VEB "Druck"                         |Matersteg 33
   709|VEB "Elektro"                       |Am Strom 9
   711|Kombinat "Computer"                 |Sackgasse 1
   712|VEB "Waelzlager"                    |Am Ring 2
sql> _
```

Angenommen, es sollen alle VEBs gesucht werden. Dazu ist ein Vergleich zwischen dem Feld name und der Zeichenkette VEB (Rest variabel) erforderlich. Die von WEGA-DATA-SQL verwendeten Zeichenketten koennen eine beliebige Kombination der Sonderzeichen *, ? und [] enthalten, die folgendermassen verwendet werden:

? - Das variable Zeichen. Das Fragezeichen entspricht einem beliebigen einzelnen Zeichen. Wenn man also alle Personen namens Meier finden will und man weiss nicht, ob sie 'Meier' oder 'Meyer' geschrieben werden, kann man angeben Me?er oder in diesem Fall sicher M??er.

* - Die variable Zeichenkette. Der Stern entspricht einer beliebigen Zeichenkette beliebiger Laenge, einschliesslich Zeichenketten der Laenge Null (auch als Nullzeichenketten bezeichnet).

[...] - Das eingeschraenkt variable Zeichen. Die drei Punkte entsprechen einem Satz von Zeichen, die eine

Zeichenklasse festlegen. Die Zeichenklasse entspricht einem beliebigen einzelnen Zeichen, das der Klasse angehoert. Zeichenbereiche koennen angegeben werden, indem zwei Zeichen durch einen Strich '-' getrennt werden. So koennten beispielsweise alle Grossbuchstaben durch die Klasse

[ABCDEFGHIJKLMNOPQRSTUVWXYZ]

angegeben werden oder aber einfacher als [A-Z]. Alle Buchstaben (Klein- und Grossbuchstaben zusammen) koennen als [a-zA-Z] dargestellt werden. Andere Klassen koennen in aehnlicher Weise aufgestellt werden.

Es ist zu beachten, dass man, sobald eine Zeichenkette nicht die Gesamtlaenge des Feldes erreicht, mit dem sie verglichen werden soll, an diese sowieso immer ein Stern anhaengen muss, um die zum Auffuellen des Feldes benutzten Blanks mit zu spezifizieren. Ausserdem werden Zeichenkettenkonstanten in Apostrophe (') eingeschlossen, um sie von Feldnamen zu unterscheiden.

BEISPIEL: Name und Ort der VEBs sind aufzulisten.

```
sql> select name, ort
sql> from her
sql> where name = 'VEB*'/
recognized query!

name                              |ort
----------------------------------------------------------
VEB "Druck"                       |Blattau
VEB "Elektro"                     |Schukow
VEB "Waelzlager"                  |Kugelberg
sql> _
```

BEISPIEL: Aus dem der Modelldatei sind alle Modelle auszuwaehlen, deren Bezeichnung mit A, B oder C anfaengt und die darueber hinaus hoechstens 7 Zeichen lang ist. Fuer diese Modelle sind Hersteller- und Modellnummer, sowie die Modellbezeichnung auszugeben.

```
sql> select mod_nummer, hersteller_num, bezeichnung
sql> from modell
sql> where bezeichnung = '[A-C]??????   *'/
recognized query!

mod_nummer|hersteller_num|bezeichnung
----------------------------------------------------------
       880|           704|Buch
      8000|           711|Bigbug
sql> _
```

Die where-Klausel kann zwei Felder miteinander vergleichen, wobei jeder der Standard-Vergleichsoperatoren verwendet werden kann: =, ^=, <, <=, > und >=.

BEISPIEL: Fuer die Artikel, bei denen die Modellnummer

kleiner als die Seriennummer ist, sind diese beiden Werte und die Bezeichnung auszugeben.

```
sql> select seriennummer, artmod_monr, artmod_mohenr
sql> from her
sql> where seriennummer >= artmod_monr/
recognized query!

seriennummer|artmod_monr|artmod_mohenr
-------------------------------------
           6|          1|         709
           7|          5|         709
          12|          8|         712
          13|          8|         712
          14|          8|         712
sql> _
```

Die Booleschen Standard-Operatoren and und or koennen verwendet werden, um einfache Vergleiche so zu verbinden, dass komplexe Ausdruecke entstehen. Die eckigen Klammern geben an, welcher Teil des Ausdrucks zuerst ausgewertet wird.

BEISPIEL: Es sind die Artikel auszuwaehlen, die am 2/18/86 oder spaeter, aber vor dem 7/24/86 in den Lagerbestand aufgenommen wurden oder die unter 10.00 M kosten. Fuer diese sind Seriennummer, Datum der Aufnahme in den Lagerbestand und Grosshandelspreis aufzulisten.

```
sql> select seriennummer, erwerbsdatum, grosshand_preis
sql> from art
sql> where [erwerbsdatum>=2/18/86 and erwerbsdatum<7/24/86]
sql>         or grosshandpreis < 10.0/
recognized query!

seriennummer|erwerbsdatum|grosshand_preis
----------------------------------------
           5|   02/15/85|           9.75
           4|   02/15/86|           9.75
          11|   02/18/86|       89000.00
          12|   07/23/86|          40.00
          13|   07/23/86|          40.00
          14|   07/23/86|          40.00
sql> _
```

where kann in verschiedener Weise zur Vereinfachung von Anfragen verwendet werden. So kann z.B. angegeben werden, dass eine Zahl, ein Datum oder ein Betrag zwischen zwei Werten liegen soll. Angenommen, es sollen Artikel ausgewaehlt werden, deren Grosshandelspreis zwischen drei und zehn Mark liegt. Bei den meisten Anfragesprachen muss die Frage folgendermassen formuliert sein:

```
  grosshand_preis >= 6.00 and grosshandpreis <= 16.00
```

SQL gestattet die Verwendung eines einfacheren, natuerlicheren Ausdrucks, wie das folgende Beispiel zeigt:

BEISPIEL: Es sind die Artikel aufzulisten, deren Grosshandelspreis zwischen 6 und 16 Mark liegt.

```
sql> select grosshand_preis
sql> from art
sql> where grosshand_preis between 6.00 and 16./
recognized query!

grosshandpreis
--------------
         11.88
          9.75
          9.75
         11.88
sql> _
```

Es ist zu beachten, dass die Grosshandelspreise mehrfach auftreten. Das kann man aendern (siehe 6.1.4).

6.1.3.2 Die Negation - not

Boolesche Ausdruecke koennen ganz oder teilweise negiert werden, um die Datensaetze auszuwaehlen, die einem speziellen Kriterium nicht entsprechen.

BEISPIEL: Es sind alle vom Hersteller 709 erzeugten Modelle aufzufuehren, deren Beschreibung nicht die Zeichenkette 'strom' enthaelt. Fuer diese sind Nummer und Beschreibung aufzulisten.

```
sql> select mod_nummer, bezeichnung
sql> from modell
sql> where bezeichnung ^= '*[Ss]trom*'
sql>    and hersteller_num = 709/
recognized query!

mod_nummer|bezeichnung
----------------------------------------
         5|Armleuchter
         3|Sicherung
         1|Verteiler
sql> _
```

BEISPIEL: Aus dem Modelldatensatz sind alle Modelle auszuwaehlen, die weder mit A, B oder C anfangen noch eine Modellnummer kleiner als 1000 haben. Fuer diese sind Herstellernummer, Modellnummer und Bezeichung aufzulisten.

```
sql> select mod_nummer, hersteller_num, bezeichnung
sql> from modell
sql> where not[bezeichnung = '[A-C]*' or mod_nummer <1000]/
recognized query!
```

```
mod_nummer|hersteller_num|bezeichnung
-----------------------------------------------------------
      8700|           700|Melkhocker
      8701|           700|Schrankwand
     23000|           701|Rohrzange
     23010|           701|Heckenschere
     23020|           701|Wasserstrahlbiegezange
     23030|           701|Schraubendreher
     23040|           701|Ofenrohr
      1715|           711|PC
      1000|           711|Weiche Ware
sql> _
```

6.1.3.3 Vergleich mit Wertemenge

In vielen Abfragen soll wahrscheinlich ein Feld nicht nur mit einem einzelnen Wert, sondern mit einer Liste von Werten verglichen werden. Beispielsweise sollen die Hersteller in Schukow und alle in Orten, die mit 'B' beginnen ausgewaehlt werden. Unter Verwendung der Standardoperatoren wird daraus eine Folge von Gleichheitszeichen, die mit or verbunden werden, wie z.B.:

```
  ort = 'Schukow*' or ort = '[B]*'
```

Das wird bei mehreren moeglichen Werten schnell umstaendlich. Zur Vereinfachung dieser Art von Abfragen liefert SQL die Moeglichkeit, diese Werte als Menge anzugeben.

BEISPIEL: Fuer die Hersteller mit Sitz in den oben genannten Orten, sind Nummer, Name und Ort aufzulisten.

```
sql> select nummer, name, ort
sql> from her
sql> where ort in <'Schukow*', '[B]*'> /
recognized query!

nummer|name                                |ort
-----------------------------------------------------------
   704|VEB "Druck"                         |Blattau
   709|VEB "Elektro"                       |Schukow
   711|Kombinat "Computer"                 |Bad Disko
sql> _
```

Diese Art der Notation kann auch verwendet werden, um Wertemengen und Bedingungen durch and zu verbinden. Das ist dann von Nutzen, wenn man auf der Grundlage einer Kombination von Feldern, z.B. alle Betriebe in Schukow mit der Nummer 709 und alle Betriebe in einem Ort, der mit 'B' beginnt und der Nummer 711. Unter Verwendung von Standardnotationen wuerde das folgendermassen ausgedrueckt werden:

```
  [ort = 'Schukow*' and nummer = 709]        or
                            [ort = '[B]*' and nummer = 711]
```

Auch diese Art der Anweisung kann einfacher ausgedrueckt werden.

BEISPIEL: Fuer die genannten Hersteller ist Nummer Name und Strasse aufzulisten. Die runden Klammern werden verwendet, um die Wertemenge zu einer Gruppe zusammenzufassen.

```
sql> select nummer, ort, strasse
sql> from her
sql> where <ort, nummer> is in (<'Schukow*', 709>,
sql>                            <'[B]*', 711 >) /
recognized query!

nummer|ort                 |strasse
------------------------------------------------------
   709|Schukow             |Am Strom 9
   711|Bad Disko           |Sackgasse 1
sql> _
```

Mengen von Konstanten (Werten) wie

```
  <'Schukow*', 709>
```

werden hier als Datentupel bezeichnet. Ein Datentupel ist einfach eine Gruppe von Konstanten , die Werte fuer spezielle Felder in einem Datensatz enthalten. Die Felder muessen gekennzeichnet sein, indem den Tupeln eine Feld-Spezifikationsliste vorangestellt wird, wie das im Beispiel oben geschah. In Abschnitt 6.2.1 wird auf eine andere Verwendung von Datentupeln eingegangen - das Einfuegen von Datensaetzen in die Datenbank.

6.1.4 Einmaliges Auflisten - unique

Wenn eine Abfrage nicht aus einem der Datensatztypen ein Primaerschluessel-Feld auswaehlt, ist es moeglich, dass mehrere voellig identische Zeilen erscheinen. Das kann passieren, da nur an den Primaerschluessel die Forderung gestellt wird, dass er nur einmal vorkommen darf. So sind z.B. in der Datei modell mehrere Modelle von einem Hersteller, so dass eine Abfrage, die nur das Feld hersteller_num auswaehlte, doppelte Zeilen ergaebe. Manchmal sind solche Zeilen unerwuenscht. Der Operator unique gestattet es, doppelte Zeilen als Ergebnis einer Abfrage zu unterdruecken.

BEISPIEL: Auswahl der Herstellernummern fuer alle vorhandenen Modelle. Als Nebenwirkung bei Verwendung von unique, wird die Ausgabe sortiert.

```
sql> select unique hersteller_num
sql> from modell /
recognized query!
```

```
hersteller_num
--------------
           700
           701
           704
           709
           711
           712
sql> _
```

Jetzt noch zu einem schon bekannten, leicht modifizierten Beispiel.

BEISPIEL: Es sind die Artikel einmalig aufzulisten, deren Grosshandelspreis zwischen 6 und 16 Mark liegt.

```
sql> select unique grosshand_preis
sql> from art
sql> where grosshand_preis between 6.00 and 16./
recognized query!

grosshandpreis
--------------
         9.75
        11.88
sql> _
```

6.1.5 Arithmetische Ausdruecke

SQL gestattet zur Berechnung numerischer Werte die Verwendung der arithmetischen Standardoperatoren (+,-,* und /). Sowohl Konstanten als auch Felder koennen in arithmetischen Ausdruecken verwendet werden, die fuer Felder aller Typen mit Ausnahme von STRING und COMB zulaessig sind. Arithmetische Ausdruecke koennen immer dort verwendet werden, wo ein einfaches Feld in den select-, where- und having-Klauseln zulaessig ist.

BEISPIEL: Fuer die vom Hersteller Nummer 709 erzeugten Artikel sind Modellnummer, Grosshandelspreis, Industrieabgabepreis und die Differenz zwischen Grosshandelspreis und Industrieabgabepreis aufzulisten.

```
sql> select artmod_monr, grosshand_preis, ind_abgabepreis,
sql>           grosshand_preis - ind_abgabepreis
sql> from art
sql> where artmod_mohenr = 709/
recognized query!
                              grosshand_preis-ind_abgabepreis
artmod_monr|grosshand_preis|ind_abgabepreis|
-------------------------------------------
          5|          12.50|          11.25|           1.25
          1|          12.50|          11.25|           1.25
sql> _
```

Die DATE-Felder koennen wie Zahlen bei der Berechnung verwendet werden, da sie ganzzahlig gespeichert werden. Angenommen es sollen Artikel gesucht werden, die mindestens eine Woche vor einem gegebenen Datum in den Lagerbestand aufgenommen wurden. Dann koennte man folgende Anfrage verwenden:

BEISPIEL: Fuer die Artikel, die mindestens eine Woche vor dem 24. Februar 1986 eingegangen sind, sollen Seriennummer, Erwerbsdatum und das Datum eine Woche nach dem Erwerbsdatum aufgelistet werden.

```
sql> select seriennummer, erwerbsdatum, erwerbsdatum + 7
sql> from art
sql> where erwerbsdatum + 7 < 2/24/86   /
recognized query!

seriennummer|erwerbsdatum|erwerbsdatum+7
-------------------------------------------
           6|    02/15/86|      02/22/86
           5|    02/15/86|      02/22/86
           4|    02/15/86|      02/22/86
           2|    02/15/86|      02/22/86
           1|    02/15/86|      02/22/86
           7|    02/15/86|      02/22/86
           8|    02/16/86|      02/23/86
           9|    02/16/86|      02/23/86
          10|    02/16/86|      02/23/86
sql> _
```

Wenn der arithmetische Ausdruck kompliziert ist, koennen runde Klammern benutzt werden, um die Reihenfolge der Berechnung festzulegen. Ausdruecke in runden Klammern werden vor Ausdruecken ausserhalb der runden Klammern berechnet.

6.1.6 Ordnen der Ausgabe - 'order by'

Alle vorherigen Anfragen brachten ihre Ergebnisse in einer von SQL bestimmten Reihenfolge. Obwohl der unique-Operator seine Ausgaben sortiert, sind sie bis jetzt noch nicht in der Lage, die Ordnung der Ausgabe zu bestimmen. Mit order by kann man die Reihenfolge der aus der Anfrage sich ergebenden Zeilen explizit angeben. Standardmaessig wird in aufsteigender Reihenfolge sortiert, wobei die STRING-Felder in alphabetischer Reihenfolge von A bis Z sortiert werden.

BEISPIEL: Alle Modelle sind nach bezeichnung sortiert aufzulisten.

```
sql> select *
sql> from modell
sql> order by bezeichnung/
recognized query!
```

```
mod_nummer|hersteller_num|bezeichnung
-----------------------------------------------------------
         5|           709|Armleuchter
      8000|           711|Bigbug
       880|           704|Buch
       882|           704|Dreckfuehler
       800|           711|Grossrechner
     23010|           701|Heckenschere
         5|           712|Kugellager
         8|           712|Matratze
      8700|           700|Melkhocker
     23040|           701|Ofenrohr
      1715|           711|PC
     23000|           701|Rohrzange
      8701|           700|Schrankwand
     23030|           701|Schraubendreher
         3|           709|Sicherung
         4|           709|Stromkreis
         2|           709|Stromschleife
        64|           711|Taschenrechner
         1|           709|Verteiler
     23020|           701|Wasserstrahlbiegezange
      1000|           711|Weiche Ware

mod_nummer|hersteller_num|bezeichnung
-----------------------------------------------------------
       881|           704|Zeitschrift
sql> _
```

Anfrageergebnisse koennen nach mehr als einem Feld sortiert werden, und man kann angeben, dass einige Felder in aufsteigender und andere in absteigender Ordnung sortiert werden sollen.

BEISPIEL: Fuer Artikel, die zwischen dem 16.2.86 und dem 1.12.86 eingegangen sind, sind Erwerbsdatum und Grosshandelspreis aufzulisten. Diese sind so zu sortieren, dass mit dem juengsten Erwerbsdatum und mit dem niedrigsten Preis begonnen wird (Datum in fallender und Preis in aufsteigender Ordnung).

```
sql> select erwerbsdatum, grosshand_preis
sql> from art
sql> where erwerbsdatum between 2/16/86 and 12/1/86
sql> order by erwerbsdatum desc, grosshand_preis asc/
recognized query!

erwerbsdatum|grosshand_preis
-----------------------------
    07/23/86|          40.00
    07/23/86|          40.00
    07/23/86|          40.00
    02/18/86|       89000.00
    02/16/86|          16.20
    02/16/86|          16.20
    02/16/86|        4300.00
```

```
sql> _
```

6.1.7 Numerische Funktionen

SQL verfuegt ueber 5 integrierte Funktionen, mit denen man bei Anfrage eine Zusammenfassung von Werten erhalten kann. Diese Funktionen sind: count, sum, min, max und avg. Numerische Funktionen koennen in der select- und having-Klausel einer Anfrage verwendet werden.

Eine numerische Funktion bezieht sich nur auf eine Gruppe von Datensaetzen mit einem gemeinsamen Merkmal. Wird eine numerische Funktion in der Klausel select verwendet, muss man darauf achten, dass nur Felder ausgewaehlt werden, die fuer jedes Mitglied der Gruppe konstant bleiben. Es hat keinen Sinn, wenn man die Bezeichnungen der Modelle zusammen mit der Modellnummer auswaehlt, da jede Modellnummer einer anderen Bezeichnung entspricht.

Numerische Funktionen werden normalerweise im Zusammenhang mit der Klausel 'group by' verwendet. Mit dieser Klausel kann man explizit die ausgewaehlten Datensaetze in Gruppen zusammenfassen bzw. teilen, fuer die die angegebenen numerischen Funktionen berechnet werden. Wird 'group by' nicht explizit angegeben, wird davon ausgegangen, dass die numerischen Funktionen sich auf den gesamten Satz der ausgewaehlten Datensaetze beziehen.

BEISPIEL: Fuer alle im Lagerbestand befindlichen Artikel ist der Durchschnittspreis zu ermitteln.

```
sql> select avg(grosshand_preis)
sql> from art /
recognized query!

avg(grosshand_preis)
--------------------
             6681.81
sql> _
```

In einer Anfrage koennen verschiedene numerische Funktionen verwendet werden. In der naechsten Anfrage wird fuer alle Artikel der hoechste und der niedrigste Preis berechnet. (Wollte man an dieser Stelle die Modellnummern erhalten, die zu diesen Werten gehoeren, muesste man eine verschachtelte Anfrage verwenden).

```
sql> select max(grosshand_preis), min(grosshand_preis)
sql> from art/
recognized query!

max(grosshand_preis)|min(grosshand_preis)
----------------------------------------
             89000.00|                9.75
sql> _
```

Die count-Funktion gibt an, wieviele Datensaetze der Anfrage genuegten. Die Notation fuer Funktionen sollte immer dieselbe sein und deshalb wurde die Syntax von count so gewaehlt, dass sie mit den anderen numerischen Funktionen uebereinstimmt. (Mit anderen Worten handelt es sich also um ein Schluesselwort, dem eine in runde Klammern eingeschlossene Angabe folgt. Diese Angabe darf aber immer nur '*' sein.)

BEISPIEL: Die Gesamtanzahl der Hersteller ist zu berechnen.

```
sql> select count(*)
sql> from her /
recognized query!

 count(*)
----------
         6
sql> _
```

6.1.8 Einteilung der Datensaetze in Gruppen - 'group by'

Die 'group by'-Klausel dient der Berechnung numerischer Funktionen fuer Gruppen von Datensaetze mit gemeinsamen Merkmalen. Somit ist die Verwendung von group by ohne eine numerische Funktion sinnlos. group by fuehrt dazu, dass die ausgewaehlten Zeilen nach den angegebenen Feldern sortiert werden. Jede erkannte Gruppe von Datensaetzen fuehrt zu einer Unterbrechung. Dann wird die verlangte Funktion berechnet. Ein zweites Mal wird nach den von den Funktionen berechneten Werten sortiert. Diese Sortierung kann aber erst bei der Angabe von mehr als einem Feld bei group by zum Tragen kommen.

BEISPIEL: Fuer die von den verschiedenen Herstellern erzeugten Artikel sind die Herstellernummer und der durchschnittliche Grosshandelspreis aufzulisten.

```
sql> select artmod_mohenr, avg(grosshand_preis)
sql> from art
sql> group by artmod_mohenr/
recognized query!

artmod_mohenr|agv(grosshand_preis)
-------------------------------------
          700|              1087.41
          701|                 9.75
          704|                16.20
          709|                11.88
          711|             89000.00
          712|                40.00
sql> _
```

Da jede Zeile der Anfragenausgabe einen berechneten Wert

von einer Gruppe von Datensaetze darstellt, kann man nur Felder auflisten lassen, die fuer die gesamte Gruppe gemeinsame Werte haben. Somit ist z.B. die zusaetzliche Auflistung der Seriennummern fuer die Artikel in dieser Anfrage nicht moeglich.

In einer 'group by'-Klausel kann mehr als ein Feld verwendet werden, wodurch im Ergebnis der Anfrage mehr Unterbrechungen entstehen (eine nach dem ersten Feld erkannte Gruppe wird durch das naechste Feld noch weiter in Gruppen zerlegt). Ausserdem kann eine where-Klausel angegeben werden, wodurch zuerst alle Datensaetze ausgewaeht werden, die in die Berechnung einbezogen werden sollen.

BEISPIEL: Fuer alle Artikel, deren Verkaufspreis mehr als 10 Mark betraegt, sind nach Hersteller und Modellnummer aufgeschluesselt Herstellernummer, Modellnummer und Anzahl der Artikel aufzulisten.

```
sql> select artmod_mohenr, artmod_monr, count(*)
sql> from art
sql> where grosshand_preis > 10.0
sql> group by artmod_mohenr, artmod_monr/
recognized query!

artmod_mohenr|artmod_monr| count(*)
-------------------------------------
          700|       8700|        3
          700|       8701|        1
          704|        880|        2
          709|          1|        1
          709|          5|        1
          711|        800|        1
          712|          8|        3
sql> _
```

Man kann numerische Funktionen auch auf das Ergebnis anderer numerischer Funktionen anwenden. Damit kann man solche Werte wie Maximaldurchschnitt oder Maximalanzahl berechnen. Werden numerische Funktionen auf diese Weise benutzt, ist eine Klausel 'group by' zwingend erforderlich. Das Ergebnis wird folgendermassen berechnet. Zuerste werden alle in Frage kommenden Datensaetze unter Verwendung der Klausel where ausgewaehlt. Dann werden sie entsprechend der Felder in der Klausel 'group by' sortiert und die innere numerische Funktion wird berechnet. Dann wird die auessere Funktion auf diese Ergebnisse angewandt. Da diese zweite Ebene der Berechnung den Gruppen ihre Identitaet nimmt, ist eine verschachtelte Abfrage erforderlich, wenn Felder aufgelistet werden sollen, die keine Funktionsergebnisse sind.

BEISPIEL: Es ist die durchnittliche Anzahl der Modelle pro Hersteller zu berechnen.

```
sql> select avg(count(*))
sql> from modell
```

```
sql> group by hersteller_num /
recognized query!

  avg(count(*))
----------------
        3.66667
sql> _
```

BEISPIEL: Es ist der kleinste durchschnittliche Grosshandelspreis der im Lager vorhandenen Artikel einer Modellnummer zu berechnen.

```
sql> select min(avg(grosshand_preis))
sql> from art
sql> group by artmod_monr /
recognized query!

min(avg(grosshand_preis))
--------------------------
                    9.75
sql> _
```

6.1.9 Geschachtelte Anfragen

Durch verschachtelte Anfragen kann ein ganz neuer Satz von Fragen beantwortet werden, die unter Verwendung der bisher beschriebenen Moeglichkeiten von SQL nicht beantwortet werden koennten. Durch Verschachtelungen kann man die Ergebnisse einer Anfrage als Eingabe fuer eine andere verwenden, so dass die Ergebnisse einer Frage zur Beantwortung einer anderen verwendet werden koennen. Angenommen man moechte z.B. die Seriennummern der teuersten im Lagerbestand befindlichen Artikel wissen. Unter Verwendung der bisher beschriebenen Moeglichkeiten, kann man den maximalen Preis fuer einen Artikel folgendermassen suchen:

```
  select max(grosshand_preis)
  from art /
```

Oder man kann die Artikel suchen, die zu einem bestimmten Preis verkauft werden, z.B. alle Artikel zum Grosshandelspreis von 10 Mark.

```
  select seriennummer
  from art
  where grosshand_preis = 10.00/
```

Um jedoch herauszufinden, welche Artikel die teuersten sind, muss man zunaechst die erste Anfrage initiieren und dann den aus dieser Anfrage resultierenden maximalen Grosshandelspreis als konstanten Wert in der zweiten Anfrage verwenden. Durch die Verschachtelung geschieht dies automatisch.

BEISPIEL: Fuer die Artikel mit dem hoechsten Grosshandels-

preis sind Seriennummer, Kaufdatum und Grosshandelspreis aufzulisten.

```
sql> select seriennummer, erwerbsdatum, grosshand_preis
sql> from art
sql> where grosshand_preis = select max(grosshand_preis)
sql>                              from art/
recognized query!

seriennummer|erwerbsdatum|grosshand_preis
-----------------------------------------
          11|    02/18/86|       89000.00
sql> _
```

Tabulatoren werden genauso wie Blanks in den Eingabezeilen uebergangen. Das Ergebnis wird in zwei Schritten errechnet. Zuerst wird zur Ermittlung des maximalen Grosshandelspreises die innere Anfrage (select max(grosshand_preis) from art) abgearbeitet. Dann wird unter Verwendung der Ergebnisse der inneren Anfrage die aeussere Anfrage erledigt. Will man verschachtelte Anfragen verstehen, muss man mit der innersten Anfrage beginnen und dann schrittweise nach aussen gehen, da auch SQL in dieser Weise vorgeht.

Anfragen koennen nicht nur in einer Stufe, sondern in beliebig vielen verschachtelt sein. Angenommen, man sucht z.B. die Artikel, die den zweit hoechsten Grosshandelspreis haben. Unter Verwendung der zur Verfuegung stehenden SQL-Funktionen kann man zunaechst die Artikel finden, die den hoechsten Verkaufspreis haben und dann diese Artikel aus der folgenden Anfrage ausschliessen, durch die unter den verbleibenden Artikeln der Hoechstpreis bestimmt wird. Eine letzte Anfrage ermittelt dann die Artikel, die zu diesem Preis abgegeben werden.

BEISPIEL: Fuer die Artikel mit dem zweit hoechsten Grosshandelspreis sind Seriennummer, Erwerbsdatum und Grosshandelspreis aufzulisten.

```
sql> select seriennummer, erwerbsdatum, grosshand_preis
sql> from art
sql> where grosshand_preis =
sql>    select max(grosshand_preis)
sql>    from art
sql>    where seriennummer ^=
sql>            select seriennummer
sql>            from art
sql>            where grosshand_preis =
sql>                    select max(grosshand_preis)
sql>                    from art /
recognized query!

seriennummer|erwerbsdatum|grosshand_preis
-----------------------------------------
           8|    02/16/86|        4300.00
```

```
sql> _
```

Verschachtelte Abfragen koennen auch als Teil komplexerer Ausdruecke verwendet werden. In diesem Fall muss die innere Abfrage mit einem Semikolon (;) abgeschlossen werden, so dass SQL feststellen kann, wo die verschachtelte Abfrage endet und wo der Rest des Booleschen Ausdrucks beginnt.

6.1.10 Die having-Klausel

Mit der having-Klausel kann man einige der Gruppen auswaehlen, die durch eine vorangegangene 'group by'-Klausel gebildet wurden, und andere, auf der Grundlage der Ergebnisse von numerischen Funktionen, ausschliessen. Damit kann man dieselben Schritte ausfuehren, die unter Verwendung einer numerischen Funktion in einer where-Klausel moeglich waeren, was aber nicht zulaessig ist. So kann man beispielsweise die having-Klausel zur Auswahl von Modellen verwenden, deren durchschnittlicher Grosshandelspreis mehr als 12 Mark betraegt.

BEISPIEL: Fuer Modelle mit einem durchschnittlichen Grosshandelspreis von mehr als 12 Mark sind Modellnummer und durchschnittlicher Grosshandelspreis aufzulisten.

```
sql> select artmod_monr, avg(grosshand_preis)
sql> from art
sql> group by artmod_monr
sql> having avg(grosshand_preis) > 12.00/
recognized query!

artmod_monr|avg(grosshand_preis)
-----------------------------------
          1|              12.50
          5|              12.50
        800|              40.00
        880|           89000.00
       8700|              16.20
       8701|            4300.00
sql> _
```

Enthaelt eine Abfrage sowohl eine Klausel having als auch eine Klausel where, wird sie folgendermassen ausgewertet: Zuerst wird die where-Klausel angewandt, um die in Frage kommenden Datensaetze auzuwaehlen, dann werden die von der 'group by'-Klausel angegebenen Gruppen gebildet, dann wird die having-Klausel angewandt, um die in Frage kommenden Gruppen auszuwaehlen.

Angenommen, die Modelle mit einem durchschnittlichen Grosshandelspreis von mehr als 12 Mark sind ueber having gefunden worden, und nun sollen die einzelnen Artikel aufgelistet werden werden.

Die vorhergehende Anfrage koennte leicht so modifiziert

werden, dass nur die einzelnen Modellnummern zurueckgegeben werden und dass sie dann mit einer anderen Anfrage verschachtelt wird, die die anderen Informationen auflistet.

BEISPIEL: Fuer Artikel der Modelle, die im Durchschnitt mehr als 12 Mark kosten, sollen Seriennummer, Modellnummer, Grosshandelspreis und Erwerbsdatum aufgelistet werden. Die Ergebnisse sind nach der Modellnummer zu ordnen.

```
sql> select seriennummer, artmod_monr, grosshand_preis,
           erwerbsdatum
sql> from art
sql> where artmod_monr =
sql>    select artmod_monr
sql>    from art
sql>    group by artmod_monr
sql>    having avg(grosshand_preis) > 12.00;
sql> order by artmod_monr /
recognized query!

seriennummer|artmod_monr|grosshand_preis|erwerbsdatum
-----------------------------------------------------
           6|          1|          12.50|    02/15/86
           7|          5|          12.50|    02/15/86
          12|          8|          40.00|    07/23/86
          13|          8|          40.00|    07/23/86
          14|          8|          40.00|    07/23/86
          11|        800|       89000.00|    02/18/86
           9|        880|          16.20|    02/16/86
          10|        880|          16.20|    02/16/86
           1|       8700|          16.55|    02/15/86
           2|       8700|          16.55|    02/15/86
           3|       8700|          16.55|    02/15/86
           8|       8701|        4300.00|    02/16/86
sql> _
```

Es ist zu beachten, dass die innere Anfrage in diesem Beispiel durch ein Semikolon abgeschlossen wird. Die gesamte Anfrage ist mit oder ohne Semikolon gueltig. Wird es aber gesetzt, so wird dadurch SQL mitgeteilt, dass 'order by' zur inneren Anfrage gehoert und dass das Ergebnis der aeusseren Anfrage in der vom System bestimmten Reihenfolge ausgegeben wird.

Die having-Klausel kann auch im Zusammenhang mit einer where-Klausel verwendet werden. Die Anfrage wird in folgender Reihenfolge verarbeitet: Zuerst wird die where-Klausel zur Auswahl der in Frage kommenden Datensaetze verwendet, dann werden die durch die 'group by'-Klausel angegebenen Gruppen gebildet, dann wird die having-Klausel zur Auswahl der in Frage kommenden Gruppen angewandt.

BEISPIEL: Fuer Modelle, die mit mindestens zwei Artikeln vertreten sind, die mehr als 10 Mark kosten, sind die Modellnummer und die Artikelanzahl aufzulisten.

```
sql> select artmod_monr, count(*)
sql> from art
sql> where grosshand_preis > 10.00
sql> group by artmod_monr
sql> having count(*) >= 2/
recognized query!

artmod_monr| count(*)
-----------------------
          8|        3
        880|        2
       8700|        3
sql> _
```

Die Klausel having kann auch verschachtelte Abfragen enthalten. Eine in der Klausel having verschachtelte Abfrage wird in gleicher Weise bearbeitet wie eine in einer where-Klausel verschachtelte Abfrage. Weitere Informationen siehe Abschnitt 6.1.9. Mit dieser Art Abfrage kann man Fragen wie 'welcher Hersteller hat eine durchschnittliche Modellanzahl, die unter der durchschnittlichen Modellanzahl aller Hersteller liegt?' beantworten.

Verschachtelte Abfragen koennen in der Klausel where und in der having-Klausel gleichzeitig verwendet werden. Damit kann man die obige Anfrage noch dahingehend erweitern, dass letztendlich noch die Modelle dieser Hersteller ausgegeben werden.

6.1.11 Anfragen an mehrere Dateien - Verbund (Join)

Bisher wurde bei allen von uns vorgenommenen Anfragen nur ein einzelner Datensatztyp verarbeitet. In einer SQL-Anweisung koennen jedoch in einer einzigen Anfrage Felder einer beliebigen Anzahl von Datensatztypen aufgelistet werden. Anfragen, durch die Felder aus verschiedenen Datensatztypen aufgelistet werden, werden als Verbund (Join) bezeichnet, da sie verschiedene Datensatztypen kombinieren. Die verschiedenen Datensatztypen, die fuer die Anfrage in Frage kommen, werden in der from-Klausel in beliebiger Reihenfolge aufgelistet. Dann bestimmt SQL die effektivste Methode zur Auswahl.

6.1.11.1 Natuerlicher Verbund (General Join)

Zusammengesetzte Abfragen beruhen auf dem kartesischen Produkt (Relationen). Eine zusammengesetzte Abfrage bildet zunaechst das kartesische Produkt der Datensatztypen und "filtert" dann die Ergebnisse entsprechend den in der where-Klausel angefuehrten Bedingungen. Das heisst, dass ein Verbund ohne eine where-Klausel einfach das kartesische Produkt der Datensatztypen liefert.

Das kartesische Produkt kann am besten an einem Beispiel illustriert werden. Es sollen die drei unten aufgefuehrten

Datensatztypen a, b und c betrachtet werden. Typ c ist das kartesische Produkt von a und b. Es wird gebildet, indem jede moegliche Kombination bzw. Zusammenfassung der in den Datensatztypen a und b enthaltenen Datensaetze gebildet wird. Dabei ist die Reihenfolge ohne Belang.

Typ a	Typ b	Typ c
ppp	xxx	ppp xxx
qqq	yyy	ppp yyy
	zzz	ppp zzz
		qqq xxx
		qqq yyy
		qqq zzz

In diesem Fall enthaelt der Datensatztyp a zwei Datensaetze, der Datensatztyp b einen mehr und das kartesische Produkt, Typ c, enthaelt eine Datensatzanzahl gleich dem Produkt der Anzahlen von a und b (6). Haette man jedoch viele Datensaetze, deren Verbund durch Bildung des kartesischen Produkts erstellt werden wuerde, muesste dafuer ausserordentlich viel Zeit aufgewandt werden. Haette man beispielsweise 2 Datensatztypen, von denen der eine 5'000 und der andere 50'000 Datensaetze enthielte, haette das kartesische Produkt dieser zwei Typen 250'000'000 Datensaetze. Die Verarbeitung wuerde Tage in Anspruch nehmen!

WEGA-DATA-SQL kann jedoch bei Dateien dieser Groesse in Sekunden einen Verbund herstellen. Dazu wird jede Abfrage optimiert, um die zur Verfuegung stehenden Zugriffsmethoden von WEGA-DATA, einschliesslich Hash-Tabellen, expliziter Beziehungen und der B-Baeume, zu nutzen. Durch diese verschiedenen Zugriffsmethoden kann man die Leistung des Systems so flexibel gestalten, dass die Bildung des kartesischen Produkts gar nicht erforderlich ist.

Im ersten Anfragebeispiel dieses Abschnitts soll einfach ein kartesisches Produkt ohne Auswahl illustriert werden. Eine solche Abfrage wuerde in der Praxis wahrscheinlich niemals verwendet werden. Sie wird hier zur Erlaeuterung angefuehrt. Es ist zu beachten, dass, wenn ein Feldname nicht nur einmal vorkommt (bei mehreren Datensatztypen moeglich), diesem der Name des Datensatztyps voranzustellen ist, damit sicher ist, aus welchem Datensatztyp das Feld dieses Namens gehohlt wird.

BEISPIEL: Natuerlicher Verbund der Orte der Hersteller und Kunden.

```
sql> select her.ort, kunde.ort
sql> from her, kunde /
recognized query!
```

```
her.ort             |kunde.ort
-----------------------------------------
Waldhausen          |Erewhon
Waldhausen          |Irgendwo
Waldhausen          |Bad Berg
Schmiedeberg        |Erenwhon
Schmiedeberg        |Irgendwo
Schmiedeberg        |Bad Berg
Blattau             |Erewhon
Blattau             |Irgendwo
Blattau             |Bad Berg
Schukow             |Erewhon
Schukow             |Irgendwo
Schukow             |Bad Berg
Bad Disko           |Erewhon
Bad Disko           |Irgendwo
Bad Disko           |Bad Berg
Kugelberg           |Erewhon
Kugelberg           |Irgendwo
Kugelberg           |Bad Berg
sql> _
```

Die wirkliche Leistungsfaehigkeit eines Datenbankbetriebssystems zeigt sich, wenn es darum geht, die Ergebnisse eines Verbundes von Datensaetzen weiter zu "filtern". Das geschieht, indem zusaetzlich eine where-Klausel verwendet wird, die Zeilen des kartesischen Produktes auswaehlt.

Zu der folgenden Abfrage muessen einige Bemerkungen gemacht werden. Erstens brauchen hier (bis auf den folgenden Ausdruck) vor den Feldern nicht die Namen der Datensatztypen stehen, da sich in beiden Datensatztypen kein (langer) Feldname wiederholt. Zweitens wird der Ausdruck modell.* verwendet, um alle Felder im Datensatztyp modell auszuwaehlen, ohne dass jedes einzelne explizit aufgelistet werden muss. Versuchen Sie drittens, eine der beiden Seiten des 'UND'-Ausdrucks wegzulassen, erscheinen ungleich mehr Ausgabezeilen. Werden naemlich fuer einen im Verbund auftretenden Datensatztyp keine Einschraenkungen gemacht, so erscheint er unbedingt mit allen selektierten Feldern (Feldkombinationen) unterschiedlichen Inhalts in der Ausgabe und jede dieser Zeilen wird mit allen ausgewaehlten Zeilen des zweiten Datensatztyps verbunden.

BEISPIEL: Es sind der Name des Herstellers 712 und alle Felder seiner Modelle aufzulisten.

```
sql> select name, modell.*
sql> from her, modell
sql> where nummer = 712 and hersteller_num = 712 /
recognized query!

her.name            |mod_nummer|hersteller_num|bezeichnung
-------------------------------------------------------------
VEB "Waelzlager"    |         8|           712|Matratze
VEB "Waelzlager"    |         5|           712|Kugellager
```

```
sql> _
```

Die Klausel where kann in einem Verbund Ausdruecke enthalten, die Felder aus beliebigen Datensatztypen enthalten, die am Verbund beteiligt sind. Als Ausdruck finden natuerlich nicht nur Vergleiche Anwendung. Auch innerhalb eines Verbunds bleiben alle Moeglichkeiten der where-Klausel erhalten.

6.1.11.2 Gleich-Verbund (Self Join)

Manchmal macht es sich erforderlich, dass man einen Verbund zwischen einem Datensatztyp selbst erstellt. In unserer Beispiel-Datenbank ist solcher, "lohnender", Datensatztyp nicht enthalten. Deshalb soll kurz verbal ein Beispiel konstruiert werden. Also stellen wir uns vor, ein Hersteller-Datensatztyp enthaelt u.a. die Nummern seiner Kunden. Es ist durchaus denkbar, dass die Kunden der Hersteller selbst Hersteller anderer Produkte sind. Dann tritt der Primaerschluessel des "Kunden" bei dem Hersteller als Inhalt des Kundenfeldes auf. So koennen wir unter Benutzung der Hersteller- und der Kundennummer einen Verbund des Datensatztyps mit sich selbst kontruieren.

Abfragen wie diese stellt man sich am besten so vor, als handle es sich um zwei Kopien des Datensatztyps Hersteller - eine in der die Hersteller sind und eine in der die Kunden sind. SQL fertigt nicht wirklich eine Kopie an, sondern gestattet dem Nutzer stattdessen, einem Datensatztyp einen temporaeren Namen zu geben. Dann kann man diese Datensatztypen wie "echte" Datensatztypen verbinden. In der folgenden Abfrage werden der Datensatztyp herst und kaeuf (der lediglich ein temporaerer Name fuer den Datensatztyp herst ist) verwendet.

BEISPIEL: Fuer die Hersteller 0, 8 und 15 sind ihre eigenen Namen und Nummern und die ihrer Kunden aufzulisten.

```
sql> select herst.name,herst.nummer,kaeuf.name,kaeuf.nummer
sql> from herst, herst kaeuf
sql> where herst.nummer in <0, 8, 15 > and
sql>       herst.kaufnr = kaeuf.nummer /
```

Warum braucht man an dieser Stelle einen Verbund und kommt nicht mit einer einfachen Anfrage an den Datensatztyp herst aus? Auf den ersten Blick ist es gar nicht notwendig. Die Nummern der "Kunden" wuerde man auch ueber eine einfache Anfrage erhalten. Aber aus der Kundennummer wiederum den Namen des Kunden/Herstellers zu konstruieren, das leistet nur der Verbund.

Abschliessend sei bemerkt, dass man auch zwischen mehr als zwei "Datensatztypen" einen Gleich-Verbund konstruieren kann.

6.1.12 Variable Anfragen

Normale geschachtelte Anfragen werden von innen nach aussen aufgeloest, d.h. zuerst wird die innerste Anfrage bearbeitet und dann wird der sich ergebende Wert (Werte) an die naechste weiter aussen stehende Anfrage uebergeben. Jede Anfrage wird nur einmal ausgefuehrt. Bei variablen Anfragen wird dieses Verfahren umgekehrt, indem von aussen nach innen vorgegangen wird. Damit kann man unter Verwendung der sich aus der aeusseren Anfrage ergebenden Werte eine innere Anfrage mehrmals ausfuehren.

Variable Anfragen kann man in dem Sinne nicht definieren, somdern sie ergeben sich zwangslaeufg aus der Logik der gestellten Anfrage. Bei derartigen Anfragen handelt es sich um geschachtelte Anfragen, die mit einem Verbund definiert sind. Ganz grob kann man sagen, dass ein Verbund versucht das vollstaendige kartesische Produkt zu bilden. Jede zusaetzliche Anfrage vermindert die Anzahl der Verknuepfungen. Aber jede innere Anfrage laesst es sich nicht nehmen, ihre Ergebnisse mit jeder durch den Verbund entstandenen Zeile zu verknuepfen.

6.2 Moeglichkeiten der Datenmanipulation - SQL als DML

Neben den oben beschriebenen Anfragemoeglichkeiten bietet SQL auch Moeglichkeiten der Datenmanipulation. Das bedeutet, dass man unter Verwendung dieser hoeheren nichtprozeduralen Programmiersprache Datensaetze interaktiv in die Datenbank des Anwenders einfuegen, modifizieren und loeschen kann. Es existiert auch ein Interface zum WEGA-DATA-Programm 'Data Base Load' (s. Abschnitt 3.5.5), so dass man eine Datenbank schnell aus regulaeren ASCII-Dateien laden kann. Alle Anfrageeigenschaften der Sprache bleiben erhalten, so dass man regulaere Anfrageanforderungen verwenden kann, um Daten aus existierenden Dateien in andere Dateien einzufuegen und zu modifizierende oder zu loeschende Datensaetze auswaehlen kann.

Wird eine Gruppe von Datensaetzen angegeben, die modifiziert werden soll, wird diese Gruppe unter Verwendung des WEGA-Systemaufrufs zur Verriegelung (lkdata) von Dateien gesperrt. Damit wird verhindert, dass mehrere Nutzer gleichzeitig dieselben Datensaetze modifizieren.

6.2.1 Die Einfuegeklausel insert

Die 'insert into'-Klausel ist eine Anweisung in der Data Manipulation Language (DML), die das Einfuegen neuer Datensaetze in die Datenbank gestattet, wobei konstante Werte oder Werte verwendet werden, die als Ergebnis einer Anfrage zurueckgegeben werden. Nur der Wert des Primaerschluessels muss angegeben werden. Andere Feldwerte, die nicht angegeben werden, werden auf den Standard-Ausgangswert gesetzt

- Nullen fuer NUMERIC-, FLOAT-, AMOUNT-, TIME- und STRING-Felder und Null (-32768) fuer DATE-Felder.

Man kann auch einen Datensatz in die Datenbank einfuegen, indem man die Daten innerhalb des SQL-Kommandos auflistet.

BEISPIEL: Ein neuer Bestellerdatensatz ist in best einzufuegen, wobei 10 die Bestellnummer und 03/14/86 das Datum sein soll, an dem die Bestellung erfolgte. Die Kundennummer soll die 1 sein.

```
sql> insert into best (best_nummer,best_datum,kundennummer):
sql> < 10, 03/14/86, 1 > /
recognized update!
1 record(s) inserted
sql> _
```

Eine Liste von Feldern, die in eckige Klammern eingeschlossen ist, wird als ein Datentupel bezeichnet. Wenn die Felder in einem Datentupel in der gleichen Reihenfolge angefuehrt sind, wie im Datensatztyp und wenn alle Felder vorhanden sind, kann die identifizierende Feldliste ausgelassen werden. Es kann auch eine Liste von Datentupeln angegeben werden.

BEISPIEL: In die Datei modell sind drei neue Modelle einzufuegen.

```
sql> insert into modell:
sql> < 3000, 711, 'Schachcomputer'>,
sql> < 3400, 711, 'Geldautomat'>,
sql> < 680000, 711, '64-Bit-Computer'> /
recognized update!
3 record(s) inserted
sql> _
```

Ausser der Verwendung von Datentupeln kann die insert-Klausel die Ergebnisse einer Anfrage gewissermassen fixieren. Daten, die sonst auf dem Bildschirm erscheinen, koennen in Datensaetze umgeleitet werden.

Angenommen in der Datenbasis existiert ein Datensatztyp namens aktuell, der die Felder mod_nummer und bezeichnung enthaelt und es sollen alle im Datensatz modell fuer einen einzelnen Hersteller vorhandenen Informationen ausgewaehlt und in diesen neuen Datensatz gebracht werden.

BEISPIEL: Es sollen alle vom Kombinat "Moebel" hergestellten Modelle ausgewaehlt werden und der entsprechende Eintrag in aktuell vorgenommen werden. (Es ist zu beachten, dass diese Aktualisierung sich nicht tatsaechlich auf die vorhandene Datenbasis auswirkt.)

```
sql> insert into aktuell:
sql>    select mod_nummer, bezeichnung
sql>    from modell, her
```

```
sql>    where hersteller_num = nummer and
sql>          name = 'Kombinat "Moebel"*' /
aktuell is an invalid record type.
sql> _
```

Da der Datensatztyp aktuell nicht existiert, konnte diese Einfuegeoperation nicht erfolgreich sein. Wenn sie wollen, editieren Sie die letzte Anfrage (Kommando edit) und streichen Sie die erste Zeile (insert-Klausel). Dann erscheinen auf dem Bildschirm die Daten, die sonst unter dem Datensatztyp aktuell abgelegt worden waeren.

Der Stern in der letzten Zeile ist notwendige Syntax. Vollstaendige Syntaxinformationen ueber insert sind im Abschnitt 6.3.3.2 enthalten.

6.2.2 Die Aktualisierungsklausel update

Mit dem Schluesselwort update wird eine Klausel der Data Manipulation Language (DML) eingefuehrt, mit der man Felder in existierenden Datensaetzen modifizieren kann. Aktualisierungen koennen durch literale Werte, Ausdruecke oder Anfragen angegeben werden. Es ist nicht erforderlich, mit einer where-Klausel anzugeben, welche Datensaetze aktualisiert werden sollen. Wird sie nicht angegeben, werden alle Datensaetze des angegebenen Typs aktualisiert.

Wird ein Satz von Datensaetzen angegeben, der modifiziert werden soll, wird dieser Satz vermittels des WEGA-Systemaufrufs zum Verriegeln von Dateien verriegelt. Damit wird verhindert, dass mehrere Nutzer gleichzeitig dieselben Datensaetze modifizieren.

BEISPIEL: Zur Erinnerung die Verbesserung eines Fehlers beim Industrieabgabepreis, die an der Beispieldatenbasis im Nutzerhandbuch durchgefuehrt wurde. Melkhockers mit der Seriennummer 3 aufgetreten ist, soll verbessert werden. Der Preis des Melkhockers wurde von 14.58 M auf 14.85 M verbessert.

```
sql> update art
sql> set ind_abgabepreis = 14.85
sql> where seriennummer = 3 /
recognized update!
1 record(s) updated
sql> _
```

Ein Feld kann unter Verwendung eines Ausdrucks aktualisiert werden und mehr als ein Feld kann in einer einzigen Aktualisierungsklausel geaendert werden.

Ein Feld kann auch aktualisiert werden, indem man zur Berechnung des neuen Wertes eine Anfrageanforderung verwendet. Diese Abfrage kann alle in Abschnitt 6.1 erlaeuterten Moeglichkeiten nutzen.

Im Nutzerhandbuch befindet sich eine Anzahl derartiger Aktualisierungen. Die vollstaendige Beschreibung der Syntax fuer update ist im Abschnitt 6.3.3.2 angegeben.

6.2.3 Die Loeschklausel delete

Das Schluesselwort delete ist eine Klausel in der Data Manipulation Language (DML), mit der man Datensaetze aus einer existierenden Datei loeschen kann. Die zu loeschenden Datensaetze werden mit einer where-Klausel angegeben. Ist keine solche vorhanden, werden alle Datensaetze des angegebenen Typs geloescht.

BEISPIEL: Bestellnummer 10 soll aus best geloescht werden.

```
sql> delete best
sql> where best_nummer = 10 /
recognized update!
1 record(s) selected, 1 record(s) deleted
sql> _
```

BEISPIEL: Die drei im Abschnitt 6.2.1 in die Datenbasis eingefuegten Modelle 3000, 3400 und 680000 sind zu loeschen.

```
sql> delete modell
sql> where mod_nummer is in < 3000, 3400, 680000 > /
recognized update!
3 record(s) selected, 3 record(s) deleted
sql> _
```

Alle Moeglichkeiten, die die where-Klausel bietet, koennen zur Spezifizierung der zu loeschenden Datensaetze benutzt werden, siehe Abschnitte 6.1.3 und 6.1.9.

6.2.4 Auslagern von Daten in WEGA-Dateien

Es gibt zwei Moeglichkeiten, wie man die Ausgabe einer SQL-Abfrage an eine WEGA-Datei uebergeben kann. Bei der ersten Moeglichkeit erfolgt die Uebergabe von aus SQL unter Verwendung des Schluesselwortes into. Die Abfrage

```
sql> select * from her into produz /
```

legt die Ergebnisse der Anfrage, in diesem Fall den Inhalt der Datei mit den Herstellern, her, in eine Datei namens produz im aktuellen Verzeichnis ab. Diese Datei enthaelt die Ueberschrift, die normalerweise am Anfang jeder ausgegebenen Seite steht. Es ist zu beachten, dass die Klausel into erst nach dem gesamten Anfrageblock erscheinen darf - into ist immer die letzte Klausel. Somit wuerde eine laengere Abfrage folgendermassen erfolgen:

```
sql> select ....
```

```
sql> from ....
sql> where ....
sql> group by ....
sql> having ....
sql> order by ....
sql> into datei_name /
```

Die zweite Moeglichkeit des Auslagerns von Daten in eine WEGA-Datei bestuende darin, dass man unter der Shell eine gespeicherte Abfrage verwendet und die Standardausgabe auf eine Datei umlenkt. Will man beispielsweise die Ergebnisse der oben aufgefuehrten Abfrage in einer Datei namens produz ablegen, ohne das Schluesselwort into zu verwenden, erstellt man mit dem Editor unter der Shell eine Datei, beispielsweise namens sprod, die die folgende Zeile enthaelt:

```
sql> select * from her /
```

Dann wird das folgende WEGA-Kommando verwendet, um die Ergebnisse in der Datei produz abzulegen:

```
% SQL sprod > produz
```

Moechte man RPT oder WEGA-Textverarbeitungsfilter wie awk oder sed verwenden, ist wahrscheinlich die erlaeuternde Ueberschrift unerwuenscht. Diese kann mit dem Schlueselwort lines eliminiert werden, das verwendet wird, um die Anzahl der Zeilen pro Seite zu setzen. Standard sind 24 Zeilen pro Seite, was der Groesse eines Standard-Bildschirms entspricht. Mit lines legt man also die Anzahl der Zeilen fest, die vor der erneuten Ausgabe der Ueberschrift ausgegeben werden. Bei 0 als Parameter werden nicht etwa gar keine Zeilen ausgegeben, sondern die Ueberschrift vollstaendig unterdrueckt.

Fuegen Sie in die Datei sprod als erste Zeile noch

```
lines 0
```

ein, so erhalten Sie eine Ausgabedatei ohne Ueberschrift.

Auch das Trennzeichen fuer Felder kann vom Nutzer festgelegt werden, und zwar folgendermassen (Beispiel: ':'):

```
sql> separator ':'
```

Standard ist das Pipesymbol (|). Beliebige einzelne Zeichen, einschliesslich <HT> und nichtdruckbare Zeichen, sind zulaessig.

6.2.5 Laden von Daten aus WEGA-Dateien

SQL gestattet die Verwendung des Programms 'Data Base Load' (Abschnitt 3.5.5) zum Einfuegen neuer Datensaetze in die Datenbank und das Modifizieren existierender Datensaetze in

der Datenbank. Von hier aus wird die Verwendung des Programms durch eine Erweiterung der Syntax der Klausel insert moeglich, die es gestattet, den zu ladenden Datensatztyp, die Folge der Felder und den Namen der als Eingabedatei verwendeten WEGA-Datei anzugeben. Damit kann man Daten aus einer anderen Datenbank oder einem anderen Anwendungssystem laden und auch Daten innerhalb des eignenen WEGA-DATA-Anwendungssystems umlagern. In Abschnitt 3.5.5 sind alle Meldungen und Bedingungen fuer die Verwendung des Programms 'Data Base Load' beschrieben, so dass hier auf eine Beschreibung verzichtet wird.

Beim Fuellen der Datenbank des Uebungsbeispiels (Nutzerhandbuch), wurde ausgiebig von dieser Moeglichkeit Gebrauch gemacht, so dass hier auf Beispiele verzichtet wird. Allerdings soll dafuer der Inhalt und Aufbau der eingelesenen Dateien gezeigt werden.

```
best_data
---------
1|03/02/86|1
2|03/05/86|3
3|03/12/86|2

kunde_data
----------
1|Buchhandlung "Samuel Butler"|Urgasse 12|Erewhon|1820|
                                                843255|446990
2|Theo Retisch|Wunschallee 1102|Irgendwo|1111|0|566331
3|1000 Grosse Dinge|Am Steilhang 1|Bad Berg|1984|449277|
                                                    653575
```

(Die Zeilen 1 und 3 sind nur aus Platzgruenden auf zwei Zeilen verteilt.)

Falls Sie sich ein Beispiel in eine Datei haben ausgeben lassen, haben Sie sicher bemerkt, dass die Felder in ihrer vollen Breit ausgegeben werden. Nicht "besetzte" Feldplaetze sind mit Blanks ausgefuellt. Wie oben zu sehen, brauchen die einzulesenden Dateien nicht die Felder in ihrer vollen Laenge zu enthalten. WEGA-DATA fuellt die "fehlenden" Plaetze richtig aus.

Entspricht die Reihenfolge der Felder in der Datei der Reihenfolge der Felder des entsprechenden Datensatztyps, braucht keine weitere Anpassung vorgenommen zu werden. SQL generiert automatisch die unter 'Data Base Load' beschriebene Spezifikationsdatei, da SQL davon ausgeht, das die oben beschriebene "Standardreihenfolge" benutzt wird. Sollte man diese nicht wissen, kann man sie u.a. durch select * from dasatyp/ feststellen.

Man kann explizit angeben, in welcher Reihenfolge die Felder in der Eingabedatei sind und welche Felder ueberhaupt. Beim Einlesen einer Datei muessen nicht alle Felder des entsprechenden Datensatztyps mit Werten aus der Eingabeda-

tei gefuellt werden.

BEISPIEL: Der Datensatztyp modell hat die Felder mod_nummer, hersteller_num und bezeichnung in dieser Reihenfolge. Es sollen nun Datensaetze dieses Datensatztyps erstellt werden, wobei die Eingabedatei nicht alle Felder enthaelt und auch nicht in der "richtigen" Reihenfolge.

```
sql> lines 0
sql> select bezeichnung, hersteller_num
sql> from modell
sql> where hersteller_num = 712
sql> into produz /
recognized query!
sql> _
     Ergebnis: produz
               ------
               Matratze                    |   712
               Kugellager                  |   712

sql> insert into modell (bezeichnung, hersteller_num):
sql> from produz /
recognized update!
Record: 2 is a duplicate key.
1 record(s) inserted
```

Was ist geschehen? Wir wollten zwei (unvollstaendige) Datensaetze zum Datensatztyp modell hinzufuegen. Felder, die in der Eingabedatei nicht enthalten sind, hier mod_nummer, werden von SQL mit dem "Null"-Wert des Feldtyps ausgefuellt, hier 0. Deshalb haben wir jetzt (wenn Sie nachsehen und natuerlich nicht nur dann) einen zusaetzlichen Datensatz mit der mod_nummer 0. Aus eben genannten Gruenden sollte der zweite einzufuegende Datensatz auch diesen Wert erhalten, aber mod_nummer ist der Primaerschluessel von modell und kann daher nur einmal vergeben werden. Deshalb verweigerte SQL die Aufnahme des zweiten Datensatzes.

Nun bereinigen wir noch unsere Aktionen. Eingabe:

```
rm -i produz
```

unter der Shell und unter SQL:

```
sql> delete modell where mod_nummer = 0 /
sql> end
```

Auf diese Weise haben wir SQL verlassen.

6.2.6 SQL ueber Bildmasken

Bei den Anwendungssystemen gibt es normalerweise eine Anzahl von Reporten, die haeufig abgearbeitet werden. Das Format des Reports bleibt gleich, aber es kann sich erforderlich machen, das aktuelle Datum einzufuegen, die Reporte

nur fuer bestimmte Kunden zu verwenden oder nur fuer bestimmte Postleitzahlen usw. SQL ueber Bildmasken ermoeglicht es, SFORM-Bildmasken mit einer SQL-Anfrage und mit bis zu acht optionellen RPT-Skripten oder Nutzerprogrammen zu verbinden und damit diese wiederkehrenden Reporte zu verarbeiten.

'SQL ueber Bildmasken' besteht aus zwei Hauptteilen. Der erste Teil - 'SQL Screen Registration', ssqlmnt (siehe Abschnitt 6.2.6.1), - ermoeglicht es dem Nutzer festzulegen, welche SFORM-Bildsmaske welchen Anfragen und Reportskripten zugordnet werden soll und wohin der Nutzer die Ausgabe des Reports senden kann. Die Registrierung fuehrt zu einem Eintrag im Datenwoerterbuch, der als SQL-Bildmaske bezeichnet wird. SQL-Bildmasken koennen in gleicher Weise in Menues abgelegt werden, wie normale ausfuehrbare Programme und ENTER-Bildmasken.

Der zweite Teil steuert die Abarbeitung der SQL-Bildmaske. Der SQL-Bildschirmtreiber (siehe Abschnitt 6.2.6.2) zeigt die SFORM-Bildmaske an und gestattet fuer jedes Bildmaskenfeld die Eingabe von Datenwerten. Bestaetigt der Nutzer, dass die Werte akzeptabel sind, wird ihm mit dem SQL 'Report Options Screen' (siehe Abschnitt 6.2.6.3) ermoeglicht anzugeben, welches Listenformat verwendet werden soll, wohin die Ausgabe zu senden ist und ob er auf die Ergebnisse warten oder einen anderen Arbeitsgang erledigen moechte, waehrend der Report im Hintergrund erstellt wird.

Eine SQL-Bildmaske ermoeglicht zur Laufzeit die Angabe von Werten, die in SQL- und RPT-Skripten ersetzt werden sollen. Das heisst, es muss eine Moeglichkeit geben, in Anfrage- und Report-Skripten anzugeben, wo die vom Nutzer gelieferten Werte zu ersetzen sind. Das geschieht, indem man Parameter in die Skripten einbringt. Dann werden die vom Nutzer auf dem Bildschirm eingegebenen Werte die vorher im Skript dort befindlichen Parameter ersetzen.

Ein Parameter ist einfach eine Platzmarkierung der Form $n, wobei n eine Zahl zwischen 1 und der Anzahl der Parameter ist, die der Nutzer liefern soll. Diese Platzmarkierungen koennen im Text eines Skriptes an beliebiger Stelle positioniert werden. Das Skript selbst wird in einer normale WEGA-DATA-Textdatei abgelegt und dann in eine SFORM-Bildmaske eingetragen (6.2.6.1). Der fuer das erste Feld der Bildmaske eingegebene Wert ersetzt $1, der zweite $2 usw.

Im allgemeinen werden Parameter in SQL-Skripten in where- und having-Klauseln verwendet, um anzuzeigen, welche Werte ausgewaehlt werden sollen. Sollen beispielsweise nach Angabe des Nutzers Datensaetze ausgewaehlt werden, die sich auf einen zwischen zwei Daten liegenden Zeitraum beziehen, wird die where-Klausel folgendermassen verwendet:

```
where liefer_datum between $1 and $2
```

Oder sollen z.B. Produktionsabteilungen ausgewaehlt werden, in denen die durchschnittliche Stueckzahl ueber einem vom Nutzer gelieferten Wert liegt, kann man folgendermassen verfahren:

```
having avg(stueckzahl) > $1
```

Selbstverstaendlich koennen im Skript so viele Parameter verwendet werden, wie es der Platz auf der SFORM-Bildmaske gestattet. Es ist jedoch darauf zu achten, dass die Anzahl der Parameter im SQL-Skript mit der Anzahl der Bildmaskenfelder auf dem Formular uebereinstimmt, da sonst der Bildschirmtreiber nicht weiss, was er mit den ueberschuessigen Werten tun soll.

Sind beispielsweise auf einem Formular drei Bildmaskenfelder und im Skript sind zwei Parameter, kann nicht gesagt werden, was mit dem zusaetzlichen Wert geschehen wird. Es ist moeglich, dass er ignoriert wird, aber wahrscheinlich wird durch ihn der Aufbau der Bildmaske oder des Skripts unkorrekt. Ist andererseits nur ein Bildmaskenfeld vorhanden und ist fuer den zweiten Parameter kein Wert vorhanden, entsteht ein Syntax-Fehler, wenn SQL das Skript verarbeitet. Deshalb meldet der Bildschirmtreiber einen Fehler, wenn die Anzahl der Bildmaskenfelder nicht mit der Anzahl der Parameter im SQL-Skript uebereinstimmt.

Parameter werden in den Skripten genauso ersetzt, wie sie vom Nutzer auf dem Bildschirm eingegeben werden. Das heisst, dass man, wenn ein Zeichenketten-Parameter verwendet werden soll, diesen in einfache Anfuehrungszeichen setzen muss. Sollen beispielsweise Datensaetze fuer kort - ein Zeichenkettenfeld der Laenge 20 - ausgewaehlt werden, ist die where-Klausel folgendermassen zu verwenden:

```
where kort = '$1'...
```

Bei Nichtangabe der Anfuehrungszeichen entsteht bei der Verarbeitung des Skripts durch SQL ein Syntaxfehler.

Soll die SQL-Ausgabe von einem anderen Programm verarbeitet werden - sei es RPT, ein WEGA-Dienstprogramm oder ein Nutzerprogramm - sollen normalerweise die Ueberschriften unterdrueckt werden, die die Felder in der Ausgabe bezeichnen. Das kann mit dem im Abschnitt 6.3.3.3 beschriebenen Kommando lines erfolgen. Zur Unterdrueckung dieser Ueberschriften wird einfach das Kommando lines 0 an den Anfang des Skripts gestellt.

6.2.6.1 Registrierung von Bildmasken mit SQL

Bevor SQL eine Bildmaske bedienen kann, muss diese registriert werden. Das kann unter Verwendung von ENTER geschehen (5.1), aber nur mit einem beider Programme. Mit Hilfe eines 'Executable Listing' kann man feststellen, mit wel-

chem beider Programme eine Bildmaske registriert wurde (2.2.1).

Der naechste Schritt bei Anfragen durch SQL ueber Bildmasken besteht darin, dass die SFORM-Bildmaske im SQL-Bildschirmtreiber abgelegt wird. Damit wird dem Menue-Handler mitgeteilt, was angezeigt werden soll, wenn eine SQL-Bildmaske aktiv ist und welches SQL-Skript zu verwenden ist.

Mit diesem Programm koennen auch Reporte abgelegt werden, die die Ausgabe der SQL-Anfrage verwenden. Jeder Report ist das Produkt eines Formatierungsprogramms, der damit verbundenen Parameter, eines Titels und der Ausgabeoptionen. Durch Anwaehlen von 5 ruft man das 'System Menu' auf und waehlt dann 7, 'SQL Screen Registration'.

Es kann immer nur eine SFORM-Bildmaske mit SQL abgelegt werden. Es erscheint folgender Bildschirminhalt:

```
[ssqlmnt]                    WDATA SYSTEM
                          24 JUL 1986 - 15:25
                       SQL Screen Registration

SCREEN NAME:
SQL SCRIPT:                          HEADING:

FORMATTING PROGRAMS:
      NAME       PARAMETERS     TITLE              OUTPUT
            -
            -
            -
            -
CMD _____________________________________________________________


[I]NQUIRE, [A]DD, [M]ODIFY, [D]ELETE _
```

Die Prompter haben folgende Bedeutungen:

[I]NQUIRE,[A]DD, [M]ODIFY, [D]ELETE

Mit diesem Prompter waehlt man einen Operationsmodus. Operationsmodi sind:

i - Abfragemodus; gestattet das Lesen der abgelegten Daten. Es ist nicht zulaessig, Daten zu aendern oder hinzuzufuegen.

a - Einfuegemodus; gestattet das Ablegen von SFORM-Bildmasken im SQL-Bildschirmtreiber.

m - Aenderungsmodus; gestattet die Aenderung aller Felder einer existierenden SQL-Bildmaske

d - Loeschmodus; gestattet das Loeschen einer SQL-Bildmaske. Aendern und Hinzufuegen von Daten ist nicht zulaessig. Wird eine SQL-Bildmaske geloescht, bleibt die urspruenglich von SFORM erzeugte Bildmaske unberuehrt. Alle Verweise auf die SQL-Bildmasken (z.B. Menueauflistung) werden jedoch geloescht. Loeschen von Bildmasken aus SFORM, siehe Abschnitt 4.1 und Erhalt von Menueauflistungen siehe Abschnitt 2.2.2.

SCREEN NAME:
Der Name der abzulegenden Bildmaske.

SQL SCRIPT:
Der Name der Datei, die das mit der Bildmaske in Zusammenhang stehende SQL-Skript enthaelt.

HEADING:
Eine Zeichenkette mit bis zu 34 Zeichen, die auf dem Bildschirm angezeigt wird, wenn eine SQL-Anfrage ueber Bildmaske erfolgt. Die Ueberschrift HEADING: wird auch auf allen Menues angezeigt, in denen sich die Bildmaske befindet. Sie kann als literaler Text eingegeben werden oder aber als Zeichen, denen ein Umschaltzeichen vorangestellt wird, das anzeigt, dass der Prompter in einem bestimmten Modus angezeigt werden soll, falls die termcap-Tabelle diesen Modus fuer das aktuelle Terminal diesen Modus zulaesst (siehe Abschnitt 1.1.4). Das Umschaltzeichen ist die Tilde (~). Folgende spezielle Modi sind zulaessig:

~r - Anfang Video invers
~s - Ende Video invers
~u - Anfang Unterstreichung
~v - Ende Unterstreichung

Beginnt man eine Ueberschrift mit einem speziellen Anzeigemodus, muss man diese auch damit beenden, da sonst eine unkorrekte Bildmaske entsteht.

FORMATTING PROGRAMS:
Der Platz unter diesem Prompter ist fuer die Aktualisierung und Anzeige der Report-Generationsprozesse fuer die SQL-Bildmaske. Die Kombination von Formatierungsprogrammnamen, Parametern, Titel und Ausgabeoptionen wird einfach als ein Report bezeichnet.

NAME
Ist der erste Prompter im Aktualisierungsbereich. Akzeptiert einen aus acht Zeichen bestehenden Namen des ablauffaehigen Formatierungsprogramms, das zur Erzeugung der gewuenschten Ausgabe ablaufen wird. Jeder SQL-Bildmaske koennen maximal 8 verschiedene Formatierungsprogramme zugeordnet werden. Als Formatierungsprogramme fuer 'SQL Screen Registration' kommen in Frage: das Formatierungsprogramm RPT (Reportgenerator), ein vom Nutzer geliefertes, nicht zu WEGA-DATA gehoeriges, aber

ablauffaehiges Programm oder ein Leerzeichen. (RPT muss in Grossbuchstaben eingegeben werden.) Ein vom Nutzer geliefertes Programm wird als Nutzerprogramm bezeichnet.

Mit Ausnahme von RPT sind alle zu WEGA-DATA gehoerigen ablauffaehigen Namen nicht zulaessig und werden nicht akzeptiert. Wird ein Leerzeichen eingegeben, wird kein Formatierungsprogramm verwendet und der Report listet die Felder in einfacher tabellarischer Form auf. Daher koennen sich durch Eingabe eines Leerzeichens Zeilenteile ueberschreiben.

Dieses Feld wird editiert, so dass nur die vor einem Leerzeichen auftretenden Zeichen verbleiben. Die Zeichenkette wird vermittels RETURN linksbuendig ausgerichtet und vorangestellte Leerzeichen und zusaetzliche Zeichen werden geloescht.

PARAMETERS

Es koennen vier Zeichenketten mit jeweils 14 Zeichen eingegeben werden, die als Parameter an das Formatierungsprogramm gegeben erden. Es ist Aufgabe des Formatierungsprogramms, die Bedeutung der Parameter zu interpretieren. Die Parameter werden vertikal nach jedem Bindestrich (-) eingegeben. Die Anzahl der zulaessigen Parameter haengt von der Art des verwendeten Formatierungsprogramms ab.

Formatierungsprogramm		Anzahl der erlaubten Parameterfelder
RPT	-	1 (Der Name eines RPT-Skripts)
(leer)	-	0
Nutzerprogramm	-	4

PARAMETERS muessen als ein Wort eingegeben werden, und es duerfen keine Leerzeichen darin enthalten oder ihm vorangestellt sein. Bezeichnet NAME jedoch ein Nutzerprogramm, werden die Parameter nicht editiert und es kann mehr als nur ein tatsaechlicher Parameter in jedes Feld PARAMETERS eingetragen werden, indem Zwischenraeume frei gelassen werden (vorausgesetzt, dass die Gesamtlaenge 14 Zeichen nicht ueberschreitet). Das Nutzerprogramm wird gestartet, als waere das folgende Kommando unter der Shell eingegeben worden.

