

MUIbase

Eine relationale programmierbare Datenbank
Version 4.0

10 Oktober 2017

Steffen Gutmann

Übersetzt von Ralph Reuchlein 1999-2000

Copyright © 2017 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.

Inhaltsverzeichnis

1	Kopierbestimmungen von MUIbase.....	1
1.1	Lizens	1
1.2	Spenden	1
1.3	Email-Verteiler	1
1.4	Verzichtserklärung	1
1.5	Fremde Hilfsmittel	2
1.5.1	Windows-, Mac-OS- und Linux-Version	2
1.5.2	Amiga-Version	2
2	Willkommen zu MUIbase.....	4
3	Installation.....	6
3.1	MUIbase für Windows installieren	6
3.2	MUIbase für Mac OS installieren	6
3.3	MUIbase für Linux installieren	6
3.4	MUIbase für Amiga installieren	7
3.5	Aktualisieren einer bereits vorhandenen MUIbase Version	7
3.6	MUIbase starten	8
3.7	MUIbase beenden	8
3.8	Konventionen für Dateinamen auf Windows, Mac OS und Linux ..	8
4	Tutorial.....	10
4.1	Wie MUIbase arbeitet	10
4.2	Ein Projekt beginnen: Der Struktureditor	10
4.3	Hinzufügen einer Tabelle	10
4.4	Hinzufügen eines Feldes	11
4.5	Darstellen des Projekts	11
4.6	Hinzufügen von zwei Datensatzbeziehungen	13
4.7	Datensätze hinzufügen	13
4.8	Filter	14
4.9	Abfragen	14
4.10	Hinzufügen einer Tabelle mit einem mehrzeiligen Text und einem Knopf	15
4.11	MUIbase programmieren, um einen Stammbaum zu erzeugen ..	16
4.12	MUIbase programmieren, um die Kinder einer Person aufzulisten ..	17
5	Grundlagen.....	20
5.1	Projekte	20
5.2	Tabellen	20
5.3	Datensätze	21
5.4	Felder	21

5.5	Feldtypen	21
5.5.1	Zeichenketten	21
5.5.2	Ganzzahlfelder	22
5.5.3	Fließkommazahlfelder	22
5.5.4	Boolesche Felder	22
5.5.5	Auswahlfelder	22
5.5.6	Datumsfelder	23
5.5.7	Zeitfelder	23
5.5.8	Mehrzeilige Textfelder	23
5.5.9	Beziehungsfelder	23
5.5.10	Virtuelles Feld	23
5.5.11	Knöpfe	24
5.6	Tabelle der Feldtypen	24
5.7	Speicherverbrauch	25
5.8	Beziehungen	25
5.8.1	Eins-zu-Eins-Beziehungen	26
5.8.2	Eins-zu-Mehrfach-Beziehungen	26
5.8.3	Mehrfach-zu-Mehrfach-Beziehungen	26
5.9	Benutzerschnittstelle	28
5.9.1	Fenster	28
5.9.2	Masken	29
5.9.3	Panels	29
5.9.4	Feldobjekte	29
5.9.5	Textobjekte	29
5.9.6	Bilder	29
5.9.7	Zwischenraumobjekte	30
5.9.8	Gruppen	30
5.9.9	Gewichtungsobjekte	30
5.9.10	Karteikarten-Gruppen	30
6	Projekte verwalten	31
6.1	Dateiformat	31
6.2	Information	31
6.3	Neues Projekt	31
6.4	Projekt leeren	32
6.5	Projekt öffnen	32
6.6	Projekt speichern	32
6.7	Admin- und Benutzermodus	33
6.8	Integrität der Daten prüfen	33
6.9	Datensätze auslagern	34
6.10	Projekt schließen	34

7	Einstellungen	35
7.1	Benutzereinstellungen	35
7.1.1	Formate	35
7.1.2	Externer Editor	35
7.1.3	Externer Anzeiger	36
7.1.4	Extra-Knöpfe in Tab-Kette	36
7.1.5	Weiterspringen bei Enter	37
7.1.6	Beenden bestätigen	37
7.1.7	MUI	37
7.2	Projekteinstellungen	37
7.2.1	Datensatzspeicher	37
7.2.2	Datensätze löschen bestätigen	38
7.2.3	Pfade relativ zu einem Projekt	38
7.2.4	Automatisches Neuladen bestätigen	38
7.2.5	Speichern & Umschichten bestätigen	39
7.2.6	Programmquellen	39
7.2.7	Externe Programmquellen aufräumen	39
7.2.8	Programm-Debug-Information	39
7.2.9	Veraltete Funktionen	39
7.2.10	Trigger-Funktionen sortieren	40
7.2.11	Programm-Include-Verzeichnis	40
7.2.12	Programm-Ausgabedatei	40
7.3	Als Voreinstellungen speichern	41
8	Log	42
8.1	Log-Tabelle	42
8.2	Protokollieren aktivieren	43
8.3	Logbeschriftung setzen	43
8.4	Log importieren	43
8.5	Log exportieren	43
8.6	Log löschen	43
8.7	Log ansehen	44
9	Datensatzbearbeitung	45
9.1	Aktive Objekte	45
9.2	Datensätze hinzufügen	45
9.3	Datensätze verändern	45
9.3.1	Zeichenkettenfelder mit einem Popup-Knopf	45
9.3.2	Eingabe von Ganzzahlwerten	46
9.3.3	Eingabe von Booleschen Werten	46
9.3.4	Eingabe von Auswahlwerten	46
9.3.5	Eingabe von Datumswerten	46
9.3.6	Eingabe von Zeitwerten	46
9.3.7	Kontextmenü des mehrzeiligen Textfelds	46
9.3.8	Kontextmenü des Select-from-where-Listenfelds	47
9.3.9	Eingabe von Beziehungswerten	47
9.3.10	Eingabe von NIL-Werten	48

9.4	Datensätze löschen	48
9.5	Datensätze durchforsten	48
9.6	Alle Datensätze ansehen	49
10	Filter	50
10.1	Datensatzfilter	50
10.1.1	Filterausdruck	50
10.1.2	Filter ändern	50
10.1.3	Filterbeispiele	51
10.2	Referenzfilter	51
11	Sortieren	52
11.1	Keine Sortierung	52
11.2	Sortieren nach Feldern	52
11.3	Sortieren nach einer Funktion	53
11.4	Sortierung ändern	53
11.5	Neu sortieren aller Datensätze	54
12	Suchen	55
12.1	Suchfenster	55
12.2	Vorwärts/Rückwärts suchen	55
12.3	Suchmusterbeispiele	56
13	Import und Export	57
13.1	Dateiformat	57
13.2	Beispiel-Importdatei	57
13.3	Datensätze importieren	58
13.4	Datensätze exportieren	58
14	Datenabfragen	60
14.1	Select-from-where Abfragen	60
14.2	Abfrageeditor	60
14.3	Abfragen als Text exportieren	61
14.4	Abfragen als PDF exportieren	62
14.5	Abfragen ausdrucken	62
14.6	Abfragebeispiele	64

15	Struktureditor	67
15.1	Tabellenverwaltung	67
15.1.1	Tabellen erstellen	67
15.1.2	Tabellen ändern	68
15.1.3	Tabellen löschen	68
15.1.4	Tabellen sortieren	68
15.2	Felderverwaltung	69
15.2.1	Felder erstellen	69
15.2.2	Typabhängige Einstellungen	69
15.2.3	Auswahltexteditor	70
15.2.4	Felder kopieren	71
15.2.5	Felder ändern	71
15.2.6	Felder löschen	72
15.2.7	Felder sortieren	72
15.3	Anzeigeverwaltung	72
15.3.1	Anzeigebereich	72
15.3.2	Tabellenobjekteditor	73
15.3.3	Feldobjekteditor	75
15.3.4	Typabhängige Einstellungen	76
15.3.5	Texteditor	80
15.3.6	Bildeditor	80
15.3.7	Zwischenraumeditor	80
15.3.8	Gruppeneditor	81
15.3.9	Karteikarteneditor	81
15.3.10	Fenstereditor	81
15.4	Struktur exportieren	82
16	MUIbase programmieren	83
16.1	Programmeditor	83
16.2	Externe Programmquellen	83
16.3	Vorverarbeitung	84
16.3.1	#define	84
16.3.2	#undef	85
16.3.3	#include	85
16.3.4	#if	85
16.3.5	#ifdef	85
16.3.6	#ifndef	85
16.3.7	#elif	86
16.3.8	#else	86
16.3.9	#endif	86
16.4	Programmiersprache	86
16.4.1	Warum Lisp?	87
16.4.2	Lisp-Aufbau	87
16.4.3	Programmarten	87
16.4.4	Namenskonventionen	88
16.4.5	Datensatzinhalte ansprechen	88
16.4.6	Datentypen zum Programmieren	89

