

# MUIbase

---

Eine relationale programmierbare Datenbank  
Version 1.5

7. Juni 2000

Steffen Gutmann

---

Übersetzt von Ralph Reuchlein 1999-2000

© 1998-2000 Steffen Gutmann

Die Erlaubnis wird erteilt, wörtliche Kopien dieser Anleitung zu erstellen und zu vertreiben, vorausgesetzt der Copyright-Vermerk und dieser Erlaubnistext bleiben auf allen Kopien erhalten.

# 1 Kopierbestimmungen von MUIbase

MUIbase ist © 1998-2000 Steffen Gutmann. Alle Rechte vorbehalten.

MUIbase ist weder Public Domain noch freie Software. Wenn Sie MUIbase benutzen, müssen Sie sich nach einer kurzen Zeit registrieren lassen. Sie dürfen MUIbase ohne Registrierung für 4 Wochen installieren und benützen. Danach müssen Sie sich Ihre Kopie von MUIbase registrieren lassen oder Sie löschen MUIbase von ihrem Rechner. Erneutes Installieren verlängert nicht Ihre Lizenz.

Nach dem Registrieren erhalten Sie ihr persönliches Keyfile für MUIbase, das alle eingeschränkten Eigenschaften von MUIbase aktiviert.

Entschlüsseln des Softwareschutzes von MUIbase ist strengstens verboten.

## 1.1 Registration

Die unregistrierte Version von MUIbase ist auf verschiedene Arten eingeschränkt:

- Nur bis zu 5 Tabellen pro Projekt.
- Nur bis zu 10 Attribute pro Tabelle.
- Nur bis zu 30 programmierbare Funktionsdefinitionen pro Projekt.
- Kein Präprozessor beim Programmieren (`#`-Direktiven).
- Kein Sichern der Fensterausmaße in Projekten.
- Keine Grafiken im Struktureditor.
- Keine Karteikarten-Gruppen im Struktureditor.

Der Autor ist der Meinung, daß diese Eigenschaften MUIbase nicht unbrauchbar machen. Sie sollten fähig sein, die Leistungsfähigkeit von MUIbase innerhalb des vierwöchigen Testzeitraumes ohne diese Eigenschaften zu testen.

Nach der Registrierung erhalten Sie eine Schlüsseldatei, das die deaktivierten Eigenschaften freischaltet. Voraussichtlich wird die Schlüsseldatei auch in allen zukünftigen Versionen von MUIbase arbeiten.

### 1.1.1 Registrationsgebühr

Die Registrationsgebühr für MUIbase beträgt 60 DM. Sie wird **nur in diesen Währungen** akzeptiert:

- 40 USD (US dollar)
- 60 DEM (Deutsche Mark)
- 30 EUR (Euro)

Die Registrationsgebühr enthält kein gedrucktes Handbuch. Wenn Bitte drucken Sie Dokumentation auf Ihrem Drucker aus oder Sie können auch während dem Arbeiten mit MUIbase die Online-Hilfe (AmigaGuide oder HTML) verwenden.

### 1.1.2 Zahlungsmethoden

**Bar** Dies ist eine der bevorzugten Zahlungsmethoden und auch die billigste für Sie und mich. Legen Sie das Bargeld zwischen zwei bedruckte Papiere, um es zu verdecken. **Schicken Sie keine Münzen.**

**Eurocheque** Füllen Sie den Euroscheck auf DEM 60 oder EUR 30 aus. **Andere Währungen als DEM und EUR sind hier nicht zulässig!** Sie müssen Ihre Kartennummer auf die Rückseite des Schecks schreiben, ansonsten kann ich ihn nicht einlösen.

**Überweisung.**  
Überweisen sie DEM 60 oder EUR 30 auf mein Konto bei Bank 24, Deutschland, BLZ 380 707 24, Kontonummer 314 762 100.  
Wenn ein Internetanschluß zur Verfügung steht, dann kann das Geld auch auf mein Konto überwiesen und die Registrierung per eMail oder per Registrierungsformular auf der MUIbase Homepage <http://www.amigaworld.com/support/muibase> an mich geschickt werden. Dies ist vermutlich der beste Weg, Ihre MUIbase-Kopie zu registrieren. Ich werde mein Bankkonto fast täglich prüfen, daher werde ich Ihr Geld ziemlich schnell sehen.

**Postanweisung**  
Sie können per Postanweisung zahlen, aber Sie müssen mir das Registrierungsformular schicken oder emailen, weil es passieren kann, daß nur Ihr Name bei der Postanweisung steht, wenn ich sie erhalte. Daher benötige ich auch ihre Adresse. Fragen Sie Ihr Postamt, wenn Sie mehr darüber wissen wollen.

**Kreditkarte über ShareIt.**  
Dies ist eine gute Möglichkeit, wenn MUIbase über das Internet bestellt werden soll. Mehr über die Anwendung dieser Registrationsmethode findet man auf der Webseite von MUIbase unter <http://www.amigaworld.com/support/muibase/>

### 1.1.3 Bestellen

Der einfachste Weg, MUIbase zu bestellen, ist das Ausfüllen eines Registrierungsformulars oder man benutzt das Programm 'Register', das Sie nach allen notwendigen Informationen fragt und ein ASCII-Registrierungsformular erstellt. Leere Registrierungsformulare sind in folgenden Formaten vorhanden:

- Final Writer
- ASCII

Sie finden diese Dateien im Verzeichnis 'Register'.

Wenn Sie keinen Drucker haben, dann erzeugen sie ein ASCII-Registrierungsformular mit dem Programm 'Register' und kopieren sich die wesentlichen Informationen auf einen Bogen Papier. Die Informationen, die ich benötige, sind:

- Ihr Name
- Adresse
- email-Adresse (wenn vorhanden)
- Zahlungsmethode und Währung
- Zustellungsmethode

Wenn Sie das Registrierungsformular ausgefüllt haben, dann senden Sie es mit der Bezahlung an mich. Meine Adresse ist:

Steffen Gutmann  
Orleanstr. 47  
81667 München  
GERMANY

oder senden Sie es per eMail an:

gutmann@ieee.org

### 1.1.4 Zustellungsmethode

Wenn Sie einen Internetzugang haben, dann werde ich Ihnen die Schlüsseldatei emailen. Anderenfalls werde ich Ihnen eine 3,5"-Diskette per Post zusenden, die die Schlüsseldatei mit der neuesten MUIbase-Version enthält.

Die neueste Version von MUIbase kann immer im Aminet oder von

<http://www.amigaworld.com/support/muibase/>

heruntergeladen werden.

## 1.2 Verteilung

Sie erhalten das Recht, MUIbase an andere weiterzugeben, solange Sie das MUIbase-Archiv mit allen darin einbezogenen Dateien genauso verteilen, wie Sie es erhalten haben. **Registrierte Benutzer dürfen ihre private Datei "MUIbase.key" nicht verteilen!**

Unter keinen Umständen dürfen Sie ohne die ausdrückliche Genehmigung des Urhebers nicht mehr verlangen, als die Kopiergebühren und Zustellungskosten, die beim Verteilen von MUIbase anfallen.

Vorausgesetzt die obigen Bestimmungen werden eingehalten, wird hiermit für folgende gestattet, ohne eine schriftliche Zustimmung und ohne Lizenzgebühren das MUIbase-Archiv zu kopieren und zu verteilen:

- Für alle, die diese Software kostenlos weitergeben!
- Für alle frei zugänglichen INTERNET-Server und Mailboxen!

- Für alle Aminet-Sites
- Für alle anderen, die NICHT mehr als \$5.- für eine Diskette verlangen, die diese Software enthält!
- Für alle anderen, die NICHT mehr als \$20.- für eine CD verlangen, die diese Software enthält!

Verteilen von MUIbase-Beta-Versionen ist strengstens untersagt.

### 1.3 Verzichtserklärung

DIESE SOFTWARE WIRD VOM AUTOR UND MITWIRKENDEN "WIE SIE IST" BEREITGESTELLT UND AUF IRGENDWELCHE AUSDRÜCKLICHEN ODER EINGESCHLOSSENEN GARANTIE EINSCHLIEßLICH DER ABER NICHT NUR DARAUF BESCHRÄNKTE EINGESCHLOSSENEN GARANTIE DER MARKTFÄHIGKEIT UND EIGNUNG FÜR BESONDERE ZWECKE WIRD VERZICHTET. IN KEINEM FALL WERDEN DER AUTOR ODER MITWIRKENDE FÜR IRGENDWELCHE DIREKTEN, INDIREKTEN, ZUFÄLLIGEN, BESONDEREN, EXEMPLARISCHEN ODER FOLGERICHTIGEN SCHÄDEN (EINSCHLIEßLICH DER ABER NICHT NUR DARAUF BESCHRÄNKTE BESCHAFFUNG VON ERSETZBARE WAREN ODER DIENSTE; BEDIENUNGSUNFÄHIGKEIT, DATENVERLUST ODER FINANZIELLER SCHADEN; ODER UNTERBRECHUNG DES GESCHÄFTS), WIE AUCH BEI BEGRÜNDETER UND IRGENDEINER THEORETISCHEN VERANTWORTUNG, OB UNTER VERTRAG, STRENGSTER VERPFLICHTUNG ODER UNRECHT (EINSCHLIEßLICH FAHRLÄSSIGKEIT ODER SONSTIGES), DER IN IRGENDEINER WEISE VON DER BENUTZUNG DER SOFTWARE AUSGEHEND ENTSTEHT, VERANTWORTLICH GEMACHT, AUCH WENN DARAUF HINGEWIESEN WIRD, DAß EIN SOLCHER SCHADEN MÖGLICH IST.

Anm.d.Übersetzers: Der Text wurde so gut es ging übersetzt, aber falls jemand eine bessere Übersetzung als meine hat, der möge folgendes Original übersetzen und es mir zukommen lassen:

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 1.4 MUI

This application uses

MUI - MagicUserInterface

(c) Copyright 1993-2000 by Stefan Stuntz

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send

DM 30.- or US\$ 20.-

to

Stefan Stuntz  
Eduard-Spranger-Straße 7  
80935 München  
GERMANY

Support and online registration is available at

<http://www.sasg.com/>

## 1.5 BetterString & TextEditor

MUIbase verwendet BetterString.mcc & TextEditor.mcc, (c) 1997-2000 by Allan Odgaard. Siehe <http://www.diku.dk/students/duff/> für mehr Informationen oder für die neueste Version.

## 1.6 Zusätzliche Custom classes

MUIbase verwendet NList.mcc (C) 1996-2000 Gilles Masson.

## 1.7 Icons

Einige Icons, die im MUIbase-Paket verwendet werden, wurden vom DefaultIcons-Paket kopiert, das z.B. auf der Meeting Pearls CD 3 im Verzeichnis 'Contrib/DefaultIcons' zu finden ist. Diese Icons sind Copyright by Michael-Wolfgang Hohmann und Angela Schmidt (für eine detaillierte Copyrightbeschreibung siehe MP3).

## 2 Willkommen zu MUIbase

MUIbase ist eine schnelle und flexible Datenbank für den Amiga. Es ist für Benutzer gedacht, die Daten bequem und einfach verwalten wollen. MUIbase kann etliche Arten von Daten verwalten, z.B. Adressen, CDs, Filme oder Ihr Einkommen und Ihre Ausgaben. Die Stärke von MUIbase liegt in seiner einfachen und mächtigen Benutzerschnittstelle und seiner Programmierfähigkeiten. Letzteres macht es möglich, beinahe alles automatisch zu berechnen und zu verwalten, angefangen beim Aufsummieren von Zahlen, wie z.B. zum Berechnen der Gesamteinnahmen oder der Gesamtdauer einer CD, bis hin zum automatischen Generieren und Ausdrucken von Briefen jeder Art.

MUIbase ist der Nachfolger von AmigaBase, welches eine hierarchische Datenbank ist, die nach wie vor vorhanden, aber mit dem Erscheinen von MUIbase als hinfällig zu betrachten ist. Alle registrierten Benutzer von AmigaBase können ein kostenloses Upgrade auf MUIbase erhalten, wenn sie ihre AmigaBase-Registrationsnummer und Rücksendeadresse an mich schicken (Email bevorzugt).

MUIbase bietet folgende Features:

- Handhaben von mehreren Projekten zur gleichen Zeit
- Attribute können vom Typ Zeichenkette, Memo (mehrzeiliger Text), Ganzzahl, Fließkommazahl, Datum, Zeit, Bool, Auswahl (eines aus mehreren), Beziehung (einfache Art, um sich auf einen Datensatz einer anderen Tabelle zu beziehen), Knopf (zum Starten von MUIbase-Programmen) und virtuell (zum beiläufigen Berechnen von Werten)
- Der Zeichenkettentyp kann auch Listen von Zeichenketten, Dateien und Zeichensätze verwalten. Ein Gadget mit einem OS3.x-Datatype erlaubt das Darstellen von externen Bildern
- Unbegrenzte Anzahl von Datensätzen
- Dynamisches Laden von Datensätzen. Datensätze, die nicht benötigt werden, können aus dem Speicher entfernt werden (z.B. bei Speichermangel)
- Programmierbarkeit. Mit einer einfachen und mächtigen Programmiersprache von MUIbase können komplexe Aufgaben implementiert werden. Die Sprache enthält auch eine SELECT FROM WHERE-Abfrage für einfache und schnelle Datenabfragen
- Sortierung von Datensätzen nach jeder Kombination von Attributen
- Flexible und mächtige Such- und Filtermöglichkeiten
- Abfrageneditor, der das Eingeben und Verwalten von SELECT FROM WHERE-Abfragen ermöglicht. Die Abfragen können gespeichert und die Ergebnisse ausgedruckt werden
- Import- und Exportmöglichkeit
- Verwendet MUI als Benutzerschnittstelle. Die Schnittstelle ist vielfältig einstellbar. Externe Bilder können über die Benutzerschnittstelle eingebunden werden
- Portierbarkeit. Die Entwicklung von MUIbase wurde unter dem Gesichtspunkt der einfachen Portierbarkeit durchgeführt. Der System-/GUI-Teil wurde vom ANSI/C-Teil getrennt, so daß das Portieren von MUIbase sich nur auf das Portieren vom System-/GUI-Teil beschränkt. Der Autor untersucht gerade die Möglichkeit einer Umsetzung der Benutzeroberfläche in Java. Dies würde die Datenbank für nahezu alle Betriebssysteme wie Linux, Windows und MacOS verfügbar machen.

Die unregistrierte Version von MUIbase ist auf verschiedene Arten eingeschränkt. Eine Liste über die Einschränkungen in der unregistrierten Version lese man bei Abschnitt 1.1 [Registration], Seite 1 nach.



## 3 Einführung

Dieses Kapitel beschreibt, wie man MUIbase auf Ihrem Computer installiert, welche Hard- und Software benötigt wird und wie man MUIbase startet.

### 3.1 MUIbase installieren

Zum Installieren von MUIbase auf Ihrer Festplatte benötigen Sie den Commodore Installer. Dieses Programm können Sie im Aminet unter dem Verzeichnis `‘/pub/aminet/util/misc’` finden. Der Installer ist vielleicht auch im MUIbase-Archiv zu finden. Seien Sie sicher, daß Sie den Installer mit der Version 43.3 oder höher verwenden, ansonsten könnte das Installationskript fehlschlagen.

Vor dem Installieren sollten Sie sicherstellen, daß Ihr Computer und die Systemsoftware kompatibel zu MUIbase ist. Siehe dazu die Abschnitte Benötigte Hardware und Benötigte Software.

#### 3.1.1 Benötigte Hardware

Sie benötigen einen Amiga mit mind. einem 68020er Prozessor, 2 MB Arbeitsspeicher und eine Festplatte mit mindestens 2 MB freiem Speicherplatz. Für größere Anwendungen ist mehr Festplattenplatz und Arbeitsspeicher notwendig.

#### 3.1.2 Benötigte Software

MUIbase braucht OS 3.0 oder höher. Es kann auch unter OS 2.x laufen, aber es gibt keine Garantie, daß dann alle Features richtig arbeiten.

Zusätzlich verlangt MUIbase, daß MUI 3.8 oder höher auf Ihrem System installiert ist.

#### 3.1.3 Installation starten

Wenn Sie MUIbase als Archiv erhalten haben, dann entpacken Sie das Archiv in ein temporäres Verzeichnis. Entpacken Sie es nicht ins Zielverzeichnis!

Doppelklicken Sie auf das MUIbase-Installerskript `‘Install-MUIbase’` und folgen sie den Anweisungen. Das Skript fragt nach einem Verzeichnis, wohin die Software installiert werden soll. Geben Sie hier nicht das Verzeichnis an, wo Sie das MUIbase-Archiv entpackt haben.

Das Skript kann auch eine vorhandene MUIbase-Installation auf den neuesten Stand bringen. In diesem Fall geben Sie das Verzeichnis an, in dem Sie MUIbase zuvor schon installiert haben. Während dem Aktualisieren der vorhandenen MUIbase-Installation werden alle notwendigen Dateien durch neue ersetzt. Dies schließt auch die Beispielprojekte im `‘Demos’`-Verzeichnis ein. Aus diesem Grund sollten Sie weder in diesem Verzeichnis Projekte plazieren, noch eines der Beispielprojekte benutzen, um ihre Daten zu verwalten!

Nach einer erfolgreichen Installation können sie das MUIbase-Archiv von ihrem System nehmen und es irgendwo speichern, z.B. auf einer Diskette. (ANM.D.ÜBERS.: Das MUIbase-Archiv ist zu

groß und paßt daher schon lange nicht mehr auf eine normale DD-Diskette. Auf ein HD-Floppy-Laufwerk mit 1.76MB paßt es noch locker drauf.)

### 3.1.4 Schlüsseldatei

Wenn Sie ein registrierter Benutzer von MUIbase sind und Sie eine Schlüsseldatei erhalten haben, dann müssen Sie die Schlüsseldatei in eines der folgenden Verzeichnisse kopieren bzw. verschieben:

- ‘Verzeichnis-in-dem-das-MUIbase-Programm-liegt’
- ‘MUIbase:’
- ‘KEYS:’
- ‘KEYFILES:’
- ‘S:’

MUIbase wird in diesen Verzeichnissen die Datei ‘**MUIbase.key**’ suchen und wenn sie gefunden wurde (und die Schlüsseldatei gültig ist), dann werden alle eingeschränkten Eigenschaften freigeschaltet.

Bitte nicht vergessen, daß die Schlüsseldatei Ihre persönlichen Daten enthält und Sie die Datei nicht weitergeben dürfen. Sie dürfen auch die Datei nicht in irgendeiner Weise verändern, ansonsten wird MUIbase abstürzen.

Bitte bewahren Sie eine Sicherheitskopie der Schlüsseldatei auf einer oder zwei Disketten auf.

## 3.2 MUIbase starten

MUIbase kann von der Workbench oder vom CLI aus gestartet werden. Von der Workbench aus klicken Sie doppelt auf das MUIbase-Piktogramm. Sie können auch auf ein MUIbase-Projekt-Piktogramm doppelklicken. Dies startet MUIbase und das gewählte Projekt wird automatisch von MUIbase geladen. Es ist auch möglich, mehrere MUIbase-Projekte durch Mehrfachauswahl mit der SHIFT-Taste auszuwählen und durch Doppelklicken auf das letzte Projekt alle Projekte in MUIbase zu laden.

Von CLI aus geben Sie *MUIbase [Datei1 ...]* ein, wobei *Datei1 ...* optionale Projekte sind, die von MUIbase geladen werden sollen.

## 3.3 MUIbase beenden

Um MUIbase zu beenden, wählen Sie den Menüpunkt ‘Projekt – Beenden’ oder schließen Sie alle geöffneten Projekte.

## 4 Tutorial

### Erstellung einer Stammbaum-Datenbank

Dieses Kapitel ist ein kleines Tutorial, welches beschreibt, wie die Hauptelemente von MUIbase arbeiten. Innerhalb des Tutorials wird ein kleines Projekt entwickelt, das Ihnen erlaubt, Ihren Stammbaum zu verwalten. Das nach dem Durchführen aller Schritte entstandene Projekt dieses Tutorials können Sie im ‘Demos’-Verzeichnis Ihrer MUIbase-Installation finden.

### 4.1 Wie MUIbase arbeitet

Man kann sagen, daß MUIbase in zwei verschiedenen Modi arbeitet: Datensatzbearbeitungs- und Strukturbearbeitungsmodus.

Im Datensatzbearbeitungsmodus ändern, löschen und fügen Sie Datensätze hinzu.

Der Struktureditor erlaubt Ihnen das Bearbeiten des Aussehens Ihrer Datenbank und welche Tabellen und Felder es enthalten soll.

Neben diesen beiden gibt es noch den Programmeditor, in dem Sie Programmfunktionen schreiben können, die entweder automatisch ausgeführt werden, wenn Sie Daten in ein Feld eingeben oder dann, wenn Sie einen Knopf drücken.

### 4.2 Ein Projekt beginnen: Der Struktureditor

Um eine Datenbank zu erstellen, müssen Sie zuerst dessen Inhalt festlegen. In MUIbase wird dies im Struktureditor durchgeführt. Um zum Struktureditor zu gelangen, drücken Sie **RAMIGA-s** (rechte Amiga-Taste und den Buchstaben ‘s’) oder durch Auswählen des Menüpunkts ‘**Struktureditor** ...’ aus dem ‘Projekt’-Menü. Sie werden drei verschiedene Bereiche vorfinden:

‘Tabellen’

In ‘Tabellen’ ändern, löschen und fügen Sie die Tabellen hinzu, die Sie benötigen.

‘Felder’

In ‘Felder’ ändern, löschen und fügen Sie Felder hinzu. Diese Felder gehören jeweils zu einer oben erwähnten Tabelle.

‘Anzeige’

In ‘Anzeige’ legen Sie das Aussehen Ihrer Datenbank fest, d.h. wie es dargestellt werden soll.

### 4.3 Hinzufügen einer Tabelle

Als erstes benötigen wir eine Tabelle. Dazu drückt man den Knopf ‘Neu’ unterhalb der Liste im Bereich ‘Tabelle’. Sie erhalten dann ein Fenster, das Sie nach Daten fragt:

‘Name’

Hier geben Sie den Namen der Tabelle an.

Der Name muß mit einem Großbuchstaben beginnen und kann bis zu 20 Zeichen lang sein. Der Name kann später geändert werden. In diesem Tutorial setzen wir den Namen auf **'Persontable'**<sup>1</sup>, da die Tabelle alle Namen der Personen speichern soll.

**'Anzahl der Datensätze'**

Eine Tabelle kann entweder nur aus genau einem oder unbegrenzt vielen Datensätzen bestehen. In diesem Fall setzen wir auf unbegrenzt, da wir mehr als nur eine Person hinzufügen wollen.

**'Auslösefunktionen'**

Jeder Aufruf des Benutzers, einen Datensatz hinzuzufügen oder zu löschen, kann durch eine Programmfunktion geregelt werden. An dieser Stelle setzt man diese Funktion, die jeweils aufgerufen wird. Nachdem wir bis jetzt noch keine Programmfunktion geschrieben haben, wird ein Blick in die Popup-Listen nichts anzeigen.

Nachdem alles getan ist, drückt man den Knopf **'Ok'** und wir haben unsere erste Tabelle namens **Persontable**.

## 4.4 Hinzufügen eines Feldes

Jetzt brauchen wir ein Textfeld für diese Tabelle. Dazu drückt man den Knopf **'Neu'** im Bereich **'Felder'**. Auch Felder benötigen einige Einstellungen:

**'Name'** Wie bei einer Tabelle ist der erste Buchstabe ein Großbuchstabe und maximal 20 Zeichen sind zulässig. Dieses Feld wird auf **'Name'** gesetzt, da es die Namen der Personen speichern soll, die wir hinzufügen werden.

**'Typ'** Hier wählen wir aus, welchen Typ dieses Feld haben soll. Es gibt hier eine Menge verschiedener Typen, aber für dieses Feld benötigen wir ein Zeichenkettenfeld.

**'max. Länge'** Hier müssen Sie die maximale Anzahl der Zeichen angeben, die ein Benutzer für die Zeichenkette eingeben kann. Wir setzen dies auf 30.

**'Vorgabewert'** Es ist möglich, für einige Felder einen Vorgabewert für jeden neuen hinzugefügten Datensatz zu setzen. In diesem Einstellfeld gibt man diesen Wert an. Wir lassen diese Zeile leer.

**'Auslösefunktion'** Ein Feld kann auch eine Programmfunktion auslösen, die ausgeführt wird. Zum Beispiel können Sie ein Programm angeben, das nach einer Eingabe eines Namens prüft, ob der Name schon existiert.

## 4.5 Darstellen des Projekts

Nach dem **'Ok'** bemerken Sie einige Veränderungen im Bereich **'Anzeige'**. Wechseln Sie über das Auswahlfeld oben im Anzeigebereich zum **'Hauptfenster'**. Nun sehen Sie, was das Hauptfenster beinhaltet, derzeit ist es **'Persontable'**. Wechseln Sie nun mit dem Auswahlfeld wieder zurück

---

<sup>1</sup> Anm.d.Übersetzers: Die englischen Namen der Tabellen, Felder und sonstige Texte werden beibehalten, da sonst das ganze Projekt umgeschrieben werden müßte.

zum ‘Tabellenschema’ und Sie können sehen, wie die Tabelle ‘**Persontable**’ dargestellt wird. Im Moment wird sie nur als ein Panel mit einem Feld angezeigt.

Nun doppelklicken Sie auf das ‘**Panel(Persontable)**’ am Beginn der Liste im Anzeigebereich und ein Fenster sollte erscheinen, in dem Sie einstellen können, wie das Panel angezeigt werden soll:

‘**Überschrift**’

Der Name einer Tabelle kann vom echten Namen abweichen. Unsere Tabelle heißt ‘**Persontable**’, aber wir können es auf ‘**THIS IS THE TABLE PERSONTABLE!**’ setzen, wenn wir es besser finden.

‘**Hintergrund**’

Der Hintergrund kann auf das eingestellt werden, was zu Ihrem Geschmack paßt.

‘**Gadgets**’ Hier legen wir fest, welche Knöpfe das Panel haben soll.

Nach dem ‘OK’ doppelklicken wir im Anzeigebereich in der Liste auf ‘**Name**’. Dies öffnet ein Fenster, in dem die Einstellungen für die Darstellung des Zeichenkettenfeldes ‘**Name**’ vorgenommen werden.

‘**Überschrift**’

Analog zum Panel wird die hier eingegebene Zeichenkette dargestellt, wenn MUIbase im Datensatzmodus ist.

‘**Tastenkürzel**’

Hier können Sie einen Buchstaben definieren, der zusammen mit *RAMIGA* verwendet wird, um zu diesem Feld zu springen, wenn MUIbase im Datensatzmodus ist.

‘**Home**’ Veranlaßt den Cursor, immer in dieses Feld zu springen, wenn ein neuer Datensatz angelegt wird. In unserem Fall werden wir immer oder meistens in einem neuen Datensatz den Namen zuerst eingeben, deshalb wird es gesetzt.

‘**Nur lesen?**’

Dieses Feld wird gesetzt, wenn es nur lesbar sein soll. Lassen Sie es ungesetzt.

‘**Gewichtung**’

Entscheidet darüber, wieviel vom Feld sichtbar sein soll, wenn es den Platz mit anderen Feldern teilen soll. Wenn z.B. drei Zeichenketten mit je 50 Zeichen in einem Fenster stehen, das nur Platz für 100 Zeichen hat, dann entscheidet diese Zahl, wieviel Platz die Zeichenkette relativ zu den anderen erhält. Lassen Sie es bei 100.

‘**Hintergrund**’

Analog zu Panel.

‘**Sprechblasenhilfe**’

Hier wird Text angegeben, der für den Benutzer hilfreich sein kann. Die Sprechblase erscheint, wenn Sie die Maus für einige Sekunden über dem Feld halten. Setzen Sie dieses Feld auf ‘**Wenn Sie Hilfe brauchen, rufen Sie den Autor unter 112 an**’.

Verlassen Sie den Struktureditor (*RAMIGA-s* oder ‘**Struktureditor verlassen**’ im Menü ‘**Projekt**’) und kehren in den Datensatzmodus zurück, um zu sehen, wie die Datenbank aussieht. Sie werden eine Überschrift sehen, die die Zeichenkette beinhaltet, den Sie im Anzeigebereich für das Panel eingegeben haben. Der Datensatzzähler sollte nun ‘**#0/0**’ anzeigen, da wir noch keine Datensätze eingefügt haben. Dahinter ist der Filterknopf und zwei Fortschrittknöpfe. Unter all dem sollten Sie ‘**Name**’ und den Text sehen, den Sie im Anzeigebereich für dieses Feld angegeben

haben. Wenn Sie keinen Text im Anzeigebereich geändert haben, dann wird das Panel den Namen **‘Persontable’** und das Zeichenkettenfeld den Namen **‘Name’** tragen. Bewegen Sie nun die Maus über das Feld **‘Name’** und lassen Sie sie für ein paar Sekunden verharren. Wenn Sie etwas für die Sprechblasenhilfe eingegeben haben, werden Sie dessen Text in einer Sprechblase zu sehen bekommen.

## 4.6 Hinzufügen von zwei Datensatzbeziehungen

Jetzt werden wir zwei Datensatzbeziehungen hinzufügen. Beziehungsfelder weichen ein wenig von den anderen Feldern ab. Wie ihr Name schon andeutet, beziehen sie sich auf andere Datensätze. Sie werden dies besser verstehen, wenn wir es kurz selbst ausprobieren.

Wechseln Sie wieder in den Struktureditor und fügen zwei weitere Felder zu **‘Persontable’** hinzu. Drücken Sie **‘Neu’** im Bereich Felder, benennen es **‘Father’** und ändern den Typ auf **‘Beziehung’**. Eine Datensatzbeziehung hat nur eine Einstellung:

**‘Stelle Beziehung her zu’**

Legt die Tabelle fest, auf die sich das Feld beziehen soll. Es sollte schon auf **‘Persontable’** verweisen. Lassen Sie es unverändert und drücken Sie **‘Ok’**.

Fügen Sie ein weiteres Feld über **‘Neu’** im Bereich Felder hinzu und nennen Sie es **‘Mother’**. Der Typ sollte auch auf **‘Beziehung’** gesetzt werden und zeigt auf die Tabelle **‘Persontable’**.

Wie Sie vielleicht schon bemerkt haben sollten, sind nun drei Felder im Anzeigebereich sichtbar. Klicken Sie einmal auf **‘Father’** und dann auf die Knöpfe **‘Rauf’** und **‘Runter’**, die gleich links davon angeordnet sind. Dies verändert die Position des Feldes **‘Father’** in der Datensatzansicht. Setzen Sie **‘Father’** an den Anfang, **‘Name’** in die Mitte und **‘Mother’** an das Ende.

Nun müssen wir den Inhalt der Beziehungsfelder **‘Father’** und **‘Mother’** setzen, der aus den bezogenen Datensätzen angezeigt werden soll. Doppelklicken Sie auf **‘Father’** im Anzeigebereich und wählen Sie **‘Extras’**. Dort wählen wir die Zeichenkette **‘Name’** aus, die angezeigt werden soll und drücken **‘Ok’**. Diese Vorgehensweise wiederholen wir mit **‘Mother’**.

## 4.7 Datensätze hinzufügen

Jetzt sollten wir einige Datensätze hinzufügen. Verlassen Sie den Struktureditor. Um einen neuen Datensatz hinzuzufügen, drücken Sie einfach **RAMIGA-n** oder wählen Sie **‘Neuer Datensatz’** aus dem Menü **‘Tabelle’**. Der Cursor sollte nun automatisch in das Feld springen, bei dem wir vorher im Anzeigebereich des Struktureditors **‘Home’** gesetzt haben. Fügen Sie nun zwei Datensätze ein: einen mit dem Namen ihres Vaters in **‘Name’** und einen mit dem Namen ihrer Mutter auch im Feld **‘Name’**<sup>2</sup>. Danach fügen Sie einen weiteren Datensatz ein, der im Feld **‘Name’** Ihren Namen erhalten soll.

Nun kommen wir zur Erklärung der Beziehungsfelder. Drücken Sie auf den Listenansichtsknopf bei **‘Father’** und wir erhalten eine Liste aller Datensätze, auf die das Beziehungsfeld verweisen

---

<sup>2</sup> Anm.d.Übersetzers: Nach dem Sie den Namen Ihres Vaters eingegeben haben, müssen Sie einen neuen Datensatz wie beschrieben hinzufügen, um den Namen Ihrer Mutter eingeben zu können.

kann. Wählen Sie den Namen ihres Vaters und führen das gleiche mit dem Listenansichtsfenster Ihrer Mutter durch.

Jetzt sollten wir drei Datensätze mit Ihnen, Ihrem Vater und Ihrer Mutter haben. In Ihrem Datensatz sollte dann oben im Feld **'Father'** der Name Ihres Vaters und im unterem Feld **'Mother'** der Name Ihrer Mutter stehen. Sie können nun die drei Datensätze durchblättern, wenn Sie **ALT** zusammen mit **Cursor hoch/runter** drücken.

Aber halt! Sie würden sagen, daß Ihre Eltern auch Eltern haben/hatten. Daher fügen Sie weitere vier Datensätze für die dritte Generation ein. Fügen Sie einfach einen Datensatz nach dem anderen ein und tragen jeweils die Namen in **'Name'** ein. Wenn Sie die Namen nicht mehr wissen, dann tragen Sie **'Vaters Vater'**, **'Mutters Vater'** oder ähnliches ein. Nun blättern Sie durch die Datensätze und setzen zu den einzelnen Datensätzen jeweils dazugehörend Vater und Mutter. Wenn Sie das erledigt haben, müssen Sie sieben Datensätze haben: Ihren Datensatz, zwei Ihrer Eltern und vier Ihrer Großeltern.

## 4.8 Filter

Nachdem wir nun einige Datensätze zum Verarbeiten haben, probieren wir die Filterfunktion aus. Der Filter kann Datensätze herausfiltern, die nicht angezeigt werden sollen, aber sie bleiben dennoch in der Datenbank, da sie nicht sichtbar sind.

Um den Filter einzugeben, drücken Sie **LAMIGA-f** oder wählen **'Ändere Filter...'** aus dem Menü **'Tabelle'**. Sie werden nun ein etwas merkwürdig aussehendes Fenster mit Unmengen von Operatoren sehen. Diese werden verwendet, um die Bedingungen zu setzen, die ein Datensatz erfüllen muß, damit er angezeigt wird.

In unserem kleinen Beispiel verwenden wir den Befehl **LIKE**, welcher einen Mustervergleich mit einem Feld ermöglicht. Drücken Sie rechts einmal auf den Knopf **LIKE**, doppelklicken auf den Eintrag **'Name'** in der linken Liste und (**LIKE Name**) sollte nun im unteren Textfeld über den Knöpfen **'Ok'** und **'Abbrechen'** zu sehen sein. Fügen Sie nun **"\*a\*"** ein, so daß die ganze Zeichenkette schließlich (**LIKE Name "\*a\*"**) enthält. Dies bedeutet, daß MUIbase nur die Datensätze anzeigt, die den Buchstaben **'a'** im Feld **'Name'** enthalten.

Drücken Sie nun **'Ok'** und Sie werden bemerken, daß Datensätze ohne **'a'** in **'Name'** nicht mehr sichtbar sind. Nachdem der Buchstabe **'a'** in den meisten Sprachen und Namen häufig verwendet wird, dürften möglicherweise alle Datensätze angezeigt werden, aber Sie können andere Buchstaben ausprobieren, um die Filterfunktion besser zu verstehen. Gehen Sie in den Datensatzmodus, wenn Sie dies nun erledigt haben.

Wie weiter oben schon erwähnt, existiert ein Knopf im Panel, das **'F'** enthält. Dieses **'F'** zeigt an, ob der Filter aktiviert ist oder nicht. Schalten Sie den Filter aus, wenn Sie mit dem Testen fertig sind, dann werden wieder alle Datensätze sichtbar.

## 4.9 Abfragen

Nachdem Sie die Filterfunktion kennengelernt haben, werden wir uns mit dem Abfragefeature von MUIbase beschäftigen. Abfragen können verwendet werden, um Informationen aus einer Datenbank zu erhalten, die bestimmten Kriterien genügen.

Wählen Sie **‘Abfragen...’** aus dem Menü **‘Programm’** oder drücken Sie **‘RAMIGA-\***, um den Abfrageeditor zu öffnen. Es erscheint ein Fenster mit einigen Knöpfen am oberen Rand und zwei größeren Bereichen darunter. Das Textfeld oben links dient dem Eintragen eines Namens, unter dem Sie die Abfrage speichern wollen.

**‘Ausführen...’**

Kompiliert und führt die Abfrage aus. Es arbeitet sich durch die Datenbank und zeigt die Daten gemäß ihren Angaben an.

**‘Drucken...’**

Druckt das Ergebnis der Abfrage aus.

**‘Laden und Speichern’**

Lädt und Speichert jede ihrer Abfragen.

Das erste große Feld dient dem Eintragen der Abfrage und das zweite große Feld zeigt das Ergebnis der Abfrage an.

Nun werden wir eine Liste ausgeben lassen, die die Personen anzeigt, die wir zuvor per Filter ermittelt haben. Geben Sie **‘Personen, die ein a im Namen haben’** in das Textfeld oben rechts ein. Im oberen großen Feld geben Sie folgendes ein:

```
SELECT Name FROM Persontable WHERE (LIKE Name "*a*")
```

Wenn Sie nun die Abfrage entweder mit **RAMIGA-r** oder über den Knopf **‘Ausführen...’** abarbeiten lassen, wird MUIbase eine Liste aller Personen ausgeben, die ein **‘a’** im Namen haben. Ändern Sie den Buchstaben, um verschiedene Ergebnisse zu erhalten.

Wir führen nun den Befehl **AND** ein. Wählen Sie den Listenansichtknopf vom linken oberen Textfeld, drücken Sie **‘Neu’** und benennen es **‘Personen, die ein a und s im Namen haben’**. Nun geben Sie ein:

```
SELECT Name FROM Persontable WHERE  
(AND (LIKE Name "*a*") (LIKE Name "*s*"))
```

Beachten Sie, daß wir immernoch den Befehl **LIKE** zur Auswahl von Datensätzen verwenden, die die Buchstaben **‘a’** und **‘s’** im Namen haben, aber der Befehl **AND** fordert, daß beide Kriterien mit **LIKE** erfüllt sein müssen. Deshalb sind nach dem Ausführen der Abfrage nur Datensätze sichtbar, die die Buchstaben **‘a’** und **‘s’** im Namen haben.

## 4.10 Hinzufügen einer Tabelle mit einem mehrzeiligen Text und einem Knopf

Bisher wurden zwei Arten zur Auswahl und Anzeige der Datenbank aufgezeigt. Ein anderer Weg zum Darstellen kann über ein Programm geschehen. Um Daten darzustellen, können wir einen Feldtyp **‘mehrzeiliger Text’** verwenden.

Wechseln Sie in den Struktureditor und wählen Sie **‘Neu’** im Bereich **‘Tabelle’**. Benennen Sie die Tabelle **‘Controlltable’** und setzen Sie die Anzahl der Datensätze auf **‘genau ein’**. Schließen Sie das Fenster mit **‘Ok’**. Klicken und halten Sie den Mausknopf auf der neuen Tabelle. Nun



verschieben Sie den Eintrag etwas über die Mitte von ‘**Persontable**’ und lassen den Mausknopf los. Im Bereich Tabelle erscheint nun ‘**Controltable**’ oben und ‘**Persontable**’ darunter.

Stellen Sie sicher, daß ‘**Controltable**’ aktiviert ist und wählen ‘**Neu**’ aus dem Bereich Felder. Ändern Sie den Typ auf ‘**mehrzeiliger Text**’ und geben Sie dem Feld den Namen ‘**Resultmemo**’. Drücken Sie ‘**Ok**’ und fügen ein weiteres Feld zu ‘**Controltable**’ hinzu, indem Sie nochmal auf ‘**Neu**’ im Bereich Felder klicken. Diesmal nennen wir das Feld ‘**Pedigree**’ und setzen den Typ auf ‘**Knopf**’.

Um der Datenbank ein besseres Aussehen zu geben, klicken Sie einmal auf ‘**Pedigree**’ im Anzeigebereich und schieben es nach oben, indem Sie den Knopf ‘**Rauf**’ einmal drucken.

## 4.11 MUIbase programmieren, um einen Stammbaum zu erzeugen

Wir haben nun einen Knopf, um ein Programm zu starten und einen mehrzeiligen Text, um Daten darin darzustellen. Nun ist es Zeit, den Programmeditor zu starten. Dies geschieht entweder durch Drücken von *RAMIGA-p* oder durch Auswählen von ‘**Ändern...**’ aus dem Menü ‘**Programm**’. Der Editor hat drei Knöpfe:

‘**Kompilieren & Schließen**’

MUIbase kompiliert das Programm und verläßt den Programmeditor.

‘**Kompilieren**’

Kompiliert das Programm, bleibt aber im Programmeditor.

‘**Rückgängig machen**’

Macht alle Änderung seit dem Öffnen des Programmeditors rückgängig.

Nachdem alle Programmfunktionen, die Sie schreiben, in diesem einen Fenster verbleiben, müssen wir sie voneinander trennen. In MUIbase wird dies durch den Befehl DEFUN erreicht. Im folgenden Beispiel ist alles zwischen zwei runden Klammern ein Teil der Funktion **stammbaum**:

```
(DEFUN stammbaum ()
; Dies ist DEFUN's abschließende Klammer
)
```

Mit diesem Sachverhalt geben wir unsere erste Funktion ein, die einen Stammbaum der gerade angezeigten Person aus der Datenbank im Feld ‘**Resultmemo**’ anzeigt. Folgende Funktion ‘**stammbaum**’ besteht genau genommen aus drei Funktionen:

- **pedigree**, das ‘**Controltable.Resultmemo**’ setzt, in dem eine andere Funktion aufgerufen wird.
- **getpedigree**, das den Stammbaum in einer Liste zusammenträgt.
- **pedigree2memo**, das diese Liste in einen mehrzeiligen Text umwandelt.

```
; Das Programm pedigree
```

```

(DEFUN pedigree ()
  (SETQ Controltable.Resultmemo
    (pedigree2memo (getpedigree Persontable NIL) 0 3)
  )
)

; Das Programm getpedigree

(DEFUN getpedigree (person:Persontable level:INT)
  (IF (AND person (OR (NULL level) (> level 0)))
    (LIST person.Name
      (getpedigree person.Father (1- level))
      (getpedigree person.Mother (1- level))
    )
  )
)

; Das Programm pedigree2memo

(DEFUN pedigree2memo (pedigree:LIST indent:INT level:INT)
  (IF (> level 0)
    (+
      (pedigree2memo (NTH 1 pedigree) (+ indent 8) (1- level))
      (IF pedigree (SPRINTF "%s%s\n" indent "" (FIRST pedigree)) "\n")
      (pedigree2memo (NTH 2 pedigree) (+ indent 8) (1- level))
    )
    ""
  )
)

```

Dies sind die fertigen Programmfunktionen. Geben Sie sie ein und achten Sie darauf, daß alle Klammern auch da stehen, wo sie sein sollen. Zu viele oder zu wenig Klammern sind beliebte Fehler, wenn MUIbase Ihre Programme vorkompiliert. Die Fehlermeldung von MUIbase wird in diesem Fall vermutlich **'Syntax Error'** lauten. Drücken Sie **'Kompilieren & Schließen'** und hoffen wir, daß MUIbase das Fenster schließt, was bedeutet, daß MUIbase keine Fehler während des Kompilierens gefunden hat.

Machen Sie sich zunächst keine Gedanken darüber, was die Befehle im einzelnen bedeuten. Wie bei allen Programmiersprachen benötigt es etwas Zeit und Übung, um sie zu meistern.

Jetzt haben wir ein lauffähiges Programm, aber wir müssen noch die Programmfunktion mit dem Knopf **'Pedigree'** verbinden. Dazu wechseln wir in den Struktureditor, wählen **'Controltable'** aus dem Tabellenbereich und doppelklicken auf das Feld **'Pedigree'** im Felderbereich. Dann klicken Sie auf die Listenansicht **'Auslösefunktion'**. In dieser Liste werden alle Programmfunktionen aufgelistet und im Moment sollten drei Funktionen zu sehen sein: **pedigree**, **getpedigree** und **pedigree2memo**. Doppelklicken Sie auf **pedigree** und diese Programmfunktion wird vom Knopf **'Pedigree'** ausgelöst. Drücken Sie schließlich **'Ok'** und verlassen Sie den Struktureditor.

Wenn nun alles korrekt durchgeführt wurde, wird ein Druck auf den Knopf 'Pedigree' einen Stammbaum der gerade angezeigten Person erzeugen. Wechseln Sie zu anderen Personen, um die verschiedenen Stammbäume zu sehen.

## 4.12 MUIbase programmieren, um die Kinder einer Person aufzulisten

Als Zusatz benötigt MUIbase weitere Datensätze. Sie sollten daher Ihre Geschwister hinzufügen. Wenn Sie keine haben, schreiben Sie einfach 'Meine Pseudoschwester 1' und 'Mein Pseudobrunder 1', die natürlich die gleichen Eltern wie Sie haben.

Nun gehen Sie in den Programmeditor und geben folgendes zusätzlich ein, um ein weiteres Programm zu erstellen:

```
; Das Programm children zählt die Anzahl der Kinder, die eine Person besitzt.
; Zuerst definieren wir die Variablen, die wir benötigen,
; wie z.B. "children", das auf "\n\n" gesetzt wird.
```

```
(DEFUN children ()
  (LET ( (children "\n\n") (nrofchildren 0) (currentperson Persontable) )
```

```
    ; Über alle Datensätze in Persontable wird folgendes durchgeführt:
    ; Wenn die aktuelle Person als Vater oder Mutter in anderen Datensätzen
    ;   auftritt, dann:
    ;   - füge den Namen zur Variable children hinzu
    ;   - erhöhe die Anzahl der Kinder um 1.
```

```
(FOR ALL Persontable DO
  (IF (OR (= currentperson Father) (= currentperson Mother))
    (
      (SETQ children (+ children Name "\n"))
      (SETQ nrofchildren (+ nrofchildren 1))
    )
  )
)
```

```
; Anschließend schreiben wir das Ergebnis in das mehrzeilige Textfeld
;   von Controltable: Resultmemo
; Wenn die aktuelle Person keine Kinder hat, wird eine Zeichenkette
;   ausgegeben.
; Wenn er/sie Kinder hat, wird eine andere Zeichenkette ausgegeben.
```

```
(SETQ Controltable.Resultmemo
  (+ Persontable.Name (IF (> nrofchildren 0)
    (+ " ist der stolze Vorfahr von " (STR nrofchildren) " Kind(ern).")
    " hat (noch :- ) keine Kinder."
  ))
)
```

```

; Wenn die aktuelle Person Kinder hat, werden sie angehängt.

(IF (<> nrofchildren 0)
  (SETQ Controltable.Resultmemo
    (+ Controltable.Resultmemo "\n\n"
      (IF (= nrofchildren 1)
        "Der Name des Kindes ist:"
        "Die Namen der Kinder sind:")
      )
    children
  )
)

; Dies ist das Ende der Klammer vom Befehl LET.
)

; Dies ist das Ende der Klammer vom DEFUN children.
)

```

Um Variablen zu erzeugen, verwenden wir den Befehl **LET**. Variablen, die mit dem Befehl **LET** erzeugt werden, sind lokal und nur sichtbar innerhalb der Klammern vom Befehl **LET**. Deshalb muß jeder Befehl, der auf diese Variablen zugreifen möchte, innerhalb dieser Klammern stehen.

Wir benötigen nur einen neuen Programmknopf, um das Programm auszuführen, daher wechseln wir wieder in den Struktureditor und fügen einen Knopf der Tabelle 'Controltable' hinzu. Benennen Sie es 'Children' und wählen Sie 'children' als Programmfunktion, die ausgelöst werden soll.

Um Ordnung in das Layout der Tabelle 'Controltable' zu bringen, wird es Zeit, Gruppen vorzustellen. Alle Objekte können in vertikal oder horizontal ausgerichteten Gruppen angeordnet werden.

Klicken Sie im Ansichtsbereich auf 'Pedigree' und klicken sie mit **SHIFT** auf 'Children'. Danach klicken sie links daneben auf den Knopf 'Gruppe'. Jetzt haben Sie zwei Programmknöpfe zusammen in einer vertikal ausgerichteten Gruppe angeordnet. Wir wollen sie jedoch horizontal angeordnet haben und doppelklicken daher auf das 'Vert.Gruppe', das im Anzeigebereich aufgetaucht ist. Dies öffnet ein Fenster, das es Ihnen erlaubt, die Einstellungen dieser Gruppe zu ändern. Setzen Sie die Überschrift auf 'Programme' und aktivieren den Knopf 'Horizontal?'.

Wir können jetzt auch gleich den Namen von 'Resultmemo' in 'Controltable' entfernen. Doppelklicken Sie auf 'Resultmemo' im Anzeigebereich und löschen den Namen. 'Resultmemo' ist nach wie vor zu sehen, aber dessen Name ist nicht mehr sichtbar.

Um es leichter zu machen, wenn wir mehrere Programme oder Felder in 'Controltable' hinzufügen wollen, sollten wir 'Resultmemo' und 'Programs' in eine vertikalen Gruppe fassen. Seien Sie sicher, daß sie nur die Gruppe 'Programs' und 'Resultmemo' ausgewählt haben und drücken Sie 'Gruppe'. Dies setzt 'Programs' und 'Resultmemo' in eine vertikalen Gruppe.

Verlassen Sie den Struktureditor und werfen Sie einen Blick auf das Ergebnis. Drücken Sie nun auf **‘Children’**, um die Anzahl und die Namen der Kinder der aktuellen Person zu sehen.

Dieses Beispiel könnte gut in ein gut ausgebautes Stammbaumprogramm erweitert werden. Wirkliche Grenzen dafür sind Ihre Fantasie und die Größe Ihrer Festplatte.

(Anm.d.Übersetzers: Durch die etwas eigenwillige Sprache des Tutorialschreibers wird der Text einigen etwas merkwürdig vorkommen. Dennoch bietet er eine gute Übersicht über die wesentlichsten Funktionen von MUIbase.)

## 5 Grundlagen

Bevor man beginnt, eigene Datenbanken zu entwickeln und Daten in ihnen einzugeben, sollte man über die Grundlagen Bescheid wissen, auf die MUIbase aufbaut.

### 5.1 Projekte

Ein MUIbase-Projekt enthält alle relevanten Informationen, die zum Verwalten von Daten benötigt werden. Dies schließt die Benutzerschnittstelle, die eingegebenen Daten und die Programme des Projekts ein.

Ein Projekt kann von einer Platte geladen, auf ihr gespeichert und von ihr gelöscht werden. Jede Änderung, die am Projekt durchgeführt wird, wird nur im Speicher durchgeführt. Jederzeit kann zum zuletzt gespeicherten Status des Projekts zurückgekehrt werden, indem es neu geladen wird.

MUIbase kann mehrere Projekte gleichzeitig handhaben. Daher ist es nicht notwendig, MUIbase nochmal aufzurufen, wenn nur ein weiteres Projekt geladen werden soll.

### 5.2 Tabellen

MUIbase verwaltet Daten in Tabellen. Eine Tabelle ist in Zeilen und Spalten angeordnet, wobei Zeilen Datensätze und Spalten Felder heißen.

Folgendes Beispiel zeigt eine Tabelle, wie eine Menge von Adressen in einer Tabelle angeordnet sind:

Name	Street	City
-----	-----	-----
Steffen Gutmann	Wiesentalstr. 30	73312 Geislingen/Eybach
Charles Saltzman	University of Iowa	Iowa City 52242
Nicola Müller	21W. 59th Street	Westmont, Illinois 60559

Es existiert eine besondere Art von Tabelle, die nur genau einen Datensatz halten kann. Diese Art von Tabelle ist manchmal nützlich, wenn eine Datenbank gesteuert werden soll, z.B. können Knöpfe zum Ausführen von diversen Aktionen oder Nur-Lesefelder zur Darstellung von projektabhängiger Information in die Tabelle eingesetzt werden. Angenommen, es gibt beispielsweise eine Finanzdatenbank, in der Ein- und Ausgaben gespeichert werden. Eine Tabelle mit genau einem Datensatz könnte ein Nur-Lesefeld vom Typ Fließkommazahl beinhalten, in dem der aktuelle Gewinn bzw. Verlust angezeigt wird.

Jede Tabelle hat zwei Datensatzzeiger: ein Zeiger, der auf den Datensatz zeigt, der gerade über die Benutzerschnittstelle angezeigt wird (genannt *GUI-Datensatzzeiger*) und ein Zeiger, der

auf den Datensatz zeigt, der beim Ausführen eines MUIbase-Programms verwendet wird (genannt *Programm-Datensatzzeiger*).

Man kann unbegrenzt viele Tabellen in einem MUIbase-Projekt anlegen. (Anmerkung: in der unregistrierten Version gibt es eine Einschränkung auf 5 Tabellen pro Projekt).

Tabellen können hinzugefügt, umbenannt und von einem Projekt gelöscht werden.

### 5.3 Datensätze

Ein Datensatz ist eine Zeile einer Tabelle. Sie trägt sämtliche Information einer Menge, d.h. in einer Tabelle, die Adressen verwaltet, hält sie eine Adresse.

Jeder Datensatz besitzt eine Nummer, die ihre Position in der Tabelle widerspiegelt. Diese Nummer kann sich ändern, wenn Datensätze hinzugefügt oder gelöscht werden.

Für jede Tabelle existiert ein Datensatz, der *Vorgabedatensatz* genannt wird, der die Vorgabewerte zum Anlegen neuer Datensätze beinhaltet. Dieser Vorgabedatensatz hat immer die Datensatznummer 0.

Datensätze können hinzugefügt, verändert und von einer Tabelle gelöscht werden. Es gibt keine Obergrenze für die Anzahl von Datensätzen einer Tabelle. Die Datensätze werden nicht notwendigerweise im Speicher gehalten, werden aber von Platte geladen oder auf ihr gespeichert, wenn es nötig sein sollte. Ok, es gibt doch zwei Obergrenzen für die maximale Anzahl von Datensätzen einer Tabelle. Die eine basiert auf der Tatsache, daß die Datensatznummer ein Wert der Größe `long` ist, der die Anzahl der Datensätze auf 4294967295 begrenzt. Die andere Einschränkung besteht darin, daß für jeden Datensatz ein kleiner Datensatz im Speicher gehalten werden muß. Diese Einschränkungen sollten MUIbase dennoch für Datensatzzahlen von 10000 und mehr verwendbar machen.

### 5.4 Felder

Ein Feld legt eine Spalte einer Tabelle fest. Es legt den Typ und die Erscheinung der betreffenden Spalte fest.