```
SQL sql_skript|programm p1 p2 p3 p4
```

Dabei ist sql_skript der Name der SQL-Skriptdatei, die zum Bildschirm gehoert. programm ist der Name eines Formatierungsprogrammes (RPT oder der Name eines Nutzerprogramms). p1 - p4 sind die vier Parameter-Zeichenketten.

TITLE

Erlaeuternde Zeile fuer den Report der erzeugt wird, wenn diese Option ueber den SQL Report Options Screen

(Abschnitt 12.3) beansprucht wird. Dieses Feld wird auf dem 'Report Options Screen' rechts von der Auswahlnummer erscheinen.

OUTPUT

Bestimmt, wohin der Report ausgegeben werden soll und ob der optionelle Debug-Modus eingeschaltet werden soll. Die Ausgabe kann pro Durchlauf auf ein oder mehrere Ziele gerichtet werden. Bis zu 4 Zeichen werden akzeptiert. Diese sind:

S - Einschalten der Bildschirmausgabe (STANDARD)
P - Einschalten der Druckerausgabe
F - Einschalten der Ausgabe auf eine Datei
N - Kein Debug-Modus (STANDARD ist: Debug aktiv)

Eine Eingabe in Kleinbuchstaben ist moeglich. Diese werden in Grossbuchstaben konvertiert. Doppelt auftretende Zeichen werden ignoriert.

Laeuft der Report unter dem Debug-Modus, werden alle Fehlermeldungen, alle verwendeten Skripts und die Laufzeit-Ergebnisse angezeigt. Bei freigegebenem Debug-Modus kann das tatsaechlich verwendete Skript geprueft werden. Als Hilfestellung fuer die Korrektur von Programm- oder Skriptfehlern, werden Syntaxfehler angezeigt.

CMD

Den verbleibenden Platz auf dem Bildschirm nimmt der Anzeigebereich ein. Hier wird die erste Zeile eines jeden abgelegten Reports angezeigt und zwar in der Reihenfolge, in der sie auf dem SQL 'Report Options Screen' erscheinen werden. Im Aktualisierungsbereich koennen einzelne Reporte modifiziert und hinzugefuegt werden. Reporte werden jedoch direkt aus diesem Bereich geloescht. Unter CMD werden Kommandos fuer eine Zeile eingegeben.

q - Verlassen des Anzeigebereiches. Kann in jeder beliebigen Zeile eingegeben werden.

d - Der aktuelle Report wird in dieser SQL-Bildmaske geloescht.

a - Ein neuer Report soll hinzugefuegt werden, der mit dieser SQL-Bildmaske zusammenhaengt. a darf nur in einer Leerzeile am Ende des Anzeigebereiches eingegeben werden.

- Eine Zahl (#) zwischen 1 und x, die eine Zeilennummer darstellt. (x ist die Anzahl der fuer diese Bildmaske abgelegten Reporte.) Das aktuelle Datenelement wird in die Zeile mit der angegebenen Nummer gebracht.

m - Gestattet die Modifizierung aller zu dem aktuellen Report gehoerenden Felder. BEMERKUNG: Wird NAME auf ein Leerzeichen geaendert, werden alle vorher existierenden Parameterfelder geloescht. Wird NAME auf RPT geaendert, werden mit Ausnahme des ersten alle Parameterfelder geloescht.

6.2.6.2 Verwendung von SQL-Bildmasken

Anfragen ueber Bildmasken arbeiten folgendermassen. Hat man hinter dem auf dem Bildschirm unten angezeigten Prompter den Betriebsmodus gewaehlt (I, A, M oder D), wird folgender Prompter angezeigt:

```
Begin search [CTRL E],Clear field [CTRL Z],Exit [CTRL X]
```

Es handelt sich hierbei um die Standard-Steuertastenwerte fuer diese Funktionen. Man kann diese veraendern, indem man die Datei termcap gemaess der in Abschnitt 1.1.4 erfolgten Erklaerung editiert. Diese Steuerzeichen werden benutzt, um folgende Funktionen auszufuehren:

CTRL/E - Beginnt die Suche mit den eingegebenen Optionen. Man kann auch, um die Suche zu starten, einfach RETURN druecken, bis man zum letzten Bildmaskenfeld gelangt, um die Suche zu starten.

CTRL/Z - Damit wird die letzte Auswahlspezifikation geloescht. Hat man beispielsweise einen Feldwert fuer Erwerbsdatum eingegeben und gelangt dann zu der Auffassung, dass das Datum eigentlich gar keine Rolle spielt, stellt man den Cursor auf den Prompter Erwerbsdatum und drueckt CTRL/Z. Die Spezifikation wird dann geloescht und Erwerbsdatum wird in der Anfrage nicht verwendet.

CTRL/X - Verlaesst die aktuelle SQL-Bildmaske und sorgt fuer Rueckkehr in den Menue-Handler. Im Gegensatz zu den zwei anderen CTRL-Eingaben, kann CTRL/X immer dann verwendet werden, wenn ein normales Zeichen auch akzeptiert wuerde. Damit kann man schnell eine SQL-Bildmaske entfernen.

Alle auf einer SQL-Bildmaske angezeigten Felder entsprechen SQL- oder RPT-Skriptparametern. Standardmaessig werden diesen Feldern folgende Werte zugeordnet.

Feldtyp	-	Standardwert
NUMERIC	-	Null
FLOAT	-	Null
AMOUNT	-	Null
STRING	-	Blank
DATE	-	**/**/** (null)
TIME	-	00:00

Um einen Parameterwert anzugeben, wird einfach das entsprechende Feld ausgefuellt. Um eine genaue Uebereinstimmung anzugeben, werden einfach die auf dem Bildschirm befindlichen Felder genau mit der Information ausgefuellt, die man sehen moechte. Zur Angabe einer teilweisen Uebereinstimmung von Zeichenkettenfeldern wird der gleiche Satz von Sonderzeichen verwendet, der von auch von ENTER verwendet wird. Die Sonderzeichen und ihre Bedeutung sind:

? - Das variable Zeichen. Das Fragezeichen entspricht einem beliebigen einzelnem Zeichen.

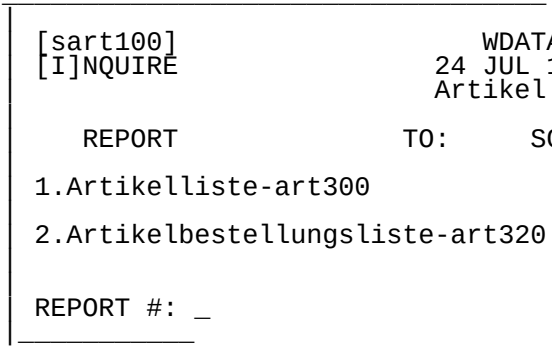
* - Die variable Zeichenkette. Der Stern entspricht einer beliebigen Zeichenkette beliebiger Laenge, einschliesslich Zeichenketten der Laenge Null.

[...] - Das eingeschraenkt variable Zeichen. Die drei Punkte entsprechen einem Satz von Zeichen, die eine Zeichenklasse festlegen. Die Zeichenklasse entspricht einem beliebigen einzelnen Zeichen, das der Klasse angehoert. Zeichenbereiche koennen angegeben werden, indem zwei Zeichen durch einen Bindestrich '-' getrennt werden. Alle Buchstaben koennen z.B. durch [A-Za-z] dargestellt werden (siehe auch Abschnitt 6.1.3.1).

6.2.6.3 Der 'Report Options Screen' von SQL

Wurde eine Menge von Datensaetzen durch Anfragen ueber Bildmasken unter SQL (Abschnitt 6.2.6.2) ausgewaehlt, d.h., wird die entsprechende Bildmaske im Anfragemodus betrieben ([I]NQUIRE), kann man mit dem SQL 'Report Options Screen' einen Report waehlen und angeben, wohin dessen Ausgabe erfolgen soll. Die Reporte, die zur Verfuegung stehen, und die Ausgabeziele werden mit 'SQL Screen Registration' (Abschnitt 6.2.6.1) festgelegt. Verfuegt eine Bildmaske ueber zugeordnete Reporte, ist der 'Report Options Screen' eine Option dieser Bildmaske ([R]EPORT), wenn sie unter dem Anfragemodus betrieben wird.

Hier ein Beispiel fuer einen 'Report Options Screen' von SQL:



Die Prompter werden folgendermassen verwendet:

REPORT #

Jede der aufgelisteten Optionsnummern kann eingegeben werden. Durch Druecken von RETURN geht man standard-maessig in das erste Datenelement des Reports. Von diesem Prompter aus kann man auch mit den Kommandos sh oder edit in die WEGA-Shell oder einen Texteditor gelangen. Diese Moeglichkeit kann genutzt werden, wenn man Formatierungsskripts mit einem Editor modifizieren muss oder wenn man sich eine Ausgabe ansehen moechte, die auf eine Datei umgeleitet wurde.

REPORT

Der ausfuehrliche Titel des Reports, der erzeugt wird, wenn diese Option gewaehlt wird. Diese Zeichenkette ist in der Bildmaske 'SQL Screen Registration' unter TITLE aufgefuehrt.

TO

Die Ausgabe kann geleitet werden auf:

SCREEN	Bildschirm des Terminals
PRINT	Drucker
FILE	Eine Datei namens FILENAME

[] Wird an dieser Stelle x eingegeben, soll die Ausgabe des aktuellen Reports auf dieses Ziel erfolgen. Die Angabe aller zur Verfuegung stehenden Optionen ist moeglich.

Wurde die Option Ausgabe an eine Datei, FILE, gewaehlt, muss hinter diesem Prompter der Name der Ausgabedatei angegeben werden. Existiert dieser Dateiname bereits, erscheint die Meldung

File already exists, destroy it?

Wird auf diese Meldung mit Y geantwortet, wird

die Datei ueberschrieben, wird mit N geantwortet, kann man einen neuen Dateinamen eingeben.

Nachdem man mit 'REPORT #:' einen gueltigen Report gewaehlt hat, wird die entsprechende Zeile invers dargestellt. Der Cursor geht zum Prompter mit dem ersten zulaessigen Ausgabeziel. Nachdem man alle gewuenschten Ausgabeziele markiert hat, kehrt man mittels CTRL/U oder RETURN zum ersten Ausgabeprompter zurueck. Es wird CTRL/U eingegeben und der Prompter zum Verarbeitungsmodus (siehe unten) wird angezeigt. Kann der Report nur auf ein gueltiges Ausgabeziel ausgegeben werden, wird diese Option automatisch angefordert und der Prompter mit dem Verarbeitungsmodus erscheint:

Proceed in [F]oreground,[B]ackground,[D]ebug or [C]ancel?

Mit diesem neuen Prompter wird gesteuert, wie der Report verarbeitet wird. Die Option [D]ebug erscheint nicht, wenn mit 'SQL Screen Registration' der Debug-Modus ausgeschaltet wurde. [B]ackground ist nicht zulaessig, wenn SCREEN als Ausgabeziel gewaehlt wurde. Folgende Moeglichkeiten gibt es also:

f - Der Report laeuft ab, waehrend der Nutzer wartet.

b - Der Report wird gestartet und der Nutzer kann waehrenddessen seine Arbeit fortsetzen (Report im Hintergrund).

d - Der Report laeuft im Vordergrund-Modus und die verwendeten Skripts, die Laufzeit-Ergebnisse und Fehlermeldungen werden angezeigt. Die Verwendung des Debug-Modes ist nuetzlich, da dadurch die verwendeten Skripts mit den tatsaechlichen Parameterwerten an der entsprechenden Stelle angezeigt werden.

c - Die Reportanfrage wird geloescht und zum Prompter 'Report #:' zurueckgegangen.

Waehrend der Reporterstellung tritt natuerlich eine Verzoegerung auf. Wenn ein Report mehr als einen Bildschirminhalt ausfuellt, kann man ihn durch Druecken von RETURN durchblaettern:

Display next page? [RETURN] continues, [n] terminates

Mit RETURN kann der naechste Bildschirm mit Daten angezeigt werden, mit n wird der Prozess beendet. Ist der Report abgeschlossen, erscheint der Prompter:

Complete. Please enter RETURN to continue ->->

Durch Eingabe von RETURN gelangen man wieder zur Report-Auswahl.

Weitere Meldungen des 'Report Options Screen' sind:

<EMPTY>

Die Ausgabedatei enthaelt keine Daten.

Unable to create output file, please reenter filename

Waehrend des Prozesses konnte die Datei mit dem angegebenen Dateinamen (FILENAME) nicht angelegt werden (Ursache ??).

6.3 SQL-Referenz

In den vorhergehenden drei Abschnitten wurden Beispiele fuer die Verwendung von WEGA-DATA-SQL dargelegt. In diesem letzten Abschnitt sollen Informationen fuer jene gegeben werden, die bereits wissen, wie SQL zu verwenden ist, die jedoch einige schnelle Informationen benoetigen. Die von SQL verwendeten reservierten Schluesselworte sind in Abschnitt 6.3.1 aufgefuehrt. In Abschnitt 6.3.2 wird eine Zusammenfassung der Fehlermeldungen und moeglicher Korrekturmassnahmen gegeben. In Abschnitt 6.3.3 ist die formale Sprachsyntax der vollstaendigen SQL-Sprache von WEGA-DATA angegeben.

6.3.1 Zusammenfassung der Schluesselworte

Die in der folgenden Liste aufgefuehrten Worte haben eine besondere Bedeutung fuer SQL. Daher koennen sie nicht fuer Datensatz- oder Feldnamen verwendet werden. Werden diese Worte bei Abfragen als Datensatz- oder Feldnamen verwendet, entstehen Syntaxfehler.

and	delete	group	is	order	start
asc	desc	having	lines	records	sum
avg	edit	help	max	restart	unique
between	end	in	min	select	unlock
by	fields	insert	not	seperator	update
count	from	into	or	set	where

6.3.2 Fehlermeldungen

In diesem Abschnitt wird eine Zusammenfassung der Fehlermeldungen gegeben, die moeglicherweise bei der Verwendung von SQL auftreten. Sie werden in alphabetischer Reihenfolge aufgelistet mit einer kurzen Erklaerung der Fehlerursache und (wenn moeglich) wird als Beispiel eine Anweisung angefuehrt, durch die eine solche Meldung entstehen koennte.

Die Fehler koennen in zwei Hauptklassen unterteilt werden - Fehler, die durch die falsche Angabe oder Ordnung der Schluesselworte innerhalb der Struktur der Anweisung entstehen (Syntaxfehler) und Fehler, durch die die Bedeutung der Anweisung verfaelscht wird (semantische Fehler). Durch Syntaxfehler wird verhindert, dass SQL versteht, was ge-

macht werden soll. Semantische Fehler ergeben sich aus einer grammatisch gueltigen Anweisung, die SQL nicht sinnvoll verarbeiten kann. Beide Fehlerarten koennen keine sinnvollen Ergebnisse hervorbringen.

Es ist jedoch zu beachten, dass SQL nicht alle moeglichen semantischen Fehler aufdecken kann. Damit ist es moeglich, dass eine Abfrage ein Ergebnis bringt, das eine andere Bedeutung hat, als man annehmen koennte. Das kann am ehesten bei der Verwendung von numerischen Funktionen auftreten, wenn man versucht ein Feld aufzulisten, das nicht zu der Gruppe gehoert, die fuer die Berechnung der numerischen Funktion verwendet wird. In diesem Fall meldet SQL keinen Fehler, aber der/die Wert(e), der /die fuer das angegebene Feld(er) aufgelistet wird, hat/haben nichts mit dem zurueckgegebenen Wert der numerischen Funktion zu tun.

Mit einem vorangegangenen Beispiel hatten wir den durchschnittlichen Grosshandelspreis aller Artikel im Lager berechnet. Nun interessiert uns, ob ein Hersteller, der einen Artikel mit diesem Preis herstellt, tatsaechlich existiert.

```
sql> select artmod_mohenr, avg(grosshand_preis)
sql> from art /
```

Es wird kein Fehler angezeigt und es wird in der Tat ein Name und die durchschnittliche Modellanzahl ermittelt. Aber der Name ist nicht der Name des gesuchten Herstellers (hoechstens durch einen unwahrscheinlichen Zufall). Es kann mehrere solcher Hersteller, aber es braucht auch nicht einen einzigen zu geben. Eine treffende Abfrage ist die folgende:

```
sql> select artmod_mohenr, grosshand_preis
sql> from art
sql> where grosshand_preis = select avg(grosshand_preis)
sql>                         from art /
recognized query!
There were no records selected
sql> _
```

Das ist gleichzusetzen mit folgenden zwei Fragen: 'Was ist der Durchschnittspreis fuer alle Artikel?' (die innere Abfrage) und 'Welche Hersteller produzieren solche Artikel?' (die aeussere Abfrage). In diesem Fall entsprach kein Datensatz der Anfrage. In den Abschnitten 6.1.7, 6.1.8 und 6.1.9 sind weitere Informationen enthalten.

A 'having' clause cannot contain nested aggregate functions.

Erklaerung:
Die Klausel having darf nur unverschachtelte numerische Funktionen enthalten. In dem gegebenen Kontext ergibt eine verschachtelte numerische Funktion keinen Sinn.

Ungueltige Anfrage:
```
select artmod_mohenr
from art
group by artmod_mohenr
having avg(sum(grosshand_preis)) > 5000. /
```

Korrektur:
```
select artmod_mohenr
from art
group by artmod_mohenr
having sum(grosshand_preis) > 5000. /
```

A 'having' clause must be preceded by a 'group by' clause.

Erklaerung:
Die having-Klausel berechnet den Wert von numerischen Funktionen auf der Grundlage von Datensatzgruppen, die von der 'group by'-Klausel gebildet werden, somit ist 'group by' fuer die Vewendung von having erforderlich.

Ungueltige Anfrage:
```
select artmod_monr
from art
having avg(grosshand_preis) > 1000./
```

Korrektur:
```
select artmod_monr
from art
group by artmod_monr
having avg(grosshand_preis) > 1000./
```

A 'having' clause requires an aggregate function.

Erklaerung:
Eine having-Klausel wird verwendet, um auf der Grundlage einer numerischen Eigenschaft einer Gruppe eine Auswahl zu treffen. Daher ist die Verwendung einer numerische Funktion erforderlich.

Ungueltige Anfrage:
```
select artmod_mohenr
from art
group by artmod_mohenr
having grosshand_preis > 5000. /
```

Korrektur:
```
select artmod_mohenr
from art
group by artmod_mohenr
having avg(grosshand_preis) > 5000. /
```

A literal tuple contains the wrong number of items.

Erklaerung:
Die Feldspezifikationsliste muss die gleiche Anzahl von Datenelementen enthalten wie jedes Datentupel. Ansonsten ist unklar, was mit dem "ueberschuessigen" Wert oder dem "fehlenden" Feld geschehen soll.

Ungueltige Anfrage:
```
select name
from her
where <nummer, ort> = <'Schukow*'>/
```

Korrektur:
```
select name
from her
where <nummer, ort> = <709, 'Schukow*'>/
```

Aggregate functions are not allowed in the 'where' clause.

Erklaerung:
Numerische Funktionen koennen nur in den Klauseln having und select verwendet werden. Soll eine Auswahl auf der Grundlage einer numerischen Funktion getroffen werden, muss die Abfrage umgeschrieben werden, so dass eine having Klausel verwendet wird.

Ungueltige Anfrage:
```
select artmod_mohenr
from art
where avg(grosshand_preis) > 5000. /
```

Korrektur:
```
select artmod_mohenr
from art
group by artmod_mohenr
having avg(grosshand_preis) > 5000. /
```

Aggregate functions imbalance in the 'select' clause.

Erklaerung:
Innerhalb einer einzelnen Abfrage muss die Verschachtelung aller numerischen Funktionen die gleiche Tiefe haben. Moechte man beide Antworten haben, muss man zwei Anfragen ausloesen.

Ungueltige Anfrage:
```
select avg(grosshand_preis), sum(avg(grosshand_preis))
from art /
```

Korrektur:
```
select avg(grosshand_preis)
from art /

select sum(avg(grosshand_preis)
from art
group by artmod_mohenr /
```

Aggregate functions may not be nested without a 'group by' clause.

Erklaerung:
Da eine numerische Funktion fuer eine Gruppe von Werten einen einzelnen Wert berechnet, ist es nicht sinnvoll eine numerische Funktion auf einen einzelnen Wert anzuwenden. Aber gerade das tut man, wenn man eine numerische Funktionen ohne die Verwendung der Klausel 'group by' verschachtelt.

Ungueltige Anfrage:
```
select sum(avg(grosshand_preis))
from art /
```

Korrektur:
```
select sum(avg(grosshand_preis)
from art
group by artmod_mohenr /
```

Aggregate functions nested too deeply in the 'select' clause.

Erklaerung:
Die Verschachtelung von numerischen Funktionen ist nur bis zu zwei Ebenen erlaubt.

Ungueltige Anfrage:
```
select max(sum(avg(grosshand_preis)))
from art /
```

Korrektur:
```
select sum(avg(grosshand_preis)
from art
group by artmod_mohenr /
```

An aggregate function is required when using a 'group by' clause.

Erklaerung:
Eine Klausel 'group by' wird nur verwendet, um Datensaetze fuer die nachfolgende Berechnung einer numerischen Funktion in eine Gruppe zusammenzufassen. Daher gibt es keinen Sinn, wenn man Datensaetze in Gruppen zusammenfasst und keine numerische Funktion berechnet.

Ungueltige Anfrage:
```
select artmod_mohenr, grosshand_preis
from art
group by artmod_mohenr /
```

Korrektur:
```
select artmod_mohenr, min(grosshand_preis)
from art
group by artmod_mohenr /
```

An expression cannot be compared with a literal tuple list.

Erklaerung:
Das Ergebnis eines Ausdrucks ist ein einzelner Wert, waehrend ein Datentupel mehrere Werte enthaelt.

Ungueltige Anfrage:
```
select bestellnummer
from art
where grosshand_preis+ind_abgabepreis is in
                                ( <5000., 'Schukow*'>
                                  <128.28, 'Bad Disko'>
                                  <76., 'Schukow'> ) /
```
Korrektur:
```
select bestellnummer
from art
where grosshand_preis+ind_abgabepreis is in
                                <5000., 128.28, 76.> /
```

Can't create 'xxxx'

Erklaerung:
SQL kann nicht die Datei namens xxx anlegen. Im aktuellen Verzeichnis sind die Zugriffsrechte daraufhin zu ueberpruefen, ob der aktuelle WEGA-Nutzer in das Verzeichnis schreiben darf.

Fatal error

Erklaerung:
SQL hat einen nicht zu behebenden internen Fehler entdeckt. Die vor dieser Meldung angezeigte Fehlermeldung verweist auf den Charakter des Fehlers. Die Abfrage ist umzuschreiben.

Field: ffff requires a password, use 'unlock'.
Field: xxxx.ffff requires a password, use 'unlock'

Erklaerung:
Das angegebene Feld wurde entweder fuer die Anweisung select, where oder having gegen unbefugten Zugriff geschuetzt. Die Sperrung des Feldes ist aufzuheben, bevor man es verwenden kann.

Insufficient memory for sort line.

Erklaerung:
In der Klausel 'group by' oder 'order by' sind zu viele Elemente enthalten. Es sind weniger Elemente zu verwenden.

Invalid 'group by' clause.

Erklaerung:
Die Syntax der Klausel 'group by' ist nicht korrekt. Viele verschiedene Fehler koennten dafuer die Ursache sein. Im Beispiel besteht der Fehler darin, dass man Gruppen nicht nach berechneten Feldern zusammenfassen kann.

Ungueltige Anfrage:
```
select avg(grosshand_preis)
from art
group by (grosshand_preis+ind_abgabepreis) /
```

Korrektur:
```
select avg(grosshand_preis)
from art
group by grosshand_preis, ind_abgabepreis /
```

Invalid 'having' clause.

Erklaerung:
Die Syntax der Klausel having ist nicht korrekt. Viele verschiedene Fehler koennen dafuer die Ursache sein. Im Beispiel besteht der Fehler darin, dass in der having-Klausel ein booleschen Ausdruck (d.h. ein Ausdruck, der einen der Operatoren =, <, > usw. verwendet) fehlt.

Ungueltige Anfrage:
```
select artmod_mohenr
from art
group by artmod_mohenr
having max(avg(grosshand_preis)) /
```

Korrektur:
```
select artmod_mohenr
from art
group by artmod_mohenr
having avg(grosshand_preis) =
    select max(avg(grosshand_preis)) from art
    group by artmod_mohenr /
```

Invalid 'select' list syntax.

Erklaerung:
Die Syntax der Klausel select ist nicht korrekt. Viele verschiedene Fehler koennen hierfuer die Ursache sein. Im Beispiel sind die zu waehlenden Felder in winklige Klammern eingeschlossen.

Ungueltige Anfrage:
```
select <name, ort>
from her /
```

Korrektur:
```
select name, ort
from her /
```

Invalid 'where' clause.

Erklaerung:
Die Syntax der Klausel where ist nicht korrekt. Viele verschiedene Fehler koennen hierfuer die Ursache sein. Im Beispiel wurde derselbe Fehler wie bei der falschen having-Klausel gemacht; der Ausdruck besitzt keinen Vergleichsoperator, mit dem ein Vergleich erfolgen koennte.

Ungueltige Anfrage:
```
select artmod_mohenr
from art
where grosshand_preis /
```

Korrektur:
```
select artmod_mohenr
from art
where grosshand_preis > ind_abgabepreis
```

Invalid date.

Erklaerung:
Das Format einer Konstanten DATE ist nicht korrekt. Datenkonstanten, DATE, haben die Form MM/DD/YY, wobei MM der Monat (1-12), DD der Tag (1-31, je nach Monat) und YY das Jahr ist.

Invalid field spezification: xxxx.ffff

Erklaerung:
Die erfolgte Feldspezifikation ergibt keinen Sinn - xxxx ist der Datensatztyp und ffff ist das Feld. Im Beispiel trat beim Datensatztyp ein Tipfehler auf.

Ungueltige Anfrage:
```
select hir.name
from her /
```

Korrektur:
```
select her.name
from her /
```

Invalid field: ffff

Erklaerung:
Der Feldname ffff wird nicht anerkannt. Es ist zu ueberpruefen, ob der wirklich der lange Feldname verwendet und ob er richtig geschrieben wurde.

Ungueltige Anfrage:
```
select hername
from her /
```

Korrektur:
```
select name
from her /
```

Invalid query block.

Erklaerung:
Der Anfrageausdruck (meist wenn in Datei enthalten), wird nicht erkannt. Kann durch viele verschiedene Fehler entstehen. Im Beispiel wurde das Schluesselwort select falsch geschrieben.

Ungueltige Anfrage:
```
sellect name
from her /
```

Korrektur:
```
select name
from her /
```

Invalid record type specification in the 'select' clause:
xxxx

Erklaerung:
Der vor dem '.*' verwendete Datensatztyp xxxx ist unbekannt. Wahrscheinlich liegt ein Schreibfehler vor.

Ungueltige Anfrage:
```
select hir.*
from her /
```

Korrektur:
```
select her.*
from her /
```

Invalid record type: xxxx

Erklaerung:
Der in der Abfrage verwendete Datensatztyp xxxx wird nicht erkannt. Wahrscheinlich wurde eine falsche Schreibweise verwandt.

Ungueltige Anfrage:
```
select *
from hir /
```

Korrektur:
```
select *
from her /
```

Invalid time.

Erklaerung:
Das Format der Zeitkonstanten TIME ist nicht korrekt. TIME-Konstante haben die Form HH:MM, wobei HH die Stunden von 0 bis 23 und MM die Minuten von 0 bis 59 darstellen.

ffff is an invalid field name for record type xxxx.

Erklaerung:
Das im Kommando unlock angegebene Feld ist nicht im angegebenen Datensatztyp enthalten. Die Schreibweise und die Liste der gueltigen Feldnamen fuer den entsprechenden Datensatztyp sind zu ueberpruefen.

ffff is an invalid field name.

Erklaerung
Das in der Feldspezifikationsliste fuer das Kommando insert angegebene Feld ist nicht korrekt oder wurde falsch geschrieben.

xxxx is an invalid file name.

Erklaerung:
Die WEGA-Datei namens xxxx existiert entweder nicht oder sie ist nicht lesbar. Es ist moeglich, dass der Name falsch geschrieben wurde oder dass die Datei einen Modus hat, so dass sie der Nutzer nicht lesen darf. Beide Moeglichkeiten koennen mit dem WEGA-Kommando ls(1) ueberprueft werden.

pppp is an invalid password for xxxx.ffff.

Erklaerung:
Das fuer das Freigeben des Feldes angegebene Passwort ist nicht korrekt.

xxxx is an invalid record type.

Erklaerung:
Der angegebene Datensatztyp existiert in der Datenbank nicht. Wahrscheinlich Schreibfehler.

No help is available on this subject.

Erklaerung:
Entweder wurde das Schluesselwort, fuer das Hilfe angefordert wurde, falsch geschrieben oder die Hilfsdatei fuer das Schluesselwort fehlt oder ist nicht lesbar. Es ist auch moeglich, dass die Umgebungsvariable WDATA nicht korrekt gesetzt wurde und dass dadurch das help-Verzeichnis nicht gefunden werden konnte.

Not enough memory

Erklaerung:
Die Abfrage ist entweder zu lang oder sie ist zu tief verschachtelt. Die Abfrage ist entweder zu vereinfachen oder aufzuteilen.

SQL: Can't open xxxx

Erklaerung:
SQL wurde mit einer Shell-Kommandozeile gestartet, um das Skript in der Datei namens xxxx abzuarbeiten. Der aktuelle WEGA-Nutzer kann diese Datei nicht oeffnen. Der Zugriffsmodus fuer die Datei ist zu aendern, damit der Nutzer die Datei lesen kann.

Syntax error ...

Erklaerung:
Die aktuelle Abfrage enthaelt einen Syntaxfehler. Unmittelbar nach dieser Fehlermeldung erscheint normalerweise eine andere Meldung, die den Fehler genauer spezifiziert.

The 'from' clause is missing or invalid.

Erklaerung:
Fuer jede Abfrage ist eine Klausel from erforderlich. Es fehlt entweder die Klausel from oder ihre Syntax ist nicht korrekt. Kann durch verschiedene Fehler verursacht werden. Im Beispiel wurde das Schluesselwort from falsch geschrieben.

Ungueltige Anfrage:
```
select *
fromm her /
```

Korrektur:
```
select *
from her /
```

The correct usage is: SQL [filename]

Erklaerung:
SQL wurde mit einer Shell-Kommandozeile mit einer falschen Anzahl von Parametern gestartet.

The field 'ffff' is not in record type 'xxxx'.

Erklaerung:
Obwohl das Feld 'ffff' ein gueltiges Datenbank-Feld ist, befindet es sich nicht im Datensatztyp 'xxxx'.

Ungueltige Anfrage:
```
select her.hersteller_num
from her /
```

Korrektur:
```
select her.nummer
from her /
```

The file 'xxxx' already exists.

Erklaerung:
Die Ausgabe der Abfrage wurde mit der Klausel into in eine Datei gelenkt, die bereits existiert. SQL ueberschreibt existierende Dateien nicht. Die Datei ist entweder zu entfernen oder die Ausgabe ist in eine Datei anderen Namens zu lenken.

Type conversion error.
Left type is XXXX, right type is XXXX, operator is XXXX.

Erklaerung:
Die Typen zweier Operanden sind unvertraeglich. Ein Typ kann nicht in einen anderen Typ konvertiert werden. Wahrscheinlich wurde einfach der falsche Feldname verwendet.

Ungueltige Anfrage:
```
select best_datum
from best
where best_datum = 2 /
```

Korrektur:
```
select best_datum
from best
where best_nummer = 2 /
```

Unable to open xxxx

Erklaerung:
SQL wurde von einer Shell-Kommandozeile aus gestartet, wobei die Eingabe von einem Skript namens xxxx kam. Auf diese Datei ist der Zugriff nicht moeglich. Es ist zu pruefen, ob der aktuelle WEGA-Nutzer Leserecht fuer die Datei hat.

Unrecognized sql command

Erklaerung:
Das erste Schluesselwort im SQL-Kommando wurde nicht erkannt. Wahrscheinlich liegt ein Schreibfehler vor.

Warning: 'group by' and 'order by' in the same query.
The 'order by' will be ignored.

Erklaerung:
Die Klauseln 'group by' und 'order by' koennen nicht in ein und derselben Abfrage verwendet werden. Die Meldung weist darauf hin, dass 'order by' nicht ausgefuehrt wird. Die restliche Abfrage wird aber ausgefuehrt.

Warning: 'unique' may not be used with aggregate functions.
The 'unique' specification will be ignored.

Erklaerung:
Das Schluesselwort unique kann nicht in Verbindung mit einer der numerischen Funktionen min, max, sum, avg oder count verwendet werden. Die Abfrage wird dennoch ausgefuehrt, aber doppelte Zeilen werden nicht eliminiert.

on or about line nn

Erklaerung:
Diese Meldung wird an die Standard-Fehlerausgabe gesendet, wenn eine Abfrage von einem Skript aus ablaeuft und ein Syntaxfehler entdeckt wird. Es wird in etwa angegeben, wo der Syntaxfehler auftrat.

6.3.3 Formale Sprachbeschreibung von SQL

Die Grammatik vo WEGA-DATA-SQL besteht aus vier grundsaetzlichen Teilen, von denen jeder seine eigene Struktur und Funktion besitzt. In diesem Abschnitt soll die Funktion der Beschreibungsmittel erlaeutert werden und in den folgenden Unterabschnitten wird die Struktur der Bestandteile von SQL genau beschrieben.

Der erste und groesste Teil ist die Anfragesprache. Mit der Anfragesprache kann man Informationen in der eigenen Datenbank abfragen, indem man die Frage "was" stellt und SQL herausfinden laesst, "wie" man die Information erhalten kann. Der zweite Teil ist die Datenmanipulationssprache (DML). Mit DML kann man die Datenbank aktualisieren, indem man existierende Datensaetze aktualisiert oder loescht und neue Datensaetze hinzufuegt.

Der dritte Teil ist die Kommandosprache. Mit der Kommandosprache kann man in vielfaeltiger Weise die interaktive Entwicklung und Ausfuehrung von Anfrage- und DML-Anweisungen erleichtern. Dazu gehoert auch das Editieren der aktuellen SQL-Anweisung, das Starten bzw. wiederholte Starten von Abfragen von Skriptdateien aus, das Freigeben von Feldern, die durch die Festlegung der Zugriffsrechte gesperrt sind und die Ausfuehrung von WEGA-Shellkommandos. Der letzte Teil ist die "Hilfe" zu SQL. Damit kann man sich schnell einen Ueberblick verschaffen, wie bestimmte Kommandos verwendet werden und auflisten, welche gueltigen Datensaetze und Feldnamen sich in der aktuellen Datenbank befinden.

Innerhalb des gesamten Abschnitts soll die folgende Notation fuer die Beschreibung der Grammatik verwendet werden:

Fettdruck	Hierbei handelt es sich um SQL-Schluesselworte. Sie haben fuer SQL eine besondere Bedeutung und duerfen nicht als Datensatz- oder Feldnamen verwendet werden.
Unterstrichen	Unterstrichene Worte dienen dazu, Stellen fuer vom Nutzer festzulegende Worte, Zahlen oder Ausdruecke zu markieren, z.B. fuer Datensatz- oder Feldnamen und Konstanten.
[...]	Wird ein Wort oder ein Ausdruck in eckige Klammern eingeschlossen, ist es/er optionell und kann auch weggelassen werden.

|...| Ist eine Liste von Optionen in senkrechte Striche eingeschlossen, soll genau eine dieser Optionen gewaehlt werden.

davor stehende Element beliebig oft wiederholt werden kann.

6.3.3.1 Anfragekommandos

Ein SQL-Anfragekommando besteht aus einem oder mehreren Anfragebloecken (query blocks), die durch Schraegstrich (/) abgeschlossen werden. Ein Abfrageblock beginnt immer mit der Klausel select, die angibt, dass Felder oder Ausdruecke extrahiert werden sollen. Dann folgt die Klausel from, die angibt, aus welchem Datensatztyp die Felder extrahiert werden sollen. Es ist moeglich, dass danach keine, eine oder mehrere optionelle Klauseln folgen. Die Klauseln muessen in der angegebenen Reihenfolge auftreten, auch wenn einige von ihnen in einer speziellen Anfrage nicht verwendet werden.

```
select ....
from ....
[ where .... ]
[ group by .... ]
[ having .... ]
[ order by .... ]
[ into .... ]
/
```

Die Punkte wurden verwendet, um die Fortsetzung der Klausel anzuzeigen. Im folgenden soll beschrieben werden, was das fuer die jeweiligen Klauseln bedeutet.

Die Klausel select

```
select [unique]  | *            |, | datsat.*     |, ...
                 | datsat.*     |  | datsat.feld  |
                 | feld         |  | konstante    |
                 | konstante    |  | count (*)    |
                 | count (*)    |  | min (ausdr)  |
                 | min (ausdr)  |  | max (ausdr)  |
                 | max (ausdr)  |  | sum (ausdr)  |
                 | sum (ausdr)  |  | avg (ausdr)  |
                 | avg (ausdr)  |  | ausdr        |
                 | ausdr        |  |              |
```

Mit der Klausel select kann man angeben, welche Felder oder Ausdruecke durch die Abfrage extrahiert werden sollen. Jede Anfrage muss mit einer select-Klausel beginnen.

unique Das Schluesselwort unique wird verwendet, um doppelte Zeilen aus dem Ergebnis der Abfrage zu

eliminieren. Stimmen in einer oder mehreren Zeilen alle Spalten ueberein, wird im Ergebnis nur eine Zeile angezeigt. Die Verwendung dieses Schluesselwortes fuehrt zur Sortierung der Abfrageausgabe.

`*` Das Sternchen bedeutet, dass alle Felder aller in der Klausel from angefuehrten Datensatztypen zu waehlen sind. Wird Stern verwendet, duerfen in der Klausel select keine anderen Felder oder Ausdruecke stehen.

`datsat.*` Ist der Name eines in der Datenbank befindlichen Datensatztyps plus Punkt und Sternchen. Erfolgt diese Angabe, werden alle Felder des angegebenen Datensatztyps ausgewaehlt.

`datsat.feld` Ist der Name eines Feldes, dem der Name des Datensatztyps vorangestellt ist, in dem dieses Feld auftritt. Wurde dem Datensatztyp in der Klausel from ein Name zugeordnet, kann dieser Name anstelle des Datensatznamens verwendet werden. Es ist erforderlich, ein Feld auf diese Weise zu bezeichnen, wenn man einen Feldnamen verwendet, der in zwei oder mehr verschiedenen Datensatztypen verwendet wird, die in der Klausel from auftreten. Sonst waere unklar, aus welchem Datensatztyp das Feld zu holen ist.

`feld` Der Name eines Feldes, das in einem der in der Klausel from aufgefuehrten Datensaetze auftritt.

`konstante` Ein konstanter Wert vom Typ NUMERIC, STRING, DATE, TIME, AMOUNT oder FLOAT. Welcher Typ in einer gegebenen Situation zu verwendenden ist, haengt vom Typ des Feldes oder des Ausdruckes ab, mit dem die Konstante verglichen wird. Zeichenkettenkonstanten, STRING, werden in Apostrophe (') eingeschlossen, um sie von Feldnamen zu unterscheiden.

`count (*)`
`min (ausdr)`
`max (ausdr)`
`avg (ausdr)`
`sum (ausdr)` Sind eingebaute numerische SQL-Funktionen. Verwendet man eine numerische Funktion, ist die Ausfuehrung einer Klausel 'group by' implizit eingeschlossen. Wurde die Gruppe nicht explizit angegeben, wird das gesamte Abfrageergebnis als eine einzige Gruppe behandelt. Daraus ergibt sich auch, dass man nur Felder oder Ausdruecke auswaehlen kann, deren Werte allen Datensaetzen der Gruppe gemeinsam sind. Die Funktion count(*) zaehlt in jeder Gruppe die Datensaetze, die den Auswahlkriterien der Anfrage entsprachen. Die Funktionen min, max und avg berechnen fuer die Datensaetze in jeder Gruppe, die den Auswahlkriterien der Anfrage

entsprachen, Minimal-, Maximal- bzw. Durchschnittswerte der angegebenen Ausdruecke. Die Funktion sum addiert fuer die Datensaetze in jeder Gruppe, die den Auswahlkriterien der Anfrage entsprachen, den angegebenen Ausdruck. Diese Funktionen koennen auf Felder vom Typ NUMERIC, AMOUNT, FLOAT, DATE und TIME angewandt werden, obwohl die Ergebnisse der Felder DATE und TIME wahrscheinlich bedeutungslos sind. Numerische Funktionen koennen auch verschachtelt werden, um Ausdruecke wie avg(count(*)) und min(avg(grosshand_preis)) zu berechnen. Die erlaubte Schachtelungstiefe ist gleich 2.

ausdr Ist ein arithmetischer Ausdruck, der aus Feldern, Konstanten und numerischen Funktionen bestehen kann, die durch die Operationszeichen +, -, * und / verbunden sind. Klammern werden verwendet, um die Reihenfolge anzugeben, in der die Operationen ausgefuehrt werden.

Die Klausel from

from | datsat [name] |, ...

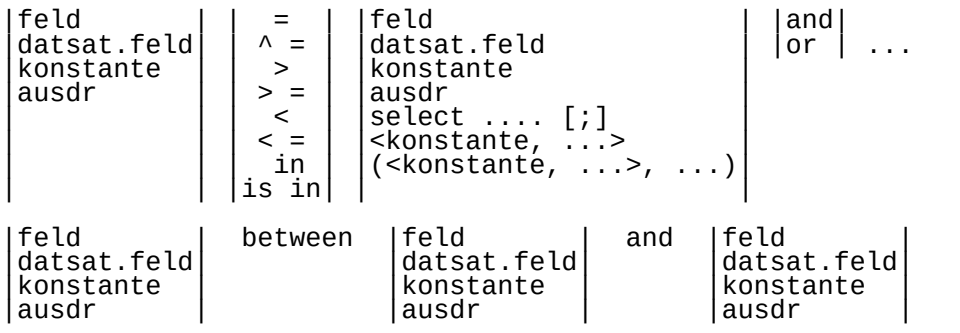
Mit der Klausel from kann man festlegen, welche Datensatztypen in der Abfrage zu verwenden sind. Man kann eine beliebige Anzahl von Datensaetzen in beliebiger Reihenfolge auflisten und SQL loest das Problem, wie die Daten geholt werden.

datsat Der Name eines in der aktuellen Datenbank befindlichen Datensatztyps.

name Ein Name, der benutzt wird, um den Datensatztyp bei spaeterer Verwendung in der Anfrage zu kennzeichnen. Namen werden in Gleich-Verbund-Anfragen (siehe Abschnitt 6.1.11.2 und variablen Anfragen (siehe Abschnitt 6.1.12) verwendet.

Die Klausel where

where [not]



Mit der Klausel where kann man Auswahlkriterien festlegen, mit denen man einzelne Datensaetze in einem Anfrageergebnis auswaehlen oder ausschliessen kann. Die oben angefuehrte Notation bedeutet, dass eine Klausel where aus einem oder mehreren Booleschen Ausdruecken besteht, die mit and bzw. or, verbunden sind. Jeder Boolesche Ausdruck kann in einer der drei angegebenen Formen auftreten. Es koennen dabei entweder die Vergleichsoperatoren oder das Schluesselwort between - and verwendet werden. Enthaelt eine where-Klausel mehrere Boolesche Ausdruecke, die mit and oder mit or miteinander verbunden sind, koennen eckige Klammern verwendet werden, um anzugeben, welcher Ausdruck zuerst berechnet werden soll.

feld — Der Name eines Feldes, das in einem der in der Klausel from aufgefuehrten Datensatztypen auftritt.

datsat.feld — Ist der Name eines Feldes, dem der Name des Datensatztyps vorangestellt ist, in dem dieses Feld auftritt. Wurde dem Datensatztyp in der Klausel from ein Name zugeordnet, kann dieser Name anstelle des Datensatznamens verwendet werden. Es ist erforderlich, ein Feld auf diese Weise zu bezeichnen, wenn man einen Feldnamen verwendet, der in zwei oder mehr verschiedenen Datensatztypen verwendet wird, die in der Klausel from auftreten. Sonst waere unklar, aus welchem Datensatztyp das Feld zu holen ist.

konstante — Ein konstanter Wert vom Typ NUMERIC, STRING, DATE, TIME, AMOUNT oder FLOAT. Welcher Typ in einer gegebenen Situation zu verwendenden ist,

haengt vom Typ des Feldes oder des Ausdrueckes ab, mit dem die Konstante verglichen wird. Zeichenkettenkonstanten, STRING, werden in Apostrophe (') eingeschlossen, um sie von Feldnamen zu unterscheiden.

ausdr Ist ein arithmetischer Ausdruck, der aus Feldern, Konstanten und numerischen Funktionen bestehen kann, die durch die Operationszeichen +, -, * und / verbunden sind. Klammern werden verwendet, um die Reihenfolge anzugeben, in der die Operationen ausgefuehrt werden.

select .... Diese Notation gibt an, dass in diesem Teil der Anweisung ein (weiterer) Anfrageblock verwendet werden kann. Weitere Informationen am Anfang dieses Abschnitts.

<konstante, ...>
Diese Notation gibt an, dass eine Liste von Konstanten (Datentupel) anstelle eines einzelnen Wertes verwendet werden kann, wenn mit der linken Seite eines Booleschen Ausdrucks auf Gleichheit verglichen wird. Ein Datentupel ist eine Liste von Konstanten, die untereinander durch Kommas getrennt und in winklige Klammern eingeschlossen sind. Haeufig wird ein solcher Ausdruck etwa so verwendet:

```
feld is in <k1, k2, k3>
```

Dieser Ausdruck ist aequivalent zu dem umstaendlicheren Ausdruck:

```
feld = k1 or feld = k2 or feld = k3
```

(<konstante, ...>, ...)
Diese Notation zeigt an, dass eine Liste von Datentupeln verwendet werden kann, wenn auf Gleichheit mit der linken Seite eines Booleschen Ausdrucks verglichen wird. Jedes Tupel ist wie oben aufgebaut, die Tupel werden untereinander durch Kommas getrennt und die gesamte Liste wird in runde Klammern eingeschlossen:

```
<f1, f2> is in (<k1, k2>, <k3, k4>)
```

Dieser Ausdruck ist aequivalent zu dem umstaendlicheren Ausdruck:

```
[f1 = k1 and f2 = k2] or [f1 = k3 and f2 = k4]
```

=, ^=, >, >=, <, <=
Das sind die Standard-Vergleichsoperatoren. Sie bedeuten: gleich, ungleich, groesser als, groesser als oder gleich, kleiner als, kleiner

als oder gleich.

in
is in — Diese beiden Operatoren sind dem Standard-Gleichheitszeichen aequivalent. Sie wurden eingefuehrt, damit man Abfragen mit Datentupeln besser lesen kann. Man kann stattdessen das normale Gleichheitszeichen verwenden.

between - and
Es handelt sich um einen einzelnen Operator,der dazu dient, dass man Wertebereiche uebersichtlicher festlegen kann. Normalerweise wird ein Bereich in der Anfragesprache folgendermassen festgelegt:

```
feld <= k1 and feld >= k2
```

Unter Verwendung dieses Operators lautet der Ausdruck einfach:

```
feld between k1 and k2
```

and — Ist der Standardoperator fuer die UND-Verknuepfung. Der Wert eines zusammengesetzten Booleschen Ausdrucks, der aus zwei durch UND miteinander verbundenen einfachen Booleschen Ausdruecken besteht, ist wahr, wenn beide einfache Ausdruecke wahr sind und ist falsch, wenn einer der beiden einfachen Ausdruecke falsch ist.

or — Ist der Standardoperator fuer die ODER-Verknuepfung. Der Wert eines zusammengesetzten Booleschen Ausdrucks, der aus zwei durch ODER miteinander verbundenen einfachen Booleschen Ausdruecken besteht, ist wahr, wenn einer der einfachen Ausdruecke wahr ist und ist falsch, wenn beide einfachen Ausdruecke falsch ist.

Die Klausel 'group by'

```
group by | feld       |, ...
         | datsat.feld |
```

Mit der Klausel group by kann man Datensaetze in Gruppen zusammenfassen, so dass man auf jede Gruppe eine numerische Funktion anwenden kann. Verwendet man entweder in der Klausel select oder in der Klausel having eine numerische Funktion, wird immer mindestens eine Gruppe gebildet. Wird die Gruppe nicht durch diese Klausel festgelegt, wird das gesamte Abfrageergebnis als eine einzige Gruppe behandelt. Die Verwendung dieser Klausel bedeutet auch, dass jedes Feld oder jeder Ausdruck in der Klausel select fuer jeden Datensatz derselben Gruppe denselben Wert hat. Obwohl kein Syntaxfehler entsteht, wenn man Felder oder Ausdruecke

auflistet, deren Werte fuer jeden Datensatz in der Gruppe verschieden sind, erhaelt man aber auch kein sinnvolles Anfrageergebnis.

feld Der Name eines Feldes, das in einem der in der Klausel from aufgefuehrten Datensatztypen auftritt.

datsat.feld Ist der Name eines Feldes, dem der Name des Datensatztyps vorangestellt ist, in dem dieses Feld auftritt. Wurde dem Datensatztyp in der Klausel from ein Name zugeordnet, kann dieser Name anstelle des Datensatznamens verwendet werden. Es ist erforderlich, ein Feld auf diese Weise zu bezeichnen, wenn man einen Feldnamen verwendet, der in zwei oder mehr verschiedenen Datensatztypen verwendet wird, die in der Klausel from auftreten. Sonst waere unklar, aus welchem Datensatztyp das Feld zu holen ist.

Die Klausel having

```
|feld        |   |  =   |   |feld                          |   |and|
|datsat.feld |   | ^ =  |   |datsat.feld                   |   |or | ...
|konstante   |   |  >   |   |konstante                     |
|ausdr       |   | > =  |   |ausdr                         |
|count (*)   |   |  <   |   |count (*)                     |
|min (ausdr) |   | < =  |   |min (ausdr)                   |
|max (ausdr) |   |  in  |   |max (ausdr)                   |
|sum (ausdr) |   |is in |   |sum (ausdr)                   |
|avg (ausdr) |                |avg (ausdr)                   |
                                |select .... [;]               |
                                |<konstante, ...>              |
                                |(<konstante, ...>, ...)       |

|feld        |  between  |feld        |  and  |feld        |
|datsat.feld |           |datsat.feld |       |datsat.feld |
|konstante   |           |konstante   |       |konstante   |
|ausdr       |           |ausdr       |       |ausdr       |
|count (*)   |           |count (*)   |       |count (*)   |
|min (ausdr) |           |min (ausdr) |       |min (ausdr) |
|max (ausdr) |           |max (ausdr) |       |max (ausdr) |
|sum (ausdr) |           |sum (ausdr) |       |sum (ausdr) |
|avg (ausdr) |           |avg (ausdr) |       |avg (ausdr) |
```

Die oben angefuehrte Notation bedeutet, dass eine Klausel having aus einem oder mehreren Booleschen Ausdruecken besteht, die mit and bzw. or verbunden sind. Jeder Boolesche Ausdruck kann in einer der drei angegebenen Formen auftreten. Es koennen dabei entweder die Vergleichsoperatoren oder der Operator between - and verwendet werden. Enthaelt eine having-Klausel mehrere Boolesche Ausdruecke, die entweder mit and oder mit or miteinander verbunden sind, koennen eckige Klammern verwendet werden, um anzugeben, welcher Ausdruck zuerst berechnet werden soll.

Mit dieser Klausel kann man auf der Grundlage einer numerischen Funktion einige der Gruppen auswaehlen, die von der Klausel 'group by' gebildet wurden und andere zurueckweisen. Deshalb muss die Klausel having immer mindestens eine numerische Funktion enthalten und es muss ihr eine Klausel 'group by' vorangestellt sein.

feld — Der Name eines Feldes, das in einem der in der Klausel from aufgefuehrten Datensaetze auftritt.

datsat.feld — Ist der Name eines Feldes, dem der Name des Datensatztyps vorangestellt ist, in dem dieses Feld auftritt. Wurde dem Datensatztyp in der Klausel from ein Name zugeordnet, kann dieser Name anstelle des Datensatznamens verwendet werden. Es ist erforderlich, ein Feld auf diese Weise zu bezeichnen, wenn man einen Feldnamen verwendet, der in zwei oder mehr verschiedenen Datensatztypen verwendet wird, die in der Klausel from auftreten. Sonst waere unklar, aus welchem Datensatztyp das Feld zu holen ist.

konstante — Ein konstanter Wert vom Typ NUMERIC, STRING, DATE, TIME, AMOUNT oder FLOAT. Welcher Typ in einer gegebenen Situation zu verwendenden ist, haengt vom Typ des Feldes oder des Ausdrueckes ab, mit dem die Konstante verglichen wird. Zeichenkettenkonstanten, STRING, werden in Apostrophe (') eingeschlossen, um sie von Feldnamen zu unterscheiden.

ausdr — Ist ein arithmetischer Ausdruck, der aus Feldern, Konstanten und numerischen Funktionen bestehen kann, die durch die Operationszeichen +, -, * und / verbunden sind. Klammern werden verwendet, um die Reihenfolge anzugeben, in der die Operationen ausgefuehrt werden.

count (*)
min (ausdr)
max (ausdr)
avg (ausdr)
sum (ausdr) — Sind eingebaute numerische SQL-Funktionen. Verwendet man eine numerische Funktion, ist die Ausfuehrung einer Klausel 'group by' implizit eingeschlossen. Wurde die Gruppe nicht explizit angegeben, wird das gesamte Abfrageergebnis als eine einzige Gruppe behandelt. Daraus ergibt sich auch, dass man nur Felder oder Ausdruecke auswaehlen kann, deren Werte allen Datensaetzen der Gruppe gemeinsam sind. Die Funktion count(*) zaehlt in jeder Gruppe die Datensaetze, die den Auswahlkriterien der Anfrage entsprachen. Die Funktionen min, max und avg berechnen fuer die Datensaetze in jeder Gruppe, die den Auswahlkriterien der Anfrage entsprachen, Minimal-, Maximal- bzw. Durchschnittswerte der angegebenen Ausdruecke. Die Funktion sum addiert fuer die Datensaetze

in jeder Gruppe, die den Auswahlkriterien der Anfrage entsprachen, den angegebenen Ausdruck. Diese Funktionen koennen auf Felder vom Typ NUMERIC, AMOUNT, FLOAT, DATE und TIME angewandt werden, obwohl die Ergebnisse der Felder DATE und TIME wahrscheinlich bedeutungslos sind. Numerische Funktionen koennen auch verschachtelt werden, um Ausdruecke wie avg(count(*)) und min(avg(grosshand_preis)) zu berechnen. Die erlaubte Schachtelungstiefe ist gleich 2.

select .... Diese Notation gibt an, dass in diesem Teil der Anweisung ein (weiterer) Anfrageblock verwendet werden kann. Weitere Informationen am Anfang dieses Abschnitts.

<konstante, ...>
Diese Notation gibt an, dass eine Liste von Konstanten (Datentupel) anstelle eines einzelnen Wertes verwendet werden kann, wenn mit der linken Seite eines Booleschen Ausdrucks auf Gleichheit verglichen wird. Ein Datentupel ist eine Liste von Konstanten, die untereinander durch Kommas getrennt und in winklige Klammern eingeschlossen sind. Haeufig wird ein solcher Ausdruck etwa so verwendet:

```
feld is in <k1, k2, k3>
```

Dieser Ausdruck ist aequivalent zu dem umstaendlicheren Ausdruck:

```
feld = k1 or feld = k2 or feld = k3
```

(<konstante, ...>, ...)
Diese Notation zeigt an, dass eine Liste von Datentupeln verwendet werden kann, wenn auf Gleichheit mit der linken Seite eines Booleschen Ausdrucks verglichen wird. Jedes Tupel ist wie oben aufgebaut, die Tupel werden untereinander durch Kommas getrennt und die gesamte Liste wird in runde Klammern eingeschlossen:

```
<f1, f2> is in (<k1, k2>, <k3, k4>)
```

Dieser Ausdruck ist aequivalent zu dem umstaendlicheren Ausdruck:

```
[f1 = k1 and f2 = k2] or [f1 = k3 and f2 = k4]
```

=, ^=, >, >=, <, <=
Das sind die Standard-Vergleichsoperatoren. Sie bedeuten: gleich, ungleich, groesser als, groesser als oder gleich, kleiner als, kleiner als oder gleich.

in
is in — Diese beiden Operatoren sind dem Standard-Gleichheitszeichen aequivalent. Sie wurden eingefuehrt, damit man Abfragen mit Datentupeln besser lesen kann. Man kann stattdessen das normale Gleichheitszeichen verwenden.

between - and
Es handelt sich um einen einzelnen Operator,der dazu dient, dass man Wertebereiche uebersichtlicher festlegen kann. Normalerweise wird ein Bereich in der Anfragesprache folgendermassen festgelegt:

```
feld <= k1 and feld >= k2
```

Unter Verwendung dieses Operators lautet der Ausdruck einfach:

```
feld between k1 and k2
```

and — Ist der Standardoperator fuer die UND-Verknuepfung. Der Wert eines zusammengesetzten Booleschen Ausdrucks, der aus zwei durch UND miteinander verbundenen einfachen Booleschen Ausdruecken besteht, ist wahr, wenn beide einfache Ausdruecke wahr sind und ist falsch, wenn einer der beiden einfachen Ausdruecke falsch ist.

or — Ist der Standardoperator fuer die ODER-Verknuepfung. Der Wert eines zusammengesetzten Booleschen Ausdrucks, der aus zwei durch ODER miteinander verbundenen einfachen Booleschen Ausdruecken besteht, ist wahr, wenn einer der einfachen Ausdruecke wahr ist und ist falsch, wenn beide einfachen Ausdruecke falsch ist.

Die Klausel 'order by'

```
order by  | feld       [|asc |] |, ...
          | datsat.feld  |desc|  |
```

Mit dieser Klausel kann man die Ausgabe einer Anfrage sortieren. Man kann fuer jedes Feld festlegen, ob in steigender (asc) oder fallender Ordnung (desc) sortiert werden soll. Wird die Klausel 'order by'nicht angegeben, die (Standard-)Sortierordnung aufsteigend.

feld — Der Name eines Feldes, das in einem der in der Klausel from aufgefuehrten Datensaetze auftritt.

datsat.feld — Ist der Name eines Feldes, dem der Name des Datensatztyps vorangestellt ist, in dem dieses Feld auftritt. Wurde dem Datensatztyp in der

Klausel from ein Name zugeordnet, kann dieser Name anstelle des Datensatznamens verwendet werden. Es ist erforderlich, ein Feld auf diese Weise zu bezeichnen, wenn man einen Feldnamen verwendet, der in zwei oder mehr verschiedenen Datensatztypen verwendet wird, die in der Klausel from auftreten. Sonst waere unklar, aus welchem Datensatztyp das Feld zu holen ist.

asc Diese Schluesselworte werden zur Festlegung der Reihenfolge verwendet, in der das davor angegebene Feld sortiert werden soll. Wuenscht man eine aufsteigende Ordnung, wird asc verwendet, und wuenscht man eine abfallende Ordnung wird desc verwendet.

Die KLausel into

into datei_name

Diese Klausel wird verwendet, um das Ergebnis einer Anfrage nicht an den Bildschirm, sondern stattdessen an eine WEGA-Datei zu senden. Das ist insbesondere dann sinnvoll, wenn man die Daten in einem anderen Programm verwenden oder in eine andere Datenbank zurueckladen moechte.

datei_name Der Name einer WEGA-Datei, in der die Ergebnisse der aktuellen Anfrage erstellt und gespeichert werden sollen. Existiert die Datei bereits, handelt es sich um einen Fehler. Enthaelt der datei_name Sonderzeichen, wie einen Punkt oder Schraegstrich (z.B. ../xxx.out), muss er in Apostrophe eingeschlossen werden.

6.3.3.2 Anweisungen zur Datenmanipulation (DML)

Eine SQL-Datenmanipulationsanweisung besteht aus einem Schluesselwort, das den Typ der auszufuehrenden Aktualisierung angibt, d.h. ob es sich um eine Einfuegung, eine Aktualisierung oder ein Loeschen handelt. Hinter dem Schluesselwort folgen die Parameter der Aktualisierung. DML-Anweisungen werden wie Anfrageanweisungen mit einem Schraegstrich (/) abgeschlossen. Bei der Datenmanipulation koennen zum grossen Teil die Moeglichkeiten der Anfrageanweisungen verwendet werden. Das hat zur Folge, dass man mit einer einzigen Anweisung einen ganzen Satz von Datensaetzen bearbeiten kann. Im folgenden wird die Syntax der einzelnen Datenmanipulationsanweisungen genauer erklaert.

Die Klausel insert

```
insert into datsat [(feld, ...)]: | from datei_name       | /
                                  | <konstante, ...>, ... |
                                  | select ....           |
```

Mit der Klausel 'insert into' kann man neue Datensaetze in eine Datenbank bringen. Zum Erhalt der Daten fuer die neuen Datensaetze kann man eine ASCII-Datei, Datentupel oder die Ergebnisse einer Anfrage verwenden.

datsat Der Name eines in der aktuellen Datenbank vorhandenen Datensatztyps.

(feld, ...) Ist die optionelle Feld-Spezifikationsliste. Sie besteht aus einer Liste von Feldnamen, die durch Kommas getrennt und in Klammern eingeschlossen sind. Die Liste legt fest, welche Felder zu laden sind und in welcher Reihenfolge sie im Eingabestrom erscheinen, ob dieser Strom von einer ASCII-Datei, Datentupeln oder einer Anfrageanweisung kommt. Wird die Liste weggelassen, wird davon ausgegangen, dass der Eingabestrom die von der Anfrage