16.4.7	Konstanten	89
16.4.8	Befehlsaufbau	91
16.5	Befehle definieren	91
16.5.1	DEFUN	91
16.5.2	DEFUN*	92
16.5.3	DEFVAR	92
16.5.4	DEFVAR*	93
16.6	Programmsteuerungsfunktionen	93
16.6.1	PROGN	93
16.6.2	PROG1	93
16.6.3	LET	93
16.6.4	SETQ	94
16.6.5	SETQ*	94
16.6.6	FUNCALL	95
16.6.7	APPLY	95
16.6.8	IF	95
16.6.9	CASE	96
16.6.10	COND	96
16.6.11	DOTIMES	96
16.6.12	DOLIST	97
16.6.13	DO	98
16.6.14	FOR ALL	98
16.6.15	NEXT	99
16.6.16	EXIT	99
16.6.17	RETURN	99
16.6.18	HALT	100
16.6.19	ERROR	100
16.7	Typaussagen	100
16.8	Typumwandlungsfunktionen	101
16.8.1	STR	101
16.8.2	MEMO	102
16.8.3	INT	102
16.8.4	REAL	103
16.8.5	DATE	103
16.8.6	TIME	103
16.9	Boolesche Funktionen	104
16.9.1	AND	104
16.9.2	OR	104
16.9.3	NOT	105
16.10	Vergleichsfunktionen	105
16.10.1	Relationsoperatoren	105
16.10.2	CMP	106
16.10.3	CMP*	106
16.10.4	MAX	106
16.10.5	MAX*	106
16.10.6	MIN	107
16.10.7	MIN*	107
16.11	Mathematik-Funktionen	107

16.11.1	Werte addieren	107
16.11.2	Werte subtrahieren	108
16.11.3	1+	108
16.11.4	1-	108
16.11.5	Werte multiplizieren (*)	108
16.11.6	Werte dividieren	109
16.11.7	DIV	109
16.11.8	MOD	109
16.11.9	ABS	109
16.11.10	TRUNC	109
16.11.11	ROUND	109
16.11.12	RANDOM	110
16.11.13	POW	110
16.11.14	SQRT	110
16.11.15	EXP	110
16.11.16	LOG	110
16.12	Zeichenkettenfunktionen	111
16.12.1	LEN	111
16.12.2	LEFTSTR	111
16.12.3	RIGHTSTR	111
16.12.4	MIDSTR	111
16.12.5	SETMIDSTR	111
16.12.6	INSMIDSTR	112
16.12.7	INDEXSTR	112
16.12.8	INDEXSTR*	112
16.12.9	INDEXBRK	112
16.12.10	INDEXBRK*	112
16.12.11	RINDEXSTR	113
16.12.12	RINDEXSTR*	113
16.12.13	RINDEXBRK	113
16.12.14	RINDEXBRK*	113
16.12.15	REPLACESTR	113
16.12.16	REPLACESTR*	114
16.12.17	REMCHARS	114
16.12.18	TRIMSTR	114
16.12.19	WORD	114
16.12.20	WORDS	115
16.12.21	FIELD	115
16.12.22	FIELDS	115
16.12.23	STRTOLIST	115
16.12.24	LISTTOSTR	116
16.12.25	CONCAT	116
16.12.26	CONCAT2	116
16.12.27	COPYSTR	117
16.12.28	SHA1SUM	117
16.12.29	UPPER	117
16.12.30	LOWER	117
16.12.31	ASC	117

16.12.32	CHR	118
16.12.33	LIKE	118
16.12.34	SPRINTF	118
16.13	Funktionen für mehrzeilige Texte	121
16.13.1	LINE	121
16.13.2	LINES	121
16.13.3	MEMOTOLIST	121
16.13.4	LISTTOMEMO	122
16.13.5	FILLMEMO	122
16.13.6	FORMATMEMO	122
16.13.7	INDENTMEMO	123
16.14	Datum- und Zeitfunktionen	123
16.14.1	DAY	123
16.14.2	MONTH	123
16.14.3	YEAR	123
16.14.4	DATEDMY	123
16.14.5	MONTHDAYS	124
16.14.6	YEARDAYS	124
16.14.7	ADDMONTH	124
16.14.8	ADDYEAR	124
16.14.9	TODAY	125
16.14.10	NOW	125
16.15	Listenfunktionen	125
16.15.1	CONS	125
16.15.2	LIST	125
16.15.3	LENGTH	125
16.15.4	FIRST	126
16.15.5	REST	126
16.15.6	LAST	126
16.15.7	NTH	126
16.15.8	APPEND	126
16.15.9	REVERSE	126
16.15.10	MAPFIRST	127
16.15.11	SORTLIST	127
16.15.12	SORTLISTGT	127
16.16	Benutzereingabefunktionen	128
16.16.1	ASKFILE	128
16.16.2	ASKDIR	128
16.16.3	ASKSTR	128
16.16.4	ASKINT	129
16.16.5	ASKCHOICE	129
16.16.6	ASKCHOICESTR	130
16.16.7	ASKOPTIONS	131
16.16.8	ASKBUTTON	131
16.16.9	ASKMULTI	132
16.17	E/A-Funktionen	134
16.17.1	FOPEN	134
16.17.2	FCLOSE	135

16.17.3	stdout	135
16.17.4	PRINT	135
16.17.5	PRINTF	135
16.17.6	FPRINTF	136
16.17.7	FERROR	136
16.17.8	FEOF	136
16.17.9	FSEEK	136
16.17.10	FTELL	137
16.17.11	FGETCHAR	137
16.17.12	FGETCHARS	137
16.17.13	FGETSTR	137
16.17.14	FGETMEMO	138
16.17.15	FPUTCHAR	138
16.17.16	FPUTSTR	138
16.17.17	FPUTMEMO	138
16.17.18	FFLUSH	138
16.18	Datensatzfunktionen	138
16.18.1	NEW	139
16.18.2	NEW*	139
16.18.3	DELETE	139
16.18.4	DELETE*	140
16.18.5	DELETEALL	140
16.18.6	GETMATCHFILTER	140
16.18.7	SETMATCHFILTER	141
16.18.8	GETISSORTED	141
16.18.9	SETISSORTED	141
16.18.10	RECNUM	141
16.18.11	COPYREC	142
16.19	Feldfunktionen	142
16.19.1	ATTRNAME	142
16.19.2	MAXLEN	142
16.19.3	GETLABELS	142
16.19.4	SETLABELS	143
16.20	Tabellenfunktionen	143
16.20.1	TABLENAME	143
16.20.2	GETORDERSTR	143
16.20.3	SETORDERSTR	144
16.20.4	REORDER	144
16.20.5	REORDERALL	145
16.20.6	GETFILTERACTIVE	145
16.20.7	SETFILTERACTIVE	145
16.20.8	GETFILTERSTR	145
16.20.9	SETFILTERSTR	145
16.20.10	RECORDS	146
16.20.11	RECORD	146
16.20.12	SELECT	146
16.21	Oberflächenfunktionen	147
16.21.1	SETCURSOR	147

16.21.2	SETBGPEN	148
16.21.3	GETWINDOWOPEN	148
16.21.4	SETWINDOWOPEN	148
16.21.5	GETVIRTUALLISTACTIVE	148
16.21.6	SETVIRTUALLISTACTIVE	149
16.22	Projektfunktionen	149
16.22.1	PROJECTNAME	149
16.22.2	PREPARECHANGE	149
16.22.3	CHANGES	150
16.22.4	GETADMINMODE	150
16.22.5	SETADMINMODE	150
16.22.6	ADMINPASSWORD	150
16.23	Systemfunktionen	150
16.23.1	EDIT	150
16.23.2	EDIT*	151
16.23.3	VIEW	151
16.23.4	VIEW*	151
16.23.5	SYSTEM	151
16.23.6	SYSTEM*	152
16.23.7	STAT	152
16.23.8	TACKON	152
16.23.9	FILENAME	152
16.23.10	DIRNAME	152
16.23.11	MESSAGE	153
16.23.12	COMPLETEMAX	153
16.23.13	COMPLETEADD	153
16.23.14	COMPLETE	154
16.23.15	GC	154
16.23.16	PUBSCREEN	154
16.24	Vordefinierte Variablen	154
16.25	Vordefinierte Konstanten	154
16.26	Funktionale Parameter	155
16.27	Typdeklarierer	156
16.28	Aufbau von Ausdrücken	157
16.29	Auslösefunktionen	158
16.29.1	onOpen	158
16.29.2	onReload	158
16.29.3	onClose	158
16.29.4	onAdminMode	159
16.29.5	onChange	159
16.29.6	logLabel	159
16.29.7	mainWindowTitle	160
16.29.8	Auslösefunktion Neu	160
16.29.9	Auslösefunktion Löschen	161
16.29.10	Vergleichsfunktion	161
16.29.11	Auslösefunktion Feld	162
16.29.12	Virtuelle Felder programmieren	163
16.29.13	Berechne-Aktiv-Funktion	164