Felder können hinzugefügt, umbenannt und von einer Tabelle gelöscht werden. Es gibt hier keine Obergrenze für die Anzahl der Felder pro Tabelle. (Anmerkung: in der unregistrierten Version von MUIbase gibt es eine Einschränkung auf 10 Felder pro Tabelle).

Für jedes Feld muß ein Typ festgelegt werden, der den Inhalt des Feldes einschränkt. Eine Liste aller Feldtypen ist im nächsten Abschnitt zu finden.

### 5.5 Feldtypen

Felder können vom Typ Zeichenkette, Ganzzahl, Fließkommazahl, Bool, Auswahl, Datum, Zeit, mehrzeiliger Text, Beziehung, virtuell oder Knopf sein. Diese Typen werden unten detaillierter beschrieben.

Einige der Felder unterstützen einen besonderen Wert NIL. Dieser Wert stellt einen undefinierten Wert dar, z.B. bei einem Datum für ein unbekanntes Datum. Der Wert NIL entspricht dem Wert NULL von anderen Datenbanksystemen.

Anzumerken ist, daß der Typ eines Feldes nachträglich nicht mehr geändert werden kann, wenn er einmal gesetzt worden ist.

### 5.5.1 Zeichenketten

Zeichenketten können eine einzelne Zeile Text speichern. Sie sind der am meisten verwendete Feldtyp in einem Datenbankenprojekt. Zum Beispiel speichert eine Adreßdatenbank den Namen, die Straße und den Ort einer Person jeweils in einem Zeichenkettenfeld.

Für ein Zeichenkettenfeld muß die maximale Länge in Zeichen festgelegt werden, die in einer Zeichenkette zulässig sind. Diese Zahl bezieht sich nicht auf den benötigten Platz im Speicher oder auf Platte, der durch die Zeichenkette beansprucht wird, da nur der aktuelle Zeichenketteninhalt gespeichert wird (andere Datenbanken nennen dieses Feature komprimierte Zeichenketten). Wenn nötig kann die Nummer nach der Erstellung des Zeichenkettenfeldes nachträglich geändert werden.

Zeichenkettenfelder können auch verwendet werden, um Zeichensatz- und Dateinamen zu speichern. Bei Dateinamen können externe Anzeigeprogramme gestartet werden, um den Dateinhalt anzuzeigen. Des weiteren erlaubt eine eingebaute Bilderklasse die Darstellung eines Bildes aus einer Datei.

Zeichenkettenfelder unterstützen nicht den Wert NIL.

### 5.5.2 Ganzzahlfelder

Ganzzahlfelder speichern ganze Zahlen im Bereich von -2147483648 to 2147483647. Sie werden meistens verwendet, um Mengenangaben wie z.B. die Anzahl der Kinder pro Person oder die Anzahl der Lieder pro CD zu speichern.

Ganzzahlfelder unterstützen den Wert NIL, der für eine undefinierte Ganzzahl steht.

### 5.5.3 Fließkommazahlfelder

Fließkommazahlen speichern Werte im Bereich von -3.59e308 bis +3.59e308.

Sie werden für das Speichern von Zahlen jeder Art verwendet, wie z.B. die Beträge in einem Projekt für Einkommen/Ausgaben.

Für jedes Fließkommazahlfeld läßt sich die Anzahl der Nachkommastellen festlegen, aber intern wird dennoch mit der vollständigen Präzision gearbeitet.

Fließkommazahlfelder unterstützen den Wert NIL, der für eine undefinierte Fließkommazahl steht.



### 5.5.4 Boolesche Felder

Boolesche Felder speichern ein Bit Information. Sie können für das Speichern von Werten wie ja/nein oder wahr/falsch verwendet werden, z.B. in einem Buchhaltungsprojekt könnte ein boolesches Feld die Information ‘hat bezahlt?’ vermerken.

Boolesche Felder verwenden TRUE (=wahr) und NIL als boolesche Werte. NIL steht in diesem Fall für falsch.

### 5.5.5 Auswahlfelder

Auswahlfelder speichern ein Element aus einer Liste von Elementen. Zum Beispiel kann das Feld in einer Adreßdatenbank zum Speichern von Landesnamen wie ‘USA’, ‘Kanada’, ‘Deutschland’ oder ‘andere’ verwendet werden.

Ein Auswahlfeld speichert im Datensatz nicht die gesamte Elementzeichenkette, sondern nur die Elementnummer (Index). Die Anzahl der Elemente und die Elemente selbst können nach dem Erstellen nachträglich geändert werden. Jedoch werden die vorhandenen Datensätze nicht auf die Änderungen an einem Auswahlfeld angepaßt, um die neue Situation zu widerspiegeln.

Auswahlfelder unterstützen den Wert NIL nicht.

### 5.5.6 Datumsfelder

Datumsfelder speichern ... ehm ... Daten<sup>1</sup>. Zum Beispiel kann ein Datumsfeld zum Speichern von Geburtstagen verwendet werden.

Das Format zum Eingeben von Datumswerten ist eines aus ‘TT.MM.JJJJ’, ‘MM/TT/JJJJ’ oder ‘JJJJ-MM-TT’, wobei ‘TT’ für den Tag, ‘MM’ für den Monat und ‘JJJJ’ für das Jahr steht.

Datumsfelder unterstützen den Wert NIL, der für ein undefiniertes Datum steht.

### 5.5.7 Zeitfelder

Zeitfelder speichern Zeiten oder Zeiträume. Zum Beispiel kann ein Zeitfeld verwendet werden, um die Spielzeiten von Musikstücken einer CD zu speichern.

Das Format zum Eingeben und Darstellen von Zeitangaben ist auf ‘HH:MM:SS’ festgelegt, wobei ‘HH’ für eine zweistellige Zahl im Bereich 0 bis 23 für Stunden, ‘MM’ eine zweistellige Zahl im Bereich von 0 bis 59 für Minuten und ‘SS’ eine zweistellige Zahl im Bereich von 0 bis 59 für Sekunden steht.

Zeitfelder unterstützen den Wert NIL, der für eine undefinierte Zeit steht.

---

<sup>1</sup> Anm.d.Übersetzers: nicht verwechseln mit den *Daten*, die eingegeben werden können, da die Mehrzahl von *Datum* leider *Daten* ist

### 5.5.8 mehrzeilige Textfelder

Mehrzeilige Textfelder speichern mehrzeilige Texte jeder Größe. Die Textgröße wird dynamisch verwaltet, was bedeutet, daß nur soviel Speicher benötigt wird, wie der Text groß ist. In einem Projekt, das Filme verwaltet, kann ein mehrzeiliges Textfeld verwendet werden, um Kurzbeschreibungen zum Film aufzunehmen.

Mehrzeilige Textfelder unterstützen nicht den Wert NIL.

### 5.5.9 Beziehungsfelder

Beziehungsfelder sind ein besonderer Feldtyp, der gewöhnlich nicht in anderen Datenbanksystemen zu finden ist. Beziehungsfelder speichern einen Zeiger auf einen anderen Datensatz. Der Datensatz, auf den verwiesen wird, kann in der selben oder in jeder anderen Tabelle liegen, zu der die Beziehung gehört.

Zum Beispiel können in einem Stammbaumprojekt zwei Beziehungsfelder verwendet werden, um Zeiger auf den Vater und die Mutter zu speichern. In einer CD-Verwaltung mit Musiktiteln kann ein Beziehungsfeld in einer Tabelle, die die Musiktitel beinhaltet, verwendet werden, um auf die Datensätze der entsprechenden CDs zu verweisen.

Zur Darstellung eines Beziehungsfeldes können ein oder mehrere Felder des Bezugsdatensatzes angegeben werden. Eingaben in ein Beziehungsfeld können durch Auswählen eines Datensatzes aus einer Liste von Datensätzen erfolgen.

Beziehungsfelder unterstützen den Wert NIL. Hierbei steht er für einen Zeiger auf den Vorgabedatensatz der Bezugstabelle.

### 5.5.10 virtuelles Feld

Virtuelle Felder sind ein besonderer Feldtyp, die keine Information in der Datenbank speichern, sondern sie nur beiläufig ermitteln, wenn es notwendig ist.

Zum Beispiel kann in einem Buchhaltungsprojekt Werte einschließlich Steuern ermittelt werden, wenn in der Datenbank nur Werte ohne Steuern gespeichert werden. Jedesmal, wenn der Wert des virtuellen Feldes benötigt wird, wie z.B. beim Darstellen, wird er aus dem entsprechenden Wert ohne Steuern berechnet.

Zum Darstellen von virtuellen Feldern existieren drei Arten: Bool, Zeichenkette und Liste. Diese drei Arten erlauben die Darstellung des virtuellen Feldes als Wahr/Falsch-Wert, als eine einzeilige Zeichenkette einschließlich Zahlen, Daten und Zeiten, und als eine Liste aus mehreren einzelnen Zeilen, z.B. zum Auflisten aller Titel einer CD.

Virtuelle Felder unterstützen den Wert NIL, der für Falsch (bei Bool), undefiniert (bei Zeichenkette) oder leer (bei Liste) steht.

### 5.5.11 Knöpfe

Knöpfe sind genaugenommen keine echten Feldtypen, da sie keine Daten speichern oder darstellen. Sie werden nur zum Auslösen von MUIbase-Programmen verwendet.

Der Grund, daß sie als Feldtyp gehandhabt werden, ist der Zugriff über ihren Feldnamen. Dies erlaubt die Benutzung des Knopfnamens in einem MUIbase-Programm, z.B. zum Aktivieren/Deaktivieren des Knopfes. Ein anderer Grund ist, daß Knöpfe ähnliche Eigenschaften wie Felder besitzen, wie z.B. Auslösefunktionen.

## 5.6 Tabelle der Feldtypen

Die folgende Tabelle zeigt alle verfügbaren Feldtypen auf:

Typ	Beschreibung	NIL zulässig?
Zeichenkette	Für Zeichenketten der Länge 1..999. Eine Zeichenkette kann auch zum Speichern von Dateinamen, Zeichensatznamen und einer aus mehreren Zeichenketten verwendet werden. Bei Dateinamen kann ein Feld hinzugefügt werden, in dem die Datei als Bild angezeigt wird.	Ja
Ganzzahl	Zum Speichern von Ganzzahlen	Ja
Fließkommazahl	Zum Speichern von reellen Zahlen	Ja
Bool	TRUE oder NIL	Ja (NIL = falsch)
Auswahl	Eine Nummer aus n Nummern, die durch Auswahltexte repräsentiert werden.	Nein
Datum	Zum Speichern eines Datum (1.1.0000 - 31.12.9999)	Ja
Zeit	Zum Speichern einer Zeit (00:00:00 - 23:59:59)	Ja
mehrzeiliger Text	mehrzeilige Texte von unbegrenzter Länge	Nein
Beziehung	Zum Speichern einer Beziehung zu einem Datensatz einer anderen Tabelle	Ja (NIL bedeutet Vorgabedatensatz)
virtuell	Zum Darstellen von Ergebnissen aus einem MUIbase-Programm	Ja
Knopf	Zum Auslösen einer Programmfunktion	Nein (k.A.)

## 5.7 Speicherverbrauch

Jeder Feldtyp benötigt eine bestimmte Menge Speicher, um einen Wert in einem Datensatz zu speichern. Alle Typen außer virtuell und Knopf haben gemeinsam, daß sie einen Kopf aus 2 Bytes benötigen, der interne Information speichert. Zusätzlich wird typenabhängiger Speicher benötigt, der zum Speichern des momentanen Wertes dient. Die folgende Tabelle zeigt auf, wieviel Speicher einschließlich des möglichen 2 Bytes-Kopfes ein Wert zu gegebenem Typ an Speicher im RAM und auf Platte benötigt.

Typ	Speicherplatz	Plattenplatz
Zeichenkette	$2 + 4 + \text{Länge} + 1$	$2 + \text{Länge} + 1$
Ganzzahl	$2 + 4$	$2 + 4$
Fließkommazahl	$2 + 8$	$2 + 8$
Bool	$2 + 0$	$2 + 0$
Auswahl	$2 + 2$	$2 + 2$
Datum	$2 + 4$	$2 + 4$
Zeit	$2 + 4$	$2 + 4$
mehrzeiliger Text	$2 + 4 + \text{Länge} + 1$	$2 + \text{Länge} + 1$
Beziehung	$2 + 4$	$2 + 4$
virtuell	0	0
Knopf	0	0

*Länge* steht für die Länge der zu speichernden Zeichenkette bzw. des mehrzeiligen Textes.

## 5.8 Beziehungen

Bis jetzt ist bekannt, wie Information in Tabellen mit Datensätzen und Feldern angeordnet werden, aber es sollten auch Beziehungen zwischen Tabellen hergestellt werden können.

Zum Beispiel würde eine Datenbank für CDs zwei Tabellen enthalten, eine für die CDs und eine für die Musiktitel der CDs. Natürlich können alle Musiktitel innerhalb der CD-Tabelle gespeichert werden, aber dadurch ist die Anzahl der Musiktitel pro CD auf eine feste Anzahl beschränkt.

Nun wird eine Verbindung zwischen den beiden Tabellen benötigt, um für jeden Musiktitel die entsprechende CD zuzuweisen. Dies wird *Beziehung* zwischen beiden Tabellen genannt. Normalerweise wird dafür ein Beziehungsfeld verwendet, um eine solche Beziehung herzustellen.

Durch das Einrichten eines Beziehungsfeldes in einer Tabelle wird automatisch eine Beziehung zwischen der Tabelle, die das Feld beinhaltet, und der Tabelle hergestellt, zu der sie verweist.

### 5.8.1 Eins-zu-Eins-Beziehungen

Eins-zu-Eins-Beziehungen sind ganz einfache Beziehungen, bei denen jeder Datensatz genau einen oder keinen Partner in der anderen oder selben Tabelle besitzt.

In einer Datenbank, die beispielsweise Lieblingsschauspieler verwaltet, könnte ein Beziehungsfeld ‘**verheiratet mit**’ eingerichtet werden, das anzeigt, mit welcher Person der/die Schauspieler/in verheiratet ist. Ein/e Schauspieler/in, die momentan nicht verheiratet ist, würde dann den Wert NIL im Beziehungsfeld haben.

Dies verhindert natürlich nicht, daß der Benutzer die Beziehungen über ‘**verheiratet mit**’ von verschiedenen Schauspieler/innen auf die gleiche Person setzen. Es läßt sich jedoch über die Programmierung von MUIbase erreichen, daß solche Fälle sofort erkannt und behandelt werden können.

### 5.8.2 Eins-zu-Mehrfach-Beziehungen

Eins-zu-Mehrfach-Beziehungen sind nützlich, um mehrere Datensätze zu einem Datensatz in einer anderen oder der selben Tabelle zu verbinden.

In einem Projekt, das z.B. Bankkonten verwaltet, könnte eine Tabelle alle Bankkonten und eine Tabelle alle Überweisungen enthalten. Nun ist es sicher sinnvoll zu wissen, welche Überweisung auf welches Konto durchgeführt werden soll. Daher wird ein Beziehungsfeld in der Überweisungstabelle eingerichtet, das auf die Bankkontentabelle verweist.

Eins-zu-Mehrfach-Beziehungen werden am meisten verwendet. Sie können zum Verwalten von hierarchisch angeordneten Strukturen verwendet werden, z.B. CDs mit Musiktiteln, Bankkonten mit Überweisungen, Stammbäume, etc.

Eins-zu-Mehrfach-Beziehungen sind auch Grundlage zum Realisieren von Mehrfach-zu-Mehrfach-Beziehungen, die im nächsten Abschnitt beschrieben werden.

### 5.8.3 Mehrfach-zu-Mehrfach-Beziehungen

Mehrfach-zu-Mehrfach-Beziehungen werden verwendet, wenn mehrere Datensätze auf eine andere Menge von mehreren Datensätzen verweisen sollen.

Ein Projekt, das Spielfilme und Schauspieler/innen verwaltet, enthält zwei Tabellen, eine für die Filme und eine für die Schauspieler/innen. Nun sollen für jeden Film die Schauspieler/innen ermittelt werden, die jeweils in den Filmen mitmachen. Jetzt könnte ein Beziehungsfeld in der Tabelle der Schauspieler/innen eingerichtet werden, um auf die Tabelle Filme zu verweisen. Wenn dies gemacht wird, dann kann ein/e Schauspieler/in nur in genau einem Film mitgespielt haben, da nur ein Beziehungsfeld in der Tabelle der Schauspieler/innen existiert. Deshalb werden eine unbeschränkte Anzahl von Beziehungen von der Tabelle der Schauspieler/innen zur Tabelle Filme benötigt.

Um dies zu bewerkstelligen, wird eine neue Tabelle hinzugefügt, die nur zwei Beziehungsfelder beinhaltet: eines, das auf die Tabelle der Schauspieler/innen und eines, das auf die Tabelle Filme verweist. Nun können Beziehungen durch neue Datensätze in dieser Tabelle hinzugefügt werden. Für jede Verbindung von Film-Schauspieler/in muß dann ein neuer Datensatz erzeugt werden, in dem Film und Schauspieler/in in den entsprechenden Beziehungsfeldern gesetzt werden.

Wenn nun ermittelt werden soll, in welchen Filmen ein/e Schauspieler/in mitgewirkt hat, dann brauchen nur alle Datensätze in der neuen Tabelle durchsucht werden, in der auf den/die gesuchte/n

Schauspieler/in verwiesen wird und betrachtet die Film-Datensätze, von denen die gefundenen Datensätze hinverweisen. Eine solche Suche kann automatisch von MUIbase durchgeführt werden und das Ergebnis in einer Listenansicht angezeigt werden.

Die folgende Tabelle Zeigt ein Beispiel, wie eine Menge Schauspieler/innen und eine Menge Filme miteinander verbunden werden.

	Titel	Land
	-----	-----
m1:	Batman	USA
m2:	Batmans Rückkehr	USA
m3:	Sprachlos	USA
m4:	Tequila Sunrise	USA
m5:	Mad Max	Australien
m6:	Braveheart	USA

	Name
	-----
a1:	Michael Keaton
a2:	Jack Nicholson
a3:	Kim Basinger
a4:	Danny DeVito
a5:	Michelle Pfeiffer
a6:	Geena Davis
a7:	Christopher Reeve
a8:	Mel Gibson
a9:	Kurt Russell
a10:	Sophie Marceau
a11:	Patrick McGoohan
a12:	Catherine McCormack
a13:	Christopher Walken

FilmBez	SchauspBez
-----	-----
m1	a1
m1	a2
m1	a3
m2	a1
m2	a4
m2	a5
m2	a13
m3	a1
m3	a6
m3	a7

m4	a8
m4	a5
m4	a9
m5	a8
m6	a8
m6	a10
m6	a11

Aus diesen Tabellen kann z.B. herausgelesen werden, daß Mel Gibson in den Filmen Tequila Sunrise, Mad Max und Braveheart mitgewirkt hat, oder daß im Film Batman die Schauspieler Michael Keaton, Jack Nicholson und Kim Basinger mitgespielt haben.

## 5.9 Benutzerschnittstelle

MUIbase verwendet eine grafische Benutzerschnittstelle (GUI), die hierarchisch organisiert ist, um Datensatzinhalte darzustellen und um die Eingabe von Daten durch Benutzer zu ermöglichen. Jedes Projekt besitzt sein eigenes Hauptfenster, in dem weitere GUI-Elemente (einschließlich Unterfenster) plazierte werden können. Die GUI-Elemente werden auch *Anzeigeelemente* genannt.

Eine Tabelle wird in einem eigenen GUI-Element *Maske* dargestellt. Eine Maske kann nur einen Datensatz zu einem Zeitpunkt darstellen. Dessen Layout und die Felder, die in der Maske eingeschlossen sind, können vom Benutzer verändert werden.

Die folgenden Abschnitte beschreiben MUIbase's GUI-Elemente, um das Layout eines Projekts gestalten zu können.

### 5.9.1 Fenster

Fenster können verwendet werden, um Informationen eines Projekts auf mehrere unabhängige Bereiche aufzuteilen.

Jedes Projekt besitzt automatisch sein eigenes Hauptfenster. Falls nötig, wie z.B. wenn der Platz des Hauptfensters nicht ausreicht, können zusätzliche Unterfenster erzeugt werden. Unterfenster können wiederum weitere Unterfenster haben.

Für jedes Unterfenster wird im darüberliegenden Fenster ein Fensterknopf eingerichtet, um das Fenster öffnen bzw. schließen zu können. Der Fensterknopf sieht wie ein normaler Textknopf auf, hat aber ein kleines Fenstericon, das den Status geöffnet/geschlossen des entsprechenden Unterfensters anzeigt.

Hauptfenster haben kein übergeordnetes Fenster und haben daher auch keinen Fensterknopf. Schließen eines Hauptfensters bedeutet Schließen des gesamten Projekts.

Ein Fenster kann jedes andere GUI-Element (ausgenommen Panels) als Kinder haben. Wenn kein Kind zu einem Fenster hinzugefügt wurde, dann wird ein Bild 'Leere Anzeige' (siehe Abschnitt 7.18 [Empty display image], Seite 41) angezeigt.

### 5.9.2 Masken

Eine Maske wird verwendet, um den Inhalt einer Tabelle darzustellen. Nur ein Datensatz der Tabelle kann zu einem Zeitpunkt angesehen werden.

Die Maske kann ein Panel (siehe nächster Abschnitt) enthalten, mit dem man die Tabelle steuern kann. Andere GUI-Elemente wie Felder oder Textobjekte können in der Maske plziert werden, um die Datensatzinhalte anzuzeigen.

Masken können nicht in anderen Masken plziert sein, da dies zu einer Hierarchie von Masken und somit auch zu einer Hierarchie von Tabellen führen würde, was in MUIbase nicht unterstützt wird. Wenn eine Hierarchie von Tabellen eingerichtet werden soll, ist eine Eins-zu-Mehrfach-Beziehung zwischen zwei Tabellen zu verwenden.

### 5.9.3 Panels

Ein Panel ist eine kleine rechteckige Fläche am oberen Rand einer Maske. Ein Panel kann einen Titel, z.B. den Namen der dazugehörigen Tabelle, ein Nummernpaar, das die Datensatznummer und die Gesamtanzahl der Datensätze anzeigt, und einige Knöpfe zum Steuern der Tabelle, z.B. zum Darstellen vom nächsten und vorhergehenden Datensatz, enthalten.

Nur ein Panel kann in einer Maske definiert werden. Wird ein Panel für eine Maske eingerichtet, dann wird ein zusätzlicher Rahmen um die Maske gezeichnet, anderenfalls nicht.

### 5.9.4 Feldobjekte

Feldobjekte werden benützt, um den Inhalt eines Elements aus einem Datensatz darzustellen.

Abhängig vom Typ des Feldes ist das GUI-Element entweder ein Zeichenkettengadget (Typen Zeichenkette, Ganzzahl, Fließkommazahl, Datum und Zeit), ein Checkmark-Gadget (Typ Bool), ein Auswahlgadget oder eine Menge von Radioknöpfen (Typ Auswahl), ein Editorgadget (Typ mehrzeiliger Text), eine Popup-Listenansicht (Typ Beziehung), ein Text, Checkmark oder Listenansicht (Typ virtuell) oder ein Text- bzw. Bildknopf (Typ Knopf). In einigen Fällen kann ein GUI-Element ein einfaches Textgadget sein, wenn das Feldobjekt auf nur lesen gesetzt ist.

### 5.9.5 Textobjekte

Textobjekte werden verwendet, um die verschiedenen Feldelemente einer Datensatzmaske zu beschreiben oder einfach nur zum Anzeigen von festen Text wie Copyrightvermerk irgendwo im Fenster.

### 5.9.6 Bilder

Bilder können überall im Fenster angezeigt werden. Ein Bild kann eine interne MUI-Grafik, ein einfaches Farbfeld oder ein Bild aus einer externen Datei sein. Die Größe des Bildes kann auf größenveränderbar oder fest gesetzt werden.



Das Bild ist fest eingebaut. Wenn Bilder in einer Tabelle gespeichert werden sollen, sollte ein Zeichenkettenfeld verwendet werden (siehe Abschnitt 5.5.1 [String type], Seite 22).

Anzumerken ist, daß Bilder in der unregistrierten Version von MUIbase nicht verfügbar sind.

### 5.9.7 Zwischenraumobjekte

Zwischenraumobjekte dienen dem Einfügen von Leerraum im Layout eines Fensters oder einer Tabellenmaske. Ein Zwischenraumobjekt kann einen vertikalen (oder horizontalen) Strich zum Abtrennen von anderen GUI-Elemente besitzen.

### 5.9.8 Gruppen

GUI-Elemente können in horizontalen oder vertikalen Gruppen angeordnet werden. Eine Gruppe plaziert ihre Kinder von links nach rechts (horizontale Gruppe) oder von oben nach unten (vertikale Gruppe).

Eine Gruppe kann seine Kindobjekte mit einem rechteckigen Rahmen umschließen, ein optionaler Titel oberhalb der Gruppe angezeigt werden und ein Schalter steuert, ob Zwischenräume zwischen den Kindobjekten eingefügt werden sollen oder nicht.

### 5.9.9 Gewichtungsobjekte

Gewichtungsobjekte können überall zwischen Kindobjekten in einem Fenster, einer Maske oder einem Gruppenobjekt eingesetzt werden. Dieses Objekt erlaubt dem Benutzer, die Gewichtungen der anderen Kindobjekte und somit den Platz der einzelnen Kinder dynamisch zu steuern.

### 5.9.10 Karteikarten-Gruppen

Eine Karteikarten-Gruppe kann benutzt werden, um einige GUI-Elemente auf verschiedenen Seiten anzuordnen, von denen jeweils eine zu einer Zeit aktiviert werden kann. Dies ist nützlich, wenn die Benutzerschnittstelle zu groß wird und man sie nicht über mehrere Fenster verteilen will.

## 6 Projekte verwalten

In diesem Kapitel wird aufgezeigt, wie MUIbase Projekte organisiert, sie öffnet, speichert, löscht und schließt, wie die Integrität der Daten geprüft wird und wie das Auslagern funktioniert.

### 6.1 Dateiformat

Ein MUIbase-Projekt besteht aus einigen Dateien, die in einem eigenen Verzeichnis gespeichert werden. Auf der Workbench wird man die Verzeichnisstruktur nicht bemerken, da das Doppelklicken auf ein Projekticon nicht das Verzeichnis öffnen wird, sondern veranlaßt MUIbase, das Projekt zu laden. Von CLI aus ist die Verzeichnisstruktur eines MUIbase-Projekts jedoch sichtbar.

Alle Dateien eines Projekts werden in einem Verzeichnis abgelegt, welches beim Speichern eines Projekts angelegt wird. Es dürfen weder Dateien aus diesem Verzeichnis entfernt noch in dieses Dateien und weitere Verzeichnisse plazierte werden! Diese werden verloren gehen, wenn das Projekt neu organisiert wird.

Das Verzeichnis enthält eine Datei mit Namen **‘Structure.mb’**, in der die Beschreibungen aller Tabellen, Felder, Filter, usw. gespeichert sind. Die Datensatzköpfe sind auch dort vermerkt. Für jede Tabelle wird eine Datei mit dem Namen der Tabelle angelegt. In ihnen werden alle Datensätze gespeichert. Schließlich gibt es eine Datei mit Namen **‘.lock’**. Sie darf nicht entfernt werden. Falls es doch versehentlich passiert sein sollte, genügt es, wenn sie neu erzeugt wird, der Inhalt ist dabei unwichtig. Diese Datei wird zum Sperren verwendet, was bedeutet, daß MUIbase diese Datei exklusiv sperrt und dann erst die anderen Dateien öffnet. Wenn das Sperren mißlingt, dann weiß MUIbase, daß bereits eine MUIbase-Anwendung an diesem Projekt arbeitet. Immer nur eine MUIbase-Anwendung ist berechtigt, zu einem Zeitpunkt an einem Projekt zu arbeiten, da Datensatzdateien im Schreib-/Lesemodus geöffnet werden und gemischt geschriebene Daten von mehreren Anwendungen sicher nicht gewünscht ist. ;-)

### 6.2 Information

MUIbase hält einige Informationen über jedes Projekt bereit. Um Informationen über das aktuelle Projekt zu erhalten, wird der Menüpunkt **‘Projekt - Information...’** ausgewählt. Die erhaltene Information enthält den Namen des Projekts, die Anzahl der Tabellen, die Gesamtanzahl aller Datensätze in allen Tabellen und ein Wert, der anzeigt, wieviele Bytes beim Neuorganisieren eingespart werden könnten. Diese Einsparung ist jedoch nur ein Schätzwert und sollte nicht als genaue Zahl angesehen werden. Besonders dann, wenn viele Änderungen an der Struktur des Projekts durchgeführt wurden (Hinzufügen oder Entfernen von Feldern), ist der Wert weit vom tatsächlichen entfernt.

### 6.3 Projekt löschen

Um ein neues Projekt zu beginnen, wird der Menüpunkt **‘Projekt - Neu - Projekt’** angewendet. Dies löscht das aktuelle Projekt und MUIbase öffnet ein leeres Fenster, das das MUIbase-Logo anzeigt. In diesem Modus befindet sich MUIbase automatisch nach dem Starten, wenn kein Projekt vorhanden ist.

Über den Menüpunkt **‘Projekt - Neu - Datensätze’** läßt sich ein Projekt neu beginnen, das auf der Struktur des aktuellen Projekts basiert. Dies bedeutet, daß alles außer die Datensatzdaten des aktuellen Projekts für das neue Projekt verwendet wird.

Wenn das Projekt noch nicht gespeichert wurde, bevor einer der beiden Menüpunkte ausgeführt wird, fragt eine Sicherheitsmeldung nach der Bestätigung der Operation.

## 6.4 Projekt öffnen

MUIbase kann mehrere Projekte gleichzeitig behandeln. Dies ist nur begrenzt durch den vorhandenen Speicher. Wenn ein anderes Projekt bearbeitet werden soll, wird **‘Projekt - Neu öffnen’** aufgerufen. Dies öffnet ein neues Projekt mit einem Hauptfenster. Nun läßt sich ein Projekt für dieses Fenster laden.

Um ein Projekt zu laden, wird **‘Projekt - Öffnen - Projekt...’** aufgerufen. Dies öffnet ein Dateiauswahlfenster, in dem ein Projekt ausgewählt werden kann. Es ist auch möglich, nur die Struktur eines Projekts zu laden, wenn das gesamte Projekt ohne die Daten geladen werden soll. Um dies zu bewerkstelligen, wird **‘Projekt - Öffnen - Struktur...’** aufgerufen.

Wird beim Ändern eines Projekts eines der oben genannten Menüpunkte ausgewählt und das Projekt wurde noch nicht gespeichert, dann erscheint eine Sicherheitsmeldung, die nach einer Bestätigung fragt.

## 6.5 Projekt speichern

Alle Änderungen an einem Projekt werden nur im Speicher durchgeführt. Um sie permanent zu machen, muß daher das Projekt auf Platte gespeichert werden. Dies wird durch **‘Projekt - Speichern’** erledigt. Wenn das Projekt noch keinen Namen trägt, erscheint zuerst ein Dateiauswahlfenster und fragt nach einem Dateinamen.

Der Grund, warum MUIbase das Projekt nicht automatisch speichert, wenn sich etwas ändert, ist der, daß der Benutzer entscheidet, wann das Projekt gespeichert wird und der Benutzer immer zur zuletzt gespeicherten Version des Projekts zurückkehren kann. Dieses Verfahren entspricht den Befehlen **‘COMMIT’** und **‘ROLLBACK’** in SQL-basierten Systemen.

Wenn ein Projekt gespeichert wird, dann werden alle geänderten Datensätze auf Platte gespeichert und die Datei **‘Structure.mb’** erzeugt. Vor dem Erzeugen der neuen Datei benennt MUIbase eine möglicherweise schon existierende Datei **‘Structure.mb’** in **‘Structure.old’**, um eine Sicherheitskopie zu haben, falls das Speichern mißlingt.

Dieser Mechanismus garantiert schnelles Laden und Speichern, ist aber nicht frei vom Umschichten. Wenn viele Datensätze geändert wurden, dann werden die physikalische Reihenfolge der Datensätze und die daraus resultierende Fragmentierung zum Nachteil. Deshalb existiert ein Menüpunkt **‘Projekt - Umschichten & Speichern’**, der eine Umschicht- und Speicheroperation durchführt. Diese Operation kann etwas Zeit in Anspruch nehmen, die von der Anzahl und Größe der Datensätze abhängt. Die Umschicht- und Speicheroperation erzeugt ein neues Verzeichnis und schreibt alle projektabhängigen Dateien erneut. Das alte Verzeichnis wird nach erfolgreicher Operation gelöscht.

Wenn Änderungen an der Struktur des Projekts durchgeführt wurden, wie z.B. Einfügen eines neuen Feldes in einer Tabelle, ist ein Umschichten auch sinnvoll. Diese Änderungen werden nicht automatisch an allen Datensätzen durchgeführt, da es zuviel Zeit in Anspruch nehmen würde, jeden Datensatz zu laden, ihn zu verändern und wieder auf Platte zurückzuschreiben. Daher werden diese Änderungen auf eine interne ‘todo’-Liste gesetzt, die nach dem Laden eines Datensatzes abgearbeitet wird. Das Anwenden dieser Liste auf einen Datensatz verbraucht nur wenig Zeit, wobei jedoch eine längere Liste mehr Zeit benötigt. Umschichten eines Projekts führt dazu, daß die ‘todo’-Liste an allen Datensätzen durchgeführt wird. Wenn also viele Änderungen am Projekt durchgeführt wurden, wird ein Umschichten des Projekts die Zeit zum Laden einzelner Datensätze verkürzen.

Es ist auch möglich, ein Projekt unter einem neuen Dateinamen umzuschichten und zu speichern, ohne daß das alte Projekt geändert wird. Dies wird über den Menüpunkt **‘Projekt - Umschichten & Speichern als ...’** angestoßen. MUIbase wird nach einem neuen Namen für das Projekt fragen.

## 6.6 Projekt entfernen

MUIbase bietet einen Menüpunkt zum Löschen eines Projekts. Nach **‘Projekt - Löschen’** wird ein Dateiname verlangt und das Löschen des angegebenen Projekts mit einer separaten Sicherheitsmeldung bestätigt, bevor das Projekt von Platte gelöscht wird.

MUIbase macht nichts anderes, als das Verzeichnis mit allen darin befindlichen Dateien zu löschen. Dies kann ohne Probleme auch über die Workbench bzw. eine Amiga-Shell geschehen.

## 6.7 Projekt schließen

Falls das Projekt fertig ist, kann es über den Menüpunkt **‘Projekt - Schließen’** geschlossen werden. Dies löscht das Projekt aus dem Speicher und schließt alle dazugehörigen Fenster. Falls das Projekt Änderungen hat, die noch nicht gespeichert wurden, bietet eine Sicherheitsmeldung an, es zu speichern, fortzusetzen oder die Operation abubrechen.

Zum Schließen eines Projekts kann auch der Menüpunkt **‘Projekt - Speichern & Schließen’** verwendet werden, bei dem das Projekt zuerst gespeichert wird, wenn Änderungen vorlagen, und dann geschlossen wird.

## 6.8 Integrität der Daten prüfen

MUIbase kann überprüfen, ob alle Daten im Projekt noch gültig sind und noch nicht durch Systemabstürze oder durch Programme beschädigt wurden, die die Projektdateien entfernen. Der Menüpunkt **‘Projekt - Prüfe Integrität der Daten...’** startet diesen Prozeß.

Normalerweise benötigt man diese Funktionalität nie und MUIbase sollte immer melden, daß die Integrität der Daten perfekt ist. Sollte es aber doch passieren, daß das Projekt **‘interne Fehler’** hat, was bedeutet, daß einige Datensätze nicht mehr geladen werden können, dann kann das Projekt über diesen Menüpunkt repariert werden.

MUIbase schreibt dann eine Logdatei mit allen betroffenen Datensätzen und man kann anschließend das Projekt umschichten und speichern. In der Logdatei werden die Datensätze, die möglicherweise beschädigt und nicht mehr erreichbar sind (so daß sie gelöscht wurden), nach ihrer Datensatznummer im alten (beschädigten) Projekt und nach ihrer Datensatznummer (in Klammern) im umgeschichteten Projekt aufgelistet.

Hinweis: Bis MUIbase v1.4 gab es einen Bug, der fehlerhaft geschriebene Projektdateien erzeugte. Mit diesem Menüpunkt sollte es möglich sein, diese Projekte zu reparieren. Seit MUIbase v1.4 ist dieser Fehler repariert und sollte nie mehr beschädigte Projekte erzeugen und das Überprüfen der Datenintegrität sollte immer melden, daß die Integrität der Daten perfekt ist.

## 6.9 Datensätze auslagern

MUIbase muß nicht alle Datensätze eines Projekts im Speicher halten. Dadurch wird das Laden und Speichern von Datensätzen beschleunigt. Beim Laden eines Projekts wird für jeden Datensatz ein Datensatzkopf angelegt. Die Daten selbst werden nur dann geladen, wenn sie benötigt werden, z.B. wenn sie auf dem Bildschirm angezeigt werden sollen. Die Gesamtanzahl der Datensätze ist nur durch den verfügbaren Speicher begrenzt, da jeder Kopf einige Bytes Speicher benötigt.

Es kann festgelegt werden, wieviel Speicher für die Datensätze eines Projekts verwendet werden darf. Dazu wird einer der vorgegebenen Werte im Menü **‘Einstellungen – Datensatzspeicher’** eingestellt. MUIbase wird keinen Speicher der angegebenen Größe vorab belegen, sondern es prüft von Zeit zu Zeit, ob die Größe des momentan belegten Speicher größer ist als die angegebene. Der angegebene Wert ist keine genaue Grenze, sondern nur ein Vorschlag. Wenn MUIbase mehr Speicher benötigt, dann wird es die Obergrenze ignorieren und ihn belegen.

Wenn MUIbase der Speicher ausgeht oder die Obergrenze für den Datensatzspeicher erreicht, dann versucht es, so viele Datensätze wie möglich freizugeben. In dem Fall schreibt MUIbase veränderte Datensätze auf Platte, um ein Maximum an verfügbaren Speicher zu erhalten. Dieser Vorgang kann über den Menüpunkt **‘Projekt – Datensätze auslagern’** erzwungen werden.

Wenn genug Speicher vorhanden ist, um alle Datensätze im Speicher zu halten und eine Obergrenze festgelegt wurde, die hoch genug ist (z.B. **‘unbegrenzt’**), dann braucht MUIbase keine Datensätze auszulagern.

MUIbase verwaltet eine Frei-Liste für jede Datensatzdatei. Ein gelöschter Datensatz wird in dieser Frei-Liste eingetragen. Auch bei einer Änderung eines Datensatzes, der auf die Platte geschrieben werden soll, wird der alte Platz der Datei in der Frei-Liste eingetragen. MUIbase stellt jedoch sicher, daß beim Neuladen immer zum Status der letzten Speicheroperation zurückgekehrt werden kann. Es wird keine Bereiche zerstören, die frei aber dennoch von einem Datensatz belegt sind, die beim Neuladen des Projekts erreicht werden könnten.

## 7 Einstellungen

MUIbase bietet diverse Einstellungen an, die der Benutzer nach seinen Wünschen setzen kann. Dieses Kapitel zeigt auf, welche Einstellungen verfügbar sind und gibt allgemeine Informationen, wie das Einstellungssystem arbeitet.

### 7.1 Datensatzspeicher

MUIbase braucht nicht alle Datensätze eines Projekts im Speicher halten. Stattdessen verwendet es einen Puffer, der nur eine kleine Anzahl von Datensätzen hält. Durch das Auswählen aus dem Menü **‘Einstellungen – Datensatzspeicher’** kann die Größe dieses Puffers gesetzt werden. Jedes Projekt hat seinen eigenen Puffer, d.h. wenn zwei Projekte geöffnet sind, die beide jeweils einen Datensatzspeicher von 1MB haben, dann wird MUIbase bis zu 2MB für die Datensätze beider Projekte benutzen.

MUIbase reserviert den Speicher nicht am Stück, sondern verwendet ein dynamisches Allokierungsschema. Die festgelegte Puffergröße ist auch keine feste Grenze. Wenn MUIbase der Meinung ist, daß es mehr Speicher benötigt, dann versucht es, diesen zu reservieren.

Ist der Puffer einmal gefüllt oder MUIbase erhält keinen Speicher mehr, dann werden alle Datensätze aus dem Puffer gelöscht. Dies bedeutet, daß unveränderte Datensätze einfach freigegeben und veränderte Datensätze zuerst auf Platte geschrieben und dann erst freigegeben werden.

Je größer der Wert für den Puffer ist, desto größer ist der Geschwindigkeitszuwachs beim Zugriff auf die Datensätze, da dann mehr Datensätze im Speicher gehalten werden und wenige Datensätze von Platte geladen werden müssen. Wenn die Datensatzspeichergröße auf **‘unbegrenzt’** gesetzt ist und alle Datensätze in den Speicher passen, dann arbeitet MUIbase am schnellsten.

### 7.2 Datensätze löschen bestätigen

Der Menüpunkt **‘Einstellungen – Datensätze löschen bestätigen’** sollte gesetzt sein, wenn MUIbase bei jedem Löschen eines Datensatzes eine Sicherheitsabfrage aktivieren soll. Ungesetzt wird jeder Datensatz stillschweigend gelöscht. Voreingestellt ist gesetzt.

### 7.3 Externer Editor zum Programmieren

Es kann der Menüpunkt **‘Einstellungen – Externer Editor zum Programmieren?’** gesetzt sein, wenn immer der externe Editor zum Bearbeiten von MUIbase-Programmen verwendet werden soll. Dies bedeutet, daß bei jedem Öffnen des Programmeditors automatisch auch der externe Editor gestartet wird. Wenn der Menüpunkt nicht gesetzt ist, kann der externe Editor dennoch gestartet werden, in dem der entsprechende Menüpunkt im Kontextmenü des Editors ausgewählt wird. Voreingestellt ist ungesetzt.

## 7.4 Icon erstellen

Durch das Setzen des Menüpunktes **‘Einstellungen – Icon erstellen?’** erstellt MUIbase für jedes Projekt ein Icon. Selbst erstellte Icons können für ein Projekt verwendet werden. MUIbase wird existierende Icon-Bilder solange nicht ersetzen, wie sie Projekt-Icons sind. Voreingestellt ist gesetzt.

## 7.5 Formate

Über die Menüpunktauswahl von **‘Einstellungen – Formate setzen...’** können die Formate zum Darstellen von Fließkommazahlen und Daten<sup>1</sup> festgelegt werden. Nach dem Auswählen erscheint ein Fenster, das folgende Punkte enthält:

- ein Feld **‘Fließkommazahlenformat’** zum Setzen des Dezimalzeichens von Fließkommazahlen. Es kann **‘Dezimalpunkt’** und **‘Dezimalkomma’** gewählt werden.
- ein Feld **‘Datumsformat’** zur Festlegung, wie Datumsangaben ausgegeben werden. Es kann zwischen **‘Tag.Monat.Jahr’**, **‘Monat/Tag/Jahr’** und **‘Jahr-Monat-Tag’** gewählt werden.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters.

Die Vorgabewerte für Fließkommazahlen und Datumsformate werden aus den Informationen ermittelt, die in der Betriebssystembibliothek **‘locale.library’** vermerkt sind.

Wenn alle Einstellungen getätigt wurden, verläßt man über **‘Ok’** das Fenster und aktualisiert die Anzeige.

## 7.6 Standardprogramm

Die Auswahl des Menüpunktes **‘Einstellungen – Standardprogramm setzen...’** erlaubt das Setzen des Standardprogrammes, das ausgeführt wird, wenn auf das Projekticon doppelgeklickt wird. Normalerweise sollte der Pfad zum MUIbase-Programm, z.B. **‘MUIBASE:MUIbase’**, gesetzt werden, das auch die Voreinstellung ist.

## 7.7 Externer Editor

Das Editorfeld von MUIbase bietet ein spezielles Menü, in dem ein externer Editor aufgerufen und mit dem der Text bearbeitet werden kann. Der Name des Editors mit seinen Parametern muß über den Menüpunkt **‘Einstellungen – Externen Editor setzen...’** angegeben werden. Es sollte ein Befehl eingegeben werden, der ausgeführt wird, wenn der externe Editor aufgerufen wird. **‘%f’** muß an Stelle des Dateinamens stehen. Die Zeichenkette **‘%f’** wird durch einen momentanen Dateinamen einer temporären Datei ersetzt, die MUIbase generiert, bevor der Befehl ausgeführt wird.

Um CED als externen Editor zu benutzen, kann z.B. **‘CED -keepio %f’** verwendet werden (es muß sichergestellt werden, daß die Stackgröße mindestens 8192 bytes beträgt, anderenfalls könnte CED abstürzen).

---

<sup>1</sup> Anm.d.Übersetzers: Auch hier ist die Mehrzahl von Datum gemeint :-)

Voreingestellt ist 'Ed %f'.

## 7.8 Externer Anzeiger

In MUIbase können Zeichenketten zum Speichern von Dateinamen verwendet werden. Zum Darstellen des Inhaltes einer Datei mit diesem Dateinamen ist ein externer Anzeiger notwendig. Normalerweise verwendet ein solcher Anzeiger das Datentypen-System des Amiga-OS, um Bilder bzw. Animationen anzuzeigen oder Musik abzuspielen. Zum Definieren des Anzeigers wird der Menüpunkt **'Einstellungen - Externen Anzeiger setzen...'** aufgerufen.

Wie beim externen Editor (siehe Abschnitt 7.7 [External editor], Seite 37) muß ein Befehl angegeben werden. Voreingestellt ist **'Multiview %f'**.

## 7.9 Popup-Knöpfe in die TAB-Kette

In der grafischen Benutzeroberfläche können Popup-Knöpfe vorhanden sein, z.B. Datei-, Zeichensatz- oder Listenansicht-Popups hinter einem Zeichenkettenfeld. Diese Knöpfe sind normalerweise nicht in der TAB-Kette eingebunden, was bedeutet, daß sie nicht mit der **Tab**-Taste erreichbar sind. Wird jedoch der Menüpunkt **'Einstellungen - Popup-Knöpfe in die TAB-Kette'** aktiviert, dann werden alle Popup-Knöpfe in die TAB-Kette eingefügt.

Zu beachten ist, daß das Ändern des Status dieses Menüpunkts nur dann eine Auswirkung hat, wenn die Benutzeroberfläche neu aufgebaut wird, z.B. durch das Wechseln zum Struktureditor und zurück zur Benutzeroberfläche.

## 7.10 Umschichten & Speichern bestätigen

Umschichten und Speichern eines Projekts kann etwas Zeit in Anspruch nehmen, je nach dem, wie groß das Projekt ist. Falls der Menüpunkt **'Projekt - Umschichten & Speichern'** oder **'Projekt - Umschichten & Speichern als ...'** ausgewählt wird, erscheint daher eine Sicherheitsabfrage, die diese Operationen erst bestätigen soll.

Diese Abfrage erscheint erst, wenn der Menüpunkt **'Einstellungen - Umschichten & Speichern bestätigen?'** gesetzt ist, aber durch das Nicht-Setzen des Menüpunktes wird die Abfrage deaktiviert.

## 7.11 Beenden bestätigen

Wenn versucht wird, MUIbase zu beenden und es liegen ungespeicherte Projekte vor, dann fragt MUIbase durch eine Sicherheitsabfrage um Erlaubnis. Das Programm beendet sich jedoch stillschweigend, falls alle Projekte schon gespeichert wurden.

Wenn MUIbase immer eine Sicherheitsabfrage anzeigen soll, wenn MUIbase beendet wird, dann muß der Menüpunkt **'Einstellungen - Beenden bestätigen?'** gesetzt sein. In diesem Fall erhält man immer eine Sicherheitsabfrage, wenn der Menüpunkt **'Projekt - Beenden'** ausgewählt wird.



Abgesehen davon läßt sich MUIbase dennoch stillschweigend beenden, indem man alle Projekte schließt.

## 7.12 Programm-Einfügedateienverzeichnis

Die Programmiermöglichkeiten von MUIbase erlauben es, externe Quellen in das Programm des Projekts einzubinden (siehe Abschnitt 15.2.3 [#include], Seite 76 für mehr Informationen). Der Menüpunkt **‘Programm - Einfügedateienverzeichnis’** erlaubt das Setzen eines Verzeichnisses, in dem MUIbase nach solchen Einfügedateien sucht. Voreingestellt ist **‘MUIbase:Include’**.

## 7.13 Programm-Debuginformation

Zum Kompilieren des Programms eines Projekts kann man wählen, ob man Debuginformationen in den ausführbaren Code einbinden soll oder nicht. Wird ohne Debuginformationen kompiliert und während der Laufzeit tritt ein Fehler auf, dann wird zwar eine Fehlermeldung generiert, aber keine Information ausgegeben, wo genau der Fehler stattfand. Wird mit Debuginformationen kompiliert, dann erhält man auch die genaue Fehlerposition.

Der Menüpunkt **‘Programm - Debuginformation’** schaltet die Debuginformation für die Kompilierung ein und aus. Nach dem Ändern dieses Status sollte man das Neukompilieren des Projektprogramms nicht vergessen, indem man den Menüpunkt **‘Programm - Kompilieren’** auswählt.

## 7.14 Programm-Ausgabedatei

Beim Ausführen eines MUIbase-Programms können sämtliche Ausgaben, die nach **‘stdout’** umgeleitet werden, in eine Datei ausgegeben werden. Der Dateiname muß im Eingabefenster eingegeben werden, das beim Menüpunkt **‘Programm - Ausgabedatei...’** erscheint. Es kann hier auch noch angegeben werden, ob die Ausgabe an eine bestehende Datei angehängt wird oder ob sie gelöscht wird, bevor Ausgaben ausgeführt werden. Neben den normalen Dateien erlaubt das Amiga-OS weitere spezielle Dateinamen, wie z.B.:

- **‘PRT:’** druckt die Ausgabe auf dem Drucker aus.
- **‘CON:////MUIbase output/CLOSE/WAIT’** gibt die Ausgabe in einem Shell-Fenster aus.
- **‘CONSOLE:’** gibt die Ausgabe in dem Shell-Fenster aus, in dem MUIbase gestartet wurde.

## 7.15 Projektabhängige Einstellungen

MUIbase kann mehrere Projekte verwalten und jedes Projekt kann seine eigenen Einstellungen besitzen. Manchmal jedoch ist es wünschenswert, daß alle Projekte sich gleiche Einstellungen für bestimmte Voreinstellungselemente teilen. Dies wird realisiert durch die Verwendung eines Statusflags für jedes Voreinstellungselement, welches anzeigt, ob jedes Projekt seine eigenen Werte für dieses Feld hat oder ob der Wert für alle Projekte gilt.

Durch die Auswahl des Menüpunktes **‘Einstellungen – Projektabhängige Einstellungen...’** erscheint ein Fenster mit einer kleinen Liste von Voreinstellungselementen. Alle Elemente der Liste können sowohl projektabhängig als auch global eingestellt werden. Die Liste enthält:

- Datensatzspeicher?
- Datensätze löschen bestätigen?
- Formate?
- Umschichten & Speichern bestätigen?
- Einfügedateienverzeichnis?
- Debug-Information?
- Programm-Ausgabedatei?

Die Statusflags der Voreinstellungselemente, die nicht in der Liste auftauchen, können nicht geändert werden, z.B. die Einstellung des Voreinstellungselements **‘Standardprogramm’** ist immer global, da es keinen Sinn macht, für verschiedene Projekte verschiedene Standardprogramme zu verwenden.

Wenn das Voreinstellungselement rechts daneben gesetzt ist, dann ist dieses Element projektabhängig, anderenfalls global. Projektabhängige Einstellungen werden in der Projektdatei **‘Structure.mb’** gespeichert, wohingegen globale Einstellungen in der globalen Voreinstellungsdatei gespeichert werden.

## 7.16 MUI

Da MUIbase eine MUI-Anwendung ist, lassen sich auch die MUI-Voreinstellungen für diese Anwendung verändern, indem der Menüpunkt **‘Einstellungen – MUI...’** ausgewählt wird.

## 7.17 Laden und Speichern der Einstellungen

Die globalen Voreinstellungen können von Platte geladen bzw. auf ihr gespeichert werden. Wenn MUIbase gestartet wird, lädt es automatisch die Voreinstellungen aus **‘ENV:MUIbase.prefs’**.

MUIbase speichert seine globalen Voreinstellungen nicht automatisch. Wenn Änderungen an den globalen Voreinstellungen durchgeführt wurden und MUIbase bei einem Neustart diese wieder verwendet soll, dann muß daher der Menüpunkt **‘Einstellungen – Einstellungen speichern’** benutzt werden. Die globalen Einstellungen werden in beide Dateien **‘ENVARC:MUIbase.prefs’** und **‘ENV:MUIbase.prefs’** gespeichert. Folgende Elemente werden gespeichert:

- Datensatzspeichergröße
- Datensätze löschen bestätigen
- Externer Editor zum Programmieren
- Icon erstellen
- Standardprogramm
- Formate
- Externer Editor

- Externer Anzeiger
- Umschichten & Speichern bestätigen
- Beenden bestätigen
- Einfügedateienverzeichnis
- Debug-Information
- Programm-Ausgabedatei
- Projektabhängige Einstellungen
- Verzeichnisnamen der Dateiauswahlfenster
- Vorgabewert der maximalen Länge bei neuen Zeichenkettenfeldern
- Vorgabetextformat für neue Feldobjekte
- Cursorposition im Programmeditor
- Name der Hilfedatei

Es gibt auch ein Menü mit dem Namen **‘Preferences - Einstellungen laden’**. Dieses wird verwendet, wenn geänderte Voreinstellungen wieder rückgängig gemacht und die zuletzt gespeicherten Werte wieder hergestellt werden sollen.

## 7.18 Bild für leere Anzeige

Wenn MUIbase ohne ein Projekt gestartet wird, dann öffnet MUIbase ein Fenster mit leerer Anzeige. Die leere Anzeige enthält normalerweise ein Bild, das aus der Datei **‘MUIbase:Images/EmptyWindow’** geladen wird.

Dieses Bild hat als Standard das MUIbase-Logo, aber es läßt sich durch ein anderes bevorzugtes Bild ersetzen, indem es in die Datei **‘MUIbase:Images/EmptyWindow.iff’** kopiert wird. Es ist auch möglich, daß kein Bild für die leere Anzeige verwendet wird; dazu muß nur die genannte Datei gelöscht bzw. so umbenannt werden, daß sie nicht gelesen werden kann. In dem Fall gibt MUIbase nur den Text **‘Keine Anzeige’** im Fenster aus.