```
select * from datsat /
```

zurueckgegebenen Felder enthaelt und auch in genau dieser Reihenfolge. Das ist die Standard-Feldordnung fuer einen Datensatztyp.

datei_name Der Name einer WEGA-Datei, die die in die Datenbank zu ladenen Daten enthaelt. Enthaelt der datei_name Sonderzeichen, wie einen Punkt oder Schraegstrich (z.B. ../xxx.out), muss er in Apostrophe eingeschlossen werden. Die Felder muessen innerhalb der Datei entweder in der gleiche Reihenfolge auftreten, wie in der Spezifikationsliste oder wie in der Standard-Feldordnung fuer den Datensatztyp, der geladen wird.

<konstante, ...>
Diese Notation gibt an, dass eine Liste von Konstanten (Datentupel), bzw. Liste von Datentupeln, verwendet werden kann, um die Werte der Felder in der Feldspezifikationsliste anzugeben. Die Felder muessen entweder in der gleiche Reihenfolge auftreten, wie in der Spezifikationsliste oder wie in der Standard-Feldordnung fuer den Datensatztyp, der geladen wird.

select .... Diese Notation gibt an, dass in diesem Teil der Anweisung ein (weiterer) Anfrageblock verwendet werden kann. Weitere Informationen in Abschnitt 6.3.3.1.

Die Klausel update

update datsat set .... [where ....] /

Diese Klausel aktualisiert die Felder in den Datensaetzen, die den Auswahlkriterien in der where-Klausel entsprechen. Wird die Klausel where nicht angegeben, werden alle Datensaetze aktualisiert.

datsat Der Name eines Datensatztyps in der aktuellen Datenbank.

set Hier wurde die Klausel set eingefuegt. Diese wird im folgenden Abschnitt naeher erklaert.

where Es kann optionell eine where-Klausel angegeben werden, um festzulegen, welche Datensaetze zu aktualisieren sind. Weitere Informationen siehe Abschnitt 6.3.3.1.

Die Klausel set

set | feld = ausdr | , ...
| feld = select ... |

Mit dieser Klausel kann man angeben, wie die Felder in einem Datensatz zu aktualisieren sind. Man kann einem Feld entweder einen Ausdruck oder das Ergebnis einer Anfrage zuordnen.

feld Der Name eines Feldes in dem Datensatztyp, der aktualisiert wird.

ausdr Ist ein arithmetischer Ausdruck, der aus Feldern und Konstanten bestehen kann, die durch die Operationszeichen +, -, * und / verbunden sind. Die Felder muessen sich in dem Datensatztyp befinden, der aktualisiert wird. Runde Klammern werden verwendet, um die Reihenfolge anzugeben, in der die Operationen ausgefuehrt werden.

select .... Diese Notation gibt an, dass in diesem Teil der Anweisung ein (weiterer) Anfrageblock verwendet werden kann. Weitere Informationen in Abschnitt 6.3.3.1.

Die Klausel delete

delete datsat [where ....] /

Mit dieser Klausel kann man in einer Datei auf der Grundlage der Ergebnisse einer where-Klausel bestimmte Datensaetze loeschen. Wird die where-Klausel nicht angege-

ben, werden alle Datensaetze in der Datei geloescht.

datsat Der Name eines Datensatztyps in der aktuellen Datenbank.

where Es kann eine optionelle where-Klausel verwendet werden, um die zu loeschenden Datensaetze zu bezeichnen. Weitere Informationen ueber die where-Klausel siehe Abschnitt 6.3.3.1.

6.3.3.3 Zusaetzliche Kommandos

WEGA-DATA-SQL umfasst eine Reihe von Kommandos, die nicht eigentlicher Bestandteil der Sprache sind. Mit diesen Kommandos kann man die aktuelle SQL-Anweisung editieren, die aktuelle Anweisung noch einmal starten, in Textdateien gespeicherte SQL-Anweisungen starten, durch Passworte geschuetzte Felder freigeben, fuer die Spaltenueberschriften die Anzahl der Zeilen pro Seite festlegen, das Standard-Trennzeichen zum Trennen von Feldern aendern, WEGA-Kommandos ausfuehren und aus SQL beenden. Im folgenden werden die Kommandos im einzelnen erklaert.

Das Kommando edit

edit

Die aktuelle SQL-Anweisung wird im Verzeichnis /tmp in einer temporaeren Datei gespeichert. Mit diesem Kommando kann man die Anweisung editieren, um Syntaxfehler zu berichtigen, oder sie modifizieren, um ein anderes Ergebnis zu erhalten. Der verwendete Standard-Editor ist vi. Man kann diesen jedoch aendern, indem man die Umgebungsvariable EDIT setzt. Weitere Informationen zu Umgebungsvariablen siehe Abschnitt 1.1.3.

Das Kommando restart

restart

Mit diesem Kommando kann man die in der temporaeren Datei gespeicherte aktuelle SQL-Anweisung ausfuehren. Immer wenn eine neue Anweisung eingegeben wird, ersetzt diese die alte und wird zur aktuellen Anweisung. Dieses Kommando wird meist nach dem Editieren der aktuellen Anweisung verwendet.

Das Kommando start

start datei_name

Das Kommando fuehrt die in der WEGA Datei namens datei_name gespeicherte SQL-Anweisung aus. Wird eine Anweisung auf

diese Weise ausgefuehrt, wird sie dadurch nicht zur aktuellen Anweisung. Die aktuelle Anweisung ist nach wie vor die letzte vom Nutzer eingegebene.

datei_name Der Name einer WEGA-Datei, die eine SQL-Anweisung enthaelt. Die Anweisung wird ausgefuehrt und die Ergebnisse werden auf dem Bildschirm angezeigt. Enthaelt der Dateiname ausser der Unterstreichung noch andere Sonderzeichen (z.B. ../anfrage), muss er in Apostrophe eingeschlossen werden, um zu verhindern, dass SQL diese Zeichen als Sonderzeichen auffasst und entsprechend bearbeitet.

Das Kommando unlock

unlock | datsat.feld, passwort |, ... /

Mit diesem Kommando kann man Datenbank-Felder freigeben, die mit dem Programm 'Field Level Security' gegen unbefugten Zugriff geschuetzt wurden (Abschnitt 3.3.1). Es ist zu beachten, dass dieses Kommando durch Schraegstrich abzuschliessen ist.

datsat.feld Der Name der Datenbank-Feldes, das freigegeben werden soll. Seinem Namen muss der Name des Datensatztyps vorangestellt werden, dessen Bestandteil das Feld ist.

passwort Das Passwort, das in 'Field Level Security' zum Sperren dieses Feldes eingegeben wurde.

Das Kommando lines

lines nummer

Mit diesem Kommando kann man bestimmen, nach wieviel Zeilen eine neue Ueberschrift ausgegeben werden soll. Standard sind 24 Zeilen, da das der Groesse der meisten Bildschirme entspricht. Das heisst, dass jeweils nach 24 Zeilen eine neue Ueberschrift ausgegeben wird.

nummer Eine Zahl zwischen 0 und 32767, mit der angegeben wird, nach wieviel Zeilen eine neue Ueberschrift ausgegeben werden soll. Ist nummer gleich 0, wird keine Ueberschrift ausgegeben. Ansonsten wird die Ueberschrift am Anfang ausgegeben und dann nach jeweils nummer Zeilen der Ausgabe.

Das Kommandos separator

separator 'zeichen'

Mit diesem Kommando kann man das von SQL verwendete Feld-Trennzeichen aendern, mit dem angegeben wird, wo ein Feld endet und das naechste beginnt. Standard ist der senkrechte Strich (|).

zeichen Ein einzelnes Zeichen, das das neue Trennzeichen ist. Es kann ein beliebiges Zeichen sein, auch ein nichtdruckbares Steuerzeichen. Soll unter Verwendung der Klausel insert eine ASCII-Datei geladen werden, muss innerhalb der gesamten ASCII-Datei dasselbe Feld-Trennzeichen verwendet werden, und das SQL-Feld-Trennzeichen muss mit dem in der zu ladenden Datei verwendeten Trennzeichen uebereinstimmen.

WEGA-Kommandos

! WEGA_kommando

Alle Kommandos, vor denen ein Ausrufungszeichen steht, werden zur Ausfuehrung an die WEGA-Shell gegeben. Nach Abarbeitung des Kommandos uebernimmt SQL wieder die Steuerung.

Das Kommando end

end

Will man SQL beenden und (in Abhaengigkeit davon, wie das Programm begonnen wurde) entweder in den Menue-Handler oder in die WEGA-Shell zurueckkehren, braucht man nur hinter dem SQL-Prompter end einzugeben.

6.3.3.4 Hilfe-Kommandos

Ein SQL-Hilfekommando (help) besteht aus einem oder zwei Schluesselworten, wobei das zweite angibt, zu welchem Problem eine Hilfe-Information gewuenscht wird. Die Hilfekommandos sollen Auskunft ueber die Syntax einer bestimmten Anweisung oder eines Kommandos geben. In den folgenden Abschnitten wird beschrieben, wie die SQL-Hilfekommandos zu verwenden sind.

Das Kommando help

help [schluewo]

Durch diese Anweisung wird auf dem Terminal eine Hilfe-Information ueber die Verwendung von SQL ausgegeben. Gibt man ueber Tastatur nur help ein, erscheint eine allgemeine Information ueber die Verwendung der Hilfemoeglichkeiten. Gibt man dagegen help schluewo ein, werden Hilfsinformatio-

nen zu diesem speziellen Schluesselwort schluewo angezeigt.

schluewo Eines der von SQL verwendeten Schluesselworte. Existiert zu diesem Schluesselwort eine Hilfedokumentation, wird diese ausgegeben. Die Schluesselworte sind in Abschnitt 6.3.1 aufgelistet.

Das Kommando records

records

Durch dieses Kommando werden die Namen aller Datensatztypen aufgelistet, die fuer den Aufbau von SQL-Anweisungen verwendet werden koennen. Fuer dieses Kommando gibt es keine Optionen.

Das Kommando fields

fields datsatyp

Durch dieses Kommando werden alle Felder des angegebenen Datensatztyps sowie deren Typ und Laenge aufgelistet.

datsatyp Der Name eines Datensatztyps in der aktuellen Datenbank. Ist kein Datensatztyp bekannt, wird erst das Hilfekommando records verwendet.

7. RPT - DER REPORTGENERATOR

In diesem Abschnitt soll der Reportgenerator von WEGA-DATA, RPT, beschrieben werden. Es erfolgt eine Aufgliederung in drei Teile. In Abschnitt 7.1 wird eine Einfuehrung in das Schreiben von Reporten und Hinweise gegeben, wie RPT zusammen mit den anderen Werkzeugen von WEGA-DATA verwendet werden kann. Im zweiten Teil, in den Abschnitten 7.2 und 7.3 werden Uebungsbeispiele angefuehrt, mit denen anhand der Uebungs-Datenbank die wichtigsten Eigenschaften von RPT erklaert werden. Im dritten Teil, ab Abschnitt 7.4 bis zum Ende des Kapitels, werden die Syntax, Schluesselworte und Fehlermeldungen von RPT beschrieben.

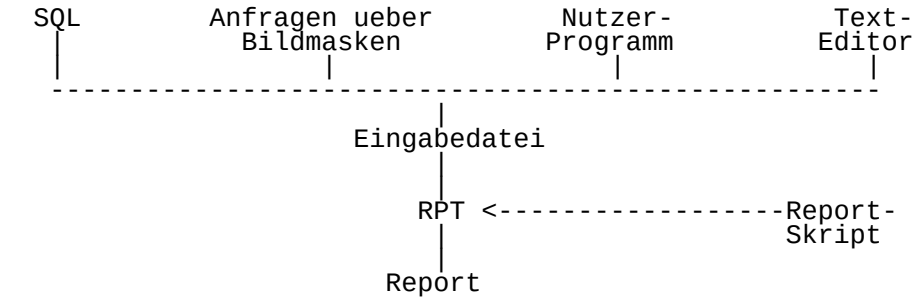
7.1 Konzepte der Reportgenerierung

Mit den Sprachelementen von RPT koennen in nichtprozeduraler Form einfach tabellarische Reporte erstellt werden. Die aufzubereitenden Daten koennen von SQL, von Anfragen ueber Bildmasken, von Nutzerprogrammen oder aus einer ASCII-Datei stammen. Es gibt grosse Freiheitsgrade bzgl. der Plazierung von Feldern, Spaltenueberschriften, Kopf- und Fusszeilen und der Seitengestaltung allgemein. Dadurch ergeben sich viele Moeglichkeiten der Reportgestaltung. Die beiden folgenden Abschnitte beschreiben grundsaetzlich die Erstellung eines RPT-Reports und den damit zusammenhaengenden Sprachgebrauch, der dann auch im ganzen restlichen Kapitel eingehalten wird.

7.1.1 Von RPT verwendete Dateien

Zur Erstellung eines Reports unter Verwendung von RPT sind mehrere Elemente erforderlich. Erstens ist eine Eingabedatei erforderlich, in der die Daten enthalten sind, die zu einen Report formatiert und zusammengefasst werden sollen. Die Eingabedatei ist eine normale ASCII-Datei, die aus verschiedenen Zeilen gleicher Struktur besteht. Jede Zeile besteht aus einer Anzahl von Datenfeldern und wird durch ein New-Line-Zeichen (<cr>) abgeschlossen. Zum Anlegen von Eingabedateien, kann man verwenden: SQL, Anfragen ueber Bildmasken, sed(1), awk(1), ein nutzerspezifisches Programm, einen Editor oder ein anderes Werkzeug zum Erstellen von Dateien.

Als naechstes wird ein Reportskript benoetigt, wobei es sich um eine Datei handelt, in der die Reportgenerator-Kommandos enthalten sind. Die Kommandos beschreiben das Format der Zeilen in der Eingabedatei und wie der fertige Report aussehen soll. Soll ein Report erzeugt werden, liest RPT dazu die Eingabedatei und benutzt die Kommandos im Reportskript. Im folgenden Diagramm wird die Beziehung zwischen Eingabedateien, Reportskripten und RPT gezeigt.



In Abschnitt 6.3.6 wird beschrieben, wie RPT mit SQL kombiniert werden kann, in den Abschnitten 5.1, 5.2 und 7.9.2 wird die Kombination von RPT und ENTER erlaeutert und in Abschnitt 7.9.3 die Kombination von RPT mit nutzerspezifischen Programmen und Shell-Skripten.

7.1.2 Beispiel-Report

Will man erlernen, wie ein Report geschrieben wird, muss man zunaechst die grundsaetzliche "Anatomie" eines RPT-Reportes und die zur Beschreibung der verschiedenen Teile verwendeten Begriffe kennen. Als "Arbeitsgegenstand" verwenden wir unsere altgediente Uebungsdatenbank. Es soll ein Report erzeugt werden, in dem alle Artikel im Lager zusammen mit einigen Kenngroessen und berechneten Werten ueber diese Kenngroessen in aufbereiteter Form ausgegeben werden.

Die Eingabedatei besteht aus einer Zeile fuer jeden Artikel im Lager, die die Seriennummer, das Erwerbsdatum, den Grosshandelspreis und die Bestellungsnummer des Artikels enthaelt. In Abschnitt 7.2 wird schrittweise das Report-Skript entwickelt, das eine "voll" aufbereitete Ausgabe der Artikel produziert.

Jedes Feld, nach dem sortiert wird, wird als Sortierschluessel bezeichnet. Die Sortierung erfolgt aus zwei Gruenden. Erstens sollen Datenelemente mit verschiedenen Sortierschluesseln in aufsteigender oder fallender Reihenfolge aufgefuehrt werden (z.B. Artikel, die an verschiedenen Tagen ins Lager kamen). Zweitens sollen Datenelemente mit demselben Sortierschluessel in Gruppen zusammengefasst werden (z.B. alle Artikel eines Tages). RPT laesst entweder einfache Datenbank-Felder oder einen komplexen Ausdruck, der aus Feldern, Konstanten, Variablen, arithmetischen Operatoren und aus Funktionen besteht, als Sortierschluessel zu.

Aendert sich der Wert eines Sortierschluessels, so tritt, einfach gesagt, eine Unterbrechung auf. Durch diesen "Gruppenwechsel" werden Gruppen von Datenelementen mit verschiedenen Sortierschluesseln voneinander getrennt. Wird

beispielsweise das Erwerbsdatum vom 2/15/86 auf den 2/16/86 geaendert, hat ein Gruppenwechsel stattgefunden. Bei RPT kann man Ausgabe- und Verarbeitungsmassnahmen festlegen, bevor Gruppen von Datenelementen durch Gruppenwechsel getrennt werden, oder auch danach.

Es soll beispielsweise davon ausgegangen werden, dass der Gruppenwechsel vom 15. zum 16. Februar stattgefunden hat. Hinter den in der Gruppe '02/15/87' aufgelisteten Datenelementen sollen die Summen fuer diesen Tag ausgegeben werden und vor der Gruppe '02/16/87' sollen fuer die folgenden Datenelemente erneut die Ueberschriften ausgegeben werden. Es wird auch als Gruppenwechsel bezeichnet, wenn man von "keine Daten" zu "Daten" uebergeht, wie das bei dem ersten Erwerbsdatum der Fall ist.

Gruppenwechsel treten aber nicht nur dann auf, wenn Sortierschluessel ihren Wert aendern, sondern es gibt noch fuenf weitere, "implizite", Arten von Gruppenwechseln. Das ist erstens der Gruppenwechsel detail, der immer dann auftritt, wenn eine neue Zeile aus der Eingabedatei gelesen wird. Es handelt sich hierbei um die niedrigste Gruppenwechselstufe, bei der man normalerweise eine Zeile ausgibt, einer Summe eine Zahl hinzufuegt oder beides. Am Anfang des Reports erfolgt immer ein Gruppenwechsel 'before report' und am Ende des Reports ein Gruppenwechsel 'after report'. Diese sind fuer die Ausgabe von Reportueberschriften oder -titeln bzw. zur Ausgabe der Gesamtsumme des ganzen Reports nuetzlich. Ausserdem gibt es die Gruppenwechsel header und footer, die auf jeder Reportseite oben bzw. unten erscheinen. Der Gruppenwechsel header kann z.B. fuer die Ausgabe fortlaufender Titel, Datumsangaben, Zeiten oder Seitennummern verwendet werden, wahrend der Gruppenwechsel footer haeufig fuer die Seitennummer oder eine kurze Notiz verwendet wird. Die Sektionen des Report-Skripts, die die bei Gruppenwechsel erforderlichen Massnahmen beschreiben, werden in folgender Reihenfolge verarbeitet:

```
before report
before <sort n>
     .
     .
     .
before <sort l>
detail
after <sort l>
     .
     .
     .
after <sort n>
after report
```

Die Gruppenwechsel header und footer koennen nicht "fest" in dieser Hierarchie untergebracht werden, da sie vor bzw. nach Ausgabe einer neuen Seite erfolgen. Damit sind sie

eindeutig von der Groesse der Seite abhaengig, d.h. mit anderen Worten, dass ein Report, der auf zwei Seiten unterschiedlicher Groesse ausgegeben wird, seine Gruppenwechsel header und footer an verschiedenen Stellen hat, auch wenn er in beiden Faellen genau dieselben Daten enthaelt. Die Verarbeitung der Gruppenwechsel header und footer erfolgt immer dann, wenn zu einer neuen Seite uebergegangen wird.

In den folgenden Abschnitten sollen die oben angefuehrten Begriffe durch Beispielreporte und die diese Reporte erzeugenden Report-Skripte genauer erklaert werden.

7.2 Einfache Beispielreporte

In den Abschnitten 7.2 und 7.3 werden Beispiele fuer Report-Skripts und ihr Ergebnis angefuehrt, um die wichtigsten Eigenschaften von RPT zu zeigen. In Abschnitt 7.2 wird die Entwicklung von zwei verschiedenen Reporten unter Verwendung verschiedener Zwischenschritte erklaert. In Abschnitt 7.3 werden kompliziertere Beispiele vorgestellt. Aufgrund der Leistungsfaehigkeit der Sprache ist es jedoch nicht moeglich, alle Methoden zu zeigen. Der Leser soll angeregt werden, 7.6, 7.7 und 7.8 zu studieren, in denen die Syntax der Sprache ausfuehrlich erlaeutert wird. Aufgrund dieser Kenntnis kann der Leser seine eigenen neuen Methoden zur Erstellung selbst der kompliziertesten Reporte entwickeln.

Alle in diesem Abschnitt angefuehrten Beispiele beruhen auf unserer sattsam bekannten Uebungsdatenbank.

7.2.1 Auflistungsbeispiel fuer Bestellungen

In diesem Abschnitt soll ein einfaches Report-Skript gezeigt werden, das in zwei Schritten erarbeitet wird. Es soll mit der einfachsten Version begonnen werden.

Mit einem ganz einfachen Report-Skript kann man einen Report erzeugen, in dem die in der Eingabedatei vorhandenen Felder ohne Formatierung, Ueberschriften oder Summenbildung aufgelistet werden. Es soll davon ausgegangen werden, dass in der Eingabedatei fuer jeden in der Datenbank eingetragenen Artikel Seriennummer, Erwerbsdatum, Bestellnummer und Grosshandelspreis enthalten sind. (In 7.3 wird gezeigt werden, wie man unter Verwendung von SQL eine Eingabedatei anlegt und wie man SQL und RPT miteinander verbindet.) Hier das Report-Skript, durch das der danach gezeigte Report erzeugt wird:

```
input
  art_seriennummer [numeric 9],
  art_erwerbsdatum [date],
  art_grosshand_preis [amount 5],
  art_bestellnummer [numeric 9]
```

```
detail
  print art_seriennummer, art_erwerbsdatum,
                      art_grosshand_preis, art_bestellnummer
end
```

Ergebnis:

```
          6 02/15/86      11.88       1
          5 02/15/86       9.75       1
          4 02/15/86       9.75       1
          3 02/15/86      16.55       1
          2 02/15/86      16.55       1
          1 02/15/86      16.55       1
          7 02/15/86      11.88       1
          8 02/16/86    4300.00       2
          9 02/16/86      16.20       1
         10 02/16/86      16.20       2
         11 02/18/86   89000.00       3
         12 02/18/86      40.00       3
         13 02/18/86      40.00       2
         14 02/18/86      40.00       2
```

input teilt RPT mit, wie die Eingabedatei aussieht und detail gibt an, wie die Zeilen auszugeben sind. Das am Ende stehende end zeigt das Ende des Report-Skripts an.

Die Sektion input ordnet jedem Feld in der Eingabedatei einen Namen zu, der innerhalb des gesamten Report-Skripts zur Bezugnahme auf das jeweilige Feld verwendet wird. Hinter dem Namen steht jeweils eine Typenspezifikation. Die Feldnamen muessen dabei in der gleichen Reihenfolge aufgelistet werden, in der die tatsaechlichen Felder in der Eingabedatei erscheinen. Die Typen werden in eckige Klammern eingeschlossen und bestehen aus der Typ- und der Laengenangabe.

Die Reportfeld-Typen sind dieselben wie die einfachen Datenbank-Feldtypen und auch die Laenge hat dieselbe Bedeutung. Die Typen sind numeric, float, string, amount, date und time (Reportfelder vom Typ COMB sind nicht zulaessig). Bei date- oder time-Feldern muss die Laenge nicht angegeben werden, waehrend sie bei den anderen Feldern erforderlich ist. Bedeutung der Laenge fuer die jeweiligen Feldtypen siehe Abschnitt 3.1.2.

Man kann Felder unter input auch noch anders spezifizieren, indem man naemlich die Datenbank-Feldnamen verwendet und keine Typenspezifikation angibt. Dann sucht RPT Typ und Laenge im Datenwoerterbuch. Es folgt das unter Verwendung der Datenbank-Feldnamen geschriebene Report-Skript:

```
input
  art.seriennummer, art.erwerbsdatum, art.grosshand_preis,
                                                art.bestellnummer
detail
```

```
  print seriennummer, erwerbsdatum, grosshand_preis,
                                                bestellnummer
end
```

Das Ergebnis dieses Reports sieht natuerlich genau so aus, wie das vorangegangene.

Es ist zu beachten, dass unter input der Name des Datensatztyps, in dem der Feldname auftritt, vor diesem Feldnamen aufgefuehrt werden muss, der Feldname jedoch in anderen Sektionen abgekuerzt werden kann.

Jetzt sollen zum Report Spaltenueberschriften, Formatierungsanweisungen und die Meldung 'Report fertig' hinzugefuegt werden. Dazu sind zwei neue Sektionen im Report erforderlich - 'before report' und 'after report'. Das folgende Report-Skript erzeugt den danach gezeigten Report.

```
input
  art.seriennummer, art.erwerbsdatum, art.grosshand_preis,
                                                art.bestellnummer
before report
  print 'Lager-Report' in column 23
  skip
  print 'SerNummer' in column 1,
        'Datum' in column 12,
        'BestNummer' in column 32,
        'Preis' in column 44
  print 53[-] in column 1
detail
  print seriennummer in column 1,
        erwerbsdatum in column 12,
        bestellnummer in column 32,
        grosshand_preis in column 44
after report
  skip
  print 'Report fertig' in column 23
end
```

Ergebnis:

Lager-Report

SerNummer	Datum	BestNummer	Preis
6	02/15/86	1	11.88
5	02/15/86	1	9.75
4	02/15/86	1	9.75
3	02/15/86	1	16.55
2	02/15/86	1	16.55
1	02/15/86	1	16.55
7	02/15/86	1	11.88
8	02/16/86	2	4300.00
9	02/16/86	1	16.20
10	02/16/86	2	16.20
11	02/18/86	3	89000.00

```
        12 02/18/86          3       40.00
        13 02/18/86          2       40.00
        14 02/18/86          2       40.00
```

In der Sektion 'before report' sind Erweiterungen zum Kommando print - die Moeglichkeit, Zeichenketten auszugeben, die Ausgabe explizit in Spalten zu positionieren und die Moeglichkeit, Zeichenketten aus sich wiederholenden Zeichen (in diesem Fall 53 Bindestriche) auszugeben. Das Kommando skip sorgt fuer Leerzeilen, ohne Parameter (Standard) eine Leerzeile. Die Kommandos in dieser Sektion werden vor der Verarbeitung der einzelnen Zeilen ausgefuehrt.

Durch das in der Sektion detail aufgefuehrte Kommando print werden Daten mit der Formatspezifikation using, die der Aufbereitung des Ausgabeformats, vor allem bei numerischen Daten, dient. ausgegeben werden. In Abschnitt 7.8.4 werden alle Eigenschaften der Option using erlaeutert. Die Felder werden unter den in der Sektion 'before report' ausgegebenen Spaltenueberschriften positioniert.

Die in der Sektion 'after report' aufgefuehrten Kommandos werden ausgefuehrt, nachdem alle Zeilen der Eingabedatei verarbeitet sind. Im angefuehrten Beispiel wird einfach eine Zeile uebersprungen und die Meldung 'Report fertig' ausgegeben.

Aus drucktechnischen Gruenden koennen, und das gilt auch im folgenden, die Ausdrucke und Bildschirmdarstellungen nicht immer maszstabsgerecht dargestellt werden.

Zur Komplettierung dieses Reports soll nun die Eingabedatei nach Erwerbsdatum und Bestellnummer sortiert werden. Die Erwerbsdaten werden in aufsteigender (ascending) und die Bestellnummern in fallender (descending) Reihenfolge aufgefuehrt. Fuer die Ausgabe der Seitenueberschriften, Spaltenueberschriften, Summen und anderer statistischer Informationen werden Gruppenwechsel herangezogen. Es folgt das Report-Skript, dass den nachfolgenden Report generiert.

```
input
  art.seriennummer,
  art.erwerbsdatum,
  art.grosshand_preis,
  art.bestellnummer
width 80
length 23
sort erwerbsdatum, bestellnummer desc
header
  print 'L A G E R B E S T A N D' centered
  print 'fuer Grosshandelslager' centered
  skip
footer
  skip
  print '--' in column 31, pageno using '%ld', '--'
```

```
before report
  print 'L A G E R B E S T A N D' centered
  print 'fuer Grosshandelslager' centered
  skip
before erwerbsdatum
  need 5
  print erwerbsdatum in column 1
  skip
  print 'SerNummer' in column 1,
        'Datum' in column 12,
        'BestNummer' in column 32,
        'Preis' in column 44
  print 53[-] in column 1
detail
  print seriennummer in column 1,
        erwerbsdatum in column 12,
        bestellnummer in column 32
        grosshand_preis in column 44 using '#####&.&&'
after erwerbsdatum
  need 2
  skip
  print 'Total' in column 32,
    total (grosshand_preis) in column 44 using '#####&.&&'
  skip
after report
  need 7
  skip
  print 'Gesamt Total' in column 27,
    total (grosshand_preis) in column 44 using '#####&.&&'
  skip
  print 'Durch-Preis' in column 1,
        'Minimal-Preis' in column 19,
        'Maximal-preis' in column 36,
        'Total Artikel' in column 54
  print avg (grosshand_preis) column 1 using '#####&.&&',
        min (grosshand_preis) column 18 using '#####&.&&',
        max (grosshand_preis) column 36 using '#####&.&&',
        count (*) column 54
  skip
  print 'Report fertig' centered
end
```

Ergebnis:

L A G E R B E S T A N D
fuer Grosshandelslager

02/15/86

SerNummer	Datum	BestNummer	Preis
6	02/15/86	1	11.88
5	02/15/86	1	9.75
4	02/15/86	1	9.75
3	02/15/86	1	16.55
2	02/15/86	1	16.55

1	02/15/86	1	16.55
7	02/15/86	1	11.88
		Total	92.91

-- 1 --

L A G E R B E S T A N D
fuer Grosshandelslager

02/16/86

SerNummer	Datum	BestNummer	Preis
8	02/16/86	2	4300.00
10	02/16/86	2	16.20
9	02/16/86	1	16.20
		Total	4332.40

02/18/86

SerNummer	Datum	BestNummer	Preis
11	02/18/86	3	89000.00
		Total	89000.00

-- 2 --

L A G E R B E S T A N D
fuer Grosshandelslager

07/23/86

SerNummer	Datum	BestNummer	Preis
12	02/18/86	3	40.00
13	02/18/86	2	40.00
14	02/18/86	2	40.00
		Total	120.00

-- 3 --

L A G E R B E S T A N D
fuer Grosshandelslager

Gesamt Total 93545.31

```
Durch-Preis   Minimal-preis  Maximal-Preis    Total Artikel
  6681.81          9.75       89000.00              14

                        Report fertig
```

Das Kommando sort legt fest, nach welchen Feldern und in welcher Reihenfolge zu sortieren ist. Die Sortierung erfolgt noch bevor der Report verarbeitet wird. Die Standard-Sortierordnung ist aufsteigend (vom niedrigsten zum hoechsten Wert). Durch das Schluesselwort desc kann diese jedoch auf fallend (vom hoechsten zum niedrigsten Wert) geaendert werden.

Die Sektion header wird oben auf jeder Seite des Reports abgearbeitet und footer wird jeweils unten abgearbeitet. Unter header ist die Ausgabeoption centered angegeben. Durch diese Option wird die gesamte ausgegebene Zeichenkette in die Zeilenmitte gerueckt. Im footer wird die interne Variable pageno verwendet, die die Seitennummer der aktuellen Seite enthaelt. Sie wird unter Verwendung des Formats ausgegeben, das Variablen vom Typ LONG entspricht.

'before report' wurde genauso verwendet wie in der vorhergehenden Skript-Version, aber es wurde eine neue Sektion, 'before erwerbsdatum', aufgenommen. Die in dieser Sektion aufgefuehrten Kommandos werden jeweils ausgefuehrt, bevor die neuen Gruppen, die aus Artikeln mit demselben Erwerbsdatum bestehen, verarbeitet werden. Das Kommando 'need anzahl' sorgt dafuer, dass auf der aktuellen Seite mindestens anzahl Zeilen frei bleiben. Ist das nicht der Fall, wird eine neue Seite angefangen. Im angefuehrten Beispiel sollen mindestens 5 Zeilen frei bleiben, was genuegt, um die Spaltenueberschriften und eine einzelne Zeile auszugeben.

Die Sektion detail ist gleich der oben angefuehrten. Zur Ausgabe von Teilsummen fuer alle Erwerbsdaten wird hier die Sektion 'after erwerbsdatum' benutzt. Diese Sektion wird abgearbeitet, nachdem alle Gruppen, die aus Artikeln mit demselben Erwerbsdatum bestehen, verarbeitet sind. Die numerische Funktion total berechnet die Summe ihres Ausdrucks, indem fuer jede Zeile der vorhergehenden Gruppe das entsprechende Feld bzw. der entsprechende Ausdruck addiert wird. Deshalb koennen numerische Funktionen auch nur in Gruppenkommandos after verwendet werden.

Das letzte Gruppenkommando, 'after report', wurde im Vergleich zur vorhergehenden Version erweitert. Die am Ende des Reports augefuehrte Funktion total berechnet die Summe der Grosshandelspreise aus allen Zeilen der Eingabedatei, wodurch die Gesamtsumme gebildet wird. Auch alle anderen numerischen Funktionen werden am Ende verwendet. Damit soll ihre Verwendung demonstriert werden. Es handelt sich dabei um avg (Durchschnitt), min (Minimum), max (Maximum) und

count (Anzahl der Zeilen).

7.2.2 Beispiel fuer einen Briefbogen

In diesem Abschnitt soll ein Briefbogen entwickelt und es soll gezeigt werden, wie man Variable und einen bedingten Druck verwendet, um auf der Grundlage der in der Eingabedatei enthaltenen Daten verschiedene Ausdrucke zu erzielen. Der Report ist ein vom Lager an alle Besteller geschickter Brief, dass ihre Waren im Lager vohanden sind. Die Eingabedatei fuer RPT muss mit einer SQL-Anfrage korrekt angelegt werden.

Im Report wird insbesondere die Gesamtsumme der bestellten Waren beachtet und eine entsprechende (bedingte) Bemerkung in den Brief eingefuegt.

Fuer die Erzeugung der Eingabedatei wird folgende SQL-Anfrage verwendet. Dadurch werden alle Artikel ausgewaehlt.

```
lines 0
select art_grosshand_preis, art_bestellnummer,
                                                best_best_nummer
from art, best
where art_bestellnummer = best_best_nummer /
```

Es folgt das Reportskript fuer den Ausdruck der Briefe:

```
input
  art.grosshand_preis,
  art.bestellnummer,
  best.best_nummer
/* Blatt-Spezifikation */
left margin 8
width 70
length 46
sort bestellnummer
before bestellnummer
  skip 6
  print '-MITTEILUNG-' in column 23
  skip 4
  print 'AN:', bestellnummer in column 10
  print 'VON: Grosshandelslager'
  print 'BETR.: Artikel-Bestellung'
  skip 2
  print 'Wir koennen Ihnen mitteilen, dass die von Ihnen'
  print'bestellten Artikel in unserem Lager vorraetig sind'
  print 'und fuer sie reserviert wurden. Ihre Lieferung'
  print 'umfasst im einzelnen:'
  skip
  print 'Nummer' in column 8, 'Preis' in column 25
  print 35[-] in column 8
detail
  print bestellnummer in column 8, grosshand_preis
```

```
                                                    in column 20
after bestellnummer
  /* Initialisieren Gesamtpreis */
  set art_total to total (grosshand_preis)
  skip 2
  print 'Wir danken Ihnen fuer die Bestellung.'
  skip
  print 'Der Auftrag belaeuft sich auf' art_total
                                    using '#####&.&&', 'M.'
  if art_total >= 1000. then
  begin
    print 'Wir freuen uns ueber die gute Zusammenarbeit.'
  end
  else
  begin
    print 'Wir wuerden uns ueber eine Vertiefung unserer'
    print 'Zusammenarbeit freuen.'
  end
  page

footer
  print 'Vielen Dank fuer Ihr Vertrauen und'
  print 'mit vorzueglicher Hochachtung'

end
```

Ergebnis:

-MITTEILUNG-

AN: 1
VON: Grosshandelslager
BETR.: Artikel-Bestellung

Wir koennen Ihnen mitteilen, dass die von Ihnen bestellten Artikel in unserem Lager vorraetig sind und fuer sie reserviert wurden. Ihre Lieferung umfasst im einzelnen:

Nummer	Preis
1	11.88
1	9.75
1	9.75
1	16.55
1	16.55
1	16.55
1	11.88
1	16.20

Wir danken Ihnen fuer die Bestellung.

Der Auftrag belaeuft sich auf 109.11 M.
Wir wuerden uns ueber eine Vertiefung unserer Zusammenarbeit freuen.

Vielen Dank fuer Ihr Vertrauen und
mit vorzueglicher Hochachtung

-MITTEILUNG-

AN: 2
VON: Grosshandelslager
BETR.: Artikel-Bestellung

Wir koennen Ihnen mitteilen, dass die von Ihnen bestellten Artikel in unserem Lager vorraetig sind und fuer sie reserviert wurden. Ihre Lieferung umfasst im einzelnen:

Nummer	Preis
2	4300.00
2	16.20
2	40.00
2	40.00

Wir danken Ihnen fuer die Bestellung.

Der Auftrag belaeuft sich auf 4396.20 M.
Wir freuen uns ueber die gute Zusammenarbeit.

Vielen Dank fuer Ihr Vertrauen und
mit vorzueglicher Hochachtung

-MITTEILUNG-

AN: 3
VON: Grosshandelslager
BETR.: Artikel-Bestellung

Wir koennen Ihnen mitteilen, dass die von Ihnen bestellten Artikel in unserem Lager vorraetig sind und fuer sie reserviert wurden. Ihre Lieferung umfasst im einzelnen:

Nummer	Preis
3	89000.00
3	40.00

Wir danken Ihnen fuer die Bestellung.

Der Auftrag belaeuft sich auf 89040.00 M.

Wir freuen uns ueber die gute Zusammenarbeit.

Vielen Dank fuer Ihr Vertrauen und
mit vorzueglicher Hochachtung

7.3 Entwicklung komplizierter Reporte

Im vorhergehenden Abschnitt wurde gezeigt, wie man einen aus mehreren Ebenen bestehenden Report schreiben kann, der Report-, Seiten- und Spaltenueberschriften, Seitennummern, Zwischensummen, Summen, Minimalwerte, Maximalwerte, Durchschnitte und Zaehlungen enthaelt. Ist man mit dem Schreiben von derartigen Reporten vertraut, kann man zu komplizierteren Verfahren uebergehen. Dabei werden komplizierte Ausdruecke, benannte Ausdruecke, Zeichenketten-Operatoren, bedingter Druck, Variablen und lokale Funktionen bearbeitet.

Zuerst sollen allgemeine Spezifikationen fuer einen komplizierten Report erlaeutert werden. Dann soll erlaeutert werden, wie die Strukturelemente des Reports implementiert werden koennen und wie man in diesem Zusammenhang eine SFORM-Bildmaske mit einer SQL-Anfrage und einem RPT-Skript kombinieren kann.

Im Report sollen Warenbestellungen entsprechend ihres Bestelldatums ausgegeben werden. Der Report soll in zwei Teile gegliedert werden. Zuerst werden die Bestellungen aufgelistet, die noch nicht befriedigt werden koennen. Danach werden die Bestellungen aufgelistet, die schon komplett sind. Innerhalb der Gruppen sollen die Zeilen nach Seriennummer, Bestellnummer und Kundennummer geordnet sein. Fuer jede Bestellung sollen die Summen gebildet werden, wobei beruecksichtigt wird, wieviel Zeit seit der Aufgabe der Bestellung vergangen ist, d.h. es wird unterschieden nach Bestellungen, die vor mehr als 90 Tagen aufgegeben wurden, nach Bestellungen, die vor 60-90 Tagen bzw. 30 - 60 Tagen bzw. vor weniger als 30 Tagen aufgegeben wurden. Liegt eine Warenbestellung mehr als 90 Tage zurueck, wird sie besonders gekennzeichnet.

Auf der Titelseite wurde der vom Nutzer fuer diesen Report festgelegte Zeitraum ausgedruckt, sowie Tag, Datum und Zeit des Reportbeginns.

```
.....................................................
B E S T E L L U N G S - I N V E N T U R   R E P O R T
_____________________________________________________

  Bestellungen ergangen vom 03/01/86 bis 04/01/86

               Freitag  5 April 1986

.....................................................
```

Zunaechst soll ermittelt werden, welche SQL-Anfrage erforderlich ist, um die Daten auszuwaehlen. Drei Tabellen aus der Datenbank muessen miteinander in Verbindung gebracht werden, und es muessen die Warenbestellungen aussortiert werden, die zwischen den zwei vom Nutzer angegebenen Terminen aufgegeben wurden, aber noch nicht befiedigt werden konnten. In der Anfrage muss also angegeben werden, wo die vom Nutzer gelieferten Daten einzusetzen sind und es muss fuer den Nutzer eine Moeglichkeit geben, diese Daten per Prompter einzugeben.

Das geschieht, indem man Parameter in die Anfrage einbaut und dann eine SFORM-Bildmaske mit dieser Anfrage koppelt (unter Verwendung von 'SQL Screen Registration', Abschnitt 6.3.6.1). Dann kann der Nutzer diese Bildmaske aus einem Menue auswaehlen und die gewunschten Daten eingeben. Diese werden in der Anfrage substituiert und dann wird die komplettierte Anfrage von SQL abgearbeitet.

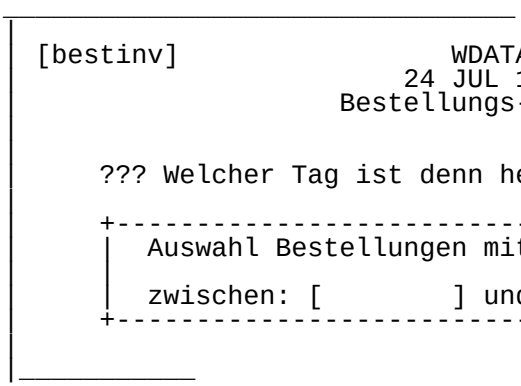
Ein Parameter ist ein Positionsparameter der Form $n, wobei n eine Zahl von 1 bis zur Anzahl der Parameter, die vom Nutzer geliefert werden sollen, ist. Da zwei Termine angegeben werden sollen, sind die zwei verwendeten Parameter $1 und $2. Diese werden in der Klausel where anstelle von "fest" eingetragenen Terminen verwendet. Dieser Teil der Klausel where sieht folgendermassen aus:

```
  where best_best_datum between $1 and $2
```

Die Anfrage selbst ist in eine Datei zu bringen und zu benennen, beispielsweise als besd.sql. Der Text der Anfrage lautet:

```
lines 0
select seriennummer, erwerbsdatum, grosshand_preis,
       bestellnummer, kunde.kundennummer, kunde.name,
       best_nummer, best_datum, best.kundennummer
from art, kunde, best
where best_datum between $2 and $3 and
      best_datum >= erwerbsdatum and
      bestellnummer = best_nummer and
      kunde.kundennummer = best.kundennummer/
```

Es folgt der Entwurf einer SFORM-Bildmaske mit den Promptern, hinter denen die Daten eingegeben werden koennen. Der in das erste Bildmasken-Feld eingegebene Wert wird anstelle von $1, der ins zweite Bildmaskenfeld eingegebene Wert anstelle von $2 eingesetzt usw.



In gleicher Weise koennen Parameter im Report-Skript (siehe Abschnitt 7.9.1) benutzt werden. Im angefuehrten Beispiel soll auf der Titelseite angegeben werden: der Zeitraum, in dem die Warenbestellungen aufgegeben wurden und Datum und Zeit des Reportbeginns. Dazu koenen in der Sektion 'before report' entweder Parameter oder die eingebauten Variablen today und hour benutzt werden. Wir benutzten im Beispielreport einen Parameter fuer das Tagesdatum, damit wir dieses willkuerlich festlegen koennen. Parameter koennen im Skript an beliebiger Stelle verwendet werden, vorausgesetzt, dass das Skript auch dann noch einen Sinn ergibt, wenn die vom Nutzer eingegebenen Werte substituiert wurden.

```
before report
          .
          .
          .

  print 'Report vom', today, 'um', hour centered
          .
          .
          .
```

Um die Warenbestellungen in bereits erfuellbare und noch nicht erfuellbare Bestellungen zu unterteilen, kann unter Verwendung des folgenden Ausdrucks eine Sortierung erfolgen:

```
  sort <dt> erwerbsdatum <= $1
```

Das Erwerbsdatum des Artikels muss also vor dem Tagesdatum liegen. Durch den oben beschriebenen Vergleich ergibt sich der Wert 0 (falsch), wenn die Bestellung noch nicht kom-

plett ist und der Wert 1 (wahr), wenn die Bestellung bereits komplett ist. Da 0 kleiner als 1 ist, werden alle noch nicht kompletten Bestellungen vor den kompletten Bestellungen in eine Gruppe zusammengefasst. Der gesamte Ausdruck erhaelt den Namen dt, so dass man auf den Gruppenwechsel Bezug nehmen kann, der spaeter im Report beim Uebergang von kompletten zu unvollstaendigen Bestellungen auftritt.

Fuer eine nutzerspezifische Verarbeitung koennen lokale Funktionen verwendet werden (siehe Abschnitt 7.4.6.2). Beispielsweise kann man damit fuer das Datum den Namen des Monats und des Wochentags ausdrucken. Obwohl man eigene lokale Funktionen verwenden kann (Abschnitt 7.4.6.2.4), liefert RPT drei derartige Funktionen, mit denen man dies tun kann. Dabei wird das heutige Datum genommen, die Monatsnummer (1 = Januar, 2 = Februar usw.) gewaehlt und dann wird diese Nummer in die entsprechende Zeichenkette uebersetzt. Gleiches kann man fuer den Wochentag tun. Die lokalen Funktionen mdy und dow werden benutzt, um die Nummer des Monats und des Wochentages zu waehlen, waehrend index verwendet wird, um die Nummern in aus Zeichen bestehende Zeichenketten zu uebersetzen. Mit der folgenden Kommandofolge kann man Variable auf den Wochentag und den Monat setzen.

```
set wocht to < tag > index(dow(vdate), 'Sonntag',
                 'Montag','Dienstag','Mittwoch',
                 'Donnerstag','Freitag','Samstag')
set monat to < mont > index(mdy(vdate,0)-1,'Januar',
                'Februar','Maerz','April','Mai',
                'Juni','Juli','August','September',
                'Oktober','November','Dezember')
print wocht, mont
```

Soll oben auf der Seite ein variabler Titel ausgedruckt werden, kann man den Wert einer Zeichenkettenvariablen, die sich im Gruppenkommando header befindet, aenedern, so dass die Variable jeweils verschiedenen Meldungen entspricht, je nachdem, wo man sich im Report befindet. Im hier besprochenen Report soll oben auf der Seite ausgegeben werden, ob auf der aktuellen Seite abgeschickte oder noch nicht abgeschickte Warenbestellungen aufgefuehrt werden. Das geschieht, indem man in das Gruppenkommando 'before report' eine Variable setzt, im Beispiel waere das pg_title. Damit wird angegeben, dass die noch nicht erfuellbaren Bestellungen ausgedruckt werden. (An dieser Stelle soll noch einmal daran erinnert werden, dass zuerst die noch nicht erfuellbaren Bestellungen aufgelistet werden. Bei Auftreten des Gruppenwechsels, dt, wird die Meldung geaendert, so dass nun die bereits abgeschickten Bestellungen ausgegeben werden und zur naechsten Seite uebergegangen wird. Hier die entsprechenden Absaetze im Report-Skript:

```
sort < tag > pb_lief_date ^= **/**/**,v_number,pb_nummer,
```

```
before report
    set pg_title to '*** Offene Bestellungen ***'
                  .
                  .
                  .
header
    print pg_title in column 1
                  .
                  .
                  .
before tag
    if pb_lief_date ^= **/**/** then
        begin
          set pg_title to '*** Komplette Bestellungen ***'
          page
          end
```

RPT liefert auch zusaetzliche Zeichenkettenoperationen. Manchmal ist es besser, wenn man Zeichenkettenvariable, Reportfelder oder Konstanten unmittelbar hintereinander druckt (d.h. ohne dass ein Leerzeichen dazwischen ist). Da RPT normalerweise als Standard ein Leerzeichen zwischen die Felder druckt, wird ein Verkettungsoperator ('+') bereitgestellt, mit dem man die Standardleerzeichen zwischen den Feldern unterdruecken kann. Es soll davon ausgegangen werden, dass Reportfelder fuer Stadt und Postleitzahl vorhanden sind, Dann wird ort mit der folgenden Anweisung print in Spalte 11 positioniert, wobei hinter ort zwei Leerzeichen und dann die Postleitzahl stehen wuerden.

```
print k_ort + '  ' + k_postleitzahl in column 11
```

Mit dem oben beschriebenen Verkettungsoperator ist es nicht moeglich, nachgestellte Leerzeichen aus einem Feld zu entfernen. Das ist aber manchmal wuenschenswert, wenn Adressen gedruckt werden und man die Luecken zwischen den Teilen der Adresse "schliessen" moechte. So wird z.B. in einer Adresse zuerst die Stadt, dann ein Komma und ein Leerzeichen, dann die Strasse und zwei Leerzeichen und dann die Postleitzahl gedruckt:

```
Rostock, Hafengasse  2551
```

Da die Stadt, ort, in der Datenbank gespeichert ist, handelt es sich um ein Feld fester Laenge, z.B. 25 Zeichen. Verwendete man fuer den Ausdruck einer Adresszeile den normalen Verkettungsoperator, wuerde fuer ort immer ein 25 Zeichen breites Feld freigelassen werden, wodurch zusaetzliche Leerstellen entstuenden, wenn ein Feld verwendet wird, dessen tatsaechlicher Inhalt weniger als 25 Zeichen lang ist. Dann fuehrte z.B. die Anweisung print:

```
print k_ort + ',' + k_strasse + ' ' + k_postleitzahl
```

zu dem Ergebnis:

```
Rostock , Hafengasse 2551
```

Das Problem kann mit dem Operator /+ geloest werden. Die Anweisung:

```
print k_ort / + ',' + k_strasse + ' ' + k_postleitzahl
```

fuehrt zu dem Ergebnis:

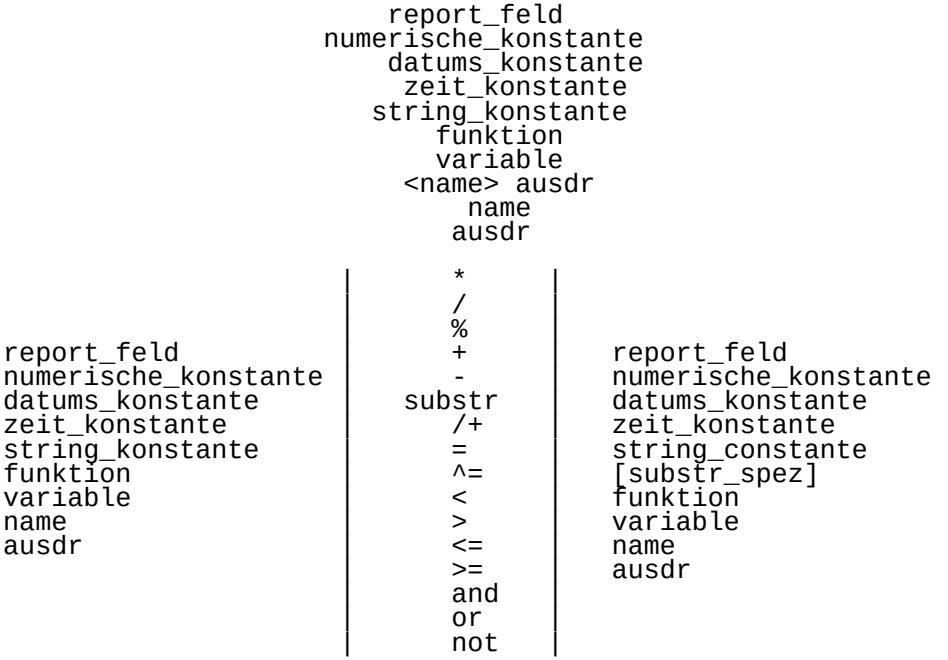
```
Rostock, Hafengasse 2551
```

Mit den oben stehenden Ausfuehrungen sollten einige neue interessante Moeglichkeiten vorgestellt werden, um einen Report weiter auszubauen, bzw. weiter aufzubereiten.

7.4 Ausdruecke

Der Begriff ausdr wird vom Reportgenerator verwendet, um Bezug auf eine Kombination aus Reportfeldern, Konstanten, Variablen, Funktionen und Operatoren zu nehmen. Reportfeld ist ein Name, der sich auf ein bestimmtes Datenfeld in der Eingabedatei bezieht. Dieser kann entweder datsat.feld sein, wenn datsat der Name eines Datensatztyps in der Datenbank und feld der Name eines Feldes innerhalb des Datensatztyps ist, oder aber ein beliebiger Name, dem eine Typenspezifikation folgt. In Abschnitt 7.4.3 wird mehr ueber Repordfelder gesagt.

Ausdruecke werden dazu verwendet, dem Reportprozessor mitzuteilen, was zu sortieren, was zu berechnen und was auszugeben ist. Im folgenden die Syntax, auf die sich ausdr bezieht:



Jeder Ausdruck oder Teilausdruck kann zur Festlegung der Prioritaet in runde Klammern eingeschlossen werden. Werden keine runden Klammern verwendet, werden die Operatoren in folgender Reihenfolge verarbeitet: hoechste Prioritaet [not], hohe Prioritaet [*/%], mittlere Prioritaet [+ - substr /+], geringe Prioritaet [= ^= < > <= >=], geringste Prioritaet [and or]. Operatoren mit gleicher Prioritaet werden in der Reihenfolge von links nach rechts verarbeitet.

Ein Ausdruck kann einen Namen erhalten und wird dann unter Verwendung dieses Namens in spaeter verwendeten Ausdruecken dargestellt (siehe Abschnitt 7.4.1). Treten benannte Ausdruecke in einem Kommando sort auf, koennen sie fuer Gruppenwechsel-Verarbeitung verwendet werden (siehe Abschnitt 7.5).

Ausdruecke, in denen unter Verwendung der Vergleichsoperatoren [= ^= < > <= >=] ein Vergleich vorgenommen wird, liefern das logische Ergebnis WAHR, TRUE oder FALSCH, FALSE. Diese logischen Ausdruecke werden numerisch dargestellt. Die Operatoren and, or und not werden zur Bildung zusammengesetzter logischer Ausdruecke verwendet (siehe Abschnitt 7.4.2).

Zahlen, Datum, Zeiten und die numerischen Operatoren (Addi-

tion, Subtraktion, Multiplikation, Division und Modulo) werden in Abschnitt 7.4.4 eroertert. Zeichenkettenkonstanten von aus Zeichen bestehenden Zeichenketten, der Vergleich von Zeichenketten und die Zeichenkettenoperatoren [+ / + substr) werden in Abschnitt 7.4.5 erlaeutert.

Es gibt zwei verschiedene Arten von Funktionen: numerische Funktionen und lokale Funktionen. Numerische Funktionen fuehren Operationen wie total oder avg an einer Gruppe von Datenelement-Werten oder Ausdruecken aus. Die Ergebnisse von lokalen Funktionen haengen immer nur von den ihnen uebergebenen Argumenten ab (siehe Abschnitt 7.4.6.2).

Das Ergebnis eines Ausdrucks kann unter Verwendung des Kommandos set einer Variablen zugeordnet werden. Diese Variable kann dann in spaeter folgenden Kommandos, die den in ihr enthaltenen Wert benoetigen, verwendet werden (siehe Abschnitt 7.4.6.2).

7.4.1 Benannte Ausdruecke

Ausdruecke, die aus Reportfeldern, Konstanten, lokalen Functionen und Namen anderer Ausdruecke bestehen, koennen unter Verwendung der folgenden Syntax einen Namen erhalten:

```
[<name>] ausdr
```

Wenn die Syntax ausdr auftritt, kann man fast immer einem Ausdruck einen Namen geben. Die Namen der Ausdruecke werden von dem das RPT-Skript schreibenden Nutzer gewaehlt. Alle Namen muessen mit einem Buchstaben beginnen. Danach koennen Buchstaben, Ziffern und Unterstreichungszeichen folgen und einen Namen bilden, der aus bis zu 32 Zeichen besteht.

Im folgenden Beispiel wird gezeigt, dass, wenn ein Ausdruck erst einmal einen Namen hat, dieser Name in spaeter folgenden Ausdruecken so verwendet werden kann, als waere er ein anderes Feld:

```
sort art, <diff> grosshand_preis - ind_abgabepreis

detail
  print art, <kumm> diff + (diff * 0.06) in col 30
  print kumm
```

Ausdruecke erhalten aus zwei Gruenden Namen:

1. Benannte Ausdruecke, die in dem Kommando sort auftreten, koennen mit den Gruppenkommandos 'before name' und 'after name' verbunden werden (siehe Abschnitt 7.5, Gruppenverarbeitung)

2. Ausdruecke koennen Namen erhalten, die ueber ihre Bedeutung Auskunft geben, so dass spaeter folgende Kommandos

besser lesbar und praegnant sind.

Man kann zwar auch einem Ausdruck, der nur ein Reportfeld enthaelt, einen Namen geben, aber das ist eigentlich nicht erforderlich. Ein Reportfeld ist bereits ein Ausdruck mit Namen. Der Name des Feldes ist standardmaessig der Name des Ausdrucks. Deshalb kann man sortierte Reportfelder in die Gruppenkommandos 'before name' und 'after name' einschliessen.

7.4.2 Logische Ausdruecke

Ein logischer Ausdruck ist eine Kombination aus Vergleichsoperatoren, logischen Operatoren und Werten, die RPT als WAHR, TRUE, oder FALSCH, FALSE interpretiert. Die Werte selbst koennen beliebige logische, arithmetische oder Zeichenketten-Ausdruecke sein. RPT ordnet den WAHREN logischen Ausdruecken den Wert 1 zu und den FALSCHEN logischen Ausdruecken den Wert 0.

Zwischen den logischen und den arithmetischen Ausdruecken besteht nur ein kleiner Unterschied. Der Typ eines Ausdrucks wird durch die Operation bestimmt, die bei Berechnung des Ausdrucks als letzte ausgefuehrt wird. Um einen logischen Ausdruck handelt es sich dann, wenn die letzte Operation entweder eine logische Operation (and, or oder not) oder eine Vergleichsoperation (=, ^=, <, >, <=, >=) ist. Um einen arithmetischen Ausdruck handelt es sich, wenn die letzte Operation eine numerische Operation (+, -, *, /, %) ist.

Ein einfacher logischer Ausdruck besteht aus zwei durch einen Vergleichsoperator verbundenen Werten. Die Vergleichsoperatoren werden genauer in Abschnitt 7.4.2.1 beschrieben. Hier einige Beispiele fuer einfache logische Ausdruecke:

```
strasse = 'Hafengasse*'
1 >= 0
a = b
x+2 < y-3
```

Zusammengesetzte logische Ausdruecke werden gebildet, indem einfache logische Ausdruecke durch die logischen Operatoren miteinander verbunden werden. Einzelheiten ueber logische Operatoren siehe Abschnitt 7.4.2.2. Hier einige Beispiele fuer zusammengesetzte logische Ausdruecke:

```
strasse = 'Hafengasse*' or strasse = 'Am Pier*'
a = b and x+2 < y-3
```

Am haeufigsten werden logische Ausdruecke in den Kommandos if verwendet. Wenn das Ergebnis eines Ausdrucks WAHR, TRUE, ist, wird die nach dem then stehende Gruppe von Kommandos ausgefuehrt. Wenn das Ergebnis FALSCH, FALSE, ist, wird die

hinter dem else stehende Gruppe von Kommandos ausgefuehrt. Es sollen die folgenden if-Kommandos betrachtet werden:

```
if 1 = 1 then            if 1 = 0 then
  print 'TRUE'             print'TRUE'
else                     else
  print 'FALSE'            print 'FALSE'
```

Das links stehende Kommando gibt "TRUE" aus, waehrend das rechts stehende Kommando "FALSE" ausgibt.

Logische Ausdruecke koennen auch als Teil arithmetischer Ausdruecke verwendet werden. RPT setzt bei Berechnungen die numerischen Werte des logischen Ausdrucks (1 oder 0) in den arithmetischen Ausdruck ein. Es ist jedoch zu beachten, dass ein arithmetischer Ausdruck in einem if-Kommando nicht anstelle eines logischen Ausdrucks verwendet werden kann.

7.4.2.1 Vergleiche

Unter Verwendung der Vergleichsoperatoren keonnen zwei Werte miteinander verglichen werden. Dabei koennen beide Werte ein Reportfeld, eine Funktion, eine Variable, ein beliebiger anderer Ausdruck oder eine Konstante sein. Hier die Vergleichsoperatoren:

```
=   (ist gleich)
^=  (ungleich)
<   (kleiner als)
>   (groesser als)
<=  (kleiner oder gleich)
>=  (groesser oder gleich)
```

Ausdruecke, in denen zwei Werte miteinander verglichen werden, fuehren entweder zu dem numerischen Ergebnis 0, wenn der Ausdruck falsch, FALSE, ist oder 1, wenn der Ausdruck wahr, TRUE, ist (siehe Abschnitt 7.4.2).

In der folgenden Tabelle wird gezeigt, wie man Vergleiche innerhalb von Ausdruecken vornehmen kann. Der Wert des jeweiligen Ausdrucks wird angegeben und es erfolgt eine logische Interpretation (TRUE oder FALSE). felda und feldb sollen Reportfelder mit den numerischen Werten 21 bzw 0 (Null) sein.

Ausdruck	RESULTAT
felda > feldb	TRUE
felda <= feldb	FALSE
felda < 34	TRUE
- 98.3428 >= felda	FALSE
felda + feldb ^= felda - feldb	FALSE
felda - 21 >= feldb	TRUE
felda - (21 >= feldb)	N/A

Der letzte Ausdruck, der der einzige arithmetische Ausdruck in der Tabelle ist, zeigt den Unterschied zwischen einem logischen und einem arithmetischen Ausdruck. Er ist gleich dem darueber stehenden Ausdruck, aber die runden Klammern erzwingen, dass der numerische Operator - zuletzt ausgefuehrt wird. Deshalb wird er von RPT als ein arithmetischer Ausdruck interpretiert.

7.4.2.2 Logische Operatoren

Es gibt drei logische Operatoren: and, or und not. Die Operatoren and und or werden zur Bildung zusammengesetzter logischer Ausdruecke verwendet, waehrend der Operator not den Wert des Ausdrucks, auf den er angewandt wird, umkehrt (oder negiert). Alle drei Operatoren koennen nur bei logischen Ausdruecken verwendet werden. Das heisst, Ausdruecke wie 1 and 1 und not 0 sind unzulaessig.

RPT berechnet zusammengesetzte logische Ausdruecke, indem zuerst der Wert der einzelnen Teilausdruecke bestimmt wird und dann die Ergebnisse wie in der folgenden Tabelle kombiniert werden.

Ausdr.	Resultat	Ausdr.	Resultat
TRUE and TRUE	TRUE	TRUE or TRUE	TRUE
TRUE and FALSE	FALSE	TRUE or FALSE	TRUE
FALSE and TRUE	FALSE	FALSE or TRUE	TRUE
FALSE and FALSE	FALSE	FALSE or FALSE	FALSE

Ausdr.	Resultat
not TRUE	FALSE
not FALSE	TRUE

Hier einige Beispiele fuer logische Ausdruecke und ihre Werte. Wie oben soll auch hier das Reportfeld felda 21 und das Reportfeld feldb 0 enthalten.

Ausdruck	Resultat
felda <= 21 and felda > 0	TRUE
not (feldb = 0)	FALSE
felda > 21 or feldb < 0	FALSE
felda > 21 or not feldb < 0	TRUE
not (feldb = 0) or (felda <= 21 and felda > 0)	TRUE

7.4.3 Felder

Ein Reportfeld oder einfach Feld, ist ein von RPT verwendeter Ausdruck. Es handelt sich hierbei um einen Namen, der sich auf ein bestimmtes Datenfeld in der Eingabedatei bezieht. Dieser kann entweder die Form datsat.feld haben, wobei datsat der Name des Datensatztyps in der Datenbank

und feld der name eines Feldes innerhalb dieses Datensatztyps ist, oder er kann ein beliebiger Name, name, sein, dem eine Typenspezifikation folgt. Ein Feld kann fast ueberall dort verwendet werden, wo der Begriff ausdr in der RPT Syntax-Dokumentation erscheint. Allerdings koennen Reportfelder, oder Ausdruecke, die diese enthalten, nicht in Kommandos verwendet werden, die in dem Gruppenkommando 'after report' auftreten, ausser wenn sie als ein Argument oder als Teil eines Argumentausdrucks fuer eine numerische Funktion erscheinen. (siehe Abschnitt 7.4.6.1). Der Grund dafuer ist, dass am Ende eines Reports bereits alle Eingabezeilen verarbeitet worden sind und die Felder undefinierte Werte haben.

Um zu sichern, dass Feldnamen nicht zu Mehrdeutigkeiten in einem RPT-Skript fuehren, sollten sie mit einem Buchstaben beginnen und nur aus Buchstaben, Ziffern und Unterstreichungszeichen bestehen. Das ist jedoch nur eine Empfehlung, die nicht notwendigerweise befolgt werden muss.

Es ist nicht erforderlich, einem Ausdruck, der nur aus einem Reportfeld besteht, einen Namen zu geben, da standardmaessig der Feldname der Name des Ausdrucks ist. Man kann jedoch einem solchen Ausdruck auch einen Namen geben, wodurch die Lesbarkeit der Kommandos oft verbessert wird.

Zu jedem Feld gehoert ein Typ, durch den die Art der Daten beschrieben wird, die in dem Feld enthalten sind. Die gueltigen Reportfeld-Typen sind dieselben wie die Datenbank-Feldtypen, naemlich numeric, float, date, time, amount und string. Reportfelder vom Typ COMB sind nicht zulaessig. Im allgemeinen werden Felder in zwei Kategorien unterteilt: Zahlen und Zeichenketten. Da die Felder numeric, float, date, time und amount numerische Werte enthalten, koennen sie unter Verwendung der numerischen Operatoren manipuliert werden (siehe die folgenden Abschnitte). Felder vom Typ string koennen unter Verwendung der Verknuepfungsoperatoren (+ / +) und des Teilzeichenketten-Operators (substr) manipuliert werden (siehe Abschnitt 7.4.5.3).

7.4.4 Zahlen, Datum und Zeiten

In der Tabelle in 7.4, in der die Syntax eines Ausdrucks definiert wurde, werden alle vorhandenen Typen von Konstanten aufgelistet. Explizit definierte Zahlen, Daten und Zeiten, wie z.B. -45.30, 12/28/86, 12:00 sind Konstanten. Ihre Syntax bleibt innerhalb von WEGA-DATA immer gleich. Der Vollstaendigkeit halber, erfolgt jedoch in 7.4.4.1 bis 7.4.4.3 eine Wiederholung.

In Abschnitt 7.4.4.4 werden Daten und Zeiten miteinander verglichen und in Abschnitt 7.4.4.5 wird die Verwendung der Operatoren /, *, %, + und - in numerischen Ausdruecken behandelt.

7.4.4.1 Numerische Konstanten

Numerische Konstanten werden in zwei Kategorien unterteilt - in Zahlen mit und Zahlen ohne Dezimalpunkt. Hier ein Ueberblick ueber die Syntax der zwei Klassen numerischer Konstanten.

Zahlen ohne Dezimalpunkt:
Optionelles Minuszeichen kann vorangestellt werden.
Bis zu 9 Wertziffern.

Zahlen mit Dezimalpunkt:
Optionelles Minuszeichen kann vorangestellt werden.
Bis zu 16 Wertziffern.
Rechts vom Dezimalpunkt koennen bis zu neun Ziffern stehen.

Eine Zahl ohne Dezimalpunkt (ganze Zahlen) kann aus bis zu neun Ziffern bestehen. Handelt es sich um eine negative Zahl, wird ein Minuszeichen (-) vorangestellt. Zwischen der Zahl und dem Minuszeichen darf kein Leerzeichen stehen. Im folgenden Kommando print wird ein Ausdruck gezeigt, der aus zwei mumerischen Konstanten und einem Feld besteht:

```
print -9 * felda + 123555789 in column 52
```

Wenn eine Zahl einen Dezimalpunkt enthalten muss, z.B. ein in Mark ausgedrueckter Wert oder ein Dezimalwert, kann diese Zahl bis zu sechzehn Ziffern enthalten. Rechts vom Dezimalpunkt koennen bis zu neun Ziffern stehen. Handelt es sich um eine negative Zahl wird ein Minuszeichen vorangestellt. Hier einige negative und positive mumerische Konstanten mit Dezimalpunkt:

```
1234567.123456789     987654321.50     -49.99
    -53.899898334        -50000.0        0.00
```

An die numerischen Konstanten kann entweder der Buchstabe n, a oder f angehaengt werden, so dass RPT diese als zum Typ numeric, amount oder float gehoerig interpretiert. Bestimmung des Typs von Variablen siehe Abschnitt 7.4.7.1.

7.4.4.2 Daten-Konstanten

Ein Datum muss aus drei durch Schraegstriche voneinander abgegrenzten Werten bestehen. Es erscheint in der mm/dd/yy, wobei mm den Monat, dd den Tag und yy das Jahr angibt. Der Jahreswert bezieht sich auf die Anzahl der Jahre, die seit 1900 vergangen sind. Hier einige gueltigen Daten:

```
1/1/1   1/1/0   12/31/99   6/3/86
```

Folgende Daten sind nicht gueltig:

```
0/0/0   13/32/178   3/14
```

Ein Datum kann keine negative Zahl sein. Man kann zu einem Datum eine Anzahl von Tagen addieren oder subtrahieren. Das soll weiter unten in Abschnitt 7.4.4.5 gezeigt werden.

RPT liefert eine eingebaute Variable, today, die aus dem aktuellen Monat, Tag und Jahr besteht und in der Form mm/dd/yy auftritt. Hier ein Beispiel einer solchen in einer print-Anweisung verwendeten Variablen:

```
print today centered
```

7.4.4.3 Zeit-Konstanten

Eine Zeit-Konstante muss in der Form hh:mm geschrieben sein, wobei hh die Stunde bezeichnet und mm die Minuten angibt, die seit dieser vollen Stunde vergangen sind. Eine Zeit-Konstante basiert auf einem 24-Stunden-Tag. Der Stundenwert liegt im Bereich 0 - 23 und der Minutenwert liegt zwischen 0 und 59. Hier einige gueltige Zeiten:

```
0:0   23:59   8:30   12:00
```

Die folgenden Zeiten sind nicht gueltig:

```
24:00   23:60   0:99   29:30
```

Eine Zeit bezieht sich auf eine Tageszeit und nicht auf eine Anzahl von Stunden und Minuten. Zeiten koennen nicht negativ sein und sie koennen nicht addiert werden. Die Differenz jedoch (gemessen in Minuten), die zwischen zwei Zeiten besteht, kann durch Subtraktion beider Zeitwerte ermittelt werden.

RPT liefert auch eine eingebaute Variable, hour, die die aktuelle Stunde und Minuten enthaelt. Die Zeit wird ausgegeben als hh:mm. Hier ein Beispiel fuer die Verwendung von hour:

```
print "Gedruckt um:" col 10, hour
```

7.4.4.4 Vergleich von Daten und Zeiten

Zwei Daten oder zwei Zeiten koennen unter Verwendung eines Vergleichsoperators miteinander verglichen werden. Man darf jedoch nur ein Datum mit einem anderen Datum und nur eine Zeit mit einer anderen Zeit vergleichen. Andere Vergleiche sind nicht zulaessig.

Eine Anzahl von Tagen kann zu einem Datum addiert oder von einem Datum subtrahiert werden (siehe Abschnitt 7.4.4.5). Das Ergebnis ist immer noch ein Datum (vorausgesetzt dass es innerhalb des akzeptierten Bereiches liegt), und es kann

mit einem anderen Datum verglichen werden.

7.4.4.5 Numerische Operationen

Die folgenden arithmetischen Operatoren koennen in Ausdruecken verwendet werden:

* - Multiplikation
/ - Division
% - Modulo
+ - Addition
- - Subtraktion

Wenn zwei ganze Zahlen (vom Typ numeric) dividiert werden, ist das Ergebnis ebenfalls eine ganze Zahl. Die Dezimalstellen werden abgeschnitten. Der Operator % gibt den Rest an, der entsteht, wenn der links stehende Wert durch den rechts stehenden Wert dividiert wird.

Zur Festlegung von Prioritaeten innerhalb der arithmetischen Ausdruecke werden runde Klammern verwendet. Bei Verwendung runder Klammern werden Multiplikations-, Divisions- und Modulo-Operationen vor Addition und Subtraktion ausgefuehrt. Operatoren auf gleicher Ebene werden von links nach rechts berechnet.

Da Daten intern im Julianischen Kalenderformat gespeichert werden, kann eine Zahl (von Tagen) vom Datum subtrahiert oder zu diesem addiert werden. Das Ergebnis ist ein Datum, das viele Tage vor oder nach dem urspruenglichen Datum liegt. Die Subtraktion zweier Daten ergibt die Anzahl der dazwischenliegenden Tage. Eine Zeit gibt die Tageszeit an, daher ist die einzige bedeutungsvolle arithmetische Operation die Subtraktion von Zeiten. Die Subtraktion zweier Zeiten ergibt die Anzahl der dazwischenliegenden Minuten.

Die folgenden als Beispiel verwendeten Ausdruecke sind sehr praktisch in ihrer Verwendung, da nur Konstanten verwendet werden. Sie zeigen jedoch, wie einige arithmetische Ausdruecke verwendet werden.

Ausdruck	Resultat
4 % 2	0
5 % 2	1
3 + 2 * 4	11
3 + 2 * 4 - 6 / 2	8
103 % 10 + 103 / 10	13
6 / 3 * 2	4
6 / (3 % 2)	6
(3 + 2) * (4 - 6)	-10
3 * (6 / (2 + 1)	6
9 - 6 + 1.5	4.50
9 + -6 + 1.5 + -3.5	1.00
3.2504 + 1.0096 - 2	2.260000

```
11/1/86 + 30                        12/1/86
3/12/86 - 3/2/86                    10
(3/12/86 - 3/2/86) * -18.50         -185.00
```

7.4.5 Zeichenketten

Reportfelder vom Typ string, aus Zeichen bestehende Zeichenketten-Konstanten und beliebige Zeichenketten, die unter Verwendung derselben gebildet werden, und die Zeichenketten-Operatoren werden als Strings bezeichnet. Die zwei Typen der aus Zeichen bestehende Zeichenketten-Konstanten werden in Abschnitt 7.4.5.1 beschrieben. In Abschnitt 7.4.5.2 wird der Vergleich von Zeichenketten und die beim Vergleich von Zeichenketten verwendeten Sonderzeichen beschrieben. Die Zeichenketten-Operatoren werden in Abschnitt 7.4.5.3 naeher erlaeutert.

7.4.5.1 Zeichenketten-Konstanten

Es gibt zwei Arten von aus Zeichen bestehenden Zeichenketten-Konstanten. Das sind einmal in einfache Anfuehrungsstriche eingeschlossene aus Zeichen bestehende Zeichenketten und zum anderen wiederholt auftretende Zeichen. Ihre Syntax ist folgendermassen:

```
'string'
anz[zeichen]
```

Das Wort string bezieht sich auf druckbare Zeichen. Ueberschriften, Unterstreichungen und andere unveraenderliche Informationen in einem Report koennen unter Verwendung von Zeichenketten-Konstanten ausgegeben werden. Eine Zeichenketten-Konstante, die keine Zeichen enthaelt (die Null-Zeichenkette, mit der Laenge Null) wird dargestellt, indem zwei einfache Anfuehrungsstriche nebeneinander geschrieben werden. Die einfachen Anfuehrungsstriche koennen in einer Zeichenketten-Konstanten verwendet werden, wenn davor das Zeichen (\) steht. Das bezieht sich auch auf das Zeichen (\) selbst. Hier einige der gueltigen Zeichenketten-Konstanten:

```
''              (die Null-Zeichenkette)
'xyz'
'   '           (aus Leerzeichen bestehende Zeichenkette)
'12!A.OK'
'\\'            (aus einem Backslashs bestehende Zeichenkette)
```

Da die in Anfuehrungszeichen stehende Zeichenkette eine Zeichenkette beschreibt (Blanks), muss sie auf einer einzigen Zeile stehen.

Beim Wiederholungszeichen (anz[zeichen]) gibt das Wort anz an, wie oft das zeichen auftreten soll. Null (0) ist fuer

anz nicht zulaessig. Mit der Zeichenketten-Konstante fuer wiederholt auftretende Zeichen wird die Darstellung von Zeichenketten erleichtert, die ansonsten nur sehr schwierig darzustellen waeren. Zum Beispiel:

```
  9[.]  entspricht  '.........'
 18[ ]  entspricht  '                  '
 20[-]  entspricht  '--------------------'
```

Zwischen den Klammern darf ein und nur ein druckbares Zeichen auftreten. Aufgrund dieser Regel kann ein beliebiges druckbares Zeichen verwendet werden. Daher sind 64[?], 128[*], 12[|], 7['] und 3[\] gueltige Zeichenketten-Konstanten fuer wiederholt auftretende Zeichen.

7.4.5.2 Vergleich von Zeichenketten

Wenn bestimmte Report-Informationen unter bestimmten Bedingungen ausgegeben werden sollen, kann das Kommando if verwendet werden (siehe Abschnitt 7.8.6). Obwohl der im Kommando if nach dem Schluesselwort if folgende Ausdruck kein Zeichenketten-Ergebnis sein kann, kann er aus einem Vergleich von Zeichenketten bestehen.

Soll beispielsweise bei jedem Auftreten des Ortes Rostock eine spezielle Meldung ausgegeben werden, koennen folgende Kommandos verwendet werden:

```
if ort = 'Rostock*' then
  begin
    skip 1
    print 'Sondermitteilung'
  end
```

Zwei Zeichenketten koennen unter Verwendung einer der Vergleichs-Operatoren [= ^= < > < = > =] verglichen werden. Zeichenketten sind nur dann gleich, wenn ihr Inhalt gleich ist und sie die gleiche Laenge haben (Anzahl der Zeichen).

Hier eine kurze Uebersicht ueber die Ordnung der meisten Zeichenketten-Zeichen (der erste Satz wird folgendermassen gelesen: das Leerzeichen ist kleiner als das Zeichen Null).