16.29.14	Berechne Datensatzbeschreibung	164
16.29.15	Auslösefunktion Doppelklick.....	164
16.29.16	Berechne Listenansichts-Auswahltexte	165
16.29.17	Berechne Referenz-Datensätze	165
16.30	Liste veralteter Funktionen.....	166
17	ARexx Schnittstelle	167
17.1	Portname	167
17.2	Befehlsaufbau	167
17.3	Rückgabewerte	168
17.4	Quit	169
17.5	Hide	169
17.6	Show	169
17.7	Info	169
17.8	Help	169
17.9	Compile.....	169
17.10	Connect	170
17.11	Disconnect.....	170
17.12	Connections.....	171
17.13	Eval	171
17.14	Transaction	172
17.15	Commit.....	173
17.16	Rollback.....	173
Menüs	174	
Menü Projekt	174	
Menü Einstellungen	175	
Menü Log	176	
Menü Tabelle.....	176	
Menü Programm	177	
Menu Hilfe	178	
Anerkennung	179	
Autor	180	
Funktionsverzeichnis	181	
Stichwortverzeichnis	184	

1 Kopierbestimmungen von MUIbase

MUIbase ist Copyright © 1998-2017 Steffen Gutmann. Alle Rechte vorbehalten.

MUIbase ist Open-Source-Software und wird unter den Bestimmungen der GNU General Public License (GPL) verteilt.

1.1 Lizens

Dieses Programm ist Freie Software: Sie können es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Wahl) jeder neueren veröffentlichten Version, weiterverbreiten und/oder modifizieren.

Dieses Programm wird in der Hoffnung, dass es nützlich sein wird, aber OHNE JEDE GEWÄHRLEISTUNG, bereitgestellt; sogar ohne die implizite Gewährleistung der MARKTFÄHIGKEIT oder EIGNUNG FÜR EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License für weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe <http://www.gnu.org/licenses/>.

1.2 Spenden

MUIbase ist völlig kostenlos. Falls Sie das Projekt gut finden und es unterstützen möchten, so sind Sie gerne eingeladen, eine Spende zu machen. Um zu spenden besuchen Sie bitte <http://www.sourceforge.net/projects/muibase/donate>.

1.3 Email-Verteiler

Ein Email-Verteiler zur Diskussion von MUIbase-bezogenen Themen wurde unter <http://www.groups.yahoo.com/groups/muibase-list> eingerichtet. Jedermann ist willkommen den Verteiler zu abonnieren. Bitte senden Sie Fehlerberichte und Wünsche an diesen Verteiler.

1.4 Verzichtserklärung

DIESE SOFTWARE WIRD VOM AUTOR UND DEN BEITRAGSLEISTENDEN OHNE JEGLICHE SPEZIELLE ODER IMPLIZIERTE GARANTIEN ZUR VERFÜGUNG GESTELLT, DIE UNTER ANDEREM EINSCHLIESSEN: DIE IMPLIZIERTE GARANTIE DER MARKTFÄHIGKEIT ODER DIE EIGNUNG DER SOFTWARE FÜR EINEN BESTIMMTEN ZWECK. AUF KEINEN FALL SIND DER AUTOR ODER DIE BEITRAGSLEISTENDEN FÜR IRGENDWELCHE DIREKTEN, INDIREKTEN, ZUFÄLLIGEN, SPEZIELLEN, BEISPIELHAFTEN ODER FOLGENDEN SCHÄDEN (UNTER ANDEREM VERSCHAFFEN VON ERSATZGÜTERN ODER -DIENSTLEISTUNGEN; EINSCHRÄNKUNG DER NUTZUNGSFÄHIGKEIT; VERLUST VON NUTZUNGSFÄHIGKEIT; DATEN; PROFIT ODER GESCHÄFTSUNTERBRECHUNG), WIE AUCH IMMER VERURSACHT UND UNTER WELCHER VERPFLICHTUNG AUCH IMMER, OB IN VERTRAG, STRIKTER VERPFLICHTUNG ODER UNERLAUBTE HANDLUNG (INKLUSIVE FAHRLÄSSIGKEIT ODER SONSTIGES) VERANTWORTLICH, AUF WELCHEM

WEG SIE AUCH IMMER DURCH DIE BENUTZUNG DIESER SOFTWARE ENTSTANDEN SIND, SOGAR, WENN SIE AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WORDEN SIND.

Anm.d.Übersetzers: Der englische Originaltext lautet wie folgt:

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.

1.5 Fremde Hilfsmittel

MUIbase benutzt externe Bibliotheken und anderes Material abhängig vom Betriebssystem.

1.5.1 Windows-, Mac-OS- und Linux-Version

MUIbase 3.x für Windows, Mac OS und Linux benutzt das Gimp Toolkit (GTK+) Version 2.10 oder höher, Copyright © 2007-2017 The GTK+ Team. GTK ist ein Toolkit für die Entwicklung graphischer Bedienoberflächen (GUI) und wird unter der GNU Library General Public License (LGPL) verteilt. Für weitere Informationen siehe <http://www.gtk.org>.

MUIbase 4.x für Windows, Mac OS und Linux benutzt Qt in Version 5.3 oder höher, Copyright © 2017 The Qt Company. Qt ist ein umfassendes plattform-übergreifendes Framework und Toolkit, welches unter den GPL- und LGPLv3-Lizenzen verfügbar ist. Für weitere Informationen siehe <http://www.qt.io>.

Das Windows-Installations-Skript für MUIbase basiert auf dem Installations-Skript für The Gimp, auf welches Jernej Simoncic die Rechte hält.

1.5.2 Amiga-Version

The Amiga version of MUIbase uses

MUI - MagicUserInterface

Copyright © 1992-2013 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/>

MUIbase für Amiga verwendet NList.mcc, Copyright ©) 2001-2015 NList Open Source Team. Für mehr Informationen oder für die neueste Version, siehe <http://www.sourceforge.net/projects/nlist-classes>.

MUIbase für Amiga verwendet BetterString.mcc, Copyright ©) 2005-2015 BetterString Open Source Team. Siehe <http://www.sourceforge.net/projects/bstring-mcc> für mehr Informationen oder für neueste Versionen.

MUIbase für Amiga verwendet TextEditor.mcc, Copyright ©) 2005-2015 TextEditor Open Source Team. Siehe <http://www.sourceforge.net/projects/texteditor-mcc> für mehr Informationen oder für neueste Versionen.

MUIbase für Amiga verwendet codesets.library, Copyright ©) 2005-2015 codesets.library Open Source Team. Siehe <http://www.sourceforge.net/projects/codesetslib> für mehr Informationen oder für neueste Versionen.

Einige Icons, die im MUIbase-Paket für Amiga verwendet werden, stammen vom DefaultIcons-Paket. Diese Icons sind Copyright ©) Michael-Wolfgang Hohmann und Angela Schmidt.

2 Willkommen zu MUIbase

MUIbase ist eine schnelle und flexible Datenbank mit graphischer Benutzerschnittstelle (GUI) und Programmierbarkeit. Die Zielgruppe sind fortgeschrittene Desktop-Benutzer, welche Daten bequem und leistungsstark verwalten wollen. MUIbase kann etliche Arten von Daten verwalten, z.B. Adressen, CDs, Filme, Fotosammlungen, Ihren Familienstammbaum, Ihre Ein- und Ausgaben, und vieles mehr. Die Stärke von MUIbase liegt in seiner einfachen und mächtigen Benutzerschnittstelle und seinen Programmierfähigkeiten. Das Programmieren von MUIbase erlaubt es, Daten auf verschiedene Arten zu verarbeiten, z.B. die automatische Durchführung von Berechnungen bei Benutzereingabe, die Erstellung von Reporten, den Import und Export von Daten, etc. Beispielsweise können die Summer aller Einnahmen, oder die Gesamtdauer einer CD berechnet werden, oder automatische Serienbriefe für Ihre Kunden erstellt und gedruckt werden.