## 8 Datensatzbearbeitung

Dieses Kapitel beschreibt, wie Datensätze einer Tabelle hinzugefügt, verändert, gelöscht und durchgeforstet werden.

### 8.1 Aktive Objekte

MUIbase verwendet einen Cursor, um anzuzeigen, welches Objekt das gerade aktive ist. Wenn das aktive Objekt ein Zeichenkettenobjekt ist, dann erscheint ein normaler Blockcursor, andere Objekte erhalten einen besonderen Rahmen um sie. Durch das Drücken der Tasten `<TAB>` oder `<SHIFT TAB>` kann zu den aktiven Objekten gesprungen werden.

Die Tabelle, in der das aktive Objekt liegt, wird *aktive Tabelle* genannt. Das Panel einer Tabelle kann als aktives Objekt gesetzt werden. Damit wird garantiert, daß immer eine Tabelle als die aktive gesetzt werden kann, obwohl die Tabelle vielleicht keine aktivierbaren Objekte beinhaltet.

### 8.2 Datensätze hinzufügen

Wenn der Menüpunkt **‘Tabelle - Neuer Datensatz’** ausgewählt wird, wird ein neuer Datensatz in der aktiven Tabelle angelegt. Dieser Datensatz wird mit den Vorgabewerten aller Felder vorbelegt. Es ist auch durch den Menüpunkt **‘Tabelle - Datensatz kopieren’** möglich, den momentanen Datensatz zu vervielfältigen.

Wurde eine Auslösefunktion für das Hinzufügen von Datensätzen eingerichtet (siehe Abschnitt 14.1.1 [Creating tables], Seite 62), dann wird diese Auslösefunktion zum Erzeugen des Datensatzes aufgerufen. Mehr über diesen Mechanismus, Siehe Abschnitt 15.28.4 [New trigger], Seite 157.

### 8.3 Datensätze verändern

Um den aktuellen Datensatz einer Tabelle zu verändern, läßt sich jedes Feld innerhalb der Tabellenmaske aktivieren und ein neuer Wert eingeben. Für Zeichenketten, Ganzzahlen, Fließkommazahlen, Daten<sup>1</sup>, Zeiten und mehrzeilige Zeichenketten können die üblichen Editierbefehle verwendet werden.

Ein Feldobjekt kann auch als Nur-Lesen konfiguriert worden sein. In diesem Fall kann der Wert des Feldes nicht verändert werden (Ausnahme: Zeichenketten mit einem Popup-Knopf).

### 8.4 Zeichenkettenfelder mit einem Popup-Knopf

Wenn ein Zeichenkettenfeld einen Popup-Knopf zugewiesen bekommen hat, dann erscheint auf Druck des Popup-Knopfes eine Meldung, um den Zeichenketteninhalt zu setzen, wie z.B. ein

---

<sup>1</sup> Anm.d.Übersetzers: Auch hier sei angemerkt, daß es sich um die Mehrzahl von **‘Datum’** handelt :-)

Dateiauswahlfenster zum Auswählen eines Dateinamens oder eine Liste mit Zeichenketten, um daraus eine auszuwählen. Der Popup-Knopf kann immer verwendet werden, um den Wert des Zeichenkettenfelds zu setzen, auch dann, wenn das Feld auf nur-lesen gesetzt ist.

Rechts neben dem Zeichenkettenfeld kann ein ‘A’ erscheinen. Mit diesem Knopf wird ein externer Anzeiger gestartet, mit dem der Inhalt der Datei angezeigt werden kann, der im Zeichenkettenfeld angegeben ist.

## 8.5 Eingabe von booleschen Werten

Der gesetzte Status eines booleschen Feldes kann mit der linken Maustaste oder mit der Leertaste, falls das Objekt aktiv ist, umgeschaltet werden.

## 8.6 Eingabe von Auswahlwerten

Bei Auswahlfeldern kann ein Wert durch Anklicken oder mit den Tasten `<UP>` und `<DOWN>` zum Durchforsten aller Auswahllemente ausgewählt werden.

## 8.7 Eingabe von Datumswerten

Datumswerte können im Format ‘DD.MM.YYYY’ eingegeben werden, wobei ‘DD’, ‘MM’ und ‘YYYY’ für 2- bzw. 4-stellige Zahlen stehen, die den Tag, Monat bzw. Jahr des Datums repräsentieren. Es ist zulässig, die Jahresangabe wegzulassen. In dem Fall wird das aktuelle Jahr zur eingegebenen Zeichenkette angehängt.

Durch die Eingabe einer einfachen Ganzzahl kann ein Datumswert relativ zum aktuellen Datum angegeben werden, z.B. wird bei der Eingabe von ‘0’ das heutige Datum verwendet oder bei der Eingabe von ‘-1’ das gestrige.

## 8.8 Eingabe von Zeitwerten

Zeitwerte können im Format ‘HH:MM:SS’ eingegeben werden, wobei ‘HH’ ein 2-stelliger Wert im Bereich von 0 bis 23 für die Stunden, ‘MM’ ein 2-stelliger Wert im Bereich von 0 bis 59 für die Minuten und ‘SS’ ein 2-stelliger Wert im Bereich von 0 bis 50 für die Sekunden darstellt.

Es ist auch möglich, die Stunden und Minuten wegzulassen. In diesen Fällen wird eine 0 angenommen. Wird z.B. ‘6:30’ eingegeben, so wird automatisch auf ‘00:06:30’ erweitert.

## 8.9 Kontextmenü vom mehrzeiligen Textfeld

Mehrzeilige Textfelder besitzen ein Kontextmenü, das weitere Editiermöglichkeiten anbietet:

- ‘Ausschneiden’, ‘Kopieren’, and ‘Einfügen’ erlauben den Datenaustausch mit der Zwischenablage.

- ‘Löschen’ löscht den gesamten Text im mehrzeiligen Textfeld.
- ‘Rückgängig’ und ‘Wiederherstellen’ erlauben das Vor- und Zurückspringen der Änderungen, die am Textfeldinhalt gemacht wurden
- Mit ‘Text laden’ und ‘Text speichern’ kann der Inhalt des mehrzeiligen Textfelds von/zur einer Datei geladen bzw. gespeichert werden.
- ‘Externer Editor’ startet einen externen Editor zum Bearbeiten des mehrzeiligen Textfeldes. Siehe Abschnitt 7.7 [External editor], Seite 37 für weitere Informationen zum externen Editor.

## 8.10 Eingabe von Beziehungswerten

Für Beziehungsfelder gibt es mehrere Möglichkeiten, einen Wert einzugeben:

- Rechts vom Beziehungsfeld befindet sich ein Popup-Knopf, der auf Knopfdruck eine Liste von Datensätzen und zwei weitere Knöpfe öffnet. Die Auswahl eines Datensatzes aus der Liste setzt die Beziehung auf diesen Datensatz, ‘Voreingestellt’ auf NIL oder ‘Aktuelles’ auf den aktuellen Datensatz der referenzierten Tabelle.
- Das Kontextmenü des Beziehungsfeldes erlaubt weitere Optionen, eine Beziehung zu setzen. Dort sind auch die entsprechenden Tastencodes zu finden.
  - Die Menüpunkte ‘Vorhergehender’ und ‘Nächster’ setzen die Beziehung auf den vorhergehenden oder nächsten Datensatz.
  - Die Menüpunkte ‘Rückwärts’ und ‘Vorwärts’ erlauben das Rück- oder Vorspringen um 10 Datensätze.
  - Der Menüpunkt ‘Suchen nach...’ öffnet ein Fenster, in dem ein Muster eingegeben werden kann, um nach einem Datensatz zu suchen. Wenn der Punkt ‘Alle Felder?’ nicht aktiviert ist, dann wird die Suche nur auf das Feld durchgeführt, das als erstes in der Liste der Felder steht, die zum Sortieren der referenzierten Tabelle verwendet wird. Das Durchsuchen dieses einen Feldes hat den Grund, daß die Suche von Anfangswörtern sehr effizient auf dieses Feld durchgeführt werden kann, weil dieses sich wie ein Index verhält. Mehr Informationen zum Suchfenster siehe Kapitel 11 [Search for], Seite 52.
  - Wenn einmal ein Suchmuster eingegeben wurde, dann kann ‘Suche rückwärts’ und ‘Suche vorwärts’ verwendet werden, um den vorhergehenden oder nachfolgenden passenden Datensatz zu finden.
- Wenn das Beziehungsfeld das aktive Objekt ist, dann kann ein Zeichen eingegeben werden, das nach einem Datensatz im ersten Feld der referenzierten Tabelle sucht, dessen Inhalt mit dem eingegebenen Zeichen beginnt. Die Suche wird größenunabhängig durchgeführt. Bei der Eingabe von z.B. ‘1’ wird das Suchmuster ‘1\*’ für die Suche verwendet. Weitere Zeichen können zum Suchpuffer ergänzt werden, wenn die Zeichen in Großbuchstaben eingegeben werden; z.B. sucht ‘1ASSIE’ nach einem Datensatz, der auf die Zeichenkette ‘1assie\*’ paßt.

## 8.11 Eingabe von NIL-Werten

Um einen NIL-Wert einzugeben, wird jede eingegebene ungültige Zeichenkette für den gegebenen Feldtyp, z.B. bei der Eingabe von ‘xyz’ in einem Ganzzahlfeld, der Wert des Feldes auf NIL gesetzt. Es ist zu beachten, daß nicht alle Feldtypen den NIL-Wert unterstützen. Siehe Abschnitt 5.6 [Table of attribute types], Seite 25 für einen Überblick über alle Feldtypen.

## 8.12 Datensätze löschen

Um den aktuellen Datensatz zu löschen, wird der Menüpunkt ‘**Tabelle – Datensatz löschen**’ ausgewählt. Vor dem Löschen des Datensatzes kann ein Sicherheitsfenster erscheinen, das um Erlaubnis fragt. Dieses Fenster kann über die Einstellungen ein- und ausgeschaltet werden (siehe Abschnitt 7.2 [Record delete requester], Seite 36).

Wurde eine Auslösefunktion zum Löschen von Datensätzen eingerichtet (siehe Abschnitt 14.1.1 [Creating tables], Seite 62), dann wird diese Auslösefunktion zum Löschen des Datensatzes ausgeführt. Mehr Informationen zu diesem Mechanismus, Siehe Abschnitt 15.28.5 [Delete trigger], Seite 158.

Es ist auch möglich, alle Datensätze aller Tabellen zu löschen, indem man den Menüpunkt ‘**Tabelle – Alle Datensätze löschen**’ aufruft. Nur die Datensätze, die dem Datensatzfilter der betreffenden Tabelle genügen, werden gelöscht. Vor dem Löschen erscheint ein Sicherheitsfenster, sofern aktiviert. Es wird keine Auslösefunktion ausgeführt, wenn alle Datensätze gelöscht werden.

## 8.13 Datensätze durchforsten

Um andere Datensätze als den gerade angezeigten zu sehen, wählt man einen der Unterpunkte des Menüpunktes ‘**Tabelle – Gehe zum Datensatz**’. Man kann zum vorhergehenden, nächsten, ersten oder letzten Datensatz, mehrere Datensätze zurück- oder vorspringen oder die Nummer des Datensatzes eingeben, den man sehen möchte. Die Datensatznummer in diesem Zusammenhang ist die Nummer, die im dazugehörigen Panel des Datensatzes angezeigt wird (siehe Abschnitt 5.9.3 [Panels], Seite 30). Das Panel kann auch zwei Pfeilknöpfe enthalten, um zum vorhergehenden und nächsten Datensatz zu springen.

Durchforsten der Datensätze läßt sich einfach mit den Cursortasten *UP* und *DOWN* in Verbindung mit den Tasten *SHIFT*, *ALT* und *CONTROL* durchführen. Alle möglichen Kombinationen sind im Menüpunkt ‘**Tabelle – Gehe zum Datensatz**’ und in der folgenden Tabelle aufgeführt:

	<i>ALT</i>	<i>CONTROL-ALT</i>	<i>SHIFT-ALT</i>
<i>UP</i>	Vorhergehender Datensatz	Erster Datensatz	Springe zurück
<i>DOWN</i>	Nächster Datensatz	Letzter Datensatz	Springe vorwärts

## 9 Filter

Filter können verwendet werden, um Datensätze auszublenden. Dieses Kapitel beschreibt, welche Typen von Filter es gibt und wie sie angewandt werden.

MUIbase kennt zwei Arten von Filter: Datensatzfilter und Referenzfilter.

### 9.1 Datensatzfilter

Ein Datensatzfilter kann in eine Tabelle eingebaut werden, um Datensätze herauszufiltern, die nicht von Interesse sind. Datensätze, die herausgefiltert werden, sind aus der Tabellenmaske ausgenommen und können daher vom Benutzer nicht angesehen bzw. durchgeforstet werden.

#### 9.1.1 Filterausdruck

Ein Filter wird durch die Angabe eines booleschen Ausdruckes festgelegt, der Funktionsaufrufe zu MUIbase-Programmierfunktionen beinhalten kann. Für jeden Datensatz in der Tabelle, zu der der Filter festgelegt wurde, wird dieser Ausdruck ausgewertet. Wenn er NIL liefert, dann wird der Datensatz ausgenommen, anderenfalls wird der in die Tabellenmaske übernommen.

Jede Tabelle kann seinen eigenen Filterausdruck besitzen.

#### 9.1.2 Filter ändern

Um den Filter einer Tabelle zu ändern, wählt man den Menüpunkt ‘Tabelle – Ändere Filter...’. Dies öffnet ein Fenster, das folgende Punkte enthält:

- den Namen der Tabelle im Fenstertitel, zu der der Filter installiert werden soll.
- eine Liste aller Felder der Tabelle, die im Filterausdruck verwendet werden können. Diese Liste ist im linken Teil des Fensters angeordnet. Wenn auf einen Namen doppelt geklickt wird, dann wird der Name im Filterausdruck an der aktuellen Cursorposition eingefügt.
- Eine Sammlung von Knöpfen, die MUIbase-Programmierfunktionen und -Operatoren anzeigen, die im rechten Teil des Fensters plziert sind. Nach einem Klick wird die entsprechende Funktion zum Filterausdruck hinzugefügt. Anzumerken ist, daß die Liste der Funktionen nicht vollständig ist. Andere MUIbase-Funktionen müssen daher von Hand eingegeben werden. Es können nur solche MUIBase-Funktionen verwendet werden, die keine Seiteneffekte haben; z.B. ist es nicht möglich, in einem Filterausdruck Daten in eine Datei zu schreiben.
- ein Zeichenkettenfeld für die Eingabe des Filterausdruckes. Feld- und Funktion/Operatormenamen werden hier eingefügt. Man kann auch direkt dort einen Filterausdruck eingeben.
- zwei Knöpfe ‘Ok’ und ‘Abbrechen’, um das Fenster verlassen zu können.

Nach der Festlegung des Filterausdruckes klickt man auf ‘Ok’, um das Fenster zu verlassen. Der eingegebene Ausdruck wird kompiliert und dann wird -wenn erfolgreich kompiliert- der Ausdruck für alle Datensätze ausgewertet. Die Datensätze, für die der boolesche Ausdruck nicht gilt, werden aus der Tabellenmaske herausgehalten.



Falls der Ausdruck nicht kompiliert werden konnte, erhält man eine Nachricht, die in der Titelleiste des Fensters angezeigt wird.

Ein Filter kann über den Knopf ‘F’ im Panel der Tabelle, wenn er installiert wurde, oder durch die Taste *F* ein- und ausgeschaltet werden, falls das Panel das aktive Objekt ist. Nachdem ein Filterausdruck für eine Tabelle festgelegt wurde, wird er Filter für diese Tabelle automatisch aktiviert.

Wenn ein Filter (de)aktiviert wird, dann werden alle Datensätze geprüft, ob sie dem Filter genügen oder nicht.

Wenn ein Filter aktiv ist und ein (filter-relevantes) Feld in einem Datensatz dieser Tabelle geändert wird, dann wird der Filterstatus des Datensatzes nicht neu berechnet und bleibt unverändert.

Wenn ein neuer Datensatz in einer Tabelle mit einem aktivierten Filter hinzugefügt wird, dann wird keine Überprüfung durchgeführt, ob der neue Datensatz dem Filter genügt und der neue Datensatz erhält den Filterstatus TRUE.

### 9.1.3 Filterbeispiele

Hier ein paar Beispiele für gültige Filterausdrücke:

- ‘NIL’ filtert alle Datensätze heraus<sup>1</sup>.
- ‘TRUE’ filtert keinen Datensatz heraus<sup>2</sup>.
- ‘0’ entspricht ‘TRUE’, sowie für alle Ausdrücke die einen Wert ungleich NIL liefern, die in MUIbase auch als TRUE betrachtet werden.
- ‘(> Wert 100.0)’ zeigt nur die Datensätze an, bei denen das Feld ‘Wert’ größer als 100.0 ist (wir setzen hier voraus, daß die Tabelle ein Feld ‘Wert’ vom Typ Fließkommazahl besitzt).
- ‘(NOT (LIKE Name "\*x\*))’ filtert alle Datensätze heraus, die den Buchstaben ‘x’ im Feld ‘Name’ (ein Zeichenkettenfeld) haben.

Es ist zu beachten, daß MUIbase’s Programmiersprache eine lisp-ähnliche Syntax hat<sup>3</sup>. Mehr dazu im Kapitel siehe Kapitel 15 [Programming MUIbase], Seite 75.

## 9.2 Referenzfilter

Beziehungsfelder besitzen auch einen Filter. Dies ist nützlich, wenn Tabellen hierarchisch aufgebaut werden sollen (wie in AmigaBase fest implementiert). Das Projekt ‘Albums’ ist ein Beispiel dafür.

Wenn der Filter eines Beziehungsfeldes aktiviert ist, dann werden folgende Eigenschaften mit eingeschaltet:

---

<sup>1</sup> Anm.d.Übersetzers: Dies hat zur Folge, daß kein Datensatz angezeigt würde.

<sup>2</sup> Anm.d.Übersetzers: Hier werden alle Datensätze angezeigt.

<sup>3</sup> Anm.d.Übersetzers: Die Installer-Scripte für den Installer von Amiga Intl. benutzen einen ähnlichen Aufbau der Funktionen.

1. Der Benutzer kann nur auf Datensätze in der Tabelle des Feldes zugreifen, die eine Referenz auf den aktuellen Datensatz der referenzierten Tabelle haben<sup>4</sup>.
2. Wenn die referenzierte Tabelle ihren aktuellen Datensatz ändert, dann wird auch ein neuer Datensatz für die Tabelle des Feldes gesucht und gesetzt<sup>5</sup>.
3. Wenn ein neuer Datensatz angelegt wird, dann wird die Beziehung automatisch auf den aktuellen Datensatz der referenzierten Tabelle gesetzt<sup>6</sup>.

Hinweis: Ein verschachteltes Löschen (wie es in AmigaBase automatisch gemacht wurde) muß vom Benutzer selbst implementiert werden (indem man die Auslösefunktion für das Löschen von Datensätzen verwendet).

Es sollten keine Referenzfilter für zyklische Pfade, wie z.B. Referenzen auf die eigene Tabelle, verwendet werden. Dies macht keinen Sinn und verwirrt nur die Benutzer.

---

<sup>4</sup> Anm.d.Übersetzers: Um diesen Sachverhalt etwas klarer darzustellen, folgendes Beispiel: Tabelle EINS hat ein Beziehungsfeld auf Datensätze der Tabelle ZWEI. In Tabelle EINS werden dann beim Filtern nur die Datensätze angezeigt, für die in Tabelle EINS eine Referenz auf den gerade angezeigten Datensatz in Tabelle ZWEI existiert.

<sup>5</sup> Anm.d.Übersetzers: Auch dies hier etwas klarer: Tabelle EINS hat ein Beziehungsfeld auf Datensätze der Tabelle ZWEI. Springt man in Tabelle ZWEI z.B. auf den nächsten Datensatz, so werden in Tabelle EINS für den neuen Datensatz in Tabelle ZWEI diejenigen Datensätze angezeigt, für die in Tabelle EINS eine Referenz auf den gerade angezeigten Datensatz in Tabelle ZWEI existiert.

<sup>6</sup> Anm.d.Übersetzers: Auch hier machen wir unser Spiel mit dem Klarmachen weiter: Tabelle EINS hat ein Beziehungsfeld auf Datensätze der Tabelle ZWEI. Fügt man nun in Tabelle EINS einen neuen Datensatz hinzu, so erhält dieser automatisch die Referenz auf den gerade in Tabelle ZWEI angezeigten Datensatz.

## 10 Sortieren

Für jede Tabelle einer Datenbank läßt sich festlegen, in welcher Reihenfolge dessen Datensätze angezeigt werden sollen. Dieses Kapitel beschreibt, wie eine Reihenfolge festgelegt werden und welche Konsequenzen dies haben kann.

### 10.1 Keine Sortierung

Standardmäßig besitzt jede neu erzeugte Tabelle keine Sortierung. Dies bedeutet, daß beim Einfügen eines neuen Datensatzes der generierte Datensatz an der aktuellen Position, d.h. hinter dem aktuellen Datensatz, eingefügt wird. Beim Verändern der Felder eines Datensatzes verändert sich die Position des Datensatzes innerhalb der Tabelle nicht.

### 10.2 Sortieren nach Feldern

Manchmal ist es sinnvoll, die Datensätze nach bestimmten Feldern zu sortieren, z.B. nach dem Feld 'Name', wenn die Tabelle ein solches hat.

In MUIbase läßt sich für jede Tabelle eine Liste von Feldern festlegen, nach denen die Datensätze sortiert werden sollen. Alle Datensätze werden zuerst nach dem ersten Feld dieser Liste sortiert. Falls zwei Datensätze in einem Feld gleich sind, dann legt das nächste Feld in der Liste die Reihenfolge fest. Für jedes Feld läßt sich zudem festlegen, ob die Datensätze auf- oder absteigend sortiert werden sollen.

Um die Reihenfolge zu erkennen, werden die Regeln der folgenden Tabelle verwendet:

Typ	Reihenfolge
Ganzzahl Auswahl	NIL < MIN_INT < ... < -1 < 0 < 1 < ... < MAX_INT (Auswahlwerte werden als Ganzzahlen angesehen)
Fließkommazahl	NIL < -HUGE_VAL < ... < -1.0 < 0.0 < 1.0 < ... < HUGE_VAL
Zeichenkette mehrz. Text	NIL < "" < ... < "a" < "AA" < "b" < ... (Zeichenkettenvergleich wird größenunabhängig durchgeführt)
Datum	NIL < 1.1.0000 < ... < 31.12.9999
Zeit	NIL < 00:00:00 < ... < 23:59:59
Bool	NIL < TRUE
Beziehung	NIL < any_record (Datensätze selbst können nicht zum Sortieren verwendet werden)

Wenn eine Sortierung für eine Tabelle festgelegt wurde, dann werden die Datensätze automatisch neu angeordnet, wenn ein neuer Datensatz hinzugefügt oder ein Feld eines Datensatzes verändert wird, das für die Sortierung relevant ist.

### 10.3 Sortieren nach einer Funktion

Manchmal ist ein komplexeres Sortierungsschema als die einfache Felderliste nützlich, die im vorherigen Abschnitt beschrieben wurde. Beispielsweise kann die Felderliste keine Beziehungsfelder aufnehmen, so daß es nicht möglich ist, die Datensätze nach einem Beziehungsfeld zu sortieren. Der Grund liegt darin, daß MUIbase mit Beziehungsfeldern nicht in der Lage sein kann, alle Datensätze zu jeder Zeit sortiert zu halten (siehe Abschnitt 15.28.6 [Comparison function], Seite 158 im Abschnitt über die Programmierung von MUIbase für mehr Details).

MUIbase bietet nun jedoch auch die Möglichkeit, eine Vergleichsfunktion zum Sortieren der Datensätze anzugeben. Es kann jede Funktion angegeben werden, die im Programmeditor von MUIbase geschrieben wurde. Die Funktion wird mit zwei Datensatz-Zeigern aufgerufen und der Rückgabewert soll die Sortierung der beiden Datensätze anzeigen. Die Funktion kann jede Operation zum Vergleich von Datensätzen verwenden, so daß es z.B. auch Datensätze mit Beziehungsfeldern vergleichen kann. Mehr zu diesem Mechanismus, siehe Abschnitt 15.28.6 [Comparison function], Seite 158.

Falls eine Vergleichsfunktion zum Sortieren einer Tabelle verwendet wird, dann sollte man beachten, daß MUIbase nicht immer erkennen kann, wann Datensätze der Tabelle sortiert werden müssen, da die Abhängigkeiten unbekannt sind. Wenn Datensätze unsortiert sind, dann ruft man den Menüpunkt **‘Tabelle - Alle Datensätze neu sortieren’** auf.

### 10.4 Sortierung ändern

Um eine Sortierung für die aktuelle Tabelle zu erstellen, wird der Menüpunkt **‘Tabelle - Ändere Sortierung...’** ausgewählt. Dies öffnet ein Fenster, das folgende Punkte enthält:

- den Namen der Tabelle in der Titelleiste des Fensters.
- ein Auswahlfeld **‘Typ’**, in dem festgelegt wird, ob die Tabelle anhand der **‘Felderliste’** oder durch Verwendung der **‘Vergleichsfunktion’** sortiert werden soll. Abhängig vom Zustand von **‘Typ’**, können Daten in die folgenden Elemente eingegeben werden.

Für eine Sortierung anhand einer Felderliste existieren folgende Elemente:

- eine Liste alle Felder der Tabelle, die für die Sortierliste verwendet werden können. Diese Liste ist links im Fenster angeordnet. Wenn auf einen Namen doppelt geklickt wird, dann wird der Name in der Sortierliste an der aktuellen Cursorposition eingefügt.
- die aktuelle Liste der Felder, die zum Sortieren verwendet wird. Diese Liste ist rechts im Fenster platziert. Das oberste Element in dieser Liste ist das erste Feld der Sortierliste. Die Reihenfolge der Elemente läßt sich durch Verschieben an andere Positionen in der Liste durchführen. Weitere Felder können auch durch Verschieben von der Feldliste in die Sortierliste hinzugefügt werden. Entfernen von Feldern aus der Sortierliste wird durch das Herausschieben des Feldes aus der Sortierliste und Ablegen desselben in der Feldliste bewerkstelligt.

Jeder Eintrag in der Sortierliste hat auf der linken Seite ein Pfeilsymbol, das nach oben oder unten zeigt. Der Status läßt sich durch Doppelklicken auf das Pfeilsymbol umschalten. Wenn der Pfeil nach oben zeigt, dann ist die Sortierreihenfolge dieses Feldes aufsteigend, zeigt er nach unten, dann absteigend.

Für eine Sortierung unter Verwendung einer Vergleichsfunktion gibt es folgende Elemente:

- ein Feld **‘Funktion für den Datensatzvergleich’**, in dem der Name der Funktion eingegeben wird, die zum Vergleichen zweier Datensätze der Tabelle aufgerufen werden soll. Es kann der Popup-Knopf rechts neben dem Zeichenkettenfeld verwendet werden, um einen Namen aus der Liste aller Funktionen auszuwählen. Bleibt das Feld leer, dann wird eine leere Sortierung verwendet<sup>1</sup> Mehr über die Anwendung der Vergleichsfunktion, einschließlich der Argumente, die ihr übergeben werden, siehe Abschnitt 15.28.6 [Comparison function], Seite 158.

Des weiteren existieren folgende Knöpfe:

- ein Knopf **‘Löschen’**, der alle Felder für eine leere Sortierung löscht.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters.

Um eine neue Feldsortierliste einzugeben, wählt man **‘Feldliste’** im Feld **‘Typ’** und drückt den **‘Löschen’**-Knopf. Um eine neue Sortierliste zu erzeugen, drückt man zuerst den Knopf **‘Löschen’**. Anschließend wird wie oben beschrieben per Verschieben & Ablegen eine neue Liste von Feldern angelegt. Wenn keine Sortierung gewünscht ist, dann fügt man einfach keine Felder der Sortierliste hinzu.

Ist die Sortierung festgelegt, wird der Knopf **‘Ok’** gedrückt. MUIbase sortiert dann alle Datensätze der Tabelle. Da dies etwas Zeit in Anspruch nimmt, wird ein Mauszeiger mit Wartesymbol erscheinen.

## 10.5 Neu sortieren aller Datensätze

Falls man das Gefühl hat, daß einige Datensätze nicht sortiert seien, wenn man z.B. eine Vergleichsfunktion zum Sortieren verwendet, dann können über den Menüpunkt **‘Tabelle - Neu sortieren aller Datensätze’** alle Datensätze neu sortiert werden.

---

<sup>1</sup> Anm.d.Übersetzers: Bedeutet soviel, daß die Datensätze nicht sortiert werden.

## 11 Suchen

Zum Durchforsten von Datensätzen kann ein Suchfenster verwendet werden, um nach einem bestimmten Datensatz zu suchen. Die Suche verwendet ein Suchmuster, das bereitgestellt wird und prüft alle Datensätze auf einen erfolgreichen Vergleich mit diesem Muster. Wenn es einen findet, dann wird der Datensatz in der Tabellenmaske dargestellt.

### 11.1 Suchfenster

Um das Suchfenster zu öffnen, wählt man den Menüpunkt **‘Tabelle – Suchen nach...’**. Dies öffnet ein Fenster, das die folgenden Punkte enthält:

- ein Zeichenkettenfeld, um das Suchmuster eingeben zu können. Die Zeichen **‘\*’** und **‘?’** können als Jokerzeichen verwendet werden. Das Zeichen **‘\*’** ersetzt jede Anzahl von Zeichen (einschließlich keine Zeichen), wohingegen **‘?’** genau ein Zeichen ersetzt.
- ein Feld **‘GROß/klein beachten?’**. Wenn markiert, dann verwendet die Suche einen größenabhängigen Zeichenkettenvergleich, anderenfalls einen größenunabhängigen.
- ein Feld **‘Alle Felder?’**. Wenn markiert, dann werden alle Felder eines Datensatzes nach einem erfolgreichen Vergleich mit dem Suchmuster hergenommen. Im anderen Fall wird nur das Feld durchsucht, das aktiv war, als das Suchfenster geöffnet wurde. Falls das aktive Objekt beim Öffnen des Suchfensters kein Feldobjekt war, dann wird das Feld automatisch markiert und deaktiviert.
- zwei Radioknöpfe für die Suchrichtung **‘Vorwärts’** und **‘Rückwärts’**.
- zwei Radioknöpfe zum Festlegen des Suchstartpunktes. Bei **‘ersten/letzten Datensatz’** beginnt je nach Suchrichtung die Suche beim ersten oder letzten Datensatz. Bei **‘aktuellen Datensatz’** startet die Suche beim gerade aktuellen Datensatz.
- zwei Knöpfe **‘Suchen’** und **‘Abbrechen’** zum Verlassen des Fensters.

Nachdem ein Suchmuster eingegeben und das Fenster mit **‘Suchen’** verlassen wurde, startet MUIbase mit der Suche nach einem passendem Datensatz. Der Vergleich eines Feldes mit dem Suchmuster wird immer zeichenkettenbasiert durchgeführt, d.h. Felder mit Datentypen, die keine Zeichenketten sind, werden erst in Zeichenketten umgewandelt.

Wird ein passender Datensatz gefunden, dann wird dieser als aktueller Datensatz in der Tabellenmaske dargestellt, anderenfalls erscheint eine Meldung **‘Suchmuster nicht gefunden’**.

Wenn in einem Feld gesucht wird, das als erstes Feld zur Sortierung verwendet wird und das Suchmuster nicht mit einem Jokerzeichen (**‘\*’** oder **‘?’**) beginnt, dann wird ein verbesserter Suchalgorithmus (binäres Suchen) verwendet, der die vorsortierten Datensätze ausnützt. Dies kann die Geschwindigkeit enorm steigern.

### 11.2 Vorwärts/Rückwärts suchen

Zwei weitere Menüpunkte erlauben das Suchen nach dem nächsten und vorhergehenden Datensatz, in dem das Suchmuster auftaucht. **‘Tabelle – Suche vorwärts’** durchforstet die Datensätze vorwärts bis zum nächsten Datensatz, der auf das Suchmuster paßt und **‘Tabelle – Suche rückwärts’**, um zum vorhergehenden passenden Datensatz zu springen.

### 11.3 Suchmusterbeispiele

Hier ein paar Suchmusterbeispiele:

- ‘**Lassie**’ sucht nach Datensätzen, die die Zeichenkette ‘**Lassie**’ in einem der Suchfelder stehen haben.
- ‘**\*x\***’ sucht nach Datensätzen, die das Zeichen ‘**x**’ in einem der Suchfelder stehen haben.
- ‘**???**’ sucht nach Datensätzen, die genau drei Zeichen in einem der Suchfelder stehen haben, z.B. ein Datensatz mit dem Eintrag ‘**UFO**’.

## 12 Import und Export

Um Datensätze mit anderen Datenbanken zu teilen, bietet MUIbase eine Möglichkeit zum Import und Export von Datensätzen von und zu anderen Datenbanken an. Der Import und Export wird durch das Lesen und Schreiben von ASCII-Dateien bewerkstelligt. Aus diesem Grund müssen die zu importierenden Daten in einem besonderen Format vorliegen, das im nächsten Abschnitt beschrieben wird.

### 12.1 Dateiformat

Zum Importieren von Datensätzen in MUIbase müssen alle Datensätze in einer einzelnen ASCII-Datei vorliegen. Wenn Datensätze mehrerer Tabellen importiert werden sollen, so müssen mehrere Importdateien verwendet werden, d.h. eine für jede Tabelle.

Eine Importdatei besteht aus Zeilen und Spalten. Zeilen werden durch ein Datensatztrennzeichen und Spalten durch ein Feldtrennzeichen aufgeteilt. Die Trennzeichen können in den Import- und Exportfenstern festgelegt werden. Da Datensatzfelder selbst auch diese Trennzeichen enthalten können, ist es möglich, diese mit doppelten Anführungszeichen um alle Felder zu schützen.

Die Importdatei muß folgende Struktur haben:

- Die erste Zeile enthält die Feldnamen. Für jeden Namen muß ein Feld mit exakt dem gleichen Namen in der Tabelle vorhanden sein, in die die Daten importiert werden sollen. Falls ein Name auftaucht, der nicht in der Tabelle vorkommt, wird eine Fehlermeldung ausgegeben.
- Die folgenden Zeilen enthalten jeweils einen Datensatz. Da alle Felder als Zeichenketten vorliegen müssen, werden diese in den Datentyp des zugeordneten Feldes umgewandelt. Bei Feldern vom Typ Bool muß das Feld entweder NIL oder TRUE (größenunabhängig) enthalten, anderenfalls wird eine Fehlermeldung ausgegeben. Felder vom Typ Auswahl müssen die genauen Auswahltexte angegeben werden (größenabhängig). Bei Beziehungsfeldern muß die Datensatznummer, beginnend bei 1, angegeben werden. Für alle anderen Felder wird der Wert NIL verwendet, falls ein Wert nicht in den geforderten Typ umgewandelt werden konnte.
- Falls doppelte Anführungszeichen gewünscht werden, dann müssen alle Datensatzfelder einschließlich der Feldnamen in der ersten Zeile mit doppelten Anführungszeichen umschlossen werden.

### 12.2 Beispiel-Importdatei

Die folgende Beispiel-Importdatei verwendet \n und \t als Datensatz- und Feldtrennzeichen und doppelte Anführungszeichen um alle Felder. Die Datei kann dann in eine Tabelle mit folgenden Feldern importiert werden:

- Name (Zeichenkette)
- AnzKinder (Ganzzahl)
- Weiblich (Bool)
- Job (Auswahl)
- Anmerkungen (mehrzeiliger Text)



```
"Name" "AnzKinder" "Weiblich" "Job" "Anmerkungen"
"Janet Jackson" "???" "TRUE" "Musikerin" "Neueste CD: The velvet rope"
"Bernt Schiele" "???" "NIL" "Wissenschaftler" "Wissenschaftsgebiete:
Robotik, Autonomie und Bilderkennung"
"Gerhard" "0" "NIL" "Feinwerkzeugmechaniker" ""
```

## 12.3 Datensätze importieren

Um Datensätze in die aktive Tabelle zu importieren, wird der Menüpunkt **‘Tabelle – Importiere Datensätze...’** ausgewählt. Dies öffnet ein Fenster, das folgende Punkte enthält:

- Ein Zeichenkettenfeld zum Eingeben des Importdateinamens. Rechts neben dem Feld gibt es drei Knöpfe. Der erste dient der Auswahl eines Dateinamens aus einem Dateiauswahlfenster. Der Knopf **‘A’** startet den externen Anzeiger, um sich die angegebene Datei anzusehen und der Knopf **‘E’** startet den externen Editor, um den Dateiinhalt verändern zu können.
- Zwei Zeichenkettenfelder zum Eingeben der Datensatz- und Feldtrennzeichen. Man kann ein einzelnes Zeichen oder einen erweiterten Code durch die Eingabe von `\n`, `\t`, `\f`, `\???` (Ok-talzahl) oder `\x??` (Hexadezimalzahl) eingeben.
- Ein Feld **‘In Anführungszeichen’**, das eingeschaltet werden kann, um anzugeben, daß die Felder mit doppelten Anführungszeichen umschlossen werden sollen.
- Zwei Knöpfe **‘Importieren’** und **‘Abbrechen’**, um das Fenster verlassen zu können.

Wenn der Knopf **‘Importieren’** gedrückt wird, beginnt MUIbase, die angegebene Datei einzuladen und alle gefundenen Datensätze zu importieren. Falls keine Fehler auftraten, fragt MUIbase nach, ob die importierten Datensätze wirklich zur Tabelle hinzugefügt werden sollen. An dieser Stelle läßt sich der Vorgang noch abbrechen.

Tritt während des Lesens der Importdatei ein Fehler auf, dann wird eine Fehlermeldung angezeigt.

Falls eine umfangreichere Import-Funktion benötigt wird, dann wird empfohlen, eine eigene Importfunktion als MUIbase-Programm zu schreiben.

## 12.4 Datensätze exportieren

Um aus der aktiven Tabelle Datensätze zu exportieren, wählt man den Menüpunkt **‘Tabelle – Exportiere Datensätze...’**. Dies öffnet ein Fenster mit folgender Struktur:

- Ein Zeichenkettenfeld zum Eingeben des Exportdateinamens. Rechts neben dem Feld gibt es einen Knopf, der der Auswahl eines Dateinamens aus einem Dateiauswahlfenster dient.
- Zwei Zeichenkettenfelder zum Eingeben der Datensatz- und Feldtrennzeichen. Man kann ein einzelnes Zeichen oder einen erweiterten Code durch die Eingabe von `\n`, `\t`, `\f`, `\???` (Ok-talzahl) oder `\x??` (Hexadezimalzahl) eingeben.
- Ein Feld **‘In Anführungszeichen’**, das eingeschaltet werden kann, um anzugeben, daß die Felder mit doppelten Anführungszeichen umschlossen werden sollen.

- Ein Feld **‘Filter’**. Wenn eingeschaltet, dann werden nur die Datensätze exportiert, die dem gerade installierten Filter genügen.
- Zwei Knöpfe **‘Exportieren’** und **‘Abbrechen’**, um das Fenster verlassen zu können.

Nach dem Druck auf den Knopf **‘Exportieren’** öffnet MUIbase die angegebene Datei und schreibt die Datensätze einschließlich der Kopfzeile mit den Feldnamen hinein. Die Exportfunktion schreibt grundsätzlich alle Felder einer Tabelle in die Exportdatei.

Für eine Exportfunktion mit mehr Möglichkeiten kann man entweder den Abfrageeditor von MUIbase (siehe Kapitel 13 [Data retrieval], Seite 57) verwenden oder eigene Exportfunktionen als MUIbase-Programm schreiben.

## 13 Datenabfragen

Zur Datenabfrage bietet MUIbase zwei Möglichkeiten an: Die Programmierung und den Abfrageeditor.

Die Programmierung ermöglicht die Einrichtung von Knöpfen in der Tabellenansicht, die auf Druck Programmfunktionen aufrufen. Die Verwendung dieser Besonderheit wird im Kapitel zum Struktureditor (siehe Kapitel 14 [Structure editor], Seite 62) und im Kapitel über die Programmierung von MUIbase (siehe Kapitel 15 [Programming MUIbase], Seite 75) beschrieben.

Dieses Kapitel beschreibt die Verwendung des Abfrageeditors, ein Fenster zum Eingeben von Abfragen und Anzeigen der Ausgabe in einer verschiebbaren Listenansicht.

### 13.1 Select-from-where Abfragen

MUIbase bietet eine Select-from-where Abfrage an, die denen in SQL-Datenbanksystemen ähnelt. Die Abfrage erlaubt es, Datensatzinhalte aus einer oder mehreren Tabellen aufzulisten. Nur die Datensätze, die dem gesamten Ausdruck genügen, werden in die Ausgabe einbezogen. Die (unvollständige) Syntax einer Select-from-where Abfrage ist

```
SELECT exprlist FROM tablelist [WHERE test-expr]  
[ORDER BY orderlist]
```

wobei *exprlist* eine mit Kommas aneinandergereihte Liste von Ausdrücken ist, die ausgegeben werden sollen (normalerweise die Feldnamen) oder ein einfacher Stern \*, der alle Felder der Tabelle einschließt. *tablelist* ist eine mit Kommas aneinandergereihte Liste von Tabellen, deren Datensätze untersucht werden sollen. *test-expr* ist der Ausdruck, der für jede Menge von Datensätzen, die in die Ausgabe eingeschlossen werden sollen, ausgewertet wird und *orderlist* ist eine mit Kommas aneinandergereihte Liste von Feldern, die die Sortierung der Ausgabeliste festlegen. Zu beachten ist, daß die Felder WHERE und ORDER BY optional sind, kenntlich gemacht durch eckige Klammern [].

Zum Beispiel listet die Abfrage

```
SELECT * FROM table
```

die Feldinhalte aller Datensätze in der gegebenen Tabelle auf.

```
SELECT attr1 FROM table WHERE (LIKE attr2 "*Madonna*")
```

listet die Inhalte des Feldes *attr1* aus allen Datensätzen der Tabelle *table* auf, deren Inhalte des Feldes *attr2* das Wort 'Madonna' beinhaltet.

Weitere Informationen zur Select-from-where Abfrage einschließlich ihrer vollständigen Syntax, siehe Kapitel 15 [Programming MUIbase], Seite 75 und für weitere Beispiele siehe Abschnitt 13.4 [Query examples], Seite 60.

## 13.2 Abfrageeditor

Zum Eingeben und Ausführen von Abfragen öffnet man den Abfrageeditor über den Menüpunkt **‘Programm – Abfragen...’**. Der Abfrageeditor kann mehrere Abfragen verwalten, aber es kann jedoch immer nur eine Abfrage zu einem Zeitpunkt ablaufen. Das Abfrageeditor-Fenster enthält folgende Elemente:

- ein Zeichenkettenfeld mit einem anhängenden Popup-Knopf. Das änderbare Zeichenkettenfeld zeigt den Namen der aktuellen Abfrage an. Über den Popup-Knopf erscheint eine Liste mit weiteren Abfragenamen und verschiedenen Knöpfen. Man kann eine der aufgelisteten Abfragen auswählen, um diese zur aktuellen zu machen; den Knopf **‘Neu’** drücken, um eine neue Abfrage zu beginnen; den Knopf **‘Duplizieren’** drücken, um eine Kopie der gewählten Abfrage zu erhalten; den Knopf **‘Sortieren’** drücken, um die Liste der Abfragen zu sortieren oder den Knopf **‘Löschen’** drücken, um die ausgewählte Abfrage zu löschen. Um das Popup-Fenster wieder zu schließen, ohne etwas zu ändern, drückt man erneut auf den Popup-Knopf.
- ein Knopf **‘Ausführen...’**, der das Abfrageprogramm kompiliert, ausführt und das Ergebnis in der Ausgabe-Listenansicht ausgibt.
- ein Knopf **‘Drucken...’**, der ein Fenster (siehe Abschnitt 13.3 [Printing queries], Seite 58) öffnet, um Ergebnisse auszudrucken.
- zwei Knöpfe **‘Laden...’** und **‘Speichern...’** zum Laden und Speichern des aktuellen Abfrageprogramms. Wird eine Abfrage gespeichert, dann wird der Text im ASCII-Format geschrieben und die erste Zeile enthält dann den Namen der Abfrage als Kommentar. Die Ladefunktion erwartet das gleiche Format, wenn eine Datei gelesen wird. Es ist auch möglich, Abfragen mit den Menüpunkten des Kontextmenüs vom Abfrageprogramm-Editorfeld zu laden und zu speichern. Allerdings speichern die Menüpunkte nur den Programmtext, schließen aber den Namen der Abfrage nicht mit ein.
- ein Editorfeld zum Eingeben des Abfrageprogramms. Hier wird normalerweise eine Select-from-where Abfrage eingeben. Es ist jedoch auch möglich, jeden Ausdruck der MUIbase-Programmiersprache zu verwenden. Dies kann nützlich sein, wenn einfache Berechnungen durchgeführt oder einige Felder einer Tabelle mit einem einfachen Programm aktualisiert werden sollen. Es ist zu beachten, daß MUIbase den Programmausdruck mit einem Paar Klammern umschließt, d.h. die äußeren Klammern können weggelassen werden.
- eine Listenansicht, die die Ausgabe nach dem Ausführen der aktuellen Abfrage anzeigt. Die Ausgabe ist in Zeilen und Spalten aufgeteilt. Die Titelzeile trägt die Namen der Select-from-where Abfrage (normalerweise die Feldnamen). Die anderen Zeilen enthalten den Inhalt des Abfrageergebnisses, pro Zeile einen Datensatz. Jeder Feldeintrag wird in einer eigenen Spalte dargestellt. Wird auf einen Eintrag in der Liste doppelgeklickt und dieser Eintrag wurde von einem Feld der Typen Zeichenkette oder Beziehung erzeugt, dann wird der ausgewählte Datensatz in der zugehörigen Tabellenansicht angezeigt. Dies ist eine einfache Möglichkeit, zu einem bestimmten Datensatz in einer Tabellenansicht zu springen.

Das Abfragefenster ist ein Nicht-modales Fenster. Das bedeutet, daß der Abfrageeditor geöffnet bleiben und dennoch mit der Anwendung weitergearbeitet werden kann. Der Abfrageeditor läßt sich jederzeit durch das Schließsymbol in der Fenster-Titelzeile schließen.

## 13.3 Abfragen ausdrucken

Wenn eine Abfrage durchgeführt wurde, dann kann das Ergebnis über den Knopf **‘Drucken’** in eine Datei oder auf den Drucker ausgedruckt werden. Dies öffnet ein Druckfenster mit den folgenden Elementen:

- ein Bereich **‘Begrenzer’**, in dem festgelegt wird, wie die Spalten voneinander getrennt werden sollen. **‘Zwischenräume’** füllt die Felder mit Leerzeichen auf. Dabei wird rechts oder links aufgefüllt, je nach Typ des Feldes (Zahlen werden links aufgefüllt, Texte rechts). **‘Tabulatoren’** fügt genau ein Tabulator-Zeichen zwischen den Spalten ein. Dies kann sinnvoll sein, wenn das Druckfenster zum Exportieren von Datensätzen verwendet werden soll (siehe unten).
- ein Bereich **‘Zeichensatz’**, in dem festgelegt wird, welche Druckqualität zum Drucken verwendet wird. **‘NLQ’** steht für **‘near letter quality’** (Beinahe-Briefqualität), das eine bessere Ausgabe als bei **‘Entwurf’** erzeugt.
- ein Bereich **‘Größe’**, in dem die Zeichengröße definiert wird. **‘Pica’** druckt mit großer Schrift (10 cpi), **‘Elite’** in mittelhocher Schrift (12 cpi) und **‘verdichtet’** in kleiner Schrift (17 cpi).
- ein Zeichenkettenfeld **‘Initialisierungssequenz’**, in der eine Zeichenkette zum Zurücksetzen des Druckers eingegeben werden kann. Der Inhalt des Feldes wird direkt zum Drucker geschickt. Zum Beispiel kann man **‘\33c’** als Rücksetzsequenz angeben, welche den Drucker zurücksetzt.
- ein Feld **‘Einzug’**, in dem die Anzahl Leerschritte eingestellt werden kann, um die jede Zeile eingerückt werden soll.
- ein Feld **‘Titelzeile?’**, das die Feldnamen in der ersten Zeile ausgibt, wenn es eingeschaltet ist.
- ein Feld **‘Steuerzeichen?’**. Wenn es nicht eingeschaltet ist, dann wird die Ausgabe aller Steuerzeichen unterdrückt, d.h. die Einstellungen bei **‘Zeichensatz’** und **‘Größe’** werden ignoriert, ebenso wird der Inhalt der Zeichenkette **‘Initialisierungssequenz’** nicht ausgegeben. Die Unterdrückung der Steuerzeichen ist sinnvoll, wenn eine ASCII-Datei erzeugt werden soll, z.B. zum Exportieren von Datensätzen.
- ein Feld **‘Anführungszeichen?’**, welches alle Felder mit Anführungszeichen umschließt, wenn es eingeschaltet ist.
- im Bereich **‘Nach dem Druck’** läßt sich einstellen, was nach dem Druck geschehen soll. **‘Seitenumbruch’** druckt ein Seitenumbruchzeichen **\f**, **‘Zeilenumbrüche’** eine Anzahl von Zeilenumbruchzeichen **\n**. Die Anzahl der Zeilenumbrüche läßt sich im Feld rechts daneben eingeben. **‘Nichts’** druckt nichts auf dem Drucker aus.
- ein Zeichenkettenfeld **‘Ausgabe’**, das mit einem Popup-Knopf versehen ist. Der Popup-Knopf kann benutzt werden, um einen Dateinamen mit dem Dateiauswahlfenster auszuwählen, oder man gibt den Dateinamen direkt im Zeichenkettenfeld ein. Für Druckausgabe sollte **‘PRT:’** im Zeichenkettenfeld eingegeben werden, zur Ausgabe in ein Fenster **‘CON:////MUIbase output/CLOSE/WAIT’**.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’**, um das Druckfenster verlassen zu können.

Wenn alle Einstellungen gemacht sind, drückt man auf den Knopf **‘Ok’**, um den Druckauftrag zu starten.

Das Druckfenster kann auch zum Exportieren von Datensätzen in eine ASCII-Datei verwendet werden. Dazu stellt man **‘Tabulatoren’** im Bereich **‘Begrenzer’** ein, setzt die Anzahl der Leerzeichen für den Einzug auf 0, schaltet **‘Titelzeile?’** ein, schaltet **‘Steuerzeichen?’** aus, um die Zeichensatz-, Größen- und Initialisierungseinstellungen zu unterdrücken, schaltet ggf. **‘Anführungszeichen?’** ein, wenn die Feldinhalte mit Anführungszeichen umschlossen werden sollen, schaltet **‘Nichts’** im Bereich **‘Nach dem Druck’** ein und gibt den Namen der Ausgabedatei im Feld **‘Ausgabe’** an. Die Benutzung des Abfrageeditors mit dem Druckfenster zum Exportieren von Datensätzen kann leistungsfähiger sein als der Im-/Export von MUIbase (siehe Kapitel 12 [Import and Export], Seite 54), da im Abfrageeditor jede Abfrage eingegeben werden kann, wohingegen das Exportfenster nur eine feste Abfrage verwendet.

## 13.4 Abfragebeispiele

Um einen Eindruck von der Leistungsfähigkeit der Select-from-where Abfragen zu bekommen, im folgenden einige Beispiele<sup>1</sup>.

Angenommen, es gibt zwei Tabellen 'Person' und 'Dog' (Hund). 'Person' besitzt ein Zeichenkettenfeld 'Name', ein Ganzzahlfeld 'Age' (Alter) und zwei Beziehungsfelder 'Father' (Vater) und 'Mother' (Mutter), die auf Datensätze in der Tabelle 'Person' für den Vater und die Mutter verweisen. Die Tabelle enthält folgende Datensätze:

	Name	Age	Father	Mother
p1:	Steffen	26	p2	p3
p2:	Dieter	58	NIL	NIL
p3:	Marlies	56	NIL	NIL
p4:	Henning	57	NIL	NIL

'Dog' (Hund) besitzt ein Zeichenkettenfeld 'Name', ein Auswahlfeld 'Color' (Farbe) und ein Beziehungsfeld 'Owner' (Besitzer), das auf den Besitzer in der Tabelle 'Person' verweist. Die Tabelle enthält folgende Datensätze:

	Name	Color	Owner
d1:	Boy	white	p3
d2:	Streuner	grey	NIL

Mit diesen Daten lassen sich folgende Select-from-where Beispielabfragen durchführen:

```
SELECT * FROM Person
```

liefert:

Name	Age	Father	Mother
Steffen	26	Dieter	Marlies
Dieter	58		
Marlies	56		
Henning	57		

(Für die Beziehungsfelder wird das Feld 'Name' des referenzierten Datensatzes ausgegeben.)

<sup>1</sup> Anm.d.Übersetzers: Im folgenden werden auch die original Beispiele verwendet, welche englische Bezeichnungen enthalten. Daher wird bei den Beschreibungen auch die deutsche Bezeichnung in Klammern dahintergesetzt, sofern sie nicht schon im Englischen so heißt, wie im Deutschen.

```
SELECT Name "Child", Age,
       Father.Name "Father", Father.Age "Age",
       Mother.Name "Mother", Mother.Age "Age"
FROM Person WHERE (AND Father Mother)
```

liefert:

Child	Age	Father	Age	Mother	Age
Steffen	26	Dieter	58	Marlies	56

```
SELECT Name, Color,
       (IF Owner Owner.Name "No owner") "Owner"
FROM Dogs
```

liefert:

Name	Color	Owner
Boy	white	Marlies
Streuner	grey	No owner

```
SELECT a.Name, a.Age, b.Name, b.Age FROM Person a, Person b
WHERE (> a.Age b.Age)
```

liefert:

a.Name	a.Age	b.Name	b.Age
Dieter	58	Steffen	26
Marlies	56	Steffen	26
Henning	57	Steffen	26
Dieter	58	Marlies	56
Henning	57	Marlies	56
Dieter	58	Henning	57

## 14 Struktureditor

MUIbase besitzt zwei verschiedene Arbeitsmodi: den Datensatzbearbeitungsmodus, in dem Datensätze bearbeitet und durchforstet werden können und den Strukturbearbeitungsmodus, in dem die Struktur, d.h. Tabellen, Felder und Erscheinungsbild, definiert wird. Dieses Kapitel beschreibt den Struktureditor und erklärt, wie die Struktur eines Projekts verwaltet wird.