```
' ' < '0'              '0' < '9'
'9' < 'A'              'A' < 'Z'
'Z' < 'a'              'a' < 'z'
```

Wie bei allen Vergleichen ist auch beim Vergleich von Zeichenketten das Ergebnis Eins oder Null. Das Ergebnis Eins gibt an, dass die durch den Vergleich getroffene Aussage WAHR ist und Null gibt an, dass diese Aussage FALSCH ist. Da das Ergebnis eines Vergleichs numerisch ist, ist das Ergebnis des folgenden Ausdrucks Zwei:

```
1 + ('Hallo' ^= 'Ade')
```

Beim Vergleich von Zeichenketten ist es oft erforderlich, eine Reihe von Zeichenketten zu ueberpruefen oder alle Zeichenketten mit bestimmten gemeinsamen Buchstaben zu ueberpruefen. Sonderzeichen koennen in einer Zeichenketten-Konstanten verwendet werden, um diese Art eines mehrfachen Vergleichs vorzunehmen. Diese Sonderzeichen und ihre Bedeutung sind:

? - Das variable Zeichen. Das Fragezeichen entspricht einem beliebigen einzelnen Zeichen. Wenn man also alle Personen namens Meier finden will und man weiss nicht, ob sie "Meier" oder "Meyer" geschrieben werden, kann man angeben Me?er oder in diesem Fall sicher M??er.

* - Die variable Zeichenkette. Der Stern entspricht einer beliebigen Zeichenkette beliebiger Laenge, einschliesslich Zeichenketten der Laenge Null (auch als Nullzeichenketten bezeichnet).

[...] - Das eingeschraenkt variable Zeichen. Die drei Punkte entsprechen einem Satz von Zeichen, die eine Zeichenklasse festlegen. Die Zeichenklasse entspricht einem beliebigen einzelnen Zeichen, das der Klasse angehoert. Zeichenbereiche koennen angegeben werden, indem zwei Zeichen durch einen Strich "-" getrennt werden. So koennten beispielsweise alle Grossbuchstaben durch die Klasse

[ABCDEFGHIJKLMNOPQRSTUVWXYZ]

angegeben werden oder aber einfacher als [A-Z]. Alle Buchstaben (Klein- und Grossbuchstaben zusammen) koennen als [a-zA-Z] dargestellt werden. Andere Klassen koennen in aehnlicher Weise aufgestellt werden.

Der Stern ist sehr nuetzlich, wenn zwei Zeichenketten verglichen werden sollen, insbesondere, wenn die Laenge der einen Zeichenketten nicht bekannt ist, oder wenn nur ein Teil ihres Wertes uebereinstimmen muss. Wir wollen ein Reportfeld vom Typ string und der Laenge 30 betrachten. Der Feldname ist kuname und das Feld enthaelt Namen der Kunden. Um alle Namen der Kunden auszugeben, deren Familienname auf "stein" endet, muss das if-Kommando folgendermassen beginnen:

```
if name = '*stein *' then ...
```

Ein vor einer Zeichenketten-Konstanten stehender Stern ueberdeckt alle vorangehenden Kombinationen. Das nach stein stehende Leerzeichen sichert, dass Namen wie (Wacker)steinchen oder ...steinmann keine Uebereinstimmung ergeben. Der Stern am Ende wird benutzt, damit alle rechts stehenden Leerzeichen keine Rolle spielen.

Soll der Reportgenerator die Sonderbedeutung der Zeichen ?, * und [] in einer in einfache Anfuehrungszeichen eingeschlossenen Zeichenketten-Konstanten ignorieren, wird diesen das Rueckstrich-Zeichen (\) vorangestellt. Das trifft

auch auf die einfachen Anfuehrungszeichen und das Backslash-Zeichen selbst zu (siehe Abschnitt 7.4.5.1).

7.4.5.3 Zeichenketten-Operatoren

Aus Zeichen bestehende Zeichenketten koennen unter Verwendung des Operators + verknuepft werden. Der /+ Operator bewirkt, dass vor der Verkettung der Zeichenketten die nachgestellten Leerzeichen der linken Zeichenkette entfernt werden. Der Operator substr dient zur Schaffung einer neue Zeichenkette, die Teil einer gegebenen Zeichenkette ist. Hinter jedem Operator substr steht ein Teilzeichenketten-Spezifizierer (substr_spez, siehe Tabelle in Abschnitt 7.4), der sich auf die gegebene Zeichenkette bezieht. Dieser Spezifizierer substr_spez hat die Form start-end, wobei start die Position des Anfangszeichens (indiziert von 1) und end die Position des letzten Zeichens ist. Im folgenden soll eine ausfuehrlichere Erlaeuterung dieser Operationen erfolgen.

Zwei Zeichenketten werden durch Aneinanderreihen miteinander verkettet. Der Operator + bewirkt die Entstehung einer neuen Zeichenkette. Somit ist das Ergebnis des Ausdrucks

```
'ALPHA' + 'BET'
```

die neue Zeichenkette

```
'ALPHABET'
```

Anstelle von ALPHA und BET koennen im obenstehenden Ausdruck beliebige Zeichenketten-Ausdruecke verwendet werden. Gleiches trifft auch auf die anderen in diesem Abschnitt erlaeuterten Operationen zu. In der folgenden Auflistung wird angefuehrt, in welcher Form Zeichenketten-Ausdruecke auftreten koennen.

- Reportfeld vom Typ string,
- In einfache Anfuerungszeichen eingeschlossene Zeichenketten-Konstanten (siehe Abschnitt 7.4.5.1),
- Konstante mit mehrfach auftretenden Zeichen (Abschnitt 7.4.5.1),
- (Zeichenketten) Variable (Abschnitt 7.4.7),
- lokale Funktionen (von Zeichenketten) (Abschnitt 7.4.6.2),
- Ausdrucksname (von Zeichenketten) (Abschnitt 7.4.1),
- Ausdruecke, die aus den oben angefuehrten Zeichenketten- Ausdruecken und den Zeichenketten-Operatoren zusammengesetzt sind.

Der oben gezeigte Ausdruck ('ALPHA' + 'BET') wird abgeaendert, indem an beide Zeichenketten Leerzeichen angehaengt werden, wodurch Zeichenketten der Laenge 7 entstehen (wenn man sie als Datenbankfelder betrachtet).

```
'ALPHA ' + 'BET  '
```

Die von diesem Ausdruck beschriebene Verkettung ergibt die Zeichenkette:

```
'ALPHA BET  '
```

Da hierbei die nachgestellten Leerzeichen der linken Zeichenkette zu unerwuenschten, eingebetteten Leerzeichen werden, ist es in diesem Fall die Verwendung des "/+" Operators angebrachter.

Wird der "/+" Operator verwendet, werden vor der Verkettung der zwei Zeichenketten die nachgestellten Leerzeichen der ersten Zeichenkette entfernt und es ensteht die Zeichenkette:

```
'ALPHABET  '
```

Was waere das Ergebnis des folgenden Ausdrucks?

```
'ALPHA ' /+ "
```

Es soll daran erinnert werden, dass zwei unmittelbar nebeneinander stehende einfache Anfuehrungszeichen die Null-Zeichenkette (der Laenge 0) darstellen. Aus der links stehenden Zeichenkette werden die nachgestellten Leerstellen entfernt und dann werden die zwei Zeichenketten miteinander verkettet. Es entsteht die Zeichenkette:

```
'ALPHA'
```

Als Eselsbruecke fuer den /+ Operator kann man sich diesen als gestrichenen Verkettungs-Operator vorstellen. Die Leerzeichen der linken Zeichenkette werden "gestrichen", bevor die Zeichenketten verknuepft werden.

Eine Teil-Zeichenkette ist Teil einer Zeichenkette. Eine Teil-Zeichenkette kann aber auch gleich der gegebenen Zeichenkette sein. Hier einige Teil-Zeichenketten der Zeichenkette: 'AEIOU'.

```
'EI' 'IOU' 'A' " 'AEIOU'
```

Jede Zeichenkette hat mindestens zwei Teilzeichenketten - die Zeichenkette selbst und die Null-Zeichenkette. Eine Ausnahme bildet die Null-Zeichenkette. Diese besitzt nur eine Teil-Zeichenkette, naemlich die Null-Zeichenkette!

Hier die Syntax der Teil-Zeichenketten-Operation:

```
string substr [subst_spez]
```

Der Operator ist substr, substr_spez spezifiziert die gewuenschte Teil-Zeichenkette von string. Dieser Spezifizierer hat die Form start-end, wobei start die Position des Anfangszeichens innerhalb der Zeichenkette (indiziert von 1) ist und end die Position des Endzeichens.

Hier einige Beispiele fuer Operationen mit Teil-Zeichenketten:

Ausdruck	Resultat
'AEIOU' substr [1 - 3]	'AEI'
'AEIOU' substr [3 - 5]	'IOU'
'AEIOU' substr [3 - 9]	'IOU'
'AEIOU' substr [6 - 99]	'' (Null-String)
'AEIOU' substr [2 - 2]	'E'
'AEIOU' substr [1 - 6]	'AEIOU'
'AEIOU ' substr [1 - 6]	'AEIOU '
'AEIOU ' substr [6 - 107]	' '

7.4.6 Funktionen

Die Verwendung einer Funktion in einem Ausdruck erfolgt aehnlich wie die Verwendung eines Ausdrucksnamens. Das Ergebnis der Funktion ersetzt einfach den Funktionsaufruf im Ausdruck. RPT liefert zwei Grundarten von Funktionen: numerische Funktionen und lokale Funktionen. Mit numerischen Funktionen werden die Ergebnisse berechnet, indem der Wert eines Ausdrucks aus vielen Zeilen der Eingabedatei entnommen wird. Mit lokalen Funktionen werden die Ergebnisse dagegen berechnet, indem die Werte der an diese Funktionen uebergebenen Argumente verwendet werden.

Die Funktion total ist ein Beispiel fuer eine numerische Funktion. Mit ihr koennen Ausdrucksergebnisse von zwei oder mehr Gruppenwechseln in einem Report addiert werde. Alle numerischen Funktionen sind in Abschnitt 7.4.6.1 beschrieben.

Die lokalen Funktionen liefern ein Ergebnis, das auf den ihnen uebergebenen Argumenten beruht. Sie koennen nur Werte verwenden, die fuer das Gruppenkommando, in dem sie als Argumente auftreten, lokal sind. Sie koennen keine Werte verwenden, die waehrend verschiedener Gruppenverarbeitungen akkumuliert oder beobachtet werden. Lokale Funktionen werden in Abschnitt 7.4.6.2 eingefuehrt und werden in den folgenden Abschnitten einzeln erlaeutert.

7.4.6.1 Numerische Funktionen

Numerische Funktionen koennen nur in after-Gruppenkommandos verwendet werden, und zwar entweder allein oder als Teil eines Ausdrucks. Innerhalb eines Ausdrucks koennen verschiedene numerische Funktionen verwendet werden und ein und dieselbe numerische Funktion kann mehrmals auftreten. Es gibt folgende numerischen Funktionen: count, total, min max und avg.

Ein after-Gruppenkommando wird abgearbeitet, wenn der zu ihr gehoerende Gruppenwechsel auftritt (siehe Abschnitt

7.5, Gruppenverarbeitung). Der Wert einer numerischen Funktion wird unter Verwendung der Gruppe von Datenwerten berechnet, die dem die Funktion enthaltenden Gruppenkommando after entspricht.

Wird z.B. in dem Gruppenkommando 'after report' die Funktion count verwendet, erhaelt man als Ergebnis die Gesamtanzahl der verarbeiteten Datensaetze (Zeilen in der Eingabedatei). Wird diese Funktion dagegen in einem 'after name'-Gruppenkommando verwendet, erhaelt man die Anzahl der Datensaetze, die seit Abarbeitung des letzten 'after name'-Gruppenkommandos verarbeitet wurden.

Die allgemeine Syntax einer numerischen Funktion ist:

```
funktions_name (ausdr [where ausdr])
```

Das Schluesselwort where ist ein Operator. Es gibt an, dass der Wert des vor dem where stehenden Ausdrucks nur dann beruecksichtigt wird, wenn der darauf folgende Ausdruck WAHR ist (ungleich Null).

Beide Ausdruecke muessen zu numerischen Ergebnissen fuehren. Der gesamte in runden Klammern stehende Ausdruck (mit oder ohne den where-Operator) ist der Argumentausdruck der numerischen Funktion. Im Argumentausdruck sind lokale Funktionen, Ausdrucksnamen, Felder und Konstanten gueltig. Variable und numerische Funktionen sind nicht gueltig.

Mit der folgenden numerischen Funktion kann man den Duchschnitt aller "Betraege" berechnen, die groesser als 77 Mark sind:

```
avg (durch where durch > 77.00)
```

Da es keinen Sinn ergibt, wenn in einer Funktion count vor dem Operator where ein Ausdruck steht, wird an seiner Stelle ein Stern verwendet. Die Funktion count hat folgende Syntax:

```
count (* [where ausdr])
```

Bei jeder Verarbeitung eines Datensatzes wird der Argumentausdruck berechnet. Die Funktion total liefert die Summe der sich ergebenden Werte. Die Funktionen min und max liefern den kleinsten bzw. den groessten dieser Werte und die Funktion avg liefert deren Durchschnitt. Nachdem das after-Gruppenkommando, in dem eine numerische Funktion enthalten ist, ausgefuehrt ist, wird die Funktion erneut angewandt, so dass die naechste Gruppe von Werten verarbeitet werden kann.

Im naechsten Beispiel fuer ein after-Gruppenkommando sind ort und grosshand_preis Reportfelder oder Ausdrucksnamen. Es ist zu beachten, dass die zwei vom ersten print-Kommando ausgegebenen Werte gleich sind, und dass auch die vom

zweiten Kommando ausgegebenen Werte gleich sind.

```
  after ort
    print count(*),total(1)
 print avg(grosshand_preis),total(grosshand_preis)/count(*)
```

Die im folgenden aufgefuehrten Ausdruecke sind zu untersuchen. Dabei ist zu beachten, dass ein Vergleich das Ergebnisse Null (fuer eine falsche Aussage , FALSE) bzw. Eins (fuer eine wahre Aussage, TRUE) liefert. Mit einem der untenstehenden Ausdruecke soll fuer Suhl der niedrigste Preis ermittelt werden. Mit einem anderen Ausdruck soll fuer alle Preise, die ueber 9 Mark liegen, der Durchschnitt berechnet werden.

```
  Ausdruck
  ____________________________________________
  count (*)
  avg (preis)
  2 * (avg (preis) + 1.75)
  (min (preis) + max (preis)) / 2
  total (preis >= 5.00 and preis < 25.00)
  max (preis where preis < 500.00)
  min (preis where ort = 'Suhl*')
  count (* where preis >= 500.00)
  avg (preis where preis > 9.00)
```

7.4.6.2 Lokale Funktionen

Eine lokale Funktion ist eine Host-Sprachfunktion, die zusammen mit RPT geladen wird und mit der man nach eigenen Beduerfnissen von einem Reportskripts aus Formatierungen, Berechnungen und Aktualisierungen der Datenbank vornehmen kann. Damit ermoeglichen lokale Funktionen eine nutzerspezifische Verwendung von RPT. RPT besitzt bereits 3 eingebaute lokale Funktionen, die in den Abschnitten 7.4.6.2.1 bis 7.4.6.2.3 beschrieben sind. In Abschnitt 7.4.6.2.4 wird beschrieben, wie man eigene lokale Funktionen erstellen und an RPT linken kann.

Lokale Funktionen akzeptieren Argumentenlisten verschiedener Laenge und koennen Werte vom Typ char *, short, long und double liefern. Will man beispielsweise Daten im Format "January 1, 1986" ausgeben und nicht im Standardformat "1/1/86", so erhaelt man ein solches Datenformat unter Verwendung einer lokalen Funktionen, die eine Datenvariable im Format der Datenbank akzeptiert und einen Pointer auf eine Zeichenkette im gewuenschten Format zurueckgibt. Oder will man z.B. unter Verwendung von in der Datenbank gespeicherten Daten eine trigonometrische Funktion berechnen, so kann man eine lokale Funktion fuer die Berechnung des Wertes und dessen Uebergabe an den Report verwenden.

Die Verwendung von lokalen Funktionen unterliegt nur einer Einschraenkung. Das fuer die fragliche Funktion erforder-

liche Eingabeargument/Eingabeargumente, muss/muessen dann zur Verfuegung stehen, wenn die Funktion aufgerufen wird. Das bedeutet, dass lokale Funktionen in sort-Ausdruecken und in jedem beliebigen Gruppenkommando verwendet werden koennen.

Fuer den Aufruf von lokalen Funktionen gilt folgende Syntax:

```
funktions_name ( [arg_1, arg_2, ...])
```

Die Anzahl der erforderlichen Argumente ist verschieden und haengt davon ab, wieviele Argumente fuer die Funktion selbst erforderlich sind. Einige lokale Funktionen verlangen keine Argumente, aber es sind immer runden Klammern zu verwenden.

Im allgemeinen kann ein Argument ein beliebiger Ausdruck sein, der dort gueltig ist, wo die lokale Funktion verwendet wird. Die einzige Ausnahme besteht darin, dass numerische Funktionen in einem Argumentausdruck nicht gueltig sind. Dieses Hindernis kann umgangen werden, indem man die numerische Funktion berechnet, sie einer Variablen zuordnet und dann die Variable an die lokale Funktion uebergibt.

7.4.6.2.1 dow

SYNTAX

dow (datum)

VERWENDUNG

Diese lokale Funktion akzeptiert ein Reportfeld, eine Konstante oder Variable vom Typ DATE und gibt eine ganze Zahl zurueck, die den Wochentag angibt (0 = Sonntag, 6 = Samstag). dow kann in Verbindung mit der lokalen Funktion index verwendet werden, und dient der Ausgabe von Tagen in ASCII. Beispielsweise soll fuer den 1/1/86 der Wochentag ausgegeben werden. Es wird folgender Ausdruck verwendet:

```
index (dow (1/1/86), 'Sunday','Monday',
     'Tuesday','Wednesday','Thursday',
     'Friday','Saturday')
```

7.4.6.2.2 index

SYNTAX
index (nummer, string1, ..., stringn)

VERWENDUNG
Diese lokale Funktion akzeptiert eine Zahl, nummer, hinter der eine Reihe konstanter Zeichenketten oder Zeichenkettenausdruecke steht. Sie gibt die Zeichenkette zurueck, die durch die Zahl nummer indiziert ist (mit Null beginnend). So wird beispielsweise bei dem an index ergehenden Aufruf

```
index (1, 'abc', 'def', 'ghi')
```

die Zeichenkette 'def' zurueckgegeben.

7.4.6.2.3 mdy

SYNTAX
mdy (datum, nummer)

VERWENDUNG
Diese lokale Funktion akzeptiert ein Reportfeld, eine Konstante oder Variable vom Typ DATE, gefolgt von einer Zahl nummer. Sie liefert eine ganze Zahl die in Abhaengigkeit von der Zahl, nummer, den Monat, den Tag oder das Jahr eines gegebenen Datums darstellt. Und zwar in folgender Weise:

nummer	Bedeutung
0	Nummer des Monats, wobei 1 January, 2 February usw. ist
1	Nummer des Tages, wobei 1 der 1. January, 2 der 2. January usw. ist
2	Nummer des Jahres, wobei 0 1900 ist, 1 ist 1901 usw. bis 1999

Diese Funktion kann in Verbindung mit index zur Ausgabe von Daten unter Verwendung von Worten verwendet werden (z.B. 'Sunday, December 11, 1986').

7.4.6.2.4 Nutzerspezifische lokale Funktionen

Nutzerspezifische lokale Funktionen sind Host-Sprachfunktionen, die der Nutzer mit RPT schreiben, kompilieren und laden kann.

Will man das in diesem Abschnitt dargelegte Material verstehen, muss man mit Programmierungskonzepten der C-Sprache vertraut sein. Man kann nutzerspezifische lokale Funktionen von Reportskripten aus aufrufen, wobei Argumente uebergeben

und Werte zurueckgegeben werden, die im Report verwendet werden koennen. Lokale Funktionen koennen uneingeschraenkt alle Arten der Verarbeitung ausfuehren. Fuer jede zu verwendende Funktion wird RPT der Name, die Adresse und und der Typ der Funktion mitgeteilt, indem eine Tabelle folgenden Formats erstellt wird.

```
struct funces {
  char *fu_funnm;                    /* funktionsname */
  union vals (*fu_ptr)();                 /* adresse */
  int fu_type;                       /* return-typ */
};
```

Das Strukturelement fu_funnm ist ein Pointer auf den Namen der Funktion, mit dem im Reportskript des Nutzers auf die Funktion Bezug genommen wird. Union vals dient dazu, dass die Funktionen verschiedene Typen von Werten zurueckgeben koennen und dennoch die Vorteile von strong data typing aufweisen. Diese Union wird folgendermassen definiert:

```
union vals {
  char *c_val;                     /* zeichen-pointer */
  short s_val;                          /* short-wert */
  long l_val;                            /* long-wert */
  double d_val;                       /* double-wert */
};
```

Das Element fu_type gibt an, zu welchem Typ der von der Funktion gelieferte Wert gehoert. Die verschiedenen Typen sind in der Datei include/dbtypes.h aufgelistet, und es handelt sich dabei um die normalen internen Feldtypen der Datenbank. Bei der Ueberprufung von fu_type, stellt RPT fest, welcher Typ von Pointer von der vals Union verwendet werden muss.

RPT geht davon aus, dass die funcs-Tabelle den Namen func hat. RPT wird mit externer Bezugnahme auf diesen Namen kompiliert, so dass die Tabelle nicht definiert ist, wenn sie einen anderen Namen erhaelt. WEGA-DATA besitzt hat einen Prototyp fuer eine func-Tabelle, die in der Datei include/ufunc.c definiert ist. Sollen die in den vorhergehenden drei Abschnitten beschriebenen Funktionen verwendet werden, muessen die ersten drei Eintraege unveraendert bleiben und die Eintraege des Nutzers danach eingetragen werden.

Die als Prototyp gelieferte func-Tabelle ist wie folgt definiert:

```
#include "../../include/rptincl.h"
#include "../../include/dbtypes.h"

union vlas dow();
union vals index();
union vas mdy();
struct funcs func[] =
```

```
    {"dow", dow, INT,
     "index", index, STRING,
     "mdy", mdy, INT,
     0,   0,  0
  };
```

Zu beachten ist die Verwendung der zwei Dateien rptincl.h und dbtypes.h. rptincl.h enthaelt die Definitionen von vals, funcs und wargs, wobei es sich um eine Struktur handelt, die die an die lokalen Funktionen uebergebenen Argumente definiert. Es ist ebenfalls zu beachten, dass die Liste der Eintraege in die Tabelle mit einem Eintrag beendet werden muss, der drei Nullen enthaelt.

Wenn RPT ein Reportskript verarbeitet und feststellt, dass in einem Ausdruck eine lokale Funktion verwendet wurde, sucht es die Tabelle func nach dem Namen der Funktion ab. Dann wird die angegebene Funktion aufgerufen und ihr die Anzahl der Argumente und die Argumentenliste uebergeben. Wenn RPT die Funktion abgearbeitet hat, nimmt es den Rueckgabewert und setzt diesen in die Reportausgabe ein. Auf der naechsten Seite beginnt eine allgemeine Beschreibung, wie nutzerspezifische lokale Funktionen verwendet werden. Danach wird der Quellcode fuer eine Beispielfunktion angefuehrt.

UL_FUNC

NAME
    ul_func - nutzerspezifische lokale Funktion

SYNTAX
```
    #include "..../include/rptincl.h"
    #include "..../include/dbtypes.h"

    union vals ul_unc (count, args)
    int count;
    struct uargs *args;
```

BESCHREIBUNG
    In der Tabelle func muss der spezifische Name, Adresse und Rueckgabetyp jeder nutzerspezifischen lokalen Funktion definiert sein. count wird von RPT erzeugt und ist die Anzahl der vom Reportskript uebergebenen Argumente. ul_func muss einen Vergleich mit count durchfuehren, um zu sichern, ob dass richtige Anzahl von Argumenten uebergeben wurde.

    args ist ein Pointer auf eine Liste von Argumenten, mit dem in der Struktur uargs definierten Format.

```
      struct uargs{
        union vals ur_val;  /* Argumentwert oder Pointer */
        int ur_type;                      /* Argumenttyp */
      };
```

args[0] enthaelt den Wert des ersten Arguments, args[1] den Wert des zweiten Argumentes usw. Zur korrekten Bezugnahme auf diese Argumente muss ihr Typ bekannt sein. Dieser wird fuer jedes Argument durch den Wert ur_type gegeben.

In der folgenden Tabelle wird gezeigt, welches Element von vals mit dem jeweiligen Typ des Arguments zu verwenden ist.

Typ	Element
INT	s_val
LONG	l_val
DATE	s_val
AMT	l_val
STRNG	c_val
HAMT	d_val
HR	s_val
FLT	d_val

Wenn z.B. args[0].ur_type ein LONG ist, nimmt man mit args[0].ur_val.l_val korrekt bezug auf den Wert des Argumentes. Jede ul_func muss den Typ eines jeden Argumentes testen, um zu sichern, dass das Reportskript die richtige Art des Wert uebergeben hat.

Soll ein Wert an RPT zurueckgegeben werden, muss ul_func eine vals Union uebergeben, wobei das Element entsprechend der oben aufgefuehrten Tabelle dem jeweiligen Typ zugeordnet wird.

Das ist unproblematisch, nur bei Zeichenketten kann es eventuell zu Problemen kommen. Bei diesen wird nicht tatsaechlich ein Wert zurueckgegeben, sondern ein Pointer. Dabei ist zu sichern, dass der Pointer sich auf einen globalen Datenbereich bezieht und nicht auf eine Position im Stack (d.h., man darf keinen Puffer verwenden, der lokal innerhalb der Funktion vereinbart ist). Bezieht sich dieser Pointer auf eine Position im Stack, ergeben sich bei erneuter Verwendung dieser Positionen unbrauchbare Daten.

RUECKGABEWERTE
Ein Datenwert oder ein Pointer des in der func-Tabelle fuer diese bestimmte Funktion definierten Typs.

Es folgt der Quellkode fuer die eingebaute lokale Funktion dow.

```
#include "../../include/rptincl.h"
#include "../../include/dbtypes.h"
union vals dow (count,args)
int count;
struct uargs *args;
{
```

```
    union vlas v;
    v.s_val = 0;
    if(count = = 0)
    {
      return v;
    }
    else if(args[0].ur_type != DATE)
         {
           return v;
         }
         else
         {
           v.s_val = (args[0].ur_val.s_val + 32764) % 7;
           return v;
         }
  }
```

Die Funktion umfasst die zwei Standarddateien rptincl.h und dbtypes.h. Sie liefert eine vals Union vom Typ INT. Diese wird, wie bereits beschrieben, in der Tabelle func vereinbart. Das Argument count enthaelt 1, da das Reportskript einen einzelnen Ausdruck vom Typ DATE uebergeben soll. Das erste Element im Argumentbereich, args[0], muss den Wert des Datums enthalten. Ist count falsch, oder ist der Typ des Arguments unkorrekt, gibt die Funktion 0 zurueck. Man hat absichtlich einen konstanten Wert gewaehlt, so dass bei inkorrekten Aufrufen kein Fehler entsteht. Ist count richtig, wird der Rueckgabewert berechnet, im s_val Element der Rueckgabevariablen v gespeichert und an RPT zurueckgegeben.

Nachdem die Funktionen geschrieben sind, wird die Tabelle func in ufunc.c so modifiziert, dass sie Verweise auf die geschriebenen Funktionen enthaelt. Das geschieht am einfachsten, indem die von WEGA-DATA als Prototyp gelieferte Datei kopiert und dann editiert wird. Dann werden alle Quelldateien kompiliert und ein Archiv angelegt. Die Objektdatei ufunc.o muss die erste Datei im Archiv sein. Das Verzeichnis wird auf das im Anwenderfall verwendete Verzeichnis bin geaendert, und RPT wird unter Verwendung des folgenden Kommandos geladen (fuer archiv name setzt der Anwender den Pfadnamen der eigenen Archivdatei ein):

```
rpt.ld archiv name
```

Damit entsteht eine neue ausfuehrbare RPT-Datei, die man entweder im lokalen bin-Verzeichnis belassen kann oder ins Standardverzeichnis fuer ausfuehrbare WEGA-DATA-Dateien bringen kann. Es ist zu sichern, dass vor Verwendung von rpt.ld die entsprechenden Umgebungsvariablen gesetzt sind (siehe Abschnitt 1.1.3).

7.4.7 Variable

Unter Verwendung des Kommandos set kann das Ergebnis eines

beliebigen Ausdrucks einer Variablen zugeordnet werden. Variable sind oft recht nuetzlich, denn hat man erst einmal eine Variable auf einen Wert gesetzt, steht dieser Wert allen Ausdruecken, in denen eine Variable gueltig ist, global zur Verfuegung.

Variable koennen in einem beliebigen unbenannten Ausdruck in einem Gruppenkommando verwendet werden, Argumentausdruecke von numerischen Funktionen bilden jedoch eine Ausnahme. Sie koennen zur Erzeugung von Seitenzahlen und Abschnittnummern verwendet werden oder fuer komplexe Berechnungen und Manipulationen an aus Zeichen bestehenden Zeichenketten.

Die Namen von Variablen werden ebenso wie die Namen von Ausdruecken von dem Nutzer gewaehlt, die das RPT-Skript schreibt. Alle Namen muessen mit einem Buchstaben beginnen. Buchstaben, Ziffern und Unterstreichungszeichen koennen darauf folgen, wodurch ein Name entsteht, der nicht mehr als 32 Zeichen enthalten darf.

7.4.7.1 Bestimmung des Typs von Variablen

Der Typ einer Variablen muss nicht explizit im Programm vereinbart werden. Eine Variable wird vereinbart, waehrend sie benutzt wird. Ihr Typ wird durch das/die zugehoerige/n set-Kommando/s bestimmt.

Da Variable automatisch durch set-Kommandos vereinbart werden, ist darauf zu achten, sie nicht auf Werte gesetzt werden, deren Typen nicht miteinander vereinbar sind. Im folgenden die impliziten variablen Typen:

Numeric (numeric, amount, float)
Date
Time
String

Zu den numerischen impliziten Typen gehoeren die numeric, amount und float Datenbank-Feldtypen. Bei diesen handelt es sich nicht um unvereinbare Typen, da jeder numerische Wert numeric (maximale Laenge ist 9, kein Dezimalpunkt) als ein amount dargestellt werden kann, bei dem rechts vom Dezimalpunkt Nullen stehen und jeder Wert amount (maximale Laenge 11 Ziffern links vom Dezimalpunkt) kann als ein float dargestellt werden.

Wenn RPT den Typ einer numerischen Variablen bestimmt, gelten folgende Regeln. Es ist dabei zu beruecksichtigen, dass alle set-Kommandos in einem Programm beruecksichtigt werden, bevor ein Typ gewaehlt wird.

numeric - Der Ausdruck in jedem set-Kommando, der zum Setzen der Variablen benutzt wird, ergibt einen numerischen Wert.

amount - Der Ausdruck in jedem set-Kommando, der zum Setzen der Variablen benutzt wird, ergibt entweder einen numerischen Wert numeric oder einen Wert amount. Mindestens ein Ausdruck ergibt einen Wert amount.

float - Der Ausdruck in jedem set-Kommando, der zum Setzen der Variablen benutzt wird, ergibt entweder einen numerischen Wert numeric, einen Wert amount oder einen Wert float. Mindestens einer hat als Ergebnis einen Wert float.

Wird der Typ des Ergebnisses eines Ausdrucks bestimmt, wird beruecksichtigt, wie die Konstante geschrieben wurde. Zum Beispiel zeigt die Tabelle unten, wie man die Zahl Eins in verschieder Form schreiben kann. Auch der von jeder Form vorgeschriebene Typ wird gezeigt. Wenn man die in dieser Tabelle veranschaulichten Konzepte versteht, kann man unerwartete Ergebnisse vermeiden.

Konstante	Typ
1	numeric
1n	numeric
1.	amount
1.0	amount
1.00	amount
1a	amount
1.000	float
1f	float

Einer der Buchstaben n, a oder f kann an eine numerische Konstante angehaengt werden, was dazu fuehrt, dass RPT diese als den dargestellten Typ (wenn moeglich) interpretiert.

7.4.7.2 Initialisierung von Variablen

Jede Variable muss auf einen Wert gesetzt werden, indem mindestens einmal ein set-Kommando verwendet wird. Hier sind zwei Richtlinien, die zu beachten sind, wenn eine Variable auf einen Ausgangswert gesetzt werden muss (siehe Abschnitt Gruppenverarbeitung, dort wird die Reihenfolge, in der die Gruppenkommandos abgearbeitet werden, erlaeutert).

1. Das set-Kommando, das die Initialisierung uebernimmt, wird in das 'before report'-Gruppenkommando gebracht, wenn die Variable nur einmal initialisiert werden muss.

2. Das set-Kommando, das die Initialisierung uebernimmt, wird in das entsprechende 'before name'-Gruppenkommando gebracht, wenn die Variable vor der Verarbeitung von Gruppen detaillierter Informationen initialisiert werden

muss.

Vor Abarbeitung des ersten Gruppenkommandos werden alle Variablen auf Nullwerte initialisiert.

7.4.8 Gueltigkeit von Ausdruecken

Im folgenden wird in einer Tabelle eine Uebersicht ueber die Gueltigkeit von Ausdruecken gegeben. In der unteren Auflistung wird aufgefuehrt, wo sich die entsprechenden Ausdruecke befinden und dahinter stehen einige Zahlen. Die Zahlen beziehen sich auf die darueber aufgefuehrte Liste von Ausdruckskomponenten.

1. Konstanten	(siehe Abschnitt 7.4.4 und 7.4.5.1)
2. Lokale Funktionen	(siehe Abschnitt 7.4.6.2)
3. Felder	(siehe Abschnitt 7.4.3)
4. Namen von Ausdruecken	(siehe Abschnitt 7.4.1)
5. Die name Syntax	(siehe Abschnitt 7.4.1)
6. Variablen	(siehe Abschnitt 7.4.7)
7. Numerische Funktionen	(siehe Abschnitt 7.4.6.1)

Im sort-Kommando	1-5
In einem hinter der name-Syntax folgenden Ausdruck	1-4
In dem header-Gruppenkommando	1-6
In dem footer-Gruppenkommando	1-6
In dem detail-Gruppenkommando	1-6
In dem before-report-Gruppenlkommando	1-6
In einem before-name-Gruppenkommando	1-6
In einem after-name-Gruppenkommando	1-7
In dem after-report-Gruppenkommando	1,2,6,7
Im Argumentausdruck einer numerischen Funktion	1-4
Im Argumentausdruck einer lokalen Funktion	1-6

7.5 Gruppenwechselverarbeitung

Gruppenwechselverarbeitung nutzt die logische Gruppierung von sortierten Daten. Vor einer Gruppe von Datenelementen koennen zu deren Bezeichnung oder Einfuehrung Ueberschriften ausgegeben werden. Es koennen Berechnungen vorgenommen werden, wie z.B. eine gruppenweise Addition oder das Finden der Extremwerte innerhalb einer Gruppe. Ausserdem kann am Anfang eines Reports eine spezielle Report-Ueberschrift ausgegeben werden und am Ende des Reports kann ein Report-Nachsatz mit der Endsumme ausgegeben werden.

Die before- und after-Gruppenkommandos nutzen Gruppenwechselverarbeitung. Sie werden waehrend der Gruppenwechsel abgearbeitet. Gruppenwechsel treten zwischen logischen Gruppen von Daten auf. Da unsortierte Daten normalerweise nicht logisch gruppiert sind, koennen nur die Ergebnisse benannter Sortierausdruecke zu Gruppenwechseln fuehren.

7.5.1 Benannte Sortierausdruecke

Der Reportgenerator unterstuetzt Gruppenwechselverarbeitung, indem er die den benannten Ausdruecken in einem sort-Kommando entsprechenden Werte prueft. Werden verschiedene Ausdruecke hierarchisch sortiert, ist der sich am seltensten aendernde Sortierausdruck der uebergeordnete Sortierausdruck.

Die den Verkauf betreffende Eingabe wird nach Orten, innerhalb der Orte nach Postleitzahlen und innerhalb der Postleitzahlen nach Kunden sortiert. In diesem Fall ist der als ort benannte Ausdruck, der uebergeordnete Sortierbegriff. Er ist damit im Kommando sort der erste Ausdruck. Der Ausdruck mit der Benennung kunde ist der am meisten untergeordnete Sortierausdruck. Er steht im Kommando sort als letzter Ausdrucksname.

Im Beispiel sind die Ausdruecke ort, postleitzahl und kunde Reportfelder. Ein Reportfeld ist der einfachste und am haeufigsten verwendete Typ der benannten Ausdruecke. Der Wert eines Reportfeldes ist der Wert des zugehoerigen Datenelementes in der aktuellen Zeile der Eingabedatei, waehrend der Name des Reportfeldes (standardmaessig) der Name des Feldes selbst ist. Komplexeren Ausdruecken die aus verschiedenen Feldern, Operatoren und Funktionen bestehen, kann im sort-Kommando ein Name zugeordnet werden. Dann erfolgt jede weitere Bezugnahme im uebrigen Report immer durch diesen Namen.

In folgenden sollen wichtige Bestandteile eines angeneommenen Reports aufgefuehrt werden. Kommandos, die nicht fuer die Erklaerung der Gruppenwechselverarbeitung benoetigt werden, werden nicht gezeigt.

```
sort ort, postleitzahl, kunde

before ort
  print 'Ort:' in column 6, ort

before kunde
  need 4
  print 'Kunde' col 10,
        'Bestellung #' col 38, 'AMOUNT' col 54
  print 9[-] col 36, 11[-] col 51
  print kunde in col 10

detail
  print best_nr col 36, preis col 52 using '$$$,$$&.&&'

after kunde
  need 2
  print 11[-] in column 51
  print total (preis) column 51 using '$$$$,$$&.&&'
  skip 1
```

```
after ort
  need 2
  print 'Verkauf:' in column 6, ort,
        total (preis) column 49 using '$$,$$$,$$&.&&'
  print 56[-] column 6
  skip 1

after ort
  need 2
  print 'Verkauf:', ort,
        total (preis) column 49 using '$$,$$$,$$&.&&'
  print 61[-] skip 1

after report
  need 4
  print 'TOTAL Verkauf in diesem Report:',
        total (preis) col 49 using '$$,$$$,$$&.&&',
  print 'Durchschnitts-Kundenpreis:',
        avg (preis) col 52 using '$$$,$$&.&&'
  print 61[-] print 61[-]
```

7.5.2 Gruppenwechsel

Wenn die sortierten Werte, die benannten Sortierausdruecken entsprechen, verarbeitet werden, wird jeder von ihnen daraufhin ueberprueft, ob sich sein Wert geaendert hat. Unterscheidet sich ein Wert vom vorhergehenden, tritt ein Gruppenwechsel auf.

Im Beispiel tritt dann ein Gruppenwechsel auf, wenn alle Verkaeufe an einen bestimmten Kunden ausgegeben worden sind und der naechste Kunde verarbeitet werden soll. Ein "groesserer" Gruppenwechsel tritt dann auf, wenn von einem Ort oder einer Postleitzahl zum/zur naechsten uebergegangen wird. Aendert sich die Postleitzahl, aendern sich auch Ort und Kunden. Laut Definition aendern sich bei Aenderung eines sortierten Wertes auch alle Werte, die den diesem Wert untergeordenten Sortierausdruecken entsprechen.

Oft wird die Ausgabe nur ueber Gruppenwechsel ausgedruckt. Bei vielen Reporten ist nur die Ausgabe der verarbeiteten Informationen erforderlich, nicht aber die von detaillierten Informationen.

7.5.3 Gruppenkommandos zur Gruppenwechselverarbeitung

Waehrend der in einem Report auftretenden Gruppenwechsel kann die Ausgabe von z.B. Ueberschriften oder Summen erfolgen. Zu diesem Zweck werden die Gruppenkommandos before und after verwendet.

Die Gruppenkommandos before werden benutzt, um Ueberschriften fuer Abschnitte oder fuer Gruppen auszugeben, bevor eine Zeilengruppe in der Eingabedatei mit identischen Sor-

tierschluesseln verarbeitet wird. Wird ein Gruppenkommando before ausgefuehrt, steht dieser die Zeile mit detaillierten Informationen zur Verfuegung, die gerade verarbeitet werden soll. after-Gruppenkommandos dienen der Ausgabe von Summen fuer Gruppen oder ahenliche nach Gruppen ermittelte Ergebnisse. Ein after-Gruppenkommando verfuegt ueber Informationen ueber die logische Gruppe detaillierter Informationen, die gerade verarbeitet wird.

Es gibt zwei Typen von before- und after-Gruppenkommandos. Die 'before name'- und 'after name'-Gruppenkommandos werden im gesamten Hauptteils eines Reports waehrend der Gruppenwechsel ausgefuehrt. 'before report'- und 'after report'-Gruppenkommandos laufen, wie ihre Namen besagen, zu Anfang und zu Ende eines Reports ab.

7.5.3.1 Die Gruppenkommandos before- und after-name

Ein 'before name'-Gruppenkommando und ein 'after name'-Gruppenkommando kann jedem im Kommando sort auftretenden benannten Ausdruck zugeordnet werden. Das ist der Zweck der Option name.

Nicht jedes existierende 'before name'- und 'after name'-Gruppenkommando wird bei Auftreten eines Gruppenwechsels abgearbeitet. Im als Beispiel verwendeten Verkaufsreport, in dem die Eingabe ortsweise, postleitzahlweise innerhalb eines Ortes und kundenweise innerhalb der Postleitzahl sortiert ist, tritt, da sich der Wert ort aendert, bei Auftritt eines Gruppenwechsels folgendes auf:

Zuerst werden die 'after kunde'-Kommandos ausgefuehrt. Dann die 'after ort'-Kommandos. Dann die 'before ort'-Kommandos und danach die Kommandos, in dem Gruppenkommando 'before kunde'. Das den Werten postleitzahl zugeordnete Gruppenkommando wird nicht abgearbeitet, weil der Sortierausdruck fuer postleitzahl, dem gerade verarbeiteten Gruppenwechsel uebergeordnet ist.

'before'-Gruppenkommandos werden verarbeitet, indem man mit dem Gruppenkommando beginnt, die dem am meisten uebergeordneten Sortierausdruck des Gruppenwechsels zugeordnet ist. Die after-Gruppenkommandos werden in umgekehrter Reihenfolge verarbeitet.

7.5.3.2 Die Gruppenkommandos before- und after-report

Selbst wenn sich waehrend des gesamten Reportprozesses keine sortierten Werte aendern, treten zwei Gruppenwechsel immer auf. Die erste am Anfang des Reports und die zweite ganz am Ende. Diese Gruppenwechsel werden anders behandelt, als die, die waehrend der Bearbeitung des Reports auftreten.

Da vor der am Anfang des Reports auftretenden Gruppenwechsel keine detaillierten Informationen stehen, wird kein after-Gruppenkommando ausgefuehrt. Das 'before report'-Gruppenkommando wird ausgefuehrt, und dann werden alle existierenden 'before name'-Gruppenkommandos abgearbeitet. Das geschieht, noch bevor irgendwelche detaillierten Informationen fuer die Ausgabe beruecksichtigt werden. Es ist sinnvoll, eine den Report kennzeichnende, beschreibende Ueberschrift unter Verwendung des 'before report'-Gruppenkommandos auszugeben.

Die Ausgabe des 'after report-Gruppenkommandos erfolgt am Ende des Reports. Sie besteht normalerweise aus der Endsumme oder aus aehnlichen Gesamtergebnissen. Das 'after report'-Gruppenkommando wird erst dann ausgefuehrt, wenn alle existierenden 'after name'-Gruppenkommandos abgearbeitet sind. Da keine detaillierten Informationen folgen, wird kein 'before'-Gruppenkommando ausgefuehrt.

7.6 Kommandos, die keine Gruppenkommandos sind

Die in diesem Abschnitt erlaeuterten Kommandos sollen einmalig in einem Report verwendet werden, naemlich zum Setzen der Grundparameter des Reports, z.B. Beschreibung der Eingabedatei, Sortierspezifikationen, die Laenge des verwendeten Papiers, Breite und Raender des Reports usw. Sie sollen ausserhalb der Gruppenkommandos verwendet werden und es wird empfohlen, dass sie am Anfang des Reportskripts erscheinen. Sie koennen jedoch auch in beliebiger Reihenfolge an beliebiger Stelle im Skript auftreten, ausser in der Eingabesektion input, die am Anfang stehen muss. Steht nach einem Gruppenkommando eines dieser Kommandos, wird dieses Gruppenkommando beendet. Das heisst, dass das naechste Kommando entweder wieder ein Kommando sein muss, das kein Gruppenkommando ist oder dass ein neues Gruppenkommando anfaengt.

7.6.1 Unterer Rand - bottom margin

SYNTAX
bottom margin nummer

VERWENDUNG
Dieses Kommando setzt die Groesse des unteren Randes. nummer bezieht sich auf die Anzahl der Zeilen, die unten auf einer Seite jeweils nicht gedruckt werden duerfen. Standardnummer ist 2.

7.6.2 Ende - end

SYNTAX
```
    end
```

VERWENDUNG
Dieses Kommando zeigt das Ende des RPT-Skripts an. Alle Kommandos, die nach end folgen, werden ignoriert.

7.6.3 Eingabe - input

SYNTAX
```
    input  |  name [  |  numeric  length  |  ]    |
           |          |  string   length  |       |
           |          |  amount   length  |       |
           |          |  float    length  |       |
           |          |  date             |       |
           |          |  time             |       |
           |  datsat.feld                         |

      [,   |  name [  |  numeric  length  |  ]    | ...]
           |          |  string   length  |       |
           |          |  amount   length  |       |
           |          |  float    length  |       |
           |          |  date             |       |
           |          |  time             |       |
           |  datsat.feld                         |
```

VERWENDUNG
Dieses Kommando beschreibt das Format jeder Zeile der Eingabedatei. Jedes Reportskript verlangt eine Eingabesektion. Jedes Feld in der Eingabedatei muss in der Eingabefeldliste eingetragen sein. Die Eintraege muessen miteinander uebereinstimmen (d.h. der erste Eintrag in der Eingabefeldliste muss das erste Feld in der Eingabedatei beschreiben, der zweite Eintrag muss das zweite Feld beschreiben usw.)

Eine Eingabefeldliste kann in zwei verschiedenen Formen auftreten. Sie kann entweder ein beliebiger Name, name sein, dem eine in eckige Klammern eingeschlossene Typspezifikation folgt, oder sie kann der ausfuehrliche Name einer Datenbank sein. Das ist derselbe Feldname, der vom SQL-Kommando generiert wird. Die Typenspezifikation ist eins der folgenden Schluesselworte: numeric, string, amount, float, date oder time, dem eine Laenge folgt (ausser bei den Feldtypen date und time, die Standardlaenge haben). Die Laenge ist gleich der Laenge der Felder in der Datenbank. Verwendet man den ausfuehrlichen Namen eines Datenbankfeldes, braucht man den Typ nicht angeben, aber man muss diesem Namen den Namen des Datensatztyps voranstellen, dem er angehoert. RPT sucht den Typ und die Laenge im Datenwoerterbuch. Es ist zu sichern, dass das Feld nicht vom Typ COMB ist. Zur Verwendung von COMB Feldinformationen, werden

die das COMB Feld bildenden Datenbankfelder benutzt, die keine COMB Felder sind.

7.6.4 Linker Rand - left margin

SYNTAX
left margin nummer

VERWENDUNG
Dieses Kommando setzt die Breite des linken Randes. nummer bezieht sich auf die Anzahl der Spalten, die auf jeder Seite links freigelassen werden. Standardnummer ist 2.

7.6.5 Laenge - length

SYNTAX
length nummer

VERWENDUNG
Dieses Kommando gibt die Laenge der Reportseite an. nummer bezieht sich auf die Anzahl der ausgegebenen Zeilen pro Seite, einschliesslich oberer und unterer Rand. Standard ist 66, wobei davon ausgegangen wird, dass Papier fuer Zeilendrucker verwendet wird.

7.6.6 Trennzeichen - separator

SYNTAX
separator 'zeichen'

VERWENDUNG
Mit diesem Kommando kann man RPT mitteilen, welches Trennzeichen in der Eingabedatei verwendet wird. Feldtrennzeichen sind erforderlich, um das Ende des einen Feldes und den Beginn des naechsten Feldes anzugeben, da Zeichenkettenfelder Leerzeichen enthalten koennen. Standardmaessig wird als Trennzeichen der senkrechte Strich verwendet, der auch bei SQL verwendet wird. Das zeichen kann ein beliebiges einzelnes Zeichen sein, einschliesslich eines nicht druckbaren Steuerzeichens.

7.6.7 Sortieren - sort

SYNTAX
sort ausdr [desc]
[, ausdr [desc] ....]

VERWENDUNG
Dieses Kommando wird benutzt, um die Reihenfolge der Reportausgabe zu bestimmen. Wird das sort-Kommando nicht verwendet, erfolgt die Ausgabe in keiner bestimm-

ten Reihenfolge. Standardmaessig fuehrt das sort-Kommando zur Ausgabe in aufsteigender Reihenfolge der angegebenen Ausdruecke. Jeder Ausdruck, hinter dem desc steht, wird vom Kommando sort in fallender Reihenfolge ausgegeben. Innerhalb eines Programms darf nur ein sort-Kommando erfolgen. sort ist kein Gruppenkommando.

Die zuerst aufgelisteten Ausdruecke (von links nach rechts) sind die uebergeordneten Sortierausdruecke und die zuletzt aufgefuehrten sind die untergeordneten Sortierausdruecke. Die Sortierung zusaetzlicher Ausdruecke aendert nicht die Ordnung der uebergeordneten Sortierausdruecke. Untergeordnete Sortierausdruecke bestimmen nur die Ordnung der Datensaetze innerhalb von Gruppen mit uebergeordneten Sortierausdruecken gleichen Wertes. Auf diese Weise werden die fuer eine sinnvolle Gruppenwechsel-Verarbeitung erforderlichen hierarchischen Gruppen gebildet.

Ein Report kann Gruppenwechselverarbeitung nur dann voll ausnutzen, wenn benannte Ausdruecke sortiert werden. Die Namen von Ausdruecken koennen zur Ausloesung von 'before name'- und 'after name'-Gruppenkommandos verwendet werden. Ein Datenbankfeld ist ein benannter Ausdruck. Der Wert, den ein Datenbankfeld auf der aktuellen Eingabezeile besitzt, ist gleich dem Wert des Ausdrucks, und der Feldname ist standardmaessig gleich dem Ausdrucksnamen. Alle Ausdruecke, die aus Datenbasisfeldern, Konstanten, lokalen Funktionen und anderen Ausdrucksnamen bestehen, koennen unter Verwendung folgender Syntax benannt werden. ausdr siehe oben (siehe Abschnitt 7.4.1):

```
<name> ausdr
```

Ein benannter Ausdruck kann ohne weiteres sortiert werden, jedoch nicht im Zusammenhang mit einem 'before name'- oder 'after name'-Gruppenkommando.

7.6.8 Oberer Rand - top margin

SYNTAX

```
top margin nummer
```

VERWENDUNG

Dieses Kommando setzt die Groesse des oberen Randes. nummer bezieht sich auf die Anzahl der Zeilen, die auf jeder Seite oben freigelassen wird. Standard ist 2.

7.6.9 Breite - width

SYNTAX
    width nummer

VERWENDUNG
    Dieses Kommando gibt die Breite der Seite eines Reports an. nummer bezieht sich auf die Anzahl der zur Verfuegung stehenden Spalten oder Ausgabepositionen. Der linke Rand (siehe Kommando linker Rand) ist inbegriffen, so dass die Breite nicht veraendert werden muss, wenn der linke Rand geaendert wird.

    Standard ist 132, d.h. es wird davon ausgegangen, dass Zeilendruckerpapier verwendet wird.

7.7 Gruppenkommandos

Die Gruppenkommandos koennen in jeder gewuenschten Reihenfolge verwendet werden. Die Reihenfolge ihrer Verarbeitung wird von der Logik des Reportgenerators bestimmt und nicht von der Reihenfolge, in der sie auftreten.

Jedes Programm muss mindestens ein Gruppenkommando enthalten. Einige Reporte koennen auch erzeugt werden, indem nur das Gruppenkommando detail verwendet wird. Bei anderen Reporten ist nur eine zusammenfassende Information erforderlich, und sie werden daher nur unter Verwendung des 'after report'-Gruppenkommandos erzeugt.

Im folgenden die Syntax des 'before report'-Gruppenkommandos:

```
before report
  gruppenkommando_kommandos
```

Die Worte 'before report' bezeichnen das Gruppenkommando. Die darauffolgenden gruppenkommando_kommandos bilden den ablauffaehigen Teil des Gruppenkommandos. Die Struktur eines Programms wird deutlicher, wenn diese etwas eingerueckt werden.

Oft ist das schrittweise Schreiben von Programmen sinnvoll. Daher braucht ein Gruppenkommando keine Kommandos enthalten. So kann beispielsweise das Gruppenkommando-Kennzeichen header nur deshalb im Programm sein, um an das zu einem spaeteren Zeitpunkt erfolgende Einfuegen der entsprechenden Gruppenkommando-Kommandos zu erinnern.

Ein Gruppenkommando wird durch ein anderes Gruppenkommando-Kennzeichen, durch ein Kommando, das kein Gruppenkommando-Kommando ist oder durch das Ende der Datei beendet.

Weitere Informationen ueber Gruppenkommandos siehe Abschnitte 7.5.3 und 7.10.2.

7.7.1 after report

SYNTAX

```
after report
  gruppenkommando_kommandos
```

VERWENDUNG

Diese Ausgabe eines Gruppenkommandos erfolgt ganz am Ende des Reports. Sie besteht normalerweise aus den abschliessenden Summen oder Ergebnissen. Die Ausfuehrung dieses Gruppenkommandos wird waehrend des letzten Gruppenwechsels ausgeloest, nachdem alle 'after name'-Gruppenkommandos ausgefuehrt wurden (siehe Abschnitt 7.5.3).

Im allgemeinen koennen in den in dem 'after report'-Gruppenkommando verwendeten Ausdruecke von Kommandos nur numerische und lokale Funktionen, Variable und Konstanten auftreten. Ausdruecke koennen in dem Gruppenkommando 'after report' nicht benannt werden. Dieses Gruppenkommando darf in einem Programm nur einmal auftreten.

Alle Gruppenkommando-Kommandos sind in dem 'after report'-Gruppenkommando gueltig.

7.7.2 after name

SYNTAX

```
after name
  gruppenkommando_kommandos
```

VERWENDUNG

Die Gruppenkommandos 'after name' werden fuer die Ausgabe von Gruppensummen oder aehnlichen Ergebnissen benutzt, die erfolgt, nachdem eine Gruppe detaillierter Informationen ausgegeben wurde.

name ist der Name eines im Kommando sort erscheinenden Ausdrucks. Hat ein Ausdruck im Kommando sort keinen Namen erhalten und ist dieser Ausdruck kein Datenbankfeld, (standardmaessig ist der Feldname der Name des ausdrucks), kann kann er nicht mit einem Gruppenkommando 'after name' assoziiert werden. Jedes Gruppenkommando 'after name' muss fuer ihre Option name einen anderen Ausdrucksnamen haben.

Bei Auftreten eines Gruppenwechsels werden die entsrechenden Gruppenkommandos 'after name' in umgekehrter Reihenfolge des Auftretens der benannten Ausdruecke im Kommando sort ausgefuert (siehe Abschnitt 7.5, Gruppenwechselverarbeitung).

Alle Gruppenkommando-Kommandos sind hier zugelassen.

7.7.3 before report

SYNTAX

before report
gruppenkommando_kommandos

VERWENDUNG

Die Abarbeitung dieses Gruppenkommandos erfolgt ganz am Anfang des Reports. Sie wird ausgeloest durch den ersten Gruppenwechsel und steht vor allen Gruppenkommandos 'before name' (siehe Abschnitt 7.5). Die erste Zeile der detaillierten Informationen wird ausgewertet und waehrend der Gruppenwechselverarbeitung beibehalten. Deshalb steht sie diesem Gruppenkommando zur Verfuegung.

Dieses Gruppenkommando kann in einem Programm nur einmal auftreten. Ist es vorhanden, wird auf der ersten Seite oben kein Gruppenkommando header ausgefuehrt. Es ist sinnvoll, eine den Report kennzeichnende Ueberschrift unter Verwendung dieses Gruppenkommandos auszugeben. Dann kann fuer den restlichen Report das Gruppenkommando header fuer die Ausgabe abgekuerzter Ueberschriften benutzt werden.

Alle Gruppenkommando-Kommandos sind hier zugelassen.

7.7.4 before name

SYNTAX

before name
gruppenkommando_kommandos

VERWENDUNG

'before name'-Gruppenkommandos werden fuer die Ausgabe von Gruppenueberschriften verwendet, durch die eine aus aehnlichen Einzelzeilen bestehende Gruppe bezeichnet oder eingeleitet wird.

name ist der Name eines in einem sort-Kommando erscheinenden Ausdrucks. Wird ein Ausdruck nicht in einem sort-Kommando benannt und ist er kein Datenbankfeld (standardmaessig ist der Feldname gleich dem Namen des Ausdrucks), kann er keinem 'before name'-Gruppenkommando zugeordnet werden.

Jedes 'before name'-Gruppenkommando muss fuer ihre Option name einen anderen Ausdrucksnamen haben.

Tritt ein Gruppenwechsel auf, werden die entsprechenden 'before name'-Gruppenkommandos in der Reihenfolge ausgefuehrt, in der die benannten Ausdruecke im sort-Kommando auftreten (siehe Abschnitt 7.5, Gruppenwechselverarbeitung).

Alle Gruppenkommando-Kommandos sind hier zugelassen.

7.7.5 detail

SYNTAX
detail
gruppenkommando_kommandos

VERWENDUNG
Dieses Gruppenkommando wird fuer jede Zeile der Eingabedatei ausgefuehrt. Es kann zur Ausgabe der Einzelzeilen eines Reports verwendet werden ebenso wie fuer die Ausfuehrung einzelner Berechnungen. Es darf innerhalb eines Programms nur einmal auftreten.

Alle Gruppenkommando-Kommandos sind hier zugelassen.

7.7.6 Nachspann - footer

SYNTAX
footer
gruppenkommando_kommandos

VERWENDUNG
Die Abarbeitung dieses Gruppenkommandos erfolgt unten auf jeder Seite unmittelbar ueber dem unteren Rand.

Da das if-Kommando innerhalb dieses Gruppenkommandos auftreten kann, kann die Anzahl der Ausgabezeilen verschieden sein. Fuer einen gegebenen Report beginnt der footer immer auf derselben Ausgabezeile einer Seite. Diese Zeile wird so gewaehlt, dass die maximale Anzahl der ausgegebenen footer-Zeilen unmittelbar ueber dem unteren Rand gedruckt wird. Dieses Gruppenkommando darf nur einmal innerhalb eines Programms auftreten.

Mit Ausnahme von page und need sind alle Gruppenkommandos zugelassen.

7.7.7 Vorspann - header

SYNTAX
header
gruppenkommando_kommandos

VERWENDUNG
Die Abarbeitung dieses Gruppenkommandos erfolgt unmittelbar nach dem oberen Rand einer jeden Seite des Hauptteils des Reports. Wird das Gruppenkommando 'before report' nicht verwendet, erfolgt die Ausgabe dieses Gruppenkommandos auch nach dem oberen Rand der ersten Seite. Dieses Gruppenkommando darf in einem Programm nur einmal auftreten.

Die Ausfuehrung dieses Gruppenkommandos kann an verschiedener Stelle des Reports unterdrueckt werden, indem ein page-Kommando mit der Option 'no header' verwendet wird.

Mit Ausnahme von page und need sind alle Gruppenkommandos zugelassen.

7.8 Kommandos in Gruppenkommandos

Die in diesem Abschnitt beschriebenen Kommandos koennen nur innerhalb von Gruppenkommandos verwendet werden. Es handelt sich um Gruppenkommando-Kommandos. Steht eines dieser Kommandos unmittelbar nach einem Kommando, das kein Gruppenkommando-Kommando ist oder steht ein solches in einem Programm vor einem Gruppenkommando-Schluesselwort, handelt es sich um einen Fehler.

7.8.1 if

SYNTAX

```
if ausdr then
          kommando
       [else
          kommando]
```

Dabei ist kommando entweder ein einzelnes Gruppenkommando-Kommando oder eine in die Schluesselworte begin und end eingeschlossene Folge von Gruppenkommando-Kommandos.

```
  begin
    gruppenkommando_kommando_1
                  .
                  .
                  .
    gruppenkommando_kommando_n
  end
```

VERWENDUNG

Dieses Kommando wird verwendet, um Ausgabe oder Berechnungen in Abhaengigkeit davon zu variieren, ob der von dem logischen Ausdruck ausdr angezeigte Zustand WAHR oder FALSCH ist. Ein logischer Ausdruck muss verwendet werden (siehe Abschnitt 7.4.2). Ist der Ausdruck WAHR (ungleich Null), wird das unmittelbar nach dem Schluesselwort then folgende Gruppenkommando-Kommando ausgefuehrt. Ist der Ausdruck FALSCH (Null), wird das Kommando ignoriert und das nach dem Schluesselwort else stehende Gruppenkommando_kommando wird ausgefuehrt. Es ist zu beachten, dass else optionell ist.

Ein in einem if-Kommando benutztes Gruppenkommando-Kommando muss auch ohne if gueltig sein, das heisst,

dass page und need in einem innerhalb von header oder footer verwendeten if-Kommandos nicht gueltig sind, da sie, wenn sie allein verwendet werden, dort nicht gueltig sind.

Es ist zu beachten, dass if-Kommandos geschachtelt sein koennen. So ist das folgende ein gueltiges if-Kommando:

```
  if ausdr1 then begin
    if ausdr2 then
      print feld1,feld2
    else
      print feld3,feld4
 end
 else begin
    if ausdr3 then
       print feld5
    else
       print feld6
 end
```

7.8.2 need

SYNTAX
need nummer

VERWENDUNG
Dieses Kommando gibt die Anzahl der Zeilen an, die auf einer aktuellen Seite zur Fortsetzung der Ausgabe auf dieser Seite zur Verfuegung stehen muessen. Sind nicht genuegend Zeilen vorhanden, erfolgt Blattvorschub (mit den entsprechenden footers und headers) und die Ausgabe wird auf der naechsten Seite fortgesetzt. Dieses Kommando ist innerhalb der Gruppenkommandos header und footer nicht gueltig.

7.8.3 Seite - page

SYNTAX
page [, no footer] [,no header]

VERWENDUNG
Dieses Kommando bewirkt einen Blattvorschub. Sind footer und header vorhanden, wird der footer auf der aktuellen Seite unten und der header auf der naechsten Seite oben ausgegeben.

Die Option 'no footer' wird verwendet, wenn die Ausfuehrung des Gruppenkommandos footer auf der aktuellen Seite unten unterdrueckt werden soll.

Die Option 'no header' unterdrueckt die Ausfuehrung des Gruppenkommandos header auf der naechsten Seite oben. Dadurch kann anstelle des header etwas anderes ausgege-

ben werden. Dieses Kommando ist nicht in den Gruppenkommandos header und footer gueltig.

7.8.4 Drucken - print

SYNTAX

```
print [ ausdr [[in] col[umn] nummer] [using format]

    [,ausdr [[in] col[umn] nummer] [using format] ....]

                          [| no newline |]]
                          |  centered   |
```

VERWENDUNG

Jede Ausgabe erfolgt unter Verwendung des Kommandos print. Das ist moeglich, weil die Ausdruecke mehrfach verwendbar sind. Jeder Ausdruck wird berechnet und sein Ergebnis wird ausgegeben. Jedes Kommando print beschreibt eine ausgegebene Zeile (oder weniger als eine Zeile - die Option 'no newline' wird weiter unten beschrieben. Dieses Kommando ist nur gueltig, wenn es in einem Gruppenkommando verwendet wird.

Wird das Schluesselwort print fuer sich allein verwendet, wird eine Leerzeile ausgegeben. Das kann auch erreicht werden, wenn man das skip-Kommando verwendet. Geht die Laenge einer Ausgabezeile ueber die vom width-Kommando festgelegte Begrenzung hinaus, wird die Zeile abgeschnitten.

Viele Reporte verlangen keine komplizierten print-Kommandos. Abstaende und Formatierung werden als Standard geliefert. Ausdruecke koennen ausgegeben werden, indem sie einfach in einem print-Kommando aufgelistet werden. Wird keine Spaltennummer angegeben, wird standardmaessig dafuer gesorgt, dass zwischen zwei nebeneinanderstehenden Ausdruecken mindestens ein Leerzeichen steht. Spaltennummern werden verwendet, wenn eine absolute Positionierung gewuenscht wird. Spalte 1 ist die unmittelbar nach dem linken Rand stehende Ausgabeposition.

Folgende Datenbankfelder werden im Beispiel unten verwendet: price und mrkdwn (Typ amount), icode (string, Laenge 3) und iname (string). Im folgenden ein Beispiel fuer ein print Kommando, in dem die Verwendung der Leerzeichen gezeigt wird.