MUIbase bietet die folgende Features:

- Gleichzeitiger Zugriff auf ein Datenbankprojekt mit mehreren MUIbase-Instanzen, welche auf dem gleichen oder auf verschiedenen Rechnern laufen.
- Unbegrenzte Anzahl von Projekten, Tabellen, Feldern und Datensätzen.
- Felder können vom Typ Zeichenkette, Memo (mehrzeiliger Text), Ganzzahl, Fließkommazahl, Datum, Zeit, Boolesch, Auswahl (ein Eintrag von mehreren möglichen Einträgen), Beziehung (einfache Möglichkeit einen Datensatz einer anderen Tabelle zu referenzieren), Knopf (zum Starten von MUIbase-Programmen) und virtuell (zum automatischen Berechnen von Werten) sein.
- Der Zeichenkettentyp kann auch Listen von Zeichenketten, Dateien und Zeichensätze verwalten. Eine Zeichenkette kann auf ein externes Bild verweisen, welches in der Benutzerschnittstelle angezeigt wird.
- Dynamisches Laden von Datensätzen. Datensätze, die nicht benötigt werden, können aus dem Speicher entfernt werden (z.B. bei Speicherplatzmangel).
- Programmierbarkeit. Mit der einfachen und leistungsstarken MUIbase-Programmiersprache können komplexe Abläufe realisiert werden. Die Sprache enthält auch eine SELECT-FROM-WHERE-Abfrage für bequeme und schnelle Datenabfragen.
- Sortierung von Datensätzen nach beliebiger Kombination von Feldern
- Flexible und leistungsstarke Such- und Filtermöglichkeit
- Abfrageneditor, der das Eingeben und Verwalten von SELECT-FROM-WHERE-Abfragen ermöglicht. Die Abfragen können gespeichert und die Ergebnisse ausgedruckt werden
- Optionales Protokollieren von Änderungen (Hinzufügen, Löschen und Ändern von Datensätzen) in eine System-Logtabelle.
- Import- und Exportmöglichkeit
- Volle Dokumentation mit Benutzer- und Programmierhandbuch in HTML und PDF (plus AmigaGuide auf dem Amiga).
- Die Amiga-Version bietet eine leistungsstarke ARexx-Schnittstelle, um auf MUIbase von anderen Programmen aus zuzugreifen. Die ARexx-Schnittstelle hat einen Transaktionsmechanismus ähnlich zu anderen relationalen Datenbanken.

- Unabhängigkeit vom Betriebssystem. MUIbase ist für Windows, Mac OS, Linux und Amiga verfügbar. Der Quelltext ist im Rahmen eines Source-Forge-Projekts erhältlich.

3 Installation

Dieses Kapitel beschreibt, welche Hardware und Software benötigt wird, um MUIbase auf Ihrem Computer zu benutzen, wie MUIbase installiert und aktualisiert wird, und wie man MUIbase startet und beendet.

3.1 MUIbase für Windows installieren

MUIbase für Windows kann auf einem Intel-kompatiblen Computer benutzt werden, auf welchem das Betriebssystem Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8 oder Windows 10 läuft. Ab MUIbase version 4.0 wird Windows 7 oder besser benötigt.

Der MUIbase-Installer wird als ausführbares Windows-Programm verteilt, beispielsweise als `'MUIbase-4.0-setup.exe'` und kann von der MUIbase Homepage <http://muibase.sourceforge.net> bezogen werden. Nach Start des Installers geben Sie das Zielverzeichnis ein, in das MUIbase installiert werden soll, und stellen weitere Konfigurationen ein. Falls Sie bereits eine ältere Version von MUIbase auf Ihrem System haben, so wird diese mit der neuen Version aktualisiert.

Ihr Windows `'Start'`-Menü sollte nach erfolgreicher Installation einen neuen Eintrag für das Starten von MUIbase enthalten. Der MUIbase-Installer wird nun nicht mehr benötigt und kann entfernt werden.

3.2 MUIbase für Mac OS installieren

MUIbase für Mac OS unterstützt Mac OS X Lion (10.7) und spätere Versionen des Mac OS X Betriebssystems. Die Software wird als 64bit-Intel-Binary verteilt.

Eine Disk-Image, z.B. `'MUIbase-4.0-Lion.dmg'` kann von der MUIbase Homepage <http://muibase.sourceforge.net> bezogen werden. Nach dem Herunterladen und Öffnen der Disk-Image wird einfach `'MUIbase.app'` per Drag-and-Drop in das `'Applications'`-Verzeichnis verschoben.

3.3 MUIbase für Linux installieren

Sie können MUIbase auf einem Intel-kompatiblen Computer, auf welchem das Linux-Betriebssystem läuft, installieren und benutzen.

MUIbase für Linux wird sowohl als Debian-Paket (z.B. `'muibase_4.0_ubuntu_trusty_amd64.deb'` für Ubuntu und Debian) als auch als RPM-Archiv (z.B. `'MUIbase-4.0-fc21.x86_64.rpm'` für Fedora) verteilt. Es gibt separate Versionen für die 32bit-Architektur, z.B. `'muibase_4.0_ubuntu_trusty_i386.deb'` und `'MUIbase-4.0-fc21.i686.rpm'`. Alle Versionen können von der MUIbase-Homepage <http://muibase.sourceforge.net> bezogen werden.

Normalerweise wird MUIbase nach dem Herunterladen automatisch installiert. Falls der Installationsprozess nicht automatisch beginnt, kann MUIbase durch die Eingabe des folgenden Befehls als Administrator (root) in einem Kommandozeilen-Interpreter (command shell) auf einem Debian-basierten Linux-Computer installiert werden:

```
dpkg --install muibase.deb
```

wobei *muibase.deb* das heruntergeladene Debian-Paket ist.

Auf einem Redhat-basiertem System (RPM) lautet der Befehl:

```
rpm -Uvh MUIbase.rpm
```

wobei *MUIbase.rpm* das heruntergeladene RPM-Archiv ist.

Nach erfolgreicher Installation von MUIbase sollten Sie einen neuen Menüpunkt im Menübaum ‘Applications - Office’ auf Ihrem (Gnome oder KDE)-Desktop finden. Das heruntergeladene Archiv wird nun nicht mehr benötigt und kann entfernt werden.

NB: On Debian-based systems you need to hold the package otherwise the next update of your system will remove it because it's not installed from a source defined in the repository list `/etc/apt/source.list`.

3.4 MUIbase für Amiga installieren

MUIbase für Amiga läuft auf Amiga OS Version 3.0 oder höher und benötigt mindestens einen 68020 Prozessor. Weiterhin muss MUI in Version 3.8 oder höher bereits auf Ihrem System vorhanden sein. Mindestens 4MB Hauptspeicher (RAM) und 10MB freier Festplattenplatz werden benötigt. Berichte besagen, dass das m68k-Binary von MUIbase für Amiga stabil unter UAE, MorphOS und Amiga OS4 funktioniert. PowerPC-Binär-Versionen für MorphOS und Amiga OS4 sind ebenfalls in dem verteilten Archiv enthalten. Eine separates Archiv enthält eine Version für AROS.

MUIbase wird als LHA-Archiv, z.B. ‘*MUIbase-3.0.1ha*’, verteilt und kann von der MUIbase Homepage <http://muibase.sourceforge.net> bezogen werden.

Für die Installation von MUIbase entpacken Sie dieses Archiv in ein temporäres Verzeichnis. Entpacken Sie es nicht ins Zielverzeichnis!

Doppel-klicken Sie auf das MUIbase-Installerskript `Install-MUIbase` und folgen sie den Anweisungen. Das Skript fragt nach einem Verzeichnis, in welches die Software installiert werden soll. Geben Sie hier nicht das Verzeichnis an, in das Sie das MUIbase-Archiv entpackt haben. Das Skript kann auch eine vorhandene MUIbase-Installation auf den neuesten Stand bringen, indem das Verzeichnis einer bereits vorhandenen MUIbase-Installation eingegeben wird.

Nach erfolgreicher Installation finden Sie eine (neue) Schublade auf Ihrem System, welches das MUIbase-Programm und alle benötigten Dateien sowie einige Beispielprojekte enthält. Zusätzlich wurde eine Zuweisung (assign) `MUIbase:` auf diese Schublade zu Ihrer System-Datei `S:User-Startup` hinzugefügt. Das LHA-Archiv und die temporären Dateien werden nun nicht weiter benötigt und können entfernt werden.

3.5 Aktualisieren einer bereits vorhandenen MUIbase Version

Beim Installieren von MUIbase für Windows, Mac OS, Linux oder Amiga kann eine bereits vorhandene MUIbase-Installation aktualisiert oder erneut installiert werden. Während der neuen Installation werden alle benötigten Dateien durch neue ersetzt. Dies beinhaltet auch die Beispielprojekte im ‘Demos’-Verzeichnis. Aus diesem Grund ist es weder ratsam, eigene Projekte in dieses Verzeichnis zu legen, noch irgendeines der Beispielprojekte für die Verwaltung von Ihren eigenen Daten zu verwenden, da diese bei einer Neuinstallation überschrieben werden.