Um vom Datensatzbearbeitungsmodus in den Strukturbearbeitungsmodus zu wechseln, wird der Menüpunkt **‘Struktureditor...’** im Menü **‘Projekt’** ausgewählt. Dies schließt alle Fenster und öffnet das Struktureditor-Fenster. Um zum Datensatzbearbeitungsmodus zurückzukehren, kann der Menüpunkt **‘Projekt – Struktureditor verlassen’** ausgewählt oder einfach das Struktureditor-Fenster über den Schließknopf der Fenstertitelzeile verlassen werden.

Das Struktureditor-Fenster ist aufgeteilt in drei Teile: links oben ist der Bereich **‘Tabellen’** zum Verwalten der Tabellen eines Projekts, links unten der Bereich **‘Felder’** für die Felder einer Tabelle und rechts der Bereich **‘Anzeige’** zum Verwalten der grafischen Elemente des Projekts.

### 14.1 Tabellenverwaltung

Im Bereich **‘Tabellen’** des Struktureditors werden Tabellen erstellt, geändert, gelöscht und sortiert.

#### 14.1.1 Tabellen erstellen

Um eine neue Tabelle zu erstellen, wird der Knopf **‘Neu’** im Bereich **‘Tabellen’** gedrückt. Dies öffnet das Fenster **‘Neue Tabelle’** mit

- einem Zeichenkettenfeld für den Namen der Tabelle. Jede Tabelle muß einen eindeutigen Namen haben, der mit einem Kleinbuchstaben beginnen muß und weitere Zeichen, Ziffern und Unterstrich-Zeichen enthält. Nicht-ASCII-Zeichen wie deutsche Umlaute sind nicht zulässig. Anzumerken ist, daß die Benutzeroberfläche zur Tabelle trotzdem Zeichenketten mit Nicht-ASCII-Zeichen darstellen kann.
- einem Bereich **‘Anzahl der Datensätze’**, in dem festgelegt wird, wie viele Datensätze die Tabelle halten darf. **‘unbegrenzt’** bedeutet, daß die Tabelle jede Anzahl von Datensätze halten kann und **‘genau ein’** kann nur einen einzigen Datensatz halten. Letzteres ist manchmal nützlich, um das Projekt zu steuern (siehe Abschnitt 5.2 [Tables], Seite 20).
- einem Bereich **‘Auslösefunktionen’**, in dem die Namen von zwei Funktionen eingegeben werden können. Im Feld **‘Neu’** gibt man die Funktion an, die immer dann aufgerufen wird, wenn der Benutzer einen neuen Datensatz anlegt und im Feld **‘Löschen’** die Funktion bei jedem Löschen eines Datensatzes. Dazu können die Popup-Knöpfe rechts davon verwendet werden, um aus einer Liste aller Namen eine Funktion auszuwählen. Wird ein Feld leer gelassen, dann werden Vorgabefunktionen ausgeführt (Datensätze werden automatisch erzeugt und Datensätze werden nach einer eventuellen Sicherheitsabfrage gelöscht). Mehr über die Anwendung von Auslösefunktionen, einschließlich der Argumente, die ihnen übergeben werden, siehe Kapitel 15 [Programming MUIbase], Seite 75.
- zwei Knöpfen **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters.



Wurden alle Einstellungen getätigt, wird **‘Ok’** gedrückt, um die neue Tabelle zu erzeugen. Falls ein Fehler gemacht wurde, z.B. Eingabe eines falschen Namens, dann weist ein Hinweisenfenster auf den Fehler hin. Tritt kein Fehler auf, dann schließt das Fenster **‘Neue Tabelle’** und die neue Tabelle erscheint in der Tabellenliste des Struktureditors.

### 14.1.2 Tabellen ändern

Nachdem eine Tabelle erzeugt wurde, läßt sie sich nachträglich ändern. Dazu klickt man doppelt auf den Namen der Tabelle und das Fenster **‘Tabelle ändern’** erscheint. Dieses Fenster gleicht dem Fenster, das beim Erstellen der Tabelle verwendet wird (siehe Abschnitt 14.1.1 [Creating tables], Seite 62) und erlaubt Änderungen in jedem Feld durch neue Eingaben.

Wurden alle Einstellungen getätigt, wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

Anzumerken ist, daß sich die Anzahl der Datensätze nicht von **‘unbegrenzt’** auf **‘genau ein’** ändern läßt, wenn die Tabelle schon mehr als einen Datensatz beinhaltet.

### 14.1.3 Tabellen löschen

Um eine Tabelle zu löschen, klickt man auf den Namen der Tabelle in der Tabellenliste des Struktureditors und drückt dann den Knopf **‘Löschen’** unterhalb der Liste. Bevor die Tabelle tatsächlich gelöscht wird, fragt eine Sicherheitsabfrage um die Bestätigung. Wenn die Sicherheitsabfrage mit dem Knopf **‘Löschen’** bestätigt wird, dann wird die Tabelle gelöscht.

Ein Problem taucht auf, wenn die Tabelle irgendwo im Programm zum Projekt verwendet wird. In diesem Fall kann die Tabelle nicht einfach gelöscht werden, sondern alle Beziehungen zur Tabelle müssen zuerst entfernt werden. Wird die zu löschende Tabelle im Programm zum Projekt verwendet, dann erscheint der Programmeditor und zeigt das erste Auftreten der Tabelle an. Nun muß man das Programm so ändern, daß keine Beziehungen zur Tabelle mehr im Programm verbleiben. Nach dem Entfernen einer Beziehung kann man zur nächsten springen, indem man den Knopf **‘Kompilieren’** drückt. An jeder Stelle kann man die gesamte Operation durch einen Druck auf den Knopf **‘Rückgängig machen’** abbrechen und den Programmeditor schließen.

### 14.1.4 Tabellen sortieren

Zum Sortieren der Tabellen im Bereich **‘Tabellen’** des Struktureditors hat man zwei Möglichkeiten: mit Verschieben & Loslassen die Reihenfolge per Hand einstellen oder den Knopf **‘Sortieren’** unterhalb der Listenansicht drücken, der die Tabelle automatisch sortiert.

## 14.2 Felderverwaltung

Im Bereich **‘Felder’** des Struktureditors können Felder der aktiven Tabelle aus dem Bereich **‘Tabellen’** erzeugt, kopiert, verändert, gelöscht und sortiert werden.

### 14.2.1 Felder erstellen

Um ein neues Feld für die aktive Tabelle zu erstellen, drückt man den Knopf ‘**Neu**’ im Bereich ‘**Felder**’. Dies öffnet dann das Fenster ‘**Neues Feld**’ mit

- einem Zeichenkettenfeld zur Eingabe des Namens des Feldes. Jedes Feld einer Tabelle muß einen eindeutigen Namen haben, der mit einem Großbuchstaben beginnt, gefolgt von weiteren Buchstaben, Ziffern und Unterstrich-Zeichen. Nicht-ASCII-Zeichen wie deutsche Umlaute sind nicht zulässig. Anzumerken ist, daß die Benutzeroberfläche trotzdem Zeichenketten mit Nicht-ASCII-Zeichen darstellen kann.
- einem Feld ‘**Typ**’, in dem der Typ des Feldes festgelegt wird. Mehr Informationen zu den Feldtypen, siehe Abschnitt 5.5 [Attribute types], Seite 21.
- einem Bereich unterhalb des Feldes ‘**Typ**’ zum Festlegen von typabhängigen Einstellungen. Mehr über diesen Bereich, siehe Typabhängige Einstellungen.
- einem Feld ‘**Auslösefunktion**’, in dem der Name einer Funktion eingegeben werden kann, die immer dann aufgerufen wird, wenn der Inhalt des Feldes im Datensatz sich ändert. Dazu kann der Popup-Knopf rechts davon verwendet werden, um aus einer Liste aller Namen eine Funktion auszuwählen. Wird das Feld leer gelassen, dann wird eine Vorgabefunktion ausgeführt, die einfach den eingegebenen Wert im Feld speichert. Mehr über die Anwendung von Auslösefunktionen, einschließlich der Argumente, die ihnen übergeben werden, siehe Kapitel 15 [Programming MUIbase], Seite 75.
- zwei Knöpfen ‘**Ok**’ und ‘**Abbrechen**’ zum Verlassen des Fensters.

Wurden alle Einstellungen getätigt, wird ‘**Ok**’ gedrückt, um das neue Feld zu erzeugen. Falls ein Fehler gemacht wurde, z.B. Eingabe eines falschen Namens, dann weist ein Hinweisfenster auf den Fehler hin. Tritt kein Fehler auf, dann schließt das Fenster ‘**Neues Feld**’ und das neue Feld erscheint in der Felderliste des Struktureditors.

### 14.2.2 Typabhängige Einstellungen

Im typabhängigen Bereich können folgende Einstellungen getätigt werden:

- Für Felder vom Typ Zeichenkette gibt es
  - ein Ganzzahlfeld ‘**max. Länge**’ für die maximale Länge der Zeichenkette für dieses Feld.
  - ein Zeichenkettenfeld ‘**Vorgabewert**’ zur Angabe eines Wertes zum Vorbelegen des Feldes. Jede Zeichenkette bis zur festgelegten maximalen Länge kann hier eingegeben werden.
- Für Felder vom Typ Ganzzahl, Fließkommazahl, Datum und Zeit bietet der typabhängige Bereich
  - einen Bereich ‘**Vorgabewert**’, in dem ein Wert zum Vorbelegen des Feldes festgelegt wird. Es kann zwischen ‘**NIL**’ und ‘**anderer**’ gewählt werden. Wurde ‘**anderer**’ gewählt, dann gibt man den Vorgabewert im Zeichenkettenfeld rechts neben dem ‘**anderer**’ an.
  - ein Zeichenkettenfeld ‘**NIL-Text**’, in dem eine Zeichenkette eingegeben werden kann, die angezeigt wird, wenn das Feld den Wert NIL beinhaltet.
- Für boolesche Felder enthält der typabhängige Bereich einen Bereich ‘**Vorgabewert**’, in dem als Vorgabewert zwischen ‘**WAHR**’ und ‘**FALSCH**’ gewählt werden kann.
- Der typabhängige Bereich für Auswahlfelder bietet
  - einen Knopf ‘**Bearbeite Auswahltexte**’ zum Öffnen des Fensters ‘**Bearbeite Auswahltexte**’, in dem die Auswahltexte für das Auswahlfeld eingegeben werden können (siehe Abschnitt 14.2.3 [Label editor], Seite 65).

- ein Auswahlfeld ‘**Vorgabewert**’ zur Festlegung des Wertes zum Initialisieren des Feldes.
- Für Beziehungsfelder enthält der typabhängige Bereich
  - eine Listenansicht, die alle Tabellen anzeigt, zu welcher Tabelle eine Beziehung hergestellt werden soll. Dazu klickt man auf die Tabelle, zu der eine Beziehung angelegt werden soll.
  - ein Feld ‘**Filtern?**’. Wenn ausgewählt, ist der Referenzfilter des Feldes aktiviert. Siehe Abschnitt 9.2 [Reference filter], Seite 47 für mehr Informationen über diese Eigenschaft.

Beziehungsfelder haben immer den Wert NIL als Vorgabewert.

- Der typabhängige Bereich für virtuelle Felder enthält ein Zeichenkettenfeld ‘**Berechne**’ für den Namen einer Funktion, die beim Berechnen eines Feldwertes ausgeführt wird. Der angehängte Popup-Knopf kann zur Auswahl eines Namens aus einer Liste von Funktionsnamen verwendet werden.
- Mehrzeilige Textfelder und Knöpfe besitzen keine typabhängigen Einstellungen. Der Vorgabewert für mehrzeilige Texte ist eine leere Zeichenkette.

### 14.2.3 Auswahltexteditor

Wenn eine Liste von Auswahltexten festgelegt werden soll, z.B. Liste von Auswahltexten für ein Auswahlfeld, dann tritt der Auswahltexteditor in Erscheinung. Der Auswahltexteditor ist ein Fenster mit

- einer Listenansicht, die die aktuelle Liste der Auswahltexte anzeigt. Man kann auf einen Auswahltext klicken, um ihn zum aktiven zu machen. Der aktive Auswahltext wird auch im Zeichenkettenfeld unterhalb der Listenansicht angezeigt. Mit Verschieben & Loslassen lassen sich die Auswahltexte anordnen.
- einem Zeichenkettenfeld ‘**Auswahltext**’, das den aktiven Auswahltext anzeigt und Änderungen zulässt. Die Änderungen werden nur dann durchgeführt, wenn die **Eingabe**-Taste gedrückt wird. Gibt es keinen aktiven Auswahltext, dann fügt **Eingabe** neue Auswahltexte der Liste hinzu.
- einem Knopf ‘**Neu**’, der den aktuellen Auswahltext inaktiviert, um neue Auswahltexte im Zeichenkettenfeld ‘**Auswahltext**’ eingeben zu können.
- einem Knopf ‘**Entfernen**’, der den aktiven Auswahltext aus der Liste entfernt.
- einem Knopf ‘**Sortieren**’, um die Liste der Auswahltexte alphabetisch sortieren zu können.
- zwei Knöpfen ‘**Ok**’ und ‘**Abbrechen**’, um den Auswahltexteditor verlassen zu können.

Wenn alle Auswahltextte eingegeben oder alle Änderungen durchgeführt wurden, drückt man ‘**Ok**’, um das Fenster zu verlassen.

### 14.2.4 Felder kopieren

Falls viele gleichartige Felder benötigt werden sollen, dann kann ein Feld kopiert werden. Einfach das gewünschte Feld auswählen und den Knopf ‘**Kopieren**’ unterhalb der Feldliste drücken. Dies öffnet das ‘**Feld kopieren**’-Fenster, in dem die Einstellungen für dieses Feld angezeigt werden. Nach dem Ändern einiger Felder, wie z.B. des Namens, drückt man ‘**Ok**’, um eine Kopie des Feldes zu erzeugen.

### 14.2.5 Felder ändern

Nachdem ein neues Feld erzeugt wurde, kann man nachträglich einige Einstellungen an ihm verändern. Dazu doppelklickt man auf den Namen des Feldes und das Fenster ‘**Attribut ändern**’ erscheint. Dieses Fenster ähnelt dem beim Erstellen von Feldern (siehe Abschnitt 14.2.1 [Creating attributes], Seite 64) und erlaubt das Ändern in einigen Feldern. Felder, die nicht verändert werden dürfen, z.B. Feldtyp, werden verdeckt angezeigt.

Folgende Hinweise sollten beachtet werden, wenn Felder verändert werden:

- Der Feldtyp kann nicht verändert werden. Falls jemals der Typ geändert werden sollte, ist es am günstigsten, ein neues Feld mit dem gewünschten Typ zu erzeugen und die Datensatzinhalte des alten Feldes mit einem einfachen MUIbase-Programm im Abfrageeditor (siehe Abschnitt 13.2 [Query editor], Seite 58) in das neue zu kopieren.
- Wenn der Vorgabewert eines Feldes geändert wird, dann erhalten nur neu erzeugte Datensätze diesen Wert.
- Bei Auswahlfeldern sollte man beim Ändern von Auswahltexten vorsichtig sein. Die Auswahltexte werden nur zum Anzeigen des Auswahlfeldinhaltes verwendet, intern werden aber Nummern gespeichert, die als Index für die Auswahltextliste dienen. Wird also die Reihenfolge der Auswahltexte geändert, dann ändert sich nicht die interne Nummer, sondern nur der Text, der dafür angezeigt wird! Daher sollte man die Reihenfolge der Auswahltexte nicht verändern, nachdem ein Auswahlfeld erzeugt wurde. Hinzufügen von neuen Auswahltexten macht jedoch keine Probleme. Für eine flexible Möglichkeit eines Auswahl-ähnlichen Feldes, in dem auch die Reihenfolge der Auswahltexte geändert werden kann, ist die Verwendung eines Zeichenkettenfeldes zusammen mit der Listenansicht-Popup-Eigenschaft (siehe Abschnitt 14.3.3 [Attribute object editor], Seite 69).
- Die referenzierte Tabelle eines Beziehungsfeldes kann nicht verändert werden.

Wurden alle Änderungen durchgeführt, dann wird ‘Ok’ gedrückt, um das Fenster zu verlassen.

### 14.2.6 Felder löschen

Um ein Feld zu löschen, wird auf dessen Name in der Felderliste des Struktureditors geklickt und der Knopf ‘**Löschen**’ unter der Liste gedrückt. Bevor das Feld tatsächlich gelöscht wird, fragt ein Sicherheitsfenster um Erlaubnis. Wird dieses Fenster über den Knopf ‘**Löschen**’ bestätigt, dann wird das Feld gelöscht.

Ein Problem tritt jedoch auf, wenn das Feld an irgendeiner Stelle im Programm zum Projekt verwendet wird. In diesem Fall kann das Feld nicht einfach gelöscht werden, sondern alle Verweise auf dieses müssen aus dem Programm entfernt werden. Wird das zu löschende Feld im Programm verwendet, dann erscheint der Programmeditor und zeigt auf das erste Auftreten dieses Feldes. Das Programm sollte nun so geändert werden, daß keine Verweise auf das Feld mehr im Programm verbleiben. Nachdem ein Verweis entfernt wurde, kann zum nächsten gesprungen werden, in dem der Knopf ‘**Kompilieren**’ gedrückt wird. Zu jedem Zeitpunkt kann die gesamte Operation über den Knopf ‘**Rückgängig machen**’ und dem Schließen des Fensters abgebrochen werden.

### 14.2.7 Felder sortieren

Zum Sortieren der Felder im Bereich ‘Felder’ des Struktureditors gibt es zwei Möglichkeiten. Zum einen läßt sich dies per Verschieben & Loslassen von einzelnen Feldern oder durch den Knopf ‘Sortieren’ unterhalb der Listenansicht zum alphabetischen Sortieren erledigen.

## 14.3 Anzeigeverwaltung

Im Bereich ‘Anzeige’ des Struktureditors wird festgelegt, wie die Datenbankelemente in der Benutzeroberfläche angeordnet werden. Der Bereich beinhaltet ein Auswahlfeld, eine Listenansicht und einige Knöpfe.

### 14.3.1 Anzeigebereich

Der Anzeigebereich enthält folgende Elemente:

- ein Auswahlelement mit zwei Einstellungen ‘Tabellenschema’ und ‘Hauptfenster’. Im ‘Tabellenschema’ wird festgelegt, wie die Felder der aktiven Tabelle auf der Benutzeroberfläche angeordnet werden. Im ‘Hauptfenster’ wird spezifiziert, wie Tabellen angeordnet werden.
- eine Listenansicht, die die aktuelle Anordnung der Benutzeroberfläche anzeigt. Die Liste ist als Baum organisiert. Elemente mit einem links angeordneten Pfeil sind gebundene Oberflächenobjekte und können durch Doppelklicken auf das Pfeilsymbol geöffnet und geschlossen werden. Ein Doppelklick auf das Element selbst öffnet ein Fenster zum Verändern von dessen Einstellungen. Alle Benutzeroberflächenobjekte, die das gleiche übergeordnete Objekt besitzen, werden in der gleichen Weise ausgerichtet (entweder horizontal oder vertikal). Das übergeordnete Benutzeroberflächenobjekt bestimmt, wie das Layout dargestellt wird: Tabellen, Panels und Fenster ordnen ihre Elemente vertikal an, Gruppen ordnen sie nach den Einstellungen im Gruppeneitor an (siehe Abschnitt 14.3.8 [Group editor], Seite 73).
- ein Knopf ‘Panel’ zum Hinzufügen eines Panels zur Tabelle. Siehe Abschnitt 14.3.2 [Panel editor], Seite 68 für mehr Informationen über das Einrichten eines Panels.
- ein Knopf ‘Hinzufügen’ zum Hinzufügen der aktuellen Tabelle oder des aktuellen Feldes (abhängig vom Status des Auswahlfeldes der Anzeige) zur Anzeigelistenansicht. Normalerweise werden beim Erzeugen Tabellen und Felder automatisch in der Anzeigelistenansicht hinzugefügt.
- ein Knopf ‘Entfernen’ zum Entfernen des aktiven Elements aus der Anzeigelistenansicht. Wird eine Tabelle von der Anzeigelistenansicht entfernt, dann wird die vollständige Tabellenansicht von der grafischen Benutzeroberfläche mit entfernt; dies bedeutet, daß die Tabelle in der Oberfläche nicht sichtbar ist<sup>1</sup>. Wird ein Feld von der Anzeigelistenansicht entfernt, dann erscheint das Feld nicht mehr in der Benutzeroberfläche. Dies ist nützlich, wenn Felder versteckt werden sollen.
- zwei Knöpfe ‘Rauf’ und ‘Runter’ zum Verschieben des aktiven Elements in der Anzeigelistenansicht nach oben bzw. unten.
- zwei Knöpfe ‘Hinein’ und ‘Heraus’, um das aktive Elements in der Anzeigelistenansicht in der Hierarchie eine Stufe nach oben oder unten zu verschieben.
- ein Knopf ‘Text’ zum Hinzufügen eines Textobjekts zur Anzeigelistenansicht. Siehe Abschnitt 14.3.5 [Text editor], Seite 72 für mehr Information über das Einrichten eines Textobjekts.

---

<sup>1</sup> Anm.d.Übersetzers: Wohl aber physikalisch vorhanden.

- ein Knopf **‘Bild’** zum Hinzufügen eines Bildobjekts (siehe Abschnitt 14.3.6 [Image editor], Seite 72).
- ein Knopf **‘Zwischenraum’** zum Einsetzen von Zwischenräumen zwischen anderen Objekten (siehe Abschnitt 14.3.7 [Space editor], Seite 72).
- ein Knopf **‘Balance’** zum Einfügen eines Balanceobjekts in die Anzeigelistenansicht. Das Balanceobjekt ist sinnvoll, wenn die Größe von anderen Benutzeroberflächenelementen geregelt werden soll.
- ein Knopf **‘Gruppe’** zum Einfügen eines Gruppenelements in die Anzeigelistenansicht. Bevor **‘Gruppe’** gedrückt wird, können mehrere Elemente in der Anzeigelistenansicht ausgewählt werden, die in die neue Gruppe verschoben werden sollen. Siehe Abschnitt 14.3.8 [Group editor], Seite 73 für mehr Informationen zum Einrichten eines Gruppenobjekts.
- ein Knopf **‘Fenster’** zum Hinzufügen eines neuen Fensters in die Anzeigelistenansicht. Wie bei Gruppenobjekten können mehrere Benutzeroberflächenelemente ausgewählt werden, die in das neue Fenster übernommen werden sollen. Mehr zum Einrichten eines Fensters siehe Abschnitt 14.3.10 [Window editor], Seite 74.

Mehr Informationen über die Benutzeroberflächenelemente einschließlich ihrer Anwendung siehe Abschnitt 5.9 [User interface], Seite 29.

### 14.3.2 Paneleditor

Wird ein Panel zu einer Tabellenmaske hinzugefügt oder wird auf ein vorhandenes Panel doppelt geklickt, dann erscheint das **‘Panel’**-Fenster. Dieses Fenster enthält folgende Elemente:

- ein Zeichenkettenfeld **‘Überschrift’** für die Eingabe einer Überschrift, die im Kopf des Panels angezeigt werden soll.
- ein Zeichenkettenfeld **‘Font’** mit einem Popup-Knopf zur Auswahl eines Zeichensatzes für den Titel. Wird dieses Feld leer gelassen, dann wird der Vorgabezeichensatz verwendet.
- ein Feld **‘Hintergrund’** mit einem Checkmark-Feld **‘Vorgabe’** zum Festlegen des Hintergrundes vom Kopf des Panels. Wird das Feld **‘Vorgabe’** aktiviert, dann wird ein Vorgabehintergrund ausgewählt. Anderenfalls kann auf den Knopf **‘Hintergrund’** geklickt werden, um eine Hintergrundeinstellung vorzunehmen (mehr zu dem erscheinenden Fenster in der Anleitung zu MUI).
- ein Feld **‘Nummer/Alle?’**. Wenn aktiviert, dann wird die Nummer des aktuellen Datensatzes und die Anzahl aller Datensätze im rechten Teil des Panelkopfes angezeigt.
- ein Feld **‘filtern?’**, das -wenn aktiviert- einen Filter-Knopf zum Panelkopf hinzufügt. Mit dem Filterknopf kann der Datensatzfilter der Tabelle ein- und ausgeschaltet werden. Wird das Feld nicht aktiviert, dann wird der Menüpunkt **‘Tabelle - Ändere Filter’** zur Tabelle auch nicht aktiviert, was bedeutet, daß auch kein Filterausdruck für die Tabelle angegeben werden kann. Mehr über Datensatzfilter siehe Abschnitt 9.1 [Record filter], Seite 46.
- ein Feld **‘Pfeile?’** zum Ergänzen von zwei Pfeilknöpfen zur Tabellenmaske. Die Pfeilknöpfe ermöglichen das Durchforsten der Datensätze einer Tabelle. Wird dieses Feld nicht aktiviert, dann kann die Tabelle nicht durchforstet werden und der Menüpunkt **‘Gehe zum Datensatz’** samt seiner Unterpunkte, die Menüpunkte **‘Suche nach’**, **‘Suche vorwärts’** und **‘Suche rückwärts’** im Menü **‘Tabelle’** werden nicht aktiviert.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters.

Wurden alle Änderungen durchgeführt, dann wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

### 14.3.3 Feldobjekteditor

Wird ein Feld zur Anzeigelistenansicht hinzugefügt, dann wird für dieses ein vordefiniertes Benutzeroberflächenobjekt erzeugt. Um die Einstellungen des Feldobjektes zu ändern, wird darauf doppelt geklickt und das Fenster **‘Zeigt Feld an’** öffnet sich. Dieses Fenster enthält verschiedene Elemente, die abhängen vom Felddatentyp. Die folgenden Elemente sind bei den meisten Felddatentypen vorhanden:

- ein Zeichenkettenfeld **‘Überschrift’** zum Eingeben einer Überschrift, die neben dem Feldobjekt (oder bei Knöpfen im Objekt selbst) dargestellt wird. Bleibt das Feld leer, dann wird keine Überschrift dargestellt.
- ein Auswahlfeld **‘Position des Titels’** zum Festlegen, an welcher Stelle, bezogen auf das Feldobjekt, die evtl. vorhandene Überschrift angezeigt wird. Man kann zwischen **‘Links’**, **‘Rechts’**, **‘Oben’** und **‘Unten’** wählen.
- ein Zeichenkettenfeld **‘Tastenkürzel’**, das einen Buchstaben aufnehmen kann, der zusammen mit der Taste **Amiga** verwendet wird, um das Objekt zu aktivieren.
- ein Feld **‘Home?’**. Wenn aktiviert, dann wird dieses Objekt zum Startobjekt<sup>2</sup>. Das Startobjekt wird verwendet, um beim Anlegen eines neuen Datensatzes den Cursor dort zu plazieren. Dies ist ziemlich nützlich, wenn nach einem Neuanlegen eines Datensatzes immer an der gleichen Stelle mit dem Eingeben von neuen Daten begonnen werden soll. Wird ein Feldobjekt als Startobjekt festgelegt, dann wird bei allen anderen Objekten der gleichen Tabelle dieses Feld gelöscht.
- ein Feld **‘Nur lesen?’**, das -wenn aktiviert- dem Objekt den Nur-Lese-Status gibt. Dies bedeutet, daß der Inhalt des Objekts nur gelesen, aber nicht verändert werden kann. Ist das Feld gesetzt, so werden die Einstellungen von **‘Tastenkürzel’** und **‘Home?’** ignoriert.
- ein Auswahlfeld **‘Formatierung’** für die Angabe, wie Feldinhalte im Objekt angezeigt werden sollen. Man kann zwischen **‘Mittig’**, **‘Links’** und **‘Rechts’** wählen, um den Inhalt zentriert, linksbündig oder rechtsbündig anzuzeigen.
- ein numerisches Feld **‘Gewichtung’**, um das Gewicht des Objekts festzulegen. Der Wert dieses Feldes gibt an, wieviel Platz bezogen auf andere Objekte das Feld im endgültigen Layout des Fensters erhält. Normalerweise betrifft der Wert dieses Feldes nur die horizontale Größe des Objekts, da die meisten vertikal angeordneten Objekte feste Höhen haben.
- ein Zeichenkettenfeld **‘Font’** mit einem Popup-Knopf zur Auswahl eines Zeichensatzes für den Titel. Wird dieses Feld leer gelassen, dann wird der Vorgabezeichensatz verwendet.
- ein Feld **‘Hintergrund’** mit einem Checkmark-Feld **‘Vorgabe’** zum Festlegen, wie der Hintergrund des Feldes aussehen soll. Wird das Feld **‘Vorgabe’** aktiviert, dann wird ein Vorgabehintergrund ausgewählt. Anderenfalls kann auf den Knopf **‘Hintergrund’** geklickt werden, um eine Hintergrundeinstellung vorzunehmen (mehr zu dem erscheinenden Fenster in der Anleitung zu MUI).
- ein Textfeld **‘Sprechblasenhilfe’**, in dem ein Text eingegeben werden kann, der als Sprechblasenhilfe zum Feldobjekt angezeigt wird.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters.

Wurden alle Änderungen durchgeführt, dann wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

---

<sup>2</sup> Anm.d.Übersetzers: Leider gibt es im Deutschen keine direkte Übersetzung für **‘Home’**, die in diesem Zusammenhang korrekt wäre. Aus diesem Grund wird in MUIbase weiterhin **‘Home’** verwendet. Hier in der Dokumentation wird auch **‘Start’** verwendet.

### 14.3.4 Typabhängige Einstellungen

Neben den obigen Elementen gibt es noch folgende, typabhängige Elemente:

- für Felder vom Typ Zeichenkette gibt es eine Seite **‘Extras’** mit
  - ein Feld **‘Bild anzeigen?’**, das -wenn aktiviert- ein Bildelement an die Zeichenkette hängt, um darin ein Bild anzuzeigen, dessen Dateiname vom Zeichenkettenfeld entnommen wird. Das Bildelement wird oberhalb des Zeichenkettenfeldes angeordnet. Wird dieses Feld nicht aktiviert, dann sind die Felder **‘Titel beim Zeichenkettenfeld?’**, **‘verstecke Zeichenkettenfeld?’** und **‘Größe’** bedeutungslos.
  - ein Feld **‘Titel beim Zeichenkettenfeld?’**. Wenn aktiviert, dann wird die Überschrift des Feldobjekts links neben dem Zeichenkettenfeld angeordnet, so daß das Bildelement mehr Platz im Fenster erhält. Wird dieser Punkt nicht aktiviert, dann wird die Überschrift links neben dem Bild angezeigt.
  - ein Feld **‘verstecke Zeichenkettenfeld?’** zum Weglassen des Zeichenkettenfeldes von der Benutzeroberfläche. Wenn aktiviert, dann wird nur das Bildelement angezeigt.
  - ein Feld **‘Größe’** zum Festlegen, wie die Größe eines Bildes im Bildbereich gehandhabt wird. Ist **‘größenveränderbar’** aktiv, dann kann das Objekt in der Größe verändert werden und das Objekt kann größer werden als die Ausmaße des Bildes. **‘fixiert’** setzt die Größe des Objekts auf die des Bildes. Ändert sich die Größe des Bildes von Datensatz zu Datensatz, dann ändert sich entsprechend auch die Größe des Objekts. **‘scrollbar’** fügt zwei Rollbalken zum Objekt hinzu, um Bilder anzeigen zu können, die größer sind als der sichtbare Ausschnitt des Objekts. Ist **‘skaliert’** aktiviert, dann wird das Bild auf die Größe des Objekts skaliert<sup>3</sup>.
  - ein Feld **‘Dateiauswahl?’**, das -wenn aktiviert- einen Popup-Knopf rechts neben das Zeichenkettenfeld hinzufügt. Dieser Knopf dient dazu, ein Dateiauswahlfenster zum Auswählen einer Datei zu öffnen.
  - ein Feld **‘Zeichensatzauswahl?’** zum Hinzufügen eines Popup-Knopfes, das ein Zeichensatzauswahlfenster öffnet. Diese Eigenschaft wurde nach einem Vorschlag von Ralphie<sup>4</sup> ergänzt. Macht ihn dafür verantwortlich, wenn ihr dieses für nutzlos haltet<sup>5</sup>. :-)
  - ein Feld **‘Listenansicht-Popup’**. Wenn aktiviert, dann wird ein Popup-Knopf rechts neben das Zeichenkettenfeld angehängt, mit dem eine Listenansicht geöffnet wird, aus dem eine Zeichenkette aus der Liste von Zeichenketten ausgewählt werden kann. Diese Liste der Zeichenketten kann im Auswahltexteditor festgelegt werden, das über den Knopf **‘Ändere Auswahltexte ...’** rechts neben dem **‘Listenansicht-Popup’**-Feld aufgerufen wird. Mehr zum Auswahltexteditor siehe Abschnitt 14.2.3 [Label editor], Seite 65.  
 Nur eines der Felder **‘Dateiauswahl?’**, **‘Zeichensatzauswahl?’** und **‘Listenansicht-Popup’** kann aktiviert werden (MUI-Programmierer wissen warum).
  - ein Feld **‘Anzeige?’**, das -wenn aktiviert- einen Knopf rechts neben das Zeichenkettenfeld ergänzt, mit dem ein externer Anzeiger gestartet werden kann, das den Inhalt des Feldes als Argument erhält. Dies ist nützlich, wenn Dateinamen im Zeichenkettenfeld gespeichert werden und man den Inhalt der Dateien über den externen Anzeiger ansehen möchte. Der externe Anzeiger kann über den Menüpunkt **‘Einstellungen - Externen Anzeiger setzen...’** festgelegt werden (siehe Abschnitt 7.8 [External viewer], Seite 38).

<sup>3</sup> Anm.d.Übersetzers: Dabei geht das Seitenverhältnis allerdings verloren!

<sup>4</sup> Anm.d.Übersetzers: Das bin ich :-)

<sup>5</sup> Anm.d.Übersetzers: Errhm, laßt mich versuchen, mich da rauszureden: Also dieser Vorschlag wurde ja nur gemeinsam mit den anderen Popups gemacht und ich habe Steffen auch nicht gezwungen, diesen reinzumachen ... @8-P Abgesehen von: FRECHHEIT! Mich da so reinzuziehn :-)



- für Felder des Typs Auswahl gibt es das Feld **‘Art’**, mit dem ausgewählt werden kann, ob die Feldinhalte als **‘Auswahlknopf’** oder als Satz von **‘Radio-Knöpfe’** dargestellt werden sollen. Wird **‘Auswahlknopf’** gewählt, dann kann die Ausrichtung des Titels auf eine von **‘Links’**, **‘Rechts’**, **‘Oben’** oder **‘Unten’** gesetzt werden. Wird dagegen **‘Radio-Knöpfe’** gewählt, dann erlauben zwei Anwählknöpfe **‘Rahmen?’** und **‘Horizontal?’** das Zeichnen eines Rahmens um die Radioknöpfe bzw. die Festlegung auf eine horizontale Ausrichtung.
- für Felder vom Typ Fließkommazahl gibt es ein Ganzzahlfeld **‘Nachkommastellen’**, in dem die Anzahl der der Nachkommastellen zum Darstellen der Fließkommazahlen eingegeben werden kann.
- für Beziehungsfelder gibt es eine Seite **‘Extras’**, die folgende Punkte enthält:
  - ein Listenansichtsfeld **‘Inhalt’**, in dem festgelegt wird, welcher Inhalt des referenzierten Datensatzes angezeigt werden soll. Es lassen sich mehrere Einträge in dieser Liste auswählen. Wird **‘Datensatznummer’** ausgewählt, dann wird die Datensatznummer des referenzierten Datensatzes in der Anzeige ergänzt. Die anderen Einträge sind die Namen der Felder in der referenzierten Tabelle.<sup>6</sup>
  - ein Textfeld **‘Gewählte Punkte’**, das anzeigt, wieviele Einträge in der darüberliegenden Listenansicht ausgewählt wurden.
  - ein Feld **‘Anzeigen?’**. Wenn ausgewählt, dann wird das Benutzeroberflächenobjekt zum Anzeigen der Beziehung als Knopf erzeugt. Wird auf diesen Knopf gedrückt, so wird der referenzierte Datensatz in der Tabellenmaske der referenzierten Tabelle angezeigt<sup>7</sup>.
  - ein Feld **‘Filtern?’**, das -wenn aktiviert- einen Knopf rechts neben dem Beziehungsfeld ergänzt, mit dem der Datensatzfilter für dieses Feld ein- und ausgeschaltet werden kann. Mehr zu Datensatzfiltern siehe Abschnitt 9.1 [Record filter], Seite 46.
- für virtuelle Felder enthält der Feldobjekteditor:
  - ein Auswahlfeld **‘Art’**, in dem festgelegt wird, wie der Inhalt des virtuellen Feldes dargestellt werden soll. Es läßt sich zwischen **‘Bool’**, das ein Checkmark-Feld zum Darstellen von booleschen Werten anzeigt, **‘Text’**, das ein Textfeld zum Anzeigen einer Zeile Text (einschließlich Datum, Zeit und numerische Werte) und **‘Liste’**, das eine Listenansicht verwendet, um eine Liste von Zeilen anzuzeigen (z.B. das Ergebnis einer SELECT-FROM-WHERE-Abfrage).
  - wenn **‘Art’** auf **‘Text’** gesetzt wird, dann erscheinen zwei weitere Felder: **‘Formatierung’**, um festzulegen, wie der Feldinhalt angezeigt wird und **‘Nachkommastellen’** für die Eingabe der Anzahl von Stellen nach dem Komma, wenn der Feldinhalt eine Fließkommazahl ist.
  - ist **‘Art’** auf **‘Liste’** gesetzt, dann wird ein Feld **‘Zeige Titel?’** verfügbar. Wenn aktiviert, dann wird die erste Zeile des Feldes als Titelzeile in der Listenansicht angezeigt. Anderenfalls wird keine Titelzeile angezeigt und die erste Zeile ignoriert.
  - ein Feld **‘Sofort’**, das -wenn aktiviert- das virtuelle Feld immer dann dazu bringt, neu berechnet zu werden, wenn von einem Datensatz zu einem anderen gewechselt wird. Ist es nicht aktiviert, dann wird das virtuelle Feld nur dann berechnet, wenn ein MUIbase-Programm dessen Wert benötigt, z.B. wenn irgendwo ein Knopf in der Benutzeroberfläche installiert ist, der den Wert des virtuellen Feldes ausliest, nachdem der Knopf gedrückt wurde.
- Für Knöpfe gibt es folgende zusätzlichen Felder:
  - ein Auswahlfeld **‘Art’**, mit dem zwischen **‘Textknopf’** und **‘Symbolknopf’** gewählt werden kann.

<sup>6</sup> Anm.d.Übersetzers: Die Reihenfolge der Einträge richtet sich nach der Reihenfolge, in der die Felder in der Felderliste zur Tabelle angeordnet sind.

<sup>7</sup> Anm.d.Übersetzers: Gegebenenfalls wird das Fenster, in dem die Tabelle eingesetzt ist, geöffnet und nach vorne geholt.

- wird die Art des Knopfes auf **‘Textknopf’** gesetzt, dann erscheinen die weiteren Felder **‘Überschrift’**, **‘Zeichensatz’**, **‘Hintergrund’** und **‘Vorgabe’** für die Eingabe des Textes, der innerhalb des Knopfes dargestellt wird, der Zeichensatz zum Darstellen des Textes und für die Festlegung des Hintergrundes.
- ist die Art des Knopfes **‘Symbolknopf’**, dann erlaubt ein Knopf **‘Bild’** die Angabe des Bildes, das angezeigt werden soll und ein Bereich **‘Größe’** die Einstellung der Handhabung der Größe des Bildes.

### 14.3.5 Texteditor

Wird ein Textobjekt zur Anzeigelistenansicht hinzugefügt oder wird eines durch doppelt klicken verändert, dann öffnet sich ein Fenster **‘Text’**. Dieses Fenster enthält folgende Einträge:

- ein Zeichenkettenfeld **‘Überschrift’** für die Eingabe des Textes, der angezeigt werden soll.
- ein numerisches Feld **‘Gewichtung’**, mit dem die horizontale und vertikale Gewichtung des Textobjekts festgelegt wird.
- ein Auswahlfeld **‘Zeichensatz’** zum Festlegen des Zeichensatzes für den Text. Wird dieses Feld leer gelassen, dann wird der Vorgabezeichensatz verwendet.
- zwei Felder **‘Hintergrund’** und **‘Vorgabe’** zur Festlegung des Hintergrunds vom Textobjekt.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters

Wurden alle Änderungen durchgeführt, dann wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

### 14.3.6 Bildeditor

Der Bildeditor erscheint, wenn ein neues Bildobjekt hinzugefügt wird oder auf ein existierendes doppelt geklickt wird. Es enthält folgende Elemente:

- ein Feld **‘Gewichtung’** zur Festlegung der Gewichtung des Bildobjekts im endgültigen Fensterlayout.
- ein Feld **‘Bild’** zur Festlegung des Bildes, das dargestellt werden soll.
- ein Bereich **‘Größe’**, in dem angegeben wird, wie die Größe des Bildes gehandhabt werden soll. Wird **‘größenveränderbar’** gewählt, dann kann das Bild in der Größe geändert werden<sup>8</sup>. Bei **‘fixed’** hingegen übernimmt das Objekt die Größe des Bildes.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters

Wurden alle Änderungen durchgeführt, dann wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

### 14.3.7 Zwischenraumeditor

Nachdem ein Zwischenraumobjekt zur Anzeigelistenansicht hinzugefügt wurde, kann man durch Doppelklicken dessen Voreinstellungen ändern. Dies öffnet das Fenster **‘Zwischenraum’** mit den folgenden Elementen:

---

<sup>8</sup> Anm.d.Übersetzers: Auch hier wird das Seitenverhältnis mißachtet!

- ein Feld **‘Trennstrich?’**, das -wenn aktiviert- eine vertikale oder horizontale Trennlinie (abhängig von der Anordnung der übergeordneten Objekte) in der Mitte des Zwischenraumobjekts anzeigt. Dies ist nützlich, um Teile innerhalb eines Fensterlayouts aufgeteilt werden sollen.
- ein numerisches Feld **‘Gewichtung’** zur Festlegung der Gewichtung des Bildobjekts.
- zwei Felder **‘Hintergrund’** und **‘Vorgabe’** zur Festlegung des Hintergrunds.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters

Wurden alle Änderungen durchgeführt, dann wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

### 14.3.8 Gruppeneditor

Nachdem ein Gruppenobjekt zur Anzeigelistenansicht hinzugefügt wurde, kann man durch Doppelklicken dessen Voreinstellungen ändern. Dies öffnet das Fenster **‘Gruppe’**, das folgende Elemente anbietet:

- ein Zeichenkettenfeld **‘Überschrift’** für die Eingabe einer Überschrift, die zentriert über der Gruppe angezeigt werden soll. Wird dieses Feld leer gelassen, dann erscheint keine Überschrift.
- ein numerisches Feld **‘Gewichtung’** zur Festlegung der Gewichtung des Gruppenobjekts.
- zwei Felder **‘Hintergrund’** und **‘Vorgabe’** zur Festlegung des Hintergrunds.
- ein Feld **‘Rahmen?’**, der -wenn aktiviert- einen Rahmen um die Gruppe zeichnet.
- ein Feld **‘Horizontal?’**. Wenn aktiviert, dann werden die Elemente der Gruppe horizontal angeordnet und die Gruppe wird in der Anzeigelistenansicht als **‘Horiz.Gruppe’** aufgelistet. Anderenfalls wird die Gruppe vertikal angeordnet und die Anzeigelistenansicht zeigt ein **‘Vert.Gruppe’** für diese Gruppe an.
- ein Feld **‘Zwischenräume?’**, das -wenn aktiviert- etwas Platz zwischen den Gruppenelementen einfügt. Anderenfalls wird kein Platz zwischen den Objekten vorgesehen.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters

Wurden alle Änderungen durchgeführt, dann wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

### 14.3.9 Karteikarten-Gruppeneditor

Man klickt doppelt auf ein Karteikarten-Gruppenobjekt, um dessen Einstellungen zu ändern. Dies öffnet das Fenster **‘Karteikarten-Gruppe’**, das folgende Elemente anbietet:

- ein numerisches Feld **‘Gewichtung’** zur Festlegung der Gewichtung des Objekts.
- zwei Felder **‘Hintergrund’** und **‘Vorgabe’** zur Festlegung des Hintergrunds.
- Ein Bereich **‘Auswahltexte’** zum Festlegen der Auswahltexte für jede Karteikarten-Seite. Man sollte genau so viele Auswahltexte angeben, wie Elemente in der Karteikarten-Gruppe sind. Mehr zum Eingeben und Ändern der Auswahltexte siehe Abschnitt 14.2.3 [Label editor], Seite 65.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters

Wurden alle Änderungen durchgeführt, dann wird **‘Ok’** gedrückt, um das Fenster zu verlassen.

### 14.3.10 Fenstereditor

Um die Einstellungen für ein Fensterobjekt zu ändern, wird doppelt draufgeklickt. Dies öffnet den Fenstereditor mit den folgenden Elementen:

- ein Zeichenkettenfeld ‘**Überschrift**’, in dem eine Zeichenkette eingegeben werden kann, die in der Fensterleiste und im Fensterknopf angezeigt werden soll.
- ein Zeichenkettenfeld ‘**Tastenkürzel**’, in der die Taste zum Aktivieren des Fensterknopfes eingegeben wird.
- ein numerisches Feld ‘**Gewichtung**’ zur Festlegung der Gewichtung des Fensterknopfes.
- ein Auswahlfeld ‘**Zeichensatz**’ zum Festlegen des Zeichensatzes für den Text des Fensterknopfes. Wird dieses Feld leer gelassen, dann wird der Vorgabezeichensatz verwendet.
- zwei Felder ‘**Hintergrund**’ und ‘**Vorgabe**’ zur Festlegung des Hintergrunds vom Fensterknopf.
- ein Feld ‘**Deaktiviert?**’, das -wenn aktiviert- einen Fensterknopf im deaktivierten Zustand aufbaut. Auf ihn läßt es sich nicht klicken und kann daher nicht zum Öffnen und Schließen des Fensters verwendet werden. Dies kann nützlich sein, wenn man nicht möchte, daß der Benutzer die Fenster selbst öffnen darf, aber diese von einem MUIbase-Programm aus geöffnet werden sollen.
- zwei Knöpfe ‘**Ok**’ und ‘**Abbrechen**’ zum Verlassen des Fensters

Wurden alle Änderungen durchgeführt, dann wird ‘**Ok**’ gedrückt, um das Fenster zu verlassen.

## 14.4 Struktur ausdrucken

Manchmal ist es nützlich, eine Übersicht über alle Tabellen und Felder eines Projekts zu erhalten, z.B. wenn ein MUIbase-Programm geschrieben werden soll. Dies läßt sich über den Menüpunkt ‘**Projekt – Struktur ausdrucken...**’ erledigen. Es wird nach einem Dateinamen gefragt, wohin die Liste der Tabellen und Felder ausgegeben werden soll.

Die Ausgabe listet zunächst den Projektnamen auf, gefolgt von allen Tabellen in diesem Projekt. Für jede Tabelle werden alle Felder mit ihren Typen ausgegeben.

## 15 MUIbase programmieren

Dieses Kapitel<sup>1</sup> beschreibt die Programmiersprache von MUIbase, einschließlich aller verfügbaren Funktionen. Das Kapitel dient jedoch nicht als allgemeine Anleitung für Programmieren. Man sollte mit den Grundzügen der Programmierung vertraut sein und schon kleinere<sup>2</sup> Programme geschrieben haben<sup>3</sup>.

### 15.1 Programmeditor

Um ein Programm für ein Projekt einzugeben, öffnet man den Programmeditor über den Menüpunkt **Programm – Ändern...**. Dies öffnet das Fenster **Ändere Programm** mit:

- einem Texteditorfeld, in dem das Programm editiert wird
- einem Knopf **Kompilieren & Schließen**, mit dem das Programm kompiliert und nach einer erfolgreichen Kompilation der Programmeditor verlassen wird.
- einem Knopf **Kompilieren**, um das Programm zu kompilieren. Enthält das Programm irgendwo einen Fehler, dann wird eine Fehlerbeschreibung in den Fenstertitel und der Cursor an die fehlerhafte Stelle gesetzt.
- einem Knopf **Rückgängig machen**, der alle Änderungen seit der letzten erfolgreichen Kompilation verwirft.

Der Programmeditor ist ein asynchrones Fenster. Das Fenster kann offen gelassen und weiterhin mit dem Rest der Anwendung gearbeitet werden. Man kann den Editor zu jeder Zeit schließen, indem dessen Fensterschließ-Knopf gedrückt wird. Wurden Änderungen seit der letzten erfolgreichen Kompilation vorgenommen, dann erscheint ein Sicherheitsfenster, das um Bestätigung fragt, ob das Fenster geschlossen werden darf.

Man kann das Projektprogramm auch ohne das Öffnen des Programmeditors kompilieren, indem der Menüpunkt **Programm – Kompilieren** ausgewählt wird. Dies ist nützlich, wenn Änderungen an einer Einfügedatei gemacht wurden und diese Änderungen in das Projektprogramm einfließen sollen.

### 15.2 Vorverarbeitung

MUIbase-Programme werden vorverarbeitet, wie ein C-Compiler einen C-Quellcode vorverarbeitet. Dieser Abschnitt beschreibt, wie die Vorverarbeitungs-Anweisungen verwendet werden.

Alle Anweisungen beginnen mit dem Rauten-Symbol **#**, welcher das erste Zeichen in einer Zeile sein muß. Leerzeichen und Tabulatoren können nach dem begonnenen **#** folgen.

---

<sup>1</sup> Anm.d.Übersetzers: einfach nur "Kapitel" ist hier reichlich untertrieben, denn es müßte besser "riesen Kapitel" heißen, da es mehr als 50% des gesamten Dokuments ausmacht.

<sup>2</sup> Anm.d.Übersetzers: richtig funktionierende ;-)

<sup>3</sup> Anm.d.Übersetzers: Wer schon einmal Installer-Scripte geschrieben hat, dem wird es am leichtesten fallen.

### 15.2.1 #define

```
#define name string
```

Definiert ein neues Symbol mit dem gegebenen Namen und Inhalt. Das Symbol *string* kann jeder Text einschließlich Leerzeichen sein und endet am Zeilenende. Paßt *string* nicht in eine Zeile, dann können weitere Zeilen durch ein Backslash-Zeichen \ am Ende jeder Zeile (außer der letzten) verwendet werden. Taucht das Symbol *name* im verbleibenden Quellcode auf, dann wird es durch den Inhalt von *string* ersetzt.

Beispiel: '(PRINTF "X ist %i" X)' gibt 'X ist 1' aus (*name*'s in Zeichenketten werden nicht verändert.)

Das Ersetzen von definierten Symbolen geschieht syntaktisch, das bedeutet, daß Symbole durch jeden Text ersetzt werden können, z.B. man kann sich seine eigene Syntax wie im folgenden Beispiel definieren:

```
#define BEGIN (
#define END )

BEGIN defun test ()
    ...
END
```

Die zu ersetzende Zeichenkette einer Definition kann ein anderes Symbol enthalten, das mit der #define-Anweisung definiert wurde, um verschachtelte Definitionen zu ermöglichen. Es gibt jedoch eine Obergrenze von 16 verschachtelten Definitionen.

Siehe auch #undef, #ifdef, #ifndef.

### 15.2.2 #undef

```
#undef name
```

Entfernt die Definition des Symbols *name*. Wurde *name* nicht definiert, dann passiert nichts.

Siehe auch #define, #ifdef, #ifndef.

### 15.2.3 #include

```
#include filename
```

Liest den Inhalt von *filename* (eine Zeichenkette in Anführungszeichen) an diese Stelle. MUIbase sucht im aktuellen Verzeichnis und im in den Einstellungen festgelegten Verzeichnis (siehe Einfügedateienverzeichnis<sup>4</sup>) nach der zu ladenden Datei.

---

<sup>4</sup> Anm.d.Übersetzers: Hier ist wegen Einschränkungen bei Texinfo leider kein Verweis möglich.

Der Dateiinhalt wird durch den Compiler verarbeitet, als ob er Teil des momentanen Quellcodes wäre.

Eine externe Datei kann eine oder mehrere externe Dateien enthalten. Es gibt jedoch eine Obergrenze von 16 verschachtelten **#include**-Anweisungen. Um zu vermeiden, daß Dateien mehr als einmal eingefügt werden, kann die bedingte Kompilation verwendet werden.

Man sollte vorsichtig sein, wenn man Quellcode in externe Dateien auslagert! Die Fehlersuche und das Auffinden von Fehlern in externen Dateien wesentlich schwerer. Man verschiebe daher nur getesteten und Projekt-unabhängigen Code in externe Dateien.

### 15.2.4 **#if**

**#if** *const-expr*

Ist der gegebene konstante Ausdruck *const-expr* nicht NIL, dann wird der Text bis zum zugehörigen **#else**, **#elif** oder **#endif** zur Kompilation verwendet, anderenfalls (der Ausdruck lieferte NIL) wird der Text bis zum zugehörigen **#else**, **#elif** oder **#endif** nicht zur Kompilation herangezogen.

Momentan können nur TRUE und NIL als konstante Ausdrücke verwendet werden.

Siehe auch **#ifdef**, **#ifndef**, **#elif**, **#else**, **#endif**.

### 15.2.5 **#ifdef**

**#ifdef** *name*

Ist das gegebene Symbol *name* mit einer **#define**-Anweisung definiert worden, dann wird der Text bis zum zugehörigen **#else**, **#elif** oder **#endif** zur Kompilation verwendet, anderenfalls nicht betrachtet.

Siehe auch **#if**, **#ifndef**, **#elif**, **#else**, **#endif**.

### 15.2.6 **#ifndef**

**#ifndef** *name*

Ist das gegebene Symbol *name* nicht mit einer **#define**-Anweisung definiert worden, dann wird der Text bis zum zugehörigen **#else**, **#elif** oder **#endif** zur Kompilation verwendet, anderenfalls nicht betrachtet.

Siehe auch **#if**, **#ifdef**, **#elif**, **#else**, **#endif**.

### 15.2.7 **#elif**

`#elif const-expr`

Beliebig viele `#elif`-Anweisungen können zwischen einem `#if`, `#ifdef` oder `#ifndef` und dem zugehörigen `#else` oder `#endif` auftreten. Die Zeilen, die der `elif`-Anweisung folgen, werden für eine Kompilation verwendet, aber nur dann, wenn jede der folgenden Bedingungen erfüllt ist:

- Der konstante Ausdruck in der vorherigen `#if`-Anweisung lieferte NIL, das Symbol *name* im vorherigen `ifdef` war nicht definiert oder das Symbol *name* in der vorherigen `ifndef`-Anweisung war definiert.
- Die konstanten Ausdrücke in allen dazwischenliegenden `elif`-Anweisungen lieferten NIL.
- Der momentane konstante Ausdruck liefert nicht NIL.