```
  print gpreis - ipreis, kuname, kort

  4.95 Retisch Irgendwo
```

Soll eine Meldung in Spalte 4 (vom linken Rand aus werden 3 Ausgabepositionen frei gelassen) beginnend ausgegeben werden, kann dies durch eins der folgenden Kommandos geschehen:

```
print 'TOTAL' in col 4
print ' ','TOTAL'
print 2[],'TOTAL'
print '  TOTAL'
```

Mit Angabe der Spaltenoption wird der Standardabstand aufgehoben. Sie kann verwendet werden, wenn zwei Ausdruecke nebeneinander ausgegeben werden sollen. Im naechsten Beispiel wird die eingebaute Variable pageno ausgegeben. pageno wird von RPT automatisch nach Ausgabe einer jeden Seite inkrementiert. Das folgende Kommando kann zur Ausgabe von pageno benutzt werden:

```
print '--' col 31,
  pageno col 33 using '&&','--' col 35
```

Damit wuerden die in den Spalten 31 bis 36 auftretenden Seitennummern z.B. so aussehen: --03-- bzw. --48--.

Wird die Option 'no newline' verwendet, steht die Ausgabe des naechsten print-Kommandos hinter der Ausgabe des aktuellen Kommandos und zwar auf derselben Zeile. Nutzen if-Kommandos die Option 'no newline', kann man das if-Kommando verwenden, um die Ausgabe auf einer einzelnen Zeile zu variieren.

Wird die Option centered verwendet, wird die gesamte ausgegebene Zeile in die Mitte der Reportbreite gerueckt. Das ist insbesondere dann zu empfehlen, wenn Titel in der Mitte der Seite stehen sollen.

Die Felder numeric, float und amount werden normalerweise in Standardfeldbreite ausgegeben. Die Standards sind folgendermassen:

```
numeric     9 Leerzeichen breit
amount     10 Leerzeichen breit
float      16 Leerzeichen breit
```

Diese Breite kann unter Verwendung der Option using und einem aus Sonderzeichen bestehenden Format geaendert werden, wobei fuer jede Position in der Feldbreite ein Sonderzeichen verwendet wird. Das Sonderzeichen gibt an, was an der entsprechenden Position auszugeben ist. Fuer die Felder numeric und amount werden folgende Sonderzeichen verwendet (dabei sollte man von der Vorstellung ausgehen, dass ein Feld von rechts nach links formatiert und ausgegeben wird.):

\# - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Leerzeichen gedruckt. Damit wird ein numerisches Feld links mit Leerzeichen aufgefuellt.

& - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird eine Null gedruckt.

Damit wird ein numerisches Feld links mit Nullen aufgefuellt.

* - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Stern gedruckt.

$ - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Dollarzeichen gedruckt. Wurde das Dollarzeichen bereits gedruckt, wird ein Leerzeichen gedruckt.

+ - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Plus-Zeichen gedruckt. Wurde das Plus-Zeichen bereits gedruckt, wird ein Leerzeichen gedruckt.

- - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Handelt es sich um eine negative Zahl, wird ein Minuszeichen gedruckt. Wurde ein Minuszeichen bereits gedruckt, wird ein Leerzeichen gedruckt.

(- Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Handelt es sich um eine negative Zahl, wird eine linke runde Klammer gedruckt. Wurde bereits eine linke runde Klammer gedruckt, wird ein Leerzeichen gedruckt.

) - Handelt es sich hierbei um eine negative Zahl, wird an dieser Stelle eine rechte runde Klammer gedruckt.

, - Befindet sich links von dieser Stelle eine Ziffer, wird ein Komma gedruckt. Andernfalls wird ein Leerzeichen gedruckt.

.- An dieser Stelle soll ein Dezimalpunkt gedruckt werden.

Fuer Felder vom Typ FLOAT wird eine Druckspezifikation benutzt, die genau der printf-Funktion aus C entspricht. Die Druckspezifikation hat folgendes Format:

```
%[-][minimale_feld_breite][.][genauigkeit] f|e|g
```

Optionen in den eckigen Klammern "[" und "]" muessen nicht angegeben werden. Die zur Abgrenzung der Elemente in der Liste verwendeten Vertikalstriche "|" geben an, dass ein Element in der Liste auszuwaehlen ist. Das Prozentzeichen "%" muss angegeben werden.

Das vor der Konvertierungsspezifikation stehende optionelle Minuszeichen "-" zeigt an, dass das Ergebnis in der Feldbreite linksbuendig ausgerichtet werden soll. Hat das Ergebnis weniger Zeichen, wird es auf minimale_feld_breite aufgefuellt. Mit genauigkeit wird die Anzahl der Ziffern angegeben, die hinter dem Dezimalpunkt stehen sollen. Wenn genauigkeit als 0 angegeben wird, werden hinter dem Dezimalpunkt keine Ziffern gedruckt. Wird kein Punkt angege-

ben, wird davon ausgegangen, dass die Zahl vor dem Konvertierungszeichen die Genauigkeit ist. Die Konvertierungszeichen (f, e und g) haben folgende Bedeutung:

f - Das Feld wird in eine dezimale Notation der Form [-]ddd.ddd konvertiert, wobei die Anzahl der nach dem Dezimalpunkt stehenden Ziffern gleich der spezifizierten Genauigkeit ist. Wird keine Genauigkeit angegeben, werden 6 Ziffern ausgegeben. Wird die Genauigkeit explizit mit 0 angegeben, wird kein Dezimalpunkt gedruckt.

e - Das Feld wird in der Form [-]d.ddde+-dd konvertiert, wobei eine Ziffer vor dem Dezimalpunkt steht und die Anzahl der Ziffern nach dem Dezimalpunkt gleich der angegebenen Genauigkeit ist. Wird die Genauigkeit nicht angegeben, werden 6 Ziffern ausgegeben. Wird die Genauigkeit explizit mit 0 angegeben, wird kein Dezimalpunkt gedruckt.

g - Das Feld wird in die Form f oder in die Form e konvertiert, je nachdem, welche der beiden Formen bei minimaler Platzbeanspruchung die volle Genauigkeit liefert.

In den folgenden Beispielen wird veranschaulicht, welche Wirkung die verschiedenen Schablonen auf die Ausgabe haben.

Format	Wert	Resultat
"#####"	123	" 123"
"#####.##"	0	" "
"#####.&&"	0	" .00"
"+++,+++,+++"	23456	" +23,456"
"---,---.&&"	23456.78	" 23,456.78"
"---,---.&&"	-2345.67	" -2,345.67"
"%10.2f"	12.3	" 12.30"
"%10.2f"	123.456	" 123.46"
"%12.4e"	123.456	" 1.235e+02"
"%10.4g"	123.456	" 123.4"
"%8.4g"	123456789	"1.23e+08"

Die Zeichenkettenfelder und Konstanten koennen auch unter Verwendung der Option using formatiert werden. Normalerweise werden diese in voller Breite ueber die ganze Seite ausgegeben. Moechte man ein langes Zeichenkettenfeld oder eine Konstante in eine Spalte formatieren, die schmaler ist als die Laenge, kann man ein Format verwenden, um die Breite der ausgegebenen Spalte anzugeben. Dann wird die Zeichenkette dort wo es erforderlich ist zwischen den Worten unterbrochen, so dass sie in die angegebene Spaltenbreite passt. In diesem Fall hat das Format die Form nx, wobei n die Breite der Ausgabespalte und x entweder j oder r ist, wodurch angegeben wird, ob der rechte Rand buendig gemacht werden soll oder nicht. Wenn x nicht spezifiziert wird,

wird davon ausgegangen, dass der rechte Rand nicht buendig ist.

Hat man beispielsweise ein Feld string (das als long_string bezeichnet wird) der Laenge 80, das in Spalte 40 beginnend in eine 20 Zeichen breite Spalte passen soll, ist folgende print-Anweisung erforderlich:

```
print long_string using '20r' col 40
```

7.8.5 Zuweisung - set

SYNTAX
set variable to ausdr

VERWENDUNG
Dieses Kommando wird verwendet, um variable auf den Wert eines Ausdrucks zu setzen. ausdr bezieht sich auf einen beliebigen in dem aktuellen Gruppenkommando gueltigen Ausdruck. Das Kommando set ist in jedem Gruppenkommando zugelassen. Fuer jede in einem Programm verwendete Variable muss mindestens ein set-Kommando vorhanden sein.

Siehe Erklaerung der Variablen in Abschnitt 7.4.7.

7.8.6 Leerzeilen - skip

SYNTAX
skip nummer

VERWENDUNG
Dieses Kommando fuehrt dazu, dass vor Wiederaufnahme der Ausgabe nummer Zeilen uebersprungen werden. Wird nummer nicht angegeben, wird 1 angenommen. Es werden nur die Zeilen gezaehlt, die fuer den Ausdruck im aktuellen Kontext zur Verfuegung stehen. So werden beispielsweise der obere und untere Rand immer ignoriert.

7.9 Verwendung von RPT zusammen mit anderen Werkzeugen

Um die Erarbeitung von Anwendungssystemen zu vereinfachen, liefert WEGA-DATA ein eingebautes Interface zwischen RPT und SQL bzw. ENTER. Da RPT ausserdem alle ASCII-Eingabedateien lesen kann, kann es von der WEGA-Shell und nutzerspezifischen Programmen verwendet werden.

In Abschnitt 6.3.6 wurde das SQL/RPT-Interface aus der Sicht von SQL beschrieben. In Abschnitt 7.9.1 wird beschrieben, wie man RPT-Skripte schreibt, damit die Vorteile dieses Interface' am besten genutzt werden. In den Ab-

schnitten 5.1, 5.2.2 und 5.2.3 wurde darueber informiert, wie sich das ENTER/RPT-Interface aus der Sicht von ENTER darstellt. In Abschnitt 7.9.2 wird dann beschrieben, welche RPT-Skripte man fuer dieses Interface schreiben kann.

In Abschnitt 7.9.3 wird schliesslich beschrieben, wie man RPT von der Shell aus ablaufen laesst und wie es in Shell-Skripten und nutzerspezifischen Programmen verwendet werden kann, die unter Verwendung des Host-Sprach-Interface' geschrieben wurden.

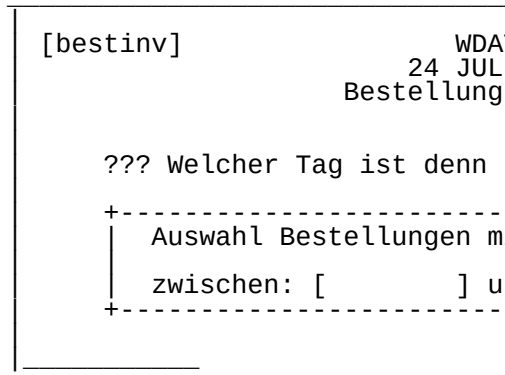
7.9.1 SQL ueber Bildmasken

Eine der wichtigsten Eigenschaften von SQL ueber Bildmasken (Abschnitt 6.3.6) besteht darin, dass es Ergebnisse einer SQL-Abfrage nehmen und unter Verwendung von RPT die Ergebnisse formatieren kann. Man kann natuerlich RPT-Skripts, wie in den vorhergehenden Abschnitten beschrieben, zur Verwendung unter SQL-Bildmasken schreiben. An diese Art der Skripten werden jedoch noch andere Anforderungen gestellt, die bekannt sein muessen.

Damit SQL-Abfrage und RPT-Skript einen Informationsaustausch vornehmen koennen, ist es erforderlich, dass die mit SQL ausgewahlten Felder und Ausdruecke zum Eingabeteil input des RPT-Skripts passen. Das heisst, dass die in der select-Klausel der SQL-Abfrage verwendeten Felder und die Felder in der input-Sektion des RPT-Skripts in derselben Reihenfolge erscheinen und vom selben Typ sein muessen. Man erreicht dies, indem man fuer einfache vom Nutzer ausgewaehlte Felder im RPT-Skript den entsprechenden Namen von datsat.feld und fuer Felder, die das Ergebnis eines Ausdrucks oder einer numerischen Funktion sind einen Namen, name, mit dahinter stehendem Typ und Laenge verwendet.

Mit einem anderen Verfahren kann man die vom Nutzer eingegebenen Auswahlparameter im Report ausgeben. Hat man beispielsweise einen Report, der einen Datenbereich und einen Statuskode vom Nutzer akzeptiert und moechte man im Report eine Titelseite haben, auf der angezeigt wird, was eingegeben wurde, so verwendet man dazu die Parameter im RPT-Skript. Parameter sind einfach Stellungsparameter, mit denen gekennzeichnet wird, wo die zugehoreigen Werte der Bildmaske eingesetzt werden. Man kann Parameter an beliebiger Stelle im RPT-Skript verwenden, vorausgesetzt, dass das Skript nach Ersetzen der Werte noch einen Sinn ergibt.

Fuer das oben angefuerte Beispiel hier eine moegliche Bildmaske. Auf ihr werden 3 Daten dargestellt.



Die Paramter sind von $1 bis zur groessten Bildmasken-Feldnummer numeriert. Es ist jedoch zu beachten, dass man in einem Skript keine bestimmten Parameter benutzen muss, man kann einige, keine oder alle Parameter verwenden. In diesem Fall ist der Parameter fuer das Tagesdatum $1, das Anfangsdatum ist $2 und das Enddatum ist $3.

Zur Ausgabe der Titelseite kann man ein Programmsegment 'before report' verwenden, das folgendermassen aussehen koennte:

```
before report'
  print 'Heute ist der: $1'
  skip
  print 'Bestellungen'
  print 'vom $2 bis $3'
```

In einem Skript kann man Parameter mehr als einmal verwenden. Wollte man also die Information auch auf jeder Seite oben ausgeben, koennte man in ein Programmsegment 'header' eine Anweisung wie folgende aufnehmen:

```
header
  print 'Datum: $1 fuer $2 bis $3'
```

Im oben stehenden Beispiel wird die Verwendung von Parametern in Zeichenkettenkonstanten demonstriert. Die Parameter koennen auch in Ausdruecken verwendet werden, z.B. in einer solchen Konstruktion:

```
if datum > = $1 - 30 then ...
```

Man koennte dies auch erreichen, indem man den Parameter einer Variablen zuordnet:

```
tdatum = $1
```

Diese Variable kann dann im restlichen Skript weiterverwendet werden.

7.9.2 ENTER und RPT

ENTER-Bildmasken haben eine Option, die gestattet, dass die Ergebnisse einer Anfrage ueber Bildmasken mit einem RPT-Skript formatiert werden koennen. Wird angegeben, dass ein bestimmter Report unter Verwendung von RPT zu formatieren ist, erzeugt ENTER fuer RPT eine Eingabedatei, in der alle auf der Bildmaske befindlichen Felder enthalten sind. Diese Eingabedatei wird an RPT uebergeben, das sie als Standard-eingabe liest. Will man RPT-Skripts schreiben, die diese Eingabedatei verwenden koennen, muss man die Reihenfolge und Typen der Felder kennen, damit man das entsprechende Programmsegment input schreiben kann. Das Kommando heisst rip bezeichnet und wird im folgenden beschrieben. Wie alle anderen zur Verfuegung stehenden WEGA-DATA-Shellkommandos verlangt rip, das Setzen der korrekten Umgebungsvariablen. Weitere Informationen siehe Abschnitt 1.1.3.

NAME
rip - Report-Input-Sektion erstellen

SYNTAX
rip bildmasken_name

BESCHREIBUNG
Dieses Kommando legt die RPT-Programmsektion input fuer ein Reportskript an, das mit einer ENTER-Bildmaske ablaufen soll. (Informationen zur input-Sektion siehe Abschnitt 7.6.3). rip gibt das korrekte Programmsegment input aus, in dem fuer eine gegebene Bildmaske alle Reportfelder in der Form datsat.feld enthalten sind. Das Programmsegment input kann direkt im Reportskript verwendet werden. Hier ein Beispiel fuer ein von rip erzeugtes Programmsegment input:

```
input
  kunde.kunden_nummer,
  kunde.name
  kunde.strasse
  kunde.ort
  kunde.postleitzahl
  kunde.ruf
  kunde.telex
```

VERWENDUNG
Dieses Kommando kann vom Editor vi zur Erzeugung eines RPT-Skripts verwendet werden. Man verwendet das Editor-kommando !, gefolgt von rip, und dem Namen der Bild-maske, mit der der Report verbunden wird. vi fuehrt das Programm aus und lenkt die sich daraus ergebende Eingabe an die aktuelle Position im Editorpuffer. Eine solche Kommandozeile in vi sieht in etwa so aus:

```
!rip bildmasken_name
```

7.9.3 Nutzerprogramme und RPT

Soll RPT von der Shell oder mit Nutzerprogrammen benutzt werden, muss man sichern, dass die ensprechenden Umgebungsvariablen gesetzt sind. Weitere Informationen siehe 1.1.3. Wird RPT von der Shell aus verwendet, sind zwei Dateien erforderlich - eine eingabe_datei, in der die zu formatierenden Daten enthalten sind und ein skript, in dem die fuer die Formatierung erforderlichen Reportgenerator-Kommandos enthalten sind. Das Skript wird immer vom Nutzer angelegt, wobei beruecksichtigt wird, wie der fertige Report aussehen soll. Die Eingabedatei kann aus beliebig vielen Quellen stammen und kann entweder als Dateiname gegeben sein oder ueber ein Pipe mit RPT verbunden sein. Die Shell-Syntax fuer den Ablauf von RPT ist folgende:

```
RPT [-r] skript [eingabe_datei|-]
```

Die Angabe von skript ist Pflicht. Das Flag -r ist optionell und weist RPT an, nach Fertigstellung des Reports die Skriptdatei zu entfernen. Dieses Flag soll insbesondere bei SQL ueber Bildmasken verwendet werden, das fuer das Skript eine temporaere Datei anlegt und benutzt. Die eingabe_datei ist optionell. Sie ist der Name einer ASCII-Datei, die die zu formatierenden Daten enthaelt. Soll die Eingabedatei ueber Pipe mit RPT verbunden werden ist ein - (Bindestrich) anstelle des Namens der Eingabedatei zu verwenden.

Wenn keine Eingabedatei spezifiziert wird (namentlich oder mit Bindestrich), kompiliert RPT einfach das Skript und gibt einen Report aus, in dem die wichtigsten internen Tabellengroessen aufgefuehrt sind und wieviele Elemente jeder Tabelle vom Skript verwendet werden. Das ist dann nuetzlich, wenn man ein komplexes Reportskript hat, von dem angenommen werden muss, dass einer oder mehrere der Maximalwerte ueberschritten wurden. In diesem Report sind die Begriffe Kommando und Anweisung untereinander austauschbar. Hier ein Beispielreport fuer den Kompilierungsprozess, gefolgt von einer Erlaeuterung der Eintraege:

RPT - Report Processor

No syntax errors were found in the report script.
RPT TABLE USAGE INFORMATION

Table Name	Used	Maximum
Expression Nodes	"	400
Variables	"	150
Constants	"	250
Commands	"	256
Print Statements	"	125
Print Items	"	256
Sort Items	"	15
Input Items	"	100
Command Groups	"	25
Set Statements	"	100
If Statements	"	50
Aggregates	"	50
Function Calls	"	50
Arguments	"	100

Expression Nodes
Jeder verwendete Ausdruck nimmt einen der 400 vorhandenen expression nodes ein. Ausdruecke werden in Abschnitt 7.4 beschrieben.

Variables
In einem einzigen Reportskript koennen hoechstens 150 verschiedene Variable verwendet werden. Variable werden in Abschnitt 7.4.7 beschrieben.

Constants
In einem Reportskript koennen hoechstens 250 Konstanten verwendet werden. Dazu gehoeren Konstanten vom Typ string, numeric, date, time und amount. Die Konstanten werden in den Abschnitten 7.4.4 und 7.4.5 beschrieben.

Commands
In einem Reportskript koennen hoechstens 256 verschiedene Kommandos verwendet werden. Fuer jede geschriebene Anweisung sort, after, detail, print, if usw. wird ein Kommando verbraucht.

Print Statements
In einem Reportskript kann eine print-Anweisung (Kommando) hoechstens 125 mal verwendet werden.

Print Items
Jeder in einer print-Anweisung stehende durch Komma getrennte Ausdruck verbraucht ein print-Datenelement. Es koennen hoechstens 256 verschiedene Datenelemente ausgegeben werden.

Sort Items
Jeder in einer sort-Anweisung stehende durch Komma getrennte Ausdruck verbraucht ein sort-Datenelement. Man kann nach hoechstens 15 verschiedenen Datenelementen (Ausdruecken) sortieren.

Input Items
In einer Eingabesektion input koennen hoechstens 100 verschiedene Felder enthalten sein.

Command Groups
Man kann hoechstens 25 verschiedene Gruppenkommandos verwenden. Gruppenkommandos sind in Abschnitt 7.7 beschrieben.

Set Statements
Hoechstens 100 set-Anweisungen sind moeglich. Diese werden benutzt, um den Variablen Werte zuzuordnen.

If Statements
Hoechstens 50 if-Anweisungen sind moeglich. Beschreibung dieses Kommandos siehe Abschnitt 7.8.1

Aggregates
Hoechstens 50 numerische Funktionen sind moeglich (z.B. min, max, avg, count und total). Diese werden in Abschnitt 7.4.6.1 beschrieben.

Function Calls
Ein Funktiosaufruf erfolgt, wenn eine der lokalen Funktionen benutzt wird, entweder eine eingebaute oder eine eigene. Es koennen hoechstens 50 Aufrufe lokaler Funktionen erfolgen. Lokale Funktionen werden in Abschnitt 7.4.6.2 beschrieben.

Arguments
Argumente (auch Parameter genannt) koennen an lokale Funktionen uebergeben werden, damit man dieselben Operationen an verschiedenen Daten vornehmen kann. Fuer die im Reportskript verwendeten Aufrufe lokaler Funktionen sind hoechstens 100 Argumente moeglich. Lokale Funktionen werden in Abschnitt 7.4.6.2 beschrieben.

Hier als Beispiel einige Kommandozeilen, die benutzt werden koennten, um RPT von der Shell aus, von einem Shell-Skript oder von einem Nutzerprogramm aus aufzurufen. Immer erfolgt die Ausgabe des Reports in der Standardausgabe, dagegen werden Syntaxfehler und Tabellennutzungsreport an die Standardfehlerausgabe gegeben.

```
RPT rskript
```

Diese Kommandozeile kompiliert das in rskript enthaltene Skript und gibt den Tabellennutzungsreport aus.

```
RPT rskript rinput
```

Diese Kommandozeile formatiert die Daten in der Datei rinput entsprechend den in in der Datei rskript enthaltenen Kommandos.

```
SQL sskript | RPT rskript -
```

Diese Kommandozeile verbindet die in sskript enthaltenen Ergebnisse der SQL-Anfrage ueber eine Pipe mit RPT und formatiert sie entsprechend den in rskript enthaltenen Kommandos. Soll SQL auf diese Weise genutzt werden, muss man unter Verwendung des lines-Kommandos (Abschnitt 6.4.3.3) die Spaltenueberschriften in der SQL-Ausgabe ausschalten. Man kann natuerlich auf diese Weise auch jedes nutzerspezifische Programm per Pipe mit RPT verbinden.

7.10 Zusammenfassung

In den vorhergehenden Beispielen wurde gezeigt, wie RPT zu verwenden ist. In diesem letzten Abschnitt soll eine zusammenfassende Information fuer Nutzer gegeben werden, die bereits mit RPT vertraut sind, die jedoch schnell einige Informationen benoetigen. Die von RPT verwendeten reservierten Schluesselworte werden in Abschnitt 7.10.1 aufgefuehrt. In Abschnitt 7.10.2 wird eine Zusammenfassung der Syntax von Kommandos und Gruppenkommandos gegeben. In Abschnitt 7.10.3 werden die von RPT erzeugten Fehlermeldungen mit den entsprechenden Korrekturmassnahmen angefuehrt.

7.10.1 Schluesselworte von RPT

Die in der folgenden Liste angefuehrten Worte haben fuer RPT eine spezielle Bedeutung und koennen daher nicht als Datensatz- oder Feldnamen verwendet werden. Werden diese Worte von Reportskripten als Feldnamen verwendet, entstehen Syntaxfehler.

```
after          header         print
amount         hour           report
and            if             separator
avg            in             set
before         input          skip
begin          left           sort
bottom         length         string
centered       margin         substr
col            max            sum
column         min            then
count          need           time
date           newline        to
desc           no             today
detail         not            top
else           numeric        using
end            or             where
float          page           width
footer         pageno
```

7.10.2 Ueberblick ueber Kommandos und Gruppenkommandos

In diesem Abschnitt wird ein Ueberblick ueber RPT-Kommandos und -Gruppenkommandos gegeben. Ihre Syntax wird angegeben und es werden einige Richtlinien bezueglich ihrer Verwendung gegeben. Ist man mit den Kommandos, Gruppenkommandos und Gruppenwechselverarbeitung vertraut, ermoeglicht dieser Ueberblick eine schnelle Bezugnahme.

Im folgenden werden die Kommandos, die keine Gruppenkommando-Kommandos sind (length bis sort) und die Gruppenkommandos angefuehrt. In einer weiteren Tabelle werden die Gruppenkommando-Kommandos angefuehrt.

```
input | name [ | numeric length | ] |
      |        | string length  |   |
      |        | amount length  |   |
      |        | float length   |   |
      |        | date           |   |
      |        | time           |   |
      | datsat.feld                 |

  [,  | name [ | numeric length | ] | ...]
      |        | string length  |   |
      |        | amount length  |   |
      |        | float length   |   |
      |        | date           |   |
      |        | time           |   |
      | datsat.feld                 |
```

```
length nummer

width nummer

top margin nummer
```

```
  bottom margin nummer

  left margin nummer

  sort ausdr [desc]
       [,ausdr [desc] ...]

  header
    gruppenkommando_kommando

  footer
    gruppenkommando_kommando

  before report
    gruppenkommando_kommando

  before name
    gruppenkommando_kommando

  detail
    gruppenkommando_kommando

  after name
    gruppenkommando_kommando

  after report
    gruppenkommando_kommando

  end
```

Die Sektion input muss am Anfang und die Sektion end zuletzt stehen. Ansonsten ist die Reihenfolge der Kommandos und Gruppenkommandos beliebig und die hier angefuehrte Reihenfolge ist nur ein Vorschlag. Es ist moeglich, dass die Kommandos length, width und margin ueberhaupt nicht verwendet werden brauchen. Jedem der Kommandos ist ein Standardwert zugeordnet. Ein optionelles sort-Kommando kann verwendet werden.

In einem Programm muss mindestens ein Gruppenkommando erscheinen. Ein Gruppenkommando wird durch das naechste Gruppenkommando oder durch ein Kommando, das kein Gruppenkommando-Kommando ist, (z.B. width) abgeschlossen. Deshalb entsteht ein Fehler, wenn unmittelbar nach einem Kommando, das kein Gruppenkommando-Kommando ist (z.B. sort) ein Kommando wie print (siehe unten) steht, da das Kommando print nicht ausserhalb eines Gruppenkommandos verwendet werden kann.

Die Gruppenkommandos 'before name' und 'after name' koennen mehr als einmal auftreten. name bezieht sich auf einen in einem sort-Kommando genannten Ausdruck. 'before name'- und 'after name'-Gruppenkommandos verwenden oft denselben Ausdruck name. Zwei after-Gruppenkommandos koennen jedoch nicht denselben Namen name verwenden, ebenso wie zwei

before-Gruppenkommandos nicht denselben Namen name verwenden koennen.

In der naechsten Tabelle werden die Gruppenkommando-Kommandos beschrieben, die innerhalb der fuer die Beschreibung einer Reportausgabe verwendeten Gruppenkommandos verwendet werden.

```
need nummer

print [ ausdr [[in] col[umn] nummer] [using format]

      [,ausdr [[in] col[umn] nummer] [using format] ...]

                      [| no newline |] ]
                       |  centered  |

skip [nummer]

page [, no footer] [, no header]

set variable to ausdr

if ausdr then
          kommando
       [else
          kommando]
```

wobei kommando entweder ein einzelnes Gruppenkommando-Kommando ist oder eine Folge von Gruppenkommando-Kommandos, die von den Schluesselworten begin und end begrenzt sind.

```
begin
  gruppenkommando_kommando_1
             .
             .
             .
  gruppenkommando_kommando_n
end
```

Die Kommandos page und need sind in den Gruppenkommandos header und footer nicht erlaubt.

Wenn die oben aufgefuehrten Kommandos in Gruppenkommandos verwendet werden (mit Ausnahme des Gruppenkommandos 'after report') kann ein Ausdruck bei seiner ersten Verwendung einen Namen erhalten und dann spaeter einfach unter Angabe dieses Namens verwendet werden. Daher impliziert ausdr oft folgendes (siehe Abschnitt 7.4.8):

```
|     ausdr      |
| [<name>] ausdr |
```

7.10.3 Fehlermitteilungen von RPT

In diesem Abschnitt sollen die Fehlermeldungen zusammengefasst werden, die bei der Verwendung von RPT auftreten koennen. Diese werden in alphabetischer Reihenfolge aufgefuehrt und es erfolgt eine kurze Erklaerung der Fehlerursache und (falls moeglich) wird ein Beispiel fuer ein Reportskript oder eine Anweisung, die eine solche Meldung verursachen koennten angefuerht.

Es gibt zwei Fehlerklassen - Fehler im Muster oder der Struktur der in der Anweisung verwendeten Reihenfolge der Schluesselworte (Syntaxfehler) und Fehler in der Bedeutung der Anweisung (semantische Fehler). Bei Auftreten von Syntaxfehlern kann RPT nicht verstehen, was gefordert ist. Semantische Fehler ergeben sich aus einer grammatikalisch gueltigen Anweisung, die RPT jedoch nicht sinnvoll verarbeiten kann. Ein typischer Syntaxfehler waere es, wenn ein Schluesselwort falsch geschrieben oder ausgelassen wuerde. Ein typischer semantischer Fehler laege dann vor, wenn man einer Zeichenkette eine Zahl hinzuzufuegen versuchte. Beide Fehlerarten fuehren dazu, dass kein sinnvolles Ergebnisses entstehen kann.

xxxx is an unrecognized function name.

Erklaerung:
Der Name der Funktion existiert entweder nicht oder ist nicht lesbar. Moeglicherweise wurde die Funktion falsch geschrieben

A string constant was expected.

Erklaerung:
Die Syntax des Kommandos erfordert eine Konstante. Es ist zu beachten, dass Konstanten in einfache Anfuehrungszeichen einzuschliessen sind.

Ungueltiges Skript:
```
separator :
```

Korrektur:
```
separator ':'
```

An integer constant was expected.

Erklaerung:
Die Syntax des Kommandos erfordert eine ganze Zahl.

Ungueltiges Skript:
```
bottom margin 7e
```

Korrektur:
```
bottom margin 7
```

Either 'desc' or a named expression was expected

Erklaerung:
Fuer ein sort-Kommando oder die Ausdruecke eines solchen Kommandos wurde eine fehlerhafte Syntax verwendet. Moeglicherweise steht am Ende der sort-Kommandozeile ein Komma zu viel.

Ungueltiges Skript:
```
sort postleitzahl, <a1+a2> b_num - t_num
```

Korrektur:
```
sort postleitzahl, <neu_num> b_num - t_num
```

Fatal error

Erklaerung:
RPT hat einen nichtbehebbaren internen Fehler gefunden. Die vor dieser Fehlermeldung stehende Fehlermeldung verweist auf den Charakter des Fehlers. Das Reportskript ist unter Verwendung einer anderen Methode neu zu schreiben.

Illegal use of variable: xxxx

Erklaerung:
In diesem Kontext darf keine Variable verwendet werden. Der Fehler kann einfach in der falschen Schreibweise einer Funktion, eines Schluesselwortes, einer Variablen oder eines Feldnamens bestehen.

Ungueltiges Skript:
```
sort ort, x
        .
        .
        .
before report
  set x to 0
```
Korrektur:
```
sort ort, x_kumm
        .
        .
        .
before report
  set x to 0
```

Input field rrrr.ffff does not have its type specified, and it is not a WDATA field.

Erklaerung:
RPT kann in der Datenbank kein zum Datensatz rrrr gehoeriges Feld ffff finden. Handelt es sich bei diesem Eingabefeld um einen beliebigen Namen, wurden moegli-

cherweise fuer dieses Feld TYPE und LENGTH nicht spezifiziert. Achtung: Diese Meldung wird auch generiert, wenn das Reportfeld einem Datenbank-Feld vom Typ COMB entspricht. Zweitens ist, wenn die Syntax datsat.feldname verwendet wird, zu pruefen, ob der Name des Eingabefeldes (ffff) der ausfuehrliche Name fuer das Datenbankfeld ist.

Ungueltiges Skript:

```
input
  kunr, kuname
detail
  print kunr, kuname
end
```

Korrektur:

```
input                                  input
  kunde.kunr,                            kunr [numeric 7],
  kunde.kuname          oder             kuname [string 30]
detail                                 detail
  print kunr, kuname                     print kunr, kuname
end                                    end
```

Insufficient memory for arglist

Erklaerung:
Bei der Beschreibung eines Nutzer-Funktionsaufrufs in einem Skript wurden zu viele Argumente fuer den Aufruf der Funktion verwendet.

Korrektur:
Es sind weniger Elementen zu verwenden.

Internal error in ckset nnnn llll

Erklaerung:
Es wurde versucht, eine Variable auf einen unvertraeglichen Typ zu setzen.

Ungueltiges Skript:

```
input
  a_zeit[time],
  wert [numeric 4]
before report
  set x to DATA
        .
        .
        .
detail
  print x, a_zeit, wert
```

Korrektur:

```
input
  a_zeit[time],
```

```
      wert [numeric 4]
    before report
      set x to 'DATA'
            .
            .
            .
    detail
      print x, a_zeit, wert
```

Invalid column nuber nnnn generated by print statement on line dddd.

Erklaerung:
Diese Meldung wird nur waehrend der Laufzeit generiert. Die Spaltennummer nnnn ist groesser als die vorgegebene Breite der Seite. Dieser Fehler tritt auf Zeile dddd des Reportskriptes auf. Es ist zu beachten, dass nnnn um Eins kleiner ist als die tatsaechlich auf dem Skript gezeigte Nummer.

Ungueltiges Skript:

```
        .
        .
        .
    width 80
        .
        .
        .
    print 'Neuer Wert' in column 89
```

Korrektur:

```
        .
        .
        .
    width 80
        .
        .
        .
    print 'Neuer Wert' in column 65
```

Invalid date.

Erklaerung:
Das Format einer Datumskonstanten, DATE, ist nicht korrekt. DATE Konstanten haben die Form MM/DD/YY, wobei MM der Monat (1-12), DD der Tag (1-31, je nach Monat) und YY das Jahr ist.

Ungueltiges Skript:

```
    set beginn_datum to 12/35/86
```

Korrektur:

```
    set begin_datum to 12/31/86
```

Invalid time.

Erklaerung:
Das Format einer Zeitkonstanten TIME ist nicht korrekt. TIME Konstanten haben die Form HH:MM, wobei HH die Stunde (von 0 bis 23) und MM die Minute (von 0 bis 59) ist.

Ungueltiges Skript:
```
set report_zeit to 12:344
```

Korrektur:
```
set report_zeit to 12:34
```

Left type is LLLL, right type is RRRR, operator is xxxx.

Erklaerung:
Die Typen, TYPE, zweier Operanden sind unvereinbar und koennen nicht konvertiert werden. Moeglicherweise wurde der falsche Feld- oder Variablenname verwendet.

Ungueltiges Skript:
```
input
  kunr [numeric 3],
  name [string 30],
  ort [string 5],
     .
     .
     .
  if ort = Floh then
       .
       .
       .
```

Korrektur:
```
input
  kunr [numeric 3],
  name [string 30],
  ort [string 5],
     .
     .
     .
  if ort = 'Floh' then
       .
       .
       .
```

on or about line nnnn

Erklaerung:
Diese Meldung wird bei der Entdeckung eines Syntaxfehlers an die Standardfehlerausgabe gesendet. Damit wird angegeben, wo der vor dieser Meldung angezeigte Syntaxfehler in etwa aufgetreten ist.

Output line overflow.

Erklaerung:
Diese Meldung wird waehrend der Laufzeit generiert. Sie wird erzeugt, wenn die Ausgabe einer print-Zeile laenger ist als die Breite der Seite.

Ungueltiges Skript:
```
input
  kunr [numeric 8],
  name [string 30],
  ort [string 20]
width 80
detail
  print 'Liste der Kunden' centered
  skip 2
  print kunr, name, ort in column 33
```

Korrektur:
```
input
  kunr [numeric 8],
  name [string 30],
  ort [string 20]
width 80
detail
  print 'Liste der Kunden' centered
  skip 2
  print kunr,ort in column 35
  print name
```

The xxxx table has overflowed.

Erklaerung:
Die Tabelle xxxx enthaelt mehr Elemente als maximal zulaessig.

Korrektur:
Innerhalb des Skripts ist die Anzahl der Elemente dieses Typs zu verringern. Die verschiedenen Tabellen und ihre Maximalwerte werden im Tabellennutzungsreport von RPT aufgelistet.

The String table has overflowed.

Erklaerung:
Die Kapazitaet der Tabelle der Zeichenkettenkonstanten wurde ueberschritten.

Korrektur:
Die Anzahl oder Groesse der Zeichenkettenkonstanten im Skript ist zu verringern.

The create on intermediate file xxxx failed.

Erklaerung:
Die RPT-Hilsdatei namens xxxx kann nicht angelegt werden. Moeglicherweise sieht das Verzeichnis kein Schreibrecht fuer die Gruppe vor, der der Nutzer angehoert oder im Verzeichnis ist kein Platz mehr frei.

The create on tag file xxxx failed.

Erklaerung:
Die RPT-tag-Datei namens xxxx kann nicht angelegt werden. Moeglicherweise sieht das Verzeichnis kein Schreibrecht fuer die Gruppe vor, der der Nutzer angehoert oder im Verzeichnis ist kein Platz mehr frei.

The if clause on line nnnn does not contain an integer expression.

Erklaerung:
RPT konnte den im if-Kommando auf Zeile nnnn verwendeten Ausdruck nicht verstehen. Ganze Zahlen sind mit den Reportfeldern NUMERIC oder AMOUNT oder den Variablen in if-Anweisungen zu vergleichen.

Ungueltiges Skript:

```
input
  kunr [numeric 3],
  name [string 30],
  ort [string 5],
  postleitzahl [numeric 4]
     .
     .
     .
  if postleitzahl = '1309' then
       .
       .
       .
```

Korrektur:

```
input
  kunr [numeric 3],
  name [string 30],
  ort [string 5],
  postleitzahl [numeric 4]
     .
     .
     .
  if postleitzahl = 1309 then
       .
       .
       .
```

The open on intermediate file xxxx failed.

Erklaerung:
RPT kann die vorher angelegte Datei xxxx nicht oeffnen. Es ist zu pruefen, ob diese Datei noch immer existiert, moeglicherweise wurde sie von einem anderen Nutzer geloescht.

The open on tag file xxxx failed.

Erklaerung:
RPT kann die vorher angelegte tag-Datei nicht oeffnen. Es ist zu pruefen, ob diese Datei noch immer existiert, moeglicherweise wurde sie von einem anderen Nutzer geloescht.

The referred to sort expression is xxxx.

Erklaerung:
Der in der vorhergehenden Fehlermeldung angefuehrte sort Ausdruck wird als xxxx aufgelistet. Diese Meldung soll helfen, den in der vorhergehenden Meldung angezeigten Fehler zu finden.

The value of variable: xxxx is never set.

Erklaerung:
Der Wert von xxxx wurde nicht mit einem set-Kommando initialisiert oder xxxx wurde nicht als Reportfeld in das Eingabesegment input aufgenommen. xxxx kann auch falsch oder ohne Anfuehrungsstriche geschrieben worden sein.

Ungueltiges Skript:

```
input
  name [string 30],
  nummer [numeric 5]
before report
  print Kunden
detail
  print name, nummer
end
```

Korrektur:

```
input
  name [string 30],
  nummer [numeric 5]
before report
  print 'Kunden'
detail
  print name, nummer
end
```

The variable type is VVVV and the expression type is XXXX.

Erklaerung:
Man kann einer vorher gesetzten Variablen keinen neuen Wert zuordnen, wenn die Typen, TYPE, der Werte nicht kompatibel sind.

Ungueltiges Skript:
```
set z_variable to 4.89
      .
      .
      .
set z_variable to 'abc' + 'def'
```

Korrektur:
```
set z_variable to 4.89
      .
      .
      .
set neu_variable to 'abc' + 'def'
```

The write to FNDFLD failed errno - nnnn

Erklaerung:
RPT konnte nicht in die Datei FNDFLD schreiben. Diese Fehlermeldung tritt normalerweise zusammen mit anderen Fehlermeldungen auf.

Korrektur:
Zuerst sind alle anderen angezeigten Fehler zu korrigieren. Danach wird ueberprueft, ob die Umgebungsvariable WDATA korrekt gesetzt wurde und ob das entsprechende Schreibrecht fuer das Verzeichnis vorliegt.

There are no input records

Erklaerung:
Die Eingabedatei enthaelt keine Daten. Wird fuer die Auswahl der Daten SQL verwendet, laesst man die Anfrage allein ablaufen, um zu sichern, dass sie Ergebnisse erzeugt.

There are too many saved strings.

Erklaerung:
Es handelt sich um einen internen Fehler, der entsteht, weil zu viele Zeichenketten mit der using-Option von print formatiert werden.

Korrektur:
Es sind weniger using Formatierungen in den print-Anweisungen des Reportes zu verwenden.

There is a command group that refers to a non-existent sort expression on line nnnn.

Erklaerung:
Das Reportskript bezieht sich in einem before- oder after-Gruppenkommando auf einen Namen, der nicht als sort-Element aufgelistet ist.

Ungueltiges Skript:

```
       .
       .
       .
    sort kunde_nummer
    before postleitzahl
      print 'Bezirk'
       .
       .
       .
```

Korrektur:

```
       .
       .
       .
    sort kunde_nummer, postleitzahl
    before postleitzahl
      print 'Bezirk'
       .
       .
       .
```

There is a nested aggregate Operator on line nnnn.

Erklaerung:
Numerische Funktionen duerfen nicht geschachtelt werden.

Ungueltiges Skript:

```
    print total (neu_kumm + total (alt_kumm))
```

Korrektur:

```
    set total_alt to total (alt_kumm)
    print total (neu_kumm + total_alt)
```

There is a saved string overflow.

Erklaerung:
Es handelt sich um einen internen Fehler, der entsteht, weil durch die using-Option von print eine zu lange Zeichenkette formatiert wird.

Korrektur:
Die Laenge der Eingabezeichenkette ist zu verringern.

There is a vaiable used in a sort expression on line nnnn.

Erklaerung:
Variable koennen nicht als Elemente in sort verwendet werden. Die Kommandozeile ist auch auf eine moeglicherweise vorhandene falsche Schreibweise zu ueberpruefen.

Ungueltiges Skript:
```
sort ort, x
    .
    .
    .
before report
  set x to 0
```

There is an aggregate expression embedded in a sort expression on line nnnn.

Erklaerung:
Numerische Funktionen koennen nicht in einem sort-Ausdruck verwendet werden.

Ungueltiges Skript:
```
sort ort, total (x_kumm)
```

There is an assignment to a non-variable on line nnnn.

Erklaerung:
In ein und demselben Reportskript darf eine Alpha-Zeichenkette (Name) nicht gleichzeitig als Variable und als Name eines Reportfeldes verwendet werden.

Ungueltiges Skript:
```
input
  x [date],
  name [string 30]
before report
  set x to 4
      .
      .
      .
```

Korrektur:
```
input
  k_datum [date],
  name [string 30]
before report
  set x to 4
      .
      .
      .
```

There is an error in an 'if' expression.
Erklaerung:
RPT hat im if-Kommando einen fehlerhaften Ausdruck gefunden. Richtige Syntax siehe Abschnitt 7.4.

There is an integer constant expected for the length.

Erklaerung:
Wird einem Reportfeld im Eingabesegment input TYPE und LENGTH zugeordnet, muss fuer LENGTH eine ganze Zahl verwendet werden. Siehe Abschnitt 7.2.1 und 7.63.

Ungueltiges Script:

```
input
  kunde_nr [numeric xx],
           .
           .
           .
```

Korrektur:

```
input
  kunde_nr [numeric 12],
           .
           .
           .
```

There is an invalid aggregate expression

Erklaerung:
Syntax der numerischen Funktion ist nicht korrekt. Siehe Abschnitt 7.4.6.1

Ungueltiges Skript:

```
print total (total)

print total (preis wher preis > 7.00)
```

Korrektur:

```
print total (richtig_feld)

print total (preis where preis > 7.00)
```

There is an invalid command

Erklaerung:
RPT erkennt das auf Zeile nnnn stehende Wort nicht als Gruppenkommando-Schluesselwort. (In der naechsten Fehlermeldung wird die Nummer der Zeile angegeben, in der der Fehler auftrat.) Haeufig wird dieser Fehler dadurch verursacht, dass das erste Wort der auf ein Gruppenkommando folgenden Zeile, die keine Kommandozeile ist, falsch geschrieben wurde.

Ungueltiges Skript:

```
          .
          .
          .
    before report
      ssset x to 0
```

Korrectur:

```
          .
          .
          .
    before report
      set x to 0
```

There is an invalid column clause

Erklaerung:
Nach der Klausel in column muss eine ganzzahlige Konstante oder Variable stehen. Diese Fehlermeldung wird i.a. generiert, wenn in dem Feld mit dem Spaltenwert ein einzelnes Sonderzeichen steht oder wenn der Spaltenwert nicht angegeben wurde.

Ungueltiges Skript:

```
    print ort in column
```

Korrektur:

```
    set ort_column to 20
          .
          .
          .
     print ort in column ort_column
```

There is an invalid function argument

Erklaerung:
Syntax des einer Funktion folgenden Argumentes ist nicht korrekt. Moeglicherweise fehlen eine runde Klammer, einfache Anfuehrungszeichen oder Komma.

There is an invalid input item

Erklaerung:
RPT erkennt einen der im Kommandosegment input angefuehrten Reportfeld-Namen nicht. RPT-Schluesselworte oder -funktionen duerfen nicht als Reportfeldnamen in der input-Sektion verwendet werden.

Dieser Fehler kann auch entstehen, wenn hinter dem letzten in input stehenden Datenelement ein Komma stehen bleibt. Dadurch wird die naechste Zeile als Teil der Eingabesektion gelesen.

```
Ungueltiges Skript:
    input
      best_nr [numeric 9],
              .
              .
              .
      total [amount 8]

Korrektur:
    input
      best_nr [numeric 9],
              .
              .
              .
      best_total [amount 8]
```

There is an invalid non-command group

Erklaerung:
Es ist zu ueberpruefen, ob die Syntax der Gruppe, die kein Gruppenkommando ist, richtig ist. Der Fehler kann durch falsche Schreibweise eines Kommandowortes, durch Weglassen eines Kommas, Weglassen der einfachen Anfuehrungszeichen bei einer aus mehreren Worten bestehenden Ueberschrift, Weglassen des Kommandos end zur Beendigung des Reportskripts entstehen oder dadurch dass das Kommando set nicht innerhalb eines Gruppenkommandos steht.

```
Ungueltiges Skript:
    input
      name [string 30],
      number [numeric 5],
    beffore record
           .
           .
           .

Korrektur:
    input
      name [string 30],
      number [numeric 5],
    before record
           .
           .
           .
```

There is an invalid print item

Erklaerung:
Syntax des print-Ausdrucks ist nicht korrekt. Es ist zu beachten, dass in print-Ausdruecken keine Schluesselworte zu verwenden sind.

Ungueltiges Skript:
```
print report
  (Report ist ein Schluesselwort)
```

There is an invalid type expression.

Erklaerung:
RPT erkennt nicht den TYPE eines in input aufgelisteten Reportfeldes. Moeglicherweise wurde das Wort numeric, string, date, time oder amount falsch geschrieben. COMB ist kein gueltiger TYPE.

Ungueltiges Skript:
```
          .
          .
          .
model_nr [Numeric 20],
```

Korrektur:
```
          .
          .
          .
model_nr [numeric 20],
```

There is insufficient memory for the input fields.

Erklaerung:
Bei Einlesen des Reportes waehrend der Laufzeit ist fuer die Verarbeitung des Reports nicht genuegend Platz.

Korrektur:
Der Speicher ist zu entlasten, indem Ausgabezeichenketten, Reportfeldgroessen, Anzahl von Variablen bzw. die Komplexitaet des Reportskripts im allgemeinen reduziert wird.

There is insufficient memory for the output line buffer.

Erklaerung:
Die Ausgabezeile ueberschreitet den zur Verfuegung stehenden Platz im Speicherpuffer.

Korrektur:
Die Anzahl der Ausgabezeilen oder die Ausgabekomplexitaet sind zu reduzieren.

There is insufficient memory to hold variables

Erklaerung:
Es steht insgesamt nicht genuegend interner Speicherraum fuer die Speicherung aller Variablenwerte zur

Verfuegung.

Korrektur:
Der insgesamt benoetigte Speicherraum ist zu verringern durch Verkleinerung der Ausgabezeichenketten, der Groesse der Reportfelder, der Anzahl von Variablen oder durch Verringerung der allgemeinen Komplexitaet des Reportskripts.

There is insufficient memory to store the constants.

Erklaerung:
Es ist insgesamt nicht genuegend interner Speicherraum vorhanden fuer die Speicherung aller zu den print-Anweisungen gehoerigen Daten.

Korrektur:
Der insgesamt benoetigte Speicherraum ist zu verringern, indem Ausgabezeichenketten, die Groesse der Reportfelder, die Anzahl der variablen bzw. die allgemeine Komplexitaet des Reportskripts verringert wird.

There is a type mismatch in the set clause on line nnnn.

Erklaerung:
Man kann einer vorher gesetzten Variablen keinen neuen Wert zuordnen, wenn die TYPEs der Werte nicht kompatibel sind.

Ungueltiges Skript:

```
          .
          .
          .
set z to 1.34
if kunde_nr = 3 then
  set z to 'abc' + 'efg'
```

Korrektur:

```
          .
          .
          .
set z to 1.34
if kunde_nr = 3 then
  set string to 'abc' + 'efg'
```

This was detected in a set expression on line nnnn.

Erklaerung:
Diese Meldung soll beim Auffinden des Fehlers helfen, durch den die vorhergehende Fehlermeldung generiert wurde.

This was detected in the column expression for the print statement on line nnnn.

Erklaerung:
Diese Meldung soll beim Auffinden des Fehlers helfen, durch den die vorhergehende Fehlermeldung generiert wurde.

This was detected in the expression on line nnnn.

Erklaerung:
Diese Meldung soll beim Auffinden des Fehlers helfen, durch den die vorhergehende Fehlermeldung generiert wurde.

This was detected in the if expression on line nnnn.

Erklaerung:
Diese Meldung soll beim Auffinden des Fehlers helfen, durch den die vorhergehende Fehlermeldung generiert wurde.

statement on line nnnn.

Erklaerung:
Diese Meldung soll beim Auffinden des Fehlers helfen, durch den die vorhergehende Fehlermeldung generiert wurde.

This was detected on line nnnn.

Erklaerung:
Diese Meldung soll beim Auffinden des Fehlers helfen, durch den die vorhergehende Fehlermeldung generiert wurde.

Type conversion error.

Erklaerung:
Die TYPEs der Operanden koennen nicht konvertiert werden, um sie zueinander passend zu machen. Dieser Fehlermeldung folgen weitere Fehlerinformationen.

syntax error

Erklaerung:
Bei der Erarbeitung eines Kommandos wurde ein fehlerhaftes Format verwendet. Dieser Fehlermeldung folgen weitere Meldungen.

Unable to execute FNDFLD - nnnn.

Erklaerung:
Es ist zu pruefen, ob die PATH- und WDATA-Umgebungsvariablen richtig gesetzt und exportiert wurden (siehe Abschnitt 1.1.3)

8. DER LISTENPROZESSOR LST

Der WEGA-DATA Listenprozessor LST ist eine Auswahl- und Formatierungssprache fuer die Erzeugung sortierter Listen, in denen Summen und Zwischensummen aus der WEGA-DATA-Datenbank enthalten sind. LST soll eine einfache Alternative zur Verwendung von SQL und RPT sein und zwar fuer den Fall, dass die volle Leistungsfaehigkeit dieser Werkzeuge nicht benoetigt wird. LST hat zwei Hauptkomponenten - den Auswahlprozessor, mit dem man auf der Grundlage von Auswahlkriterien Datensaezte aus einer Datei auswaehlen kann, und den Listenprozessor, mit dem man die ausgewaehlten Datensaetze sortieren, formatieren und totalisieren kann.

Der Auswahlprozessor ist eine leicht zu verwendende vergleichende Anfragesprache, mit der man Anfragen an die WEGA-DATA-Datenbank richten kann. Man kann in einer einzigen Anfrage mehrere Datensatztypen miteinander kombinieren. Die Syntax der Anfrage ist unabhaengig von den verschiedenen fuer eine WEGA-DATA-Datenbank zulaessigen Zugriffsmethoden. Das heisst, dass man Anfragen auch dann unveraendert beibehalten kann, wenn die physische oder logische Beschreibung der Datenbank modifiziert wird.

Durch eine Anfrage des Auswahlprozessors wird eine Auswahldatei erzeugt, in der die Adressen aller Datensaezte enthalten sind, die die Auswahlkriterien erfuellten. Eine Auswahldatei kann auch als Teilmenge bezeichnet werden. Wie bei ENTER koennen die vom Auswahlprozessor ausgewaehlten Datensaetze vom Listenprozessor verwendet werden, wodurch eine nutzerspezifische Ausgabe ermoeglicht wird. Die Auswahldatei kann aber auch von einem Nutzerprogramm verarbeitet werden, wodurch maximale Flexibilitaet gewaehrleistet wird.

Der Listenprozessor ist ein allgemein verwendbares Werkzeug fuer die Erzeugung von nutzerspezifisch formatierten, aus Auswahldateien entnommenen Listen. Er ist leicht zu handhaben und dennoch leistungsfaehig. Er liefert Standardwerte fuer fast alle Optionen.

In seiner einfachsten Einsatzform werden die aufzulistenden Felder spezifiziert und das Programm ordnet selbstaendig Spalten, Ueberschriften und Format zu. Es ist jedoch auch moeglich, dass der Nutzer selbst spezifiziert, in welcher Spalte und Zeile jedes Feld oder jeder Ausdruck ausgegeben und welches Format verwendet werden soll. Summen, Zwischensummen und Sortierordnung koennen definiert werden. Ausser-

dem koennen mehrzeilige Spalten- und Listenueberschriften spezifiziert werden.

8.1 Die Auswahl von Datensaetzen

Der erste Schritt beim Anlegen eines LST-Report besteht darin, dass die gewuenschten Datensaetze ausgewaehlt werden. Auswahldateien koennen auf drei verschiedene Arten erzeugt werden: durch Verwendung des Auswahlprozessors, durch Verwendung der von ENTER gebotenen Anfragen ueber Bildmasken oder durch Verwendung eines Programm. Die durch diese drei verschiedenen Methoden erzeugte Auswahldatei enthaelt den gewaehlten Datensatztyp, der dann vom Listenprozessor zur Erzeugung der endgueltigen Ausgabe verwendet wird.

8.1.1 Starten der Auswahl

Der Auswahlprozessor kann entweder vom Menue-Handler, von der Shell oder von einem Shell-Skript gestartet werden. Wird er vom Menue-Handler oder von einer (interaktiven) Shell aus aufgerufen, gibt er den Prompter * aus und akzeptiert so lange Kommandos, bis das Kommando end eingegeben wird. Zur nicht interaktiven Ausfuehrung einer Anfrage wird folgende Syntax verwendet:

```
LST anfrage_skript
```

Wird der Auswahlprozessor auf diese Weise gestartet, fuehrt er die im anfrage_skript enthaltenen Kommandos so lange aus, bis das Ende der Datei erreicht ist.

8.1.2 Syntax der Auswahl

Die folgenden Symbole werden zur Beschreibung der Syntax des Auswahlprozessors verwendet:

unterstrichen - bedeutet, dass der Nutzer anstelle dieses unterstrichenen Wortes den entsprechenden Namen oder Wert einsetzen muss. So wuerde beispielsweise datei_name bedeuten, dass hier der gewuenschte Dateiname stehen sollte. Weitere in dieser Weise verwendete Werte sind: ausdr fuer einen Ausdruck (siehe 8.1.2.1), subset fuer eine vom Auswahlprozessor angelegte benannte Auswahldatei, feld fuer den Namen eines Datenbank-Feldes, password fuer ein Passwort, das einem geschuetzten Feld zugeordnet ist und string fuer eine beliebige alphanumerische Zeichenkette.

[ ] Die eckigen Klammern geben an, dass dieser Ausdruck

bzw. diese Ausdruecke optionell sind.

beliebig oft wiederholt werden kann.

|| Senkrechte Striche dienen zur Begrenzung von Optionen, von denen eine zur Erstellung einer gueltigen Anfrage gewaehlt werden muss.

8.1.2.1 Ausdruecke

Der Begriff ausdr wird vom Auswahlprozessor zur Bezugnahme auf eine algebraische Kombination aus Feldern, Operatoren und Konstanten benutzt. Hier die Syntax fuer ausdr.

ausdr	+	ausdr
feld	-	feld
nummer	*	nummer
string	/	string
ref_feld	=	substr_spez
	!=	ref_feld
	<=	
	>=	
	<	
	>	
	substr	
	and	
	or	

Jeder Ausdruck oder Teilausdruck kann zur Angabe der Prioritaet in runde Klammern eingeschlossen werden. Werden keine runden Klammern verwendet, werden die Operatoren folgendermassen bearbeitet: hoechste Prioritaet [*,/], mittlere [+,-], niedrigste [<,>,<=,>=]. Zwei Elemente vom Typ string koennen unter Verwendung des Operators + miteinander verkettet werden. Ansonsten werden nur die Vergleichszeichen und substr auf diese Elemente angewandt. substr wird nur auf Elemente vom Typ string angewandt. substr_spez hat die Form start-end, wobei start die Position des Anfangszeichens (von 1 an indiziert) und end die Position des Endezeichens ist. So sind beispielsweise die ersten 10 Zeichen einer Zeichenkette 1-10.

Referenzfelder ref_feld haben die Syntax:

```
feld.feld ...
```

Sie spezifizieren Felder, die in anderen Datensatztypen als dem verwendeten sind. Das erste Feld muss denselben Typ und dieselbe Laenge haben wie der Schluessel des Datensatztyps, der das zweite Feld enthaelt usw. Jedes Feld wird fuer den Zugriff auf das naechste Feld verwendet bis das letzte Feld erreicht ist, das der Wert des Elementes ist.

8.1.2.2 select

SYNTAX

```
select | subset    | [into string] where ausdr end
       | datei_name |
```

VERWENDUNG

Dieses Kommando wird verwendet, um einen Teil einer Datei oder die aktuelle Teilmenge, subset, auszuwaehlen. Die Ergebnisse der Auswahl werden in eine Teilmenge, subset, gebracht, die entweder dem Listenprozessor oder einem Programm uebergeben wird oder in einem spaeter folgenden select verwendet wird.

Ist der auf select folgende Name keine vorher ausgewaehlte Teilmenge, wird die Auswahl unter Verwendung der gesamten Menge der in der spezifizierten Datei angegebenen Datensaetze vorgenommen. Andernfalls erfolgt die Auswahl aus der angegebenen Teilmenge, subset. Ein Datensatz wird dann ausgewaehlt, wenn das Ergebnis des Ausdrucks ausdr fuer diesen Datensatz ungleich Null ist.

Standardmaessig werden die Ergebnisse in eine Teilmenge gebracht, die denselben Namen hat, wie die Teilmenge oder Datei, aus der sie stammt. Bei Verwendung der Syntax into wird der Name der ausgegebenen Teilmenge durch string spezifiziert.

8.1.2.3 remove

SYNTAX

```
remove subset
```

VERWENDUNG

Dieses Kommando wird verwendet, um die vorher ausgewahlten Teilmengen zu loeschen. Da select (8.1.2.2) nur eine Auswahl aus einer Datei vornimmt, (wenn kein subset gleichen Namens vorhanden ist), ist es i.a. angebracht, Teilmengen nach ihrer Verwendung zu loeschen. Es ist jedoch auch moeglich, den Namen der Teilmenge, subset, mit call (8.1.2.4) zu aendern.

8.1.2.4 call

SYNTAX

```
call subset string
```

VERWENDUNG

Dieses Kommando wird verwendet, um den Namen einer existierenden Teilmenge zu aendern. Die spezifizierte Teilmenge, subset, wird in string umbenannt. Die umbenannte Teilmenge behaelt alle Eigenschaften der urspruenglichen Teilmenge bei.

8.1.2.5 copy

SYNTAX
copy subset string

VERWENDUNG
Dieses Kommando wird verwendet, um eine Kopie einer existierenden Teilmenge anzufertigen. Das Kommando ist dann nuetzlich, select erneut ausgefuehrt werden soll, waehrend die Ergebnisse der ersten Auswahloperation noch verarbeitet werden. Die mit einem Namen belegte Teilmenge, subset, wird in eine neue Teilmenge kopiert, die den Namen string hat. Die Kopie behaelt alle Eigenschaften des Originals bei.

8.1.2.6 list

SYNTAX
list generic_subset_name

VERWENDUNG
Dieses Kommando wird zur Auflistung der existierenden Teilmengen verwendet. generic_subset_name hat die Charakteristika der Dateispezifikation im WEGA-Kommando ls (1). In generic_subset_name entspricht * einer beliebigen Anzahl von Zeichen, ? entspricht einem einzelnen Zeichen und [...] entspricht einem einzelnen Zeichen der spezifizierten Menge. Fuer jede der Spezifikation entsprechende Teilmenge wird der WEGA-DATA-Datensatztyp und die Anzahl der Datensaetze aufgelistet.

8.1.2.7 unlock

SYNTAX
unlock feld password

VERWENDUNG
Dieses Kommando gestattet das Lesen eines gesicherten Feldes (siehe Abschnitt 3.3.1 Field Level Security). Es kann entweder das das fuer das Lesen der Datei erforderliche Passwort oder das fuer das Beschreiben der Datei erforderliche Passwort verwendet werden. Jeder Versuch, vor diesem Kommando Zugriff auf ein gesperrtes Feld zu erlangen,fuehrt zu einer Fehlermeldung.

8.1.2.8 report

SYNTAX
report subset

VERWENDUNG
Mit diesem Kommando wird der Listenprozessor aufgerufen (siehe Abschnitt 8.2). Der Listenprozessor ist ein allgemeines Werkzeug, das einfache Listen ausgewahlter Datensaetze anlegen kann.

Der Name subset ist die Teilmenge der Datenbank, die der Listenprozessor bearbeitet.

8.2 Auflisten von Datensaetzen

Wurde unter Verwendung von ENTER, des Auswahlprozessors oder eines Programmes eine Menge von Datensaetzen ausgewaehlt, kann man den Listenprozessor zur Erzeugung eines formatierten Reports verwenden. Er ist leicht zu handhaben und dennoch sehr leistungsfaehig. Ist der zu bearbeitende Report so kompliziert, dass er vom Listenprozessor nicht bewaeltigt werden kann, kann der Reportgenerator RPT gemaess der in Kapitel 7 gegebenen Beschreibung verwendet werden.

Die einfachste Form der Verwendung besteht darin, dass man einfach die aufzulistenden Felder angibt und LST ordnet die Spalten, Ueberschriften und Formate zu. Bei der komplexesten Form kann man angeben, in welche Spalten und auf welchen Zeilen die jeweiligen Ausdruecke und Felder auszugeben sind und welches Format zu verwenden ist. Summen, Teilsummen und die Reihenfolge der Sortierung koennen definiert werden. Ausserdem ist es moeglich mehrzeilige Spalten- und Listenueberschriften zu spezifizieren.

8.2.1 Starten des Listenprozessors

Es gibt mehre Moeglichkeiten, wie man den Listenprozessor starten kann. Die einfachste Moeglichkeit waere die Verwendung des Kommandos report im Auswahlprozessor. Der Listenprozessor gibt den Prompter '-' aus und aktzeptiert die Kommandos interaktiv. Das Programm wird mit dem Kommando end beendet. Man kann den Listenprozessor auch benutzen, indem man ihn als einen mit einer ENTER-Bildmaske registrierten Report verwendet. Dazu werden die Auflistungskommandos in eine Datei gebracht und der Dateiname wird dann unter Verwendung von 'ENTER Screen Registration' (siehe Abschnitt 5.1) registriert. Ausserdem kann man den Listenprozessor direkt von der Shell oder einem Shell-Skript aus ablaufen lassen. Die dafuer erforderliche Syntax ist folgende:

```
LST -r selektions_datei report_skript
```

8.2.2 Syntax des Listenprozessors

Die folgenden Symbole werden zur Beschreibung der Syntax des Listenprozessors verwendet:

unterstrichen - bedeutet, dass der Nutzer anstelle dieses unterstrichenen Wortes den entsprechenden Namen oder Wert einsetzen muss. So wuerde beispielsweise datei_name bedeuten, dass hier der gewuenschte Dateiname stehen sollte. Weitere in dieser Weise verwendete Werte sind: ausdr fuer einen Ausdruck (siehe 8.1.2.1), subset fuer eine vom Auswahlprozessor angelegte benannte Auswahldatei, feld fuer den Namen eines Datenbank-Feldes, password fuer ein Passwort, das einem geschuetzten Feld zugeordnet ist und string fuer eine beliebige alphanumerische Zeichenkette.

[ ] Die eckigen Klammern geben an, dass dieser Ausdruck bzw. diese Ausdruecke optionell sind.

beliebig oft wiederholt werden kann.

|| Senkrechte Striche dienen zur Begrenzung von Optionen, von denen eine zur Erstellung einer gueltigen Anfrage gewaehlt werden muss.

8.2.2.1 Ausdruecke

Der Begriff ausdr wird vom Listenprozessor zur Bezugnahme auf eine algebraische Kombination aus Feldern, Operatoren und Konstanten benutzt. Hier die Syntax fuer ausdr.

ausdr	+	ausdr
feld	-	feld
nummer	*	nummer
string	/	string
ref_feld	=	substr_spez
	!=	ref_feld
	<=	
	>=	
	<	
	>	
	substr	
	and	
	or	

Jeder Ausdruck oder Teilausdruck kann zur Angabe der Prioritaet in runde Klammern eingeschlossen werden. Werden keine runden Klammern verwendet, werden die Operatoren folgendermassen bearbeitet: hoechste Prioritaet [*,/],

mittlere [+,-], niedrigste [<,>,<=,>=]. Zwei Elemente vom Typ string koennen unter Verwendung des Operators + miteinander verkettet werden. Ansonsten werden nur die Vergleichszeichen und substr auf diese Elemente angewandt. substr wird nur auf Elemente vom Typ string angewandt. substr_spez hat die Form start-end, wobei start die Position des Anfangszeichens (von 1 an indiziert) und end die Position des Endezeichens ist. So sind beispielsweise die ersten 10 Zeichen einer Zeichenkette 1-10.

Referenzfelder ref_feld haben die Syntax:

```
feld.feld ...
```

Sie spezifizieren Felder, die in anderen Datensatztypen als dem verwendeten sind. Das erste Feld muss denselben Typ und dieselbe Laenge haben wie der Schluessel des Datensatztyps, der das zweite Feld enthaelt usw. Jedes Feld wird fuer den Zugriff auf das naechste Feld verwendet bis das letzte Feld erreicht ist, das der Wert des Elementes ist.

8.2.2.2 list

SYNTAX

```
list [column nummer [line nummer]]
     [name] ausdr [using format]
     [, [name] ausdr [using format]....]
     [under name] end
```

VERWENDUNG

Das Kommando list wird fuer die Ausgabe einer Serie von Ausdruecken verwendet. Man kann eine beliebige Anzahl von list-Kommandos eingeben. Die im Kommando list aufgefuehrten Ausdruecke werden fuer jeden in der Auswahldatei befindlichen Datensatz einmal ausgegeben. Die Reihenfolge wird durch vorher erfolgte oder nachfolgende sort-Kommandos bestimmt.

Wurde ein Name, name, vorher in einem sort-Kommando definiert, kann er nicht mehr umdefiniert werden. Andernfalls ordnet name dem Ausdruck eine Spaltenueberschrift zu. Erscheint in name ein /, wird dadurch angezeigt, dass die Ueberschrift in eine neue Zeile kommt. Ein Name kann eine beliebige Anzahl von / enthalten, wodurch eine mehrzeilige Ueberschrift entsteht. Aus einem einzelnen Feld bestehende Ausdruecke haben den Feldnamen als Standardueberschrift. Ansonsten gibt es keine Standardueberschrift. Wurde fuer einen Ausdruck eine Ueberschrift erstellt, ist diese Ueberschrift in den darauf folgenden Kommandos anstelle des Ausdruckes zu verwenden.

Standardmaessig werden den Ueberschriften auf Zeile 1 Spalten von links nach rechts zugeordnet. Die Anzahl der verwendeten Leerstellen wird durch die Ausgabe-

groesse des Ausdrueckes bestimmt, wenn dieser groesser ist als die Groesse der Ueberschrift, oder sie wird durch die Groesse der Ueberschrift bestimmt, wenn diese groesser ist. Diese standardmaessige Anzahl von Leerstellen kann geaendert werden, indem man nach dem Kommando list die Zeile, line, und die Spalte, column, absolut setzt, oder man kann die Syntax under verwenden, um anzuzeigen, dass die Ausdruecke unter einem vorher definierten Ausdruck erscheinen sollen.

Die Felder numeric, float und amount werden normalerweise in Standardfeldbreite ausgegeben. Die Standards sind folgendermassen:

numeric	9 Leerzeichen breit
amount	10 Leerzeichen breit
float	16 Leerzeichen breit

Diese Breite kann unter Verwendung der Option using und einem aus Sonderzeichen bestehenden format geaendert werden, wobei fuer jede Position in der Feldbreite ein Sonderzeichen verwendet wird. Das Sonderzeichen gibt an, was an der entsprechenden Position auszugeben ist. Fuer die Felder numeric und amount werden folgende Sonderzeichen verwendet (dabei sollte man von der Vorstellung ausgehen, dass ein Feld von rechts nach links formatiert und ausgegeben wird.):

\# - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Leerzeichen gedruckt. Damit wird ein numerisches Feld links mit Leerzeichen aufgefuellt.

& - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird eine Null gedruckt. Damit wird ein numerisches Feld links mit Nullen aufgefuellt.

* - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Stern gedruckt.

$ - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Dollarzeichen gedruckt. Wurde das Dollarzeichen bereits gedruckt, wird ein Leerzeichen gedruckt.

\+ - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Andernfalls wird ein Plus-Zeichen gedruckt. Wurde das Plus-Zeichen bereits gedruckt, wird ein Leerzeichen gedruckt.

\- - Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Handelt es sich um eine negative Zahl, wird ein Minuszeichen gedruckt. Wurde ein Minuszeichen bereits gedruckt, wird ein Leerzeichen gedruckt.

(- Befindet sich an dieser Stelle eine Ziffer, wird diese Ziffer gedruckt. Handelt es sich um eine negative Zahl, wird eine linke runde Klammer gedruckt. Wurde bereits eine linke runde Klammer gedruckt, wird ein Leerzeichen gedruckt.

) - Handelt es sich hierbei um eine negative Zahl, wird an dieser Stelle eine rechte runde Klammer gedruckt.

, - Befindet sich links von dieser Stelle eine Ziffer, wird ein Komma gedruckt. Andernfalls wird ein Leerzeichen gedruckt.

.- An dieser Stelle soll ein Dezimalpunkt gedruckt werden.

Fuer Felder vom Typ FLOAT wird eine Druckspezifikation benutzt, die genau der printf-Funktion aus C entspricht. Die Druckspezifikation hat folgendes Format:

%[-][minimale_feld_breite][.][genauigkeit] f|e|g

Optionen in den eckigen Klammern "[" und "]" muessen nicht angegeben werden. Die zur Abgrenzung der Elemente in der Liste verwendeten Vertikalstriche "|" geben an, dass ein Element in der Liste auszuwaehlen ist. Das Prozentzeichen "%" muss angegeben werden.

Das vor der Konvertierungsspezifikation stehende optionelle Minuszeichen "-" zeigt an, dass das Ergebnis in der Feldbreite linksbuendig ausgerichtet werden soll. Hat das Ergebnis weniger Zeichen, wird es auf minimale_feld_breite aufgefuellt. Mit genauigkeit wird die Anzahl der Ziffern angegeben, die hinter dem Dezimalpunkt stehen sollen. Wenn genauigkeit als 0 angegeben wird, werden hinter dem Dezimalpunkt keine Ziffern gedruckt. Wird kein Punkt angegeben, wird davon ausgegangen, dass die Zahl vor dem Konvertierungszeichen die Genauigkeit ist. Die Konvertierungszeichen (f, e und g) haben folgende Bedeutung:

f - Das Feld wird in eine dezimale Notation der Form [-]ddd.ddd konvertiert, wobei die Anzahl der nach dem Dezimalpunkt stehenden Ziffern gleich der spezifizierten Genauigkeit ist. Wird keine Genauigkeit angegeben, werden 6 Ziffern ausgegeben. Wird die Genauigkeit explizit mit 0 angegeben, wird kein Dezimalpunkt gedruckt.

e - Das Feld wird in der Form [-]d.ddde+-dd konvertiert, wobei eine Ziffer vor dem Dezimalpunkt steht und die Anzahl der Ziffern nach dem Dezimalpunkt gleich der angegebenen Genauigkeit ist. Wird die Genauigkeit nicht angegeben, werden 6 Ziffern ausgegeben. Wird die Genauigkeit explizit mit 0 angegeben, wird kein Dezimalpunkt gedruckt.

g - Das Feld wird in die Form f oder in die Form e konver-

tiert, je nachdem, welche der beiden Formen bei minimaler Platzbeanspruchung die volle Genauigkeit liefert.

In den folgenden Beispielen wird veranschaulicht, welche Wirkung die verschiedenen Schablonen auf die Ausgabe haben.

Format	Wert	Resultat
"#####"	123	" 123"
"#####.##"	0	" "
"#####.&&"	0	" .00"
"+++,+++,+++"	23456	" +23,456"
"---,---.&&"	23456.78	" 23,456.78"
"---,---.&&"	-2345.67	" -2,345.67"
"%10.2f"	12.3	" 12.30"
"%10.2f"	123.456	" 123.46"
"%12.4e"	123.456	" 1.235e+02"
"%10.4g"	123.456	" 123.4"
"%8.4g"	123456789	"1.23e+08"

8.2.2.3 sort

SYNTAX

sort [reverse]
[uniquely]
[name] ausdr [, [name] ausdr....] end

VERWENDUNG

Dieses Kommando wird zur Bestimmung der Reihenfolge der ausgegebenen Listen verwendet. Wird das Kommando sort nicht verwendet, erfolgt die Ausgabe in keiner bestimmten Reihenfolge. Standardmaessig erfolgt die Ausgabe nach dem Kommando sort in der Reihenfolge der angegebenen Ausdruecke. Wird die Syntax reverse verwendet, erfolgt die Ausgabe in umgekehrter Reihenfolge. Das optionelle Argument name gibt ausdr einen Namen, der dann in spaeter auftretenden Kommandos total und list verwendet werden kann. Wird ein Name spezifiziert und wird der Ausdruck aufgelistet oder addiert, wird name als Spaltenueberschrift verwendet.

Die Syntax uniquely spezifiziert, dass identische Zeilen nur einmal ausgegeben werden. Soll die Zeile unterdrueckt werden, muss die gesamte Zeile und nicht nur der Sortierausdruck vollstaendig identisch sein.

8.2.2.4 total

SYNTAX