Es wird empfohlen, eigene Projekte in einem separaten Verzeichnis unabhängig zur MUIbase-Installation zu verwalten.

3.6 MUIbase starten

MUIbase kann entweder von der grafischen Benutzerumgebung oder von einem Kommandozeilen-Interpreter aus gestartet werden. Auf Windows finden Sie MUIbase im ‘Start’-Menü. Unter Mac OS wird MUIbase über den ‘Finder’ im ‘Applications’-Verzeichnis gestartet. Falls Sie Gnome oder KDE auf Linux benutzen, so kann MUIbase durch Wählen des zugehörigen Menüpunktes im Menü ‘Applications - Office’ gestartet werden. Auf der Workbench (Amiga) doppel-klicken Sie das MUIbase-Piktogramm oder das Icon eines MUIbase-Projekts, welches dann automatisch nach dem Start von MUIbase geladen wird (es ist auch möglich, mehrere MUIbase-Projekte bei gedrückter SHIFT-Taste anzuklicken und das letzte doppelt zu klicken).

Beim Starten von MUIbase von einem Kommandozeilen-Interpreter aus auf Windows, Linux oder Amiga, können Argumente und optional zu ladende Projekte angegeben werden. Es gibt zwei Möglichkeiten, MUIbase von der Kommandozeile aus zu starten:

```
MUIbase [project1 ...]
MUIbase -n
```

Die erste Form startet MUIbase und lädt die optionalen Projekte, welche durch die Dateinamen ‘*project1*’ . . . angegeben sind. Die zweite Form startet MUIbase ohne graphische Benutzerschnittstelle. Dies kann nützlich sein, um MUIbase als Server im Hintergrund laufen zu lassen und mittels der externen Schnittstellen auf MUIbase zuzugreifen (z.B. ARexx auf Amiga, siehe Kapitel 17 [ARexx Schnittstelle], Seite 167).

3.7 MUIbase beenden

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

3.8 Konventionen für Dateinamen auf Windows, Mac OS und Linux

MUIbase für Windows, Mac OS und Linux besitzt ein paar spezielle Konventionen für Dateinamen, die normalerweise nicht in anderer Software für diese Systeme zu finden sind.

Beim Start von MUIbase wird eine Umgebungsvariable ‘MUIbase’ gesetzt, welche auf das Installationsverzeichnis von MUIbase verweist. Auf Windows ist dies das Verzeichnis, das bei der Installation eingegeben wurde. Auf Mac OS ist es das Verzeichnis `/Applications/MUIbase.app/Contents/Resources/share/MUIbase`. Auf Linux ist es das Verzeichnis `/usr/share/MUIbase`.

Beim Interpretieren von Dateinamen (entweder vom Benutzer eingegebene oder die Dateinamen in dieser Dokumentation) wird ein Dateiname auf das Vorkommen von Umgebungsvariablen hin untersucht. Enthält ein Dateiname ein Dollarzeichen ‘\$’, so werden die darauf folgenden Zeichen bis zum ersten nicht-alpha-numerischen Zeichen als eine Umgebungsvariable interpretiert. Ist diese Umgebungsvariable gesetzt, so wird der Teil des Dateinamens durch den Inhalt der Umgebungsvariable ersetzt (ansonsten bleibt der Teil des Dateinamens inklusive des ‘\$’-Zeichens unverändert). Sie können Klammern um die

Umgebungsvariable benutzen, sollte diese nicht-alpha-numerische Zeichen enthalten. Zum Beispiel expandiert `$HOME/data` zu dem Pfad, bei dem `$HOME` durch den Pfadnamen Ihres Heimatverzeichnisses ersetzt wurde.

Weiterhin werden Amiga-ähnliche ‘Assign-Namen’ wie folgt gehandhabt. Falls ein Dateiname einen Doppelpunkt ‘:’ enthält, so wird alles vor dem Doppelpunkt wie eine Umgebungsvariable (genannt ‘Assign-Name’) behandelt und durch ihren Inhalt ersetzt, wenn die Umgebungsvariable gesetzt ist. ‘Assign-Namen’ sollten mindestens 2 Zeichen lang sein, um sie von Laufwerk-Bezeichnern unter Windows zu unterscheiden.

Diese Konventionen erlauben es, z.B. auf die Beispiel-Filmdatenbank im MUIbase-Verzeichnis durch `MUIbase:demos/Movie.mb` zu verweisen.

Eine weitere Anwendung ist, wenn Sie eine Umgebungsvariable, z.B. ‘EXTERNAL_DATA’ auf ein Verzeichnis setzen, in welchem Sie externe Daten wie Bilder, etc. eines Ihrer Projekte speichern. Sie können dann auf die Daten in diesem Verzeichnis mittels `EXTERNAL_DATA:some-file` zugreifen und diese Dateinamen in der Projekt-Datenbank speichern.

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 als Projekt ‘FamilyTree.mb’ 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, wählen Sie den Menüpunkt ‘Struktureditor’ aus dem ‘Projekt’-Menü. Sie werden drei verschiedene Bereiche vorfinden:

‘Tabellen’

Hier ändern, löschen und fügen Sie Tabellen hinzu.

‘Felder’

Hier ändern, löschen und fügen Sie Felder in der aktuell ausgewählten Tabelle hinzu.

‘Anzeige’

Hier legen Sie das Aussehen der Datenbank fest, d.h. wie Tabellen und Felder dargestellt werden sollen.

4.3 Hinzufügen einer Tabelle

Als erstes benötigen wir eine Tabelle. Dazu drückt man den Knopf ‘Neu’ unterhalb der Liste im Bereich ‘Tabelle’. Es erscheint ein Fenster, das nach den folgenden 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 ‘Person’¹, da die Tabelle alle Namen der Personen speichern soll.

¹ Die englischen Namen der Tabellen, Felder und sonstige Texte werden beibehalten, da sonst das ganze Projekt umgeschrieben werden müsste.

‘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’

Das Hinzuzufügen und Löschen von Datensätzen kann durch Programmfunktionen geregelt werden. Diese Funktionen werden hier angegeben. Nachdem wir bis jetzt noch keine Programmfunktion geschrieben haben, lassen wir die Felder leer.

Nachdem alles eingestellt ist, drückt man den Knopf ‘Ok’. Damit haben wir unsere erste Tabelle namens ‘Person’.

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 Verlassen des Fensters bemerken Sie einige Veränderungen im Bereich ‘Anzeige’. Wechseln Sie über das Auswahlfeld oben im Anzeigebereich zum ‘Hauptfenster’. Nun sehen Sie, was das Hauptfenster beinhaltet, bis jetzt besteht es lediglich aus der Tabelle ‘Person’. Wechseln Sie nun mit dem Auswahlfeld wieder zurück zum ‘Tabellenschema’ und Sie können sehen, wie die Tabelle ‘Person’ dargestellt wird. Im Moment wird sie nur als ein Panel mit einem Feld angezeigt.

Nun doppel-klicken Sie auf ‘Person’ am Beginn der Liste im Anzeigebereich und ein Fenster erscheint. Klicken Sie auf die ‘Panel’-Leiste und Sie können nun einstellen, wie das Panel angezeigt werden soll:

‘Überschrift’

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

‘Hintergrund’

Der Hintergrund kann auf Ihren Geschmack passend eingestellt werden.

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

Nach dem Bestätigen durch Drücken von ‘OK’ doppel-klicken 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, durch den zu diesem Feld im Datensatzmodus gesprungen werden kann. Hierzu muss der Buchstabe bei gehaltener *Alt*-Taste (Windows, Mac OS und Linux), bzw. *Amiga*-Taste gedrückt werden.

‘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 (‘**Struktureditor verlassen**’ im Menü ‘Projekt’) und kehren in den Datensatzmodus zurück. Sie werden eine Überschrift sehen, die die Zeichenkette beinhaltet, den Sie im Anzeigebereich für das Panel eingegeben haben. Der Datensatzzähler sollte ‘#0/0’ anzeigen, da wir noch keine Datensätze eingefügt haben. Dahinter ist der Filterknopf und zwei Knöpfe mit Rückwärts- und Vorwärtspfeil. Unter all dem sollten Sie das Feld ‘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 Text `Person` und das Zeichenkettenfeld den Text `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 einem Hilfsfenster sehen.

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 mal ausprobieren.

Wechseln sie wieder in den Struktureditor und fügen zwei weitere Felder zu `Person` 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 `Person` 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 auf Tabelle `Person` zeigen.