Wenn die oben genannten Bedingungen stimmen, dann werden die nachfolgenden Anweisungen `#elif` und `#else` bis zum zugehörigen `#endif` ignoriert.

Siehe auch `#if`, `#ifdef`, `#ifndef`, `#else`, `#endif`.

### 15.2.8 `#else`

`#else`

Dies kehrt den Sinn einer bedingten Anweisung um, der sonst gestimmt hätte. Wenn die vorhergehende Bedingung anzeigen würde, daß Zeilen eingefügt werden, dann werden die Zeilen zwischen `#else` und dem zugehörigen `#endif` ignoriert. Zeigt die vorhergehende Bedingung an, daß Zeilen ignoriert werden, dann werden nachfolgende Zeilen für die Kompilation eingefügt.

Bedingte Anweisungen und dazugehörige `#else`-Anweisungen können verschachtelt werden. Es gibt jedoch eine maximale Verschachteltiefe von 16 verschachtelten bedingten Anweisungen.

Siehe auch `#if`, `#ifdef`, `#ifndef`, `#elif`, `#endif`.

### 15.2.9 `#endif`

`#endif`

Beendet einen Bereich von Zeilen, der mit einer der bedingten Anweisungen `#if`, `#ifdef` oder `#ifndef` beginnt. Jeder dieser Anweisungen muß eine zugehörige `#endif`-Anweisung besitzen.

Siehe auch `#if`, `#ifdef`, `#ifndef`, `#elif`, `#else`.

## 15.3 Programmiersprache

MUIbase verwendet eine Programmiersprache mit einen lisp-ähnlichen Aufbau. Tatsächlich sind einige Konstrukte und Funktionen vom Standard-lisp entnommen worden. MUIbase ist jedoch nicht vollständig kompatibel zu Standard-lisp. Viele Funktionen fehlen (z.B. destruktive Befehle) und die Bedeutung einiger Befehle unterscheiden sich (z.B. der Befehl `return`).



### 15.3.1 Warum lisp?

Der Vorteil einer lisp-ähnlichen Sprache ist, daß man sowohl funktionell als auch mit Befehlen programmieren kann. Funktionelle Sprachen sind in mathematischen Anwendungen weit verbreitet. Das Grundelement von funktionellen Sprachen ist die Verwendung von Ausdrücken. Funktionen werden in mathematischer Schreibweise definiert und häufig wird Rekursion benutzt.

Befehlssprachen (wie C, Pascal, Modula) benutzen eine Befehlsvorschrift, wie etwas berechnet werden soll<sup>5</sup>. Hier ist das Grundelement der Zustand (z.B. Variablen) und ein Programm berechnet dessen Ausgabe durch den Sprung von einem Zustand in einen anderen (z.B. durch Zuweisen von Werten an Variablen).

Lisp kombiniert beide Techniken und deshalb kann man auswählen, auf welche Art etwas implementiert werden soll. Man verwendet diejenige, die besser zum vorgegebenen Problem paßt oder die man lieber mag.

### 15.3.2 Lisp-Aufbau

Ein lisp-Ausdruck ist entweder eine Konstante, eine Variable oder ein Funktionsausdruck. Beim Funktionsaufruf verwendet lisp eine Prefix-Notation<sup>6</sup>. Die Funktion und seine Parameter werden von runden Klammern eingeschlossen. Um zum Beispiel zwei Werte **a** und **b** zu addieren, schreibt man

```
(+ a b)
```

Alle Ausdrücke liefern einen Wert, z.B. im obigen Beispiel die Summe von **a** und **b**. Ausdrücke können verschachtelt werden, d.h. man kann Ausdrücke als Unterausdruck in eine andere einsetzen.

Funktionen werden nach dem call-by-value-Schema<sup>7</sup> aufgerufen, das bedeutet, daß zuerst die Parameter berechnet werden, bevor die Funktion aufgerufen wird.

<sup>5</sup> Anm.d.Übersetzers: genauer gesagt ist es eine Zustandsänderungsvorschrift

<sup>6</sup> Anm.d.Übersetzers: Für diejenigen, die nicht wissen, was es mit Prefix-, Infix- und Postfix-Notationen auf sich hat: Es gibt verschiedene Möglichkeiten, ein Programm intern zu verarbeiten, damit Parameter von Funktionen verarbeitet werden können. Jeder Ausdruck wird i.d.R. auf zwei Parameter plus einem Operator reduziert, bevor er ausgeführt wird (davon merkt der Programmierer nichts, diese Aufgabe übernimmt der Compiler). Dieser Operator für die Verarbeitung der beiden Parameter kann dabei an drei Stellen stehen. Die uns geläufige und im Alltag verwendete Form ist die Infix-Notation. Eine Addition sieht danach so aus: '1 + 2'. Der Operator steht hier in der Mitte, d.h. er ist Infix. Die Postfix-Notation hat den Operator -wie der Name 'post' schon sagt- am Ende des Ausdrucks. Die Addition wäre dann '1 2 +'. Diese Methode wird bei Stack-basierten Interpretern verwendet, die erst beide Parameter auf den Stack legen, dann anhand des Operators (hier +) beide Zahlen vom Stack holen, die Operation durchführen und das Ergebnis wieder auf den Stack legen. Schließlich gibt es noch die Prefix-Notation. Die Addition hat dann das Aussehen '+ 1 2'. Dies ist für zustandsorientierte Interpreter nützlich, die anhand des zuerst gelesenen Operators wissen, wieviele Parameter sie benötigen und lesen sie dementsprechend ein. Dafür wird nicht mal ein Stack verwendet.

<sup>7</sup> Anm.d.Übersetzers: Neben Call-by-value gibt es noch Call-by-reference. Bei letzterem wird nicht der Wert einer Variable übergeben, sondern eine Referenz auf diese Variable. Dies hat z.B. in C den großen Vorteil, daß man in Unterprogrammen auf Datenbestände in übergeordneten Funktionen zugreifen kann. So kann man eine Zeichenkette im Unterprogramm ändern und

Wenn nicht anders angegeben, sind alle Funktionen strikt, d.h. *alle* Parameter einer Funktion müssen zuerst ermittelt werden, bevor sie an die Funktion übergeben und diese ausgeführt wird. Einige Funktionen sind jedoch nicht strikt, z.B. **IF**, **AND** und **OR**. Diese Funktionen brauchen nicht alle Parameter ermitteln.

### 15.3.3 Programmarten

MUIbase kennt drei Programmarten. Die erste ist das Projektprogramm. Im Programm dieser Art können Funktionen und globale Variablen definiert werden. Die Funktionen können als Auslösefunktionen für Felder verwendet werden. Ein Projektprogramm wird im Programmeditor (siehe Abschnitt 15.1 [Program editor], Seite 75) festgelegt.

Die zweite Art ist das Abfrageprogramm. Für dieses können nur Ausdrücke eingegeben werden. Ein solcher Ausdruck darf globale Variablen enthalten und Funktionen aufrufen, die im Projektprogramm definiert wurden. Man kann jedoch in einem Abfrageprogramm keine neuen globalen Variablen und Funktionen festlegen. Abfrageprogramme werden im Abfrageeditor (siehe Abschnitt 13.2 [Query editor], Seite 58) eingegeben.

Die dritte Programmart sind Filterausdrücke. Hier lassen sich nur Ausdrücke eingeben, die vordefinierte MUIbase-Funktionen aufrufen. Es sind nicht alle vordefinierten Funktionen verfügbar und zwar nur solche die keine Seiteneffekte aufweisen, z.B. kann man keine Funktion verwenden, die Daten in eine Datei schreibt. Filterausdrücke werden im Filterfenster (siehe Filter ändern<sup>8</sup>) verändert.

### 15.3.4 Namenskonventionen

In einem MUIbase-Programm können Symbole, wie Funktionen, lokale und globale Variablen, definiert werden. Die Namen dieser Symbole müssen folgenden Punkten genügen:

- das erste Zeichen eines Namens muß ein Kleinbuchstabe sein. Dies unterscheidet Programmsymbole von Tabellen- und Feldnamen.
- die folgenden Zeichen können Buchstaben<sup>9</sup>, Ziffern und Unterstriche sein. Andere Zeichen wie deutsche Umlaute sind nicht zulässig.

### 15.3.5 Datensatzinhalte ansprechen

Um auf Tabellen und Felder in einem MUIbase-Programm zugreifen zu können, muß ein Pfad zu diesen angegeben werden. Ein Pfad ist eine durch Punkte getrennte Liste von Komponenten, wobei jede Komponente der Name einer Tabelle oder eines Feldes ist.

---

die neue Zeichenkette in der Funktion, von der aus das Unterprogramm aufgerufen wurde, zur Weiterbearbeitung verwenden. In C hat man zudem auch den Vorteil, daß man eine Funktion in Form einer Referenz auf diese übergeben und diese Funktion dann vom Unterprogramm aus aufrufen kann.

<sup>8</sup> Anm.d.Übersetzers: Hier ist wegen Einschränkungen bei Texinfo leider kein Verweis möglich.

<sup>9</sup> Anm.d.Übersetzers: a-z, groß und klein

Pfade können entweder relativ oder absolut sein. Absolute Pfade beginnen mit dem Tabellennamen als erste Komponente, gefolgt von einer Liste von Feldern, die zum gewünschten Feld hinführen, auf das man zugreifen möchte. Zum Beispiel greift der absolute Pfad `'Person.Name'` auf das Feld `'Name'` im aktuellen Datensatz der Tabelle `'Person'` zu oder der absolute Pfad `'Person.Vater.Name'` auf das Feld `'Name'` in dem Datensatz, der durch das Feld `'Vater'` referenziert wird (der ein Beziehungsfeld zur Tabelle `'Person'` ist).

Relative Pfade haben schon eine aktuelle Tabelle, auf die sie sich beziehen. Zum Beispiel ist in einem Filterausdruck die aktuelle Tabelle diejenige, für die der Filterausdruck geschrieben wird. Der relative Pfad<sup>10</sup> für ein Feld in der aktuellen Tabelle ist dann nur der Feldname selbst. Auf Felder, auf die nicht direkt von der aktuellen Tabelle aus zugegriffen werden kann, sondern indirekt über eine Beziehung, dann gelten die gleichen Regeln wie für absolute Pfade.

Es ist nicht immer eindeutig, ob ein angegebener Pfad ein relativer oder ein absoluter ist, z.B. bei einem Filterausdruck für eine Tabelle `'Foo'`<sup>11</sup>, das ein Feld `'Bar'` besitzt, wenn es auch eine Tabelle `'Bar'` gibt. Wird nun `'Bar'` eingegeben, dann ist unklar, was gemeint ist: die Tabelle oder das Feld? Daher werden alle Pfade zuerst als relative Pfade betrachtet. Wird kein Feld für den angegebenen Pfad gefunden, dann wird der Pfad global betrachtet. In unserem Beispiel würde das Feld bevorzugt.

Was aber, wenn im Beispiel oben auf die Tabelle zugegriffen werden soll? In dem Fall muß der Pfad absolut angegeben werden. Um einen Pfad als global zu kennzeichnen, müssen vor dem Pfad zwei Doppelpunkte angefügt werden. In unserem Beispiel müßte man `'::Bar'` eingeben, um auf die Tabelle zuzugreifen.

Um Pfade und ihre Anordnungen besser zu verstehen, betrachten wir hier als Beispiel, daß das Feld `'Bar'` in der Tabelle `'Foo'` eine Beziehung zur Tabelle `'Bar'` ist und die Tabelle `'Bar'` enthält ein Feld `'Name'`. Nun kann man auf das Feld `'Name'` zugreifen, indem man `'Bar.Name'` oder `'::Bar.Name'` eingibt. Beide Ausdrücke haben unterschiedliche Bedeutungen. `'::Bar.Name'` bedeutet, daß der aktuelle Datensatz der Tabelle `'Bar'` hergenommen wird und der Wert des Feldes `'Name'` in diesem Datensatz zurückgeliefert wird, wohingegen `'Bar.Name'` den aktuellen Datensatz von `'Foo'` hernimmt, die Beziehung des Feldes `'Bar'` ermittelt und diesen Datensatz zum Ermitteln des Wertes vom Feld `'Name'` verwendet.

Um ein komplexeres Beispiel anzubringen, stellen wir uns vor, daß die Tabelle `'Bar'` zwei Datensätze hat. Der eine enthält im Feld `'Name'` den Eintrag `'Mats'`<sup>12</sup> und der andere `'Steffen'`. Der erste Datensatz ist der momentan aktive. Des weiteren hat die Tabelle `'Foo'` einen Datensatz (den aktuellen), dessen Feld `'Bar'` auf den zweiten Datensatz der Tabelle `'Bar'` verweist. `'::Bar.Name'` liefert jetzt `'Mats'` und `'Bar.Name'` liefert `'Steffen'`.

<sup>10</sup> Anm.d.Übersetzers: Auch innerhalb von (SELECT)-Anweisungen und (FOR ALL)-Schleifen sind relative Pfade gültig.

<sup>11</sup> Anm.d.Übersetzers: Das Wort `'Foobar'` tritt in sehr vielen Beispielen -nicht nur bei MUIbase- auf. Es wurde aus `'fubar'` gebildet, da es sich genauso spricht und welches eine Abkürzung mit 5 Wörtern ist. Ich habe diverse Interpretationen und kann auch nicht sagen, was es bedeutet.

<sup>12</sup> Anm.d.Übersetzers: Mats war leider schneller beim Registrieren von AmigaBase, dem Vorgänger von MUIbase und hat auch früher mit dem Beta-Testen von MUIbase begonnen als ich. Drum hab ich ihn hier beim Übersetzen auch nicht umbenannt ;-)

### 15.3.6 Datentypen zum Programmieren

Die Programmiersprache von MUIbase kennt die folgenden Datentypen:

Typ	Beschreibung
Bool	alle Ausdrücke. Nicht-NIL-Ausdrücke werden als TRUE betrachtet.
Integer	lange Ganzzahl, 32 bit, Auswahlwerte werden automatisch in Integer umgewandelt
Real	double, 64 bit
String	Zeichenketten von unterschiedlicher Länge
Memo	wie Zeichenketten, aber zeilenorientiertes Format
Date	Datumswerte
Time	Zeitwerte
Record	Zeiger auf einen Datensatz
File	Dateideskriptor zum Lesen/Schreiben
List	Liste von Elementen, NIL ist eine leere Liste.

Alle Programmiertypen unterstützen den Wert NIL.

### 15.3.7 Konstanten

Die Programmiersprache von MUIbase kann konstante Ausdrücke handhaben, die abhängig von ihrem Typ eingegeben werden kann:

Typ	Beschreibung
Integer	Ganzzahlkonstanten im Bereich von -2147483648 bis 2147483647 können wie üblich angegeben werden.
Real	Fließkommazahlen im Bereich von -3.59e308 bis 3.59e308 können wie üblich angegeben werden, sowohl im wissenschaftlichen als auch nicht-wissenschaftlichen Format. Wird der Dezimalpunkt vergelassen, so wird die Zahl nicht als Real betrachtet, sondern als Integer.
String	Zeichenkettenkonstanten sind jedes Zeichen in einer Kette, umschlossen mit doppelten Anführungsstrichen,

z.B. "Beispielzeichenkette". Innerhalb der doppelten Anführungszeichen kann jedes Zeichen angegeben werden, mit Ausnahme von Steuerzeichen und neuen Zeilen. Es gibt jedoch besondere Escapezeichen zum Eingeben solcher Zeichen:

<code>\n</code>	neue Zeile (nl)
<code>\t</code>	horizontaler Tabulator (ht)
<code>\v</code>	vertikaler Tabulator (vt)
<code>\b</code>	Rückschritt (bs)
<code>\r</code>	Wagenrücklauf (cr)
<code>\f</code>	Seitenumbruch (ff)
<code>\\</code>	der Backslash selbst
<code>\"</code>	doppeltes Anführungszeichen
<code>\e</code>	Escapezeichen 033
<code>\nnn</code>	Zeichen mit dem Oktalcode <i>nnn</i>
<code>\xnn</code>	Zeichen mit dem Hexcode <i>nn</i>

Memo	Es gibt keine konstanten Werte, sondern stattdessen nur eine konstante Zeichenkette.
Date	Konstante Datumswerte können in einem der Formate 'TT.MM.JJJJ', 'MM/TT/JJJJ' oder 'JJJJ-MM-TT' angegeben werden, wobei 'TT', 'MM' und 'JJJJ' die zwei- und vierstelligen Werte für Tag, Monat bzw. Jahr darstellen.
Time	Konstante Zeitwerte werden im Format 'HH:MM:SS' angegeben, wobei 'HH' ein zweistelliger Wert im Bereich 0 bis 23 ist, der Stunden darstellt, 'MM' ein zweistelliger Wert im Bereich 0 bis 59 ist, der Minuten repräsentiert und 'SS' ein zweistelliger Wert im Bereich 0 bis 59 ist, der Sekunden anzeigt.

Für andere vordefinierte Konstanten, siehe Abschnitt 15.24 [Pre-defined constants], Seite 152.

### 15.3.8 Befehlsaufbau

Im Rest dieses Kapitels findet man die Beschreibung aller Befehle und Funktionen, die für die Programmierung von MUIbase zur Verfügung stehen. Der folgende Aufbau wird verwendet, um die Befehle zu beschreiben:

- Text innerhalb von `[]` ist optional. Wird der Text innerhalb der Klammern weggelassen, dann wird ein Vorgabewert angenommen.
- Texte, die durch senkrechte Striche `|` getrennt werden, geben verschiedene Optionen an. Z.B. bedeutet `'a | b'`, daß entweder `'a'` oder `'b'` angegeben werden kann.
- Text, der im Schriftstil *var* geschrieben wird, ist ein Platzhalter, der mit anderen Ausdrücken gefüllt werden kann.

- Punkte ... zeigen an, daß weitere Ausdrücke folgen können.
- jeder andere Text ist obligatorisch.

Anm.d.Übersetzers: Bei der Beschreibung der Befehle wird die englische Namensgebung der Parameter etc. beibehalten. Dies vermeidet Fehler bei der Übersetzung und bleibt dennoch verständlich.

## 15.4 Befehle definieren

Dieser Abschnitt listet Befehle zum Definieren von Funktionen und globalen Variablen auf. Die Befehle sind nur für Projektprogramme verfügbar.

### 15.4.1 DEFUN

DEFUN definiert eine Funktion mit einem festgelegten Namen, einer Liste von Parametern, die an die Funktion weitergereicht und eine Liste von Ausdrücken, die abgearbeitet werden.

```
(DEFUN name (varlist) expr ...)
```

Der Name der Funktion muß mit einem Kleinbuchstaben beginnen, gefolgt von weiteren Zeichen, Ziffern und Unterstrich-Zeichen. (siehe Abschnitt 15.3.4 [Name conventions], Seite 80).

Der Parameter *varlist* legt die Parameter der Funktion fest:

```
varlist: var1 ...
```

wobei *var1* ... die Namen der Parameter sind. Die Namen müssen den gleichen Regeln wie die des Funktionsnamens genügen.

Es ist auch möglich, Typdeklarierer an die Parameter zu hängen (siehe Abschnitt 15.26 [Type specifiers], Seite 154).

Die Funktion führt die Ausdrücke *expr*, ... der Reihe nach aus und liefert den Wert des letzten Ausdrucks. Die Funktion kann auch weitere Funktionen einschließlich sich selbst aufrufen. Eine selbstdefinierte Funktion wird wie eine vordefinierte Funktion aufgerufen.

Um zum Beispiel die Anzahl der Elemente einer Liste zu zählen, kann folgende Funktion definiert werden:

```
(DEFUN len (l)
  (IF (= l NIL)
    0
    (+ 1 (len (REST l)))
  )
)
```

Mit **DEFUN** definierte Funktionen werden in Popuplisten von Tabellen- und Feldfenstern aufgelistet (siehe Abschnitt 14.1.1 [Creating tables], Seite 62 und Abschnitt 14.2.1 [Creating attributes], Seite 64).

Dieser Befehl ist nur für Projektprogramme verfügbar.

Siehe auch **DEFUN\***, **DEFVAR**.

### 15.4.2 **DEFUN\***

**DEFUN\*** ist die Stern-Variante von **DEFUN** und hat den selben Effekt wie **DEFUN** (siehe Abschnitt 15.4.1 [**DEFUN**], Seite 84). Der einzige Unterschied ist, daß Funktionen, die mit **DEFUN\*** definiert wurden, beim Erzeugen oder Ändern von Tabellen und Feldern nicht in den Popuplisten aufgelistet werden. Es ist jedoch möglich, den Funktionsnamen in den entsprechenden Zeichenkettenfeldern einzugeben.

Dieser Befehl ist nur für Projektprogramme verfügbar.

Siehe auch **DEFUN**, **DEFVAR**.

### 15.4.3 **DEFVAR**

**(DEFVAR var [expr])**

Definiert eine globale Variable mit dem Vorgabewert aus *expr* oder **NIL**, wenn *expr* fehlt. Der Name der Variablen muß mit einem Kleinbuchstaben beginnen, gefolgt von weiteren Zeichen, Ziffern und Unterstrich-Zeichen. (siehe Abschnitt 15.3.4 [Name conventions], Seite 80).

Man kann Typdeklarierer an den Variablennamen anhängen (siehe Abschnitt 15.26 [Type specifiers], Seite 154).

**DEFVAR** ist nur verfügbar für Projektprogramme. Alle **DEFVAR**-Befehle sollten am Anfang vor allen Funktionsdefinitionen platziert werden.

Beispiel: **'(DEFVAR x 42)'** definiert eine globale Variable **'x'** mit dem Wert 42.

Es gibt einige vordefinierte Variablen in MUIbase (siehe Abschnitt 15.23 [Pre-defined variables], Seite 152).

Siehe auch **DEFUN**, **DEFUN\***, **LET**.

## 15.5 Programmsteuerungsfunktionen

Dieser Abschnitt listet Funktionen zur Programmflußkontrolle auf, z.B. Funktionen zum Definieren von lokalen Variablen, Schleifenfunktionen, bedingte Programmausführung, Schleifenkontrollfunktionen und mehr.

### 15.5.1 PROGN

Um mehrere Ausdrücke der Reihe nach auszuführen, wird der **PROGN**-Aufruf<sup>13</sup> verwendet.

```
([expr ...])
```

führt *expr* ... der Reihe nach aus. Liefert das Ergebnis des letzten Ausdrucks (oder NIL, wenn kein Ausdruck angegeben wurde). In Lisp ist dieser Aufruf als (**PROGN** [*expr* ...]) bekannt.

Beispiel: '(1 2 3 4)' liefert 4.

Siehe auch PROG1.

### 15.5.2 PROG1

Neben **PROGN** gibt es mit **PROG1** eine andere Möglichkeit, mehrere Ausdrücke zu errechnen.

```
(PROG1 [expr ...])
```

führt *expr* ... aus und liefert den Wert des ersten Ausdrucks (oder NIL, wenn kein Ausdruck angegeben wurde).

Beispiel: '(PROG1 1 2 3 4)' liefert 1.

Siehe auch PROGN.

### 15.5.3 LET

**LET** definiert einen neuen Block von lokalen Variablen. Dies ist nützlich, um z.B. lokale Variablen einer Funktion zu definieren. Der Aufbau ist:

```
(LET (varlist) expr ...)
```

wobei *varlist* eine Liste von lokalen Variablen ist.

```
varlist: varspec ...
```

```
varspec: (var expr) | var
```

Hier ist *var* der Name der Variable, der mit einem Kleinbuchstaben beginnt, gefolgt von weiteren Zeichen, Ziffern und Unterstrich-Zeichen. (siehe Abschnitt 15.3.4 [Name conventions], Seite 80).

---

<sup>13</sup> Anm.d.Übersetzers: Um Irritationen zu vermeiden: Der Befehl **PROGN** wird nicht(!) angegeben, sondern nur die Parameter; siehe auch das Beispiel!



Im Falle von `(var expr)` wird die neue Variable mit dem gegebenen Ausdruck initialisiert<sup>14</sup>. Im anderen Fall ist die neue Variable auf NIL gesetzt.

Es ist auch möglich, Typdeklarierer an die Variablen zu hängen (siehe Abschnitt 15.26 [Type specifiers], Seite 154).

Nach dem Initialisieren aller Variablen wird die Liste der Ausdrücke `expr ...` ausgewertet und der Wert der letzten zurückgegeben.

Der folgende LET-Ausdruck

```
(LET ((x 0) y (z (+ x 1)))
      (+ x z)
)
```

liefert zum Beispiel 1.

Siehe auch DOTIMES, DOLIST, DO, DEFVAR.

### 15.5.4 SETQ

Die Funktion SETQ setzt Werte in Variablen, Feldern und Tabellen.

```
(SETQ lvalue1 expr ...)
```

Setzt *lvalue1* auf den Wert von *expr*. Die Punkte zeigen weitere Zuweisungen für *lvalues* an. Ein *lvalue* ist eine Variable, ein Feld einer Tabelle oder eine Tabelle. Im Falle einer Variable muß diese vorher definiert worden sein (z.B. mit dem LET-Ausdruck).

Setzen des Wertes einer Tabelle bedeutet das Setzen seines Programm- oder Oberflächenzeigers: ‘(SETQ Table expr)’ setzt den Programm-Datensatzzeiger von *Table* auf den Wert von *expr* und ‘(SETQ Table\* expr)’ setzt dessen Oberflächen-Datensatzzeiger und aktualisiert die Anzeige. Mehr Informationen über Programm- und Oberflächen-Datensatzzeigern, siehe Abschnitt 5.2 [Tables], Seite 20.

SETQ liefert den Wert des letzten Ausdrucks.

Beispiel: ‘(SETQ a 1 b 2)’ weist 1 der Variable ‘a’ zu, 2 der Variable ‘b’ und liefert 2.

Siehe auch SETQ\*, LET, DEFVAR, Tables, Semantics of expressions.

### 15.5.5 SETQ\*

SETQ\* ist die Stern-Variante von SETQ (siehe Abschnitt 15.5.4 [SETQ], Seite 87) und hat ähnliche Auswirkungen. Der Unterschied ist, daß SETQ\* beim Zuweisen eines Wertes zu einer Variable

---

<sup>14</sup> Anm.d.Übersetzers: Die Klammern um `var expr` nicht vergessen!

die Auslösefunktion dieses Feldes aufruft (siehe Auslösefunktion Feld<sup>15</sup>), statt den Wert direkt zuzuweisen. Ist zum Feld keine Auslösefunktion zugewiesen, dann verhält sich **SETQ\*** genauso wie **SETQ** und weist einfach den Wert der Variable zu.

Beispiel: ‘(SETQ\* Table.Attr 0)’ ruft die Auslösefunktion von ‘Table.Attr’ mit dem Parameter 0 auf.

Achtung: Mit dieser Funktion ist es möglich, Endlosschleifen zu schreiben, z.B. wenn eine Auslösefunktion für ein Feld definiert wurde und diese Funktion **SETQ\*** aufruft, das sich selbst einen Wert setzt.

Siehe auch SETQ\*, LET, DEFVAR.

### 15.5.6 FUNCALL

**FUNCALL** wird verwendet, um eine Funktion aufzurufen.

```
(FUNCALL fun-expr [expr ...])
```

Ruft die Funktion *fun-expr* mit den gegebenen Parametern auf. Der Ausdruck *fun-expr* kann jeder Ausdruck sein, dessen Wert eine vor- oder benutzerdefinierte Funktion ist, z.B. eine Variable, die die Funktion enthält, die aufgerufen werden soll. Stimmt die Anzahl der Parameter nicht, dann wird eine Fehlermeldung erzeugt.

**FUNCALL** liefert den Rückgabewert des Funktionsaufrufes oder **NIL**, wenn *fun-expr* **NIL** ist.

Mehr Informationen über funktionelle Ausdrücke, siehe Abschnitt 15.25 [Functional parameters], Seite 153.

### 15.5.7 IF

**IF** ist ein Bedingungsoperator.

```
(IF expr1 expr2 [expr3])
```

Der Ausdruck *expr1* wird getestet. Wenn er nicht **NIL** liefert, dann wird der Wert von *expr2* geliefert, anderenfalls der von *expr3* (oder **NIL**, wenn nicht vorhanden).

Diese Funktion ist nicht strikt, das bedeutet, daß entweder der eine oder der andere Ausdruck ausgewertet wird.

Siehe auch **CASE**, **COND**.

---

<sup>15</sup> Anm.d.Übersetzers: Hier ist wegen Einschränkungen bei Texinfo leider kein Verweis möglich.

### 15.5.8 CASE

CASE ähnelt der `switch`-Anweisung in der Sprache C.

```
(CASE expr [case ...])
```

Hier ist *expr* der Auswahl Ausdruck und *case* ... sind Paare bestehend aus:

```
case: (value [expr ...])
```

wobei *value* ein einzelner Ausdruck oder eine Liste von Ausdrücken<sup>16</sup> ist und *expr* ... die Ausdrücke sind, die ausgeführt werden, wenn der Fallausdruck paßt.

Der Ausdruck CASE wertet erst *expr* aus. Dann wird jedes Fallpaar geprüft, ob es (oder einer der Ausdrücke in der Liste) zum ausgewerten Ausdruck paßt. Wird ein passender Fallausdruck gefunden, dann werden die dazugehörigen Ausdrücke ausgeführt und der Wert des letzten Ausdrucks zurückgeliefert. Paßt kein Fall, dann wird NIL zurückgeliefert.

Beispiel: '(CASE 1 ((2 3 4) 1) (1 2))' liefert 2.

Siehe auch IF, COND.

### 15.5.9 COND

COND ist wie IF ein Bedingungsoperator.

```
(COND [(test-expr [expr ...]) ...])
```

COND prüft der Reihe nach den ersten Ausdruck jeder Liste. Für den ersten, der nicht NIL liefert, wird der dazugehörige Ausdruck *expr* ... ausgeführt und der Wert des letzten Ausdrucks zurückgeliefert.

Liefern alle geprüften Ausdrücke NIL, dann wird NIL zurückgegeben.

#### Beispiel

```
(COND ((> 1 2) "1 > 2")
      ((= 1 2) "1 = 2")
      ((< 1 2) "1 < 2")
      )
```

liefert "1 < 2".

Siehe auch IF, CASE.

---

<sup>16</sup> Anm.d.Übersetzers: Diese Liste muß in Klammern eingeschlossen sein, siehe auch Beispiel.

### 15.5.10 DOTIMES

Für einfache Schleifen kann der Befehl `DOTIMES` verwendet werden.

```
(DOTIMES (name int-expr [result-expr ...]) [loop-expr ...])
```

Hier ist *name* der Name einer neuen Variable, die innerhalb der Schleife verwendet wird<sup>17</sup>. Der Name muß mit einem Kleinbuchstaben beginnen, gefolgt von weiteren Zeichen, Ziffern und Unterstrich-Zeichen. (siehe Abschnitt 15.3.4 [Name conventions], Seite 80).

Die Anzahl der Schleifendurchläufe wird über *int-expr* angegeben. In *result-expr ...* können Ausdrücke angegeben werden, die nach dem Beenden der Schleife ausgeführt werden sollen. *loop-expr* enthält den Körper der Schleife, darin sind die Ausdrücke, die bei jedem Schleifendurchlauf ausgewertet werden.

Bevor die Schleife ausgeführt wird, errechnet `DOTIMES` den Wert von *int-expr*, um die Anzahl festzustellen, wie oft die Schleife ausgeführt werden soll. Hier wird *int-expr* nur einmal zu Beginn der Schleife ausgewertet und muß einen Ganzzahlwert liefern. Danach setzt `DOTIMES` die Schleifenvariable schrittweise in jedem Durchlauf auf 0 bis *int-expr*-1. Zuerst wird die Variable auf Null gesetzt und geprüft, ob diese schon größer oder gleich dem Wert von *expr* ist. Ist *int-expr* negativ oder NIL, oder ist die Variable größer oder gleich dem Wert von *expr*, dann wird die Schleife abgebrochen und die Rückgabewerte ermittelt. Anderenfalls werden die Ausdrücke in der Schleife abgearbeitet und die Variable um eins erhöht. Danach kehrt die Schleife wieder zum Abbruchtest zurück und führt -wenn möglich- weitere Schleifendurchläufe aus.

Der Ausdruck `DOTIMES` liefert den Wert des letzten Rückgabewert-Ausdrucks oder NIL, wenn kein solcher angegeben wurde.

### Beispiel

```
(DOTIMES (i 50 i) (PRINT i))
```

Gibt die Nummern von 0 bis 49 aus und liefert den Wert 50.

Siehe auch `DOLIST`, `DO`, `FOR ALL`, `LET`.

### 15.5.11 DOLIST

Für Schleifen über Listen kann der Ausdruck `DOLIST` verwendet werden.

```
(DOLIST (name list-expr [result-expr ...]) [loop-expr ...])
```

Hier ist *name* der Name einer neuen Variable, der in der Schleife verwendet wird. Der Name muß mit einem Kleinbuchstaben beginnen, gefolgt von weiteren Zeichen, Ziffern und Unterstrich-Zeichen. (siehe Abschnitt 15.3.4 [Name conventions], Seite 80).

---

<sup>17</sup> Anm.d.Übersetzers: Diese Variable legt MUIbase selbsttätig an, man braucht sie also nicht selbst mit `LET` anlegen. Dies gilt für diese und nachfolgende Schleifenformen, die MUIbase bietet.

In *list-expr* wird die Liste festgelegt, über diese die Schleife ausgeführt werden soll, *result-expr* ... sind Ausdrücke, die nach dem Beenden der Schleife ausgeführt werden und *loop-expr* ... bilden den Körper der Schleife.

Bevor die Schleife ausgeführt wird, berechnet DOLIST den Wert von *list-expr*. Dieser Ausdruck wird nur einmal beim Start der Schleife ausgewertet und muß einen Listenwert liefern. Dann setzt DOTIMES bei jedem Schleifendurchlauf die Schleifenvariable der Reihe nach auf jedes Element der Liste. Zuerst wird die Schleifenvariable mit dem ersten Element der Liste initialisiert. Ist die Liste schon leer (NIL), dann wird die Schleife beendet und die Rückgabewerte berechnet. Anderenfalls werden die Schleifenausdrücke ausgeführt und die Variable auf das nächste Element in der Liste gesetzt. Danach kehrt die Schleife wieder zum Abbruchtest zurück und führt -wenn möglich- weitere Schleifendurchläufe aus.

Der Ausdruck DOLIST liefert den Wert des letzten Rückgabewert-Ausdrucks oder NIL, wenn kein solcher angegeben wurde.

## Beispiel

```
(DOLIST (i (SELECT * FROM Accounts)) (PRINT i))
```

Gibt alle Datensätze der Tabelle 'Accounts' aus und liefert NIL.

Siehe auch DOTIMES, DO, FOR ALL, LET.

### 15.5.12 DO

Mit dem Ausdruck DO können beliebige Schleifen programmiert werden.

```
(DO ([binding ...]) (term-expr [result-expr ...]) [loop-expr ...])
```

wobei *binding* ... die Variablenzuweisungen sind, die jeweils wie folgt aussehen können:

- ein neuer Name für eine Variable (der mit NIL vorbelegt wird)
- eine Liste der Form (*name init [step]*) mit *name* als Namen für eine neue Variable, *init* ist der Startwert der Variable und *step* der Ausdruck für Zehlschrittweite.

Des weiteren ist *term-expr* der Abbruchbedingungsauadruck, *result-expr* ... sind die Rückgabewertausrücke (voreingestellt ist NIL) und *loop-expr* ... bilden den Körper der Schleife.

Die DO-Schleife initialisiert zuerst alle lokalen Variablen mit den Startwerten und testet dann die Abbruchbedingung. Liefert sie TRUE, dann wird die Schleife unterbrochen und die Rückgabewertausrücke ausgeführt. Der Wert des letzten Rückgabewerts wird zurückgeliefert. Anderenfalls werden die Schleifenausdrücke (*loopexpr* ...) ausgeführt und jede Variable wird mit dem Wert der Schrittweite aktualisiert. Danach kehrt die Ausführung zum Test der Abbruchbedingung zurück und so weiter.

## Beispiel

```
(DO ((i 0 (+ i 1))) (>= i 5) i) (PRINT i))
```

Gibt die Werte 0, 1, 2, 4 und 4 aus und liefert den Wert 5. Natürlich ist das ein ziemlich komplizierter Weg, eine einfache FOR-Schleife zu bauen. Dafür gibt es mit dem Ausdruck `DOTIMES` eine einfachere Version.

Siehe auch `DOTIMES`, `DOLIST`, `FOR ALL`, `LET`.

### 15.5.13 FOR ALL

Der Ausdruck `FOR ALL` wird verwendet, um eine Liste von Datensätzen abzuarbeiten.

```
(FOR ALL table-list [WHERE where-expr] [ORDER BY order-list] DO expr ...)
```

Hier ist *table-list* eine durch Kommas getrennte Liste von Tabellen, *where-expr* ein Ausdruck zum Testen einer jeden Menge von Datensätzen, *order-list* eine durch Kommas getrennte Liste von Ausdrücken, nach denen sortiert werden soll und *expr* ... sind die Ausdrücke, die für jeden Datensatz ausgeführt werden sollen.

`FOR ALL` erzeugt zuerst eine Liste aller Datensätze, für die der Schleifenkörper ausgeführt werden soll. Dies wird wie bei einem `SELECT`-Ausdruck durchgeführt. Siehe Abschnitt 15.19.12 [`SELECT`], Seite 145 für mehr Informationen, wie diese Liste erzeugt wird. Für jedes Element dieser Liste wird der Schleifenkörper *expr* ... ausgeführt.

Zum Beispiel kann ein Aufsummieren eines Tabellenfeldes folgendermaßen durchgeführt werden:

```
(SETQ sum 0)
(FOR ALL Accounts DO
  (SETQ sum (+ sum Accounts.Amount))
)
```

Der Ausdruck `FOR ALL` liefert `NIL`.

Siehe auch `SELECT`, `DOTIMES`, `DOLIST`, `DO`.

### 15.5.14 NEXT

`NEXT` kann verwendet werden, um `DOTIMES`-, `DOLIST`-, `DO`- und `FOR ALL`-Schleifen zu steuern.

Ein Aufruf von `NEXT` im Schleifenkörper springt zum nächsten Schleifendurchlauf. Dies kann benutzt werden, wenn uninteressante Schleifendurchläufe übersprungen werden sollen, wie z.B. in folgendem Beispiel:

```
(FOR ALL Table DO
  (IF nicht-interessant-im-aktuellen-Datensatz (NEXT))
  ...
)
```

Siehe auch EXIT, DOTIMES, DOLIST, DO, FOR ALL.

### 15.5.15 EXIT

EXIT kann verwendet werden, um eine Schleife zu beenden.

```
(EXIT [expr ...])
```

EXIT innerhalb eines Schleifenkörpers beendet die Schleife und führen die optionalen Ausdrücke *expr* ... aus und liefern den Wert des letzten Ausdrucks (oder NIL, wenn kein Ausdruck angegeben wurde) als Rückgabewert der Schleife. Mögliche Rückgabewerte der Schleife im Beispiel

```
(DOTIMES (x 10 ret-expr ...) ...)
```

werden nicht ausgeführt.

Man kann die Funktion EXIT z.B. verwenden, um eine FOR ALL-Schleife zu beenden, wenn ein Datensatz gefunden wurde, der uns interessiert:

```
(FOR ALL Table DO
  (IF interessanter-aktueller-Datensatz (EXIT Table))
  ...
)
```

Siehe auch NEXT, RETURN, HALT, DOTIMES, DOLIST, DO, FOR ALL.

### 15.5.16 RETURN

Innerhalb einer Funktionsdefinition kann mit dem Befehl RETURN zur aufrufenden Funktion zurückgesprungen werden.

```
(RETURN [expr ...])
```

beendet die Funktion, führt die optionalen Ausdrücke *expr* ... aus und liefert den Wert des letzten Ausdrucks (oder NIL, wenn kein Ausdruck angegeben wurde).

### Beispiel

```
(DEFUN find-record (name)
  (FOR ALL Table DO
    (IF (= Name name) (RETURN Table))
  )
)
```

Das Beispiel sucht nach einem Datensatz, dessen Feld Name zum gegebenen Namen paßt. Die Funktion liefert den ersten Datensatz, der gefunden wurde oder NIL, wenn kein Datensatz gefunden wurde.

Siehe auch `HALT`, `EXIT`.

### 15.5.17 `HALT`

`HALT` kann verwendet werden, um die Programmausführung abubrechen.

`(HALT)`

stoppt stillschweigend die Programmausführung.

Siehe auch `ERROR`, `EXIT`, `RETURN`.

### 15.5.18 `ERROR`

Zum Abbrechen der Programmausführung mit einer Fehlermeldung kann die Funktion `ERROR` verwendet werden.

`(ERROR fmt [arg ...])`

stoppt die Programmausführung und öffnet ein Fenster mit einer Fehlermeldung. Die Fehlermeldung wird wie in der Funktion `SPRINTF` (siehe Abschnitt 15.12.28 [`SPRINTF`], Seite 114) aus *fmt* und den optionalen Parametern *arg* ... erzeugt.

Siehe auch `HALT`, `SPRINTF`.

## 15.6 Typaussagen

Für jeden Datentyp ist eine Aussage definiert, die `TRUE` liefert, wenn der übergebene Ausdruck vom gegebenen Typ ist, anderenfalls `NIL`. Die Aussagen sind:

Aussage	Beschreibung
<code>(STRP <i>expr</i>)</code>	<code>TRUE</code> wenn <i>expr</i> vom Typ Zeichenkette ist, sonst <code>NIL</code> .
<code>(MEMOP <i>expr</i>)</code>	<code>TRUE</code> wenn <i>expr</i> vom Typ mehrzeiliger Text ist, sonst <code>NIL</code> .
<code>(INTP <i>expr</i>)</code>	<code>TRUE</code> wenn <i>expr</i> vom Typ Ganzzahl ist, sonst <code>NIL</code> .
<code>(REALP <i>expr</i>)</code>	<code>TRUE</code> wenn <i>expr</i> vom Typ Fließkommazahl ist, sonst <code>NIL</code> .
<code>(DATEP <i>expr</i>)</code>	<code>TRUE</code> wenn <i>expr</i> vom Typ Datum ist, sonst <code>NIL</code> .
<code>(TIMEP <i>expr</i>)</code>	<code>TRUE</code> wenn <i>expr</i> vom Typ Zeit ist, sonst <code>NIL</code> .
<code>(NULLP <i>expr</i>)</code>	<code>TRUE</code> wenn <i>expr</i> vom Typ <code>NIL</code> (leere Liste) ist, sonst <code>NIL</code> .



(CONSP <i>expr</i> )	TRUE wenn <i>expr</i> keine leere Liste ist, sonst NIL.
(LISTP <i>expr</i> )	TRUE wenn <i>expr</i> eine Liste (die NIL sein kann) ist, sonst NIL.
(RECP <i>table expr</i> )	TRUE wenn <i>expr</i> ein Datensatzzeiger der gegebenen Tabelle ist. Wenn <i>expr</i> NIL ist, dann wird TRUE geliefert (Anfangsdatsatz). Wenn <i>table</i> NIL ist, dann wird geprüft, ob <i>expr</i> ein Datensatzzeiger irgendeiner Tabelle ist.

## 15.7 Typumwandlungsfunktionen

Dieser Abschnitt listet Funktionen auf, die zum Umwandeln von einem Datentyp in einen anderen verwendet werden.

### 15.7.1 STR

STR kann verwendet werden, um einen Ausdruck in eine Zeichenkettendarstellung umzuwandeln.

(STR *expr*)

wandelt *expr* in eine Zeichenkettendarstellung um. Der Typ von *expr* bestimmt die Umwandlung:

Typ	Ergebniszeichenkette
Zeichenkette	Die Zeichenkette selbst.
mehrzeiliger Text	Der gesamte mehrzeilige Text in einer Zeichenkette.
Ganzzahl	Zeichenkettendarstellung einer Ganzzahl.
Fließkommazahl	Zeichenkettendarstellung einer Fließkommazahl. Wenn <i>expr</i> ein Feld ist, dann wird die Anzahl der Nachkommastellen dieses Feldes, anderenfalls zwei Ziffern verwendet.
Auswahl	Auswahltext des Auswahlfeldes.
Datum	Zeichenkettendarstellung des Datumswertes.
Zeit	Zeichenkettendarstellung des Zeitwertes.
Bool	Die Zeichenkette "TRUE"
NIL	Benutzerdefinierte Zeichenkette für NIL, wenn <i>expr</i> ein Feld ist, anderenfalls die Zeichenkette "NIL"

Datensatz	Zeichenkettendarstellung der Datensatznummer.
Andere	Zeichenkettendarstellung des internen Adreßzeigers <sup>18</sup> .

Siehe auch MEMO, SPRINTF.

### 15.7.2 MEMO

MEMO kann verwendet werden, um einen Ausdruck in einen mehrzeiligen Text umzuwandeln.

(MEMO *expr*)

wandelt *expr* in eine mehrzeilige Textdarstellung. Es faßt den Ausdruck wie bei der Funktion STR (siehe Abschnitt 15.7.1 [STR], Seite 95) auf, liefert aber einen mehrzeiligen Text statt einer Zeichenkette.

Siehe auch STR.

### 15.7.3 INT

INT wird verwendet, um einen Ausdruck in eine Ganzzahl umzuwandeln.

(INT *expr*)

wandelt *expr* in eine Ganzzahl. Mögliche Umwandlungen sind:

Typ	Ergebniswert
Zeichenkette	Wenn die gesamte Zeichenkette eine Ganzzahl darstellt, dann wird die Zeichenkette in eine Ganzzahl umgewandelt. Führende und folgende Leerzeichen werden ignoriert. Stellt es keine Ganzzahl dar, wird NIL geliefert.
mehrzeiliger Text	Analog Zeichenkette.
Ganzzahl	Der Wert selbst.
Fließkommazahl	Wenn der Wert im Ganzzahlbereich liegt, wird der Fließkommazahlenwert gerundet und geliefert, anderenfalls NIL.
Auswahltext	Die interne Nummer (beginnend bei 0) des aktuellen Auswahltextes.

---

<sup>18</sup>

Anm.d.Übersetzers: Diese wird in hexadezimaler Schreibweise mit führendem "0x" angegeben

Datum	Anzahl Tage seit dem 01.01.0000.
Zeit	Anzahl Sekunden seit 00:00:00.
Datensatz	Datensatznummer.
NIL	NIL
Andere	Eine Fehlermeldung wird erzeugt und die Programmausführung abgebrochen.

Siehe auch REAL, ASC.

### 15.7.4 REAL

REAL wird verwendet, um einen Ausdruck in einen Wert vom Typ Fließkommazahl umzuwandeln.

`(REAL expr)`

wandelt *expr* in eine Fließkommazahl. Es faßt den Ausdruck wie bei der Funktion INT (siehe Abschnitt 15.7.3 [INT], Seite 96) auf, liefert aber einen Wert vom Typ Fließkommazahl anstatt einer Ganzzahl.

Siehe auch INT.

### 15.7.5 DATE

DATE wird verwendet, um einen Ausdruck in ein Datum umzuwandeln<sup>19</sup>.

`(DATE expr)`

wandelt den gegebenen Ausdruck in ein Datum<sup>20</sup>. Mögliche Umwandlungen sind:

Typ	Rückgabewert
Zeichenkette	Wenn die gesamte Zeichenkette einen Datumswert darstellt, dann wird die Zeichenkette in ein Datum umgewandelt. Führende und folgende Leerzeichen werden ignoriert.

<sup>19</sup> Anm.d.Übersetzers: Im Original lautet der Text "DATE is used to convert an expression into a date (with your best girl friend :-)." Der hintere Teil ist so zu verstehen, daß durch diese Funktion ein Date mit der besten Freundin ermöglicht würde. q.e.d., was zu beweisen wäre :-).

<sup>20</sup> Anm.d.Übersetzers: Hier hat Steffen wieder eine Zeile hineingeschmuggelt: "(only one date guys, needs two good girl friends for getting more dates!)". Übersetzt: "(nur ein Date, Jungs; es braucht schon zwei gute Freundinnen für weitere Dates!)" :-)

	Stellt es kein Datum dar, wird NIL geliefert.
mehrzeiliger Text	Analog Zeichenkette.
Ganzzahl	Ein Datumswert wird erzeugt, wenn die gegebene Ganzzahl die Anzahl der Tage seit dem 01.01.0000 darstellt. Ist die Ganzzahl zu groß (Datum würde größer als der 31.12.9999) oder negativ, dann wird NIL zurückgegeben.
Fließkommazahl	Analog Ganzzahl.
Datum	Der Wert selbst.
NIL	NIL
Andere	Eine Fehlermeldung wird erzeugt und die Programmausführung abgebrochen.

### 15.7.6 TIME

TIME wird verwendet, um einen Ausdruck in einen Zeitwert umzuwandeln.

(TIME *expr*)

wandelt den gegebenen Ausdruck in einen Zeitwert. Mögliche Umwandlungen sind:

Type	Return value
Zeichenkette	Wenn die gesamte Zeichenkette einen Zeitwert darstellt, dann wird die Zeichenkette in eine Zeit umgewandelt. Führende und folgende Leerzeichen werden ignoriert. Stellt es keine Zeit dar, wird NIL geliefert.
mehrzeiliger Text	Analog Zeichenkette.
Ganzzahl	Ein Zeitwert wird erzeugt, wenn die gegebene Ganzzahl die Anzahl der Sekunden seit 00:00:00 darstellt. Die Ganzzahl modulo der Anzahl der Sekunden pro Tag wird hergenommen, d.h. z.B. '(TIME 86400)' liefert 00:00:00.
Fließkommazahl	Analog Ganzzahl.
Zeit	Der Wert selbst.
NIL	NIL
Andere	Eine Fehlermeldung wird erzeugt und die Programmausführung abgebrochen.

## 15.8 Boolesche Funktionen

Dieser Abschnitt listet die booleschen Operatoren auf.

### 15.8.1 AND

AND prüft, ob alle seine Parameter TRUE sind.

`(AND [expr ...])`

prüft der Reihe nach *expr ...*, bis ein Ausdruck zu NIL wird. Sind alle Ausdrücke zu Nicht-NIL aufgelöst worden, dann wird der Wert des letzten Ausdrucks zurückgeliefert, anderenfalls NIL.

Diese Funktion ist nicht strikt, dies bedeutet, daß nicht alle Parameter von AND ausgewertet werden müssen, z.B. wird in `'(AND NIL (+ 1 2))'` der Ausdruck `'(+ 1 2)'` nicht ausgewertet, da ein NIL-Wert schon ermittelt wurde. In `'(AND (+ 1 2) NIL)'` jedoch wird der Ausdruck `'(+ 1 2)'` ausgewertet.

Siehe auch OR, NOT.

### 15.8.2 OR

OR prüft, ob alle seiner Parameter NIL sind.

`(OR [expr ...])`

prüft der Reihe nach *expr ...*, bis ein Ausdruck zu Nicht-NIL wird. Liefert den Wert des ersten Ausdrucks, der zu Nicht-NIL wurde, oder NIL, wenn alle Ausdrücke zu NIL wurden.

Diese Funktion ist nicht strikt, dies bedeutet, daß nicht alle Parameter von AND ausgewertet werden müssen, z.B. wird in `'(OR TRUE (+ 1 2))'` der Ausdruck `'(+ 1 2)'` nicht ausgewertet, da ein Nicht-NIL-Wert (hier TRUE) schon ermittelt wurde. In `'(OR (+ 1 2) TRUE)'` jedoch wird der Ausdruck `'(+ 1 2)'` ausgewertet.

Siehe auch AND, NOT.

### 15.8.3 NOT

NOT wird verwendet, um den Wert eines booleschen Ausdrucks zu invertieren.

`(NOT expr)`

liefert TRUE, wenn *expr* NIL ist, sonst NIL.

Siehe auch AND, OR.

## 15.9 Vergleichsfunktionen

In diesem Abschnitt findet man Funktionen zum Vergleich von Werten vor<sup>21</sup>.

### 15.10 Relationsoperatoren

Zum Vergleichen zweier Werte in einem MUIbase-Programm verwendet man

*(op expr1 expr2)*

wobei *op* einer aus {=, <>, <, >, >=, <=, =\*, <>\*, <\*, >\*, >=\*, <=\*} ist. Der Stern wird für besondere Vergleiche (Zeichenkettenvergleiche ohne Groß-/Kleinschreibung, Datensatzvergleich mit der benutzerdefinierten Reihenfolge) verwendet.

Die folgende Tabelle zeigt alle Regeln beim Vergleich von zwei Werten in einem MUIbase-Programm.

Typ	Vergleichsreihenfolge
Ganzzahl	NIL < MIN_INT < ... < -1 < 0 < 1 < ... < MAX_INT
Fließkommazahl	NIL < -HUGE_VAL < ... < -1.0 < 0.0 < 1.0 < ... < HUGE_VAL
Zeichenkette	NIL < "" < "Z" < "a" < "aa" < "b" < ... (mit groß/klein) NIL <* "" <* "a" <* "AA" <* "b" < ... (groß/klein egal)
mehrzeiliger Text	wie bei Zeichenketten
Datum	NIL < 1.1.0000 < ... < 31.12.9999
Zeit	NIL < 00:00:00 < ... < 23:59:59
Bool	NIL < TRUE
Datensatz	NIL < jeder_Datensatz (Datensätze selbst können nicht mit < verglichen werden) NIL <* rec1 <* rec2 (Reihenfolge festgelegt durch Benutzer)

---

<sup>21</sup> Anm.d.Übersetzers: Die einen finden Informationen, die anderen ein schlimmes Deutsch vor :-)

Anm.d.Übersetzers: HUGE\_VAL steht für den größtmöglichen Fließkommazahlenwert, den ein Prozessor handhaben kann. MIN\_INT ist die kleinste Ganzzahl und MAX\_INT ist die größte Ganzzahl (nein, man kehrt nicht nur das Vorzeichen um, um die 'andere' größte Zahl zu erhalten).

Siehe auch CMP, CMP\*, LIKE.

### 15.10.1 CMP

CMP liefert eine Ganzzahl, die eine Sortierung ihrer Argumente repräsentiert.

`(CMP expr1 expr2)`

liefert einen Wert kleiner als 0, wenn *expr1* kleiner ist als *expr2*; 0, wenn *expr1* gleich *expr2* ist und einen Wert größer 0, wenn *expr1* größer ist als *expr2*. Zum Erkennen der Sortierung wird die einfache (ohne Stern) Sortierrelation wie bei den Relationaloperatoren (siehe Abschnitt 15.10 [Relational operators], Seite 100) verwendet.

Es darf nicht angenommen werden, daß der Rückgabewert immer -1, 0 oder 1 ist!

Beispiel: `'(CMP "Bike" "bIKE")'` liefert -32.

Siehe auch CMP\*, Relational operators.