```
    total  |  [name] ausdr  |
           |   name         |  [using format], ....
           |   ausdr        |
           |                |  [by name [, name] end
```

VERWENDUNG

Dieses Kommando wird zur Ausgabe von Summen und Zwischensummen von Ausdruecken verwendet. Standardmaessig werden die benannten Ausdruecke fuer die gesamte Menge der ausgewaehlten Datensaetze addiert. Die Syntax by ermoeglicht die Erzeugung von Summen bei Auftreten von Gruppenwechseln. Die nach by spezifizierten Namen, name, sind die Namen der vorher in sort-Anweisungen aufgetretenen Ausdruecke. Fuer diese Ausdruecke wird bei jeder Aenderung des Wertes eine Summe gebildet.

Hat der Ausdruck vorher einen Namen erhalten, ist nur der name des Ausdrucks zu verwenden. Erhielt der Ausdruck seinen Namen in einer list-Anweisung, sind Ausgabespalte und -format gleich den in der Anweisung list spezifizierten. Trat der Ausdruck nicht in einer Anweisung list auf, kann der Name als Spaltenueberschrift fuer den Ausdruck verwendet werden. Wird im Namen ein / verwendet, bedeutet das, dass die Ueberschrift auf eine neue Zeile kommt. In einem Namen koennen beliebig viele / vorkommen. Wird kein Name spezifiziert und besteht der Ausdruck aus einem einzelnen Feld, ist standardmaessig der Feldname die Ueberschrift. Ansonsten gibt es keine Standardueberschrift.

Hat der Ausdruck in einer vorangegangenen Anweisung list keinen Namen erhalten, kann die Syntax using fuer die Definition des Ausgabeformates verwendet werden. Siehe Abschnitt 8.2.2.2.

8.2.2.5 go

SYNTAX

```
    go
```

VERWENDUNG

Dieses Kommando fuehrt dazu, dass die spezifizierte Auflistung in die Standardausgabe geschrieben wird. Die Ausgabe wird jeweils nach 23 Zeilen unterbrochen, um eine Wechselwirkung mit dem Nutzer zu ermoeglichen. die Spaltenueberschriften werden fuer jede neue Seite erneut ausgegeben, ausser wenn vorher das Kommando nohead gegeben wurde.

Es wird empfohlen, print in den LST-Skripten zu verwenden, die als Formatierungsparameter fuer ENTER-Bildmasken-Reporte bestimmt sind (siehe Abschnitt 5.1 und

Benutzerhandbuch, Abschnitt 14.).

8.2.2.6 print

SYNTAX
print report_kopf

VERWENDUNG
Dieses Kommando fuehrt dazu, dass die spezifizierte Auflistung an das Standardausgabegereaet geschickt wird. Neben den angegebenen Spaltenueberschriften werden Seitennummern, Datum, Zeit und Reportueberschrift ausgegeben. Die Reportueberschrift, report_kopf, wird auf jeder Seite ausgegeben. Ein in der Reportueberschrift stehendes / erzwingt eine neue Zeile. Man kann beliebig viele / in einer Reportueberschrift verwenden, was bedeutet, dass eine mehrzeilige Ueberschrift ausgegeben wird.

Es wird empfohlen, print in den LST-Skripten zu verwenden, die als Formatierungsparameter fuer ENTER-Bildmasken-Reporte bestimmt sind (siehe Abschnitt 5.1 und Benutzerhandbuch, Abschnitt 14.).

8.2.2.7 nohead

SYNTAX
nohead

VERWENDUNG
Wird dieses Kommando vor einem Kommando go ausgegeben, erfolgt die Ausgabe ohne Spaltenueberschriften. Das ist dann sinnvoll, wenn Daten ohne zusaetzliche Informationen von der Datenbank in WEGA-Dateien abgezogen werden sollen.

8.2.2.8 unlock

SYNTAX
unlock feld password

VERWENDUNG
Dieses Kommando wird verwendet, um ein Feld, das durch 'Field Level Security' geschuetzt war, zu entriegeln. Zum Entriegeln eines Feldes kann entweder das Lese- oder Schreibpasswort verwendet werden. Wird versucht, ohne dieses Kommando, auf ein gesichertes Feld zuzugreifen, wird eine Fehlermeldung ausgegeben.

9. DER DATENBANK-TESTTREIBER

Dieses Programm gestattet , dass die in Abschnitt 10.3 beschriebenen Datensatz- und expliziten Beziehungsfunktionen im interaktiven Modus verwendet werden koennen, wenn eine Datenbank-Datei angelegt wurde. Dadurch koennen Testdaten schnell und einfach eingegeben und die Ergebnisse von Testlaeufen koennen ueberprueft werden. Die verwendete Syntax ist sehr einfach. Ist das Programm bereit zur Eingabe, wird am Terminal der Doppelpunkt (:) als Prompter angezeigt. Die Namen der Funktionen und die Namen der Datensaetze und Felder werden genauso eingetragen wie sie im Schema verwendet wurden. Sind fuer die Funktion zusaetzliche Daten erforderlich, fragt das Programm per Prompter nach. Um zu ueberpruefen, ob die richtigen Namen verwendet wurden, wird auf Gueltigkeit ueberprueft. Kann eine Funktion nicht verwendet werden, wird der Wert des Statuskodes ausgegeben. Die Bedeutung dieser Werte ist in Abschnitt 10.3 angefuehrt. Soll aus dem Testtreiber ausgetreten werden, wird im Prompter end oder q eingegeben. Gestartet wird dieses Programm als Punkt 9 des System-Menues.

acckey datsat
  oder
access datsat
: Es wird nach dem Datensatzschluessel gefragt und es wird auf den angegebenen Datensatz zugegriffen.

addrec datsat
: Es wird nach dem Datensatzschluessel gefragt und der angegebene Datensatz wird in die Datei eingefuegt. Dann wird der Reihe nach nach allen Feldern im Datensatz gefragt. Nach Eingabe aller Felder wird der Datensatz angezeigt.

clr feld rfeld
: Der angegebene Satz wird geloescht. Entspricht der Funktion clrset.

delete datsat
: Der aktuelle Datensatz vom angegebenen Typ wird geloescht.

faccess datsat feld
: Auf den angegebenen Datensatz wird unter Verwendung des Wertes des angegebenen Feldes zugegriffen. Es ist zu beachten, dass ein Datensatz von einem das Feld enthaltenden Typ der aktuelle sein muss.

loc datsat
: Die Position des aktuellen Datensatzes vom spezifizier-

ten Typ wird ausgegeben. Diese Position kann spaeter fuer die Funktion setloc verwendet werden.

makeset feld rfeld option
Der angegebene Satz der miteinander in Beziehung stehenden Datensaetze wird fuer spaeter folgende Aufrufe von nextrec oder preverc markiert. Die option wird aus dem Satz (first, last) ausgewaehlt.

next feld rfeld
Der naechste im angegebenen Satz kommende Datensatz wird zum aktuellen Datensatz gemacht. Entspricht der Funktion nextrec.

field
Der Inhalt eines Feldes kann einfach dadurch geaendert werden, dass ueber Tastatur der Name des Feldes eingegeben wird. Entspricht der Programmfunktion pfield. Das Programm fragt per Prompter nach den Felddaten und nimmt die angegebene Aktualisierung vor.

prev feld rfeld
Der im angegebenen Satz zuvor stehende Datensatz wird zum aktuellen. Entspricht der Funktion prevrec.

samerec feld rfeld
Der aktuelle Datensatz des angegebenen Satzes wird wiederhergestellt.

seq datsat option
Der angegebene Datensatz wird zum aktuellen Datensatz und wird angezeigt. Die Option wird aus dem Satz (first, next, last, prev) ausgewaehlt.

setloc datsat adresse
Der auf der angegebenen Adresse befindliche Datensatz wird zum aktuellen. Ungueltige Datensatz-Adressen sind nicht erlaubt.

setsize feld rfeld
Die Anzahl der im angegebenen Satz vorhandenen Datensaetze wird angezeigt.

10. C-SPRACH-INTERFACE

In diesem Abschnitt werden die durch die C-Programmiersprache zur Verfuegung stehenden Funktionen zur Manipulation der Datenbank beschrieben. Obgleich die C-Sprache die am haeufigsten unter WEGA verwendete Sprache ist, besitzt WEGA-DATA auch Interfaces zu anderen Sprachen, wie z.B. zu COBOL. Diese Funktionen bilden die Verbindung zwischen Nutzerprogrammen, Terminals, Druckern und der Datenbank. Es wird davon ausgegangen, dass der Leser bereits mit den Konzepten des Datenbankbetriebssystems, Datensatzbeziehungen, und der prozeduralen Manipulation von Datensaetzen in einer Datenbank vertraut ist. Die WEGA-DATA-Routinen ermoeglichen den Nutzerprogrammen folgendes:

1. Hinzufuegen und Loeschen von Datensaetzen.
2. Aenderung des Inhalts von Datenfeldern.
3. Akzeptieren von ueber Bildschirm erfolgten Eingaben und Ausgaben.
4. Durchforsten der Datenbank unter Verwendung der verschiedenen Zugriffsmethoden.
5. Formatierung von an den Drucker gehenden Ausgaben.
6. Sortierung und Auswahl von Datensaetzen.

Sollen diese Funktionen verwendet werden, muss vorher eine Datenbank-Datei angelegt worden sein. In Abschnitt 2 wurde die Verwaltung der Datenbank erlaeutert. Hat man eine Datenbank angelegt, hat man auch eine Datei file.h, die eine Textdatei ist, die allen Datensatz- und Feldnamen interne WDATA-Nummern zuordnet. Sie muss in jede Nutzerfunktion eingeschlossen sein, bei der Feld- oder Datensatznamen verwendet werden.

Ausserdem muss man beim Laden von Datenbank-Funktionen benutzenden Programmen, das WDATA-Datenbank-Archiv beruecksichtigen. Dieses Archiv heisst libd.a und befindet sich im Verzeichnis lib des (Datenbank-)Systems. Wurde die Umgebungsvariable WDATA so gesetzt, dass sie auf dieses Verzeichnis zeigt, kann man mit $WDATA/libd.a darauf Bezug nehmen. Die nicht-datenbankbezogenen Dienstfunktionen befinden sich im Archiv libx.a, das sich im Verzeichnis lib des Systems befindet und auf das man mit $WDATA/libx.a Bezug nehmen kann. Im Verzeichnis bin des Systems befindet sich eine Datei uld, die Shell-Ladekommandos enthaelt. Die Verwendung dieser Datei wird im naechsten Abschnitt, Kompilieren und Laden von Programmen, erlaeutert.

In den folgenden Funktionsbeschreibungen werden fuer die Benennung von Argumenten einige Vereinbarungen getroffen. Hier eine kurze Beschreibung dieser Vereinbarungen:

- rnum - ein beliebiger Datensatzname
- feld - ein beliebiger Feldname
- rfeld - ein Feld, das sich auf das Schluesselfeld eines anderen Datensatzes bezieht, d.h. eine explizite Beziehung

Es ist wichtig, dass man den Unterschied zwischen internen und externen Feldformaten kennt. Zur Platzeinsparung und Vereinfachung von Manipulationen verwendet WEGA-DATA sowohl in der Datenbank als auch bei den Nutzerfunktionen kompakte Speicherungsformen. Diese werden auf externe Form erweitert, wenn ein Feld an ein Anzeigegeraet wie Bildschirm oder Drucker gesendet wird. Im Schema wird die externe Form angegeben, die interne Form kann davon abgeleitet werden.

Name	intern	extern
NUMERIC (1-4)	short	9999
(5-9)	long	999999999
FLOAT	double	variabel
STRING	char[?]	xxxxxxx...
AMOUNT (1-7)	long	999999.99
(8-12)	double	99999999999.99
DATE	short (Julianisch)	MM/DD/YY
TIME	short	HH:MM

Nur die Felder vom Type DATE werden auf einen Leerwert ungleich Null initialisiert. Alle anderen Felder werden auf Nullen initialisiert. Bei DATE-Feldern wird der Leerwert intern durch -32768 und extern durch **/**/** dargestellt. Man kann ein AMOUNT-Feld auf Leer setzen (was etwas anderes als Null ist), indem man -32768 im Wort hoeherer Ordnung und Null im Wort niedriger Ordnung speichert. Auf Leer gesetzte AMOUNT-Felder werden extern als Leerstellen angezeigt.

Die Datentypen NUMBERIC, FLOAT und STRING werden genauso gespeichert, wie ihr Typ impliziert. AMOUNT-Felder werden intern als Pfennig und nicht als Mark und Bruchteile gespeichert. Die Ausgaberoutinen benutzen zum Setzen des Dezimalpunktes die Operatoren mod und div. TIME-Felder dienen zum Speichern der Information "Tageszeit" und nicht zum Speichern eines Zeitraumes, obwohl ein Zeitraum bis zu 23 Stunden und 59 Minuten dargestellt werden kann. Die Zeit 12:59 wird intern als ganze Zahl 1259 gespeichert. Die Ausgaberoutinen benutzen (wie bei AMOUNT-Feldern) mod und div, um die ganze Zahl zur Anzeige zu trennen.

Die WDATA-Funktionen verwenden die Notation eines "aktuellen" Datensatzes. Fuer jeden Datensatztyp kann ein Datensatz der aktuelle sein. Dieser aktuelle Datensatz ist fuer viele Funktionen das implizierte Argument. Bevor man mit einem Datensatz arbeiten kann, muss er zunaechst unter Verwendung von addrec, acckey, faccess, nextrec, prevrec, samerec, nextsel, prevsel usw. zum aktuellen Datensatz gemacht werden. Danach bleibt dieser Datensatz so lange der aktuelle, bis er durch einen anderen Datensatz desselben Typs ersetzt wird.

Zu Beginn des Programms werden alle Datensatztypen als nichtaktuell angesehen. WDATA fuehrt kein Programm aus, das

eine Funktion zu verwenden versucht, fuer die kein aktueller Datensatz vorhanden ist.

10.1 Kompilieren und Laden von Programmen

In diesem Abschnitt soll beschrieben werden, wie Programme kompiliert und geladen werden, wenn das Hostsprachen-Interface von WDATA fuer C-Programmen verwendet wird. Im allgemeinen erfolgt dieser Prozess aehnlich wie das Kompilierung und Laden von C-Programmen, aber es muss ein spezieller Preprozessor verwendet werden und die WDATA-Bibliotheken muessen mit geladen werden. Die beiden folgenden Funktionsbeschreibungen erlaeutern, wie man ucc, den C-Compiler von WEGA-DATA, und uld, den Lade-"Makro" von WDATA, verwendet.

Da beide Kommandodateien von der Shell verwendet werden, ist zu sichern, dass die Umgebungsvariablen PATH und WDATA korrekt gesetzt sind. Weitere Informationen siehe 1.1.3.

UCC

NAME
ucc - Preprozessor des C-Compilers von WEGA-DATA

SYNTAX
ucc -c [-O] datei ...

BESCHREIBUNG
ucc ist ein Programm, das zur Kompilierung von in C geschriebenen Anwenderprogrammen verwendet wird. ucc ruft zur Manipulation der include-Datei datei.h einen Vorprozessor auf. Der Preprozessor wird benoetigt, da sonst Namen, die mehr als acht Zeichen lang sind, nicht bewaeltigt koennen. Von WEGA-DATA generierte Feldnamen koennen viel laenger sein als acht Zeichen und befinden sich in der Datei datei.h. Der Preprozessor von WDATA sucht nicht #include, sondern $include. Deshalb muss ein Anwenderprogramm, das WEGA-DATA-Feldnamen verwendet, eine Zeile wie die folgende enthalten:

```
$include "../../def/file.h"
```

Nach Ablauf des Preprozessors ruft ucc mit denselben ihm uebergebenen Argumenten cc auf. Hier das Kommando ucc:

```
ucc -c [-O] datei1.c ... datein.c
```

-O ist optionell. Mit dieser Option wird der Optimierer fuer den C-Compiler aufgerufen. Erfolgt die Kompilierung der Dateien problemlos, bleiben die Objekte in den .o Dateien. Gegenwaertig ist es nicht moeglich, ucc zur direkten Erzeugung eines ablauffaehigen Objektes durch Weglassen von -c zu verwenden. Die erstellten Objektdateien sind unter Verwendung von uld, der Datei mit

den Shell-Ladekommandos von WDATA, zu laden.

RUECKKEHRKODES
0 - Programm kompiliert
1 - Preprozessor-Schritt nicht moeglich
2 - Kompilieren nicht moeglich
3 - Kompilieren durch Unterbrechung beendet

SIEHE AUCH
uld

FEHLER
Treten in einem Kommentar Apostrophe auf, wird die Vorverarbeitung beendet. Moeglicherweise bleiben Datenbank-Felder undefiniert.

Beispiel: /* Setze Theo's Bestelldatum */

Ein Kommando dieser Art fuehrt zu Problemen - also lieber gegen Grammatik verstossen.

ULD

NAME
uld, uuld - Lademakros von WEGA-DATA

SYNTAX
uld datei ...

BESCHREIBUNG
Die Kommandodateien uld und uuld sind universelle Makros fuer das Laden von WDATA-Programmen. Sie nehmen Bezug auf libd.a und libx.a, die Archive, in denen die Host-Sprachfunktionen enthalten sind. Die Kommandodatei uuld ist gleich der Kommandodatei uld, enthaelt jedoch die Bibliothek upintf.a, in der herausgeloeste Interfacefunktionen fuer die Host-Sprachfunktionen enthalten sind. Diese Kommandodateien verwenden die Umgebungsvariable $WDATA, die somit vor Verwendung der Kommandodateien gesetzt werden muss. Die C-Erweiterungs- und mathematischen) Bibliotheken heissen libc.a und libm.a und sind ebenfalls enthalten.

Hier das uld (uuld) Kommando:

uld binary [datei1.o...datein.o] [datei1.a...datein.a]

Das Argument binary ist der Name der ausfuehrbaren Datei,die durch das Laden als Ausgabe entsteht. Die verbleibenden Argumenten sind die fuer die Erstellung der ausfuehrbaren Datei erforderlichen .o- und .a-Dateien. Es koennen maximal acht .o und .a Dateien als Argumente uebergeben werden. Weitere Informationen ueber die Verwendung von uld und den Aufbau der entstehenden Speicherabbilddatei siehe Abschnitt 2.1.6.

10.2 Fehlerbehandlung von WEGA-DATA

In C-Programmen auftretende Fehler koennen in zwei Hauptgruppen eingeteilt werden: fatale Fehler und nichtfatale Fehler. Bei Auftreten von fatalen Fehlern erfolgt eine Meldung, die auf das Problem verweist, dann wird das entsprechende Programm abgebrochen. Danach kann mit adb der entstehende Dump (core) untersucht werden. Ein fataler Fehler wuerde beispielsweise dann auftreten, wenn man die Datanbankdatei (file.db) nicht zum Lesen und Schreiben eroeffnen koennte. Fatale Fehler duerfen in einem normal funktionierenden System nicht auftreten. Ihr Auftreten zeigt an, dass ein schwerwiegender Fehler im Programm oder in file.db vorhanden ist.

Nichtfatale Fehler geben einfach einen Statuskode an das Programm zurueck. Beispielsweise waere es ein nichtfataler, "normaler", Fehler, wenn bei Verwendung eines bestimmten Schluesselwertes keine Zugriff auf den Datensatz moeglich waere. Allerdings kann die Nichtbeachtung nichtfataler Fehler zu fatalen Fehlern fuehren. Erfolgte beispielsweise ohne vorherige Ueberpruefung des Status' vorangegangener acckey oder nextrec der Versuch, ein Feld von einem Datensatz zu erhalten, kann ein fataler Fehler entstehen, wenn acckey (nextrec) nicht erfolgreich waren.

Im folgenden Abschnitt sollen nur fatale Fehler beschrieben werden, da die nichtfatalen Fehler in den Abschnitten behandelt werden, in denen die einzelnen Funktionen abgehandelt werden. In den einzelnen Abschnitten werden die fatalen Fehler nicht beruecksichtigt, da sie bei den meisten Funktionen moeglich sind. Aber nicht nur fatale Fehler sind moeglich. Jede Funktion, die Datensatz- und/oder Feldnummern als Argumente verwendet, kann vollstaendig unerwartete Ergebnisse liefern, wenn eine ungueltige Nummer uebergeben wird. Das geschieht am haeufigsten, wenn statt einer Datensatznummer eine Feldnummer uebergeben wird. Unerwartete Probleme (Speicherfehler, Busfehler oder unerwartete fatale Fehler) sind manchmal auf diese Art Fehler zurueckzufuehren.

Fatale Fehler fallen in drei Kategorien: 1. Nutzerfehler, die auftreten, wenn das Nutzerprogramm eine unzulaessige oder undefinierte Operation auszufuehren versucht. 2. interne Fehler, bei denen an einer WEGA-DATA-Funktion ein Problem vorliegt. 3. Datanbank-Fehler, bei denen etwas mit file.db nicht stimmt. Die Funktion kann nicht immer unterscheiden, welche Art Fehler vorliegt, das ist Aufgabe des Programmierers.

Folgende Regeln koennen angewandt werden, um zwischen den verschiedenen Fehlerarten zu unterscheiden. Ist das Programm noch im Entwicklungsstadium und wird eine kleine, permanente Testdatenbank verwendet, handelt es sich bei dem fatalen Fehler fast immer um einen Programmfehler. Es ist zu ueberpruefen, ob alle Funktionen korrekt verwendet

wurden, ob bei auftretenden nichtfatalen Fehlern die entsprechenden Korrekturen vorgenommen wurden und ob Programme ihren Daten- oder Kodebereich ueberschreiben (in dieser Reihenfolge). Ein Programmfehler liegt dann mit grosser Wahrscheinlichkeit vor, wenn nur bei diesem Programm Probleme auftreten und andere ausgetestete Programme normal ablaufen.

Auch bei einem sorgfaeltig ausgetesteten Betriebsprogramm besteht die Moeglichkeit, dass ein Fehler nicht entdeckt wurde. Ergibt eine Ueberpruefung des Programms, dass es in Ordnung ist, besteht die naechstliegende Moeglichkeit darin, dass die Datenbank-Datei file.db verstuemmelt ist. Das ist insbesondere dann wahrscheinlich, wenn waehrend einer Aktion ein fataler Programmfehler oder ein Hardware-Ausfall aufgetreten ist. Treten auch bei anderen Programmen aehnliche Probleme auf, ist der Fehler mit grosser Wahrscheinlichkeit bei der Datenbank-Datei selbst zu suchen.

Das Datenwoerterbuch und/oder die Header-Information koennte verstuemmelt sein. Der Hash-Index koennte ungueltig sein oder eine Anzahl von Pointern koennte falsch sein. In jedem Fall laesst sich der Fehler am schnellsten und leichtesten dadurch beheben, dass man die neueste Backup-Kopie von file.db einliest und von vorn beginnt. Ist das nicht moeglich oder wuenschenswert, gibt es mehrere Moeglichkeiten zur Korrektur der Datei. Durch Ablauf von 'Reconfigure Data Base' (Abschnitt 3.2.2) kann man das Datenwoerterbuch und die Header-Information neu aufbauen. Durch Ablauf von 'Hash Table Maintenance' (Abschnitt 3.5.3) kann man den Hash-Index neu aufbauen und die Pointer koennen durch 'Explicit Relationship Maintenance' (Abschnitt 3.5.4) bearbeitet werden. Wenn auch durch den Neuaufbau der Datei das Problem nicht behoben werden kann, koennen die Daten unter Verwendung von SQL (Abschnitt 6.3.4) in ASCII-Dateien zwischengespeichert, die Datenbank neu angelegt (Abschnitt 3.2.1) und dann die Daten erneut unter Verwendung von 'Data Base Load' (Abschnitt 3.5.5 oder 6.3.5) zurueckgeladen werden.

Sollen fatale Fehler aufgespuert werden, ohne dass ein Abbruch erfolgt, kann man die nutzereigenen Fehlerbehandlungsroutinen vor den von WEGA-DATA gelieferten Routinen laden. Zum Auffinden von Fehlern stehen vier verschiedene Routinen zur Verfuegung. Die Namen und die Reihenfolge ihres Aufrufs wird im folgenden angefuehrt. Soll eine andere Routine verwendet werden, wird einfach vor den WEGA-DATA Archiven libd.a und libx.a explizit xxxx.o geladen. Dann wird anstelle der WEGA-DATA-Routinen die nutzereigene Routine verwendet.

ERROR

NAME
error, uerror, dbherr, pferr - Suchen von fatalen Fehlern

SYNTAX
```
error (str, ier)
char *str;

uerror (str, ier)
char *str;

dbherr (str)
char *str;

pferr (ier, str)
char *str;
```

BESCHREIBUNG
Diese Routinen werden von WEGA-DATA zum Auffinden fataler Fehler benutzt. Die angezeigte Meldung wird ausgegeben und mit Statuskode 99 wird exit aufgerufen. error wird in der Hauptsache verwendet, um Datenbankfehler und interne Fehler anzuzeigen; uerror wird fuer die Anzeige von Nutzerfehlern; dbherr fuer die Anzeige interner Fehler, die bei der Bearbeitung des Datenwoerterbuchs auftraten, verwendet und pferr wird fuer die Anzeige von Nutzerfehlern verwendet, die durch eine unsachgemaesse Verwendung der Funktion pform entstanden.

Sollen noch andere Arten der Fehlerbehandlung ausgefuehrt werden, muessen C-Routinen mit den gleichen Namen und Aufruffolgen geschrieben und vor den WEGA-DATA-Archiven libd.a und libx.a als .o's geladen werden.

RUECKGABEWERTE
An die Shell wird der Status '99' zurueckgegeben.

/bin/sort not found

Erklaerung:
Das WEGA-Sortierdienstprogramm /bin/sort konnte nicht abgearbeitet werden.

Moegliche Ursachen und/oder Korrekturen:
1. /bin/sort existiert nicht. sort ist in /usr/bin zu suchen, wenn es nicht in /bin ist. In diesem Fall wird die Datei kopiert oder per Link mit /usr/bin verbunden.

2. Der aktuelle WEGA-Nutzer kann das Programm nicht ablaufen lassen. Moeglicherweise muss der Modus der im Pfadnamen stehenden Verzeichnisse und des Sortierprogramms

selbst geaendert werden.

can't open xxxxxxxx
data base error
from loadscr ier= 1

Erklaerung:
Nutzerfehler. Die angegebene Bildmasken-Datei .q kann nicht zum Lesen geoeffnet werden.

Moegliche Ursachen und/oder Korrekturen:
1. Existiert die Datei nicht (suchen in . und /usr/lib), wird sie angelegt oder wiederhergestellt.

2. Existiert die Datei, ist ihr Modus so zu aendern, dass sie fuer den aktuellen WEGA-Nutzer lesbar ist oder man meldet sich durch Login als Nutzer mit Leserecht an.

data base error
from gett ier= n

Erklaerung:
Nutzerfehler, interner Fehler oder Datenbank-Fehler. Hier 4 verschiedene Fehlertypen, die je nach dem Wert von n diagnostiziert werden:

Moegliche Ursachen und/oder Korrekturen:
-1 bedeutet, dass eine ungueltige Datensatznummer an gett uebergeben wurde (kleiner als 1 oder groesser als die maximale Datensatznummer). Der am haeufigsten durch den Nutzer verursachte Fehler besteht darin, dass statt einer Datensatznummer eine Feldnummer uebergeben wird, z.B. Verwendung von acckey mit einer Feldnummer statt mit einer Datensatznummer. Sind die Funktionsparameter korrekt, handelt es sich um einen internen Fehler. Moeglicherweise ist der Fehler dadurch entstanden, dass das Programm sich selbst ueberschreibt oder dass eine WDATA-Routine sich selbst ueberschreibt oder dass file.db fehlerhaft ist.

-2 bedeutet, dass innerhalb des Datensatzes, aus dem die Daten zu holen sind, der Offset kleiner als 1 ist. Bezueglich der internen Fehler treffen die unter Punkt (-1) angefuehrten Bemerkungen zu.

-3 bedeutet, dass die Anzahl der zu lesenden Worte kleiner als 1 ist oder dass diese Anzahl dazu fuehrt, dass die Datensatzlaenge ueberschritten wird. Bezueglich der internen Fehler treffen die unter Punkt (-1) angefuehrten Kommentare zu.

-4 bedeutet, dass es keinen aktuellen Datensatz des Typs gibt, in dem das spezifizierte Feld enthalten ist. Gewoehnlich handelt es sich um einen Nutzerfehler, bei

dem z.B. acckey, nextrec oder prevrec nicht ausgefuehrt werden konnte und das Programm dennoch weiter abgelaufen ist. In seltenen Faellen kann es sich auch um einen Datenbank-Fehler handeln.

data base error
from input ier= 1

Erklaerung:
Nutzerfehler. Es wurde versucht, ein Bildmaskenfeld einzugeben (d.h. in der Datenbank zu speichern), fuer das kein Datenbank-Feld definiert wurde.

Moegliche Ursachen und/oder Korrekturen:
1. Fuer das Bildmaskenfeld ist ein Datenbank-Feld zu definieren und die Bildmaske ist neu zu kompilieren.

2. Andernfalls kann man inbuf verwenden, um die Eingabe an den Programmpuffer zurueckzugeben.

data base error
from lifo ier= -1

Erklaerung:
Interner Fehler. dieser Fehler tritt nur auf, wenn eine interne Konsistenzpruefung nicht durchgefuehrt werden konnte.

Moegliche Ursachen und/oder Korrekturen:
1. Interner Ueberlauf.

data base error
from put ier= n

Erklaerung:
Nutzerfehler, interner Fehler oder Datenbank-Fehler. Hier 4 verschiedene Fehlertypen, die je nach dem Wert von n diagnostiziert werden:

Moegliche Ursachen und/oder Korrekturen:
-1 bedeutet, dass eine ungueltige Datensatznummer an gett uebergeben wurde (kleiner als 1 oder groesser als die maximale Datensatznummer). Der am haeufigsten durch den Nutzer verursachte Fehler besteht darin, dass statt einer Datensatznummer eine Feldnummer uebergeben wird, z.B. Verwendung von acckey mit einer Feldnummer statt mit einer Datensatznummer. Sind die Funktionsparameter korrekt, handelt es sich um einen internen Fehler. Moeglicherweise ist der Fehler dadurch entstanden, dass das Programm sich selbst ueberschreibt oder dass eine WDATA-Routine sich selbst ueberschreibt oder dass file.db fehlerhaft ist.

-2 bedeutet, dass innerhalb des Datensatzes, aus dem die Daten zu holen sind, der Offset kleiner als 1 ist. Bezueglich der internen Fehler treffen die unter Punkt (-1) angefuehrten Bemerkungen zu.

-3 bedeutet, dass die Anzahl der zu lesenden Worte kleiner als 1 ist oder dass diese Anzahl dazu fuehrt, dass die Datensatzlaenge ueberschritten wird. Bezueglich der internen Fehler treffen die unter Punkt (-1) angefuehrten Kommentare zu.

-4 bedeutet, dass es keinen aktuellen Datensatz des Typs gibt, in dem das spezifizierte Feld enthalten ist. Gewoehnlich handelt es sich um einen Nutzerfehler, bei dem z.B. acckey, nextrec oder prevrec nicht ausgefuehrt werden konnte und das Programm dennoch weiter abgelaufen ist. In seltenen Faellen kann es sich auch um einen Datenbank-Fehler handeln.

```
db:nnn addr:nnn errno:nnn
  data base error
   from put-dwrit ier= -1
```

Erklaerung:
Interner Fehler oder Datenbank-Fehler. Der Fehler entstand beim Versuch, die Datenbank zu beschreiben. db ist der Schreibdatei-Deskriptor, addr ist die Schreibadresse in Byte. Und errno ist die WEGA-Fehlernummer, die von der Operation zurueckgegeben wird.

Moegliche Ursachen und/oder Korrekturen:
1. file.db existiert nicht oder kann vom aktuellen WEGA-Nutzer nicht beschrieben werden.

2. Ungueltige Anfangsadresse, in die geschrieben werden soll oder unsinnig viele zu schreibende Bytes. Entweder ueberschreibt sich das Programm selbst oder file.db ist fehlerhaft und muss wiederhergestellt werden.

```
data base error
    from recphys ier= -2
```

Erklaerung:
Datenbank-Fehler. Es handelt sich um einen schwerwiegenden Fehler in der Headerinformation von file.db.

Moegliche Ursachen und/oder Korrekturen:
1. Die Datenbank-Datei ist durch Ablauf von 'Reconfigure Data Base' neu zu erstellen.

data base error
from rloc ier= -1

Erklaerung:
Interner Fehler oder Datenbank-Fehler. Es handelt sich um einen schwerwiegenden Fehler in einer WEGA-DATA Routine oder der Datei file.db selbst.

Moegliche Ursachen und/oder Korrekturen:
1. Durch Ablauf von 'Reconfigure Data Base' ist die Datenbank neu zu erstellen.

data base error
from unlk ier= -1

Erklaerung:
Interner Fehler. Dieser Fehler tritt nur auf, wenn die interne Konsistenzpruefung nicht durchgefuehrt werden kann.

Moegliche Ursachen und/oder Korrekturen:
1. Ungewiss.

error from dread nnn
addr nnn
data base error
from dread-gett ier= -1

Erklaerung:
Interner Fehler oder Datenbank-Fehler. Dieser Fehler entstand beim Versuch, die Datenbank zu lesen. Der in dread (nnn) angegebene Fehlerkode ist der Rueckgabewert des Leseversuches, addr ist die in Worten gegebene Anfangsleseadresse.

Moegliche Ursachen und/oder Korrekturen:
1. file.db existiert nicht oder kann vom aktuellen WEGA-Nutzer nicht gelesen werden.

2. Ungueltige Adresse, von der an zu lesen ist oder sinnlose Anzahl von zu lesenden Bytes. Das Programm ueberschreibt sich entweder selbst oder file.db ist fehlerhaft und muss wiederhergestellt werden.

HTLEND nnn HTLSTRT nnn lloc nnn
data base error
from gthsh ier= -1

Erklaerung:
Interner Fehler oder Datenbank-Fehler. Wie wie der vorhergehende Fehler.

Moegliche Ursachen und/oder Korrekturen:

1. Siehe oben.

HTLEND nnn HTLSTRT nnn lloc nnn
data base error
from gthsh-hcheck ier= -1

Erklaerung:
Interner Fehler oder Datenbank-Fehler. Es handelt sich um einen schwerwiegenden Programm- oder Datenbank-Fehler. Es wurde eine Hash-Tabellen-Adresse generiert, die ausserhalb der Grenzen der Hash-Tabelle liegt. HTLEND ist die Endadresse der Hash-Tabelle. HTLSTRT ist die Anfangsadresse der Hash-Tabelle und lloc ist die den Fehler verursachende Adresse.

Moegliche Ursachen und/oder Korrekturen:
1. Moeglicherweise ist die Headerinformation in file.db verstuemmelt. Tritt bei allen anderen Programmen dasselbe Problem auf, liegt hier wahrscheinlich die Ursache. Die Datei muss durch Rekonfigurieren wiedergewonnen werden.

2. Tritt bei anderen Programmen das Problem nicht auf, handelt es sich um einen Nutzerfehler oder um einen internen Fehler. Es ist zu pruefen, dass sich das Programm nicht selbst ueberschreibt.

illegal field -gtube- nnn
data base error
from gtube ier= nnn

Erklaerung:
Nutzerfehler. Eine nichtanerkannte Feldnummer wurde an gtube uebergeben. Der Fehlerkode (ier) ist die an gtube uebergebene Feldnummer.

Moegliche Ursachen und/oder Korrekturen:
1. Das Datenbank-Feld kann geloescht worden sein. Ist das der Fall, muss das Programm umgearbeitet werden.

2. Das Programm kann eine falsche oder veraltete file.h verwendet haben. Die richtige file.h ist zu suchen und das Programm ist neu zu kompilieren und zu laden.

3. Wenn statt einer definierten Variablen aus file.h. eine ganzzahlige Variable als Feldnummer verwendet wird, ist zu ueberpruefen, ob diese richtig verwendet wird.

Insufficient memory for data dictionary

Erklaerung:
Nutzerfehler. Im Speicher ist nicht genuegend Platz

fuer die Informationen aus dem Datenwoerterbuch.

Moegliche Ursachen und/oder Korrekturen:
1. Wurde das Programm nicht bereits auf diese Weise geladen, ist es mit der -i-Option des Ladeprogramms zu laden. Dadurch werden 64k Daten- und 64k Kodebereich ermoeglicht.
2. Der den globalen Variablen zugewiesene Platz ist zu reduzieren. Entweder sind einige Daten und Programme zu loeschen oder einige globale Daten sind in den Stack umzuspeichern, indem diese lokal gemacht werden.

insufficient memory for additional dictionary information

Erklaerung:
Nutzerfehler. Ist im wesentlichen dasselbe Problem wie oben.

Moegliche Ursachen und/oder Korrekturen:
1. Siehe oben.

NO MORE SPACE FOR RECORD nnn
data dase error
from fetchbuff ier= -1

Erklaerung:
Nutzerfehler. In file.db ist kein Platz mehr zur Speicherung weiterer Datensaetze vom Typ nnn. Tritt auf, wenn die Datei 66% mehr Datensaetze hat, als beim Entwurf vorgesehen.

Moegliche Ursachen und/oder Korrekturen:
1. Einige existierende Datensaetze vom Typ nnn sind zu loeschen. Damit wird Platz frei, der zum Hinzufuegen weiterer Datensaezte verwendet werden kann.

2. 'Schema Maintenance' ist zu verwenden, um die Anzahl der erwarteten Datensaetze des angegebenen Typs zu vergroessern und die Datenbank zu rekonfigurieren.

Pform error no 1 from getpf

Erklaerung:
Nutzerfehler. Unzureichende Argumente auf der aktuellen Zeile in der Eingabedatei pform oder Argumente wurden nicht durch Kommas voneinander getrennt.

Moegliche Ursachen und/oder Korrekturen:
1. Die fehlerhafte Zeile der Eingabedatei ist zu suchen und zu korrigieren.

Pform errror no 2 from getpf

Erklaerung:
Nutzerfehler. Die Umwandlung einer x-Koordinate von ASCII in Dezimal war nicht erfolgreich.

Moegliche Ursachen und/oder Korrekturen:
1. Die aktuelle Eingabedatei pform ist auf nichtnumerische Zeichen in der x-Koordinaten-Position zu ueberpruefen.

Pform error no 3 from getpf

Erklaerung:
Nutzerfehler. Die Umwandlung einer y-Koordinate von ASCII in Dezimal war nicht erfolgreich.

Moegliche Ursachen und/oder Korrekturen:
1. Die aktuelle Eingabedatei pform ist auf nichtnumerische Zeichen in der y-Koordinaten-Position zu ueberpruefen.

Pform error no 4 from getpf

Erklaerung:
Interner Fehler.

Moegliche Ursachen und/oder Korrekturen:
1. Dieser Fehler darf nicht auftreten.

Pform error no 5 from getpf

Erklaerung:
Nutzerfehler. Eine x- oder y-Koordinate ist mehr als 3 numerische Zeichen lang.

Moegliche Ursachen und/oder Korrekturen:
1. Die aktuelle Eingabedatei pform ist zu untersuchen und die fehlerhafte Zeile zu berichtigen.

Pform error no 6 from getpf

Erklaerung:
Nutzerfehler. Der vom Nutzer gelieferte Name der Funktion ist mehr als 10 Zeichen lang.

Moegliche Ursachen und/oder Korrekturen:
1. Die aktuelle Eingabedatei pform ist auf den langen Namen zu ueberpruefen. Dann ist der Name auf 10 oder weniger Zeichen zu kuerzen. Das kann bedeuten, dass die Funktion umbenannt werden muss und dass alle Aufrufe zur Beruecksichtigung des neuen Namens geaendert

werden muessen.

Too many unisort sets are current

Erklaerung:
Nutzerfehler. Es koennen immer nur hoechstens 8 verschiedene unisort-Saetze gleichzeitig aktuell sein.

Moegliche Ursachen und/oder Korrekturen:
1. Das Programm ist zu ueberpruefen, ob zu viele verschiedene unisort-Saetze gleichzeitig verwendet werden. Insbesondere ist zu sichern, dass clrset immer dann verwendet wird, wenn dies erforderlich ist.

unable to open data base file

Erklaerung:
Nutzerfehler. file.db im aktuellen Verzeichnis konnte nicht zur Aktualisierung geoeffnet werden.

Moegliche Ursachen und/oder Korrekturen:
1. file.db existiert im aktuellen Verzeichnis nicht.
2. Der aktuelle WEGA-Nutzer hat fuer file.db kein Lese/Schreibrecht. Es ist entweder der Modus der Datei zu aendern oder es hat ein Login als Nutzer mit Lese/Schreibrecht fuer die Datei zu erfolgen.

unexpected EOF on FN file

Erklaerung:
Datenbank-Fehler oder interner Fehler. Dieser Fehler zeigt ein schwerwiegendes Problem mit der Datenwoerterbuch-Information am Anfang von file.db an.

Moegliche Ursachen und/oder Korrekturen:
1. Die Datei file.db ist durch Ablauf von 'Reconfigure Data Base' neu zu erstellen.

user error
illegal field number *gfield*
error code nnn

Erklaerung:
Nutzerfehler. Der Funktion gfield wurde eine ungueltige Feldnummer uebergeben. Der Fehlerkode (nnn) ist die ungueltige Feldnummer.

Moegliche Ursachen und/oder Korrekturen:
1. Moeglicherweise wird das Problem durch eine nicht korrekte oder ueberalterte Datei file.h verursacht. Es ist zu sichern, dass die mit dem Programm verwendete file.h die richtige ist.

2. Moeglicherweise wird eine ganzzahlige Variable als das Argument uebergeben, das einen ausserhalb der Grenzen liegenden Wert enthaelt.

3. Moeglicherweise ueberschreibt das Programm sich selbst.

```
user error
  illegal field number *pfield*
   error code nnn
```

Erklaerung:
Nutzerfehler. Der Funktion pfield wurde eine ungueltige Feldnummer uebergeben. Der Fehlerkode (nnn) ist die ungueltige Feldnummer.

Moegliche Ursachen und/oder Korrekturen:
1. Moeglicherweise wird das Problem durch eine nicht korrekte oder ueberalterte Datei file.h verursacht. Es ist zu sichern, dass die mit dem Programm verwendete file.h die richtige ist.

2. Moeglicherweise wird eine ganzzahlige Variable als das Argument uebergeben, das einen ausserhalb der Grenzen liegenden Wert enthaelt.

3. Moeglicherweise ueberschreibt das Programm sich selbst.

10.3 Host-Sprachfunktionen von WEGA-DATA

In diesem Abschnitt werden beschrieben:

1. Datensatz-Funktionen
2. Auswahlprozessor-Funktionen
3. Funktionen expliziter Beziehungen
4. Sekundaerindex-Funktionen
5. Gepufferte sequentielle Zugriffsfunktionen
6. Terminal E/A-Funktionen
7. Drucker E/A-Funktionen
8. Dienstprogramm-Funktionen

Datensatzorientierte WEGA-DATA-Datenbankfunktionen gestatten den Nutzerprogrammen entsprechend der Schemabeschreibung die Manipulation von Datensaetzen und Feldern innerhalb der Datensaetze. Diese Funktionen sind Funktionen der unteren Ebene, bei denen jeweils ein Datensatz bearbeitet wird.

Auswahlprozessor-Funktionen dienen der Erstellung und Manipulierung von Auswahldateien, die entweder vom Listenprozessor benutzt oder von Programmen weiterverarbeitet werden sollen. Diese Funktionen gestatten dem Nutzer, die Erstellung von Auswahldateien oder sie verwenden bereits existierende vom Listenprozessor, ENTER oder anderen Programmen erstellte Dateien fuer eine Weiterverarbeitung. Die

Datensaetze werden entweder in beliebiger Reihenfolge oder nach Bedarf sortiert zurueckgegeben, wobei Sortierschluessel unterschiedlicher Ebenen verwendet werden koennen. Die zu verwendende Zugriffsmethode wird auf der Basis der vom Nutzer spezifizierten Auswahlparameter von den Funktionen gewaehlt. Somit bilden diese Funktionen ein hoeheres Interface zu den vier verschiedenen Zugriffsmethoden von WDATA (Hashing, B-Baeume, explizite Beziehungen und gepufferter sequentieller Zugriff).

Die Routinen der hoechsten Ebene sind unisel, ufsel und selsort. Diese Routinen legen entweder eine Auswahldatei (unisel) an, rufen statt der Erstellung einer Auswahldatei fuer jeden gewaehlten Datensatz eine Nutzerfunktion auf (ufsel) oder erstellen eine Liste sortierter ausgewaehlter Datensaetze (selsort). Je nach Aufgabenstellung, waehlt ein Programm eine dieser Funktionen.

Die zweite Ebene der Routinen ist fuer den Aufbau des Auswahlprozesses da. Sie spezifizieren die bei der Auswahl zu verwendenden Felder und die der Auswahl zugrunde liegenden Kriterien. Diese Routinen sind entsitm, mtchitm, sfncitm, clrsitm, clrfitm, uqsrch und uqmtch.

Die dritte Ebene besteht aus Routinen, die der Manipulation der Auswahldateien selbst dienen, nachdem diese erste einmal erstellt sind. Diese Routinen sind opensf, closesf, frstsel, nextsel und prevsel. Einige Programme verwenden nur diese Funktionen zur Manipulation der vom Anfrageprozessor, ENTER oder einem anderen Programm angelegten Auswahldateien.

Funktionen fuer explizite Beziehungen ermoeglichen die Verfolgung der Pointer-Listen, die gefuehrt werden, wenn eine explizite Beziehung beim Entwurf vereinbart wurde. Die Wiedergewinnung zugehoeriger Datensaetze unter Verwendung von Pointern anstelle von B-Baeumen ist ausserordentlich vorteilhaft, wenn die Datensaetze nicht in einer vorgegebenen Reihenfolge wiedergewonnen werden muessen.

Funktionen hoeherer Ordnung koennen verwendet werden, um auf der Grundlage expliziter Beziehungspfade Datensaetze auszuwaehlen und zu sortieren. Diese Funktionen erarbeiten auf der Grundlage der vom Nutzer gegebenen Spezifikationen eine sortierte tag-Datei. Damit kann man Datensaetze in einer spezifischen Reihenfolge wiedergewinnen.

Sekundaerindex-Funktionen sind B-Baum Indexfunktionen, die durch eine Host-Sprache geliefert werden. Zur effektiveren Gestaltung von Datenbank-Feld-Anfragen kann fuer jedes Datenbank-Feld ein B-Baum-Index entweder in aufsteigender oder in fallender Reihenfolge angelegt werden. Man kann jedoch auch einen Index erarbeiten, wenn vom spezifizierten Feld sehr haeufig Datensaetze in einer bestimmten Reihenfolge wiedergewonnen werden muessen und man keine explizite Beziehung und unisort verwenden moechte. Im allgemei-

nen wird durch die Schaffung eines B-Baum-Indexes die Wiedergewinnungsleistung einer Datenbank verbessert, jedoch die Aktualisierung des indizierten Feldes verlangsamt.

Ein B-Baum-Suchvorgang besteht aus zwei Grundoperationen: Suchen und naechsten Datensatz holen. Die Such-Operation macht einen Datensatz zum aktuellen Datensatz. Ausserdem zeigt sie an, ob eine Eintragung gefunden wurde, die genau mit dem Suchargument uebereinstimmt. Die next-Operation wird verwendet, um den Index durchzugehen, wobei an der Position begonnen wird, die unmittelbar auf die von der Such-Operation gefundene Position folgt. Treten Felder mehrfach auf, wird durch eine Such-Operation und eine oder mehrere next-Operation(en) auf all Datensaetze mit demselben Feldwert zugegriffen.

Felder, die ein Kombinationsfeld bilden, koennen in einem Index aufgefuehrt werden; das entsprechende Kombinationsfeld jedoch nicht. WEGA-DATA verwaltet automatisch alle B-Baum-Indizes, um die Fehlerlosigkeit der Daten auch dann zu sichern, wenn mehrere Nutzer gleichzeitig Datensaetze aktualisieren und wiedergewinnen. Indizes, die fuer Zeichenketten-Felder mit einer Laenge groesser als 80 aufgebaut wurden, lassen alle Zeichen nach der 80. Position unberuecksichtigt.

Gepufferte sequenzielle Zugriffsfunktionen gestatten das Durchmustern einer Datei unter Verwendung von 'raw'-E/A und grosser Puffer, wenn es keine andere Moeglichkeit gibt, auf die Datensaetze zuzugreifen (z.B. Hash, B-Baum oder explizite Beziehung). Das Verfahren gestattet eine sehr schnelle Verarbeitung aller Datensaetze in einer Datei, so dass es auch dann empfehlenswert ist, wenn zwar andere Zugriffsmethoden moeglich waeren, aber jeder einzelne Datensatz untersucht werden muss. Die Geschwindigkeit ist darauf zurueckzufuehren, dass beim Lesen in sehr grossen Bloecken die Plattenkopfbewegung nur sehr minimal ist, und bekanntlich ist bei der Auswahl von Datensaetzen in den meisten Faellen die Suchzeit eine kritische Groesse.

Da 'raw'-E/A verwendet wird, sind diese Funktionen nur fuer Nur-Lese-Vorgaenge geeignet, z.B. fuer die Auflistung oder Auswahl von Datensaetzen. Die Daten werden in den programmeigenen Puffer gelesen, und wuerde man in diesem Puffer eine Aktualisierung der Daten vornehmen, koennen Widersprueche zu den Daten im WEGA-Haupspeicher entstehen. Daher sind keine Aktualisierungsfunktionen moeglich. Wurde jedoch ein Datensatz unter Verwendung von bsetloc, bseqacc oder bfaccess zum aktuellen Datensatz gemacht, kann man die Standardfunktionen fuer die Aktualisierung von Datensaetzen verwenden.

Terminal E/A-Funktionen zeigen Daten und Prompter auf dem Bildschirm an und holen die vom Nutzer vorgenommenen Eingaben. Es gibt Funktionen zweier verschiedener Ebenen - hoehere Funktionen, die eine Kopplung zu unter Verwendung von

SFORM definierten Bildmasken herstellen und niederere Funktionen, die fuer den Dialog mit den Terminals xy-Koordinaten und literalen Text verwenden. Diese Funktionen verwenden alle die termcap-Bibliotheksfunktionen fuer die Ausfuehrung terminalunabhaengiger E/A.

Fuer die beschriebenen SFORM-Funktionen sind ssfld und esfld Bildmasken-Feldnamen, die in der vom Programm 'Process Screen' erzeugten Datei .h definiert sind (Abschnitt 4.3). ssfld ist die erste Bildmasken-Feldnummer, und sie muss vor der letzten, esfld, stehen.

Drucker E/A-Funktionen uebergeben Datenbank-Informationen an den Drucker. Diese Funktionen ermoeglichen die nutzerspezifische Formatierung von Reporten.

Dienstprogramm-Funktionen koennen zum Schreiben von Programmen verwendet werden. Sie umfassen Funktionen zur Zeichenmanipulation, Terminalsteuerung und Geraeteverriegelung.

Alphabetischer Index fuer WEGA-DATA Host-Sprach-Funktionen

Funktion	Beschreibung
acckey	Zugriff auf Datenbank-Datensatz mittel Primaer-schluessel
accsfld	Zugriff auf geschuetztes Datenbank-Feld ermoeglichen
addrec	Einfuegen eines neuen Datensatzes in die Datenbank
bfaccess	gepufferte Version von faccess
bgfield	gepufferte Version von gfield
bseqacc	gepufferte Version von seqacc
bsetloc	gepufferte Version von setloc
btnext	unter Verwendung eines B-Baum-Index naechsten Datensatz holen
btsrch	B-Baum-Index eines Feldes durchsuchen
cfill	Zeichenkette mit einem Zeichen fuellen
cleancrt	Vordergrund loeschen
clearscr	Bildschirm loeschen
closbt	B-Baum-Index schliessen
closedb	Datenbank-Datei schliessen
closesf	Auswahldatei schliessen
clr_crt	Terminal loeschen
clrfitm	Auswahlelement der Funktion loeschen
clrset	vorher gekennzeichneten Satz loeschen
clrsitm	Auswahl-Datenelement loeschen
dbproc	Tochterprozess der Datenbank initialisieren
delete	Datensatz aus Datenbank loeschen
dsply	Daten fuer Bildmasken-Felder anzeigen
endtrans	Aktualisierungslauf beenden
entsitm	Auswahl-Datenelement eingeben
era_ln	eine Zeile auf dem Terminal loeschen
erasprmp	Daten fuer Bildmasken-Felder loeschen
faccess	unter Verwendung des spezifizierten Feldes auf zugehoerigen Datensatz zugreifen
fldesc	Beschreibung des Datenbank-Feldes
fldidxd	feststellen, ob ein Feld in einem Index verzeichnet ist
flush	Druck-Puffer ausgeben
frstsel	zuerst ausgewaehlten Datensatz holen
gdata	Felddaten vom Bildschirm holen
gfield	ein Feld von der Datenbank in einen Puffer holen
glob	eine Zeichenkette mit einer Metazeichenmaske vergleichen
gtube	ein Feld vom Bildschirm in einen Puffer eingeben
inbuf	ein Bildmasken-Feld in einen Puffer eingeben
iniubuf	den Lesepuffer fuer bsequacc (und andere) initialisieren
input	ein Feld eingeben
ivcmp	zwei Felder vergleichen
kdate	Julianisches Datum konvertieren
kday	ins Julianische Datum-Format konvertieren
keybrd	Tastatur des Terminals sperren oder freigeben
lastchr	letztes Zeichen einer Zeichenkette suchen

len	Laenge einer Zeichenkette bestimmen
loadscr	Bildmaske anzeigen, Parameter laden
loc	Datensatzadresse suchen
lock_r	Geraet verriegeln
lockrec	aktuellen Datensatz verriegeln
makeset	Identifizieren einer Menge zusammenhaengender Datensaetze
mtchitm	fuer unisel ein Suchfeld eingeben
mv_cur	Cursorposition setzen
nextrec	den naechsten Datensatz im Satz holen
nextsel	naechsten Datensatz der Auswahlmenge holen
oblank	Druck-Puffer initialisieren
obuf	Ausgabe von einem Puffer an den Drucker
odata	Ausgabe eines Datenbank-Feldes an den Drucker
opendb	Oeffnen einer Datenbank-Datei
opensf	Oeffnen einer Auswahldatei
opnbts	Oeffnen eines B-Baum-Index zum Suchen
oprf	eine Pipe fuer lpr aufbauen
outbuf	Puffer in einem Bildmasken-Feld anzeigen
output	ein Feld anzeigen
pdata	Daten aus der Datenbank auf dem Bildschirm anzeigen
pfield	ein Feld in der Datenbank aktualisieren
pform	spezielle Masken ausgeben
prevrec	vorhergehenden Datensatz in einem Satz holen
prevsel	zuletzt ausgewaehlten Datensatz holen
priamd	nach Betriebsmodus fragen
prmp	eine Zeichenkette mit geringer Intensitaet auf dem Bildschirm anzeigen
prmpf	eine Zeichenkette mit hoher Intensitaet auf dem Bildschirm anzeigen
prmprv	eine Zeichenkette invers auf dem Bildschirm anzeigen
prstr	eine Zeichenkette an Druck-Puffer ausgeben
prtmsg	Meldung anzeigen und auf Quittierung warten
ptct_crt	Status des Terminal-Schutzmodus' aendern
ptct_wrt	Schreibschutzstatus des Terminals aendern
ptube	ein Datenbank-Feld aus einem Puffer auf den Bildschirm schreiben
qmove	Kopieren einer Zeichenkette in eine andere
samerec	den aktuellen Datensatz einer ausgewaehlten Menge wiederherstellen
scomp	zwei Zeichenketten miteinander vergleichen
selsort	Auswahldatei sortieren
seqacc	sequentieller Zugriff auf alle Datensaetze einer Datei
setcook	Terminal auf cooked-Modus setzen
setloc	aktuellen Datensatz setzen
setraw	Terminal auf raw-Modus setzen
setsize	Anzahl der Datensaetze einer Menge bestimmen
sfldesc	Beschreibung des Bildmasken-Feldes
sfncitm	Eingabe eines Such-Feldes
strstrec	den ersten Datensatz einer sortierten Menge holen
slastrec	den letzten Datensatz einer sortierten Menge holen

snextrec	den naechsten Datensatz einer sortierten Menge holen
sprevrec	den vorhergehenden Datensatz einer sortierten Menge holen
startrans	Aktualisierungslauf beginnen
strcmp	zwei Zeichenketten miteinander vergleichen
ufsel	unisel-Funktionsauswahl
ulockrec	aktuellen Datensatz freigeben
unisel	Datenbank-Datensaetze auswaehlen
unisort	Datensaetze auswaehlen und sortieren
unlock_r	ein durch lock_r reserviertes Geraet freigeben
uqmtch	Such-Feld fuer schnelles unisel
uqsrch	Auswahlelement fuer schnelles unisel
valchar	Zeichen durch Vergleich mit einem Zeichensatz auf Gueltigkeit ueberpruefen
valstr	Zeichenkette durch Vergleich mit Satz von Zeichenketten auf Gueltigkeit ueberpruefen
yorn	Prompter fuer y/n Antwort

ACCKEY

NAME
acckey - Zugriff auf Datenbank-Datensatz mittels Primaerschluessel

SYNTAX
```
acckey (rnum, key)
char *key;
```

BESCHREIBUNG
acckey sucht und findet einen Datensatz in der Datenbank unter Verwendung des Schluessels. rnum spezifiziert den Typ des zu suchenden Datensatzes. key ist ein Pointer auf einen den Schluessel des Datensatzes enthaltenden Puffer. Da angenommen wird, dass der Puffer die Laenge des Schluessels hat, wird kein Terminator benoetigt.

Anfangs wurde diese Funktion als access bezeichnet. Der Name wurde geaendert, um eine Ueberschneidung mit dem WEGA-Systemaufruf access(2) zu vermeiden. Zur Gewaehrleistung der Kompatibilitaet steht die alte access-Funktion im Archiv libcomp.a bereit. Beabsichtigt der Nutzer auch in Zukunft access zu verwenden, ist diese Bibliothek in die Kommandodateien des Ladeprogrammes aufzunehmen.

RUECKGABEWERTE
```
 0 - Datensatz wurde gefunden und ist der aktuelle
-1 - kein Schluessel fuer diesen Datensatztyp
-2 - kein Datensatz dieses Typs mit diesem Schluessel
```

FEHLER
Wird mit einem Kombinationsfeld auf einen Datensatz zugegriffen, muss eine Struktur verwendet werden, deren Format gleich dem internen Format des Schluesselfeldes ist. Da jedoch alle Felder an Wortgrenzen beginnen, muessen Zeichenfelder ungerader Laenge aufgefuellt werden. Bei Zugriff auf den Datensatz muss dasselbe Auffuellzeichen verwendet werden oder der Datensatz ist unauffindbar. ENTER und der Datenbank-Testtreiber verwenden Null zum Auffuellen.

SIEHE AUCH
addrec, delete

ACCSFLD

NAME
accsfld - Zugriff auf ein geschuetztes Datenbank-Feld ermoeglichen

SYNTAX
```
accsfld (fnum, pass, priv)
```

```
    int fnum;
    char *pass,priv;
```

BESCHREIBUNG

Diese Funktion wird verwendet, um einem Programm den Zugriff auf ein durch 'Field Level Security' geschuetztes Datenbank-Feld zu ermoeglichen (Abschnitt 3.3.1). Zum Entriegeln des Feldes wird ein Passwort benoetigt. Wird das richtige Passwort verwendet, werden alle Felder derselben Gruppe fuer den Zugriff eroeffnet. Der Nutzer kann spezifizieren, ob Leserecht oder Lese- und Schreibrecht erforderlich sind. Moeglicherweise ist dafuer ein anderes Passwort erforderlich.

ARGUMENTE

fnum - die Feldnummer
pass - Pointer auf ein auf Null endendes Passwort
priv - r fuer Leserecht, w fuer Lese/Schreibrecht

RUCKGABEWERTE

-1 - Ungueltiges Passwort
 0 - Zugriff auf Feld ist nicht moeglich

ADDREC

NAME

addrec - Einfuegen eines neuen Datensatzes in die Datenbank

SYNTAX

```
    addrec (rnum, key)
    char *key;
```

oder

```
    addrec (rnum)
```

BESCHREIBUNG

addrec fuegt mit dem gegebenen nur einmal auftretenden Schluessel key einen neuen Datensatz in die Datenbank ein. Ist der Schluessel wirklich nur einmalig vorhanden, wird der Datensatz hinzugefuegt und zum aktuellen gemacht. Ist der Schluessel noch einmal vorhanden, wird ein Fehler zurueckgegeben und der Datensatz, der den Schluessel bereits besitzt, wird zum aktuellen Datensatz. Hat der Datensatztyp keinen Schluessel wird das zweite Argument nicht benoetigt, und der Datensatz wird in jedem Fall eingefuegt.

RUCKGABEWERTE

 0 - der Datensatz wurde hinzugefuegt und ist nun der aktuelle Datensatz
-1 - Der Schluessel war bereits vorhanden und der Datensatz, der diesen Schluessel bereits hatt, wird zum aktuellen Datensatz.
-2 - Der Schluessel bezieht sich auf einen nichtexistenten Datensatz. Entspricht dem Rueckgabewert -3

in pfield.
-3 - Fuer ein Feld im Datensatz ist Schreibzugriff nicht zulaessig (siehe accsfld).

FEHLER
Wird mit einem Kombinationsfeld auf einen Datensatz zugegriffen, muss eine Struktur verwendet werden, deren Format gleich dem internen Format des Schluesselfeldes ist. Da jedoch alle Felder an Wortgrenzen beginnen, muessen Zeichenfelder ungerader Laenge aufgefuellt werden. Bei Zugriff auf den Datensatz muss dasselbe Auffuellzeichen verwendet werden oder der Datensatz ist unauffindbar. ENTER und der Datenbank-Testtreiber verwenden Null zum Auffuellen.

SIEHE AUCH
acckey, delete

BFACCESS

NAME
bfaccess - gepufferte Version von faccess

SYNTAX
bfaccess (rnum, feld)

BESCHREIBUNG
Diese Funktion wird zur Wiedergewinnung eines zugehoerigen Datensatzes verwendet, wobei der Wert eines spezifizierten Feldes verwendet wird. rnum ist der Datensatz, auf den zugegriffen werden soll und feld ist das Feld, dem der Wert entnommen werden soll. Diese Version verwendet eine gepufferte Konfiguration, daher muss der Datensatz, in dem feld enthalten ist, unter Verwendung von bseqacc zum aktuellen Datensatz gemacht werden. Der Rest der Beschreibung von faccess hier trifft zu.

RUECKGABEWERTE
0 - Zugriff erfolgt, Datensatz vom Typ rnum ist der aktuelle Datensatz
1 - Zugriff konnte nicht erfolgen

SIEHE AUCH
faccess, bseqacc, bgfield, iniubuf, bsetloc

BGFIELD

NAME
bgfield - gepufferte Version von gfield

SYNTAX
```
bgfield (feld, buf)
char *buf;
```

BESCHREIBUNG
Diese Funktion gewinnt ein Feld aus der Datenbank zurueck und speichert es in einem Puffer. Diese Version verwendet eine gepufferte Konfiguration und kann erst aufgerufen werden, nachdem der Datensatz unter Verwendung von bseqacc zum aktuellen Datensatz gemacht wurde.

RUECKGABEWERTE
keine

SIEHE AUCH
gfield, bseqacc, bfaccess, iniubuf, bsetloc

BSEQACC

NAME
bseqacc - gepufferte Version von seqacc

SYNTAX
bseqacc (rnum, richtung)

BESCHREIBUNG
Diese Funktion wird verwendet, um alle Datensaetze in einer Datei in der Reihenfolge wiederzugewinnen, in der sie gespeichert wurden. Diese Version verwendet eine gepufferte Konfiguration, um die Geschwindigkeit der Wiedergewinnung zu erhoehen. rnum ist der wiederzugewinnende Recordtyp, richtung wird aus folgender Menge ausgewaehlt:

first - ersten Datensatz in der Datei holen
next - den naechsten Datensatz in der Datei holen
last - den letzten Datensatz in der Datei holen
prev - den vorherigen Datensatz in der Datei holen

RUECKGABEWERTE
0 - der spezifizierte Datensatz ist der aktuelle
-1 - der spezifiierte Datensatz existiert nicht

SIEHE AUCH
seqacc, bgfield, bfaccess, iniubuf, bseqacc, bsetloc

BSETLOC

NAME
bsetloc - gepufferte Version von setloc

SYNTAX
bsetloc (rnum, loc)
long loc;

BESCHREIBUNG
Diese Funktion gestattet dem Nutzer das Setzen des aktuellen Datensatzes fuer einen gegebenen Datensatz-

typ. Diese Version wird mit den gepufferten Routinen verwendet. Erfolgt bei Verwendung von bseqacc kein Zugriff auf einen Datensatz, muss diese Funktion aufgerufen werden, bevor bgfield und bfaccess aufgerufen werden. rnum ist der Datensatztyp und loc ist die Position des Datensatzes, die durch Aufruf von loc bereits vorher gewonnen wurde.

RUECKGABEWERTE
keine

SIEHE AUCH
bgfield, bseqacc, bfaccess, iniubuf, loc

BTNEXT

NAME
btnext - unter Verwendung eines B-Baum-Index naechsten Datensatz holen

SYNTAX
btnext (feld)

BESCHREIBUNG
btnext holt das als naechstes auftretende Feld im B-Baum-Index des spezifizierten feldes und macht den Datensatz zum aktuellen Datensatz. Der Index muss von der Funktion opnbts fuer den Suchvorgang geoeffnet worden sein. Die Funktion btsrch muss vor Aufruf dieser Funktion zur Durchsuchung des Index' angewandt worden sein.

RUECKGABEWERTE
-3 - der Index des spezifizierten Feldes ist nicht fuer den Suchvorgang geoeffnet
-2 - die Funktion btsrch wurde nicht vor dieser Funktion aufgerufen
-1 - Ende des Index wurde erreicht. Es wurde kein Datensatz zum aktuellen Datensatz gemacht
0 - der Datensatz ist der aktuelle Datensatz

FEHLER
Da diese Funktion eine look-ahaed-Pufferung verwendet, kann man moeglicherweise Datensaetze, die waehrend der Anwendung der Funktion in die Datenbank eingefuegt werden, nicht finden.

SIEHE AUCH
btsrch, closbt, fldidxd, opnbts

BTSRCH

NAME
btsrch - B-Baum-Index eines Feldes durchsuchen

SYNTAX
```
btsrch (feld, buf)
char *buf;
```

BESCHREIBUNG
btsrch durchsucht den vorher von der Funktion opnbts zum Durchsuchen geoffneten B-Baum-Index eines Feldes und macht einen Datensatz zum aktuellen Datensatz. buf ist die Adresse, an der feld in interner Form auftritt. Diese Adresse ist das Such-Argument. Suchziel ist die Stelle im Index, an der das Feld auftritt und die logisch gleich nach dem Such-Argument kommt oder gleich dem Such-Argument ist. Nachdem btsrch den Index durchsucht hat, kann die Funktion btnext verwendet werden, um den Index zu durchlaufen.

RUECKGABEWERTE
-2 - Der B-Baum-Index des Feldes wurde nicht zum Durchsuchen durch die Funktion opnbts geoeffnet
-1 - Das Such-Argument kommt logisch nach allen im Index auftretenden Feldern. Es wurde kein Datensatz zum aktuellen Datensatz.
0 - Das Feld wurde an einer Stelle gefunden, die nicht mit dem Such-Argument uebereinstimmt. Der Datensatz ist der aktuelle Datensatz.
1 - Das Feld wurde an einer Stelle gefunden, die mit dem Suchargument uebereinstimmt. Der Datensatz ist der aktuelle Datensatz.

SIEHE AUCH
btnext, closbt, fldidxd, opnbts

CFILL

NAME
cfill - Zeichenkette mit einem Zeichen fuellen

SYNTAX
```
cfill(c, str, len)
char c, *str;
int len;
```

BESCHREIBUNG
Diese Funktion fuellt ein Feld mit einem spezifizierten Zeichen. Das erste Argument (c) ist das Zeichen, mit dem das Feld zu fuellen ist. Das zweite Argument (str) ist ein Pointer auf das zu fuellende Feld. Das dritte Argument (len) ist die Anzahl der zu fuellenden Feldelemente.

RUECKGABEWERTE
keine

CLEANCRT

NAME
cleancrt - Vordergrund loeschen

SYNTAX
cleancrt()

BESCHREIBUNG
Diese Funktion loescht den Bildschirm-Vordergrund. Es gibt zwei verschiedene Methoden; eine fuer Terminals mit dieser Hardware-Option und eine fuer Terminals ohne eine solche Moeglichkeit. Bei Terminals, bei denen das Loeschen des Vordergrundes moeglich ist, loescht diese Funktion selbigen, und nur die Bildmaskenfeld-Prompter verbleiben. Bei anderen Terminals ruft die Funktion fuer jedes Bildmaskenfeld erasprmp auf. Alle anderen auf dem Bildschirm befindlichen Daten muessen getrennt geloescht werden.

RUECKGABEWERTE
keine

FEHLER
Unerwartete Ergebnisse koennen auftreten, wenn nicht loadscr nicht zum Laden der Bildmaske verwendet wurde.

SIEHE AUCH
clearscr

CLEARSCR

NAME
clearscr - Bildschirm loeschen

SYNTAX
clearscr()

BESCHREIBUNG
Diese Funktion loescht den Bildschirm ab Zeile 3. Das heisst, die vom Menue-Handler angezeigte Ueberschrift bleibt erhalten, waehrend der Rest des Bildschirms geloescht wird. Wird diese Funktion in Verbindung mit loadscr verwendet, koennen Bildmasken mit mehreren Seiten verwendet werden.

RUECKGABEWERTE
keine

SIEHE AUCH
cleancrt

CLOSBT

NAME
closbt - B-Baum-Index schliessen

SYNTAX
closbt (feld)

BESCHREIBUNG
closbt schliesst den B-Baum-Index des spezifizierten Feldes, der von der Funktion opnbts zum Durchsuchen geoeffnet worden war.

RUECKGABEWERTE
-1 - Der Index des spezifizierten Feldes (sofern er existiert) war nicht zum Durchsuchen geoeffnet.
0 - Der Index des spezifizierten Feldes ist geschlossen

SIEHE AUCH
btnext, btsrch, fldidxd, opnbts

CLOSEDB

NAME
closedb - Datenbank-Datei schliessen

SYNTAX
closedb()

BESCHREIBUNG
Diese Funktion schliesst eine von opendb geoeffnete Datenbank-Datei.

RUECKGABEWERTE
keine

SIEHE AUCH
opendb

CLOSESF

NAME
closesf - Auswahldatei schliessen

SYNTAX
#include "unisel.h"
close (sf)
SELFILE *sf;

BESCHREIBUNG
Diese Funktion wird verwendet, um eine Auswahldatei zu schliessen, die unter Verwendung von opensf geoeffnet wurde. sf ist der vom Aufruf opensf zurueckgegebene

Pointer.

RUECKGABEWERTE
keine

SIEHE AUCH
opensf, frstsel, nextsel, prevsel

CLR_CRT

NAME
clr_crt - Terminal loeschen

SYNTAX
```
clr_crt(fd, flg)
int fd;
char flg;
```

BESCHREIBUNG
Diese Funktion wird zum Loeschen des Terminals verwendet. Das erste Argument (fd) ist der Ausgabedatei-Deskriptor. Das zweite Argument (fig) muss ein a sein, wenn sowohl Vorder- als auch Hintergrund zu loeschen sind. Andernfalls muss es eine Leerstelle sein.

RUECKGABEWERTE
keine

CLRFITM

NAME
clrfitm - Auswahlelement der Funktion loeschen

SYNTAX
```
clrfitm (sfnc)
int (*sfnc) ();
```

BESCHREIBUNG
Diese Funktion wird verwendet, um ein Funktions-Auswahldatenelement sfnc aus der von unisel verwendeten Auswahltabelle zu loeschen. sfnc ist die Adresse der Funktion und wird fuer die Kennzeichnung des zu loeschenden Datenelementes verwendet.

RUECKGABEWERTE
keine

SIEHE AUCH
sfncitm, clrsitm, entsitm, mtchitm, uqmtch, uqsrch

CLRSET

NAME
clrset - vorher gekennzeichneten Satz loeschen

SYNTAX
clrset (feld, rfeld)

BESCHREIBUNG
clrset wird verwendet, um einen Satz zu loeschen, der vorher durch einen Aufruf von makeset definiert wurde. Die Kombination feld, rfeld muss mit einer existierenden Eintragung in der Tabelle der aktiven Saetze uebereinstimmen. Die maximale Anzahl der aktiven Saetze ist gegenwaertig 10.

RUECKGABEWERTE
0 - der Satz wurde geloescht
-1 - ein solcher Satz war nicht aktiv

SIEHE AUCH
faccess, makeset, nextrec, prevrec, samerec, setsize, sfrstrec, slastrec, snextrec, sprevrec, unisort

CLRSITM

NAME
clrsitm - Auswahl-Datenelement loeschen

SYNTAX
clrsitm (feld, rfeld)

BESCHREIBUNG
Diese Funktion wird verwendet, um ein Auswahl-Datenelement aus der von unisel verwendeten Auswahltabelle zu loeschen. Die Argumente werden verwendet, um zu kennzeichnen, welches Element zu loeschen ist, da die Kombination in der Tabelle nur einmal vorkommen darf.

ARGUMENTE
feld - Das Feld, auf dem die Auswahl basiert.
rfeld - Das Feld, auf das Bezug genommen wird und das zum Vergleich der Datenelemente benutzt wird. rfeld muss literal Null sein, wenn auf kein Feld Bezug genommen werden soll.

RUECKGABEWERTE
keine

SIEHE AUCH
entsitm, mtchitm, unisel, clrfitm, sfncitm, uqmtch, uqsrch

DBPROC

NAME
dbroc - Tochterprozess der Datenbank initialisieren

SYNTAX
dbproc (dbname, opt)

BESCHREIBUNG
Diese Funktion wird zum Initialisieren des Tochterprozesses der Datenbank verwendet, der die tatsaechlichen Dantenbank-Funktionen ausfuehren wird. Das fuehrt dazu, dass bei einigen Datenbank-Funktionen, auf die das Nutzerprogramm Bezug nimmt, der Umfang des Kodes verringert wird - d.h. man kann umfangreichere Programme schreiben. Beim Laden eines dbproc aufrufenden Programms muss anstelle von uld uuld benutzt werden. Der Unterschied besteht in der Hauptsache darin, dass sich uuld auf die Bibliothek upintf.a bezieht, die die Datenbank-Interfacefunktionen enthaelt, die mit dem Tochterprozess kommunizieren. Neben dem Aufruf von dbproc sind keine weiteren Quellkodeaenderungen im Nutzerprogramm erforderlich.