Wie Sie vielleicht schon bemerkt haben, sind nun drei Felder im Anzeigebereich sichtbar. Klicken Sie einmal auf `Father` und dann auf die Knöpfe `Hoch` 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, wählen Sie `Neuer Datensatz` aus dem Menü `Tabelle` (dieses Menü ist in der Windows- und Linux-Version ein Popup-Menü, das durch drücken und halten der rechten Maustaste in der Tabelle sichtbar wird). Der Cursor sollte nun automatisch in das Feld springen, bei dem wir vorhin im Anzeigebereich des Struktureditors `Home` gesetzt haben. Geben Sie nun zwei Datensätze ein: einen mit dem Namen ihres Vaters in `Name` und nach Hinzufügen eines zweiten Datensatzes einen mit dem Namen ihrer Mutter auch im Feld `Name`. 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 kann. Wählen Sie den Namen ihres Vaters und führen das gleiche mit dem Listenansichtfenster 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 den Cursortasten *Up* oder *Down* 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. Diese verweilen dennoch in der Datenbank, sie sind einfach nur nicht mehr sichtbar.

Um den Filter einzugeben, wählen Sie **‘Ändere Filter’** aus dem Menü **‘Tabelle’**. Sie werden nun ein Fenster mit einer Menge 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 zuerst einmal auf den Knopf **LIKE**, doppelklicken dann 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.

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 kennen gelernt haben, wenden wir uns der Abfragefunktion von MUIbase zu. Abfragen können verwendet werden, um Informationen aus einer Datenbank zu erhalten, die bestimmten Kriterien genügen.

Wählen Sie **‘Abfragen’** aus dem Menü **‘Programm’**, 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.

‘Drucken’ Druckt das Ergebnis der Abfrage in eine Datei oder auf Ihrem Drucker aus.

‘Laden und Speichern’

Lädt und speichert Ihre Abfragen.

Das erste große Feld dient dem Eingeben 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 links ein. Im oberen großen Feld geben Sie folgendes ein:

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

Wenn Sie nun die Abfrage 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 Person WHERE
(AND (LIKE Name "*a*") (LIKE Name "*s*"))
```

Beachten Sie, dass wir immer noch 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 ‘Control’ und setzen Sie die Anzahl der Datensätze auf ‘genau ein’. Schließen Sie das Fenster mit ‘Ok’. Klicken und halten Sie die Maustaste auf der neuen Tabelle. Nun verschieben Sie den Eintrag etwas über die Mitte von ‘Person’ und lassen die Maustaste los. Im Bereich Tabelle erscheint nun ‘Control’ oben und ‘Person’ darunter.

Stellen Sie sicher, dass ‘Control’ aktiviert ist und wählen ‘Neu’ aus dem Bereich Felder. Ändern Sie den Typ auf ‘mehrzeiliger Text’ und geben Sie dem Feld den Namen ‘Result’. Drücken Sie ‘Ok’ und fügen ein weiteres Feld zu ‘Control’ 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 einmal auf den Knopf ‘Hoch’ drücken.

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 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 'Result' anzeigt. Folgende Funktion 'stammbaum' besteht genau genommen aus drei Funktionen:

- `pedigree`, das 'Control.Result' setzt, in dem eine andere Funktion aufgerufen wird.
- `getpedigree`, das den Stammbaum in einer verschachtelten Liste zusammenträgt.
- `pedigree2memo`, das diese Liste in einen mehrzeiligen Text umwandelt.

```
; Das Programm pedigree

(DEFUN pedigree ()
  (SETQ Control.Result
    (pedigree2memo (getpedigree Person NIL) 0 3)
  )
)
```

```
; Das Programm getpedigree
```

```
(DEFUN getpedigree (person:Person 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))
    )
    ""
  )
)

```

Achten Sie beim Eingeben des Programmes darauf, dass alle Klammern an der richtigen Stelle stehen. Zu viele oder zu wenig Klammern sind häufige Fehler, die beim Übersetzen des Programmes gemeldet werden. Die Fehlermeldung von MUIbase lautet in diesem Fall schlicht ‘Syntaxfehler’. Drücken Sie ‘Kompilieren & Schließen’ und das Fenster wird geschlossen, was bedeutet, dass MUIbase keine Fehler im Program vorgefunden hat.

Machen Sie sich keine Sorgen, wenn Sie nicht alle Befehle sofort verstehen. Es gibt ein eigenes Kapitel (siehe Kapitel 16 [MUIbase programmieren], Seite 83), das alle Programmfunktionen detailliert beschreibt.

Jetzt haben wir ein lauffähiges Programm, aber wir müssen noch die Programmfunktion mit dem Knopf ‘Pedigree’ verbinden. Dazu wechseln wir in den Struktureditor, wählen ‘Control’ aus dem Tabellenbereich und doppel-klicken 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 Pseudobrueder 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 einer Person.
; Zuerst definieren wir die Variablen, die wir benötigen,
; z.B. "names", das die Namen der Kinder enthält.