### 15.10.2 CMP\*

CMP\* ist die Stern-Version von CMP. Der Unterschied ist, daß CMP\* eine erweiterte Sortierung wie bei den Relationsoperatoren (siehe Abschnitt 15.10 [Relational operators], Seite 100) verwendet, bei denen Zeichenketten zeichengrößenunabhängig und Datensätze gemäß ihrer benutzerdefinierten Datensatzreihenfolge verglichen werden.

Beispiel: `'(CMP* "Bike" "bIKE")'` liefert 0.

Siehe auch CMP, Relational operators.

## 15.11 Mathematik-Funktionen

Einige mathematische Funktionen werden hier aufgelistet.

### 15.11.1 Werte addieren

Zum Zusammenzählen von Zahlen verwendet man

`(+ expr ...)`

Liefert die Summe der Parameter *expr* . . . . Wenn irgendein Parameter NIL ist, dann ist das Ergebnis NIL. Wenn die Werte vom Typ Fließkomma- oder Ganzzahl sind, dann ist das Ergebnis auch eine Fließkommazahl (oder Ganzzahl).

Es lassen sich auch Zeichenketten und mehrzeilige Texte 'addieren'. In diesem Fall ist das Ergebnis die zusammengehängten Zeichenketten bzw mehrzeiligen Texte.

Ist *expr* vom Typ Datum und der Rest der Parameter sind Ganz-/Fließkommazahlen, dann wird die Summe der Ganz-/Fließkommazahlen als Anzahl Tage aufgefaßt und zu *expr* addiert. Ist das zurückgelieferte Ergebnis außerhalb des gültigen Bereichs (vor dem 1.1.0000 oder nach dem 31.12.9999), dann ist das Ergebnis NIL.

Ist *expr* vom Typ Zeit und der Rest der Parameter sind Ganz-/Fließkommazahlen, dann wird die Summe der Ganz-/Fließkommazahlen als Anzahl Sekunden aufgefaßt und zu *expr* addiert. Es können jedoch in den restlichen Parametern Ausdrücke vom Typ Zeit vorhanden sein. Das zurückgelieferte Ergebnis wird modulo 24:00:00 genommen.

## Beispiele

Ausdruck	Wert
(+ 1 2 3)	6
(+ 5 1.0)	6.0
(+ "Hallo" " " "Welt!")	"Hallo Welt!"
(+ 28.11.1968 +365 -28 -9)	22.10.1969
(+ 07:30:00 3600)	08:30:00
(+ 03:00:00 23:59:59)	02:59:59

Siehe auch -, 1+, \*, CONCAT, CONCAT2.

### 15.11.2 Werte subtrahieren

Zum Subtrahieren von Werten verwendet man

(- *expr1* *expr2* . . .)

Zieht die Summe *expr2* . . . von *expr1* ab. Hier gelten die gleichen Regeln wie beim Addieren von Werten (siehe Abschnitt 15.11.1 [add], Seite 101), außer daß Zeichenketten und mehrzeilige Texte nicht subtrahiert werden können.

(- *expr*) hat eine spezielle Bedeutung: Es liefert den negativen Wert von *expr* (Ganz- oder Fließkommazahl), z.B. '(- (+ 1 2))' liefert -3.

Siehe auch +, 1-.



### 15.11.3 1+

1+ erhöht den Wert einer Ganz- bzw Fließkommazahl (-ausdrucks) um Eins.

(1+ *expr*)

Liefert den Wert von *expr* (Ganz- oder Fließkommazahl) plus Eins. Wenn *expr* NIL ist, dann wird NIL geliefert.

Siehe auch +, 1-.

### 15.11.4 1-

1- verringert den Wert einer Ganz- bzw Fließkommazahl (-ausdrucks) um Eins.

(1- *expr*)

Liefert den Wert von *expr* (Ganz- oder Fließkommazahl) minus Eins. Wenn *expr* NIL ist, dann wird NIL geliefert.

Siehe auch -, 1+.

### 15.11.5 Werte multiplizieren (\*)

Zum Multiplizieren von Ganz-/Fließkommazahlen verwendet man

(\* *expr* ...)

Liefert die Multiplikation der Ganz-/Fließkommazahlen *expr* .... Wenn alle Parameter Ganzzahlen sind, dann wird eine Ganzzahl geliefert, anderenfalls ist der Wert vom Typ Fließkommazahl.

Siehe auch +, /.

### 15.11.6 Werte dividieren

Zum Dividieren von Ganz-/Fließkommazahlen verwendet man

(/ *expr1* [*expr2* ...])

Teilt *expr1* durch die Multiplikation der restlichen Parameter. Liefert eine Fließkommazahl. Bei einer Division durch 0 wird NIL geliefert.

Siehe auch \*, DIV, MOD.

### 15.11.7 DIV

DIV verwendet man zur Ganzzahldivision.

(DIV *int1 int2*)

Liefert die Ganzzahldivision von *int1* mit *int2*. Zum Beispiel liefert '(DIV 5 3)' den Wert 1.

Siehe auch /, MOD.

### 15.11.8 MOD

MOD wird zur Modulo-Berechnung verwendet.

(MOD *int1 int2*)

Liefert *int1* modulo *int2*. Zum Beispiel liefert '(MOD 5 3)' den Wert 2.

Siehe auch DIV.

### 15.11.9 MAX

MAX liefert den Parameter mit dem größten Wert.

(MAX *expr ...*)

Liefert den größten Wert der Argumente *expr ...* (alles Ganz- oder Fließkommazahlen). Ist eine der Ausdrücke NIL, dann wird NIL geliefert.

Siehe auch MIN.

### 15.11.10 MIN

MIN liefert den Parameter mit dem kleinsten Wert.

(MIN *expr ...*)

Liefert den kleinsten Wert der Argumente *expr ...* (alles Ganz- oder Fließkommazahlen). Ist eine der Ausdrücke NIL, dann wird NIL geliefert.

Siehe auch MAX.

### 15.11.11 ABS

ABS berechnet den absoluten Wert eines Ausdrucks.

(ABS *expr*)

Liefert den absoluten Wert<sup>22</sup> von *expr* (Ganz- oder Fließkommazahl). Ist *expr* NIL, dann wird NIL geliefert.

### 15.11.12 TRUNC

TRUNC schneidet die Nachkommastellen einer Zahl ab.

(TRUNC *real*)

Liefert die größte Ganzzahl (als Fließkommazahl), die nicht größer ist als die angegebene Fließkommazahl. Ist *real* NIL, dann wird NIL geliefert.

Beispiele: '(TRUNC 26.1)' liefert 26, '(TRUNC -1.2)' liefert -2.

Siehe auch ROUND.

### 15.11.13 ROUND

ROUND rundet einen Fließkommawert.

(ROUND *real digits*)

Liefert den angegebenen Fließkommawert aufgerundet auf *digits* Stellen. Ist *real* oder *digits* gleich NIL, dann wird NIL geliefert.

Beispiele: '(ROUND 70.70859 2)' liefert 70.71, '(ROUND 392.36 -1)' liefert 392.0.

Siehe auch TRUNC.

### 15.11.14 RANDOM

RANDOM kann zum Generieren von Zufallszahlen verwendet werden.

(RANDOM *expr*)

Liefert eine Zufallszahl. Beim ersten Aufruf wird der Zufallszahlengenerator mit dem Wert aus der aktuellen Uhrzeit initialisiert. RANDOM erzeugt eine Zufallszahl im Bereich von 0 ... *expr*, ausgenommen *expr* selbst. Der Typ von *expr* (Ganz- oder Fließkommazahl) ist auch der Rückgabewert-Typ. Ist *expr* NIL, dann wird NIL geliefert.

---

<sup>22</sup> Anm.d.Übersetzers: Ein absoluter Wert ist immer positiv. Die Funktion wandelt einfach einen negativen Ausdruck in einen positiven und läßt einen positiven unverändert.

## Beispiele:

Beispiel	Bedeutung
(RANDOM 10)	liefert einen Wert von 0 bis 9,
(RANDOM 10.0)	liefert einen Wert von 0.0 bis 9.99999...

## 15.12 Zeichenkettenfunktionen

Dieser Abschnitt behandelt Funktionen für Zeichenketten.

### 15.12.1 LEN

LEN berechnet die Länge der Zeichenkette.

(LEN *str*)

Liefert die Länge der gegebenen Zeichenkette oder NIL wenn *str* NIL ist.

Siehe auch WORDS, LINES, MAXLEN.

### 15.12.2 LEFTSTR

LEFTSTR extrahiert eine Teilzeichenkette aus einer Zeichenkette.

(LEFTSTR *str len*)

Liefert den linken Teil einer gegebenen Zeichenkette mit höchstens *len* Zeichen. Ist *str* oder *len* gleich NIL oder ist *len* negativ, dann wird NIL geliefert.

Beispiel: '(LEFTSTR "Hallo Welt!" 5)' liefert "Hallo".

Siehe auch RIGHTSTR, MIDSTR, WORD, LINE.

### 15.12.3 RIGHTSTR

RIGHTSTR extrahiert eine Teilzeichenkette aus einer Zeichenkette.

(RIGHTSTR *str len*)

Liefert den rechten Teil einer gegebenen Zeichenkette mit höchstens *len* Zeichen. Ist *str* oder *len* gleich NIL oder ist *len* negativ, dann wird NIL geliefert.

Beispiel: ‘(RIGHTSTR "Hallo Welt!" 5)’ liefert "Welt!".

Siehe auch LEFTSTR, MIDSTR, WORD, LINE.

### 15.12.4 MIDSTR

MIDSTR extrahiert eine Teilzeichenkette aus einer Zeichenkette.

(MIDSTR *str pos len*)

Liefert einen Teil einer gegebenen Zeichenkette mit höchstens *len* Zeichen. Ist *len* NIL, dann ist die Anzahl der zurückgegebenen Zeichen nicht eingeschränkt<sup>23</sup>. Die Teilzeichenkette beginnt an der Stelle *pos* (beginnend bei Null). Wenn *str* oder *len* NIL sind oder wenn *len* negativ ist, dann wird NIL zurückgeliefert. Ist *pos* außerhalb des gültigen Bereichs (negativ oder größer als Zeichenkettenlänge), dann wird NIL geliefert.

Beispiel: ‘(MIDSTR "Hallo Welt!" 3 5)’ liefert "lo We".

Siehe auch LEFTSTR, RIGHTSTR, WORD, LINE, SETMIDSTR, INSMIDSTR.

### 15.12.5 SETMIDSTR

SETMIDSTR setzt eine Teilzeichenkette in einer Zeichenkette.

(SETMIDSTR *str index set*)

Liefert eine Kopie der Zeichenkette *str*, in dem die Zeichenkette beginnend bei *index* mit der Zeichenkette *set* überschrieben wird. Die Länge der zurückgelieferten Zeichenkette ist größer oder gleich der von *str*. Ist einer der Parameter NIL oder ist *index* außerhalb des gültigen Bereichs, dann wird NIL geliefert.

Beispiel: ‘(SETMIDSTR "Hallo Welt!" 6 "Melanie!")’ liefert "Hallo Melanie!".

Siehe auch INSMIDSTR, REPLACESTR.

### 15.12.6 INSMIDSTR

INSMIDSTR wird verwendet, um eine Teilzeichenkette in eine Zeichenkette einzufügen.

(INSMIDSTR *str index insert*)

Liefert eine Kopie der Zeichenkette *str*, in der die Zeichenkette *insert* an der gegebenen Stelle eingefügt wurde. Ist einer der Parameter NIL oder ist *index* außerhalb des gültigen Bereichs, dann wird NIL zurückgeliefert.

---

<sup>23</sup> Anm.d.Übersetzers: Sie kann natürlich nicht über das Ende der Ursprungszeichenkette gehen.

Beispiel: ‘(INSMIDSTR "Hallo Welt!" 6 "MUIbase-")’ liefert "Hallo MUIbase-Welt!".

Siehe auch SETMIDSTR, REPLACESTR.

### 15.12.7 INDEXSTR

INDEXSTR sucht in einer Zeichenkette nach dem ersten Vorkommen der Teilzeichenkette.

(INDEXSTR *str substr*)

Sucht nach dem ersten Vorkommen von *substr* in *str*. Der Zeichenkettenvergleich wird mit Beachtung der Groß-/Kleinschreibung durchgeführt. Liefert die Stelle (beginnend bei 0) von der Teilzeichenkette in *str* oder NIL, wenn die Teilzeichenkette nicht vorhanden ist. Ist eines der Argumente NIL, dann wird NIL zurückgegeben.

Beispiel: ‘(INDEXSTR "Hallo Welt!" "Welt")’ liefert 6.

Siehe auch INDEXSTR\*, RINDEXSTR, RINDEXSTR\*, INDEXBRK, INDEXBRK\*.

### 15.12.8 INDEXSTR\*

INDEXSTR\* hat den selben Effekt als INDEXSTR (siehe Abschnitt 15.12.7 [INDEXSTR], Seite 108), außer daß der Zeichenkettenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch INDEXSTR, RINDEXSTR, RINDEXSTR\*, INDEXBRK, INDEXBRK\*.

### 15.12.9 INDEXBRK

INDEXBRK sucht nach dem ersten Vorkommen eines Zeichens in einer Zeichenkette.

(INDEXBRK *str brkstr*)

Sucht nach dem ersten Vorkommen eines Zeichens von *brkstr*<sup>24</sup> in *str*. Der Zeichenkettenvergleich wird mit Beachtung der Groß-/Kleinschreibung durchgeführt. Liefert die Stelle (beginnend bei 0) des ersten Zeichens, das in *str* gefunden wurde, anderenfalls NIL. Ist eines der Parameter NIL, dann wird NIL geliefert.

Beispiel: ‘(INDEXBRK "Hallo Welt!" "aeiou")’ liefert 1.

Siehe auch INDEXBRK\*, RINDEXBRK, RINDEXBRK\*, INDEXSTR, INDEXSTR\*.

---

<sup>24</sup> Anm.d.Übersetzers: Hier ist gemeint, daß jedes Zeichen in *brkstr* einzeln zum Zeichenvergleich betrachtet wird. Intern wird jedes dieser Zeichen mit jedem Zeichen von *str* verglichen, bis eines paßt.

### 15.12.10 INDEXBRK\*

INDEXBRK\* hat den selben Effekt wie INDEXBRK (siehe Abschnitt 15.12.9 [INDEXBRK], Seite 108), außer daß der Zeichenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch INDEXBRK, RINDEXBRK, RINDEXBRK\*, INDEXSTR, INDEXSTR\*.

### 15.12.11 RINDEXSTR

RINDEXSTR sucht in einer Zeichenkette nach dem letzten Vorkommen der Teilzeichenkette.

(RINDEXSTR *str substr*)

Sucht nach dem letzten Vorkommen von *substr* in *str*. Der Zeichenkettenvergleich wird mit Beachtung der Groß-/Kleinschreibung durchgeführt. Liefert die Stelle (beginnend bei 0) von der Teilzeichenkette in *str* oder NIL, wenn die Teilzeichenkette nicht vorhanden ist. Ist eines der Argumente NIL, dann wird NIL zurückgegeben.

Beispiel: '(RINDEXSTR "Do itashimashite." "shi")' liefert 11<sup>25</sup>.

Siehe auch RINDEXSTR\*, INDEXSTR, INDEXSTR\*, RINDEXBRK, RINDEXBRK\*.

### 15.12.12 RINDEXSTR\*

RINDEXSTR\* hat den selben Effekt als RINDEXSTR (siehe Abschnitt 15.12.11 [RINDEXSTR], Seite 109), außer daß der Zeichenkettenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch RINDEXSTR, INDEXSTR, INDEXSTR\*, RINDEXBRK, RINDEXBRK\*.

### 15.12.13 RINDEXBRK

RINDEXBRK sucht nach dem letzten Vorkommen eines Zeichens in einer Zeichenkette.

(RINDEXBRK *str brkstr*)

Sucht nach dem letzten Vorkommen eines Zeichens von *brkstr*<sup>26</sup> in *str*. Der Zeichenkettenvergleich wird mit Beachtung der Groß-/Kleinschreibung durchgeführt. Liefert die Stelle (beginnend bei 0) des letzten Zeichens, das in *str* gefunden wurde, anderenfalls NIL. Ist eines der Parameter NIL, dann wird NIL geliefert.

Beispiel: '(RINDEXBRK "Konnichiwa" "chk")' liefert 6.

---

<sup>25</sup> Anm.d.Übersetzers: Oha, nu färbt des auch schon in die Dokumentation ab :-)

<sup>26</sup> Anm.d.Übersetzers: Hier ist gemeint, daß jedes Zeichen in *brkstr* einzeln zum Zeichenvergleich betrachtet wird. Intern wird jedes dieser Zeichen mit jedem Zeichen von *str* verglichen, bis eines paßt.

Siehe auch RINDEXBRK\*, INDEXBRK, INDEXBRK\*, RINDEXSTR, RINDEXSTR\*.

#### 15.12.14 RINDEXBRK\*

RINDEXBRK\* hat den selben Effekt wie RINDEXBRK (siehe Abschnitt 15.12.13 [RINDEXBRK], Seite 109), außer daß der Zeichenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch RINDEXBRK, INDEXBRK, INDEXBRK\*, RINDEXSTR, RINDEXSTR\*.

#### 15.12.15 REPLACESTR

REPLACESTR ersetzt Teilzeichenketten durch andere.

(REPLACESTR *str substr replacestr*)

Ersetzt alle Vorkommen von *substr* in *str* durch *replacestr*. Wenn eine der Zeichenketten NIL ist oder *substr* leer ist, dann wird NIL zurückgeliefert.

Beispiel: '(REPLACESTR "From Freiburg to San Francisco" "Fr" "X")' liefert "Xom Xeiburg to San Xancisco".

Siehe auch SETMIDSTR, INSMIDSTR, REMCHARS.

#### 15.12.16 REMCHARS

REMCHARS entfernt Zeichen aus einer Zeichenkette.

(REMCHARS *str chars-to-remove*)

Liefert eine Kopie von *str*, in der alle Zeichen von *chars-to-remove* entfernt wurden. Ist *str* oder *chars-to-remove* NIL, dann wird NIL geliefert.

Beispiel: '(REMCHARS *deine-zeichenkette* " \t\n")' entfernt alle Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen aus *deine-zeichenkette*.

Siehe auch REPLACESTR, TRIMSTR.

#### 15.12.17 TRIMSTR

TRIMSTR entfernt führende und abschließende Leerzeichen von einer Zeichenkette.

(TRIMSTR *str*)

Liefert eine Kopie von *str*, in der alle führenden und abschließenden Leerzeichen entfernt wurden.



Beispiel: `(TRIMSTR " Ich fuhr Selmas Fahrrad zu Schrott. ")` liefert `"Ich fuhr Selmas Fahrrad zu Schrott."`.

Siehe auch `REMCHARS`.

### 15.12.18 WORD

`WORD` liefert ein Wort einer Zeichenkette.

`(WORD str num)`

Liefert das *num*-te Wort (beginnend bei Null) aus der gegebenen Zeichenkette. Wörter in einer Zeichenkette sind nicht-leere Teilzeichenketten, die durch Leerzeichen-ähnliche Zeichen (z.B. Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen) getrennt werden.

Ist *str* oder *num* gleich `NIL` oder ist *num* außerhalb des gültigen Bereichs (negativ oder größer als Anzahl Wörter in der Zeichenkette), dann wird `NIL` zurückgeliefert.

Beispiel: `'(WORD "Deshalb lieb ich Selma mein Fahrrad." 3)'` liefert `"Selma"`.

Siehe auch `WORDS`, `LINE`, `LEFTSTR`, `RIGHTSTR`, `MIDSTR`.

### 15.12.19 WORDS

`WORDS` zählt die Anzahl der Wörter in einer Zeichenkette.

`(WORDS str)`

Liefert die Anzahl der Wörter in der gegebenen Zeichenkette oder `NIL`, wenn *str* `NIL` ist. Wörter sind Teilzeichenketten, die durch Leerzeichen-ähnliche Zeichen (z.B. Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen) getrennt werden.

Beispiel: `'(WORDS "In Wirklichkeit ist es aber nicht mein Fahrrad.")'` liefert `8`.

Siehe auch `WORD`, `LINES`, `LEN`.

### 15.12.20 CONCAT

`CONCAT` verbindet Zeichenketten.

`(CONCAT [str ...])`

Liefert die Verknüpfung der gegebenen Liste von Zeichenketten, wobei einzelne Leerzeichen zwischen den Zeichenketten eingefügt werden. Ist eine der Zeichenketten `NIL` oder die Liste leer, dann wird `NIL` zurückgeliefert.

Beispiele: ‘(CONCAT "Ich" "dachte," "es" "war" "ein "verlassenes" "Fahrrad.")’ liefert "Ich dachte, es war ein verlassenes Fahrrad."

Siehe auch CONCAT2, +, COPYSTR, SPRINTF.

### 15.12.21 CONCAT2

CONCAT2 verbindet Zeichenketten.

```
(CONCAT2 insert [str ...])
```

Liefert die Verknüpfung der gegebenen Liste von Zeichenketten. Zwischen den Zeichenketten wird jeweils die gegebene Zeichenkette *insert* eingefügt. Ist eine der Zeichenketten NIL oder die Liste leer, dann wird NIL zurückgeliefert.

Beispiel: ‘(CONCAT2 "! " "Aber" "es" "war es nicht!")’ liefert "Aber! es! war es nicht!".

Siehe auch CONCAT, +, COPYSTR, SPRINTF.

### 15.12.22 COPYSTR

COPYSTR erzeugt Kopien einer Zeichenkette.

```
(COPYSTR str num)
```

Liefert eine Zeichenkette, die *num* mal die Zeichenkette *str* enthält. Ist *str* NIL, *num* gleich NIL oder kleiner als NULL, dann wird NIL zurückgegeben.

Beispiel: ‘(COPYSTR "+-" 5)’ liefert "+-+-+-+--"

Siehe auch CONCAT, CONCAT2, +, SPRINTF.

### 15.12.23 UPPER

UPPER wandelt eine Zeichenkette in Großbuchstaben.

```
(UPPER str)
```

Liefert eine Kopie der gegebenen Zeichenkette, in der alle Zeichen zu Großbuchstaben umgewandelt wurden<sup>27</sup>. Ist *str* NIL, dann wird NIL geliefert.

Beispiel: ‘(UPPER "Selma fand einen Brief, der an mein Fahrrad geheftet war.")’ liefert "SELMA FAND EINEN BRIEF, DER AN MEIN FAHRRAD GEHEFTET WAR."

---

<sup>27</sup> Anm.d.Übersetzers: Natürlich nur die, bei denen es auch Sinn macht, d.h. Ziffern bleiben Ziffern usw.

Siehe auch LOWER.

### 15.12.24 LOWER

LOWER wandelt eine Zeichenkette in Kleinbuchstaben.

(LOWER *str*)

Liefert eine Kopie der gegebenen Zeichenkette, in der alle Zeichen zu Kleinbuchstaben umgewandelt wurden. Ist *str* NIL, dann wird NIL geliefert.

Beispiel: '(LOWER "Der Brief war von Silke.")' liefert "der brief war von silke".

Siehe auch UPPER.

### 15.12.25 ASC

ASC wandelt ein Zeichen in ihren ASCII-Code.

(ASC *str*)

Liefert den ASCII-Code des ersten Zeichens von *str*. Ist *str* leer, wird 0 geliefert. Ist *str* NIL, dann wird NIL geliefert.

Beispiel: (ASC "A") liefert 65.

Siehe auch CHR, INT.

### 15.12.26 CHR

CHR wandelt einen ASCII-Code in ein Zeichen um.

(CHR *int*)

Liefert eine Zeichenkette, die das Zeichen mit dem ASCII-Code *int* enthält. Ist *int* gleich 0, dann wird eine leere Zeichenkette geliefert. Ist *int* NIL oder nicht inner halb des ASCII-Zeichensatzes (0..255), dann wird NIL geliefert.

Beispiel: '(CHR 99)' liefert to "c".

Siehe auch ASC, STR.

### 15.12.27 LIKE

LIKE vergleicht Zeichenketten.

(LIKE *str1 str2*)

Liefert TRUE, wenn *str1* mit *str2* übereinstimmt, anderenfalls NIL. Die Zeichenkette *str2* kann die Jokerzeichen '?' und '\*' enthalten, wobei '?' genau irgendein einzelnes Zeichen und '\*' eine Zeichenkette jeder Länge irgendeines Inhalts<sup>28</sup> darstellt. Der Zeichenkettenvergleich wird ohne Beachtung der Groß-/Kleinschreibung durchgeführt.

Beispiel: '(LIKE "Silke war für ein Jahr in Frankreich." "\*Jahr\*")' liefert TRUE.

Siehe auch Comparison functions.

### 15.12.28 SPRINTF

SPRINTF formatiert eine Zeichenkette mit verschiedenen Daten.

(SPRINTF *fmt* [*expr* ...])

SPRINTF erhält eine Reihe von Parametern, die in Zeichenketten umgewandelt werden und in aufbereiteter Form als einzelne Zeichenkette zurückgegeben wird. Die Zeichenkette *fmt* entscheidet genau, was in die zurückgegebene Zeichenkette geschrieben werden soll und kann zwei Arten von Elementen enthalten: ordinäre Zeichen, die unverändert kopiert werden und Umwandlungsbefehle, die SPRINTF anweisen, die Parameter aus seiner Parameterliste zu nehmen und zu formatieren. Umwandlungsbefehle beginnen immer mit dem Zeichen '%'.

Umwandlungsbefehle benötigen immer diese Form:

%[*flags*] [*width*] [.*precision*] *type*

where

- Das optionale Feld *flags* steuert die Ausrichtung der Ausgabe, das Vorzeichen von numerischen Werten, Dezimalpunkte und führende Leerzeichen.
- Das optionale Feld *width* legt die minimale Anzahl von Zeichen fest, die ausgegeben werden sollen (die Feldbreite), gegebenenfalls wird mit Leerzeichen oder Nullen aufgefüllt.
- Das optionale Feld *precision* legt entweder die minimale Anzahl von auszugebenden Ziffern für die Typen Ganzzahl, Zeichenkette, Bool, Datum und Zeit oder die Anzahl der Zeichen nach dem Dezimalpunkt zur Ausgabe eines Fließkommawertes fest.
- Das Feld *type* legt den gewünschten Type des Parameters fest, den SPRINTF umwandeln soll, wie etwa Zeichenkette, Ganzzahl, Fließkommazahl etc.

Zu beachten ist, daß alle Felder außer *type* optional sind. Die folgenden Tabellen listen die gültigen Optionen für diese Felder auf.

<sup>28</sup> Anm.d.Übersetzers: Eine Zeichenkette der Länge 0 ist auch zulässig, d.h. (LIKE "Automat" "\*Auto\*") liefert TRUE, weil der vordere '\*' eine leere Zeichenkette darstellt und der hintere die Zeichenkette "mat".

## Flaggenfeld *flags*

- : Das Ergebnis ist linksbündig, das rechts mit Leerzeichen aufgefüllt wird. Normalerweise wird das Feld rechtsbündig ausgerichtet und links mit Leerzeichen oder '0'en aufgefüllt, wenn kein '-' angegeben wird.
- +: Das Ergebnis erhält immer ein Zeichen '-' oder '+' vorangestellt, wenn es eine numerische Umwandlung ist.

### *Leerzeichen:*

Positive Zahlen erhalten ein Leerzeichen anstatt dem Zeichen '+', aber negative Zahlen bekommen nach wie vor das Zeichen '-' vorangestellt.

## Breitenfeld *width*

- n*: Mindestens *n* Zeichen werden ausgegeben. Hat die Umwandlung weniger als *n* Zeichen ergeben, dann wird mit Leerzeichen aufgefüllt.
- \*: Der Breite-Parameter wird in der Parameterliste als Ganz- oder Fließkommazahl vor dem eigentlichen Umwandlungsparameter mitgeliefert. Der Wert ist beschränkt auf 0 bis 999.

## Genauigkeitsfeld *precision*

- .n*: Für Ganzzahl-, Zeichenketten-, Bool-, Datums- und Zeit-Werte ist *n* die Anzahl der auszugebenden Zeichen vom umgewandelten Element. Für Umwandlungen von Fließkommazahlen legt *n* die Anzahl der Nachkommastellen fest.
- .\**: Die Genauigkeit wird in der Parameterliste als Ganz- oder Fließkommazahl vor dem eigentlichen Umwandlungsparameter mitgeliefert. Der Wert ist beschränkt auf 0 bis 999.

(Anm.d.Übersetzers: Bitte nicht den davorstehenden Punkt übersehen!)

## Typenfeld *type*

- b: Wandelt einen booleschen Parameter nach "TRUE" (wahr) oder "NIL" (falsch).
- i: Wandelt eine Ganzzahl um.
- e: Wandelt eine Fließkommazahl in das Format '[-]d.ddde+dd' um. Genau eine Ziffer erscheint vor dem Dezimalpunkt, gefolgt von Nachkommastellen, einem 'e' und dem Exponenten. Die Anzahl der Nachkommastellen wird im Genauigkeitsfeld festgelegt oder wenn nicht, dann ist sie 2. Der Dezimalpunkt erscheint nicht, wenn sie 0 ist.
- f: Wandelt eine Fließkommazahl in das Format '[-]ddd.ddd' um. Die Anzahl der Nachkommastellen wird im Genauigkeitsfeld festgelegt oder wenn nicht, dann ist sie 2. Der Dezimalpunkt erscheint nicht, wenn sie 0 ist.

- s:           Schreibt eine Zeichenkette bis zum Ende der Zeichenkette oder soviele Zeichen, wie im Präzisionsfeld angegeben.
- d:           Wandelt einen Datumswert um.
- t:           Wandelt einen Zeitwert um.
- %:          Nur das Zeichen '%' wird geschrieben und kein Parameter umgewandelt.

SPRINTF liefert die formatierte Zeichenkette oder NIL, wenn *fmt* NIL ist.

## Beispiele

Aufruf	Ergebnis
(SPRINTF "Hallo")	"Hallo"
(SPRINTF "%s" "Hallo")	"Hallo"
(SPRINTF "%10s" "Hallo")	"       Hallo"
(SPRINTF "%-10.10s" "Hallo")	"Hallo       "
(SPRINTF "%010.3s" "Hallo")	"        Hal"
(SPRINTF "%-5.3b" TRUE)	"TRU   "
(SPRINTF "%i" 3)	"3"
(SPRINTF "%03i" 3)	"003"
(SPRINTF "% 0+- 5.3i" 3)	" 003 "
(SPRINTF "%f" 12)	"12.00"
(SPRINTF "%10e 12.0)	"    1.20e+1"
(SPRINTF "%+-10.4f" 12.0)	"+12.0000 "
(SPRINTF "%10.5t" 12:30:00)	"       12:30"
(SPRINTF "%d" 28.11.1968)	"28.11.1968"
(SPRINTF "Ha%s %4.4s!" "llo"	
"Weltmeisterschaft")	"Hallo Welt!"

Siehe auch PRINTF, FPRINTF, STR, +, CONCAT, CONCAT2, COPYSTR.

## 15.13 Funktionen für mehrzeilige Texte

Dieser Abschnitt behandelt Funktionen für mehrzeilige Texte.

### 15.13.1 LINE

LINE holt eine Zeile aus einem mehrzeiligen Text.

(LINE *memo num*)

Liefert die *num*-te Zeile (beginnend bei Null) aus dem gegebenen mehrzeiligen Text. Die Zeile hat dann kein abschließendes Neue-Zeile-Zeichen.

Ist *str* oder *num* gleich NIL oder ist *num* außerhalb des gültigen Bereichs (negativ oder größer als Anzahl Zeilen), dann wird NIL zurückgeliefert.

Siehe auch LINES, WORD.

### 15.13.2 LINES

LINES liefert die Anzahl der Zeilen in einem mehrzeiligen Text.

(LINES *memo*)

Liefert die Anzahl der Zeilen des gegebenen mehrzeiligen Textes oder NIL, wenn *memo* NIL ist.

Siehe auch LINE, WORDS, LEN.

### 15.13.3 MEMOTOLIST

MEMOTOLIST wandelt einen mehrzeiligen Text in eine Liste von Zeichenketten um.

(MEMOTOLIST *memo*)

Wandelt den gegebenen mehrzeiligen Text in eine Liste. Ist *memo* gleich NIL, dann wird NIL geliefert, anderenfalls wird eine Liste erzeugt, in der jedes Element eine Zeile des mehrzeiligen Textes enthält.

Beispiel: '(MEMOTOLIST "Meine Versicherung\nzahlt für\ndas kaputte Fahrrad.")' liefert ( "Meine Versicherung" "zahlt für" "das kaputte Fahrrad." ).

Siehe auch LISTTOMEMO.

### 15.13.4 LISTTOMEMO

LISTTOMEMO wandelt eine Liste in einen mehrzeiligen Text um.

(LISTTOMEMO *list*)

Wandelt eine gegebene Liste in einen mehrzeiligen Text. Ist *list* gleich NIL, dann wird NIL zurückgegeben, anderenfalls wird ein mehrzeiliger Text erzeugt, dessen einzelne Zeilen die Zeichenkettendarstellung des entsprechenden Listenelements enthalten<sup>29</sup>.

Beispiel: '(LISTTOMEMO (LIST "Silke" "leiht mir" "'mein' Fahrrad" "bis zum" 01.09.1998))' liefert: "Silke\nleiht mir\n'mein' Fahrrad\nbis zum\n01.09.1998".

---

<sup>29</sup> Anm.d.Übersetzers: Alle Listenelemente, die keine Zeichenketten oder mehrzeilige Texte sind, werden zuerst in ihre Zeichenkettendarstellung umgewandelt, bevor sie in den mehrzeiligen Text eingefügt werden.

Siehe auch MEMOTOLIST.

### 15.13.5 FILLMEMO

FILLMEMO füllt einen mehrzeiligen Text mit den Ergebnissen von Ausdrücken.

(FILLMEMO *memo*)

Erzeugt eine Kopie des gegebenen mehrzeiligen Textes, in dem alle Teilzeichenketten der Form '\$(expr)' durch ihre Ergebnisse nach der Auswertung ersetzt werden.

Beispiel: '(FILLMEMO "(+ 1 1) ist \$(+ 1 1).")' liefert "(+ 1 1) ist 2."

Man sollte nur kleine Ausdrücke in einem mehrzeiligen Text verwenden, da die Fehlersuche nicht einfach ist<sup>30</sup>.

Siehe auch FORMATMEMO, INDENTMEMO.

### 15.13.6 FORMATMEMO

FORMATMEMO formatiert einen mehrzeiligen Text.

(FORMATMEMO *memo width fill*)

Formatiert *memo* in einen mehrzeiligen Text mit Zeilen, die nicht länger sind als *width* Zeichen. Ist *fill* nicht NIL, dann werden Leerzeichen zum Auffüllen der Zeilen verwendet, damit die Zeilen genau die Länge *width* erhalten. Der mehrzeilige Text wird abschnittsweise abgearbeitet. Ein Abschnitt beginnt beim ersten Zeichen, das kein Leerzeichen ist. Die Zeile mit diesem Zeichen mit den nachfolgenden Zeilen bis zur Zeile, in der das erste Zeichen ein Leerzeichen ist, wird als Abschnitt aufgefaßt. Der gesamte Abschnitt wird wortweise formatiert, dies bedeutet, es werden so viele Wörter in eine Zeile gesetzt, wie dafür Platz ist<sup>31</sup>.

Siehe auch FILLMEMO, INDENTMEMO.

### 15.13.7 INDENTMEMO

INDENTMEMO rückt einen mehrzeiligen Text ein, in dem links Leerzeichen eingefügt werden.

(INDENTMEMO *memo indent*)

---

<sup>30</sup> Anm.d.Übersetzers: Man kann hier schon komplexe Ausdrücke hinschreiben, nur wird man nicht mit Fehlermeldungen und Positionsangaben unterstützt, falls der Ausdruck falsch sein sollte. Mein Tip ist der, daß man alles in Form einer Funktion in ein Programm auslagert und dann nur noch die Funktion aufruft. In diesem Fall reagiert die Programmausführung und kann den Fehler bestimmen und lokalisieren.

<sup>31</sup> Anm.d.Übersetzers: Die restlichen Wörter werden dann natürlich in der darauffolgenden Zeile und ggf. weiteren untergebracht.



Liefert eine Kopie des gegebenen mehrzeiligen Textes, in dem jede Zeile mit *indent* Leerzeichen eingerückt wird. Ist *memo* oder *indent* NIL, dann wird NIL zurückgeliefert. Ist *indent* negativ, dann wird 0 angenommen.

Siehe auch FILLMEMO, FORMATMEMO.

## 15.14 Listenfunktionen

Dieser Abschnitt listet Funktionen zum Verarbeiten von Listen auf.

### 15.14.1 CONS

CONS erzeugt ein Paar von Ausdrücken.

(CONS *elem list*)

Erzeugt eine neue Liste. Das erste Element der neuen Liste ist *elem*, der Rest sind die Elemente von *list* (die eine Liste sein muß oder NIL). Die Liste *list* wird nicht kopiert, sondern nur ein Zeiger auf diese wird verwendet<sup>32</sup>!

Beispiel: '(CONS 1 (CONS 2 NIL))' liefert ( 1 2 ).

Die Elemente der Liste können von jedem Typ sein, z.B. ist es auch möglich, eine Liste von Listen zu haben (z.B. siehe Abschnitt 15.19.12 [SELECT], Seite 145). Der Konstruktor CONS kann auch verwendet werden, um Element-Paare zu erzeugen, z.B. '(CONS 1 2)' ist ein Paar mit den Ganzzahlen 1 und 2.

Siehe auch LIST, FIRST, REST.

### 15.14.2 LIST

LIST erzeugt eine Liste anhand ihrer Parameter.

(LIST [*elem ...*])

nimmt die Parameter *elem ...* und generiert daraus eine Liste. Dies ist gleichbedeutend dem Aufruf von (CONS *elem* (CONS ... NIL)). Man beachte, daß NIL alleine für eine leere Liste steht.

Siehe auch CONS, LENGTH.

---

<sup>32</sup> Anm.d.Übersetzers: Dies hat zur Konsequenz, daß wenn man *list* löscht, daß in der neuen Liste ebenso die Elemente fehlen! Möchte man also die Elemente behalten, muß man sich eine Kopierfunktion schreiben!

### 15.14.3 LENGTH

LENGTH ermittelt die Länge einer Liste.

(LENGTH *list*)

liefert die Länge der gegebenen Liste.

Beispiel: ‘(LENGTH (LIST "a" 2 42 3))’ liefert 4.

Siehe auch LIST.

### 15.14.4 FIRST

FIRST holt das erste Element aus einer Liste.

(FIRST *list*)

liefert das erste Element der gegebenen Liste. Ist *list* leer (NIL), dann wird NIL geliefert.

Siehe auch REST, LAST, NTH, CONS.

### 15.14.5 REST

REST liefert die Teilliste nach dem ersten Element einer Liste.

(REST *list*)

liefert den Rest der gegebenen Liste (die Liste ohne dem ersten Element). Ist *list* leer (NIL), dann wird NIL zurückgeliefert.

Beispiel: ‘(REST (LIST 1 2 3))’ liefert ( 2 3 ).

Siehe auch FIRST, CONS.

### 15.14.6 LAST

LAST holt das letzte Element aus einer Liste.

(LAST *list*)

Liefert das letzte Element der gegebenen Liste oder NIL, wenn *list* NIL ist.

Siehe auch FIRST, NTH.

### 15.14.7 NTH

NTH holt das  $n$ -te Element aus einer Liste.

```
(NTH  $n$  list)
```

Liefert das  $n$ -te Element der gegebenen Liste (beginnend bei 0) oder NIL, wenn das Element nicht existiert.

Siehe auch FIRST, LAST.

### 15.14.8 APPEND

APPEND verbindet Listen.

```
(APPEND [list ...])
```

liefert die Verknüpfung von *list* ....

Beispiel: ‘(APPEND (list 1 2) (list 3 4) (list 5))’ liefert ( 1 2 3 4 5 ).

Siehe auch LIST.

### 15.14.9 REVERSE

REVERSE kehrt eine Liste um.

```
(REVERSE list)
```

liefert die umgekehrte Liste<sup>33</sup>.

Beispiel: ‘(REVERSE (list 1 2 3))’ liefert ( 3 2 1 ).

### 15.14.10 MAPFIRST

MAPFIRST wendet eine Funktion auf alle Listenelemente an.

```
(MAPFIRST func list [...])
```

Erzeugt eine Liste, deren Elemente das Ergebnis einer angegebenen Funktion sind, die als Parameter die einzelnen Listenelemente der Reihe nach bekommen hat. Die Länge der zurückgelieferten Liste ist genau so lang, wie die längste angegebene Liste. Ist eine der gegebenen Listen zu kurz, dann wird die Liste mit NIL aufgefüllt.

---

<sup>33</sup> Anm.d.Übersetzers: Hier wird die Reihenfolge der Listenelemente umgekehrt, d.h. das erste Element der Ursprungsliste wird zum letzten, das zweite zum vorletzten etc.

## Beispiele

Ausdruck	Wert
(MAPFIRST 1+ (LIST 1 2 3))	( 2 3 4 )
(MAPFIRST + (LIST 1 2 3) (LIST 2 3))	( 3 5 NIL )

### 15.14.11 SORTLIST

SORTLIST sortiert die Elemente einer Liste.

```
(SORTLIST func list)
```

Liefert eine Kopie der gegebenen Liste, die mit der Funktion *func* sortiert wurde. Die Sortierfunktion muß zwei Parameter für jedes Element verarbeiten und einen Ganzzahlwert liefern, der kleiner als Null ist, wenn das erste Element 'kleiner' ist als das zweite, einen Wert größer Null, wenn das zweite 'größer' ist als das erste und einen Wert gleich Null, wenn beide Elemente 'gleich' sind.<sup>34</sup>

Beispiel für eine Zeichenkettenvergleichsfunktion für die Sortierung:

```
(DEFUN cmp_str (x y)
  (COND
    ((< x y) -1)
    (> x y) 1)
  (TRUE 0)
)
```

Nun läßt sich eine Liste durch den Aufruf

```
(SORTLIST cmp_str (LIST "hi" "gut" "großartig" "ok"))
```

sortieren, die ( "großartig" "gut" "hi" "ok" ) liefert.

Siehe auch SORTLISTGT, MAPFIRST.

### 15.14.12 SORTLISTGT

SORTLISTGT sortiert die Elemente einer Liste.

```
(SORTLISTGT gtfunc list)
```

---

<sup>34</sup> Anm.d.Übersetzers: Es ist nicht zwingend, daß ein positiver Wert geliefert wird, wenn das zweite Element größer ist als das erste. Man kann dies auch umkehren, dann wird die Liste absteigend statt aufsteigend sortiert.

Arbeitet wie `SORTLIST`, aber hier wird eine Sortierfunktion angegeben, die einen Wert ungleich `NIL` liefert, wenn das erste Element 'größer' ist als das zweite, anderenfalls `NIL`.

Beispiel: `'(SORTLISTGT > (LIST "hi" "gut" "großartig" "ok"))'` liefert `( "großartig" "gut" "hi" "ok" )`.

Siehe auch `SORTLIST`, `MAPFIRST`.

## 15.15 Benutzereingabefunktionen

Zum Abfragen von Benutzereingaben können folgende Funktionen verwendet werden:

### 15.15.1 ASKFILE

`ASKFILE` fragt den Benutzer nach einem Dateinamen.

`(ASKFILE title oktext default savemode)`

Öffnet ein Dateiauswahlfenster zur Eingabe eines Dateinamens. Der Fenstertitel kann in *title*, der Text des 'Ok'-Knopfes in *oktext* und der vorgegebene Dateiname in *default* gesetzt werden. Für jeden dieser Argumente kann `NIL` gesetzt werden, um Vorgabewerte zu verwenden. Der letzte Parameter *savemode* (bool) setzt das Dateiauswahlfenster in den Speichermodus. Dieser Modus sollte verwendet werden, wenn nach einem Dateinamen gefragt wird, im etwas in eine Datei zu schreiben.

`ASKFILE` liefert den eingegebenen Dateinamen als Zeichenkette oder `NIL`, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Siehe auch `ASKDIR`, `ASKSTR`.

### 15.15.2 ASKDIR

`ASKDIR` fragt den Benutzer nach einem Verzeichnisnamen.

`(ASKDIR title oktext default savemode)`

Öffnet ein Dateiauswahlfenster zur Eingabe eines Verzeichnisnamens. Die Parameter werden auf die gleiche Weise verwendet als wie<sup>35</sup> in `ASKFILE` (siehe Abschnitt 15.15.1 [ASKFILE], Seite 123).

`ASKDIR` liefert den eingegebenen Verzeichnisnamen als Zeichenkette oder `NIL`, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Siehe auch `ASKFILE`, `ASKSTR`.

---

<sup>35</sup> Anm.d.Übersetzers: 'als wie' ist eine Universallösung für alle Dialekte; die einen benutzen nur 'wie', die anderen nur 'als' und wer weiß, vielleicht gibt es sogar welche, die es überhaupt nicht benutzen? :-)

### 15.15.3 ASKSTR

ASKSTR fragt den Benutzer nach einer Zeichenkette.

(ASKSTR *title oktext default maxlen*)

Öffnet ein Fenster, das nach einer Zeichenketteneingabe fragt. Der Fenstertitel, der Text des ‘Ok’-Knopfes und der Vorgabewert können mit *title*, *oktext* beziehungsweise *default* (mit Zeichenketten oder NIL für die Vorgabewerte) gesetzt werden. *maxlen* bestimmt die maximale Anzahl Zeichen, die der Benutzer eingeben kann.

ASKSTR liefert die eingegebene Zeichenkette oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Siehe auch ASKFILE, ASKDIR, ASKCHOICESTR, ASKINT.

### 15.15.4 ASKINT

ASKINT fragt den Benutzer nach einer Ganzzahl.

(ASKINT *title oktext default min max*)

Öffnet ein Eingabefenster, das nach einer Ganzzahleingabe fragt. Der Fenstertitel und der Text des ‘Ok’-Knopfes können mit *title* und *oktext* (mit Zeichenketten oder NIL für Vorgabewerte) gesetzt werden. In *default* wird der Vorgabewert übergeben oder NIL, wenn mit einem leeren Feld begonnen werden soll. In *min* und *max* wird der erlaubte Zahlenbereich festgelegt. Eingegebene Werte außerhalb dieses Bereichs werden vom Eingabefenster nicht akzeptiert. Man verwende NIL für den Vorgabebereich<sup>36</sup>.

ASKINT liefert die eingegebene Ganzzahl oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Siehe auch ASKSTR.

### 15.15.5 ASKCHOICE

ASKCHOICE fragt den Benutzer nach einer Auswahl aus mehreren Elementen.

(ASKCHOICE *title oktext choices default*)

Öffnet ein Eingabefenster, das den Benutzer nach einem Element aus mehreren fragt. Der Fenstertitel und der Text des ‘Ok’-Knopfes können mit *title* und *oktext* (mit Zeichenketten oder NIL für Vorgabewerte) gesetzt werden. In *choices* wird eine Liste angegeben, aus dieser sich der Benutzer einen Eintrag auswählen kann. Diese Listeneinträge können alle Arten von Ausdrücken haben, die in Zeichenketten umgewandelt werden können. Die vorgegebene Auswahl wird in *default* angegeben,

---

<sup>36</sup> Anm.d.Übersetzers: Dies bedeutet, daß jede Ganzzahl möglich ist.

die ein Ganzzahlexindex auf die Auswahlliste (beginnend bei Index 0 für das erste Element) ist und NIL markiert keine vorgegebene Auswahl.

ASKCHOICE liefert den Index des gewählten Elements oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

## Beispiel

```
(LET ((items (LIST "Erster Eintrag" 2 3.14 "Letzter Eintrag"))) index)
  (SETQ index (ASKCHOICE "Wähle ein Element" "Ok" items NIL))
  (IF index
    (PRINTF "Benutzer wählte Element Nummer %i mit dem Inhalt <%s>\n"
      index (STR (NTH index items))
    )
  )
)
```

Siehe auch ASKCHOICESTR, ASKOPTIONS.

### 15.15.6 ASKCHOICESTR

ASKCHOICESTR fragt den Benutzer nach einer Zeichenkette und bietet vorgegebene an.

```
(ASKCHOICESTR title oktext strings default)
```

Öffnet ein Eingabefenster, das dem Benutzer erlaubt, eine Zeichenkette aus mehreren auszuwählen oder jede beliebige Zeichenkette im separaten Zeichenkettenfeld einzugeben. Der Fenstertitel und der Text des 'Ok'-Knopfes können mit *title* und *oktext* (mit Zeichenketten oder NIL für Vorgabewerte) gesetzt werden. In *strings* wird eine Liste von Zeichenketten übergeben, aus der der Benutzer eine auswählen kann. Der Vorgabewert des Zeichenkettenfeldes kann mit *default* (als Zeichenkette oder NIL für ein leeres Zeichenkettenfeld) gesetzt werden.

ASKCHOICESTR liefert die ausgewählte Zeichenkette oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

## Beispiel

```
(LET ((strings (LIST "Claudia" "Mats" "Ralphie"))) likebest)
  (SETQ likebest
    (ASKCHOICESTR "Wen mögen Sie am liebsten?" "Ok" strings "Meine Collie-Hunde")
  )
  (IF likebest (PRINTF "Benutzer wählte <%s>\n" likebest))
)
```

Siehe auch ASKCHOICE, ASKOPTIONS.

### 15.15.7 ASKOPTIONS

ASKOPTIONS fragt den Benutzer nach mehreren aus einer Liste von Elementen.

(ASKOPTIONS *title oktext options selected*)

Öffnet ein Eingabefenster, das dem Benutzer erlaubt, mehrere Optionen aus vielen auszuwählen. Der Fenstertitel und der Text des 'Ok'-Knopfes können mit *title* und *oktext* (mit Zeichenketten oder NIL für Vorgabewerte) gesetzt werden. In *options* wird eine Liste von Optionen angegeben, aus der der Benutzer mehrere auswählen kann. Die Listenelemente können jeder Art von Ausdrücken sein, die sich in Zeichenketten umwandeln lassen. Die Vorgabeauswahl läßt sich in *selected* festlegen, die eine Liste von Ganzzahlen ist und jeweils den Index des entsprechenden Elements in der Liste *options* angibt, der vorab ausgewählt wird. Wird NIL statt der Liste angegeben, dann werden keine Elemente vorab ausgewählt.

ASKOPTIONS liefert eine Liste von Ganzzahlen, die den Index der ausgewählten Elemente enthält oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen oder kein Element ausgewählt hat.

#### Beispiel

```
(LET ((options (LIST "Salva Mea" "Insomnia" "Don't leave" "7 days & 1 week"))
      (selected (LIST 0 1 3))
      )
  (SETQ selected (ASKOPTIONS "Wähle Musiktitel" "Ok" options selected))
  (IF selected
    (
      (PRINTF "Benutzer wählte folgende Einträge:\n")
      (DOLIST (i selected)
        (PRINTF "\tNummer %i enthält: <%s>\n" i (STR (NTH i options)))
      )
    )
  )
)
```

(Anm.d.Übersetzers: Hier hören Programmierer und Übersetzer die gleiche Musik von Faithless :-)

### 15.15.8 ASKBUTTON

ASKBUTTON fragt den Benutzer nach einen Knopfdruck.

(ASKBUTTON *title text buttons canceltext*)

Öffnet ein Eingabefenster mit dem gegebenen Fenstertitel *title* (als Zeichenkette oder NIL für einen Vorgabetitel) und dem gegebenen Beschreibungstext *text* (als Zeichenkette oder NIL für keinen Text). Die Funktion wartet auf einen Druck der in *buttons* (als Liste von Zeichenketten) festgelegten Knöpfe oder des 'Abbrechen'-Knopfes. Der Text des Abbruchknopfes läßt sich mit



*cancelttext* ändern. Wird hier NIL angegeben, dann wird ein Vorgabetext verwendet, der sich nach der Anzahl der festgelegten Knöpfe richtet.

ASKBUTTON liefert die Nummer des gedrückten Knopfes (beginnend bei 0 mit dem am weitesten links angeordneten Knopf) oder NIL, wenn der Benutzer den 'Abbruch'-Knopf gedrückt hat.

## Beispiele

```
(LET ((buttons (LIST "Zuhause" "Im Bett" "Vor meinem Amiga"))) index)
  (SETQ index (ASKBUTTON "Bitte beantworten:"
    "Wo werden sie morgen sein?" buttons "Weiß nicht")
  )
  (IF index
    (PRINTF "Benutzer entschied sich für: <%s>\n" (NTH index buttons))
  )
)

(ASKBUTTON "Info"
  "Du hast nun schon für fünf\nStunden mit deinem Amiga gespielt.\nGeh ins Bett!"
  NIL NIL
)
```

Siehe auch ASKCHOICE.

### 15.15.9 ASKMULTI

ASKMULTI fragt den Benutzer nach verschiedenartigen Informationen.

```
(ASKMULTI title oktext itemlist)
```

ASKMULTI ist ein Mehrzweck-Eingabefenster. Es öffnet ein Fenster mit dem angegebenen Titel *title*, einem Satz von grafischen Objekten für die Dateneingabe und zwei Knöpfen ('Ok' und 'Abbrechen') zum Beenden des Eingabefensters. Der Text für den 'Ok'-Knopf kann mit *oktext* verändert werden (als Zeichenkette oder NIL für einen Vorgabetext). Der Satz der grafischen Objekte werden in *itemlist* festgelegt, das eine Liste ein Elementen ist, in der jedes eine der folgenden Formate hat:

(LIST <i>title</i> "String" <i>initial</i> [ <i>help</i> ])	zum Bearbeiten einer Textzeile,
(LIST <i>title</i> "Memo" <i>initial</i> [ <i>help</i> ])	zum Bearbeiten von mehrzeiligen Texten,
(LIST <i>title</i> "Integer" <i>initial</i> [ <i>help</i> ])	zum Bearbeiten einer Ganzzahl,
(LIST <i>title</i> "Real" <i>initial</i> [ <i>help</i> ])	zum Bearbeiten einer Fließkommazahl,
(LIST <i>title</i> "Date" <i>initial</i> [ <i>help</i> ])	zum Bearbeiten eines Datums,
(LIST <i>title</i> "Time" <i>initial</i> [ <i>help</i> ])	zum Bearbeiten einer Zeit,
(LIST <i>title</i> "Bool" <i>initial</i> [ <i>help</i> ])	zum Bearbeiten eines booleschen Wertes,
(LIST <i>title</i> "Choice" <i>initial</i> (LIST <i>choice</i> ...) [ <i>help</i> ])	für ein Auswahlfeld.
(LIST <i>title</i> "ChoiceList" <i>initial</i>	

<pre>         (LIST choice ...) [help]       )       (LIST title "Options" initial         (LIST choice ...) [help]       )       non-list-expr </pre>	<pre>         zum Auswählen eines Elements         aus einer Liste.          zum Auswählen mehrerer Elemente         aus einer Liste.          für statischen Text </pre>
--	---

Der Titel *title* (als Zeichenkette oder NIL für keinen Titel) wird links neben dem grafischen Objekt angeordnet. Ist der Vorgabewert *initial* NIL, dann wird ein Vorgabewert verwendet (z.B. ein leeres Textfeld). Für Auswahlfelder muß der Vorgabewert der Index (beginnend bei 0) sein, für Auswahllistenfelder darf der Vorgabewert NIL (kein Eintrag ausgewählt) sein und für Optionsfelder muß der Vorgabewert eine Liste von Ganzzahlen sein, die die Indexe (beginnend bei 0) der Elemente sein, die vorgelegt sein sollen. Das optionale Hilfsfeld (eine Zeichenkette) kann verwendet werden, um dem Benutzer mehr Informationen über die Verwendung des Feldes mitzugeben.