```
beis()
{
  dbproc("file.db", 0);
}
```

ARGUMENTE
dbname - Zeichenpointer auf den Datenbank-Dateinamen. Dieser ist normalerweise file.db.
opt - Siehe Funktion opendb.

RUECKGABEWERTE
keine

SIEHE AUCH
opendb

DELETE

NAME
delete - Datensatz aus Datenbank loeschen

SYNTAX
delete (rnum)

BESCHREIBUNG
Diese Funktion loescht einen Datensatz aus der Datenbasis. Der aktuelle Datensatz wird geloescht und es sind keine weiteren Zugriffe auf diesen Datensatz durch die Funktionsaufrufe acckey, nextrec, prevrec oder samerec mehr moeglich. Gibt es irgendwelche mit diesem Datensatz in Beziehung stehende Datensaetze, ist diese

Funktion nicht zulaessig. Alle in Beziehung stehenden Datensaezte muessen geloescht werden, bevor der Datensatz geloescht werden kann.

Es ist nicht vorhersehbar, welcher Datensatz nach Loeschen des aktuellen Datensatzes zum aktuellen wird. Kann das Loeschen nicht vollzogen werden, bleibt der aktuelle Datensatz erhalten. Es ist zu beachten, dass der durch das Loeschen eines Datensatzes frei werdender Raum spaeter wieder zum Einfuegen eines Datensatzes desselben Datensatztyps verwendet werden kann.

RUECKGABEWERTE
 0 - Datensatz wurde geloescht
-1 - Datensatz wurde nicht geloescht
-2 - Schreibzugriff fuer ein Feld im Datensatz nicht zulaessig (siehe accsfld)
-3 - Datensatz ist durch einen anderen Prozess verriegelt (siehe lockrec)

SIEHE AUCH
addrec, acckey

DSPLY

NAME
dsply - Daten fuer Bildmasken-Felder anzeigen

SYNTAX
dsply(ssfld, esfld)

BESCHREIBUNG
Mit dieser Funktion werden die zu einem Satz von Bildmaskenfeldern zugehoerigen Daten angezeigt. Die Funktion beginnt mit ssfld und endet mit esfld und durchlaeuft den Bildschirm von oben nach unten und ruft fuer jedes Feld output auf. Es ist zu beachten, dass mit dieser Funktion immer Daten aus der Datenbank angezeigt werden, deshalb muessen die angeforderten Datensaetze aktuelle sein, um Fehler auszuschliessen.

RUECKGABEWERTE
keine

ENDTRANS

NAME
endtrans - Aktualisierungslauf beenden

SYNTAX
endtrans()

BESCHREIBUNG
Diese Funktion wird verwendet, um die Datenbank am Ende

eines Aktualisierungslaufes zu entriegeln. Sie ist das Gegenteil von startrans.

In Abhaengigkeit davon, ob ein WEGA-Systemaufruf zum Dateiverriegeln vorhanden ist oder nicht, werden zwei verschiedene Implementierungsmethoden verwendet. Ist ein Systemaufruf zum Verriegeln vorhanden, wird mit dem Systemaufruf Byte 0 der Datei lockfile (ddlockfile, wenn wdata.db entriegelt wird) entriegelt. Ist kein Systemaufruf zum Verriegeln vorhanden wird stattdessen die WEGA-DATA Funktion unlock_r verwendet (siehe Beschreibung von unlock_r).

Die Verriegelungsdateien werden im Arbeitsverzeichnis des Nutzers angelegt. Normalerweise ist das Verzeichnis bin das Arbeitsverzeichnis, wenn aber $DBPATH gesetzt wird, ist das Verzeichnis, auf das $DBPATH zeigt, das Arbeitsverzeichnis. Das bedeutet, dass fuer jeden Anwendungsfall ein einziges Arbeitsverzeichnis vorhanden sein muss, das von allen verwendet wird - entweder das Verzeichnis bin oder fuer alle derselbe $DBPATH. Wird diese Regel nicht beachtet, ist die Steuerung der konkurierenden Arbeit nicht gewaehrleistet und die Integritaet der Datenbank wird zerstoert.

RUECKGABEWERTE
keine

SIEHE AUCH
startrans, unlock_r

ENTSITM

NAME
entsitm - Auswahl-Datenelement eingeben

SYNTAX
```
#include "unisel.h"
entsitm (feld, val1, val2, modus)
int modus;
char *val1, *val2;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um ein Auswahl-Datenelement in die von unisel verwendete Auswahltabelle einzutragen. Befindet sich das Datenelement bereits in der Tabelle (feld kommt nur einmal vor), wird der Versuch gemacht, das aktuelle Datenelement mit dem neuen Datenelement zu kombinieren und dadurch einen Auswahlbereich zu erstellen. Gelingt dieser Versuch nicht, wird das Datenelement ersetzt.

ARGUMENTE
feld - Das Feld auf dem die Auswahl basiert
val1 - Die Adresse eines Puffers, der den Wert des bei

der Auswahl verwendeten Feldes enthaelt. Der Typ des Wertes muss mit dem internen Feldtyp uebereinstimmen

val2 - Die Adresse eines Puffers, der den zweiten Wert enthaelt, wenn die Auswahl ein Bereich ist. Andernfalls ist val2 auf 0 gesetzt. Der Modus fuer einen Bereich muss EQ oder NOT sein. Fuer Zeichenkettenfelder sind Bereiche nicht moeglich, deshalb sind Metazeichen zu verwenden.

modus - Modus der Auswahl: GT, GTE, LT, LTE, EQ, NOT. Diese Modi sind in unisel.h definiert. Zeichenketten-Felder koennen nur mit EQ oder NOT verwendet werden.

RUECKGABEWERTE

0 - Normale Beendigung
-1 - Das Feld ist ungueltig
-2 - Das Argument val1 ist nicht gesetzt
-3 - Der Modus ist unzulaessig
-4 - Die Auswahltabelle ist voll

SIEHE AUCH

mitchitm, sfncitm, clrsitm, clrfitm, uqmtch, uqsrch

ERAS_LN

NAME

eras_ln - eine Zeile auf dem Terminal loeschen

SYNTAX

```
eras_ln (fd)
int fd;
```

BESCHREIBUNG

Diese Funktion gibt das Kommando Zeile loeschen an das Terminal. Argument ist der Dateideskriptor des Terminals.

RUECKGABEWERTE

keine

ERASPRMP

NAME

erasprmp - Daten fuer Bildmasken-Felder loeschen

SYNTAX

```
erasprmp (ssfld, esfld)
```

BESCHREIBUNG

Diese Funktion loescht den Datenteil eines Satzes von Bildmaskenfeldern vom Bildschirm. Die Funktion beginnt mit ssfld und endet mit esfld und durchlaeuft den Bildschirm von oben nach unten, wobei die maximale

Laenge aller Bildmasken-Felder mit Leerstellen ueberschrieben wird.

RUECKGABEWERTE
keine

FACCESS

NAME
faccess - unter Verwendung des spezifizierten Feldes auf zugehoerigen Datensatz zugreifen

SYNTAX
faccess (rnum, feld)

BESCHREIBUNG
faccess wird zur Rueckgewinnung eines zugehoerigen Datensatzes unter Verwendung des Wertes in einem spezifizierten Feld verwendet. rnum ist der Datensatz, auf den zugegriffen werden soll und feld ist das den Wert liefernde Feld. Der das Feld enthaltende Datensatz muss der aktuelle sein oder das Programm laeuft fehlerhaft ab. Handelt es sich bei dem Feld um ein Feld, das eine explizite Beziehung zum spezifizierten Datensatz hat, wird der Pointer verwendet. Ansonsten wird mit dem Schluessel auf den Datensatz zugegriffen, wobei der Wert im Feld benutzt wird. Deshalb muss das Feld denselben Typ und dieselbe Laenge haben wie der Schluessel des spezifizierten Datensatzes.

RUECKGABEWERTE
0 - Der Zugriff erfolgte, Datensatz vom Typ rnum ist aktueller Datensatz
-1 - Der Zugriff erfolgte nicht, der Datensatz ist nicht der aktuelle

SIEHE AUCH
clrset, makeset, nextrec, prevrec, samerec, setsize, sfrstrec, slastrec, snextrec, sprevrec, unisort

FLDESC

NAME
fldesc - Beschreibung des Datenbank-Feldes

SYNTAX
```
#include "bdtypes.h"
"include "fdesc.h"
fldesc (feld, fdsc)
int feld;
FLDESC *fdsc;
```

BESCHREIBUNG
Es folgt die Definition der FLDESC-Struktur:

```
#define FLDESC struct fdesc
struct fdesc  {
int  f_rec;       /* record */
int  f_typ;       /* field type */
int  f_len;       /* display length */
int  f_par;       /* comb field if a subfield */
int  f_rprec;     /* referenced record if a relation */
int  f_rpfld;     /* referenced field if a relation */
};
```

Die moeglichen, in dbtypes.h definierten Feldtypen:

INT, LONG, DATE, AMT, STRING, HAMT, HR, FLT, COMB

Diese Funktion gibt die Attribute eines Datenbank-Feldes im Strukturpointer fdsc zurueck.

ARGUMENTE

feld - Das Datenbank-Feld
fdsc - Die Adresse einer FLDESC-Struktur

RUECKGABEWERTE

0 - ungueltiges Feld
1 - normale Rueckkehr

SIEHE AUCH

sfldesc

FLDIDXD

NAME

fldidxd - feststellen, ob ein Feld in einem Index verzeichnet ist

SYNTAX

fldidxd (feld)

BESCHREIBUNG

fldidxd bestimmt, ob ein Feld in einem B-Baum-Index indiziert ist oder nicht und, falls es indiziert ist, ob der Index von aufsteigender oder fallender Ordnung ist.

RUECKGABEWERTE

0 - Feld ist nicht in einem B-Baum-Index verzeichnet
1 - Feld ist in aufsteigender Ordnung indiziert
2 - Feld ist in fallender Ordnung indiziert

SIEHE AUCH

btnext, btsrch, closbt, opnbts

FLUSH

NAME
flush - Druck-Puffer ausgeben

SYNTAX
```
flush (fd)
int fd;
```

BESCHREIBUNG
Diese Funktion gibt den Druckpuffer an den vom Nutzer spezifizierten Dateideskriptor aus. Der Druck-Puffer muss initialisiert worden sein (siehe oblank) und dann unter Verwendung von odata, obuf, prstr usw. gefuellt worden sein. Das einzige Argument ist der Dateideskriptor, der als Ergebnis eines vorher erfolgten Aufrufes open oder oprf zurueckgegeben worden sein muss.

Nach Aufruf der Funktion wird der Druckpuffer geloescht, so dass weitere Aufrufe von flush Leerzeilen erzeugen. Zu bemerken ist auch, dass nur die Anzahl der auf einer Zeile formatierten Zeichen tatsaechlich ausgegeben wird, es gibt keine nachgestellten Leerzeichen.

RUECKGABEWERTE
keine

SIEHE AUCH
oblank, obuf, odata, pform, prstr

FRSTSEL

NAME
frstsel - zuerst ausgewaehlten Datensatz holen

SYNTAX
```
#include "unisel.h"
frstsel (sf)
SELFILE *sf;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um den als ersten ausgewaehlten Datensatz zum aktuellen Datensatz zu machen. sf ist eine Adresse der Struktur SELFILE, die im Ergebnis eines vorangegangenen Aufrufs opensf zurueckgegeben wurde.

RUECKGABEWERTE
1 - Der erste ausgewaehlte Datensatz ist der aktuelle
-1 - In der Auswahldatei sind keine ausgewaehlten Datensaetze

BEMERKUNG
Die unter Verwendung von frstsel und nextsel wiedergewonnen Datensaetze werden in der Reihenfolge der logi-

schen Datensatznummern zurueckgegeben.

SIEHE AUCH
nextsel, prevsel, opensf, closesf

GDATA

NAME
gdata - Felddaten vom Bildschirm holen

SYNTAX
```
gdata (fx, fy, feld)
int fx, fy;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um Daten von einer spezifizierten Position auf dem Bildschirm zu holen, sie auf Gueltigkeit zu pruefen, zu konvertieren und dann direkt in der Datenbank zu speichern. Nur WDATA-bezogene Ueberpruefungen werden am Datenbank-Feld vorgenommen. Ist beispielsweise ein Datenbank-Feld im Schema NUMERIC 3, werden vor seiner Speicherung drei numerische Zeichen auf Gueltigkeit ueberprueft. Sind zusaetzliche Ueberpruefungen erforderlich, muss das Datenbank-Feld vor seiner Speicherung zuerst mittels gtube in einen Puffer eingegeben werden.

Die Argumente fx und fy sind ganze Zahlen, die die Position des Datenbank-Feldes auf dem Bildschirm enthalten. fx gibt die Spalte (von 0 bis 79) an und fy die Zeile (von 0 bis 23).

Zum Speichern des Datenbank-Feldes verwendet die Funktion pfield. Deshalb werden die Daten im aktuellen Datensatz des das Datenbank-Feld enthaltenden Typs gespeichert.

Gibt der Nutzer das "Neue Aktion"-Zeichen ein (normalerweise CTRL/U), gibt gdata sofort -2 zurueck und es werden keine Daten im Datenbank-Feld gespeichert. Gibt der Nutzer nur einen RETURN ein, werden keine Daten im Datenbank-Feld gespeichert und die Funktion kehrt normal zurueck.

Werden Eingaben vorgenommen, die nicht dem Feldtyp entsprechen (in einem numerischen Feld alpha) wird eine Fehlermeldung ausgegeben und der Cursor wird wieder an den Anfang des Feldes gesetzt.

RUECKGABEWERTE
0 - Die Eingabe wurde akzeptiert
-2 - Es wurde "Neue Aktion" (CTRL/U) eingegeben

SIEHE AUCH
gtube

GFIELD

NAME
gfield - ein Feld von Datenbank in einen Puffer holen

SYNTAX
```
gfield (feld, buf)
char *buf;
```

BESCHREIBUNG
Diese Funktion gewinnt ein Feld aus der Datenbank zurueck und speichert es in einem Puffer. Das Feld behaelt auch im Puffer sein internes Format. Das Feld wird aus dem aktuellen Datensatz des das Feld enthaltenden Typs wiedergewonnen.

RUECKGABEWERTE
0 - Das Feld wurde wiedergewonnen
-1 - Fuer dieses Feld ist der Lesezugriff nicht zulaessig (siehe accsfld).

SIEHE AUCH
pfield

GLOB

NAME
glob - eine Zeichenkette mit einer Metazeichenmaske vergleichen

SYNTAX
```
glob (strng, mask)
char *strng, *mask;
```

BESCHREIBUNG
Diese Funktion vergleicht eine Zeichenkette mit einer speziellen Zeichenkettenmaske. Die Zeichenkettenmaske hat dasselbe Format wie das, im WEGA-Programmierhandbuch bei sh(1), unter 'Erzeugung von Dateinamen', beschriebene Argument des Kommandowortes. Hier die Regeln:

Die Maske wird nach den Zeichen *, ? und [abgesucht. Tritt eines dieser Zeichen auf, wird die Maske als ein Muster betrachtet. Hier die Bedeutung der speziellen Zeichen:

* stimmt mit beliebiger Zeichenkette, einschliesslich der Nullzeichenkette, ueberein
? stimmt mit einem beliebigen Zeichen ueberein
[...] stimmt mit einem der eingeschlossenen Zeichen ueberein. Ein durch - getrenntes Zeichenpaar entspricht jedem beliebigen, lexikalisch zwi-

schen dem Paar liegenden Zeichen.

RUECKGABEWERTE
0 - Die Zeichenkette stimmt nicht mit der Maske ueberein
1 - Die Zeichenkette stimmt mit der Maske ueberein.

GTUBE

NAME
gtube - ein Feld vom Bildschirm in einen Puffer eingeben

SYNTAX
```
gtube (fx, fy, feld, buf)
int fx, fy;
char *buf;
```

BESCHREIBUNG
Diese Funktion wirkt im wesentlichen wie gdata, ausser dass die Daten nicht in der Datenbank, sondern in einem Puffer gespeichert werden. Das Datenbank-Feld wird dem Bildschirm an der spezifizierten Position entnommen, auf WDATA-Gueltigkeit ueberprueft, in internes Format konvertiert und dann in den angegebenen Puffer gespeichert. Der Pufferr muss einen Typ haben, der die eingehenden Datenbank-Felder folgendermassen akzeptiert:

NUMERIC	(1-4)	short
	(5-9)	long
FLOAT		double
AMOUNT	(1-7)	long
	(8-12)	double
DATE		short
TIME		short
STRING		char buf[?]
COMB		eine Struktur, die die Struktur der Komponenten der Datenbank-Felder widerspiegelt

Diese Funktion ist insbesondere dann nuetzlich, wenn ueber die Standard WDATA-Editierung hinausgegangen werden soll. In diesem Fall kann gtube aufgerufen werden, um das Datenbank-Feld einzugeben. Es koennen zusaetzliche Gueltigkeitsueberpruefungen vorgenommen werden und wenn das Datenbank-Feld gueltig ist, kann es unter Verwendung von pfield gespeichert werden.

Wenn der Nutzer das "Neue Aktion"-Zeichen CTRL/U eingibt, wird sofort von der Funktion -2 zurueckgegeben. Wird nur ein RETURN eingegeben, wird -3 zurueckgegeben und der Inhalt des Puffers ist undefiniert.

RUECKGABEWERTE
0 - Das gerade eingegebene Datenbank-Feld ist im

Puffer
-2 - Ein "Neue Aktion"-Zeichen wurde eingegeben
-3 - RETURN wurde eingegeben

SIEHE AUCH
gdatea

INBUF

NAME
inbuf - ein Bildmasken-Feld in einen Puffer eingeben

SYNTAX
```
inbuf (sfld, buf)
char *buf;
```

BESCHREIBUNG
Diese Funktion gestattet dem Nutzer, ein Bildmaskenfeld in einen Puffer einzugeben. Nach dem Aufruf sind die Daten im internen Format. Die Bildschirmkoordinaten und der Feldtyp werden dem Namen des Bildmaskenfeldes (sfld) entnommen. Die Editierung erfolgt entsprechend dem gueltigen externen Format der Daten, da diese Funktion gtube verwendet. Die Daten werden in buf gespeichert.

RUECKGABEWERTE

-3 - Es wurde nur RETURN eingegeben. DATE-Felder geben -32768 in buf zurueck, alle anderen Feldtypen lassen den Inhalt von buf undefiniert.
-2 - CTRL/U wurde eingegeben
0 - Ein gueltiges Bildmasken-Feld wurde eingegeben und ist in buf.

SIEHE AUCH
input

INIUBUF

NAME
iniubuf - den Lesepuffer fuer bseqacc (und andere) initialisieren

SYNTAX
```
iniubuf (bufadr, bufsz)
char *bufadr;
int bufsz;
```

BESCHREIBUNG
Diese Funktion gestattet dem Nutzer die Spezifizierung des "Lese-"Puffers fuer die Puffer-Routinen bseqacc, bgfield und bfaccess. Wird diese Routine nicht aufgerufen, wird mittels sbrk standardmaessig ein 2k-Byte-

Puffer zugewiesen. iniubuf ist zu Beginn des Nutzerprogramms aufzurufen und der Puffers sollte so gross wie moeglich sein, um die Geschwindigkeit der sequentiellen Suchvorgaenge in der Datenbank zu vergroessern.

ARGUMENTE
bufadr - Die Adresse eines im Nutzerprogramm zugewiesenen Puffers
bufsz - Puffergroesse in Bytes

RUECKGABEWERTE
keine

SIEHE AUCH
bseqacc, bgfield, bfaccess, bsetloc

INPUT

NAME
input - ein Feld eingeben

SYNTAX
```
input (sfld)
```

BESCHREIBUNG
Diese Funktion gibt ein Feld direkt von einem Bildmaskenfeld aus ein und speichert es in der Datenbank. Die x- und y-Koordinaten und der Feldtyp werden dem Namen des Bildmasken-Feldes (sfld) entnommen. Der, das von sfld spezifizierte, Feld enthaltende Datensatztyp muss der aktuelle sein. Da diese Funktion gdata verwendet, erfolgt die Editierung im Zusammenhang mit dem gueltigen externen Format der Daten.

RUECKGABEWERTE
0 - Bildmasken-Feld wurde eingegeben und akzeptiert
-2 - CTRL/U wurde eingegeben

SIEHE AUCH
inbuf

IVCMP

NAME
ivcmp - zwei Felder vergleichen

SYNTAX
```
ivcmp (ptr1, ptr2, length)
char *ptr1, *ptr2;
int length;
```

BESCHREIBUNG
Mit dieser Funktion werden zwei Zeichenfelder verglichen, um festzustellen, ob sie gleich sind. ptr1 und

ptr2 sind Pointer auf diese Felder. length ist die Anzahl der zu vergleichenden Zeichen.

RUECKGABEWERTE
0 - Die Zeichenketten stimmen nicht ueberein
1 - Die Zeichenketten sind gleich

KDATE

NAME
jdate - Julianisches Datum konvertieren

SYNTAX
```
kdate (jday, arr)
int jday, arr[3];
```

BESCHREIBUNG
Diese Funktion konvertiert ein Datum von einem ganzzahligen Format in ein ganzzahliges Feld, das den Monat, Tag und das Jahr enthaelt. Das erste Argument (jday) ist das Datum im Julianischen Format und das zweite Argument (arr) enthaelt das konvertierte Datum nach der Rueckgabe.

RUECKGABEWERTE
keine

KDAY

NAME
kday - ins Julianische Datum-Format konvertieren

SYNTAX
```
kday (arr)
int arr[3];
```

BESCHREIBUNG
Diese Funktion wird verwendet, um ein ganzzahliges Feld, das Monat, Tag und Jahr enthaelt, in eine Julianische sequentielle ganze Zahl zu konvertieren. Will man feststellen, ob das Datum gueltig war, kann dies nur geschehen, indem das sich ergebende Julianische Datum wieder in ein ganzzahliges Feld zurueckkonvertiert und mit dem Original verglichen wird. Liegt Uebereinstimmung vor, war das Datum gueltig.

RUECKGABEWERTE
das Datum im Julianischen Format

KEYBRD

NAME
keybrd - Tastatur des Terminals sperren oder freigeben

SYNTAX
```
keybrd (fd, flg)
int fd, tlg;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um die Terminaltastatur zu sperren oder freizugeben. Das erste Argument (fd) ist der Ausgabe-Dateideskriptor. Das zweite Argument (flg) spezifiziert, ob die Tastatur gesperrt oder freigegeben werden soll. Ist es ungleich Null, wird die Tastatur blockiert.

RUECKGABEWERTE
keine

LASTCHR

NAME
lastchr - letztes Zeichen einer Zeichenkette suchen

SYNTAX
```
lastchr (strng, len)
char *strng;
int len;
```

BESCHREIBUNG
Diese Funktion gibt den Index (zur Basis 0) des ersten Blanks oder Nullzeichens am Ende einer Zeichenkette zurueck. Sie beginnt am Ende der Zeichenkette und durchsucht diese, bis sie ein Zeichen verschieden von Blank oder ungleich Null findet. Die Argumente der Funktion sind ein Pointer auf die Zeichenkette und eine maximale Laenge.

RUECKGABEWERTE
Der Index des ersten auf das Ende der Zeichenkette folgenden Blanks oder Nullzeichens.

LEN

NAME
len - Laenge einer Zeichenkette bestimmen

SYNTAX
```
len (str)
char *str;
```

BESCHREIBUNG
Diese Funktion gibt die Laenge einer auf Null endenden Zeichenkette zurueck. Das Argument ist ein Pointer auf die Zeichenkette.

RUECKGABEWERTE

Die Laenge der Zeichenkette.

LOADSCR

NAME
loadscr - Bildmaske anzeigen, Parameter laden

SYNTAX
```
loadscr (scrn)
char *scrn;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um eine Bildmaske anzuzeigen und die zugehoerigen Parameter in den Speicher zu laden. Sie ist aufzurufen, bevor andere Funktionen, die Bildmaskenfelder verwenden, aufgerufen werden koennen. scrn ist der auf Null endende Name der zu verwendeneden Bildmaske. Existiert die Bildmaske nicht, oder existiert eines der von den Bildmaskenfeldern spezifizierten Datenbank-Felder im Schema nicht, bricht loadscr ab.

RUECKGABEWERTE
keine

LOC

NAME
loc - Datensatzadresse suchen

SYNTAX
```
loc (rnum, adr)
long *adr;
```

BESCHREIBUNG
Diese Funktion gibt die Adresse des aktuellen Datensatzes vom Typ rnum zurueck. Existiert kein aktueller Datensatz, ist die zurueckgegebene Adresse 0.

RUECKGABEWERTE
Diese Funktion hat keine Rueckgabewerte.

SIEHE AUCH
setloc

LOCK_R

NAME
lock_r - Geraet verriegeln

SYNTAX
```
lock_r (c)
char c;
```

BESCHREIBUNG

Diese Funktion wird verwendet, um ein Geraet fuer alle anderen Nutzer des Systems zu verriegeln. Sie kann verwendet werden, wenn man nicht alle Aktualisierungen der Datenbank verriegeln moechte, was bei Verwendung von startrans geschehen wuerde. Argument von lock_r ist einfach ein ASCII-Zeichen, das fuer das zu verriegelnde Geraet steht. Der Nutzer kann zur Bezeichnung der verschiedenen Geraete einen eigenen Zeichensatz erarbeiten. Von WEGA-DATA verwendet bereits einige Buchstaben belegt, die unten angefuehrt werden.

Verriegelt derselbe Nutzer ein und dasselbe Geraet zweimal, muss er das Geraet auch genauso oft entriegeln, um es fuer andere Nutzer frei zu machen. lock_r kann dann sinnvoll sein, wenn Geraete zwei Nutzern geteilt verwendet werden.

Versucht ein anderer Nutzer dieselbe Verriegelung vorzunehmen, wartet lock_r eine Sekunde und wiederholt dann den Versuch. Nach 5 Sekunden wird auf dem Bildschirm in Zeile 23 die Meldung "Waiting for lock" angezeigt. Kann lock_r nach einer Wartezeit von 30 Sekunden nicht fortfahren, tritt es aus dem Unterprozess aus und gibt die Steuerung wieder an den Vater-Prozess, normalerweise an den Menue-Handler, zurueck. Wahrscheinlich wurde von einem mittlerweile beendeten Prozess eine Verriegelungsdatei hinterlassen, die nun geloescht werden muss.

Die Funktion lock_r funktioniert, indem im aktuellen Verzeichnis (oder in $DBPATH, falls diese Umgebungsvariable gesetzt ist), Dateien angelegt werden, die den Namen LOCKx erhalten, wobei x das an die Funktion uebergebene Zeichen ist. WEGA-DATA verwendet d fuer die Datenbank-Datei und u fuer die Datenwoerterbuch-Datei (wdata.db). Ausserdem verwendet das Rekonfigurationsprogramm k, wenn eine Backup-Kopie des Datenwoerterbuchs angelegt wird.

Fruehere Versionen verwendeten fuer die Funktion den Namen lock. Der Name wurde geaendert, um eine Verwechslung mit dem WEGA-Systemaufruf lock(2) zu vermeiden. Zur Gewaehrleistung der Kompatibilitaet steht die alte WEGA-DATA-Funktion lock im Archiv libcomp.a zur Verfuegung. Soll weiterhin der Name lock verwendet werden, ist diese Bibliothek in die Kommandodateien des Ladeprogramms einzubeziehen.

In der folgenden Tabelle werden die von WEGA-DATA verwendeten Verriegelungen zusammengefasst:

Datei	Bedeutung
LOCKd	Die WDATA-Datenbank-Datei (file.db) wurde bei

Auftreten des Ausfalls gerade aktualisiert und muss wiederhergestellt werden. Die Datei muss vor Fortsetzung der Verarbeitung geloescht werden.

LOCKu Die Datenwoerterbuch-Datei (wdata.db) wurde gerade aktualisiert, als der Ausfall auftrat und muss wiederhergestellt werden. Vor Fortsetzung der Verarbeitung muss die Datei geloescht werden.

LOCKk Interne Verriegelung, die bei Rekonfiguration und Anlegen der Datenbank verwendet wird. Das Programm gewinnt bei erneutem Ablauf wdata.db aus wdata.bu zurueck. Diese Datei darf nicht geloescht werden.

LOCKl Die von der Programmladefunktion des Menue-Handlers (lfilegen) verwendete Verriegelung. Wird verwendet, um eine gleichzeitige Verwendung der vom Ladeprogramm verwendeten Datei a.out zu vermeiden. Wird lfilegen unterbrochen (Interrupt), muss LOCKl aus dem Verzeichnis build geloescht werden, bevor das Laden erfolgen kann.

Die Geraeteverriegelung funktioniert aehnlich dem WEGA-Druckspooler, der eine Datei namens /usr/spool/lpd/lock anlegt. Bei dieser Methode sind zwei sehr wichtige Gesichtspunkte zu beachten:

1. Alle Nutzer von WEGA-DATA MUESSEN ausschliesslich normale WEGA-Nutzer sein - und nicht der Superuser! Der Superuser kann die Verriegelungsdatei auch dann anlegen, wenn bereits eine solche existiert, wodurch er die konkurierende Steuerung ausser Kraft setzt.

2. Alle Nutzer, auf die Datenbank-Datei zugreifen, MUESSEN sich im selben Verzeichnis befinden wie die Datenbank-Datei (d.h. das Arbeitsverzeichnis muss das Verzeichnis sein, in dem sich die Datenbank-Datei befindet) oder alle Nutzer muessen $DBPATH auf dasselbe Verzeichnis gesetzt haben. Nutzer, mit verschiedenen Arbeitsverzeichnissen koennen untereinander nicht verriegelt werden.

Tritt ein Absturz auf, bricht ein Programm ab oder tritt ein Programmfehler auf, existiert moeglicherweise eine dieser Dateien und muss geloescht werden, bevor man weitermachen kann. Ist jedoch nach einem Ausfall eine LOCKx-Datei vorhanden, wird dadurch angezeigt, dass die zugehoerige Datenbank aus einem Backup zurueckgewonnen werden muss, da bei Auftreten des Ausfalls eine Operation zur Aktualisierung lief.

RUECKGABEWERTE

keine

SIEHE AUCH
lockrec, unlock_r

MAKESET

NAME
makeset - Identifizieren einer Menge zusammenhaengender Datensaetze

SYNTAX
makeset (feld, rfeld, richtung)

BESCHREIBUNG
Identifiziert eine Menge zusammenhaengender Datensaetze fuer nachfolgende Aufraufe von nextrec, prevrec und samerec. makeset kann nur dort benutzt werden, wo explizite Beziehungen definiert wurden.

ARGUMENTE
feld - der Name des Primaerschluessel-Feldes eines Datensatztyps. Ein Datensatz dieses Typs muss bei Aufruf dieser Funktion der aktuelle sein. Der Wert des Schluessel-Feldes bestimmt, welche Datensaetze ausgewaehlt werden. Wenn alle Datensaetze eines bestimmten Typs wiedergewonnen werden sollen, greift man am besten sequentiell zu (seqacc).

rfeld - der Name des Feldes einen anderen Datensatztyps, zu dem eine explizite Beziehung besteht. Es werden die Datensaetze ausgewaehlt, bei denen die Inhalte von rfeld und feld uebereinstimmen.

richtung - damit wird festgelegt, wohin der Datensatz-Pointer fuer spaetere Aufrufe von nextrec und prevrec zeigen soll. Zu diesem Zweck sind in datei.h zwei Zeichenketten definiert: first und last. Bei Angabe von first wird der Pointer auf den Anfang gesetzt und der erste Aufruf von nextrec liefert den zuletzt hinzugefuegten Datensatz. Bei last zeigt der Pointer auf das Dateiende und der erste Aufruf von prevrec liefert den "aeltesten" Datensatz der Datei. unisort kann verwendet werden, wenn eine andere Reihenfolge gewuenscht wird. Beispielsweise liefert der folgende Aufruf alle Modelle des aktuellen Herstellers in "LIFO":

```
makeset (monr, mohenr, first);
```

Der Aufruf liefert alle Modell-Datensaetze, die dieselbe Herstellernummer haben, wie der aktuelle Hersteller-Datensatz.

RUECKGABEWERTE
 0 - das Set wurde erstellt
-1 - Beziehung existiert nicht
-2 - Tabellenueberlauf

SIEHE AUCH
clrset, faccess, nextrec, prevrec, samerec, setsize, sfrstrec, slastrec, sprevrec, unisort

MTCHITM

NAME
mtchitm - fuer unisel ein Suchfeld eingeben

SYNTAX
```
mtchitm (selfile, feld, rfeld)
char *selfile;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um in die Auswahltabelle fuer unisel ein Such-Feld einzugeben. Ein Such-Feld beschreibt einen Vergleich zwischen dem Typ des Zieldatensatzes und einer existierenden Auswahldatei.

Man beginnt mit einer von einem vorher ergangenen Aufruf unisel, uqsrch oder uqmtch erzeugten Auswahldatei. Die Datensaetze in der Auswahldatei sind "Vaeter" des von unisel ausgewaehlten Zieldatensatzes.

Es soll beispielsweise von den Datensatztypen kunde und bestellung ausgegangen werden:

```
  kunde     100                 best     200
    *kunr    NUMERIC    4        *benr   NUMERIC    6
    kort     STRING    20        bekun   NUMERIC    4
```

Es sollen alle Bestellungen ausgewaehlt werden, die von Kunden aus Dresden eingegangen sind. Zuerst wird uqsrch verwendet, um eine selfile zu schaffen, in der alle Kunden aus Dresden enthalten sind und dann wird mtchitm verwendet, um ein Such-Feld einzugeben. feld muss bekun und rfeld muss kunr sein. Der Aufruf unisel wuerde dann alle Datensaetze best auswaehlen, bei denen der Wert von bekun mit einem Wert kunr uebereinstimmt. Eine explizite Beziehung ist nicht erforderlich, aber der Suchvorgang verlaeuft schneller wenn eine solche Beziehung existiert.

ARGUMENTE
selfile - Eine auf Null endende Zeichenkette, die den Namen der Auswahldatei enthaelt
feld - Das Zieldatensatz-Feld, das in Beziehung mit einem Feld im Datensatztyp selfile steht
rfeld - Das Feld in selfile, auf das Bezug genommen

wird.

RUECKGABEWERTE
0 - normale Beendigung
-1 - die zugehoerige Auswahldatei ist nicht im korrekten Format
-2 - die Auswahltabelle ist voll

SIEHE AUCH
entsitm, sfncitm, clrsitm, clrfitm, uqmtch, uqsrch

MV_CUR

NAME
mv_cur - Cursorposition setzen

SYNTAX
```
mv_cur (fd, x, y)
int fd, x, y;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um den Cursor in eine spezifizierte Position zu bringen. Das erste Argument (fd) ist der Ausgabedateideskriptor. Das zweite und dritte Argument sind die x- und y-Koordinaten, an die der Cursor gebracht werden soll.

RUECKGABEWERTE
keine

NEXTREC

NAME
nextrec - den naechsten Datensatz der Auswahlmenge holen

SYNTAX
```
nextrec (feld, rfeld)
```

BESCHREIBUNG
Mit nextrec wird in einem durch den Aufruf makeset erzeugten Satz von Datensaetzen der naechste Datensatz zum aktuellen Datensatz gemacht. Das Paar feld, rfeld muss mit dem Paar im vorhergehenden Aufruf makeset uebereinstimmen. feld ist der Schluessel eines Datensatztyps und rfeld ist ein normales Feld in einem anderen Datensatztyp, das sich auf feld bezieht. Die Datensaetze werden in der Reihenfolge LIFO, in der sie auch in die Datei eingefuegt wurden, zurueckgegeben.

RUECKGABEWERTE
0 - Der naechste Datensatz der Menge ist der aktuelle
-1 - Im Moment ist keine solche Menge definiert
-2 - Ende der Liste wurde erreicht

SIEHE AUCH
clrset, faccess, makeset, prevrec, samerec, setsize, sfrstrec, slastrec, snextrec, sprevrec, unisort

NEXTSEL

NAME
nextsel - naechsten ausgewaehlten Datensatz holen

SYNTAX
```
#include "unisel.h"
nextsel (sf)
SELFILE *sf;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um den naechsten ausgewaehlten Datensatz einer Auswahldatei zum aktuellen zu machen. sf ist die Adresse der Struktur SELFILE, die nach einem Aufruf von opensf zurueckgegeben wurde.

RUECKGABEWERTE
1 - Der naechste ausgewaehlte Datensatz ist der aktuelle
-1 - Das Ende der Auswahldatei ist erreicht.

SIEHE AUCH
frstsel, prevsel, opensf, closesf

OBLANK

NAME
oblank - Druck-Puffer initialisieren

SYNTAX
```
oblank()
```

BESCHREIBUNG
Diese Funktion wird zur Initialisierung des Druck-Puffers am Anfang eines eine Druckerausgabe ausfuehrenden Programmes aufgerufen. Die Funktion braucht fuer jedes Laden in den Hauptspeicher nur einmal aufgerufen werden, allerdings vor dem ersten Aufruf zur Ausgabe.

RUECKGABEWERTE
keine

SIEHE AUCH
flush, obuf, odata, pform, prstr

OBUF

NAME
obuf - Ausgabe von einem Puffer an den Drucker

SYNTAX
```
obuf (col, feld, buf)
char *buf;
int col;
```

BESCHREIBUNG
Diese Funktion formatiert den Inhalt eines Puffers und bringt die Ergebnisse zur spaeteren Ausgabe durch flush in die aktuelle Druckzeile. Die Daten werden entsprechend des Typs des feld-Argumentes konvertiert und im Druckpuffer in der Spalte col gespeichert.

RUECKGABEWERTE
keine

SIEHE AUCH
flush, oblank, odata, pform, prstr

ODATA

NAME
odata - Ausgabe eines Datenbank-Feld an den Drucker

SYNTAX
```
odata (col, feld)
int col;
```

BESCHREIBUNG
Diese Funktion formatiert ein Feld zur spaeteren Ausgabe durch flush in die aktuelle Druckzeile. Das Feld wird wie durch pdata aus der Datenbank entnommen und dann in die Druckzeile, Spalte col gebracht.

RUECKGABEWERTE
keine

SIEHE AUCH
flush, oblank, obuf, pform, prstr

OPENDB

NAME
opendb - Oeffnen einer Datenbank-Datei

SYNTAX
```
opendb (dbname, opt)
char *dbname;
int opt;
```

BESCHREIBUNG

Diese Funktion wird verwendet, um einem Programm das Eroeffnen von Datenbank-Dateien, die nicht die Standarddatei file.db sind, zu gestatten. Die Host-Sprachfunktionen auf Nutzerebene oeffnen file.db, wenn keine Datenbank-Datei geoeffnet ist. Deshalb muss zum Eroeffnen einer anderen Datenbank-Datei opendb noch vor Aufruf einer anderen WEGA-DATA-Funktion aufgerufen werden. Das Einlesen der Namen und/oder Synonyme kann erheblich viel Platz beanspruchen. Daher muss im Programm genug Platz fuer Daten vorhanden sein. opendb zeigt einen Fehler an, wenn nicht genuegend Speicherplatz vorhanden ist.

ARGUMENTE

dbname - Pointer auf den Namen der Datenbank-Datei

opt - Teilt opendb mit, welche zusaetzlichen Datenwoerterbuch-Informationen einzulesen sind, falls solche eingelesen werden sollen. Die Option wird gebildet, indem eine Kombination aus folgenden in file.h vereinbarten Identifizierungen durch logisches ODER verbunden werden. Sollen keine Namen eingelesen werden, wird eine literale 0 uebergeben.

  FN - Feldnamen lesen
  FS - Feldsynonyme lesen
  RN - Datensatznamen lesen
  RS - Datensatzsynonyme lesen

RUECKGABEWERTE

Diese Funktion hat keine Rueckgabewerte.

SIEHE AUCH

dbproc, closedb

OPENSF

NAME

opensf - Oeffnen einer Auswahldatei

SYNTAX

```
#include "unisel.h"
SELFILE *opensf (selfile)
char *selfile;
```

BESCHREIBUNG

Diese Funktion wird verwendet, um eine von unisel, dem Listenprozessor oder ENTER angelegte Auswahldatei zu eroeffnen. Im Ergebnis dieses Aufrufes wird eine Adresse vom Typ SELFILE zurueckgegeben, die in darauf folgenden Aufrufen von frstsel, nextsel, prevsel und closesf verwendet wird. Am einfachsten kann die Funktion folgendermassen verwendet werden:

```
beis ()
{
  SELFILE *ss, *opensf();
  ss = opensf ("selfile");
}
```

ARGUMENTE

selfile - Die Adresse einer auf Null endenden Zeichenkette, die den Namen der zu eroeffnenden Auswahldatei enthaelt.

RUECKGABEWERTE

-1 - Es trat ein Fehler auf und errno wird folgendermassen gesetzt:
- 101 - selfile konnte nicht geoeffnet werden
- 102 - selfile ist nicht im korrekten Format
- 103 - die maximale Anzahl der offenen Auswahldateien wurde erreicht
- 104 - ein auf sbrk zurueckzufuehrender Fehler

Der normale Rueckgabewert ist eine auf eine SELFILE-Struktur zeigende Adresse.

SIEHE AUCH

frstsel, nextsel, prevsel, closesf

OPNBTS

NAME

opnbts - Oeffnen eines B-Baum-Index zum Suchen

SYNTAX

opnbts (feld)

BESCHREIBUNG

opnbts oeffnet den B-Baum eines Feldes zum Durchsuchen. Fuer das entsprechende Feld muss ein Index existieren. Ist ein Index erst einmal geoeffnet, kann er wiederholt durchsucht werden. Die Funktionen btsrch und btnext werden zum Durchsuchen, und dann zum Durchlaufen (optionell) eines fuer den Suchvorgang geoeffneten B-Baumes verwendet. Es koennen bis zu fuenf Indizes geoeffnet sein und gleichzeitig durchsucht werden.

RUECKGABEWERTE

-3 - Das spezifizierte Feld ist nicht in einem B-Baum-Index verzeichnet
-2 - Die maximale Anzahl der Indizes ist eroeffnet
-1 - Der Index des spezifizierten Feldes ist bereits geoeffnet
0 - Der Index des spezifizierten Feldes ist zum Durchsuchen geoeffnet.

SIEHE AUCH

btnext, btsrch, closbt, fldidxd

OPRF

NAME
oprf - eine Pipe fuer lpr aufbauen

SYNTAX
oprf()

BESCHREIBUNG
Diese Funktion wird von Programmen verwendet, die ueber den Druckspooler lpr ausgeben. Die Funktion gibt einen Dateideskriptor zum Schreiben in die Pipe zurueck. oprf verwendet die Umgebungsvariable SPOOLER, lpr ist Standard.

RUECKGABEWERTE
Der Ausgabedatei-Deskriptor

-1 - Die Pipe konnte nicht aufgebaut werden.

OUTBUF

NAME
outbuf - Puffer in einem Bildmaskenfeld anzeigen

SYNTAX
```
outbuf (sfld, buf)
char *buf;
```

BESCHREIBUNG
Diese Funktion nimmt die Daten aus einem Puffer, formatiert sie zur Ausgabe und zeigt sie in einem dem angegebenen Bildmasken-Feld an. Position auf dem Bildschirm und Feldtyp werden dem Bildmasken-Feldnamen (sfld) entnommen. Die Daten werden buf entnommen.

RUECKGABEWERTE
keine

SIEHE AUCH
output

OUTPUT

NAME
output - ein Feld anzeigen

SYNTAX
```
output (sfld)
```

BESCHREIBUNG
Diese Funktion gibt ein Bildmasken-Feld direkt von der

Datenbank an die Datenposition des spezifizierten Bildmasken-Feldes (sfld) aus. Die x- und y-Koordinaten und der Feldtyp werden durch den Namen des Bildmasken-Feldes bestimmt. Der, das von sfld spezifizierte Datenbankfeld enthaltende, Datensatztyp muss der aktuelle sein.

RUECKGABEWERTE
keine

SIEHE AUCH
outbuf

PDATA

NAME
pdata - Daten aus der Datenbank auf dem Bildschirm anzeigen

SYNTAX
```
pdata (fx, fy, feld)
int fx, fy;
```

BESCHREIBUNG
Diese Funktion nimmt ein Datenbank-Feld aus der Datenbank, konvertiert es zur Ausgabe und zeigt es dann an der spezifizierten Position auf dem Bildschirm an. fx und fy sind die x- bzw. y-Koordinaten, an denen die Daten auf dem Bildschirm angezeigt werden. Da gfield zum Holen der Daten verwendet wird, wird das Datenbank-Feld dem aktuellen Datensatz des, das Feld enthaltenden, Typs entnommen.

RUECKGABEWERTE
keine

SIEHE AUCH
ptube

PFIELD

NAME
pfield - ein Feld in der Datenbank aktualisieren

SYNTAX
```
pfield (feld, buf)
char *buf;
```

BESCHREIBUNG
pfield aktualisiert das spezifizierte Feld mit dem Pufferinhalt. Die im Puffer befindlichen Daten muessen im internen Format vorliegen. Die Daten werden in den aktuellen Datensatz des, das Feld enthaltenden, Typs gespeichert. pfield kann mit beliebigen Feldern verwen-

det werden, bei Schluesselfeldern und Beziehungsfeldern sind jedoch einige Einschrankungen zu beachten. Wird fuer den Schluessel eines Datensatzes pfield ausgefuehrt, wird der Schluessel geaendert, d.h. die Daten muessen wie in addrec einen nur einmal vorkommenden Schluessel definieren. Sind zugehoerige Datensaetze vorhanden, die auf den Schluessel Bezug nehmen, ist pfield nicht zulaessig, da diese Datensaetze alle Felder enthalten, die dem der Aenderung unterliegenden Schluessel entsprechen. Das wuerde aber zu einer logischen Inkonsistenz der Datenbank fuehren. Richtig wird vorgegangen, indem das Beziehungsfeld in jedem der zugehoerigen Datensaetze durch pfield auf den gewuenschten neuen Wert gesetzt wird. Dabei ist jedoch zu beachten, dass fuer den neuen Schluesselwert ein Datensatz existieren muss. Diese Tests sichern , dass die im Schema definierte logische Integritaet der Datenbank erhalten bleibt.

RUECKGABEWERTE
 0 - das Feld wurde gespeichert
-1 - es wurde versucht, den Schluessel eines Datensatzes zu aendern, auf den andere Datensaetze Bezug nehmen
-2 - es wurde versucht, einen Schluessel zu speichern, der bereits existiert
-3 - es wurde versucht, eine explizite Beziehung zu einem nicht existenten Datensatz herzustellen.
-4 - Schreibzugriff unzulaessig (siehe accsfld)
-5 - Datensatz ist durch einen anderern Prozess verriegelt (siehe lockrec)

SIEHE AUCH
gfield

PFORM

NAME
pform - spezielle Masken ausgeben

SYNTAX
```
pform (xbuf, ofd, pftab)
char xbuf[];
int ofd;
struct pftable
{
  int (*func) ();
  char *str;
} pftab[];
```

BESCHREIBUNG
pform gestattet, dass das Format von Reporten in einer externen Textdatei spezifiziert werden kann, die waehrend der Laufzeit gelesen wird. Datenbank-Felder, literale Zeichenketten und vom Nutzer definierte Funktions-

ergebnisse koennen an die in der externen Datei spezifizierten x-y-Koordinaten gebracht werden. Dadurch koennen Aenderungen am Format vorgenommen werden, ohne dass das Programm neu kompiliert oder geladen werden muss, d.h. ein Nutzer kann diese Veraenderungen auch ohne Hilfe des Programmierers vornehmen.

pform wird verwendet, indem die Funktion pform aufgerufen wird, die dann eine einzelne Kopie des in der Textdatei beschriebenen Formates ausgibt. xbuf ist eine Zeichenkette, die den Namen der externen Formatdatei enthaelt. ofd ist der Ausgabedatei-Deskriptor, der von einem Aufruf open oder create zurueckgegeben wird. pftab ist eine Tabelle mit vom Nutzer definierten Funktionsadressen und Namen zum Druck spezieller Zeichenketten.

Die Verwendung von pform wird am besten am Beispiel illustriert. Im folgenden ist ein einfaches Programm angefuehrt, mit dem unter Verwendung von pform fuer jeden im Lagerbestand vorhandenen Gegenstand ein Etikett ausgegeben werden soll, wobei das im Benutzerhandbuch vorgestellte Schema fuer den Lagerbestand verwendet wird.

```
#
/*********************************************************
*   inv300.c
*
*   Das ist ein einfaches Report-Programm fuer den Lagerbe-
*   stand, dass Etiketts fuer alle Artikel druckt. Es be-
*   nutzt pform, um die Ausgabe zu formatieren.
*
*   Parameter:
*
*   Ausgabe:
*
*********************************************************/
$include "../../def/file.h"

extern int pdate ();   /* Nutzerfunktion fuer Tagesdatum */

struct
{
  int (*func) ();
  char *str;
} pftab [] = {
               pdate, "pdate   ",
               0,0
             };

main ()
{
  int fd;                     /* Drucker-Dateideskriptor */

  fd = oprf ();               /* Eroeffnen Spooler-Datei */
```

```
  if (seqacc (art, first) == 0)
  do
  {
    faccess (modell, artmod);
    faccess (her, mohenr);
    pform ("inv300", fd, pftab);
  } while (seqacc (art, next) == 0);
}


#
/*********************************************************
*
*   pdate.c
*
*   Nutzerfunktion fuer pform zum Druck des Tagesdatums.
*
*   Parameter:
*
*   Ausgabe:
*
*********************************************************/
#include <time.h>

pdate (xbuf, xarg)
char xbuf[], xarg[];
{
  long tvec;          /* fuer Zeit in Sekunden seit 1970 */

  struct tm *localtime(),
            *t;      /* fuer Ergebnis des Rufs Lokalzeit */

  time (&tvec);                     /* Hohlen Tagesdatum */
  t = localtime (&tvec);
                  /* Konvertieren zu etwas Sinnvollerem */

  sprintf (xbuf, "%2d/%2d/%2d", (t->tm_mon)+1, t->tm_mday,
           t->tm_year);

        /* Datum in den Ausgabe-Puffer von pform bringen */
}
```

Text der Report-Formatdatei (inv.300.p):

```
6
1,1,SERIEN-NUMMER
15,1,sernr
1,2,MODELL-NUMMER
15,2,artmod_monr
1,3,HERGESTELLT VON
17,3,hename
1,5,ETIKETTIER-DATUM:
19,5,pdate
```

Beispieletikett:

```
SERIEN-NUMMER 7
MODELL-NUMMER 23000
HERGESTELLT VON "PGH Metall"

ETIKETTIER-DATUM: 7/23/86
```

Dieses Programm druckt fuer jede im Lagerbestand vorhandene Ware ein Etikett. pform geht davon aus, dass die Report-Formatdatei in dem Verzeichnis ist, in dem das Programm ablaeueft, und zwar in einer Datei namens inv300.p.

pform nimmt weiterhin an, dass die linke obere Ecke des Formulars in den x-y-Koordinaten 1-1 liegt. Die x-Koordinate wird nach rechts groesser und die y-Koordinate wird mit der Maske groesser. Die erste Zeile der Datei hat die Laenge des Formulars, im vorliegenden Fall 6 Zeilen. Diese Laenge umfasst auch Zeilen, die nicht gedruckt werden, damit pform die Anzahl der Zeilenvorschuebe vornehmen kann, die erforderlich sind, um an den Anfang des naechsten Formulars zu gelangen. Jede weitere Zeile definiert auf dem Formular eine Position, an der Text zu drucken ist. Diese Zeilen haben folgendes Format:

```
x, y, string
```

wobei x die x-Koordinate des Textes und y die y-Koordinate des Textes ist und die Zeichenkette string ist entweder

1. literaler Text (in dem keine Kommas enthalten sein duerfen)
2. der Name eines WEGA-DATA Datenbank-Feldes
3. Funktionsname, Parameterzeichenkette

Zeile 2 ist ein Beispiel fuer eine literale Text-Zeichenkette. Die Zeichenkette SERIEN-NUMMER ist in Zeile 1, Spalte 1 zu drucken. Zeile 3 ist ein Beispiel fuer den Namen eines WDATA-Datenbank-Feldes. Der das Feld enthaltende Datensatz muss bei Aufruf von pform der aktuelle sein, ansonsten wird ein WDATA-Fehler generiert. Der Inhalt des Feldes sernr wird in Zeile 1, Spalte 15 gedruckt.

Zeile 9 ist ein Beispiel fuer einen Funktionsnamen, im vorliegenden Fall ohne Parameter. Der Funktionsname ist gleich dem in pftab im Programm definierten Funktionsnamen. pform sucht pftab nach dem Namen der Funktion ab, der 8 Zeichen lang oder kuerzer sein kann. Der Parameter kann eine beliebige aus Zeichen bestehende Zeichenkette ohne ein Leerzeichen sein. Die Nutzer-Funktion ist fuer das Zerlegen des Parameters verantwortlich.

pform ruft die Nutzer-Funktion wie folgt mit zwei Argumenten auf:

```
(*pftab[i].func) (pfstr, pfarg)
char pfstr[];
char pfarg[];
```

pfstr ist ein globales Feld, wobei die Nutzer-Funktion die zu drueckende Zeichenkette kopiert und pfarg die Parameter-Zeichenkette ist. Was die Nutzer-Funktion tun kann, unterliegt keinen Beschraenkungen. Die ausgegebene Zeichenkette kann jedoch maximal nur 128 Zeichen lang sein.

Es ist zu beachten, dass die Zeilen in der Formatdatei nach y- und x-Koordinaten sortiert werden. Man kann sich das in etwa wie den Ausdruck eines Reportes auf einem Drucker vorstellen - naemlich von links nach rechts ohne ruecklaeufige Listenverarbeitung.

SIEHE AUCH
flush, oblank, obuf, odata, prstr

PREVREC

NAME
prevrec - vorhergehenden Datensatz in einem Satz holen

SYNTAX
prevrec (feld, rfeld)

BESCHREIBUNG
prevrec macht innerhalb einer von dem Aufruf makeset erstellten Menge von Datensaetzen den vorhergehenden Datensatz zum aktuellen Datensatz. Das Paar feld, rfeld muss mit dem Paar des vorangegangenen makeset-Aufrufes uebereinstimmen. feld ist der Schluessel eines Datensatzes und rfeld ist ein sich darauf beziehendes normales Feld in einem anderen Datensatz. Die Datensaetze werden in FIFO-Reihenfolge zurueckgegeben, in der sie auch in die Datei eingefuegt wurden.

RUECKGABEWERTE
0 - der vorhergehende Datensatz im Satz ist jetzt der aktuelle
-1 - gegenwaertig ist kein solcher Satz definiert
-2 - der Anfang der Liste ist erreicht

SIEHE AUCH
clrset, faccess, makeset, nextrec, samerec, setsize, sfrstrec, slastrec, snextrec, sprevred, unisort

PREVSEL

NAME
prevsel - zuletzt ausgewaehlten Datensatz holen

SYNTAX
```
#include "unisel.h"
prevsel (sf)
SELFILE *sf;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um den zuletzt ausgewaehlten Datensatz einer Auswahldatei zum aktuellen Datensatz zu machen. sf ist eine Adresse einer SELFILE-Struktur, die nach einem Aufruf von opensf zurueckgegeben wurde.

RUECKGABEWERTE
1 - der zuletzt ausgewaehlte Datensatz ist der aktuelle Datensatz
-1 - Der Anfang der Auswahldatei wurde erreicht.

SIEHE AUCH
nextsel, frstsel, opensf, closesf

PRIAMD

NAME
priamd - nach Betriebsmodus fragen

SYNTAX
```
priamd (line)
int line;
```

BESCHREIBUNG
Diese Funktion bildet ein Interface zum Sicherheitssystem des Menue-Handlers, um den gewuenschten Betriebsmodus fuer interaktive Datenbank-Verwaltungsprogramme zu erhalten. Sie verwendet vom Menue-Handler an die Funktion sysrecev uebergebene Parameter zur Bestimmung der aktuellen Nutzer-Identifikation und der Aktualisierungsrechte. Deshalb muessen Programme, die priamd aufrufen mit sysrecev als "Haupt"-Einsprungstelle geladen werden. Weitere Details siehe Abschnitt 2. priamd zeigt auf Zeile 22 des Bildschirms einen Prompter an, in dem einige der folgenden Optionen angeboten werden:

```
[I]NQUIRE,[A]DD,[M]ODIFY,[D]ELETE
```

Welche Optionen tatsaechlich angezeigt werden, haengt von dem in "Employee Maintenance" eingegebenen Aktualisierungsrecht des aktuellen Nutzers ab (siehe Abschnitt 2.1.4). Fuer jeden Nutzer kann eine beliebige Kombination der oben angefuehrten Rechte spezifiziert werden.

Trifft der Nutzer eine gueltige Auswahl, wird der ausgewaehlte Mode auf dem Bildschirm in den x-y-Koordinaten (1,line) angezeigt. Dann wird die Steuerung an die aufrufende Funktion zurueckgegeben. Die aufrufende Funktion muss den Wert des Rueckgabestatus fuer die Ausfuehrung der entsprechenden Verwaltungsoperationen verwenden. Wird der Statuskode fuer CTRL/U (-2) empfangen, muss die aufrufende Funktion durch Austritt die Steuerung an den Menue-Handler zurueckgeben.

Ist die Umgebungsvariable UUACL vorhanden, erhaelt priamd den Wert der Zugriffsebene von der Umgebungsvariablen UUACL. Ein UUACL-Wert ist eine ASCII-Zeichenkette, z.B.: UUACL=02. priamd konvertiert die Zeichenkette unter Verwendung der WEGA-Subroutine atoi in ein dezimales ganzzahliges Format. Ist UUACL nicht vorhanden und nicht auf eine gueltige Zugriffsebene gesetzt (Wert 0-15), wird davon ausgegangen, dass die globale Variable logacl bereits einen gueltigen Zugriffsebenen-Wert enthaelt. Ist der Wert von UUACL gueltig, wird er in logacl gebracht.

Die globale Variable int locacl wird von sysrecev auf die Zugriffsebene des aktuellen Nutzers gesetzt. Das Format des Wortes ist folgendes:

```
logacl & 010 - Anfragerecht
logacl & 004 - Hinzufuegungsrecht
logacl & 002 - Modifizierungsrecht
logacl & 001 - Loeschrecht
```

Wird die COBOL-Routine Usub.c verwendet, die von der Routine sub.c aufgerufen wird, setzt sie die globale Variable logacl auf 15, wodurch die Verwendung von priamd auch dann ermoeglicht wird, wenn die Umgebungsvariable UUACL nicht verwendet wird.

RUECKGABEWERTE

```
-2 - CTRL/U wurde eingegeben
 0 - Abfragemodus
 1 - Hinzufuegungsmodus
 2 - Modifizierungsmodus
 3 - Loeschmodus
```

PRMP

NAME

prmp - eine Zeichenkette in geringer Intensitaet auf dem Bildschirm anzeigen.

SYNTAX

```
prmp(x, y, str)
int x, y;
char *str;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um eine Zeichenkette auf dem Bildschirm auf einer bestimmten Position anzuzeigen. Die ersten beiden Argumente sind die x- bzw. y-Koordinaten des Anfangs der Zeichenkette. Das letzte Argument (str) ist die anzuzeigende, auf Null endende Zeichenkette. prmp zeigt die Zeichenkette in geringer Intensitaet an (Hintergrundmodus).

RUECKGABEWERTE
keine

SIEHE AUCH
prmpf, prmprv

PRMPF

NAME
prmpf - eine Zeichenkette mit hoher Intensitaet auf dem Bildschirm anzeigen

SYNTAX
```
prmpf (x, y, str)
int x, y;
char *str;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um eine Zeichenkette auf dem Bildschirm auf einer bestimmten Position anzuzeigen. Die ersten beiden Argumente sind die x- bzw. y-Koordinaten des Anfangs der Zeichenkette. Das letzte Argument (str) ist die anzuzeigende, auf Null endende Zeichenkette. prmp zeigt die Zeichenkette mit hoher Intensitaet an (Vordergrundmodus).

RUECKGABEWERTE
keine

SIEHE AUCH
prmpf, prmprv

PRMPRV

NAME
prmprv - eine Zeichenkette invers auf dem Bildschirm anzeigen

SYNTAX
```
prmprv (x, y, str)
int x, y;
char *str;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um eine Zeichenkette auf

dem Bildschirm auf einer bestimmten Position anzuzeigen. Die ersten beiden Argumente sind die x- bzw. y-Koordinaten des Anfangs der Zeichenkette. Das letzte Argument (str) ist die anzuzeigende, auf Null endende Zeichenkette. prmp zeigt die Zeichenkette im Modus 'video invers' auf dem Bildschirm an.

RUECKGABEWERTE
keine

SIEHE AUCH
prmp, prmpf

PRSTR

NAME
prstr - eine Zeichenkette an den Druck-Puffer ausgeben

SYNTAX
```
prstr (str, col)
char *str;
int col;
```

BESCHREIBUNG
Diese Funktion gestattet dem Nutzer eine feste Zeichenkette an den Druckpuffer auszugeben, so dass diese von dort aus durch flush ausgegeben werden kann. Das erste Argument ist eine auf Null endende Zeichenkette. Das zweite Argument (col) ist die fuer die Ausgabe vorgesehene Spalte.

RUECKGABEWERTE
keine

SIEHE AUCH
flush, oblank, obuf, odata , pform

PRTMSG

NAME
prtmsg - Meldung anzeigen und auf Quittierung warten

SYNTAX
```
prtmsg (x, y, strng)
int x, y;
char *strng;
```

BESCHREIBUNG
Diese Funktion zeigt strng an x,y auf dem Bildschirm an und wartet darauf, dass der Nutzer ueber Tastatur eine Eingabe vornimmt. Nach der Eingabe wird die Zeichenkette geloescht. Die Funktion kann zur Anzeige von Fehlermeldungen verwendet werden.

RUECKGABEWERTE
keine

PTCT_CRT

NAME
ptct_crt - Status des Terminal-Schutzmodus' aendern

SYNTAX
```
ptct_crt (fd, flg)
int fd, flg;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um das Terminal in den "Schutz"-Modus zu bringen bzw. diesen aufzuheben. Ist das Terminal im Schutzmodus, koennen schreibgeschuetzte Zeichen nicht ueberschrieben werden. Auch das Loeschen dieser Zeichen ist nicht moeglich. Das erste Argument (fd) ist der Terminaldatei-Deskriptor. Das zweite Argument (flg) ist ein Flag, das angibt, ob der Schutzmodus an- oder ausgeschaltet werden soll. Alle Werte ungleich Null bewirken Schutzmodus.