```

```

(DEFUN children ()
  (LET ( (names "") (count 0) (parent Person) )

    ; Über alle Datensätze in Tabelle Person wird folgendes durchgeführt:
    ; Wenn Variable parent als Vater oder Mutter in anderen Datensätzen
    ;   auftritt, dann:
    ;   - füge den Namen zur Variablen names hinzu
    ;   - erhöhe count um 1.

    (FOR ALL Person DO
      (IF (OR (= parent Father) (= parent Mother))
        (
          (SETQ names (+ names Name "\n"))
          (SETQ count (1+ count))
        )
      )
    )

    ; Anschließend schreiben wir das Ergebnis in Control.Result
    ; Wenn die Person keine Kinder hat, wird eine Zeichenkette ausgegeben.
    ; Wenn er/sie Kinder hat, wird eine andere Zeichenkette ausgegeben.

    (SETQ Control.Result
      (+ Person.Name (IF (> count 0)
        (+ " ist der stolze Vorfahr von " (STR count) " Kind(ern).")
        " hat (noch :- ) keine Kinder."
      ))
    )

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

    (IF (<> count 0)
      (SETQ Control.Result
        (+ Control.Result "\n\n"
          (IF (= count 1)
            "Der Name des Kindes ist:"
            "Die Namen der Kinder sind:"
          )
        )
      )
    )

    ; 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 nun einen neuen Programmknopf, um das Programm auszuführen. Daher wechseln wir wieder in den Struktureditor und fügen einen Knopf in der Tabelle ‘`Control`’ hinzu. Benennen Sie es ‘`Children`’ und wählen Sie ‘`children`’ als Programmfunktion, die ausgelöst werden soll.

Um Ordnung in das Layout der Tabelle ‘`Control`’ 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 dann bei gedrückter *Shift*-Taste 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 ‘`Result`’ in ‘`Control`’ entfernen. Doppelklicken Sie auf ‘`Result`’ im Anzeigebereich und löschen Sie den Titel. Das Feld ‘`Result`’ ist nach wie vor zu sehen, nur dessen Name ist nicht mehr sichtbar.

Um es leichter zu machen, wenn wir mehrere Programme oder Felder in ‘`Control`’ hinzufügen wollen, sollten wir ‘`Result`’ und ‘`Programs`’ in eine vertikalen Gruppe fassen. Dazu wählen Sie ‘`Programme`’ und ‘`Result`’ aus und drücken dann auf ‘`Gruppe`’. Dies setzt ‘`Programme`’ und ‘`Result`’ 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 nur Ihre Fantasie und die Größe Ihrer Festplatte.

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.

Weiterhin können mehrere MUIbase-Instanzen auf das gleiche Projekt zugreifen. Mehrere MUIbase-Instanzen können das Projekt lesen, aber nur eine Instanz kann Änderungen vornehmen. Das funktioniert auch über mehrere Rechner auf einem Netzwerk, sofern das Projekt auf einem Netzwerklaufwerk liegt.

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. Tabellen können hinzugefügt, umbenannt und 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. Die Datensätze werden nicht notwendigerweise immer im Speicher gehalten sondern werden von Platte geladen oder auf ihr gespeichert, wenn es nötig sein sollte. Es gibt zwei Beschränkungen für die maximale Anzahl von Datensätzen einer Tabelle. Die eine basiert darauf, dass die Datensatznummer ein 32Bit Integerwert ist, was die Anzahl der Datensätze auf (theoretisch) 4294967295 begrenzt. Die andere und größere Einschränkung besteht darin, dass für jeden Datensatz ein kleiner Datensatzkopf im Speicher gehalten werden muss. Diese Einschränkungen machen MUIbase dennoch für Datensatzzahlen von 10000 und mehr verwendbar.

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.

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, Boolesch, 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 in 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 könnte eine Adressdatenbank den Namen, die Straße und den Ort einer Person jeweils in einem Zeichenkettenfeld speichern.

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 Dateiinhalt 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 bis 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 nicht den Wert NIL.

5.5.6 Datumsfelder

Datumsfelder speichern Kalenderdaten. 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 eine Uhrzeit oder einen Zeitraum. Zum Beispiel kann ein Zeitfeld verwendet werden, um die Spielzeiten von Musikstücken einer CD zu speichern.

Das Format zum Eingeben und Darstellen von Zeitangaben kann auf ‘HH:MM:SS’, ‘MM:SS’ oder ‘HH:MM’ festgelegt werden, wobei ‘HH’ für die Stunden, ‘MM’ für die Minuten und ‘SS’ für die Sekunden stehen. Intern wird ein Zeitwert als die Anzahl Sekunden seit 0 Uhr gespeichert. Zeitwerte größer als 23:59:59 sind bis zum maximalen Wert von 596523:14:07 möglich, aber negative Werte werden nicht unterstützt.

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

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 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. Jedes mal, 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: Boolesch, 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 Boolesch), undefiniert (bei Zeichenkette) oder leer (bei Liste) steht.

5.5.11 Knöpfe

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

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.	Nein
Ganzzahl	Zum Speichern von Ganzzahlen	Ja
Fließkommazahl	Zum Speichern von reellen Zahlen	Ja
Boolesch	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 - 596523:14:07)	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	$2 + 8$	$2 + 8$
Boolesch	$2 + 0$	$2 + 0$
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 Liebblingsschauspieler 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 setzt. 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 verwaltet, enthält zwei Tabellen, eine für die Filme und eine für die Schauspieler. Es sollen nun für jeden Film die Schauspieler ermittelt werden, die in diesem Film auftreten. In einem ersten Ansatz könnte hierfür ein Beziehungsfeld in der Tabelle der Schauspieler eingerichtet werden, das auf die Film-Tabelle verweist. Dies bedeutet jedoch, dass ein Schauspieler nur in höchstens einem Film auftreten kann, da nur ein Beziehungsfeld in der Tabelle der Schauspieler existiert. Es werden aber eine unbeschränkte Anzahl Beziehungen von der Tabelle der Schauspieler 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 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 muss dann ein neuer Datensatz erzeugt werden, in dem Film und Schauspieler in den entsprechenden Beziehungsfeldern gesetzt werden.

Wenn nun ermittelt werden soll, in welchen Filmen ein Schauspieler mitgewirkt hat, dann müssen nur alle Datensätze in der neuen Tabelle bestimmt werden, in der auf den gefragten Schauspieler verwiesen wird, und die Film-Datensätze, auf welche die gefundenen Datensätze verweisen, bilden dann die Menge der gesuchten Filme. Eine solche Suche kann von MUIbase automatisch durchgeführt und das Ergebnis in einer Listenansicht angezeigt werden.

Die folgende Tabelle zeigt ein Beispiel, wie eine Menge von Schauspieler 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 zu ermöglichen. Jedes Projekt besitzt sein eigenes Hauptfenster, in dem weitere GUI-Elemente (einschließlich Unterfenster) plaziert 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 kann im darüberliegenden Fenster ein Fensterknopf eingerichtet werden, welcher zum Öffnen des Unterfensters dient. Der Fensterknopf sieht wie ein normaler Textknopf aus, kann aber ein kleines Piktogramm enthalten, um sich von anderen Knöpfen hervorzuheben.

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

Ein Fenster kann andere GUI-Elemente als Kinder haben. Wenn kein Kind zu einem Fenster hinzugefügt wurde, dann wird ein voreingestelltes Bild 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 platziert werden, um die Datensatzinhalte anzuzeigen.

Masken können nicht in anderen Masken platziert sein, da dies zu einer Hierarchie von Masken und somit zu einer Hierarchie von Tabellen führen würde. Falls eine Hierarchie von Tabellen eingerichtet werden soll, so sollte dies durch eine Eins-zu-Mehrfach-Beziehung zwischen zwei Tabellen realisiert werden.

5.9.3 Panels

Ein Panel ist eine breite, schmale 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.

Ein Panel ist Teil einer Tabellenmaske und kann, falls gewünscht, weggelassen werden. Wird ein Panel für eine Maske eingerichtet, dann wird zusätzlicher ein Rahmen um die gesamte Maske gezeichnet.

5.9.4 Feldobjekte

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

Abhängig vom Typ des Feldes ist das GUI-Element entweder ein Zeichenkettenfeld (Typen Zeichenkette, Ganzzahl, Fließkommazahl, Datum und Zeit), ein Checkmark-Knopf (Typ Boolesch), ein Auswahlknopf oder eine Menge von Radioknöpfen (Typ Auswahl), ein Editorfeld (Typ mehrzeiliger Text), eine Popup-Listenansicht (Typ Beziehung), ein Textfeld, Checkmark-Feld oder eine Listenansicht (Typ virtuell), oder ein Text- bzw. Bildknopf (Typ Knopf). In einigen Fällen kann ein GUI-Element ein einfaches Textfeld 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 festem Text.

5.9.6 Bilder

Bilder können überall im Fenster angezeigt werden. Ein Bild kann ein Muster, ein einfaches Farbfeld oder ein Bild aus einer externen Datei sein. Die Größe des Bildes kann größenveränderbar oder fest sein.

Das Bild ist fest eingebaut. Sollen Bilder in einer Tabelle gespeichert werden, so kann hierfür ein Zeichenkettenfeld verwendet werden (siehe Abschnitt 5.5.1 [Zeichenketten], Seite 21).

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 platziert ihre Kinder von links nach rechts (horizontale Gruppe) oder von oben nach unten (vertikale Gruppe).

Eine Gruppe kann seine Kinderobjekte mit einem rechteckigen Rahmen umschließen, einen optionaler Titel oberhalb der Gruppe anzeigen, und Zwischenräume zwischen den Kinderobjekten einfügen.

5.9.9 Gewichtungsobjekte

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

5.9.10 Karteikarten-Gruppen

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

6 Projekte verwalten

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

6.1 Dateiformat