ASKMULTI liefert eine Liste von Werten, die der Benutzer bearbeitet und über den 'Ok'-Knopf bestätigt hat. Jeder Ergebniswert eines Feldes hat das gleiche Format wie der für den Vorgabewert, z.B. für ein Auswahllistenfeld ist der Rückgabewert der Index des ausgewählten Elements (oder NIL, wenn keines ausgewählt wurde) oder für Optionsfelder ist er die Liste von Ganzzahlen, die die Indexe der ausgewählten Elemente darstellen. Für statischen Text wird NIL zurückgegeben.

Wurde z.B. ein Datumsfeld, ein statischer Text, ein Auswahlfeld, ein Optionsfeld und ein Zeichenkettenfeld mit dem Vorgabewert "Welt" festgelegt und der Benutzer gab 11.11.1999 ein, wählte den Auswahleintrag mit dem Index 2, wählte das dritte und vierte Element des Optionsfeldes und ließ das Zeichenkettenfeld unberührt, dann liefert die Funktion die Liste ( 11.11.1999 NIL 2 ( 3 4 ) "world" ).

Brach der Benutzer das Eingabefenster ab, wird NIL geliefert.

## Beispiel

```

(ASKMULTI "Bitte bearbeiten:" NIL (LIST
  (LIST "_Name" "String" "")
  (LIST "_Geburtstag" "Date" NIL)
  (LIST "Ge_schlecht" "Choice" 0 (LIST "männlich" "weiblich"))
  (LIST "_Hat ein Auto?" "Bool" NIL)
  (LIST "_Mag", "Options" (LIST 0 2)
    (LIST "Bier" "Wein" "Whisky" "Wodka" "Schnaps"))
  ))
)
```

Man sehe sich auch das Projekt 'AskDemo.mb' für weitere Beispiele an.

## 15.16 E/A-Funktionen

Dieser Abschnitt listet die Funktionen und Variablen zur Dateiein- und ausgabe (z.B. drucken) auf.

### 15.16.1 FOPEN

FOPEN öffnet eine Datei zum Lesen/Schreiben.

(FOPEN *filename mode*)

Öffnet eine Datei mit dem Dateinamen *filename* (als Zeichenkette). Das Argument *mode* (als Zeichenkette) steuert den Zugriff auf die Datei. Mit `"w"` wird die Datei zum Schreiben geöffnet, mit `"a"` zum Anfügen an die bestehende Datei und mit `"r"` zum Lesen aus einer Datei. Es sind auch andere Zeichen (oder Kombinationen von ihnen) möglich, wie z.B. `"r+"` zum Lesen und Schreiben. Es gibt keine Überprüfung, ob die angegebenen Modi gültig sind. Es wird jedoch NIL zurückgeliefert, wenn die Datei nicht geöffnet werden konnte.

FOPEN liefert bei Erfolg einen Dateihandler. Schlug er fehl, wird NIL geliefert. Sind *filename* oder *mode* NIL, dann wird NIL zurückgeliefert.

Beispiel: `(FOPEN "PRT:" "w")` öffnet und liefert einen Dateihandle zum Drucker.

Siehe auch FCLOSE, stdout, FFLUSH.

### 15.16.2 FCLOSE

FCLOSE schließt eine Datei.

(FCLOSE *file*)

Schließt gegebene Datei. Es liefert 0 bei Erfolg oder NIL, wenn ein Fehler auftrat. Ist *file* NIL, dann wird 0 geliefert (keine Fehler)! Der Zugriff auf eine Datei nach dem Schließen einer Datei ist eine illegale Operation und führt zum Abbruch der Programmausführung mit einer Fehlermeldung.

Siehe auch FOPEN, FFLUSH.

### 15.16.3 stdout

Die globale Variable `stdout` trägt den Dateihandle zur Standardausgabe von MUIbase. Der Ausgabedateinamen kann im Menüpunkt `Programm - Ausgabedatei...` gesetzt werden. Man kann `'PRT:'` verwenden, um die Ausgabe zum Drucken zu schicken oder `'CON:////Output/CLOSE/WAIT'` für die Ausgabe in ein (neues) Shellfenster.

Beim ersten Zugriff auf diese Variable (entweder direkt, z.B. durch den Aufruf von `(FPRINTF stdout ...)` oder indirekt, z.B. durch den Aufruf von `(PRINTF ...)`) wird die Datei geöffnet.

Die Datei wird nicht vor der Programmausführung geöffnet. Dies verhindert das Öffnen der Datei, wenn keine Ausgabe erzeugt wird, z.b. wenn einfach nur Berechnungen und Änderungen an einigen Datensätzen durchgeführt werden.

Wenn MUIbase die Programmausgabedatei nicht öffnen kann, dann wird die Ausführung unterbrochen und eine Fehlermeldung ausgegeben.

Siehe auch FOPEN, PRINTF.

### 15.16.4 PRINT

PRINT wandelt einen Ausdruck in eine Zeichenkette und gibt ihn aus.

(PRINT *elem*)

Wandelt den Wert von *elem* in eine lesbare Zeichenkette und gibt ihn über `stdout` aus. Diese Funktion ist hauptsächlich zu Prüfzwecken vorhanden.

Siehe auch PRINTF, stdout.

### 15.16.5 PRINTF

PRINTF gibt eine formatierte Zeichenkette aus.

(PRINTF *format* [*expr* ...])

Formatiert eine Zeichenkette aus der gegebenen Formatzeichenkette und seinen Parameter und gibt sie an `stdout` aus. Die Formatierung entspricht der von `SPRINTF` (siehe Abschnitt 15.12.28 [SPRINTF], Seite 114).

PRINTF liefert die Anzahl der ausgegebenen Zeichen oder NIL bei einem Fehler. Ist *format* NIL, dann wird NIL geliefert.

Beispiel: '(PRINTF "%i Tage und %i Woche" 7 1)' gibt die Zeichenkette "7 Tage und 1 Woche" nach `stdout` aus und liefert 18.

Siehe auch PRINT, FPRINTF, stdout.

### 15.16.6 FPRINTF

FPRINTF gibt eine formatierte Zeichenkette in eine Datei aus.

(FPRINTF *file* *format* [*expr* ...])

Formatiert eine Zeichenkette aus der gegebenen Formatzeichenkette und seinen Parameter und gibt sie in die angegebene Datei aus. Die Formatierung entspricht der von `SPRINTF` (siehe Abschnitt 15.12.28 [`SPRINTF`], Seite 114).

`PRINTF` liefert die Anzahl der ausgegebenen Zeichen oder `NIL` bei einem Fehler. Ist *file* `NIL`, dann liefert `FPRINTF` dennoch die Anzahl der potentiell geschriebenen Zeichen zurück, macht aber keine Ausgabe. Ist *format* `NIL`, dann wird `NIL` geliefert.

Siehe auch `PRINTF`, `FOPEN`.

### 15.16.7 FERROR

`FERROR` prüft, ob ein Ein-/Ausgabefehler einer Datei aufgetreten ist.

(`FERROR file`)

liefert `TRUE`, wenn ein Fehler bei der gegebenen Datei auftrat, anderenfalls `NIL`. Ist *'file'* `NIL`, wird `NIL` geliefert.

Siehe auch `FEOF`, `FOPEN`, `FCLOSE`.

### 15.16.8 FEOF

`FEOF` prüft auf den Endestatus einer Datei.

(`FEOF file`)

Überprüft den Dateende-Indikator der gegebenen Datei und liefert `TRUE`, wenn er gesetzt ist, anderenfalls `NIL`. Ist *'file'* `NIL`, wird `NIL` geliefert.

Siehe auch `FERROR`, `FTELL`, `FOPEN`, `FCLOSE`.

### 15.16.9 FSEEK

`FSEEK` setzt die Schreib-/Leseposition in einer Datei.

(`FSEEK file offset whence`)

Setzt die Schreib-/Leseposition für die gegebene Datei. Die neue Position -gemessen in Bytes- wird erreicht durch das Hinzufügen von *offset* bytes bezogen auf die Position, die durch *whence* festgelegt wird. Ist *whence* auf `SEEK_SET`, `SEEK_CUR` oder `SEEK_END` gesetzt, dann ist *offset* relativ zum Beginn der Datei, der aktuellen Position beziehungsweise zum Ende der Datei.

Bei Erfolg liefert `FSEEK` 0, anderenfalls `NIL` und die Dateiposition bleibt unverändert. Ist *file*, *offset* oder *whence* `NIL`, oder ist *whence* nicht eine der Konstanten `SEEK_SET`, `SEEK_CUR` oder `SEEK_END`, dann wird `NIL` geliefert.

Siehe auch FTELL, Pre-defined constants.

### 15.16.10 FTELL

FTELL liefert die Schreib-/Leseposition der Datei.

(FTELL *file*)

Ermittelt die aktuelle Schreib-/Leseposition relativ zum Anfang der gegebenen Datei und liefert sie als Ganzzahl. Tritt ein Fehler auf oder ist '*file*' NIL, dann wird NIL geliefert.

Siehe auch FSEEK, FEOF.

### 15.16.11 FGETCHAR

FGETCHAR liest ein Zeichen aus einer Datei.

(FGETCHAR *file*)

Liefert das nächste Zeichen von der gegebenen Datei als Zeichenkette oder NIL, wenn *file* NIL ist, das Ende der Datei erreicht wurde oder ein Fehler auftrat. Ist das nächste Zeichen ein Nullbyte, dann wird eine leere Zeichenkette geliefert.

Siehe auch FGETCHARS, FGETSTR, FPUTCHAR.

### 15.16.12 FGETCHARS

FGETCHARS liest Zeichen aus einer Datei.

(FGETCHARS *num file*)

liefert eine Zeichenkette, die die nächsten *num* Zeichen aus der gegebenen Datei enthält. Ist das Ende der Datei erreicht worden, bevor *num* Zeichen gelesen werden konnten, oder wenn ein Nullbyte gelesen wurde, dann werden nur die bisher gelesenen Zeichen zurückgegeben. Ist *num* oder *file* NIL, *num* negativ, das Ende der Datei erreicht worden, bevor das erste Zeichen gelesen wurde, oder ein Lesefehler aufgetreten, dann wird NIL zurückgeliefert.

Siehe auch FGETCHAR, FGETSTR.

### 15.16.13 FGETSTR

FGETSTR liest eine Zeichenkette aus einer Datei.

(FGETSTR *file*)

liefert die nächste Zeile aus der gegebenen Datei oder NIL, falls *file* NIL ist, das Ende der Datei erreicht wurde oder ein Fehler auftrat. Das Ende einer Zeile wird entweder durch ein Neue-Zeile-Zeichen oder durch ein Nullbyte gekennzeichnet oder falls das Ende der Datei erkannt wurde. In jedem Fall enthält die Zeichenkette keine Neue-Zeile-Zeichen.

Siehe auch FGETCHAR, FGETCHARS, FGETMEMO, FPUTSTR.

#### 15.16.14 FGETMEMO

FGETMEMO liest einen mehrzeiligen Text aus einer Datei.

(FGETMEMO *file*)

liefert einen mehrzeiligen Text, der den Inhalt der gegebenen Datei bis zum nächsten Nullbyte oder zum Ende der Datei enthält. Ist *file* NIL, das Ende der Datei erreicht worden, bevor ein Zeichen gelesen wurde oder trat ein Fehler auf, dann wird NIL zurückgeliefert.

Siehe auch FGETSTR, FPUTMEMO.

#### 15.16.15 FPUTCHAR

FPUTCHAR schreibt ein Zeichen in eine Datei.

(FPUTCHAR *str file*)

Schreibt das erste Zeichen von *str* in die gegebene Datei. Ist *str* leer, dann wird ein Nullbyte geschrieben. Sind *str* oder *file* NIL, dann passiert nichts. Liefert *str* oder NIL, wenn ein Ausgabefehler auftrat.

Siehe auch FPUTSTR, FGETCHAR.

#### 15.16.16 FPUTSTR

FPUTSTR schreibt eine Zeichenkette in eine Datei.

(FPUTSTR *str file*)

Gibt *str* zusammen mit einem Neue-Zeile-Zeichen in die gegebene Datei aus. Sind *str* oder *file* NIL, dann passiert nichts. Liefert *str* oder NIL, wenn ein Ausgabefehler auftrat.

Siehe auch FPUTCHAR, FPUTMEMO, FGETSTR.

#### 15.16.17 FPUTMEMO

FPUTMEMO schreibt einen mehrzeiligen Text in eine Datei.

(FPUTMEMO *memo file*)

Gibt *memo* in die gegebene Datei aus. Sind *memo* oder *file* NIL, dann passiert nichts. Liefert *memo* oder NIL, wenn ein Ausgabefehler auftrat.

Siehe auch FPUTSTR, FGETMEMO.

### 15.16.18 FFLUSH

FFLUSH leert den Schreibpuffer in eine Datei.

(FFLUSH *file*)

Schreibt den Schreibpuffer<sup>37</sup> der gegebenen Datei. Liefert 0 bei Erfolg, NIL bei einem Fehler. Ist *file* NIL, dann wird 0 zurückgegeben (kein Fehler).

Siehe auch FOPEN, FCLOSE.

## 15.17 Datensatzfunktionen

Dieser Abschnitt behandelt Funktionen für Datensätze.

### 15.17.1 NEW

NEW legt einen neuen Datensatz für eine Tabelle an.

(NEW *table init*)

Legt einen neuen Datensatz für die gegebene Tabelle an. Das Argument *init* legt den Datensatz fest, der zum Einrichten des neuen Datensatzes verwendet werden soll. Ein Wert von NIL steht für den Vorgabedatensatz.

NEW liefert den Datensatzzeiger für den neuen Datensatz.

Die Funktion NEW hat zudem den Nebeneffekt, daß der Programm-Datensatzzeiger der gegebenen Tabelle (siehe Abschnitt 5.2 [Tables], Seite 20) auf den neuen Datensatz gesetzt wird.

Beispiel: '(NEW table NIL)' legt einen neuen Datensatz in der gegebenen Tabelle an und richtet ihn mit dem Vorgabedatensatz ein.

---

<sup>37</sup> Anm.d.Übersetzers: Jede Ein- und Ausgabe von einem Medium wird gepuffert, um die Geschwindigkeit zu erhöhen. Erst dann wenn der Puffer voll ist, wird beim Lesen der nächste Datenblock aus der Datei gelesen bzw. beim Schreiben der Schreibpuffer in die Datei geschrieben. Insbesondere beim Schreiben in ein Konsolenfenster (also CON:) wird man dann den Effekt von unvollständigen Zeilen bemerken, da der Rest der Zeile noch im Puffer steht. In der Regel schreibt ein Neue-Zeile-Zeichen den Puffer in die Ausgabedatei, aber um es explizit zu schreiben, wird (FFLUSH) verwendet.



Siehe auch `NEW*`, `DELETE`, `Tables`.

### 15.17.2 `NEW*`

`NEW*` ist die Version von `NEW` (siehe Abschnitt 15.17.1 [`NEW`], Seite 134) mit dem Stern.

`(NEW* table init)`

`NEW*` prüft, ob eine Auslösefunktion für die gegebene Tabelle (siehe Auslösefunktion `Neu`<sup>38</sup>) definiert wurde. Ist eine vorhanden, dann wird diese zum Anlegen des Datensatzes ausgeführt und dessen Ergebnis zurückgeliefert. Das Argument `init` gibt den Datensatz an, anhand dessen der neue Datensatz initialisiert werden soll (`NIL` für den Vorgabedatensatz).

Wurde keine Auslösefunktion eingerichtet, dann verhält sich die Funktion wie `NEW`.

Achtung: Mit dieser Funktion ist es möglich, Endlosschleifen zu schreiben, wenn z.B. für eine Tabelle eine Auslösefunktion für `'New'` definiert wurde und diese Funktion `NEW*` aufruft, um den Datensatz anzulegen.

Siehe auch `NEW`, `DELETE*`.

### 15.17.3 `DELETE`

`DELETE` löscht einen Datensatz einer Tabelle.

`(DELETE table requester)`

Löscht den aktuellen Programm-Datensatz der gegebenen Tabelle, nachdem ein optionales Löschenfenster bestätigt wurde. Das erste Argument definiert die Tabelle, für die der aktuelle Programm-Datensatz gelöscht werden soll und das zweite ist ein boolescher Ausdruck. Ist dieser `NIL`, dann wird der Datensatz stillschweigend gelöscht, anderenfalls wird der Status des Menüpunkts `'Datensätze löschen bestätigen?'` geprüft. Ist dieser nicht gesetzt, dann wird der Datensatz auch stillschweigend gelöscht, anderenfalls erscheint das Löschenbestätigungsfenster, das bestätigt werden muß. Bricht der Benutzer die Löschenfunktion ab, dann wird der Datensatz nicht gelöscht.

Der Rückgabewert der Funktion `DELETE` widerspiegelt die ausgewählte Aktion. Liefert sie `TRUE`, dann ist der Datensatz gelöscht worden, anderenfalls `NIL` (wenn der Benutzer die Funktion unterbrochen hat).

Beim Löschen setzt `DELETE` den Programm-Datensatzzeiger (siehe Abschnitt 5.2 [`Tables`], Seite 20) der gegebenen Tabelle auf `NIL`.

Beispiel: `'(DELETE table NIL)'` löscht stillschweigend den aktuellen Datensatz der gegebenen Tabelle.

---

<sup>38</sup> Anm.d.Übersetzers: Hier ist wegen Einschränkungen bei Texinfo leider kein Verweis möglich.

Siehe auch DELETE\*, DELETEALL, NEW, Tables.

#### 15.17.4 DELETE\*

DELETE\* ist die Version von DELETE (siehe Abschnitt 15.17.3 [DELETE], Seite 135) mit dem Stern.

(DELETE\* *table requester*)

DELETE\* prüft, ob eine Auslösefunktion für die gegebene Tabelle (siehe Auslösefunktion Löschen<sup>39</sup>) definiert wurde. Ist eine vorhanden, dann wird diese zum Löschen des Datensatzes ausgeführt und dessen Ergebnis zurückgeliefert. Das Argument *requester* gibt an, ob die Auslösefunktion ein Bestätigungsfenster öffnen soll, bevor der Datensatz gelöscht wird.

Wurde keine Auslösefunktion eingerichtet, dann verhält sich die Funktion wie DELETE.

Achtung: Mit dieser Funktion ist es möglich, Endlosschleifen zu schreiben, wenn z.B. für eine Tabelle eine Auslösefunktion für 'Delete' definiert wurde und diese Funktion DELETE\* aufruft, um den Datensatz zu löschen.

Siehe auch DELETE, DELETEALL, NEW\*.

#### 15.17.5 DELETEALL

DELETEALL löscht alle Datensätze einer Tabelle.

(DELETEALL *table*[\*])

Löscht alle Datensätze der gegebenen Tabelle. Wird ein Stern hinter dem Tabellennamen angehängt, dann werden nur die Datensätze gelöscht, die dem aktuellen Filter der Tabelle genügen. Es erscheint kein Sicherheitsfenster, bevor die Datensätze gelöscht werden!

DELETEALL liefert TRUE, wenn alle Datensätze erfolgreich gelöscht werden konnten, anderenfalls NIL. Ist *table* NIL, dann wird NIL geliefert.

Beispiel: '(DELETEALL *table*\*)' löscht alle Datensätze in der gegebenen Tabelle, die dem Filter der Tabelle genügen.

Siehe auch DELETE, Tables.

#### 15.17.6 GETMATCHFILTER

GETMATCHFILTER liefert den Status der Filterübereinstimmung eines Datensatzes.

---

<sup>39</sup> Anm.d.Übersetzers: Hier ist wegen Einschränkungen bei Texinfo leider kein Verweis möglich.

(GETMATCHFILTER *rec*)

Liefert TRUE, wenn der gegebene Datensatz dem Filter seiner Tabelle entspricht, anderenfalls NIL. Ist der Filter der Tabelle momentan nicht aktiviert, dann wird TRUE geliefert. Ist *rec* NIL (der Vorgabedatensatz), dann wird NIL geliefert.

Siehe auch SETMATCHFILTER, GETISSORTED, GETFILTERSTR, SETFILTERSTR.

### 15.17.7 SETMATCHFILTER

SETMATCHFILTER setzt den Status der Filterübereinstimmung eines Datensatzes.

(SETMATCHFILTER *rec on*)

Ändert den Status der Filterübereinstimmung beim gegebenen Datensatz auf den Wert von *on*. SETMATCHFILTER liefert den neuen Status der Filterübereinstimmung des gegebenen Datensatzes. Der neue Status kann vom erwarteten abweichen, weil das Setzen auf NIL nur dann wirksam ist, wenn der Filter der dazugehörigen Tabelle aktiviert ist, anderenfalls wird TRUE geliefert. Der Aufruf von SETMATCHFILTER mit dem Wert NIL für *rec* (der Vorgabedatensatz) liefert immer NIL.

Siehe auch GETMATCHFILTER, SETISSORTED, GETFILTERSTR, SETFILTERSTR.

### 15.17.8 GETISSORTED

GETISSORTED liefert den Sortierstatus eines Datensatzes.

(GETISSORTED *rec*)

Liefert TRUE, wenn der gegebene Datensatz nach der für die Tabelle definierten Reihenfolge sortiert ist, ansonsten NIL. Ist *rec* NIL, dann wird NIL geliefert.

Siehe auch SETISSORTED, GETMATCHFILTER, REORDER, GETORDERSTR, SETORDERSTR, Comparison function.

### 15.17.9 SETISSORTED

SETISSORTED setzt den Sortierstatus eines Datensatzes.

(SETISSORTED *rec on*)

Ändert den Sortierstatus des angegebenen Datensatzes auf *on*. Die Funktion wird verwendet, wenn man der Meinung ist, daß der Datensatz in der richtigen Reihenfolge steht (*on* = TRUE) oder er neu sortiert werden sollte (*on* = NIL). Neusortieren aller unsortierten Datensätze kann mit der Funktion REORDER (siehe Abschnitt 15.19.4 [REORDER], Seite 142) durchgeführt werden.

SETISSORTED liefert den neuen Sortierstatus des gegebenen Datensatzes. Der Aufruf von SETISSORTED mit dem Wert NIL für *rec* (der Anfangsdatsatz) wird NIL liefern.

Für ein Beispiel, wie diese Funktion angewendet wird, siehe Abschnitt 15.28.6 [Comparison function], Seite 158.

Siehe auch GETISSORTED, SETMATCHFILTER, REORDER, GETORDERSTR, SETORDERSTR, Comparison function.

### 15.17.10 RECNUM

RECNUM liefert die Datensatznummer des Datensatzes.

(RECNUM *record*)

Liefert die Datensatznummer des gegebenen Datensatzes. Man beachte, daß die Nummerierung der Datensätze von z.B. der der Listen abweicht. Bei Listen, Zeichenketten und anderem beginnt die Zählung bei Null, bei den Datensätzen beginnt sie jedoch bei 1. Die Nummer 0 ist für den Vorgabedatensatz reserviert. Dies scheint mit den restlichen MUIbase-Funktionen unvereinbar zu sein, aber hier macht es wirklich Sinn, da die Datensatznummern auch in der Fensteranzeige verwendet werden.

Siehe auch RECORDS, INT.

### 15.17.11 COPYREC

COPYREC kopiert Datensätze.

(COPYREC *rec source*)

Kopiert den Inhalt des Datensatzes *source* in den Datensatz *rec*. Ist *source* NIL, dann wird *rec* auf die Werte des Vorgabedatensatzes gesetzt. Ist *rec* NIL, dann wird eine Fehlermeldung erzeugt.

COPYREC liefert *rec*.

Siehe auch NEW.

## 15.18 Feldfunktionen

Dieser Abschnitt behandelt Funktionen für Felder einer Tabelle.

### 15.18.1 ATTRNAME

ATTRNAME liefert den Namen des Feldes.

(ATTRNAME *attr*)

Liefert eine Zeichenkette mit dem Namen des angegebenen Feldes.

Siehe auch `TABLERNAME`

### 15.18.2 MAXLEN

`MAXLEN` liefert die maximale Anzahl von Zeichen eines Zeichenkettenfeldes.

`(MAXLEN string-attr)`

Liefert die maximale Anzahl von Zeichen, die das gegebene Zeichenkettenfeld aufnehmen kann.

Siehe auch `LEN`.

### 15.18.3 GETLABELS

`GETLABELS` liefert alle Auswahltexte eines Auswahl- oder Zeichenkettenfeldes.

`(GETLABELS attr)`

Liefert die Auswahltexte des gegebenen Auswahl- oder Zeichenkettenfeldes. Im Falle eines Auswahlfeldes werden die im Auswahltexteditor (siehe Typabhängige Einstellungen<sup>40</sup>) eingegebenen Texte zurückgegeben, Bei Zeichenkettenfeldern werden die Auswahltexte zurückgegebene, die für das Listenansicht-Popup (siehe Abschnitt 14.3.3 [Attribute object editor], Seite 69) eingegeben wurden.

Die Auswahltexte werden in einer einzelnen Zeichenkette zurückgegeben und werden jeweils durch ein Neue-Zeile-Zeichen getrennt.

Beispiel: Man nehme an, man hat ein Auswahlfeld mit den Auswahltexten 'Auto', 'Haus' und 'Öl'. Der Aufruf von `GETLABELS` mit diesem Feld liefert dann die Zeichenkette "Auto\nHaus\nÖl".

Hinweis: Diese Rückgabezeichenkette läßt sich einfach mit `MEMOTOLIST` (siehe Abschnitt 15.13.3 [MEMOTOLIST], Seite 117) in eine Liste umwandeln.

Siehe auch `SETLABELS`.

### 15.18.4 SETLABELS

`SETLABELS` wird verwendet, um die Auswahltexte eines Zeichenkettenfeldes zu setzen.

`(SETLABELS attr str)`

Setzt die Auswahltexte des Zeichenkettenfeldes *attr* auf die Auswahlfelder, die im Parameter *str* aufgelistet sind. Der Parameter *str* enthält für jeden Auswahltext eine Zeile. Die Auswahltexte

---

<sup>40</sup> Anm.d.Übersetzers: Hier ist wegen Umlaut-Einschränkungen bei Texinfo leider kein Verweis möglich.

ersetzen diejenigen, die in dem Listenansicht-Popup des Feldobjekteditors (siehe Abschnitt 14.3.3 [Attribute object editor], Seite 69) eingegeben wurden.

SETLABELS liefert den Wert des Parameters *str*.

Beispiel: `'(SETLABELS Table.String "Mein Haus\nist\ndein Haus")'` setzt die Listenansicht-Auswahltexte des gegebenen Zeichenkettenfeldes auf `sets the list-view labels of the specifies string attribute to 'Mein Haus', 'ist' und 'dein Haus'`.

Hinweis: Man kann eine Liste von Auswahltexte durch den Aufruf von LISTTOMEMO in das benötigte Zeichenkettenformat umwandeln.

Siehe auch GETLABELS.

## 15.19 Tabellenfunktionen

### 15.19.1 TABLENAME

TABLENAME liefert den Namen einer Tabelle.

`(TABLENAME table)`

Liefert eine Zeichenkette, die den Namen der angegebenen Tabelle enthält.

Siehe auch ATTRNAME

### 15.19.2 GETORDERSTR

GETORDERSTR liefert die Datensatzreihenfolge einer Tabelle.

`(GETORDERSTR table)`

Verwendet die Tabelle eine Felderliste zum Sortieren, dann enthält die gelieferte Zeichenkette die Feldnamen, getrennt durch Leerzeichen. Jedes Feld hat ein '+' oder ein '-' vorangestellt, um eine auf- bzw. absteigende Sortierung anzuzeigen.

Wird die Tabelle anhand einer Vergleichsfunktion sortiert, dann wird der Name dieser Funktion geliefert.

Eine leere Zeichenkette zeigt an, daß keine Sortierung vorliegt.

## Beispiel

Angenommen, es gibt eine Tabelle `'Person'`, die nach ihren Feldern `'Name'` (aufsteigend), `'Stadt'` (aufsteigend) und `'Geburtstag'` (absteigend) sortiert ist. Dann liefert `'(ORDERSTR Person)'` die Zeichenkette `" +Name +Stadt -Geburtstag"`.

Siehe auch `SETORDERSTR`, `REORDER`, `REORDERALL`, `GETISSORTED`, `SETISSORTED`, `Order`, `Comparison function`.

### 15.19.3 SETORDERSTR

`SETORDERSTR` setzt die Sortierreihenfolge einer Tabelle.

`(SETORDERSTR table order)`

Setzt die Sortierreihenfolge der gegebenen Tabelle auf die Felder in der Zeichenkette `order`. Die Zeichenkette `order` kann entweder eine Liste von Feldnamen enthalten oder den Namen der Vergleichsfunktion.

Zum Sortieren einer Feldliste muß die Zeichenkette `order` die Feldnamen für die Sortierung enthalten, die durch eine beliebige Anzahl von Leerzeichen, Tabulatoren oder Neue-Zeile-Zeichen getrennt sind. Jedem Feldnamen kann ein `'+'` oder ein `'-'` für auf- bzw. absteigende Sortierung vorangestellt werden. Wird dieses Zeichen weggelassen, dann wird aufsteigende Sortierung angenommen.

Zum Sortieren anhand einer Vergleichsfunktion muß die Zeichenkette `order` den Namen der Funktion tragen.

`SETORDERSTR` liefert `TRUE`, wenn es möglich war, die neue Sortierung zu setzen, anderenfalls `NIL`, wenn z.B. ein unbekanntes Feld angegeben wurde oder das Typ des Feldes für die Sortierung nicht erlaubt ist. Wird `NIL` für `order` angegeben, dann passiert nichts und es wird `NIL` zurückgeliefert.

Hinweis: Zum Erzeugen der Sortierzeichenkette sollten man nicht direkt den Feldnamen in die Zeichenkette einfügen, weil bei einer Änderung des Feldnamens der Name in der Zeichenkette nicht mit verändert wird. Besser ist es, die Funktion `ATTRNAME` (siehe Abschnitt 15.18.1 [ATTRNAME], Seite 138) zu verwenden und dessen Ergebnis in die Sortierzeichenkette zu kopieren.

## Beispiel

Man betrachte eine Tabelle `'Person'` mit den Feldern `'Name'`, `'Stadt'` und `'Geburtstag'`. `'(SETORDERSTR Person (SPRINTF "+%s" (ATTRNAME Person.Name)))'` setzt dann die Sortierreihenfolge der Tabelle `'Person'` auf `'Name'` als (aufsteigendes) Sortierfeld.

Siehe auch `GETORDERSTR`, `REORDER`, `REORDERALL`, `GETISSORTED`, `SETISSORTED`, `Order`, `Comparison function`.

### 15.19.4 REORDER

REORDER bringt alle unsortierten Datensätze zurück in die richtige Reihenfolge.

(REORDER *table*)

Untersucht alle Datensätze der gegebenen Tabelle nach unsortierten Datensätzen und fügt diese in ihrer korrekten Position ein. Nach dem Einfügen eines unsortierten Datensatzes wird der Sortierstatus des Datensatzes auf TRUE gesetzt, so daß bei nach Beendigung der Funktion REORDER der Sortierstatus aller Datensätze auf TRUE steht.

REORDER liefert NIL.

Normalerweise wird diese Funktion nur dann aufgerufen, wenn eine Vergleichsfunktion für die Sortierung der Tabelle definiert wurde. Sortierungen anhand einer Felderliste sind automatisch, das bedeutet, daß ein Datensatz automatisch sortiert wird, wenn er benötigt wird.

Für einen Anwendungsfall zur Anwendung dieser Funktion siehe Abschnitt 15.28.6 [Comparison function], Seite 158.

Siehe auch REORDERALL, GETORDERSTR, SETORDERSTR, GETISSORTED, SETISSORTED, Order, Comparison function.

### 15.19.5 REORDERALL

REORDERALL sortiert alle Datensätze einer Tabelle neu.

(REORDERALL *table*)

Sortiert alle Datensätze der gegebenen Tabelle neu, indem der Sortierstatus aller Datensätze auf NIL gesetzt und dann REORDER zum kompletten Neusortieren aufgerufen wird.

REORDERALL liefert NIL.

Siehe auch REORDER, GETORDERSTR, SETORDERSTR, GETISSORTED, SETISSORTED, Order, Comparison function.

### 15.19.6 GETFILTERACTIVE

GETFILTERACTIVE liefert den Filterstatus einer Tabelle.

(GETFILTERACTIVE *table*)

Liefert TRUE, wenn der Filter gegebenen Tabelle momentan aktiviert ist, anderenfalls NIL.

Siehe auch SETFILTERACTIVE, GETFILTERSTR, GETMATCHFILTER.



### 15.19.7 SETFILTERACTIVE

SETFILTERACTIVE setzt den Filterstatus einer Tabelle.

(SETFILTERACTIVE *table bool*)

Setzt den Filterstatus der gegebenen Tabelle. Ist *bool* nicht NIL, dann wird er Filter aktiviert, anderenfalls deaktiviert.

SETFILTERACTIVE liefert den neuen Status des Filters. Der neue Status muß nicht dem erwarteten entsprechen, falls beim Aktivieren des Filters ein Fehler auftrat und der Filter deshalb nicht aktiviert werden konnte. Deaktivieren des Filters gelingt jedoch immer..

Siehe auch GETFILTERACTIVE, SETFILTERSTR, SETMATCHFILTER.

### 15.19.8 GETFILTERSTR

GETFILTERSTR liefert den Datensatzfilterausdruck einer Tabelle.

(GETFILTERSTR *table*)

Liefert den Datensatzfilterausdruck der gegebenen Tabelle als Zeichenkette. Eine leere Zeichenkette bedeutet, daß kein Filterausdruck für diese Tabelle gesetzt wurde.

Siehe auch SETFILTERSTR, GETFILTERACTIVE, GETMATCHFILTER.

### 15.19.9 SETFILTERSTR

SETFILTERSTR setzt den Datensatzfilterausdruck einer Tabelle.

(SETFILTERSTR *table filter-str*)

Setzt den Datensatzfilterausdruck der gegebenen Tabelle auf den Ausdruck im Parameter *filter-str*<sup>41</sup>. Ist der Filter der gegebenen Tabelle momentan aktiviert, dann wird der neue Filterausdruck sofort auf alle Datensätze angewendet und der Status der Filterübereinstimmung aller Datensätze neu berechnet.

SETFILTERSTR liefert TRUE, wenn es möglich war, den gegebenen Filterzeichenkettenausdruck zu kompilieren, anderenfalls wird NIL geliefert. Man beachte, daß man nur das Ergebnis der Kompilierung erhält. Ist der Filter der gegebenen Tabelle momentan aktiviert und das Neuberechnen aller Stati der Filterübereinstimmungen fehlschlägt, dann wird man nicht über das Ergebnis dieser Funktion informiert. Daher ist der empfohlene Weg, einen neuen Filterausdruck zu setzen, folgender:

---

<sup>41</sup> Anm.d.Übersetzers: Hier wird ein Ausdruck angegeben, der in einer Zeichenkette stehen muß, also nicht der Ausdruck selbst! Letzteres liefert sonst nur das Ergebnis als Ausdruck, was hier zu einem Fehler führt.

```

(SETFILTERACTIVE Table NIL)                ; gelingt immer.
(IF (NOT (SETFILTERSTR Table filter-string))
  (ERROR "Kann den Filter für %s nicht setzen!" (TABLENAME Table))
)
(IF (NOT (SETFILTERACTIVE Table TRUE))
  (ERROR "Kann den Filter für %s nicht aktivieren!" (TABLENAME Table))
)

```

Wird SETFILTERSTR mit dem Wert NIL für den Parameter *filter-str* aufgerufen, dann passiert nichts und NIL wird zurückgeliefert.

Beispiel: '(SETFILTERSTR Table "> Wert 0.0)")'.

Siehe auch GETFILTERSTR, SETFILTERACTIVE, SETMATCHFILTER.

### 15.19.10 RECORDS

RECORDS liefert die Anzahl der Datensätze in einer Tabelle.

```
(RECORDS table)
```

Liefert die Anzahl der Datensätze in der gegebenen Tabelle. Man kann einen Stern zum Tabellennamen hinzufügen, um die Anzahl der Datensätze zu ermitteln, die dem Filter der Tabelle genügen.

Siehe auch RECORD, RECNUM.

### 15.19.11 RECORD

RECORD liefert einen Datensatzzeiger für eine gegebene Datensatznummer.

```
(RECORD table num)
```

Liefert den Datensatzzeiger des *num*-ten Datensatzes in der gegebenen Tabelle oder NIL, wenn ein Datensatz mit dieser Nummer nicht existiert. Man kann einen Stern zum Tabellennamen hinzufügen, um den *num*-ten Datensatz zu erhalten, der dem Datensatzfilter genügt<sup>42,43</sup>.

Es ist darauf zu achten, daß Datensatznummern bei 1 beginnen und die Datensatznummer 0 für den Vorgabedatensatz verwendet wird.

Siehe auch RECORDS, RECNUM.

---

<sup>42</sup> Anm.d.Übersetzers: Die Nummerierung gefilterter Datensätze weicht von der der ungefilterten ab, sobald Datensatz nicht zum Filter passen und daher sind die Nummern nicht identisch!

<sup>43</sup> Anm.eines gefilterten Datensatzes: Da hat der Übersetzer recht! :-)

### 15.19.12 SELECT

SELECT ermittelt und liefert diverse Daten von Datensätzen.

```
(SELECT [DISTINCT] exprlist FROM tablelist
      [WHERE where-expr] [ORDER BY orderlist])
```

wobei *exprlist* entweder ein einfacher Stern '\*' oder eine Liste von durch Komma getrennten Ausdrücken mit optionalen Titeln ist:

```
exprlist:      * | expr "Titel", ...
```

und *tablelist* eine Liste von Tabellennamen:

```
tablelist:      table[*] [ident], ...
```

Für jede Tabelle in der Tabellenliste kann ein Identifier<sup>44</sup> Dies kann nützlich sein, wenn eine Tabelle mehr als einmal in der Tabellenliste vorkommt (siehe unten das Beispiel zum Vergleichen von Altersangaben). Wird ein Stern zum Tabellennamen hinzugefügt, dann werden nur die Datensätze der Tabelle betrachtet, die dem momentanen Filter der Tabelle genügen.

Die Sortierliste hat den folgenden Aufbau:

```
orderlist:      expr [ASC | DESC], ...
```

wobei *expr*, ... beliebige Ausdrücke oder Feldnummern sein können. Zum Beispiel sortiert '(SELECT Name FROM ... ORDER BY 1)' das Ergebnis nach dem Feld 'Name'. Man kann zudem ASC oder DESC für eine auf- bzw. absteigende Sortierung angeben. Ist keiner der beiden vorhanden, dann wird aufsteigende Sortierung angenommen.

### Wie es arbeitet

Die SELECT-FROM-WHERE-Abfrage bildet das (mathematische) Kreuzprodukt aller Tabellen in der Tabellenliste (es wertet alle Datensatzmengen in *table*, ... aus) und prüft den WHERE-Ausdruck (wenn vorhanden). Liefert der WHERE-Ausdruck TRUE (oder es gibt keinen WHERE-Ausdruck), dann wird eine Liste erzeugt, dessen Elemente anhand der Ausdrucksliste im SELECT-Teil berechnet wurden. Wurde ein einzelner Stern in der Ausdrucksliste angegeben, dann enthält die Liste die Werte aller Felder, die zu den Tabellen in der Tabellenliste gehören (hiervon ausgenommen sind die virtuellen Felder und Knöpfe).

Das Ergebnis der Abfrage ist eine Liste von Listen. Der erste Listeneintrag enthält die Titelzeichenketten, die restlichen die Werte der FROM-Liste für die passenden Datensätze.

---

<sup>44</sup> Anm.d.Übersetzers: Das Wort 'Identifier' übersetzt man am besten mit 'Benenner'. Da dies -mit Verlaub- etwas blöd klingt, habe ich mich entschlossen, den englischen Ausdruck beizubehalten. Andere Übersetzungsideen sind jederzeit willkommen => eMail schicken.

## Beispiele

Siehe Abschnitt 13.4 [Query examples], Seite 60 für einige Beispiele mit der Funktion **SELECT**.

Siehe auch **FOR ALL**.

## 15.20 Oberflächenfunktionen

Dieser Abschnitt beschreibt die Funktionen zum Verändern von Benutzeroberflächenelementen.

### 15.20.1 SETCURSOR

**SETCURSOR** setzt den Cursor auf ein Benutzeroberflächenelement.

`(SETCURSOR attr-or-table)`

Setzt den Cursor auf das gegebene Feld oder Benutzeroberflächenelement der Tabelle. Die Funktion öffnet auch das Fenster, in dem das Feld/die Tabelle eingebettet ist, wenn das Fenster noch geschlossen ist.

**SETCURSOR** liefert **TRUE**, wenn kein Fehler auftrat (Fenster konnte geöffnet werden) oder **NIL** bei einem Fehler.

### 15.20.2 GETDISABLED

**GETDISABLED** liefert den Inaktiv-Status eines Feldes.

`(GETDISABLED attr)`

Liefert den Inaktiv-Status des angegebenen Feldes im aktuellen Datensatz.

Siehe auch **SETDISABLED**, **GETWINDOWDISABLED**.

### 15.20.3 SETDISABLED

**SETDISABLED** setzt den Inaktiv-Status eines Feldes.

`(SETDISABLED attr bool)`

Setzt den Inaktiv-Status des angegebenen Feldes im aktuellen Datensatz auf den Wert von *bool*. Liefert den neuen Wert des Inaktiv-Status.

Siehe auch **GETDISABLED**, **SETWINDOWDISABLED**.

#### 15.20.4 GETWINDOWDISABLED

GETWINDOWDISABLED liefert den Inaktiv-Status eines Fensters.

(GETWINDOWDISABLED *attr-or-table*)

Liefert den Status des Inaktiv-Flags für das Fenster, in dem das angegebene Feld oder die Tabelle eingebettet ist. Ist das Feld oder die Tabelle im Hauptfenster eingebettet, dann wird NIL geliefert<sup>45</sup>.

Siehe auch SETWINDOWDISABLED, GETWINDOWOPEN, GETDISABLED.

#### 15.20.5 SETWINDOWDISABLED

SETWINDOWDISABLED setzt den Inaktiv-Status eines Fensters.

(SETWINDOWDISABLED *attr-or-table disabled*)

Setzt den Status des Inaktiv-Flags des Fensters, in dem das angegebene Feld bzw. die Tabelle eingebettet ist, auf den Wert von *disabled*. Wenn man ein Fenster deaktiviert, dann wird das Fenster geschlossen und der dazugehörige Fensterknopf wird deaktiviert. Das Hauptfenster eines Projekts kann nicht deaktiviert werden.

SETWINDOWDISABLED liefert den neuen Inaktiv-Status des Fensters.

Siehe auch GETWINDOWDISABLED, SETWINDOWOPEN, SETDISABLED.

#### 15.20.6 GETWINDOWOPEN

GETWINDOWOPEN liefert den Geöffnet-Status eines Fensters.

(GETWINDOWOPEN *attr-or-table*)

Liefert den Geöffnet-Status des Fensters, in dem das Feld bzw. die Tabelle eingebettet ist.

Siehe auch SETWINDOWOPEN, GETWINDOWDISABLED.

#### 15.20.7 SETWINDOWOPEN

SETWINDOWOPEN öffnet und schließt ein Fenster.

(SETWINDOWOPEN *attr-or-table open*)

---

<sup>45</sup> Anm.d.Übersetzers: Da das Hauptfenster nie geschlossen werden kann, ohne daß dabei das Projekt geschlossen wird, wird auch immer NIL geliefert.

Öffnet oder schließt das Fenster, in dem das gegebenen Feld bzw. die gegebenen Tabelle eingebettet ist. Ist *open* nicht NIL, dann wird das Fenster geöffnet, anderenfalls wird es geschlossen. Das Hauptfenster eines Projekts kann nicht geschlossen werden.

SETWINDOWOPEN liefert den neuen Geöffnet-Status des Fensters.

Siehe auch GETWINDOWOPEN, SETWINDOWDISABLED.

## 15.21 Projektfunktionen

Dieser Abschnitt listet Funktionen auf, die mit Projekten zu tun haben.

### 15.21.1 PROJECTNAME

PROJECTNAME liefert den Projektnamen.

(PROJECTNAME)

PROJECTNAME liefert den Namen des aktuellen Projekts als Zeichenkette oder NIL, wenn noch kein Name definiert wurde.

Siehe auch CHANGES.

### 15.21.2 CHANGES

CHANGES liefert die Anzahl der bisher gemachten Änderungen am aktuellen Projekt.

(CHANGES)

Liefert eine Ganzzahl mit der Anzahl der Änderungen seit der letzten Speicherung des aktuellen Projekts.

Siehe auch PROJECTNAME.

## 15.22 Systemfunktionen

Dieser Abschnitt listet Funktionen auf, die auf das Betriebssystem zugreifen.

### 15.22.1 EDIT

EDIT startet den externen Editor.

(EDIT *filename*)

Startet den externen Editor zum Bearbeiten der gegebenen Datei. Der externen Editor kann unter dem Menüpunkt ‘**Einstellungen – Externen Editor setzen**’ (siehe Abschnitt 7.7 [External editor], Seite 37) eingestellt werden. **EDIT** starten den externen Editor asynchron, das bedeutet, die Funktion kehrt sofort wieder zurück (und wartet nicht auf das Ende des Editors).

**EDIT** liefert **TRUE**, wenn der Editor erfolgreich gestartet werden konnte, anderenfalls **NIL**.

Siehe auch **EDIT\***, **VIEW**, **SYSTEM**.

### 15.22.2 **EDIT\***

**EDIT\*** ist die Version von **EDIT** mit dem Stern und hat den selben Effekt wie **EDIT** (siehe Abschnitt 15.22.1 [**EDIT**], Seite 148). Der einzige Unterschied ist, daß **EDIT\*** den externen Editor synchron startet und wartet, bis der Benutzer den Editor verlassen hat.

Siehe auch **EDIT**, **VIEW\***, **SYSTEM**.

### 15.22.3 **VIEW**

**VIEW** startet den externen Anzeiger.

(**VIEW** *filename*)

Startet den externen Anzeiger zum Anzeigen der gegebenen Datei. Der externen Anzeiger kann unter dem Menüpunkt ‘**Einstellungen – Externen Anzeiger setzen**’ (siehe Abschnitt 7.8 [External viewer], Seite 38) eingestellt werden. **VIEW** starten den externen Anzeiger asynchron, das bedeutet, die Funktion kehrt sofort wieder zurück (und wartet nicht auf das Ende des Anzeigers).

**VIEW** liefert **TRUE**, wenn der Anzeiger erfolgreich gestartet werden konnte, anderenfalls **NIL**.

Siehe auch **VIEW\***, **EDIT**, **SYSTEM**.

### 15.22.4 **VIEW\***

**VIEW\*** ist die Version von **VIEW** mit dem Stern und hat den selben Effekt wie **VIEW** (siehe Abschnitt 15.22.3 [**VIEW**], Seite 149). Der einzige Unterschied ist, daß **VIEW\*** den externen Anzeiger synchron startet und wartet, bis der Benutzer den Anzeiger verlassen hat.

Siehe auch **VIEW**, **EDIT\***, **SYSTEM**.

### 15.22.5 **SYSTEM**

**SYSTEM** ruft ein externes Programm auf.

(**SYSTEM** *fmt* [*arg* ...])

Ruft ein externes Programm auf. Die Befehlszeile zum Programmaufruf wird aus *fmt* und den optionalen Parametern wie in der Funktion **SPRINTF** (siehe Abschnitt 15.12.28 [SPRINTF], Seite 114) erzeugt. **SYSTEM** wartet, bis das aufgerufene Programm beendet wurde. Wenn nicht gewünscht ist, daß **SYSTEM** warten soll, dann benutzt man eine Befehlszeile, die das Programm im Hintergrund startet.

**SYSTEM** liefert bei Erfolg TRUE, anderenfalls NIL, wenn z.B. die Befehlszeile nicht ausgeführt werden konnte oder der aufgerufene Befehl einen Fehlercode lieferte.

Beispiel: '(SYSTEM "run %s %s" "clock" "digital")' startet die Systemuhr mit Digitalmodus als Hintergrundprozeß.

Siehe auch EDIT, EDIT\*, VIEW, VIEW\*.

### 15.22.6 STAT

STAT untersucht eine Datei.

(STAT *filename*)

Untersucht, ob der angegebene Dateiname im Dateisystem existiert. **STAT** liefert NIL, wenn der Dateiname nicht gefunden werden konnte; 0, wenn der Dateiname existiert und ein Verzeichnis ist, und eine Ganzzahl größer 0, wenn der Dateiname existiert und eine gültige Datei ist<sup>46</sup>

### 15.22.7 TACKON

TACKON erzeugt einen Pfadnamen.

(TACKON *dirname filename*)

Verknüpft *dirname* und *filename* zu einem Pfadnamen. **TACKON** weiß, wie es mit Doppelpunkten und Schrägstrichen (Slashes) in *dirname* umzugehen hat. Es liefert den Pfadnamen als Zeichenkette oder NIL, wenn *dirname* oder *filename* NIL ist.

Beispiel: '(TACKON "Sys:System" "CLI")' liefert "Sys:System/CLI".

Siehe auch FILENAME, DIRNAME.

### 15.22.8 FILENAME

FILENAME extrahiert den Dateinamen aus einem Pfadnamen.

(FILENAME *path*)

---

<sup>46</sup> Anm.d.Übersetzers: Ich weiß nicht, wie Links behandelt werden. Wird für **STAT** eine AmigaOS-Funktion verwendet, dann können auch negative Werte auftreten, die die verschiedenen Link-Varianten repräsentieren.



Extrahiert die letzte Komponente eines gegebenen Pfadnamens. Es wird nicht geprüft, ob die letzte Komponente momentan auf eine Datei verweist, so daß es auch möglich ist, `FILENAME` zu verwenden, um den Namen eines Unterverzeichnisses zu erhalten. `FILENAME` liefert sein Ergebnis als Zeichenkette oder `NIL`, wenn *path* `NIL` ist.

Beispiel: ‘(`FILENAME "Sys:System/CLI"`)’ liefert `"CLI"`.

Siehe auch `DIRNAME`, `TACKON`.

### 15.22.9 DIRNAME

`DIRNAME` extrahiert den Verzeichnis-Teil eines Pfadnamens.

(`DIRNAME path`)

Extrahiert den Verzeichnis-Teil des gegebenen Pfadnamens. Es wird nicht geprüft, ob *path* momentan auf eine Datei verweist, so daß es auch möglich ist, `DIRNAME` zu verwenden, um den Namen eines übergeordneten Verzeichnisses zu erhalten. `DIRNAME` liefert sein Ergebnis als Zeichenkette oder `NIL`, wenn *path* `NIL` ist.

Beispiel: ‘(`DIRNAME "Sys:System/CLI"`)’ liefert `"Sys:System"`.

Siehe auch `FILENAME`, `TACKON`.

### 15.22.10 TODAY

`TODAY` liefert das heutige Datum.

(`TODAY`)

Liefert das heutige Datum als Datumswert.

Siehe auch `NOW`.

### 15.22.11 NOW

`NOW` liefert die aktuelle Uhrzeit.

(`NOW`)

Liefert die aktuelle Uhrzeit als Zeitwert.

Siehe auch `TODAY`.

### 15.22.12 MESSAGE

MESSAGE gibt eine Meldung für den Benutzer aus.

```
(MESSAGE fmt [arg ...])
```

Setzt den Fenstertitel des Pause/Abbrechen-Fensters (wenn es geöffnet ist). Die Titelzeichenkette wird aus *fmt* und den optionalen Parametern wie in der Funktion `SPRINTF` (siehe Abschnitt 15.12.28 [SPRINTF], Seite 114) erzeugt.

MESSAGE liefert die formatierte Titelzeichenkette.

Beispiel: `'(MESSAGE "6 * 7 = %i" (* 6 7))'`.

Siehe auch `PRINT`, `PRINTF`.

### 15.22.13 GC

GC erzwingt das Aufräumen des Speichers<sup>47</sup>.

```
(GC)
```

Erzwingt das Aufräumen des Speichers und liefert `NIL`. Im Normalfall wird das Aufräumen automatisch von Zeit zu Zeit durchgeführt.

## 15.23 Vordefinierte Variablen

MUIbase kennt einige vordefinierte globale Variablen.

Momentan existiert nur eine einzige globale Variable: `stdout` (siehe Abschnitt 15.16.3 [stdout], Seite 129).

## 15.24 Vordefinierte Konstanten

Die folgenden vordefinierten Konstanten können in jedem Ausdruck bei der Programmierung verwendet werden:

Name	Typ	Wert	Bemerkung
-----			
<code>NIL</code>	<code>jeder</code>	<code>NIL</code>	
<code>TRUE</code>	<code>bool</code>	<code>TRUE</code>	
<code>RESET</code>	<code>Zeichenkette</code>	<code>"\33c"</code>	

<sup>47</sup> Anm.d.Übersetzers: Im Original nennt es sich 'Garbage collection', also 'Müllabfuhr'. Ein Garbage Collector reinigt also den von MUIbase selbst verwalteten Speicher von unbenutzten Speicherblöcken, damit dieser wieder für irgendwelche Zwecke verwendet werden kann.