RUECKGABEWERTE
keine

PTCT_WRT

NAME
ptct_wrt - Schreibschutzstatus des Terminals aendern

SYNTAX
```
ptct_wrt (fd, flg)
int fd, flg;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um den Schreibschutzmodus des Terminals zu modifizieren. Befindet man sich im Schreibschutzmodus, werden alle Zeichen im Schutzmodus (Hintergrund) geschrieben. Auf diese Weise geschriebene Zeichen koennen nicht ueberschrieben oder geloescht werden (ausser durch generelles Loeschen), wenn das Terminal im Schutzmodus ist (siehe ptct_crt). Das erste Argument (fd) ist der Ausgabedatei-Deskriptor. Das zweite Argument (flg) ist ein Flag, das angibt, ob das Terminal nach dem Aufruf im Schreibschutzmodus sein wird. Ist dieses ungleich Null, geht das Terminal in Schreibschutzmodus ueber.

RUECKGABEWERTE
keine

PTUBE

NAME
ptube - ein Datenbank-Feld aus einem Puffer auf den Bildschirm schreiben

SYNTAX
```
ptube (fx, fy, feld, buf)
int fx, fy;
char *buf;
```

BESCHREIBUNG
Diese Funktion ist pdata ganz aehnlich , ausser dass die Daten aus dem Puffer und nicht aus der Datenbank entnommen werden. Die Funktion konvertiert die Daten in dem Puffer in das externe Format und zeigt sie dann auf dem Bildschirm an. Die Art des Puffers, die benoetigt wird, um jeden Feldtyp von WEGA-DATA zu beinhalten, wird unter gtube beschrieben.

Diese Funktion ist dann sinnvoll, wenn die anzuzeigenden tatsaechlichen Daten berechnete Daten sind. Man waehlt einfach ein Datenbank-Feld aus, das vom selben Typ ist wie diese Werte und verwendet es als das Argumentfeld.

RUECKGABEWERTE
keine

SIEHE AUCH
pdata

QMOVE

NAME
qmove - Kopieren einer Zeichenkette in eine andere

SYNTAX
```
qmove (ptr1, ptr2, length)
char *ptr1, *ptr2;
int length;
```

BESCHREIBUNG
Diese Funktion kopiert Zeichen, bis zum Erreichen von length, aus dem ersten Feld ins zweite Feld.

RUECKGABEWERTE
keine

SAMEREC

NAME
samerec - den aktuellen Datensatz einer ausgewaehlten Menge wiederherstellen

SYNTAX
samerec (feld, rfeld)

BESCHREIBUNG
samerec stellt den aktuellen Datensatz einer ausgewaehlten Menge wieder her, damit mit den anderen Datenbank-Funktionen Operationen an dieser Menge ausgefuehrt werden koennen. Diese Funktion ist nuetzlich, wenn verschiedene Datensaetze desselben Datensatztyps manipuliert werden sollen. Das Paar feld, rfeld muss mit dem im vorhergehenden Aufruf makeset verwendeten Paar uebereinstimmen. feld ist der Schluessel eines Datensatztyps und rfeld ist ein normales Feld in einem anderen Datensatztyp, das sich auf dieses bezieht.

RUECKGABEWERTE
0 - der Datensatz ist der aktuelle
-1 - es wurde kein solcher Satz erstellt
-2 - fuer den spezifizierten Satz gibt es keinen aktuellen Datensatz

SIEHE AUCH
clrset, faccess, makeset, nextrec, prevrec, setsize, sfrstrec, slastrec, snextrec, sprevrec, unisort

SCOMP

NAME
scomp - zwei Zeichenketten miteinander vergleichen

SYNTAX
```
scomp (str1, str2, len)
char *str1, *str2;
int len;
```

BESCHREIBUNG
Diese Funktion vergleicht zwei Zeichenketten und gibt einen Wert zurueck, der angibt, ob sie gleich sind. Stimmen sie nicht ueberein, wird ein Wert zurueckgegeben, der angibt, welche der beiden Zeichenketten vor der anderen kommt (in aufsteigender Ordnung). len ist die Anzahl der zu vergleichenden Zeichen. Ist len gleich Null geht die Funktion davon aus, dass beide Zeichenketten auf Null enden und vergleicht sie entsprechend.

RUECKGABEWERTE
<0 - str1 kommt vor str2
0 - beide Zeichenketten stimmen ueberein
>0 - str2 kommt vor str1

SELSORT

NAME
selsort - Auswahldatei sortieren

SYNTAX
```
#include "unisel.h"
selsort (rnum, fldtbl, srtfunc)
int *fldtbl;
int (*functbl[])() = {selfunc,confunc,srtfunc,tagfunc};
int selfunc (), confunc (), srtfunc (), tagfunc ();
```

BESCHREIBUNG
Diese Funktion bildet um unisel eine Shell, die die von unisel gebildete Auswahldatei sortiert und zwar in aehnlicher Weise, in der unisort einen von einem Aufruf makeset definierten Satz von Datensaetzen sortiert. Vor Aufruf von selsort muss eine Auswahltabelle angelegt werden, die die Aufrufe von entsitm, mtchitm und/oder sfncitm enthaelt. Dann ruft selsort unisel auf, um eine Auswahldatei des Datensatztyps rnum anzulegen. Danach werden die in der Auswahldatei enthaltenen Datensaetze sortiert, wobei man mit den in functbl gegebenen Funktionen zusaetzliche Auswahlen treffen kann und moeglicherweise den von den Eintraegen in fldtbl definierten Sortierschluessel modifizieren kann. Wenn selsort zurueckkehrt, sind die Funktionen sfrstrec, snextrec, sprevrec und slastrec fuer die Wiedergewinnung der Datensaezte in sortierter Reihenfolge zu verwenden. Die mit diesen Funktionen zu verwendenden Argumente sind jedes Feld im Datensatz rnum (statt feld) und SS (eine Definition in unisel, statt rfeld).

ARGUMENTE
fldtbl - Ist ein ganzzahliges Feld der den Sortierschluessel bildenden Felder. Die Reihenfolge geht von hoher Wertigkeit zu niedriger Wertigkeit. Die Felder in der Liste muessen entweder im Zieldatensatz oder in einem direkten oder indirekten "Vater" des Zieldatensatzes sein. Die Liste besteht aus durch Kommas voneinander getrennten Listenelementen und wird mit einer -1 abgeschlossen. Jedes Element der Liste sieht folgendermassen aus:

```
feld, rfeld, rfeld, ...0,
```

feld ist der Sortierschluessel, die folgende Liste der mit 0 abgeschlossenen rfeld(er) beschreibt den Pfad zu dem, das Feld enthaltenden, Datensatz.

functbl - Es handelt sich um ein Feld aus Funktionen, die durch selsort in der vor der Sortierung liegenden Auswahlphase verschiedentlich aufgerufen werden. (Die durch unisel ausgefuehr-

te Auswahl wurde bereits vorgenommen). Mit diesen Funktionen kann man eine Untermenge von Datensaetzen spezifizieren und die von selsort verwendeten Methoden ueberschreiben. Die Funktionen und ihre Verwendung werden im folgenden noch beschrieben.

selfunc - Diese Funktion gestattet es dem Nutzerprogramm zu spezifizieren, welche Zieldatensaetze in die Sortierung einbezogen werden. Immer wenn ein neuer Zieldatensatz aufgesucht wird, ruft selsort die Auswahlfunktion. Die Funktion kann alle Verarbeitungen ausfuehren, die erforderlich sind, um zu bestimmen, ob ein Datensatz akzeptiert oder zurueckgewiesen wird. Wenn die Adresse der Auswahlfunktion im Feld functbl gleich 0 ist, wird die Funktion nicht aufgerufen und selsort waehlt alle Datensaetze entsprechend der Festlegung der vorher erfolgten Aufrufe von entsitm, mtchitm, und/oder sfncitm aus.

confunc - Diese Funktion wird von selsort nicht verwendet. An dieser Stelle ist in functbl immer eine 0 einzutragen.

srtfunc - Diese Funktion wird verwendet, um waehrend des Auswahlprozesses die ausgewaehlten Sortierfelder zu aendern. Sie kann dann verwendet werden, wenn der Sortierwert des Datensatzes von verschiedenen Feldern abhaengt.

Zwei Dinge sind dabei wichtig. Erstens muss die Feldtabelle mit einem Prototyp (Felder mit richtigem Typ und Laenge) initialisiert werden, auch wenn die srtfunc verwendet wird. Zweitens darf ein Feld von srtfunc nur durch ein Feld desselben Typs ersetzt werden. Wird anstelle von srtfunc eine Null verwendet, wird die initialisierte Feldliste immer verwendet.

tagfunc - Die Markierungsfunktion gestattet dem Nutzer fuer jeden gegebenen Datensatz die Erstellung seiner eigenen Markierung. Das ist dann sinnvoll, wenn die Datensaetze von einem berechneten Feld sortiert werden sollen. selsort uebergibt einen Pointer an die Funktion, die Funktion setzt ihre gewuenschte Markierung (die ASCII sein muss) in den Puffer und gibt die Anzahl der Zeichen zurueck. Sie muss immer Markierungen gleicher Laenge zurueckgeben. Existiert eine Markierungsfunktion, wird die Feldtabelle ignoriert und nur die zurueckgegebene Markierung wird verwendet.

RUECKGABEWERTE
-1 - Von unisel wurden keine Datensaetze ausgewaehlt
-2 - Die von unisel angelegte Auswahldatei konnte nicht geoffnet werden
0 - Normale Beendigung

SIEHE AUCH
unisel, unisort, sfrstrec, snextrec, slastrec, sprevrec, entsitm, mtchitm, sfncitm, ufsel

SEQACC

NAME
seqacc - sequentieller Zugriff auf alle Datensaetze einer Datei

SYNTAX
seqacc (rnum, richtung)

BESCHREIBUNG
Diese Funktion wird verwendet, um alle Datensaetze in einer Datei in der Reihenfolge, in der sie gespeichert wurden (d.h. in der Reihenfolge ihrer logischen Datensatznummer) zurueckzugewinnen. Die Reihenfolge ist zufaellig, da durch geloeschte Datensaetze freigewordener Speicherplatz fuer die Speicherung der neu in die Datei eingefuegten Datensaezte verwendet wird. rnum ist der wiederzugewinnende Datensatztyp und die Option richtung wird aus der folgenden Menge gewaehlt:

first - ersten Datensatz in der Datei holen
next - den in der Datei als naechsten kommenden Datensatz holen
last - den letzten Datensatz in der Datei holen
prev - den vorhergehenden Datensatz der Datei holen

RUECKGABEWERTE
0 - der spezifizierte Datensatz ist der aktuelle
-1 - der spezifizierte Datensatz existiert nicht

SETCOOK

NAME
setcook - Terminal auf cooked-Modus setzen

SYNTAX
setcook ()

BESCHREIBUNG
Diese Funktion setzt das Terminal (Standardausgabe) auf den Modus, der die uebergebenen Steuerzeichen auswertet (siehe WEGA-Programmierhandbuch, Funktion stty(1)).

RUECKGABEWERTE

keine

SETLOC

NAME
setloc - aktuellen Datensatz setzen

SYNTAX
```
setloc (rnum, loc)
long loc;
```

BESCHREIBUNG
Diese Funktion gestattet dem Nutzer, den aktuellen Datensatz fuer einen gegebenen Datensatztyp zu setzen. Das zweite Argument ist eine Datensatzposition, die vorher durch Verwendung von loc erlangt wurde. Nach dem Funktionsaufruf befindet sich der aktuelle Datensatz an der spezifizierten Position.

RUECKGABEWERTE
```
 0 - der spezifizierte Datensatz ist der aktuelle
-1 - ungueltiger Datensatztyp
-2 - ungueltige Datensatzadresse
-3 - Datensatz ist geloescht
```

SIEHE AUCH
loc

SETRAW

NAME
setraw - Terminal auf raw-Modus setzen

SYNTAX
```
setraw ()
```

BESCHREIBUNG
Diese Funktion setzt das Terminal (Standardausgabe) auf den Raw-Modus (siehe WEGA-Programmierhandbuch, Funktion stty(1)).

RUECKGABEWERTE
keine

SETSIZE

NAME
setsize - Anzahl der Datensaetze einer Menge bestimmen

SYNTAX
```
setsize (feld, rfeld, count)
long *count;
```

BESCHREIBUNG
Diese Funktion gibt die Anzahl der im spezifizierten Satz vorhandenen Datensaetze zurueck. Das Paar feld, rfeld muss mit dem Paar im vorangegangenen Aufruf von makeset uebereinstimmen.

RUECKGABEWERTE
0 - Die Anzahl der Datensaetze wird in count zurueckgegeben
-1 - Gegenwaertig ist kein solcher Satz definiert

SIEHE AUCH
clrset, faccess, makeset, nextrec, prevrec, samerec, sfrstrec, slastrec, snextrec, sprevrec, unisort

SFLDESC

NAME
sfldesc - Beschreibung eines Bildmasken-Feldes

SYNTAX
```
#include "fdesc.h"
sfldesc (sfeld, sfdsc)
int sfeld;
SFLDESC *sfdsc;
```

Es folgt die Definition der SFLDESC-Struktur:

```
#define SFLDESC struct sfdesc
struct sfdesc {
  int  sf_fld;      /* zugeordnetes Datenbank-Feld */
  int  sf_col;                    /* Spaltennummer */
  int  sf_lin;                     /* Zeilennummer */
};
```

BESCHREIBUNG
Diese Funktion gibt die Attribute des Bildmasken-Feldes sfdsc zurueck. Zuerst muss unter Verwendung von loadscr die Bildmaske geladen werden. Hier ein Beispiel fuer die Verwendung dieser Funktion:

```
#include "fdesc.h"
beis ()
{
  SFLDESC sfbuf;

  loadscr ("kunde");
  sfldesc (kunde01, &sfbuf);
printf("line=%d,col=%d\n",sfbuf.sf_lin,sfbuf.sf_col);
}
```

ARGUMENTE
sfeld - das Bildmasken-Feld
sfdsc - die Adresse einer SFLDESC-Struktur

RUECKGABEWERTE
0 - ungueltiges Bildmasken-Feld
1 - normale Rueckkehr

SIEHE AUCH
fldesc

SFNCITM

NAME
sfncitm - Eingabe eines Such-Feldes

SYNTAX
```
sfncitm (sfunc)
int (*sfunc) ();
```

BESCHREIBUNG
Diese Funktion wird verwendet, um ein Such-Feld in die unisel-Auswahltabelle einzutragen, die eine Nutzerfunktion benennt, die fuer jeden waehrend des Suchvorgangs untersuchten Datensatz aufgerufen wird. Das Argument sfunc enthaelt die Adresse der Nutzerfunktion. Die Nutzerfunktion muss 1 zurueckgeben, wenn der Datensatz akzeptiert wird, andernfalls muss 0 zurueckgegeben werden.

RUECKGABEWERTE
0 - Normale Beendigung
-1 - die Auswahltabelle ist voll

SIEHE AUCH
entsitm, mtchitm, clrfitm, uqmtch, uqsrch

SFRSTREC

NAME
sfrstrec - den ersten Datensatz einer sortierten Menge holen

SYNTAX
```
sfrstrec (feld, rfeld)
```

BESCHREIBUNG
sfrstrec macht in einem durch einen Aufruf von unisort (oder selsort, siehe unten) angelegten Satz sortierter Datensaetze den ersten zum aktuellen Datensatz. Das Paar feld, rfeld muss dem in einem vorher ergangenen Aufruf unisort stehenden Paar entsprechen. feld ist der Schluessel des Datensatzes, der an unisort uebergeben wurde, rfeld ist gleich rfeld im Aufruf unisort. Wenn als Datensatzname in unisort eine literale 0 verwendet wurde, muss diese auch als feld-Parameter fuer sfrstrec verwendet werden.

Wenn die vorher aufgerufene Funktion nicht unisort ist sondern selsort, ist die Beschreibung der zu verwendenen Argumente der Beschreibung von selsort zu entnehmen.

RUECKGABEWERTE
0 - der erste Datensatz der Menge ist der aktuelle Datensatz
-1 - gegenwaertig ist kein solcher Satz definiert
-2 - im Satz sind keine Datensaetze

SIEHE AUCH
clrset, faccess, makeset, nextrec, prevrec, samerec, setsize, slastrec, snextrec, sprevrec, unisort

SLASTREC

NAME
slastrec - den letzten Datensatz einer sortierten Menge holen

SYNTAX
slastrec (feld, rfeld)

BESCHREIBUNG
slastrec macht in einem durch einen Aufruf von unisort (oder selsort, siehe unten) angelegten Satz sortierter Datensaetze den letzten zum aktuellen Datensatz. Das Paar feld, rfeld muss dem in einem vorher ergangenen Aufruf unisort stehenden Paar entsprechen. feld ist der Schluessel des Datensatzes, der an unisort uebergeben wurde, rfeld ist gleich rfeld im Aufruf unisort. Wenn als Datensatzname in unisort eine literale 0 verwendet wurde, muss diese auch als feld-Parameter fuer sfrstrec verwendet werden.

Wenn die vorher aufgerufene Funktion nicht unisort ist sondern selsort, ist die Beschreibung der zu verwendenen Argumente der Beschreibung von selsort zu entnehmen.

RUECKGABEWERTE
0 - der letzte Datensatz der Menge ist der aktuelle Datensatz
-1 - gegenwaertig ist kein solcher Satz definiert
-2 - im Satz sind keine Datensaetze

SIEHE AUCH
clrset, faccess, makeset, nextrec, prevrec, samerec, setsize, sfrstrec, snextrec, sprevrec, unisort

SNEXTREC

NAME
snextrec - den naechsten Datensatz einer sortierten Menge holen

SYNTAX
snextrec (feld, rfeld)

BESCHREIBUNG
snextrec macht in einem durch einen Aufruf von unisort (oder selsort, siehe unten) angelegten Satz sortierter Datensaetze den naechsten zum aktuellen Datensatz. Das Paar feld, rfeld muss dem in einem vorher ergangenen Aufruf unisort stehenden Paar entsprechen. feld ist der Schluessel des Datensatzes, der an unisort uebergeben wurde, rfeld ist gleich rfeld im Aufruf unisort. Wenn als Datensatzname in unisort eine literale 0 verwendet wurde, muss diese auch als feld-Parameter fuer sfrstrec verwendet werden.

Wenn die vorher aufgerufene Funktion nicht unisort ist sondern selsort, ist die Beschreibung der zu verwendenen Argumente der Beschreibung von selsort zu entnehmen.

RUECKGABEWERTE
0 - der naechste Datensatz der Menge ist der aktuelle Datensatz
-1 - gegenwaertig ist kein solcher Satz definiert
-2 - im Satz sind keine Datensaetze

SIEHE AUCH
clrset, faccess, makeset, nextrec, prevrec, samerec, setsize, sfrstrec, slastrec, sprevrec, unisort

SPREVREC

NAME
sprevrec - den vorhergehenden Datensatz einer sortierten Menge holen

SYNTAX
sprevrec (feld, rfeld)

BESCHREIBUNG
sprevrec macht in einem durch einen Aufruf von unisort (oder selsort, siehe unten) angelegten Satz sortierter Datensaetze den vorhergehenden zum aktuellen Datensatz. Das Paar feld, rfeld muss dem in einem vorher ergangenen Aufruf unisort stehenden Paar entsprechen. feld ist der Schluessel des Datensatzes, der an unisort uebergeben wurde, rfeld ist gleich rfeld im Aufruf unisort. Wenn als Datensatzname in unisort eine literale 0 verwendet wurde, muss diese auch als feld-Parame-

ter fuer sfrstrec verwendet werden.

Wenn die vorher aufgerufene Funktion nicht unisort ist sondern selsort, ist die Beschreibung der zu verwendenen Argumente der Beschreibung von selsort zu entnehmen.

RUECKGABEWERTE
 0 - der vorhergehende Datensatz der Menge ist der aktuelle Datensatz
-1 - gegenwaertig ist kein solcher Satz definiert
-2 - im Satz sind keine Datensaetze

SIEHE AUCH
clrset, faccess, makeset, nextrec, prevrec, samerec, setsize, sfrstrec, slastrec, snextrec, unisort

STARTRANS

NAME
startrans - Aktualisierungslauf beginnen

SYNTAX
startrans ()

BESCHREIBUNG
Diese Funktion wird zum Verriegeln der Datenbank zu Beginn eines Aktualisierungslaufes verwendet. Die WDATA-Funktionen addrec, delete und fpield verwenden startrans. Alle weiteren Aktualisierungen der Datenbank werden bis zum Aufruf von endtrans unterbrochen.

Es werden zwei verschiedene Implementierungsmethoden verwendet. Welche verwendet wird, haengt davon ab, ob ein WEGA Systemaufruf zur Verfuegung steht, der die Verriegelung einer Datei unterstuetzt. Steht ein Systemaufruf zum Verriegeln einer Datei zur Verfuegung, wird unter Verwendung des Systemaufrufs Byte 0 der Datei lockfile (ddlockfile, wenn wdata.db verriegelt wird) verriegelt. Steht kein Systemaufruf zum Verriegeln zur Verfuegung, wird stattdessen die WDATA-Funktion lock_r verwendet (siehe Beschreibung von lock_r).

Die Verriegelungsdateien werden im Arbeitsverzeichnis des Nutzers angelegt. Normalerweise ist das das Verzeichnis bin. Ist jedoch $DBPATH gesetzt ist das Verzeichnis, auf das $DBPATH zeigt das Arbeitsverzeichnis. Das bedeutet indirekt, dass jeder Anwenderfall ein von allen Nutzern benutztes Arbeitsverzeichnis haben muss - entweder das fuer den Anwendungsfall verwendete Verzeichnis bin oder fuer alle dasselbe #DBPATH. Wird diese Regel nicht befolgt, erfolgt keine Steuerung der konkurierenden Arbeitsweise und die Integritaet der Datenbank wird zerstoert.

RUECKGABEWERTE
keine

SIEHE AUCH
endtrans, lock_r, lockrec, ulockrec

STRCMP

NAME
strcmp - zwei Zeichenketten miteinander vergleichen

SYNTAX
```
strcmp (str1, str2)
char *str1, *str2;
```

BESCHREIBUNG
Diese Funktion vergleicht zwei Zeichenketten und gibt einen Wert zurueck, der angibt, ob sie gleich sind. Stimmen sie nicht ueberein, wird ein Wert zurueckgegeben, der angibt, welche der beiden Zeichenketten vor der anderen kommt (in aufsteigender Ordnung). Die Funktion geht davon aus, dass beide Zeichenketten auf Null enden.

RUECKGABEWERTE
<0 - str1 kommt vor str2
0 - beide Zeichenketten stimmen ueberein
>0 - str2 kommt vor str1

UFSEL

NAME
ufsel - unisel-Funktionsauswahl

SYNTAX
```
ufsel (rec, count, tfunc)
long *count;
int (*tfunc) ();
```

BESCHREIBUNG
Diese Funktion wird in gleicher Weise wie unisel verwendet. Der Unterschied besteht darin, dass der Nutzer die fuer jeden ausgewahelten Datensatz aufzurufende Funktion spezifiziert. unisel verwendet eine Standardfunktion, mit der fuer jeden ausgewaehlten Datensatz eine Auswahldatei aktualisiert wird. ufsel gestattet dem Nutzer die Spezifizierung einer Alternative, z.B. einer Druckfunktion.

ARGUMENTE
rec - der auszuwaehlende Datensatztyp
count - die Adresse vom Typ long zur Rueckgabe der Anzahl der ausgewaehlten Datensaetze

tfunc - die Adresse einer Funktion zum Aufruf fuer jeden ausgewaehlten Datensatz

RUECKGABEWERTE
0 - normale Rueckkehr
-1 - die Datei enthaelt keine Datensaetze
-2 - eins der Such-Felder spezifizierte eine ungueltige Auswahldatei

SIEHE AUCH
unisel, selsort

ULOCKREC

NAME
ulockrec - aktuellen Datensatz freigeben

SYNTAX
ulockrec (rnum)

BESCHREIBUNG
Mit dieser Funktion wird der aktuelle Datensatz des spezifizierten Typs freigegeben. Sie hat die umgekehrte Wirkung wie lockrec.

Es werden zwei verschiedene Implementierungsmethoden verwendet. Welche verwendet wird, haengt davon ab, ob ein WEGA-Systemaufruf zur Verfuegung steht, der die Verriegelung einer Datei unterstuetzt. Steht ein Systemaufruf zum Verriegeln einer Datei zur Verfuegung, werden unter Verwendung des Systemaufrufs die entsprechenden Bytes der Datei lockfile (ddlockfile, wenn wdata.db freigegeben wird) entriegelt.

Steht kein Systemaufruf zum Verriegeln zur Verfuegung, wird stattdessen die WDATA-Funktion lock_r verwendet, um gegen einen gleichzeitigen Zugriff zu schuetzen, dann wird der entsprechende Eintrag in lockfile (ddlockfile) geloescht.

Die Verriegelungsdateien werden im Arbeitsverzeichnis des Nutzers angelegt. Normalerweise ist das das Verzeichnis bin, ist jedoch $DBPATH gesetzt ist das Verzeichnis, auf das $DBPATH zeigt das Arbeitsverzeichnis. Das bedeutet indirekt, dass jeder Anwendungsfall ein von allen Nutzern benutztes Arbeitsverzeichnis haben muss - entweder das fuer den Anwendungsfall verwendete Verzeichnis bin oder fuer alle dasselbe $DBPATH. Wird diese Regel nicht befolgt, erfolgt keine Steuerung der konkurierenden Arbeit und die Integritaet der Datenbank wird zerstoert.

Vor seiner Beendigung muss ein Prozess alle verriegelten Datensaetze entriegeln. Wenn ein Prozess mit verriegelten Datensaetzen abbricht oder abgebrochen wird

und kein Systemaufruf zum Verriegeln zur Verfuegung steht, bleiben die Datensaetze verriegelt. Diese verriegelten Datensaetze koennen nur durch Loeschen der Verriegelungsdatei (entweder lockfile oder ddlockfile) freigegeben werden. Nur wenn niemand am Anwendungssystem arbeitet, koennen diese Dateien ohne Schaden geloescht werden. Wird eine dieser Dateien geloescht, waehrend noch am Anwendungssystem gearbeitet wird, kann die Integritaet der Datenbank zerstoert werden.

RUECKGABEWERTE
keine

SIEHE AUCH
lockrec, lock_r

UNISEL

NAME
unisel - Datenbank-Datensaetze auswaehlen

SYNTAX
```
unisel (selfile, rnum, count)
char *selfile;
long *count;
```

BESCHREIBUNG
Diese Funktion dient der Auswahl von Datensaetzen auf der Grundlage von Auswahlkriterien, die durch die Aufrufe entsitm, mtchitm und sfncitm erstellt wurden. Es entsteht eine Auswahldatei mit allen Datensaetzen, die die Auswahlkriterien erfuellen. Aufruf von opensf, frstsel, nextsel und closesf gestatten dem Nutzer die Wiedergewinnung der Datensaetze. Die Funktion wird normalerweise in folgender Weise verwendet:

1. Das Nutzerprogramm beschreibt die Auswahlkriterien durch Aufruf von entsitm, mtchitm und sfncitm.
2. Der Name einer Auswahldatei wird generiert und beim Aufruf von unisel verwendet.
3. Nach erfolgreicher Rueckkehr wird die Anzahl ueberprueft und wenn diese groesser als 0 ist, wird die Auswahldatei mit opensf eroeffnet.
4. frstsel und nextsel werden fuer die Wiedergewinnung der ausgewaehlten Datensaetze verwendet und normale Funktionen auf Datensatzebene werden verwendet, um Informationen in den Datensaetzen zu aktualisieren oder an diese auszugeben.
5. closesf wird zum Schliessen der Auswahldatei verwendet, es folgt ein Aufruf von unlink.

ARGUMENTE
selfile - Die Adresse einer auf Null endenden Zeichenkette mit dem Namen der anzulegenden Auswahl-

datei fuer unisel
rnum - Der auszuwaehlende Datensatztyp
count - Die Adresse einer long-Variablen, mit der Anzahl der von unisel ausgewaehlten Datensaetze. Der Wert ist nach erfolgreicher Rueckgabe zu ueberpruefen.

RUECKGABEWERTE
0 - erfolgreiche Rueckkehr
-1 - die Datei, in der ausgewaehlt werden soll, ist leer
-2 - eins der Such-Felder hat den falschen Auswahldatei-Namen
-3 - die Auswahldatei kann nicht angelegt werden

BEMERKUNGEN
Es kann gleichzeitig nur ein Suchvorgang vorgenommen werden. Siehe uqsrch und uqmtch. unisel verwendet Pufferroutinen wie bseqacc, weshalb empfohlen wird, uniubuf im Nutzerprogramm aufzurufen, um den E/A-Puffer zu vergroessern, dessen Standardgroesse 2k betraegt.

SIEHE AUCH
entsitm, mtchitm, sfncitm, slrsitm, clrfitm, uqsrch, uqmtch, opensf, frstsel, nextsel, prevsel, closesf, iniubuf, ufsel, selsort

UNISORT

NAME
unisort - Datensaetze auswahlen und sortieren

SYNTAX
```
unisort (rnum, rfeld, fldtbl, functbl)
int fldtbl[];
int (*functbl[])() = {selfunc,confunc,srtfunc,tagfunc};
int selfunc(), confunc(), srtfunc(), tagfunc();
```

BESCHREIBUNG
Diese Funktion wird verwendet, um eine Menge von Datensaetzen auszuwaehlen und zu sortieren, die spaeter bei den Aufrufen von snextrec oder sprevrec durchsucht wird. Wenn die Funktion zurueckkehrt, ist die Sortierung und Auswahl abgeschlossen. unisort ermoeglicht Flexibilitaet bei der Verwendung der vom Nutzer definierten Funktionen.

Der Hauptteil der Arbeit von unisort erfolgt waehrend der Auswahlphase. In dieser Phase sucht unisort auch den spezifizierten Satz von Datensaetzen auf, akzeptiert oder weist die einzelnen Datensaetze zurueck und schreibt einen "Markierungs"-Datensatz in eine temporaere Datei. Ein solcher Datensatz ist einfach eine ASCII-Darstellung der zu sortierenden Felder gefolgt von der Position des Datensatzes. Nach Anlegen dieser

Datei wird eine allgemeine Funktion zum Sortieren aufgerufen, wodurch die Datensaetze entsprechend der Schluessel in den Markierungsdatensaetzen sortiert werden.

ARGUMENTE

rnum - Dieser Datensatz ist der Ausgangspunkt fuer die Sortierung. Der gewuenschte Datensatz muss vor Aufruf von unisort der aktuelle sein. unisort beginnt damit, makeset aufzurufen, wobei der Schluessel des Datensatzes rnum und rfeld benutzt wird, um den interessierenden Satz zu erkennen.

Wird die Fortsetzungsfunktion confunc verwendet, kann bei der Sortierung mehr als nur ein Auftreten des Datensatztyps rnum verwendet werden. unisort verwendet nextrec, um jeden ueberreinstimmenden zugehoerigen Datensatz im Satz aufzusuchen. Bei Erreichen des Listenendes ruft unisort die Funktion confunc auf (falls eine solche vorhanden ist), um zu ueberpruefen, ob es noch weitere Saetze gibt, aus denen Datensaetze ausgewaehlt und sortiert werden koennen.

Wenn fuer die bestimmte Reportanforderung kein expliziter Satz von Datensaetzen im Schema definiert wurde, muessen alle Datensaetze in der Datei uebergeben werden. unisort erledigt das, wenn als rnum-Parameter eine literale 0 uebergeben wird. Das ermoeglicht zur Auswahl und Sortierung die Uebergabe einer ganzen Datensatzdatei. Das dauert natuerlich erheblich laenger als die Verfolgung einer expliziten Beziehung, wenn tatsaechlich nur eine geringe Anzahl von Datensaetzen auszuwahlen ist.

rfeld - Es handelt sich um ein Bezugsfeld in einem anderen Datensatz, das sich auf den Schluessel des rnum-Datensatztyps bezieht. Es kennzeichnet den Zieldatensatz in der Sortierung und den Pfad in der Datenbank, der zum Holen dieser Datensaetze verfolgt werden muss. Wird als rnum-Parameter eine literale 0 verwendet, kann rfeld auch anstelle eines expliziten Bezugsfeldes der Name eines Feldes im Zieldatensatzes sein (z.B. das Schluesselfeld).

fldtbl - Ist ein ganzzahliges Feld der den Sortierschluessel bildenden Felder. Die Reihenfolge geht von hoher Wertigkeit zu niedriger Wertigkeit. Die Felder in der Liste muessen entweder im Zieldatensatz oder in einem direkten oder indirekten "Vater" des Zieldatensatzes sein. Die Liste besteht aus durch Kommas voneinander getrennten Listenelementen und wird mit einer

-1 abgeschlossen. Jedes Element der Liste sieht folgendermassen aus:

feld, rfeld, rfeld, ...0,

feld ist der Sortierschluessel, die folgende Liste der mit 0 abgeschlossenen rfeld(er) beschreibt den Pfad zu dem das Feld enthaltenden Datensatz.

functbl - Es handelt sich um ein Feld aus Funktionen, die durch unisort in der vor der Sortierung liegenden Auswahlphase verschiedentlich aufgerufen werden. (Die durch unisel ausgefuehrte Auswahl wurde bereits vorgenommen). Mit diesen Funktionen kann man eine Untermenge von Datensaetzen spezifizieren und die von unisort verwendeten Methoden ueberschreiben. Die Funktionen und ihre Verwendung werden im folgenden noch beschrieben.

selfunc - Diese Funktion gestattet es dem Nutzerprogramm zu spezifizieren, welche Zieldatensaetze in die Sortierung einbezogen werden. Immer wenn ein neuer Zieldatensatz aufgesucht wird, ruft unisort die Auswahlfunktion. Die Funktion kann alle Verarbeitungen ausfuehren, die erforderlich sind, um zu bestimmen, ob ein Datensatz akzeptiert oder zurueckgewiesen wird. Wenn die Adresse der Auswahlfunktion im Feld functbl gleich 0 ist, wird die Funktion nicht aufgerufen und unisort waehlt alle Datensaetze entsprechend der Festlegung der vorher erfolgten Aufrufe von entsitm, mtchitm, und/oder sfncitm aus.

confunc - Diese Funktion gestattet dem Nutzer den aktuellen rnum-Datensatz zu aendern und dann die Auswahl fortzusetzen. Damit kann der Nutzer verschiedene Saetze von Zieldatensaetzen fuer eine Sortierung zusammenfassen.

srtfunc - Diese Funktion wird verwendet, um waehrend des Auswahlprozesses die ausgewaehlten Sortierfelder zu aendern. Sie kann dann verwendet werden, wenn der Sortierwert des Datensatzes von verschiedenen Feldern abhaengt.

Zwei Dinge sind dabei wichtig. Erstens muss die Feldtabelle mit einem Prototyp (Felder mit richtigem Typ und Laenge) initialisiert werden, auch wenn die srtfunc verwendet wird. Zweitens darf ein Feld von srtfunc nur durch ein Feld desselben Typs ersetzt werden. Wird anstelle von srtfunc eine Null verwendet,

wird die initialisierte Feldliste immer verwendet.

tagfunc - Die Markierungsfunktion gestattet dem Nutzer fuer jeden gegebenen Datensatz die Erstellung seiner eigenen Markierung. Das ist dann sinnvoll, wenn die Datensaetze von einem berechneten Feld sortiert werden sollen. unisort uebergibt einen Pointer an die Funktion, die Funktion setzt ihre gewuenschte Markierung (die ASCII sein muss) in den Puffer und gibt die Anzahl der Zeichen zurueck. Sie muss immer Markierungen gleicher Laenge zurueckgeben. Existiert eine Markierungsfunktion, wird die Feldtabelle ignoriert und nur die zurueckgegebene Markierung wird verwendet.

RUECKGABEWERTE
0 - Datensaetze werden sortiert
-1 - die Auswahldatei ist leer

SIEHE AUCH
clrset, faccess, makeset, nextrec, prevrec, samerec, setsize, sfrstrec, slastrec, snextrec, sprevrec

UNLOCK_R

NAME
unlock_r - ein durch lock_r reserviertes Geraet freigeben

SYNTAX
```
unlock_r (c)
char c;
```

BESCHREIBUNG
Diese Funktion wird verwendet, um ein durch die Funktion lock_r reserviertes Geraet freizugeben. Der damals benutzte Buchstabenkode wird fuer die Bezeichnung des Geraetes verwendet. Im Ergebnis wird die Datei namens LOCKx geloescht, wobei x das fuer die Bezeichnung des Geraetes verwendete Zeichen ist. Die Datei befindet sich entweder im Arbeitsverzeichnis des Nutzers oder in $DBPATH, falls diese Umgebungsvariable gesetzt ist.

Bei frueheren Versionen wurde diese Funktion als unlock bezeichnet. Der Name wurde geaendert, um einen Zusammenhang mit lock_r herzustellen. Zur Gewaehrleistung der Kompatibilitaet wird die alte WEGA-DATA unlock Funktion im Archiv libcomp.a bereitgestellt. Soll auch weiterhin unlock verwendet werden, ist diese Bibliothek in die Ladekommandodateien aufzunehmen.

RUECKGABEWERTE

0 - das Geraet ist freigegeben
-1 - das Geraet war nicht verriegelt

SIEHE AUCH
lock_r

UQMTCH

NAME
uqmtch - Such-Feld fuer schnelles unisel

SYNTAX
```
uqmtch (s, feld, rfeld, msf, count)
char *sf, *msf;
long *count;
```

BESCHREIBUNG
Diese Funktion wird fuer das Erstellen einer Auswahldatei auf der Grundlage eines Such-Feldes verwendet. Da diese Funktion nicht die unisel-Auswahltabelle verwendet, kann sie aufgerufen werden, waehrend die Auswahltabelle aufgestellt wird.

ARGUMENTE
sf - Ein Pointer auf eine Zeichenkette mit dem Namen der anzulegenden Auswahldatei
feld - Das Datenbank-Feld, auf dem die Auswahl basiert
rfeld - Das Feld im Datensatztyp der Auswahldatei, auf das Bezug genommen wird
msf - Ein Pointer auf eine Zeichenkette mit dem Namen einer existierenden Auswahldatei, die fuer den Vergleich der Datensaetze verwendet wird
count - Die Adresse einer long-Variablen, die die Anzahl der ausgewaehlten Datensaetze zurueckgibt

RUECKGABEWERTE
0 - normale Rueckkehr
-1 - die Auswahldatei existiert nicht oder hat nicht das richtige Format
-4 - die Datei enthaelt keine Datensaetze
-6 - die Auswahldatei kann nicht angelegt werden

SIEHE AUCH
unisel, mtchitm, uqsrch, sfncitm, entsitm, clrsitm, clrfitm

UQSRCH

NAME
uqsrch - Auswahlelement fuer schnelles unisel

SYNTAX
```
#include "unisel.h"
uqsrch (sf, feld, val1, val2, modus, count)
```

```
    char *sf, *val1, *val2;
    int modus;
    long *count;
```

BESCHREIBUNG

Diese Funktion wird verwendet, um auf der Grundlage eines Auswahldatenelementes eine Auswahldatei anzulegen. Da die Funktion nicht die unisel Auswahltabelle verwendet, kann sie aufgerufen werden, waehrend die unisel-Auswahltabelle aufgebaut wird.

ARGUMENTE

- sf - Ein Pointer auf eine Zeichenkette mit dem Namen der anzulegenden Auswahldatei
- feld - Das Datenbank-Feld, auf dem die Auswahl beruht
- val1 - Die Adresse eines Puffers, der den Wert des bei der Auswahl verwendeten Feldes enthaelt
- val2 - Die Adresse eines Puffers, der einen zweiten Wert enthaelt, wenn die Auswahl ein Intervall ist; wird ansonsten auf Null gesetzt
- modus - Der Modus der Auswahl: GT, GTE, LT, LTE, EQ, NOT. Die Modi sind in unisel.h definiert.
- count - Die Adresse einer long-Variablen, die die Anzahl der ausgewaehlten Datensaetze zurueckgibt

RUECKGABEWERTE

- 0 - normaler Rueckkehr
- -1 - ungueltiges Feld
- -2 - val1 ist nicht gesetzt
- -3 - ungueltiger Modus
- -4 - die Datei enthaelt keine Datensaetze
- -6 - die Auswahldatei kann nicht angelegt werden

SIEHE AUCH

unisel, entsitm, uqmtch, sfncitm, mtchitm, clrsitm, clrfitm

VALCHAR

NAME

valchar - ein Zeichen durch Vergleich mit einem Zeichensatz auf Gueltigkeit ueberpruefen

SYNTAX

```
    valchar (c, str)
    char c, *str;
```

BESCHREIBUNG

Diese Funktion wird verwendet, um zu pruefen, ob ein Zeichen einem spezifizierten Zeichensatz angehoert. Das erste Argument ist das zu ueberpruefende Zeichen. Das zweite Argument ist eine Zeichenkette mit allen gueltigen Zeichen. Die Zeichenkette muss auf Null enden.

RUECKGABEWERTE

0 - Das Zeichen gehoert nicht zum Satz
1 - Das Zeichen gehoert zum Satz

VALSTR

NAME
valstr - Zeichenkette durch Vergleich mit Satz von Zeichenketten auf Gueltigkeit ueberpruefen

SYNTAX
```
valstr (strng, set)
char *strng, **set;
```

BESCHREIBUNG
Diese Funktion ueberprueft, ob eine uebergebene Zeichenkette einem Satz uebergebener Zeichenketten angehoert. Das erste Argument ist die zu ueberpruefende Zeichenkette. Das zweite Argument ist ein Feld von Zeichenketten, mit dem verglichen werden soll. Alle Zeichenketten muessen auf Null enden. Der Zeichenkettensatz muss auf einen Nullpointer (Wert 0) enden.

RUECKGABEWERTE
0 - die Zeichenkette stimmte mit keiner Zeichenkette im Satz ueberein
1 - die Zeichenkette stimmte mit einer Zeichenkette im Satz ueberein

YORN

NAME
yorn - Prompter fuer y/n-Antwort

SYNTAX
```
yorn (str)
char *str;
```

BESCHREIBUNG
Diese Funktion zeigt die angegebene Zeichenkette auf Zeile 22 (Spalte 1) des Bildschirms an und wartet auf die Erwiderung des Nutzers, die entweder Y oder N sein kann (y oder n werden auch akzeptiert). Erfolgt eine falsche Eingabe, wird auf Zeile 23 eine andere Meldung angezeigt, die eine Y/N-Erwiderung verlangt.

RUECKGABEWERTE
0 - N oder n wurde eingegeben
1 - Y oder y wurde eingegeben
-2 - CTRL/U wurde eingegeben

A n h a n g A

A. DAS WEGA-DATA - COBOL-INTERFACE

In diesem Abschnitt sollen die Aufruffolgen und Prozeduren beschrieben werden, die COBOL-Programme an WEGA-DATA anpassen. WEGA-DATA verwendet die CALL-USING-Syntax zum Aufruf der verschiedenen Datenbank-Funktionen. Den COBOL-Programmen stehen nicht alle im WEGA-DATA-Systemhandbuch aufgelisteten Funktionen zur Verfuegung. Es wurden nur die Routinen, die nicht C-sprachspezifisch sind, als Teil des Interface' aufgenommen.

Die Datei sub.c wird bereitgestellt, damit der COBOL-Programmierer mit COBOL-Programmen nicht zu WEGA-DATA gehoerige Subroutinen aufrufen kann. Unsere Version von sub.c ist in $WDATA/lib abgelegt. Weitere Informationen ueber die Datei sub.c sind im COBOL-Handbuch, Abschnitt "Installation" und im Abschnitt A.6 des vorliegenden Systemhandbuches enthalten.

Es gibt drei Funktionen, die zwar nicht Teil des C-Interface' sind, die aber dem COBOL-Programmierer zur Verfuegung stehen; diese sind: SETREC, READREC und WRITEREC (siehe Abscnitt A.4). Mit diesen Funktionen kann der Programmierer Datensaetze festlegen, die in einer einzigen Operation gelesen oder geschrieben werden koennen, wodurch ein feldweises Lesen entfaellt.

A.1 Kompatibilitaet der Datentypen

COBOL verwendet haeufig das BCD-Format zur Darstellung numerischer Felder. Da WEGA-DATA nicht ueber diesen Datentyp verfuegt, muessen zur Unterstuetzung von COBOL-Operationen Konvertierungen erfolgen. Im folgenden erfolgt eine Zurodnung der WEGA-DATA Datenbeschreibungen und der diese Beschreibungen unterstuetzenden COBOL-Datentypen:

WDATA	LAENGE	COBOL (VERWENDUNG)	BILDBEISPIEL *	FORMAT
STRING		DISPLAY	X(L)	
NUMERIC	1-4	COMP-1	S9(L)	
NUMERIC	5-9	COMP-3	S9(L)	
AMOUNT		COMP-3	S9(L)V99	
DATE		COMP-3	S9(6)	MMDDYY
TIME		COMP-3	S9(4)	HHMM
FLOAT		(nicht)		

* Alle numerischen Felder muessen mit Vorzeichen versehen sein. L repraesentiert den Eintrag der WEGA-DATA Schemalaenge des Datenbank-Feldes.

Im allgemeinen werden diese Konvertierungen von den WEGA-DATA-Routinen ausgefuehrt. Der Programmierer braucht sich nicht damit beschaeftigen, ausser wenn er Lese-/Schreib-

puffer fuer WEGA-DATA anlegt.

Es duerfen keine impliziten oder expliziten FILLER zwischen den Feldern in einer Sicht (view) (siehe Abschnitt A.4) oder zwischen den, ein COMB-Feld bildenden, Feldern existieren. Bei Zugriff auf eine solche Gruppe von Feldern darf WEGA-DATA keine ueberfluessiegen Bytes vorfinden. Es ist zu beachten, dass oft bei Daten auf Plaetzen 01 oder 77 implizite FILLER vorhanden sind, die dafuer sorgen sollen, dass die Datenelemente an bestimmten Adressgrenzen (Halbwortgrenzen, Wortgrenzen usw.) beginnen. Die die Sichten und COMB-Felder bildenden Felder sind in den Datensaetzen als Elementardatenelemente zu vereinbaren.

A.2 In Programme zu kopierende Dateien

Damit das Interface verwendet werden kann, muessen verschiedene Dateien in ein Quellprogramm kopiert werden. Die erste Datei, UCOBOL.H besteht aus den Eintraegen fuer Sektionen des Arbeitsspeichers, die jeder WEGA-DATA-Funktion eine Zahl zuordnen. Auch die anderen copy-Dateien werden auch aus derartigen Eintraegen bestehen. Sie koennen unter Verwendung des in Abschnitt A.2.1 beschriebenen Konvertierungsprogramms HFILECC angelegt werden.

file.h enthaelt die WEGA-DATA-Datensatz- und Feldnamen und deren zugehoerige Nummern. Diese Nummern bleiben solange konstant, wie die Datenbank existiert, so dass bei Aenderung des Aufbaus der Datenbank keine erneute Kompilierung erforderlich ist. Wird der Bildschirm-Driver verwendet, muss auch die Datei bildmaske.h fuer jede Bildmaske konvertiert werden.

A.2.1 Das Programm HFILECC

Das Programm HFILECC steht zur Konvertierung der .h-Dateien von WEGA-DATA (die im vorhergehenden Abschnitt beschrieben wurden) in Sektionseintraegen des Arbeitsspeichers zur Verfuegung. Das Programm liest die als Argument gegebene WEGA-DATA-Datei und schreibt die resultierende Ausgabe in das Standardausgabe-Geraet. Im folgenden Beispiel wird die Ausgabe an eine temporaere Datei namens temp umgelenkt.

```
HFILECC file.h > temp
```

Programm HFILECC ignoriert die Eingabezeilen, die nicht konvertiert werden koennen. Es versucht die Feld-, Datensatz- und Bildmasken-Feldnamen in COBOL Datennamen zu konvertieren, indem alle alphabetischen Zeichen in Grossbuchstaben umgewandelt und alle Unterstreichungszeichen durch eine Bindestrich ersetzt werden. Wenn der Name noch weiter editiert werden muss, wird in den Spalten 73-80 die Meldung EDIT ausgegeben. Die zugehoerige Zahl darf kein Vorzeichen besitzen. Nur die linken vier Ziffern werden geschrieben.

HFILECC kann nicht feststellen, ob der angelegte Datenname ein fuer COBOL reserviertes Wort ist. Existieren in der entstehenden Datei reservierte Worte, koennen sie geaendert werden oder der Eintrag kann, wenn er nicht benoetigt wird, geloescht werden

A.3 Die Aufruffolge

Bei Aufruf von WEGA-DATA-Routinen mit COBOL wird die folgende Syntax verwendet

```
CALL WDATA
  USING funktions-name [,status], argument-1] ...
```

Der Name WDATA ist ein COBOL-Datenname, der folgendermassen vereinbart ist:

```
01 WDATA PIC X(5) USAGE IS DISPLAY VALUE IS "WDATA".
```

funktions-name ist der Dame der WEGA-DATA-Funktion, die der Programmierer aufrufen moechte.

status ist eine COMP-1 Variable, der der Rueckgabewert der Funktion uebergeben wird. Eine solche Variable wird nicht verwendet, wenn von der aufgerufenen Funktion keine Rueckgabewerte uebergeben werden. argumente sind i.a. die fuer die betreffende Funktion in Abschnitt 10, C-Sprachen-Interface, dokumentierten Argumente. Ausnahmen werden in Abschnitt A.5 angefuehrt. Im folgenden Beispiel wird ein Aufruf an addrec gezeigt:

```
CALL WDATA
  USING ADDREC, ADD-STATUS, KUNDE, KUNDENNUMMER
```

Mit diesem Aufruf wird ein Datensatz KUNDE in die Datenbank mit dem in KUNDENNUMMER enthaltenen Schluessel eingefuegt. Der Status der Anforderung zum Einfuegen wird in ADD-STATUS zurueckgegeben. Dieser Status entspricht den Rueckgabewerten, die in Abschnitt 10.3 unter addrec angegeben sind.

Da CALL WDATA Anweisungen mit verschiedenen Argumenten verwendet werden, fuehren diese Anweisungen oft dazu, dass der Compiler Warnungen generiert. Diese Meldungen sind zwar nicht wuenschenswert, aber unvermeidlich.

A.4 E/A-Funktionen auf Datensatz-Ebene

gfield und pfield erscheinen nicht im COBOL-Interface. Sie wurden durch drei Funktionen auf Datensatzebene ersetzt: SETREC, READREC und WRITEREC.

Die Funktion SETREC legt einen Satz gleichzeitig gelesener oder geschriebener Felder an. Diese Felder muessen nicht

alle denselben WEGA-DATA Datensatztyp haben. Mit dieser Funktion kann daher der Programmierer Sichten anlegen. SETREC erfordert einen Puffer, der nachfolgend von den Aufrufen READREC und WRITEREC benutzt wird. Typ und Reihenfolge der im Puffer enthaltenen Variablen muss mit Typ und Reihenfolge der Felder uebereinstimmen, die die SETREC-Sicht bilden. Die folgende Syntax wird fuer den Aufruf von SETREC verwendet:

```
CALL WDATA
 USING SETREC, sicht, buffer, feld-liste, error-subskript
```

sicht ist eine COMP -1 Variable, der von SETREC eine Zahl zugeordnet wird. Diese Zahl ist nichtnegativ, sie kennzeichnet die Sicht (der aufgelisteten Felder) und muss bei Aufruf von READREC und WRITEREC verwendet werden. Ein negativer sicht-Wert zeigt einen Fehlerzustand an, daher ist er wie ein Statuswert zu testen. Unten werden die moeglichen negativen Werte und ihre Bedeutung beschrieben:

- -1 - In der feld-liste wurde ein ungueltiges Feld festgestellt.
- -2 - Der Typ eines Feldes ist ungueltig.
- -3 - Es sind keine Felder in der Liste
- -4 - Das Feld von WEGA-DATA mit der Feldliste ist voll.

Wenn view -1 oder -2 enthaelt, enthaelt das letzte Argument, error-subskript, das Subskript des ungueltigen Feldes der Feldliste (feld-liste). error-subskript muss eine COMPT-1 Variable sein.

buffer ist der Name der Variablen, die zum Lesen und Schreiben der Sicht verwendet wird. feld-liste ist ein Feld aus COMP-1 Variablen, das eine auf Null endende Liste von Feldern der Sicht enthaelt. Keins der Felder kann vom Typ COMB sein.

Wird mit SETREC eine Sicht angelegt und sind die die Felder enthaltenden Datensaetze in der Sicht die aktuellen, lesen und schreiben READREC und WRITEREC Daten aus der und in die Datenbank. Die Aufruffolgen sind:

```
CALL WDATA
  USING READREC, status, sicht, error-subskript

CALL WDATA
  USING WRITEREC, status, sicht, error-subskript.
```

sicht ist eine COMP-1 Variable die den nichtnegativen sicht-Identifizierer enthaelt, der durch Aufruf von SETREC erstellt wurde. Der Status Null zeigt an, dass der Aufruf erfolgreich war. REDREC ruft einfach einen Satz von gfield-Aufrufen auf und WRITEREC ruft pfield-Aufrufe auf. Deshalb ist ein gutes Verstaendnis der Funktionen gfield und pfield erforderlich - insbesondere beim Aktualisieren des Schluessels eines existierenden Datensatzes. Generell ist Vorsicht

geboten, wenn Schluesselfelder in eine Sicht gebracht werden. Unten sind die Werte aufgelistet, die status nach Aufruf dieser Funktionen enthalten kann.

Fuer READREC:

-1 - Lesezugriff fuer das Feld ist nicht zulaessig
-9 - Die Identifizierernummer ist unbekannt

Fuer WRITEREC:

-1 - Es wurde versucht, den Schluessel eines Datensatzes zu aendern, auf den gerade andere Datensaetze Bezug nehmen
-2 - Es wurde versucht, einen bereits vorhandenen Schluessel zu speichern
-3 - Es wurde versucht, eine explizite Beziehung zu einem nicht existenten Datensatz herzustellen
-4 - Schreibzugriff fuer das Feld ist unzulaessig
-5 - Der Datensatz wird von einem anderen Prozess verriegelt
-9 - sicht-Bezeichnungsnummer ist unbekannt

Fuer alle aufgelisteten Statuswerte, mit Ausnahme von -9, wurde ins error-subskript ein Feldlistensubskript gebracht. Wenn bei Ausfuehrung einer dieser beiden Funktionen einer der Datensatztypen, auf den sicht zugegriffen hat, nicht der aktuelle ist, wird das Programm abgebrochen.

A.5 Weitere WEGA-DATA-Funktionen

In diesem Abschnitt sollen die anderen von COBOL bereitgestellten WEGA-DATA-Funktionen zusammengefasst werden. Der Abschnitt enthaelt die Auffruffolgen fuer die zur Verfuegung stehenden Funktionen und die nur fuer das COBOL-Interface zutreffende Dokumentation. Dokumentation der Funktionen siehe Abschnitt 10, Syntax der Aufruffolgen siehe Abschnitt A.3.

Das Interface erwartet, dass alle fuer die Uebergabe oder den Empfang von ganzzahligen Argumenten verwendeten Datenelemente als COMP-1 vereinbart werden, ausser wenn eine anderweitige Vereinbarung angegeben wird. Wenn eine auf Null endende Zeichenkette erforderlich ist, muss diese explizit als eine Zeichenkette (USAGE IS DISPLAY) vereinbart sein, der mindestens ein Byte binaerer Nullen folgt.

ADDREC

SYNTAX

```
    CALL WDATA
      USING ADDREC, status, rnum, key
```

BTNEXT

SYNTAX
```
  CALL WDATA
    USING BTNEXT, status, feld
```

BTSRCH

SYNTAX
```
    CALL WDATA
      USING BTSRCH, status, feld, buf
```

CLEANCRT

SYNTAX
```
    CALL WDATA
      USING CLEANCRT
```

CLEARSCR

SYNTAX
```
    CALL WDATA
      USING CLEARSCR
```

CLOSBT

SYNTAX
```
    CALL WDATA
      USING CLOSBT, status, feld
```

CLOSESF

SYNTAX
```
    CALL WDATA
      USING CLOSESF, sf
```

ARGUMENTE

sf - Dieses Argument erhaelt eine Speicheradresse, die nur von anderen WEGA-DATA-Funktionen benoetigt wird. Die ausgewaehlte, aus Zeichen bestehende, Zeichenkette PICTURE des COBOL-Datennamens soll X(6) sein. Das ist mehr als noetig, ist jedoch besser als zu wenig.

CLRSITM

SYNTAX
```
    CALL WDATA
      USING CLRSITM, feld, rfeld
```

DSPLY

SYNTAX
```
    CALL WDATA
      USING DSPLY, ssfld, esfld
```

ENTSITM

SYNTAX
```
    CALL WDATA
 USING ENTSITM, status, feld, val1, val2, modus, range-flag
```

ARGUMENTE

range-flag - Wenn die Auswahl ein Bereich (val1 und val2 sind feld-Werte, die den Bereich definieren) ist, wird diese COMP-1 Variable mit 1 gleichgesetzt, ansonsten wird sie gleich Null gesetzt und das Argument val1 wird an der Position val2 wiederholt.

ERASPRMP

SYNTAX
```
    CALL WDATA
      USING ERASPRMP, ssfld, esfld
```

FACCESS

SYNTAX
```
    CALL WDATA
      USING FACCESS, status, rnum, feld
```

FRSTSEL

SYNTAX
```
    CALL WDATA
      USING FRSTSEL, status, sf
```

ARGUMENTE

sf - Dieses Argument erhaelt eine Speicheradresse, die nur von anderen WEGA-DATA-Funktionen benoetigt wird. Die ausgewaehlte, aus Zeichen bestehende, Zeichenkette PICTURE des COBOL-Datennamens soll X(6) sein. Das ist mehr als noetig, ist jedoch besser als zu wenig.

INBUF

SYNTAX
```
    CALL WDATA
      USING INBUF, status, sfld, buf
```

LOADSCR

SYNTAX
```
    CALL WDATA
      USING LOADSCR, scrn
```

LOC

SYNTAX
```
    CALL WDATA
      USING LOC, rnum, adr
```

ARGUMENTE

adr - Dieses Argument erhaelt einen aus 4 Byte bestehenden binaeren Wert. Die ausgewaehlte aus Zeichen bestehende Zeichenkette PICTURE des COBOL-Datennamens muss X(4) sein.

LOCKREC

SYNTAX
```
    CALL WDATA
      USING LOCKREC, status, rnum
```

MTCHITM

SYNTAX
```
    CALL WDATA
      USING MTCHITM, status, sel-datei, feld, rfeld
```

NEXTSEL

SYNTAX
```
    CALL WDATA
      USING NEXTSEL, status, sf
```

ARGUMENTE

sf - Dieses Argument erhaelt eine Speicheradresse, die nur von anderen WEGA-DATA-Funktionen benoetigt wird. Die ausgewaehlte, aus Zeichen bestehende, Zeichenkette PICTURE des COBOL-Datennamens soll X(6) sein. Das ist mehr als noetig, ist jedoch besser als zu wenig.

OPENSF

SYNTAX
    CALL WDATA
      USING OPENSF, status, sel-datei, sf

ARGUMENTE
    status - Es handelt sich um die moeglichen Statuswerte und deren Bedeutung.

              0 - Das Oeffnen war erfolgreich
             -1 - sel-datei konnte nicht geoeffnet werden
             -2 - sel-datei hat nicht das richtige Format
             -3 - die maximale Anzahl geoeffneter Auswahldateien wurde erreicht
             -4 - ein auf sbrk zurueckzufuehrender Fehler
    sf - Dieses Argument erhaelt eine Speicheradresse, die nur von anderen WEGA-DATA-Funktionen benoetigt wird. Die ausgewaehlte, aus Zeichen bestehende, Zeichenkette PICTURE des COBOL-Datennamens soll X(6) sein. Das ist mehr als noetig, ist jedoch besser als zu wenig.

SIEHE AUCH
    CLOSESF, FRSTSEL, NEXTSEL, PREVSEL

OPNBTS

SYNTAX
    CALL WDATA
      USING OPNBTS, status, feld

OUTBUF

SYNTAX
    CALL WDATA
      USING OUTBUF, sfld, buf

PREVSEL

SYNTAX
    CALL WDATA
      USING PREVSEL, status, sf

ARGUMENTE
    sf - Dieses Argument erhaelt eine Speicheradresse, die nur von anderen WEGA-DATA-Funktionen benoetigt wird. Die ausgewaehlte, aus Zeichen bestehende, Zeichenkette PICTURE des COBOL-Datennamens soll X(6) sein. Das ist mehr als noetig, ist jedoch besser als zu wenig.

PRIAMD

SYNTAX
    CALL WDATA
      USING PRIAMD, status, zeile

BESCHREIBUNG
    Die globale Variable, die den Login-Zugriffsebenenkode (locacl, Abschnitt 10.3) enthaelt, wird beim ersten Aufruf von PRIAMD vom Interface gesetzt. Das Interface erhaelt den Wert von der Umgebungsvariablen UUACL (siehe Abschnitt 1.1.3) locacl kann jedoch waehrend der Laufzeit unter Verwendung der Funktion SETUACL gesetzt werden.

SIEHE AUCH
    SETUACL

PRMP

SYNTAX
    CALL WDATA
      USING PRMP, x, y, str

PRTMSG

SYNTAX
    CALL WDATA
      USING PRTMSG, x, y, strng

SELSORT

SYNTAX
    CALL WDATA
      USING SELSORT, status, rnum, fldtbl

ARGUMENTE
    fldtbl - Muss der Name eines aus COMP-1 Variablen bestehenden Feldes sein. Das Feld muss mit der gewuenschten Liste von Elementen und, wie in der unisort-Dokumentation in Abschnitt 10.3 beschrieben, mit dem abschliessenden Wert -1 belegt werden. Die Liste darf nicht mehr als 48 Elemente enthalten.

SEQACC

SYNTAX
    CALL WDATA
      USING SEQACC, status, rnum, richtung

SETLOC

SYNTAX
```
CALL WDATA
  USING SETLOC, status, rnum, loc
```

ARGUMENTE
loc - Datensatzposition nach vorangegangenem Aufruf von LOC

SETUACL

SYNTAX
```
CALL WDATA
  USING SETUACL, status, uaclval
```

BESCHREIBUNG
Diese Funktion wurde erstellt, damit die globale Variable logacl (siehe Abschnitt 10.3) einem anderen, als den in der UUACL gespeicherten Umgebungsvariablen (siehe Abschnitt 1.1.3) gespeicherten Wert gleichgesetzt werden kann. Die gueltigen Werte sind 1 bis 15.

Diese Funktion gehoert nicht zum C-Interface.

ARGUMENTE
status - wird gleich 0 (Null) gesetzt, wenn uaclval einen gueltigen Zugriffsebenen-Wert enthaelt, ansonsten ist status gleich -1.

uaclval - diese COMP-1 Variable wird gleich dem gewuenschten Zugriffsebenen-Wert gesetzt.

SIEHE AUCH
PRIAMD

SFRSTREC

SYNTAX
```
CALL WDATA
  USING SFRSTREC, status, feld, rfeld
```

SLASTREC

SYNTAX
```
CALL WDATA
  USING SLASTREC, status, feld, rfeld
```

```
SNEXTREC

SYNTAX
    CALL WDATA
      USING SNEXTREC, status, feld, rfeld


SPREVREC

SYNTAX
    CALL WDATA
      USING SPREVREC, status, feld, rfeld


UACCESS

SYNTAX
    CALL WDATA
      USING UACCESS, status, rnum, key

SIEHE AUCH
    access (Abschnitt 10.3)


UDELETE

SYNTAX
    CALL WDATA
      USING UDELETE, status, rnum

SIEHE AUCH
    delete (Abschnitt 10.3)


UINPUT

SYNTAX
    CALL WDATA
      USING UINPUT, status, sfld

SIEHE AUCH
    input (Abschnitt 10.3)


ULOCK

SYNTAX
    CALL WDATA
      USING ULOCK, c

SIEHE AUCH
    lock (Abschnitt 10.3)
```

UNISEL

SYNTAX
    CALL WDATA
      USING UNISEL, status, sel-datei, xrec, count

ARGUMENTE
    count - muss eine COMP-3 Variable mit mit der aus Zeichen bestehenden Zeichenkette PICTURE S9(9) sein

ULOCKREC

SYNTAX
    CALL WDATA
      USING ULOCKREC, rnum

UOUTPUT

SYNTAX
    CALL WDATA
      USING UOUTPUT, sfld

SIEHE AUCH
    output (Abschnitt 10.3)

UUNLOCK

SYNTAX
    CALL WDATA
      USING UUNLOCK, status, c

SIEHE AUCH
    unlock (Abschnitt 10.3)

YORN

SYNTAX
    CALL WDATA
      USING YORN, status, str

A.6 Installation

Das Shell-Skript runcobol.ld (siehe Verzeichnis bin von WEGA-DATA) wird verwendet, um das WEGA-DATA/COBOL-Interface mit der COBOL-Datei zu laden: runcobol.o. Wenn runcobol.ld arbeitet, erzeugt es /bin/runcobol. Das bedeutet, dass das aktuelle /bin/runcobol geloescht wird, wenn es nicht umgespeichert wird. Das neue /bin/runcobol wird genauso verwendet wie das aktuelle /bin/runcobol.

runcobol.ld erwartet, die Datei runcobol.o im Verzeichnis /usr/lib zu finden. Es kann jedoch erforderlichenfalls nutzerspezifisch gespeichert werden. Soll sub.c modifiziert werden, wird sub.c kompiliert und dann laesst man runcobol.ld gefolgt vom neuen sub.o Pfadnamen ablaufen.

Die folgenden Schritte sind auszufuehren:

1) Das aktuelle /bin/runcobol wird umgespeichert oder in einem Backup gesichert.

2) Erforderlichenfalls wird sub.c nutzerspezifisch bearbeitet.

3) Start von runcobol.ld
   a. Ohne Modifizierung von sub.c

      runcobol.ld

   b. Mit Modifizierung von sub.c

      runcobol.ld arg

      Wobei arg der Pfadname von sub.o oder der Pfadname des Archivs ist, in dem sub.o enthalten ist.

Bemerkung: Es ist moeglich, dass dieses Interface fuer COBOL-Versionen, die aelter als 1.6 sind, nicht richtig arbeitet. Wenn bei Beendigung eines, WEGA-DATA benutzenden, COBOL-Programms ein Speicherfehler auftritt, kann die Ursache darin zu suchen sein.

A.7 Registrieren eines COBOL-Programms mit MENUH

Zur Registrierung eines COBOL-Programmes mit MENUH wird ein Shell-Skript zum Aufruf von runcobol verwendet und dann wird das Shell-Skript mit 'Executable Maintenance' registriert.

Zur Vereinfachung soll der Prozesses der Registrierung anhand eines Beispiels beschrieben werden. Zuerst wird ein Shell-Skript angelegt, das als COBLPRG bezeichnet werden soll und folgendes enthaelt:

```
runcobol CPROG
```

wobei CPROG der Name der COBOL-Objektdatei ist.

In der COBOL-Nutzerdokumentation wird erlaeutert, wie das Programm runcobol zu verwenden ist. Dabei darf man nicht das Kommando chmod vergessen, mit dem es zu einer ausfuehrbaren Datei wird.

Danach wird mit 'Executable Maintenance' das Shell-Skript

COBLPRG registriert. Der Name ist gleich dem Shell-Skript, sysrecev wird nicht verwendet. Der Standard-Programmname ist coblprg.

Nachdem das ausfuehrbare Programm registriert wurde, kann dieses neue Programm in ein Menue aufgenommen werden oder man kann es einfach vom Menue-Handler aus mittels seines Programmnamens ablaufen lassen.

Soll das COBOL-Programm mit 'Program Loading' (lfilegen) kompiliert werden, kann ein dem folgenden aehnliches Shell-Skript verwendet werden. Im angefuehrten Beispiel waere der Name des Skripts coblprg.ld und es wuerde im Verzeichnis build gespeichert werden:

```
cobol ../src/cbl/cblprg.cbl -o ../bin/CBLPRG
```

Die Besonderheiten bei der Verwendung von cobol sind der COBOL-Nutzerdokumentation zu entnehmen. Im angefuehrten Beispiel geht coblprg.ld davon aus, dass die vom COBOL Programm cblprg.cbl verwendeten COPY Datei-Pfadnamen dann gueltig sind, wenn im Verzeichnis build auf sie Bezug genommen wird. Soll vom Verzeichnis ../src/cbl Bezug auf die COPY Datei-Pfadnamen genommen werden koennen, kann man dieses mit dem Kommando cd zum aktuellen Arbeitsverzeichnis machen.

Ein Beispiel dafuer wird unten gezeigt. In diesem Beispiel wird auch die WEGA-DATA Umgebungsvariable DBPATH und dann der von cobol zurueckgegebene Exit-Status verwendet werden:

```
if test ! "$DBPATH"; then
  DBPATH = `pwd`/../bin
fi
cd ../src/cbl
cobol cblprg.cbl -o $DBPATH/CBLPRG -1 > cblprg.list
exitstat = $?
if test $exitstat -eq 0; then
  rm cblprg.list
else
  mv cblprg.list $DBPATH
  echo 'See compile listing file: cblprg.list'
fi
```

KOMBINAT VEB
ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW
»FRIEDRICH EBERT«

HEIM-ELECTRIC
EXPORT-IMPORT
Volkseigener Außenhandelsbetrieb
der Deutschen Demokratischen Republik

EAW-Automatisierungstechnik Export-Import

Storkower Straße 97
Berlin, DDR - 1055
Telefon 432010 · Telex 114158 heel dd

VEB ELEKTRO-APPARATE-WERKE BERLIN-TREPTOW
»FRIEDRICH EBERT«
Stammbetrieb des Kombinats EAW
DDR - 1193 Berlin, Hoffmannstraße 15-26
Fernruf: 2760
Fernschreiber: 0112263 eapparate bln
Drahtwort: eapparate bln

Die Angaben über technische Daten entsprechen
dem bei Redaktionsschluß vorliegenden Stand.
Änderungen im Sinne der technischen
Weiterentwicklung behalten wir uns vor.