NORMAL	Zeichenkette	"\33[0m"	
ITON	Zeichenkette	"\33[3m"	
ITOFF	Zeichenkette	"\33[23m"	
ULON	Zeichenkette	"\33[4m"	
ULOFF	Zeichenkette	"\33[24m"	
BFON	Zeichenkette	"\33[1m"	
BFOFF	Zeichenkette	"\33[22m"	
ELITEON	Zeichenkette	"\33[2w"	
ELITEOFF	Zeichenkette	"\33[1w"	
CONDON	Zeichenkette	"\33[4w"	
CONDOFF	Zeichenkette	"\33[3w"	
WIDEON	Zeichenkette	"\33[6w"	
WIDEOFF	Zeichenkette	"\33[5w"	
NLQON	Zeichenkette	"\33[2\"z"	
NLQOFF	Zeichenkette	"\33[1\"z"	
INT_MAX	Ganzzahl	2147483647	Größter Ganzzahlwert
INT_MIN	Ganzzahl	-2147483648	Kleinsten Ganzzahlwert
HUGE_VAL	Fließkommazahl	1.797693e+308	Größte absolute Fließkommazahl
PI	Fließkommazahl	3.14159265359	
OSVER	Ganzzahl	<OS-Version>	
OSREV	Ganzzahl	<OS-Revision>	
MBVER	Ganzzahl	<MUIbase-Version>	
MBREV	Ganzzahl	<MUIbase-Revision>	
LANGUAGE	Zeichenkette	hängt von der lokalen Sprache ab	
SEEK_SET	Ganzzahl	siehe stdio.h	Suche vom Beginn der Datei.
SEEK_CUR	Ganzzahl	siehe stdio.h	Suche von aktueller Position.
SEEK_END	Ganzzahl	siehe stdio.h	Suche vom Ende der Datei.

Siehe Abschnitt 15.3.7 [Constants], Seite 82 für weitere Informationen über Konstanten. Zum Definieren eigener Konstanten benutzt man die Preprozessor-Anweisung `#define` (siehe Abschnitt 15.2.1 [`#define`], Seite 76).

## 15.25 Funktionale Parameter

Es ist möglich, eine Funktion als einen Parameter an eine andere Funktion zu übergeben. Dies ist nützlich für die Definition von übergeordneten Funktionen, wie z.B. zum Sortieren oder Abbilden einer Liste.

Um eine Funktion anzurufen, die als Parameter übergeben wurde, muß die Funktion `FUNCALL` (siehe Abschnitt 15.5.6 [`FUNCALL`], Seite 88) verwendet werden.

### Beispiel:

```
(DEFUN map (l fun)
  (LET (res)
    (DOLIST (i l)
      (SETQ res
```

# Parameter: Liste und Funktion  
# lokale Variable res, vorbelegt mit NIL  
# jedes Element der Reihe nach

```

                (CONS (FUNCALL fun i) res)      # Funktion anwenden und
            )                                     # neue Liste erzeugen
        )
    (REVERSE res)      # die Liste muß nun umgekehrt werden
)

```

Jetzt kann diese Abbildfunktion zum Beispiel verwendet werden, um alle Elemente einer Liste mit Ganzzahlen um 1 zu erhöhen:

‘(map (LIST 1 2 3 4) 1+)’ liefert ( 2 3 4 5 ).

Siehe auch FUNCALL, MAPFIRST.

## 15.26 Typdeklarierer

Es ist möglich, den Typ einer lokalen Variable durch Anfügen eines Typdeklarierers hinter dem Namen festzulegen. Die folgenden Typdeklarierer existieren:

### Deklarierer Beschreibung

:INT	für Ganzzahlen
:REAL	für Fließkommazahlen
:STR	für Zeichenketten
:MEMO	für mehrzeilige Zeichenketten
:DATE	für Datumswerte
:TIME	für Zeitwerte
:LIST	für Listen
:FILE	für Dateihandler
:FUNC	für Funktionen jedes Typs
:table	für Datensatzzeiger auf table

Der Typdeklarierer wird an den Variablennamen wie im folgenden Beispiel angehängt:

```
(LET (x:INT (y:REAL 0.0) z) ...)
```

Das Beispiel definiert drei neue Variablen ‘x’, ‘y’ und ‘z’, wobei ‘x’ vom Typ Ganzzahl ist und mit NIL vorbelegt wird, ‘y’ vom Typ Fließkommazahl ist und mit 0.0 vorbelegt wird, und ‘z’ eine Variable ohne Typ ist, die mit NIL vorbelegt wird.

Der Vorteil von Typspezifizierern ist, daß der Compiler mehr Typfehler entdecken kann, z.B. wenn eine Funktion

```
(DEFUN foo (x:INT) ...)
```

definiert ist und sie mit `(foo "bar")` aufgerufen wird, dann erzeugt der Compiler eine Fehlermeldung. Wird `foo` jedoch mit einem Wert ohne Typ aufgerufen, z.B. `(foo (FIRST list))`, dann kann keine Fehlerprüfung durchgeführt werden, da zum Zeitpunkt des Kompilierens der Typ von `(FIRST list)` nicht bekannt ist.

Aus Geschwindigkeitsgründen wird beim Programmlauf keine Typüberprüfung durchgeführt. Es könnte eingebaut werden, aber dies würde eine kleine Verlangsamung bewirken, die nicht wirklich notwendig ist, da ein falscher Typ früher oder später in einem Typfehler endet.

Typdeklarierer für Datensatzzeiger haben eine andere nützliche Eigenschaft. Wird eine Variable als Datensatzzeiger auf eine Tabelle belegt, dann kann auf alle Felder dieser Tabelle mit dem Variablennamen statt des Tabellennamens im Feldpfad zugegriffen werden. Hat man z.B. eine Tabelle `Foo` mit einem Feld `Bar` und man definiert eine Variable `foo` als

```
(LET (foo:Foo)
```

dann kann man das Feld `Bar` des dritten Datensatzes mit

```
(SETQ foo (RECORD Foo 3)) (PRINT foo.Bar)
```

ausgeben.

Zu beachten ist in Select-from-where Ausdrücken, daß die Variablen in der FROM-Liste automatisch vom Typ des Datensatzzeigers des zugeordneten Tabelle sind.

## 15.27 Aufbau von Ausdrücken

Der Aufbau von Ausdrücken ist von sehr großer Bedeutung, um zu verstehen, was ein Programm tut.

Dieser Abschnitt beschreibt die Semantik, abhängig vom Aufbau der Ausdrücke:

*(func [expr ...])*

Errechnet *expr ...* und ruft dann die Funktion *func* (Aufruf mit Wert) auf. Liefert den Rückgabewert der aufgerufenen Funktion. In MUIbase gibt es einige nicht-strikte Funktionen, z.B. `AND`, `OR` und `IF`. Diese Funktionen müssen nicht zwingend alle Ausdrücke errechnen. Mehr zu nicht-strikten Funktionen, siehe Abschnitt 15.3.2 [Lisp syntax], Seite 79, Abschnitt 15.8.1 [AND], Seite 99, Abschnitt 15.8.2 [OR], Seite 99, and Abschnitt 15.5.7 [IF], Seite 88.

*([expr ...])*

Errechnet *expr ...* und liefert den Wert des letzten Ausdrucks (siehe Abschnitt 15.5.1 [PROGN], Seite 86). Ein leerer Ausdruck `()` wird zu `NIL`.

*Table*

Liefert den Programmdatensatzzeiger der gegebenen Tabelle.

*Table\**

Liefert den Datensatzzeiger der Benutzeroberfläche von der gegebenen Tabelle.

*AttrPath*

Liefert den Inhalt des gegebenen Feldes. Der Feldpfad legt fest, welcher Datensatz verwendet wird, aus dem der Feldinhalt geholt wird. Zum Beispiel benutzt `'Table.Attribute'` den Programmdatensatzzeiger von `'Table'`, um den Wert des Feldes zu ermitteln; `'Table.ReferenceAttribute.Attribute'` verwendet den Programmdatensatzzeiger von `'Table'`, um den Wert des Beziehungsfeldes zu ermitteln (der ein Datensatzzeiger ist) und verwendet diesen Datensatz, um den Wert von `'Attribute'` zu erhalten.

*localvar*

Liefert den Inhalt der lokalen Variable. Lokale Variablen können mit z.B. `LET` (siehe Abschnitt 15.5.3 [LET], Seite 86) definiert werden.

*localvar.AttrPath*

Verwendet den Datensatzzeiger von *localvar*, um den Wert des gegebenen Feldes zu ermitteln.

## 15.28 Auslösefunktionen

Zum automatischen Ausführen von MUIbase-Programmen können Auslösefunktionen für Projekte, Tabellen und Felder festgelegt werden, wie in bestimmten Fällen aufgerufen werden. Dieser Abschnitt beschreibt alle vorhandenen Auslösemöglichkeiten.

### 15.28.1 onOpen

Nach dem Öffnen eines Projekts durchsucht MUIbase das Programm des Projekts nach einer Funktion mit dem Namen `onOpen`. Existiert eine solche Funktion, dann wird diese ohne Parameter aufgerufen.

#### Beispiel

```
(DEFUN onOpen ()
  (ASKBUTTON NIL "Danke für das Öffnen!" NIL NIL)
)
```

Siehe auch `onClose`, `onChange`, Beispielprojekt `'Trigger.mb'`.

### 15.28.2 onClose

Bevor ein Projekt geschlossen wird, durchsucht MUIbase das Programm des Projekts nach einer Funktion mit dem Namen `onClose`. Existiert eine solche Funktion, dann wird diese ohne Parameter aufgerufen. In der jetzigen Version wird der Rückgabewert der Funktion ignoriert und das Projekt unabhängig vom Rückgabewert geschlossen.

Wurden in der Funktion `onClose` Änderungen am Projekt durchgeführt, dann fragt MUIbase nach, ob das Projekt zuerst gespeichert werden soll, bevor das Projekt geschlossen wird. Wird der Menüpunkt `'Projekt - Speichern & Schließen'` zum Schließen des Projekts aufgerufen, dann wird die Auslösefunktion aufgerufen, bevor das Projekt gespeichert wird, so daß die Änderungen automatisch gespeichert werden.

## Beispiel

```
(DEFUN onClose ()
  (ASKBUTTON NIL "Auf Wiedersehen!" NIL NIL)
)
```

Siehe auch `onOpen`, `onChange`, Beispielprojekt `'Trigger.mb'`.

### 15.28.3 onChange

Wann immer der Benutzer eine Änderung am Projekt durchführt oder nach dem Speichern eines Projekts, durchsucht MUIbase das Programm des Projekts nach einer Funktion mit dem Namen `onChange`. Existiert eine solche Funktion, dann wird diese ohne Parameter aufgerufen. Dies kann verwendet werden, um die Anzahl der Änderungen zu erfassen, die ein Benutzer an diesem Projekt durchgeführt hat.

## Beispiel

```
(DEFUN onChange ()
  (SETQ Control.NumChanges (CHANGES))
)
```

Im obigen Beispiel könnte `'Control.NumChanges'` ein virtuelles Feld sein, das in einer `'Nur-ein-Datensatz'`-Tabelle zum Anzeigen der Anzahl von Projektänderungen verwendet wird.

Siehe auch `onOpen`, `onClose`, Beispielprojekt `'Trigger.mb'`.

### 15.28.4 Auslösefunktion Neu

Sobald der Benutzer einen neuen Datensatz durch Auswählen der Menüpunkte `'Neuer Datensatz'` oder `'Datensatz kopieren'` anlegen möchte und die Auslösefunktion `'Neu'` für diese Tabelle auf eine MUIbase-Funktion gesetzt wurde, dann wird diese Auslösefunktion ausgeführt. Die Auslösefunktion für `'Neu'` kann im Tabellenfenster (siehe Abschnitt 14.1.1 [Creating tables], Seite 62) gesetzt werden.

Die Auslösefunktion erhält `NIL` oder einen Datensatzzeiger als ersten und einzigen Parameter. `NIL` bedeutet, daß der Benutzer einen neuen Datensatz anlegen möchte und ein Datensatzzeiger zeigt an, daß der Benutzer einen Datensatz eine Kopie dieses Datensatzes anlegen will. Hat die Auslösefunktion mehr als einen Parameter, dann werden diese mit `NIL` vorbelegt. Die Auslösefunktion sollte nun einen neuen Datensatz mit `NEW` (siehe Abschnitt 15.17.1 [NEW], Seite 134) anlegen. Der Rückgabewert der Auslösefunktion wird ausgewertet. Ist er ein Datensatzzeiger, dann wird dieser Datensatz angezeigt.

Die Auslösefunktion `'Neu'` wird auch aufgerufen, wenn ein MUIbase-Programm die Funktion `NEW*` (siehe Abschnitt 15.17.2 [NEW\*], Seite 135) aufruft.

## Beispiel einer Auslösefunktion Neu

```
(DEFUN newRecord (init)
  (PROG1
    (NEW Table init)
    ...
  )
)
```

Siehe auch NEW\*, Delete trigger.

### 15.28.5 Auslösefunktion Löschen

Sobald der Benutzer einen Datensatz durch Auswählen des Menüpunkts ‘**Datensatz löschen**’ löschen möchte und die Auslösefunktion ‘**Löschen**’ für diese Tabelle auf eine MUIbase-Funktion gesetzt wurde, dann wird diese Auslösefunktion ausgeführt. Die Auslösefunktion für ‘**Löschen**’ kann im Tabellenfenster (siehe Abschnitt 14.1.1 [Creating tables], Seite 62) gesetzt werden.

Die Auslösefunktion erhält einen booleschen Parameter als einzigen Parameter. Ist er nicht NIL, dann sollte die Funktion nachfragen, ob der Benutzer wirklich diesen Datensatz löschen möchte. Wenn er es möchte, dann sollte die Funktion DELETE (siehe Abschnitt 15.17.3 [DELETE], Seite 135) zum Löschen des Datensatzes aufrufen.

Die Auslösefunktion ‘Delete’ wird auch aufgerufen, wenn ein MUIbase-Programm die Funktion DELETE\* (siehe Abschnitt 15.17.4 [DELETE\*], Seite 136) aufruft..

## Beispiel einer Auslösefunktion Löschen

```
(DEFUN deleteRecord (requester)
  (DELETE Table requester)
)
```

Siehe auch DELETE\*, New trigger.

### 15.28.6 Comparison function

Um eine Sortierung von Datensätzen einer Tabelle zu definieren, kann eine Vergleichsfunktion verwendet werden, die zwei Datensatzzeiger als Argumente erhält und eine Ganzzahl zurückliefert, die das Sortierverhältnis der beiden Datensätze anzeigt. Die Vergleichsfunktion sollte einen Wert kleiner 0 liefern, wenn ihr erstes Argument kleiner ist als das zweite; 0, wenn sie gleich sind und einen Wert größer 0, wenn das erste Argument größer ist als das zweite.

Angenommen, man hat eine Tabelle ‘**Persons**’ mit dem Zeichenkettenfeld ‘**Name**’, dann könnte man folgende Funktion zum Vergleich zweier Datensätze verwenden:

```
(DEFUN cmpPersons (rec1:Persons rec2:Persons)
  (CMP rec1.Name rec2.Name)
```



)

Dies wird alle Datensätze bezüglich dem Feld 'Name' sortieren, wobei Zeichengrößen unterschieden werden. Anzumerken ist, daß Sortieren anhand einer Felderliste nicht das gleiche Ergebnis liefert, da der Zeichenkettenvergleich zeichengrößenunabhängig durchgeführt wird.

Mit einer Vergleichsfunktion lassen sich sehr komplexe Sortierungen definieren. Man achte jedoch darauf, keine rekursiven Funktionsaufrufe zu erzeugen, die sich selbst aufrufen. MUIbase wird seine Programmausführung anhalten und dies mit einer Fehlermeldung quittieren, sollte so etwas versucht werden. Auch sollten keine Befehle verwendet werden, die Seiteneffekte erzeugen könnten, wie z.B. einen Wert einem Feld zuweisen.

Wird eine Vergleichsfunktion verwendet, dann weiß MUIbase nicht immer, wann es die Datensätze neu zu sortieren hat. Nehmen wir im obigen Beispiel zusätzlich an, daß es die Tabelle 'Toys' mit dem Zeichenkettenfeld 'Name' und das Beziehungsfeld 'Owner' gibt, das auf 'Persons' verweist. Zudem nehmen wir folgende Vergleichsfunktion an:

```
(DEFUN cmpToys (rec1:Toys rec2:Toys)
  (CMP* rec1.Owner rec2.Owner)
)
```

Diese Funktion verwendet die Sortierung von 'Persons', um die Sortierung der Datensätze festzustellen, so daß die Datensätze von 'Toys' analog der Sortierung von 'Persons' sortiert sind.

Ändert nun der Benutzer einen Datensatz in der Tabelle 'Persons' und dieser Datensatz erhält eine neue Position, dann müßten auch Datensätze in 'Toys' neu sortiert werden, die auf diesen Datensatz verweisen. MUIbase kennt jedoch diese Abhängigkeit nicht.

Neben der Verwendung des Menüpunktes 'Tabelle - Alle Datensätze neu sortieren' auf die Tabelle 'Toys' zum Neusortieren kann auch ein automatisches Neusortieren implementiert werden, indem folgende Auslösefunktion für das Feld 'Name' der Tabelle 'Persons' festgelegt wird:

```
(DEFUN setName (newValue)
  (SETQ Persons.Name newValue)
  (FOR ALL Toys WHERE (= Toys.Owner Persons) DO
    (SETISSORTED Toys NIL)
  )
  (REORDER Toys)
)
```

Diese Funktion löscht die Sortierzustände aller Datensätze, die auf den aktuellen Datensatz der Tabelle 'Persons' verweisen und sortiert anschließend alle unsortierten Datensätze der Tabelle 'Toys'.

Siehe auch Order, GETISSORTED, SETISSORTED, REORDER, REORDERALL, GETORDERSTR, SETORDERSTR, Demo 'Order.mb'.

### 15.28.7 Auslösefunktion Feld

Im Feldfenster (siehe Abschnitt 14.2.1 [Creating attributes], Seite 64) kann eine Auslösefunktion definiert werden, die immer dann aufgerufen wird, wenn der Benutzer den Inhalt des Feldes ändern möchte.

Wurde eine solche Auslösefunktion definiert und der Benutzer ändert den Wert dieses Feldes, dann wird der Datensatzinhalt nicht automatisch auf den neuen Wert gesetzt. Stattdessen wird der Wert als erster Parameter an die Auslösefunktion übergeben. Die Auslösefunktion kann nun den Wert überprüfen und ihn ablehnen. Um den Wert im Datensatz zu speichern, muß die Funktion `SETQ` verwendet werden.

Die Auslösefunktion sollte das Ergebnis des `SETQ`-Aufrufs (siehe Abschnitt 15.5.4 [`SETQ`], Seite 87) oder den alten Wert des Feldes, wenn sie den neuen Wert abzulehnt, zurückgeben.

Die Auslösefunktion wird auch ausgeführt, wenn ein MUIbase-Programm die Funktion `SETQ*` (siehe Abschnitt 15.5.5 [`SETQ*`], Seite 87) zum Setzen eines Feldwertes aufruft.

### Beispiel einer Auslösefunktion Feld

```
(DEFUN setAmount (amount)
  (IF some-expression
    (SETQ Table.Amount amount)
    (ASKBUTTON NIL "Ungültiger Wert!" NIL NIL)
  )
  Table.Amount ; liefert momentanen Wert zurück
)
```

Siehe auch `SETQ*`

### 15.28.8 Virtuelle Felder programmieren

In MUIbase sind virtuelle Felder besondere Felder, die ihren Inhalt nebenbei berechnen, wenn er benötigt wird. Wird z.B. zu einem anderen Datensatz gewechselt, indem man auf einen der Pfeile in der Pannelleiste einer Tabelle klickt und ein virtuelles Feld in dieser Tabelle das Flag 'Sofort' (siehe Abschnitt 14.3.3 [Attribute object editor], Seite 69) gesetzt hat, dann wird der Wert dieses Feldes berechnet und dargestellt. Zum Berechnen des Wertes wird die Auslösefunktion 'Berechne' des Feldes aufgerufen. Diese Auslösefunktion kann im Feldfenster (siehe Typabhängige Einstellungen<sup>48</sup>). festgelegt werden. Der Rückgabewert dieser Funktion definiert den Wert des virtuellen Feldes. Wurde keine 'Berechne'-Auslösefunktion für ein virtuelles Feld festgelegt, dann ist der Wert des Feldes `NIL`.

Man kann auch die Berechnung eines virtuellen Feldes auslösen, indem man einfach in einem MUIbase-Programm darauf zugreift, so daß man z.B. auf Knopfdruck zum Berechnen des Wertes eines virtuellen Feldes wie im folgenden nur eine Funktion für den Knopf festlegen muß:

```
(DEFUN buttonHook ()
```

---

<sup>48</sup> Anm.d.Übersetzers: Hier ist wegen Einschränkungen bei Texinfo leider kein Verweis möglich.

```
    virtual-attr  
)
```

Man kann auch den virtuellen Wert auf jeden Wert setzen, indem man die Funktion **SETQ** verwendet:

```
(SETQ virtual-attr expr)
```

Wird jedoch nach dem **SETQ**-Aufruf auf das virtuelle Feld zugegriffen, dann wird der Wert des virtuellen Feldes neu berechnet.

Der Wert eines virtuellen Feldes wird nicht zwischengespeichert, da nicht einfach festzustellen ist, wann der Wert neu berechnet werden muß und wann nicht. Daher sollte man auf virtuelle Felder möglichst sparsam zugreifen und den Wert in lokalen Variablen für die weitere Verwendung selbst zwischenspeichern.

Für ein Beispiel, wie virtuelle Felder benutzt werden, sehe man sich das Beispielprojekt **‘Movie.mb’** an.

Siehe auch Virtual, Beispielprojekt **‘Movie.db’**.

## ABConvert

Um ein AmigaBase-Projekt in ein MUIbase-Projekt zu konvertieren, existiert ein kleines Hilfsprogramm namens *ABConvert*. Es lädt ein AmigaBase-Projekt, das mit AmigaBase Version 2.0 oder höher erstellt wurde, und speichert es als MUIbase-Projekt. Projekte älterer AmigaBase-Versionen müssen erst in AmigaBase Version 2.4 eingeladen und gespeichert und dann erst konvertiert werden.

Ein ausführbares Programm für Solaris liegt im Verzeichnis *solaris*. Sollte man jemals Probleme mit zu wenig Speicher auf dem Amiga haben, dann lasse man sich *ABConvert* auf einem Solaris-Rechner laufen und die Speicherprobleme sollten dann nicht mehr auftreten.

Um die Konvertierung zu starten, gibt man *ABConvert ab-Datei mb-Datei* ein, wobei *ab-Datei* ein vorhandenes AmigaBase-Projekt und *mb-Datei* das neue zu erzeugende MUIbase-Projekt ist.

Da MUIBase eine komplett neue Datenbankanwendung ist, werden nur die Strukturen und Datensätze von AmigaBase konvertiert. Programme, Filter, Sortierungen, etc. müssen in MUIbase neu eingerichtet werden. Zum Konvertieren von AmigaBase-Programmen in solche für MUIbase, ist es am sinnvollsten, daß man sich alle Programme von AmigaBase ausdruckt und dann zum Konvertieren der Programme die Hilfen verwendet, die in der Textdatei '*PortingABPrograms*' beschrieben sind.

Wichtig ist noch, daß Ausdrücke in AmigaBase in MUIbase umbenannt wurden, z.B. wird ein AmigaBase-*'Datensatz'* jetzt *'Tabelle'* genannt, eine *'Variable'* heißt nun *'Feld'* und eine *'Kartei'* ist jetzt ein *'Datensatz'*.

MUIbase ist relational. Daher wird die Hierarchie eines AmigaBase-Projekts in Tabellen umgewandelt. Dies geschieht durch Ergänzen eines Beziehungsfeldes zu jeder Tabelle (ausgenommen die "Wurzel"-Tabelle), die auf ihre "Vater"-Tabellen zeigen.

## Anerkennung

Dank geht an:

- Mats Granstrom für das lang andauernde Beta-Testen von MUIbase; seine Ideen, es zu verbessern und für das Schreiben des Tutorials. Mats schreibt manchmal sehr amüsante eMails. Es ist immer ein Vergnügen, sie zu lesen.
- Ralph Reuchlein (Ralphie) für riesige Fehlerreports und -listen, Unmengen von eMails mit Ideen und Verbesserungen, und seine unbegrenzte Zeit, diese zu diskutieren. Ohne seine Ideen (wie z.B. die #-Direktiven zum Programmieren) wäre MUIbase nicht das, was es heute ist. Ralphie baute auch die MUIbase Homepage  
`http://www.amigaworld.com/support/muibase/index.html`  
auf und verwaltet sie.
- Thomas Fricke für das Beta-Testen und seine Grafiken, um das Erscheinungsbild von MUIbase zu verbessern. Er ist derjenige, der die MUIbase- und Projekt-Icons, die Fenster-Offen/-Geschlossen-Symbole und andere gemalt hat.
- André Schenk und Klaus Gessner für das Beta-Testen und ihr Wissen über relationale Datenbanken und SQL.
- Allan Odgaard für seine TextEditor-Klasse und seine prima Unterstützung, es für MUIbase zu verbessern.
- Oliver Roberts zum Beta-Testen und einige sehr gute Fehlerreports, und für F1GP-Ed (siehe <http://www.nanunanu.org/~oliver/>).
- Petri Nordlund für die Erlaubnis, seine Registrationsdateien von Executive für MUIbase zu verwenden. Wer Executive nicht kennt, der sollte es unbedingt mal ausprobieren. Es ist im Aminet verfügbar.
- Henning Thielemann für Ideen und Betatesten.

## Autor

MUIbase wurde entwickelt von:

Steffen Gutmann  
Orleanstr. 47  
81667 München  
GERMANY

Email: [gutmann@ieee.org](mailto:gutmann@ieee.org)

Die über 10000zeilige Dokumentation im Texinfo-Format übersetzte mit viel Zeit und Geduld vom Englischen ins Deutsche:

Ralph Reuchlein  
Eibseestr. 18c  
86163 Augsburg  
GERMANY

Email: [muibase@rripley.de](mailto:muibase@rripley.de)

# Funktionsverzeichnis

## #

#define .....	75
#elif .....	77
#else .....	78
#endif .....	78
#if .....	77
#ifdef .....	77
#ifndef .....	77
#include .....	76
#undef .....	76

## \*

* .....	103
---------	-----

## -

- .....	102
---------	-----

## /

/ .....	103
---------	-----

## =

= .....	100
=* .....	100

## +

+ .....	101
---------	-----

## >

> .....	100
>* .....	100
>= .....	100
>=* .....	100

## <

< .....	100
<* .....	100
<= .....	100
<=* .....	100
<> .....	100
<>* .....	100

## 1

1- .....	103
1+ .....	102

## A

ABS .....	104
AND .....	99
APPEND .....	121

ASC .....	113
ASKBUTTON .....	126
ASKCHOICE .....	124
ASKCHOICESTR .....	125
ASKDIR .....	123
ASKFILE .....	123
ASKINT .....	124
ASKMULTI .....	127
ASKOPTIONS .....	125
ASKSTR .....	123
ATTRNAME .....	138

## C

CASE .....	88
CHANGES .....	148
CHR .....	113
CMP .....	101
CMP* .....	101
CONCAT .....	111
CONCAT2 .....	112
COND .....	89
CONS .....	119
CONSP .....	94
COPYREC .....	138
COPYSTR .....	112

## D

DATE .....	97
DATEP .....	94
DEFUN .....	84
DEFUN* .....	85
DEFVAR .....	85
DELETE .....	135
DELETE* .....	136
DELETEALL .....	136
DIRNAME .....	151
DIV .....	103
DO .....	91
DOLIST .....	90
DOTIMES .....	89

## E

EDIT .....	148
EDIT* .....	149
ERROR .....	94
EXIT .....	93

## F

FCLOSE .....	129
FEOF .....	131

FERROR .....	131
FFLUSH .....	134
FGETCHAR .....	132
FGETCHARS .....	132
FGETMEMO .....	133
FGETSTR .....	132
FILENAME .....	150
FILLMEMO .....	118
FIRST .....	120
FOPEN .....	129
FOR ALL .....	92
FORMATMEMO .....	118
FPRINTF .....	130
FPUTCHAR .....	133
FPUTMEMO .....	133
FPUTSTR .....	133
FSEEK .....	131
FTELL .....	132
FUNCALL .....	88

## G

GC .....	152
GETDISABLED .....	146
GETFILTERACTIVE .....	142
GETFILTERSTR .....	143
GETISSORTED .....	137
GETLABELS .....	139
GETMATCHFILTER .....	136
GETORDERSTR .....	140
GETWINDOWDISABLED .....	146
GETWINDOWOPEN .....	147

## H

HALT .....	94
------------	----

## I

IF .....	88
INDENTMEMO .....	118
INDEXBRK .....	108
INDEXBRK* .....	108
INDEXSTR .....	108
INDEXSTR* .....	108
INSMIDSTR .....	107
INT .....	96
INTP .....	94

## L

LAST .....	120
LEFTSTR .....	106
LEN .....	106
LENGTH .....	119
LET .....	86
LIKE .....	113

LINE .....	116
LINES .....	117
LIST .....	119
LISTP .....	94
LISTTOMEMO .....	117
LOWER .....	113

## M

MAPFIRST .....	121
MAX .....	104
MAXLEN .....	139
MEMO .....	96
MEMOP .....	94
MEMOTOLIST .....	117
MESSAGE .....	151
MIDSTR .....	107
MIN .....	104
MOD .....	104

## N

NEW .....	134
NEW* .....	135
NEXT .....	92
NOT .....	99
NOW .....	151
NTH .....	120
NULLP .....	94

## O

onChange .....	157
onClose .....	156
onOpen .....	156
OR .....	99

## P

PRINT .....	130
PRINTF .....	130
PROG1 .....	86
PROGN .....	85
PROJECTNAME .....	148

## R

RANDOM .....	105
REAL .....	97
REALP .....	94
RECNUM .....	138
RECORD .....	144
RECORDS .....	144
RECP .....	94
REMCHARS .....	110
REORDER .....	141
REORDERALL .....	142
REPLACESTR .....	110



REST .....	120
RETURN .....	93
REVERSE .....	121
RIGHTSTR .....	106
RINDEXBRK .....	109
RINDEXBRK* .....	110
RINDEXSTR .....	109
RINDEXSTR* .....	109
ROUND .....	105

## S

SELECT .....	144
SETCURSOR .....	146
SETDISABLED .....	146
SETFILTERACTIVE .....	142
SETFILTERSTR .....	143
SETISSORTED .....	137
SETLABELS .....	139
SETMATCHFILTER .....	137
SETMIDSTR .....	107
SETORDERSTR .....	141
SETQ .....	87
SETQ* .....	87
SETWINDOWDISABLED .....	147
SETWINDOWOPEN .....	147
SORTLIST .....	122
SORTLISTGT .....	122

SPRINTF .....	114
STAT .....	150
stdout .....	129
STR .....	95
STRP .....	94
SYSTEM .....	149

## T

TABlename .....	140
TACKON .....	150
TIME .....	98
TIMEP .....	94
TODAY .....	151
TRIMSTR .....	110
TRUNC .....	105

## U

UPPER .....	112
-------------	-----

## V

VIEW .....	149
VIEW* .....	149

## W

WORD .....	111
WORDS .....	111

# Stichwortverzeichnis

## 1

1:1-Beziehungen .....	26
1:n-Beziehungen .....	27

## A

ABConvert .....	161
Abfragebeispiele .....	59
Abfrageeditor .....	57
Abfragen ausdrucken .....	58
Aktive Objekte .....	42
Aktive Tabellen .....	42
Anerkennung .....	163
Anzeigebereich .....	67
Anzeigeverwaltung .....	67
Aufbau von Ausdrücken .....	155
Ausgabedatei .....	39
Auslösefunktion Feld .....	159
Auslösefunktion Löschen .....	158
Auslösefunktion Neu .....	157
Auslösefunktionen .....	156
Auswahlfelder .....	23
Auswahltexteditor .....	65
Autor .....	164

## B

Beenden bestätigen .....	38
Befehle definieren .....	84
Befehlsaufbau .....	83
Beispiel-Importdatei .....	54
Benutzereingabefunktionen .....	123
Benutzerschnittstelle .....	29
Beschädigte Datenbank .....	34
BetterString .....	5
Beziehungen .....	26
Beziehungsfelder .....	24
Bild für leere Anzeige .....	41
Bildeditor .....	72
Bilder .....	30
Bildfelder .....	22
Boolesche Felder .....	22
Boolesche Funktionen .....	99

## C

Comparison function .....	158
Custom classes .....	5

## D

Dankeschön .....	163
Dateiformat .....	32

Dateiformat für Import und Export .....	54
Dateinamenfelder .....	22
Datenabfragen .....	56
Datensatzbearbeitung .....	41
Datensätze .....	21
Datensätze auslagern .....	35
Datensätze durchforsten .....	45
Datensätze exportieren .....	55
Datensätze importieren .....	55
Datensätze kopieren .....	42
Datensätze löschen bestätigen .....	36
Datensätze vervielfältigen .....	42
Datensatzfilter .....	46
Datensatzfunktionen .....	134
Datensatzinhalte ansprechen .....	80
Datensatzspeicher .....	36
Datentypen zum Programmieren .....	81
Datumsfelder .....	23
Debuginformation .....	39
Delete record .....	44

## E

E/A-Funktionen .....	128
Eingabe von Auswahlwerten .....	43
Eingabe von Beziehungswerten .....	44
Eingabe von booleschen Werten .....	43
Eingabe von Datumswerten .....	43
Eingabe von NIL-Werten .....	44
Eingabe von Zeitwerten .....	43
Eins-zu-Eins-Beziehungen .....	26
Eins-zu-Mehrfach-Beziehungen .....	27
Einstellungen .....	36
Einstellungsdatei .....	40
Externe Klassen .....	5
Externer Anzeiger .....	38
Externer Editor .....	37
Externer Editor zum Programmieren .....	36

## F

Felder .....	21
Felder ändern .....	65
Felder erstellen .....	63
Felder kopieren .....	65
Felder löschen .....	66
Felder sortieren .....	66
Felderverwaltung .....	63
Feldfunktionen .....	138
Feldobjekte .....	30
Feldobjekteditor .....	68
Feldtypen .....	21

Feldtypen (Tabelle) .....	25
Fenster .....	29
Fenstereditor .....	73
Filter .....	45
Filter ändern .....	46
Filterausdruck .....	46
Filterbeispiele .....	47
Fließkommazahlfelder .....	22
Formate .....	37
Fremde Software .....	4
Funktionale Parameter .....	153
Funktionen für mehrzeilige Texte .....	116

## G

Ganzzahlfelder .....	22
Gewichtungsobjekte .....	31
Gruppen .....	31
Gruppeneditor .....	73

## H

Hauptfenster .....	29
--------------------	----

## I

Icon erstellen .....	36
Icons .....	5
Import und Export .....	53
Importdatei-Beispiel .....	54
Information .....	32
Integrität der Daten prüfen .....	34
Interne Fehler in der Datenbank .....	34

## K

Karteikarten-Gruppen .....	31
Karteikarten-Gruppeneditor .....	73
Keine Sortierung .....	49
Knöpfe .....	24
Konstanten .....	82
Kontextmenü vom mehrzeiligen Textfeld .....	43
Kopieren von MUIbase .....	1

## L

Laden der Einstellungen .....	40
Lisp-Aufbau .....	79
Listenfunktionen .....	119

## M

Masken .....	29
Mathematik-Funktionen .....	101
Mehrfach-zu-Mehrfach-Beziehungen .....	27
mehrzeilige Textfelder .....	23
MUI .....	4, 40
MUI-Voreinsteller .....	40
MUIbase beenden .....	8

MUIbase installieren .....	7
MUIbase starten .....	8

## N

n:m-Beziehungen .....	27
Namenskonventionen for program symbols .....	80
Neu sortieren aller Datensätze .....	51
Neue Tabelle .....	62
Neuer Datensatz .....	42
Neues Feld .....	63

## O

Oberflächenfunktionen .....	146
onChange .....	157
onClose .....	156
onOpen .....	156

## P

Paneleditor .....	68
Panels .....	30
Popup-Knöpfe in die TAB-Kette .....	38
Programm-Ausgabedatei .....	39
Programm-Debuginformation .....	39
Programm-Einfügedateienverzeichnis .....	39
Programmarten .....	80
Programmeditor .....	75
Programmieren .....	74
Programmiersprache .....	78
Programmsteuerungsfunktionen .....	85
Projekt entfernen .....	34
Projekt löschen .....	32
Projekt öffnen .....	33
Projekt schließen .....	34
Projekt speichern .....	33
Projektabhängige Einstellungen .....	39
Projekte .....	20
Projektfunktionen .....	148

## R

Referenzfilter .....	47
Registration .....	1
Relationsoperatoren .....	100
Reorganisieren .....	33

## S

Select-from-where Abfragen .....	57
Sortieren .....	48
Sortierung ändern .....	50
Speichern der Einstellungen .....	40
Speicherverbrauch .....	25
Standardprogramm .....	37
Struktur ausdrucken .....	74
Struktureditor .....	61

Suchen .....	51
Suchfenster .....	52
Suchmusterbeispiele .....	52
Systemfunktionen .....	148

## T

Tabellarische Ansicht .....	56
Tabellen .....	20
Tabellen ändern .....	63
Tabellen erstellen .....	62
Tabellen löschen .....	63
Tabellen sortieren .....	63
Tabellenfunktionen .....	140
Tabellenverwaltung .....	62
Texteditor .....	72
TextEditor .....	5
Textobjekte .....	30
Tutorial .....	9
Typabhängige Einstellungen .....	64
Typabhängige Einstellungen für Felder .....	69
Typaussagen .....	94
Typdeklarierer .....	154
Typumwandlungsfunktionen .....	95

## U

Umschichten & Speichern bestätigen .....	38
--	----

## V

Vergleichsfunktionen .....	100
Verteilung .....	3
Verzichtserklärung .....	4
Virtuelle Felder programmieren .....	160
virtuelles Feld .....	24
Vordefinierte Konstanten .....	152
Vordefinierte Variablen .....	152
Vorgabedatensatz .....	21
Vorverarbeitung .....	75
Vorwärts/Rückwärts suchen .....	52

## W

Warum lisp? .....	78
Wie registriert man .....	1

## Z

Zeichenketten .....	22
Zeichenkettenfunktionen .....	106
Zeichensatznamenfelder .....	22
Zeitfelder .....	23
Zwischenraumeditor .....	72
Zwischenraumobjekte .....	31

# Inhaltsverzeichnis

<b>1</b>	<b>Kopierbestimmungen von MUIbase .....</b>	<b>1</b>
1.1	Registration .....	1
1.1.1	Registrationsgebühr .....	1
1.1.2	Zahlungsmethoden .....	2
1.1.3	Bestellen .....	2
1.1.4	Zustellungsmethode .....	3
1.2	Verteilung .....	3
1.3	Verzichtserklärung .....	4
1.4	MUI .....	4
1.5	BetterString & TextEditor .....	5
1.6	Zusätzliche Custom classes .....	5
1.7	Icons .....	5
<b>2</b>	<b>Willkommen zu MUIbase .....</b>	<b>6</b>
<b>3</b>	<b>Einführung .....</b>	<b>7</b>
3.1	MUIbase installieren .....	7
3.1.1	Benötigte Hardware .....	7
3.1.2	Benötigte Software .....	7
3.1.3	Installation starten .....	7
3.1.4	Schlüsseldatei .....	8
3.2	MUIbase starten .....	8
3.3	MUIbase beenden .....	8
<b>4</b>	<b>Tutorial .....</b>	<b>9</b>
4.1	Wie MUIbase arbeitet .....	9
4.2	Ein Projekt beginnen: Der Struktureditor .....	9
4.3	Hinzufügen einer Tabelle .....	9
4.4	Hinzufügen eines Feldes .....	10
4.5	Darstellen des Projekts .....	10
4.6	Hinzufügen von zwei Datensatzbeziehungen .....	12
4.7	Datensätze hinzufügen .....	12
4.8	Filter .....	13
4.9	Abfragen .....	13
4.10	Hinzufügen einer Tabelle mit einem mehrzeiligen Text und einem Knopf .....	14
4.11	MUIbase programmieren, um einen Stammbaum zu erzeugen ....	15
4.12	MUIbase programmieren, um die Kinder einer Person aufzulisten .....	17
<b>5</b>	<b>Grundlagen .....</b>	<b>20</b>
5.1	Projekte .....	20
5.2	Tabellen .....	20
5.3	Datensätze .....	21
5.4	Felder .....	21
5.5	Feldtypen .....	21
5.5.1	Zeichenketten .....	22
5.5.2	Ganzzahlfelder .....	22

5.5.3	Fließkommazahlfelder .....	22
5.5.4	Boolesche Felder .....	23
5.5.5	Auswahlfelder .....	23
5.5.6	Datumsfelder .....	23
5.5.7	Zeitfelder .....	23
5.5.8	mehrzeilige Textfelder .....	24
5.5.9	Beziehungsfelder .....	24
5.5.10	virtuelles Feld .....	24
5.5.11	Knöpfe .....	25
5.6	Tabelle der Feldtypen .....	25
5.7	Speicherverbrauch .....	26
5.8	Beziehungen .....	26
5.8.1	Eins-zu-Eins-Beziehungen .....	26
5.8.2	Eins-zu-Mehrfach-Beziehungen .....	27
5.8.3	Mehrfach-zu-Mehrfach-Beziehungen .....	27
5.9	Benutzerschnittstelle .....	29
5.9.1	Fenster .....	29
5.9.2	Masken .....	30
5.9.3	Panels .....	30
5.9.4	Feldobjekte .....	30
5.9.5	Textobjekte .....	30
5.9.6	Bilder .....	30
5.9.7	Zwischenraumobjekte .....	31
5.9.8	Gruppen .....	31
5.9.9	Gewichtungsobjekte .....	31
5.9.10	Karteikarten-Gruppen .....	31
<b>6</b>	<b>Projekte verwalten .....</b>	<b>32</b>
6.1	Dateiformat .....	32
6.2	Information .....	32
6.3	Projekt löschen .....	32
6.4	Projekt öffnen .....	33
6.5	Projekt speichern .....	33
6.6	Projekt entfernen .....	34
6.7	Projekt schließen .....	34
6.8	Integrität der Daten prüfen .....	34
6.9	Datensätze auslagern .....	35
<b>7</b>	<b>Einstellungen .....</b>	<b>36</b>
7.1	Datensatzspeicher .....	36
7.2	Datensätze löschen bestätigen .....	36
7.3	Externer Editor zum Programmieren .....	36
7.4	Icon erstellen .....	37
7.5	Formate .....	37
7.6	Standardprogramm .....	37
7.7	Externer Editor .....	37
7.8	Externer Anzeiger .....	38
7.9	Popup-Knöpfe in die TAB-Kette .....	38
7.10	Umschichten & Speichern bestätigen .....	38
7.11	Beenden bestätigen .....	38
7.12	Programm-Einfügedateienverzeichnis .....	39
7.13	Programm-Debuginformation .....	39
7.14	Programm-Ausgabedatei .....	39
7.15	Projektabhängige Einstellungen .....	39
7.16	MUI .....	40

7.17	Laden und Speichern der Einstellungen .....	40
7.18	Bild für leere Anzeige .....	41
<b>8</b>	<b>Datensatzbearbeitung .....</b>	<b>42</b>
8.1	Aktive Objekte .....	42
8.2	Datensätze hinzufügen .....	42
8.3	Datensätze verändern .....	42
8.4	Zeichenkettenfelder mit einem Popup-Knopf .....	42
8.5	Eingabe von booleschen Werten .....	43
8.6	Eingabe von Auswahlwerten .....	43
8.7	Eingabe von Datumswerten .....	43
8.8	Eingabe von Zeitwerten .....	43
8.9	Kontextmenü vom mehrzeiligen Textfeld .....	43
8.10	Eingabe von Beziehungswerten .....	44
8.11	Eingabe von NIL-Werten .....	44
8.12	Datensätze löschen .....	45
8.13	Datensätze durchforsten .....	45
<b>9</b>	<b>Filter .....</b>	<b>46</b>
9.1	Datensatzfilter .....	46
9.1.1	Filterausdruck .....	46
9.1.2	Filter ändern .....	46
9.1.3	Filterbeispiele .....	47
9.2	Referenzfilter .....	47
<b>10</b>	<b>Sortieren .....</b>	<b>49</b>
10.1	Keine Sortierung .....	49
10.2	Sortieren nach Feldern .....	49
10.3	Sortieren nach einer Funktion .....	50
10.4	Sortierung ändern .....	50
10.5	Neu sortieren aller Datensätze .....	51
<b>11</b>	<b>Suchen .....</b>	<b>52</b>
11.1	Suchfenster .....	52
11.2	Vorwärts/Rückwärts suchen .....	52
11.3	Suchmusterbeispiele .....	53
<b>12</b>	<b>Import und Export .....</b>	<b>54</b>
12.1	Dateiformat .....	54
12.2	Beispiel-Importdatei .....	54
12.3	Datensätze importieren .....	55
12.4	Datensätze exportieren .....	55
<b>13</b>	<b>Datenabfragen .....</b>	<b>57</b>
13.1	Select-from-where Abfragen .....	57
13.2	Abfrageeditor .....	58
13.3	Abfragen ausdrucken .....	58
13.4	Abfragebeispiele .....	60

<b>14</b>	<b>Struktureditor</b>	<b>62</b>
14.1	Tabellenverwaltung	62
14.1.1	Tabellen erstellen	62
14.1.2	Tabellen ändern	63
14.1.3	Tabellen löschen	63
14.1.4	Tabellen sortieren	63
14.2	Felderverwaltung	63
14.2.1	Felder erstellen	64
14.2.2	Typabhängige Einstellungen	64
14.2.3	Auswahltexteditor	65
14.2.4	Felder kopieren	65
14.2.5	Felder ändern	66
14.2.6	Felder löschen	66
14.2.7	Felder sortieren	67
14.3	Anzeigeverwaltung	67
14.3.1	Anzeigebereich	67
14.3.2	Paneleditor	68
14.3.3	Feldobjekteditor	69
14.3.4	Typabhängige Einstellungen	70
14.3.5	Texteditor	72
14.3.6	Bildeditor	72
14.3.7	Zwischenraumeditor	72
14.3.8	Gruppeneditor	73
14.3.9	Karteikarten-Gruppeneditor	73
14.3.10	Fenstereditor	74
14.4	Struktur ausdrucken	74
<b>15</b>	<b>MUIbase programmieren</b>	<b>75</b>
15.1	Programmeditor	75
15.2	Vorverarbeitung	75
15.2.1	#define	76
15.2.2	#undef	76
15.2.3	#include	76
15.2.4	#if	77
15.2.5	#ifdef	77
15.2.6	#ifndef	77
15.2.7	#elif	77
15.2.8	#else	78
15.2.9	#endif	78
15.3	Programmiersprache	78
15.3.1	Warum lisp?	79
15.3.2	Lisp-Aufbau	79
15.3.3	Programmarten	80
15.3.4	Namenskonventionen	80
15.3.5	Datensatzinhalte ansprechen	80
15.3.6	Datentypen zum Programmieren	82
15.3.7	Konstanten	82
15.3.8	Befehlsaufbau	83
15.4	Befehle definieren	84
15.4.1	DEFUN	84
15.4.2	DEFUN*	85
15.4.3	DEFVAR	85
15.5	Programmsteuerungsfunktionen	85
15.5.1	PROGN	86
15.5.2	PROG1	86
15.5.3	LET	86



15.5.4	SETQ	87
15.5.5	SETQ*	87
15.5.6	FUNCALL	88
15.5.7	IF	88
15.5.8	CASE	89
15.5.9	COND	89
15.5.10	DOTIMES	90
15.5.11	DOLIST	90
15.5.12	DO	91
15.5.13	FOR ALL	92
15.5.14	NEXT	92
15.5.15	EXIT	93
15.5.16	RETURN	93
15.5.17	HALT	94
15.5.18	ERROR	94
15.6	Typaussagen	94
15.7	Typumwandlungsfunktionen	95
15.7.1	STR	95
15.7.2	MEMO	96
15.7.3	INT	96
15.7.4	REAL	97
15.7.5	DATE	97
15.7.6	TIME	98
15.8	Boolesche Funktionen	99
15.8.1	AND	99
15.8.2	OR	99
15.8.3	NOT	99
15.9	Vergleichsfunktionen	100
15.10	Relationsoperatoren	100
15.10.1	CMP	101
15.10.2	CMP*	101
15.11	Mathematik-Funktionen	101
15.11.1	Werte addieren	101
15.11.2	Werte subtrahieren	102
15.11.3	1+	103
15.11.4	1-	103
15.11.5	Werte multiplizieren (*)	103
15.11.6	Werte dividieren	103
15.11.7	DIV	104
15.11.8	MOD	104
15.11.9	MAX	104
15.11.10	MIN	104
15.11.11	ABS	104
15.11.12	TRUNC	105
15.11.13	ROUND	105
15.11.14	RANDOM	105
15.12	Zeichenkettenfunktionen	106
15.12.1	LEN	106
15.12.2	LEFTSTR	106
15.12.3	RIGHTSTR	106
15.12.4	MIDSTR	107
15.12.5	SETMIDSTR	107
15.12.6	INSMIDSTR	107
15.12.7	INDEXSTR	108
15.12.8	INDEXSTR*	108
15.12.9	INDEXBRK	108

15.12.10	INDEXBRK*	109
15.12.11	RINDEXSTR	109
15.12.12	RINDEXSTR*	109
15.12.13	RINDEXBRK	109
15.12.14	RINDEXBRK*	110
15.12.15	REPLACESTR	110
15.12.16	REMCHARS	110
15.12.17	TRIMSTR	110
15.12.18	WORD	111
15.12.19	WORDS	111
15.12.20	CONCAT	111
15.12.21	CONCAT2	112
15.12.22	COPYSTR	112
15.12.23	UPPER	112
15.12.24	LOWER	113
15.12.25	ASC	113
15.12.26	CHR	113
15.12.27	LIKE	113
15.12.28	SPRINTF	114
15.13	Funktionen für mehrzeilige Texte	116
15.13.1	LINE	116
15.13.2	LINES	117
15.13.3	MEMOTOLIST	117
15.13.4	LISTTOMEMO	117
15.13.5	FILLMEMO	118
15.13.6	FORMATMEMO	118
15.13.7	INDENTMEMO	118
15.14	Listenfunktionen	119
15.14.1	CONS	119
15.14.2	LIST	119
15.14.3	LENGTH	120
15.14.4	FIRST	120
15.14.5	REST	120
15.14.6	LAST	120
15.14.7	NTH	121
15.14.8	APPEND	121
15.14.9	REVERSE	121
15.14.10	MAPFIRST	121
15.14.11	SORTLIST	122
15.14.12	SORTLISTGT	122
15.15	Benutzereingabefunktionen	123
15.15.1	ASKFILE	123
15.15.2	ASKDIR	123
15.15.3	ASKSTR	124
15.15.4	ASKINT	124
15.15.5	ASKCHOICE	124
15.15.6	ASKCHOICESTR	125
15.15.7	ASKOPTIONS	126
15.15.8	ASKBUTTON	126
15.15.9	ASKMULTI	127
15.16	E/A-Funktionen	129
15.16.1	FOPEN	129
15.16.2	FCLOSE	129
15.16.3	stdout	129
15.16.4	PRINT	130
15.16.5	PRINTF	130

	15.16.6	FPRINTF .....	130
	15.16.7	FERROR .....	131
	15.16.8	FEOF .....	131
	15.16.9	FSEEK .....	131
	15.16.10	FTELL .....	132
	15.16.11	FGETCHAR .....	132
	15.16.12	FGETCHARS .....	132
	15.16.13	FGETSTR .....	132
	15.16.14	FGETMEMO .....	133
	15.16.15	FPUTCHAR .....	133
	15.16.16	FPUTSTR .....	133
	15.16.17	FPUTMEMO .....	133
	15.16.18	FFLUSH .....	134
15.17		Datensatzfunktionen .....	134
	15.17.1	NEW .....	134
	15.17.2	NEW* .....	135
	15.17.3	DELETE .....	135
	15.17.4	DELETE* .....	136
	15.17.5	DELETEALL .....	136
	15.17.6	GETMATCHFILTER .....	136
	15.17.7	SETMATCHFILTER .....	137
	15.17.8	GETISSORTED .....	137
	15.17.9	SETISSORTED .....	137
	15.17.10	RECNUM .....	138
	15.17.11	COPYREC .....	138
15.18		Feldfunktionen .....	138
	15.18.1	ATTRNAME .....	138
	15.18.2	MAXLEN .....	139
	15.18.3	GETLABELS .....	139
	15.18.4	SETLABELS .....	139
15.19		Tabellenfunktionen .....	140
	15.19.1	TABlename .....	140
	15.19.2	GETORDERSTR .....	140
	15.19.3	SETORDERSTR .....	141
	15.19.4	REORDER .....	142
	15.19.5	REORDERALL .....	142
	15.19.6	GETFILTERACTIVE .....	142
	15.19.7	SETFILTERACTIVE .....	143
	15.19.8	GETFILTERSTR .....	143
	15.19.9	SETFILTERSTR .....	143
	15.19.10	RECORDS .....	144
	15.19.11	RECORD .....	144
	15.19.12	SELECT .....	145
15.20		Oberflächenfunktionen .....	146
	15.20.1	SETCURSOR .....	146
	15.20.2	GETDISABLED .....	146
	15.20.3	SETDISABLED .....	146
	15.20.4	GETWINDOWDISABLED .....	147
	15.20.5	SETWINDOWDISABLED .....	147
	15.20.6	GETWINDOWOPEN .....	147
	15.20.7	SETWINDOWOPEN .....	147
15.21		Projektfunktionen .....	148
	15.21.1	PROJECTNAME .....	148
	15.21.2	CHANGES .....	148
15.22		Systemfunktionen .....	148
	15.22.1	EDIT .....	148

15.22.2	EDIT*	149
15.22.3	VIEW	149
15.22.4	VIEW*	149
15.22.5	SYSTEM	149
15.22.6	STAT	150
15.22.7	TACKON	150
15.22.8	FILENAME	150
15.22.9	DIRNAME	151
15.22.10	TODAY	151
15.22.11	NOW	151
15.22.12	MESSAGE	152
15.22.13	GC	152
15.23	Vordefinierte Variablen	152
15.24	Vordefinierte Konstanten	152
15.25	Funktionale Parameter	153
15.26	Typdeklarierer	154
15.27	Aufbau von Ausdrücken	155
15.28	Auslösefunktionen	156
15.28.1	onOpen	156
15.28.2	onClose	156
15.28.3	onChange	157
15.28.4	Auslösefunktion Neu	157
15.28.5	Auslösefunktion Löschen	158
15.28.6	Comparison function	158
15.28.7	Auslösefunktion Feld	160
15.28.8	Virtuelle Felder programmieren	160
<b>ABConvert</b>		<b>162</b>
<b>Anerkennung</b>		<b>163</b>
<b>Autor</b>		<b>164</b>
<b>Funktionsverzeichnis</b>		<b>165</b>
<b>Stichwortverzeichnis</b>		<b>168</b>