Ein MUIbase-Projekt besteht aus einer Sammlung von Dateien, die in einem eigenen Verzeichnis gespeichert werden. Dieses Verzeichnis wird erstmalig beim Speichern des Projekts erstellt. Machen Sie keinerlei Annahmen über die Verzeichnisstruktur oder Dateinamen in diesem Verzeichnis. Insbesondere dürfen weder Dateien aus diesem Verzeichnis entfernt noch in dieses Dateien oder weitere Verzeichnisse plaziert werden! Diese gehen verloren, 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 kann es noch mehrere Dateien geben die `.lock`, `.plock` und `.rlock` heißen. Diese regeln den Zugriff auf das Projekt. Mehrere MUIbase-Instanzen können ein Projekt lesen und benötigen dazu einen geteilten Zugriff auf die `.lock`-Datei. Aber nur eine Instanz darf das Projekt ändern, indem es einen Zugriff auf `.rlock` erhält, und speichern, indem der Zugriff auf `.plock` erfolgt. Zu beachten ist, dass der Zugriffsmechanismus auf bestimmten Garantien des Dateisystems beruht. Am besten ist es, das Teilen eines Projekts zwischen verschiedenen MUIbase-Instanzen auf einem Dateisystem zu testen, bevor man auf diesen Mechanismus vertraut.

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 Informationen enthalten 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 eine Schätzung und sollte nicht als exakter Wert angesehen werden. Insbesondere wenn viele Änderungen an der Struktur des Projekts durchgeführt wurden (Hinzufügen oder Entfernen von Feldern), ist der Wert sehr ungenau.

6.3 Neues Projekt

Mit MUIbase ist es möglich, mehrere Projekte gleichzeitig zu bearbeiten. Die Anzahl der Projekte ist nur durch den vorhandenen Speicher begrenzt. Soll ein weiteres Projekt gestartet werden, so wird Menüpunkt `Projekt - Neu` gewählt. Dies öffnet ein neues Fenster mit einem leeren Projekt. Es ist nun möglich die Struktur des neuen Projekts festzulegen (siehe Kapitel 15 [Struktureditor], Seite 67) oder ein bereits bestehendes Projekt von Platte zu laden (siehe Abschnitt 6.5 [Projekt öffnen], Seite 32).

6.4 Projekt leeren

Das aktuelle Projekt lässt sich durch Menüpunkt ‘Projekt - Leeren - Projekt’ zurücksetzen. Dies schließt das aktuelle Projekt und ersetzt es durch ein leeres Projekt. In diesem Modus befindet sich MUIbase automatisch nach dem Starten ohne Angabe eines Projekts.

Über den Menüpunkt ‘Projekt - Leeren - 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.

6.5 Projekt öffnen

Um ein Projekt zu laden, wird ‘Projekt - Öffnen - Projekt’ aufgerufen. Dies öffnet ein Dateiauswahlfenster, in dem ein Projekt ausgewählt werden kann. Es gibt auch mehrere Demoprojekte, welche die Fähigkeiten von MUIbase zeigen. Um eines dieser Demoprojekte zu laden, wird ‘Projekt - Öffnen - Demo’ aufgerufen.

Falls das geladene Projekt die Programmquellen als extern gesetzt hat, so wird nach dem Öffnen die externe Quelldatei erzeugt (siehe Abschnitt 16.2 [Externe Programquellen], Seite 83).

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

Man kann auch eine neue Version eines Projekts laden, indem man Menüpunkt ‘Project - Neu laden’ wählt.

6.6 Projekt speichern

Alle Änderungen an einem Projekt werden nur im Speicher durchgeführt oder beim Auslagern von Datensätzen temporär gespeichert (siehe Abschnitt 6.9 [Datensätze auslagern], Seite 34). 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 so der Benutzer entscheidet, wann das Projekt gespeichert wird, und der Benutzer durch Wählen von ‘Projekt - Zurück zum Gespeicherten’ immer zur zuletzt gespeicherten Version des Projekts zurückkehren kann. Dieses Verfahren entspricht den Befehlen ‘COMMIT’ und ‘ROLLBACK’ in SQL-Datenbanksystemen.

Beim Speichern eines Projekts werden alle geänderten Datensätze auf Platte geschrieben und die Datei ‘Structure.mb’ neu erzeugt. Vor dem Erzeugen der Datei benennt MUIbase eine möglicherweise existierende Datei ‘Structure.mb’ in ‘Structure.old’ um, um eine Sicherheitskopie zu haben, falls das Speichern misslingt.

Dieser Mechanismus garantiert schnelles Laden und Speichern, er erfordert aber gelegentliches Neuorganisieren (Umschichten) der Daten. 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 projektbezogenen Dateien neu. Das alte Verzeichnis wird nach erfolgreicher Operation gelöscht.

Wenn Änderungen an der Struktur des Projekts durchgeführt wurden, wie z.B. das Einfügen eines neuen Feldes in einer Tabelle, so ist ein Umschichten auch sinnvoll. Strukturänderungen 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 benötigt nur wenig Zeit. Je länger diese Liste jedoch wird, desto mehr Zeit wird beim Laden eines Datensatzes verbraucht. Das Umschichten eines Projekts führt dazu, dass die **‘todo’**-Liste an allen Datensätzen durchgeführt wird. Wenn also viele Änderungen am Projekt durchgeführt wurden, so verkürzt das Umschichten des Projekts die Zeit zum Laden einzelner Datensätze.

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, welches nach einem neuen Namen für das Projekt fragt.

6.7 Admin- und Benutzermodus

MUIbase arbeitet entweder im Admin- (voreingestellt) oder im Benutzermodus. Durch Wählen der Menüpunkte **‘Project - Zum Admin-Modus wechseln’** oder **‘Project - Zum Benutzer-Modus wechseln’** kann zwischen den beiden Modi gewechselt werden. Im Benutzermodus sind mehrere Menüpunkte gesperrt und Struktur-, Programm- und Abfrageeditor sind nicht verfügbar. Es sind daher nur die Grundfunktionen der Datensatzbearbeitung möglich. Im Admin-Modus sind alle Operationen erlaubt.

Durch Auswahl des Menüpunkts **‘Project - Admin-Passwort ändern’** kann ein Admin-Passwort für ein Projekt gesetzt werden. Wurde dieses einmal gesetzt, so muss bei jedem Wechsel in den Admin-Modus das Passwort bestätigt werden. Bei falscher Passworteingabe wird der Zugriff verweigert und das Projekt verbleibt im Benutzermodus.

Wird ein Projekt geöffnet, für das ein Admin-Passwort gesetzt wurde, so wird dieses im Benutzermodus gestartet. Andernfalls (es wurde kein Admin-Passwort gesetzt), wird das Projekt im Admin-Modus gestartet.

6.8 Integrität der Daten prüfen

MUIbase kann überprüfen, ob die Daten in einem Projekt noch gültig sind und nicht durch Systemabstürze oder durch Programme beschädigt wurden. Der Menüpunkt **‘Projekt - Prüfe Integrität der Daten’** startet diesen Prozess.

Normalerweise sollte diese Funktionalität nie benötigt werden und MUIbase immer melden, dass die Integrität der Daten perfekt ist. Sollte es aber doch mal passieren, dass ein Projekt **‘interne Fehler’** aufweist, 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.

6.9 Datensätze auslagern

MUIbase muss nicht alle Datensätze eines Projekts im Speicher halten. Dadurch wird das Laden und Speichern von Projekten erheblich 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 dennoch durch den verfügbaren Speicher begrenzt, da jeder Datensatzkopf 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 (siehe Abschnitt 7.2.1 [Datensatzspeicher], Seite 37). MUIbase belegt nicht vorab den Speicher der angegebenen Größe, sondern prüft von Zeit zu Zeit, ob die Größe des momentan belegten Speichers größer ist als die angegebene.

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

MUIbase verwaltet eine Frei-Liste für jede Datensatzdatei. Wird ein Datensatz gelöscht, so wird der zugehörige Dateiplatz zu dieser Frei-Liste hinzugefügt. 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 Stand der letzten Speicheroperation zurückgekehrt werden kann. Es werden keine Bereiche beschädigt, die frei aber dennoch von Datensätzen belegt sind, die beim Neuladen des Projekts erreicht werden könnten.

6.10 Projekt schließen

Wenn die Bearbeitung eines Projekt abgeschlossen ist, kann es über den Menüpunkt **‘Projekt - Schließen’** geschlossen werden. Dies gibt den Speicher und alle Ressourcen des Projekts frei. Falls das Projekt Änderungen besitzt, die noch nicht gespeichert wurden, so bietet eine Sicherheitsmeldung an, es zu speichern, fortzusetzen oder die Operation abzubrechen.

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

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.

Der Satz aller Einstellungen ist aufgeteilt in Benutzereinstellungen und Projekteinstellungen.

7.1 Benutzereinstellungen

Die Benutzereinstellungen umfassen Einstellungen, welche von der Wahl des Benutzers abhängen, z.B. bezogen auf Sprache, Land oder Geschmack des Benutzers. Die Benutzereinstellungen werden im oberen Teil des Menüs 'Einstellungen' angezeigt und können dort geändert werden.

Benutzereinstellungen werden in der Umgebung des Benutzers gespeichert. Auf Windows, Mac OS und Linux werden sie im Heimatverzeichnis des Anwenders in der Datei `.MUIbase.prefs` gespeichert. Auf dem Amiga werden sie in `ENV:MUIbase.prefs` und `ENVARC:MUIbase.prefs` abgelegt.

Die folgenden Punkte sind Bestandteil der Benutzereinstellungen. Sobald einer dieser Punkte im Menü 'Einstellungen' geändert wird, werden die Einstellungen auf Platte gespeichert, um diese dauerhaft zu machen.

7.1.1 Formate

Über Menüpunkt 'Einstellungen - Formate' können die Formate zum Darstellen von Fließkommazahlen und Kalenderdaten festgelegt werden. Nach dem Auswählen erscheint ein Fenster, das folgende Punkte enthält:

- ein Feld 'Fließkommazahlenformat' zum Setzen des Dezimalzeichens von Fließkommazahlen. Es kann zwischen '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 der 'Locale'-Umgebung des jeweiligen Betriebssystems ermittelt.

Wenn alle Einstellungen getätigt wurden, verlässt man über 'Ok' das Fenster und aktualisiert die Anzeige.

7.1.2 Externer Editor

Neben dem intern eingebauten Editor (des verwendeten GUI-Toolkits), erlaubt MUIbase auch, Textinhalte durch einen externen Editor zu bearbeiten (z.B. enthält das Kontextmenü des internen Editors einen Menüpunkt zum Aufrufen des externen Editors, um den Inhalt zu editieren). Der Name des Editors und seine Parameter werden über den Menüpunkt 'Einstellungen - Externer Editor' angegeben. Hier sollte der Befehl eingegeben werden, der ausgeführt wird, wenn der externe Editor aufgerufen wird. Die spezielle Zeichenkette '%f' kann hierbei für den Dateinamen verwendet werden und wird vor Ausführung durch

den tatsächlichen Dateinamen (oder den Namen einer temporären Datei, die MUIbase für den Datenaustausch generiert) ersetzt.

Zum Beispiel kann auf Linux `emacs %f` eingegeben werden, um den leistungsstarken GNU-Emacs-Editor zu benutzen, oder auf dem Amiga kann `CED %f -keepio` verwendet werden, um zu dem beliebten CED-Editor als externer Editor zu wechseln.

Voreingestellt ist `Notepad %f` auf Windows, `open -tWn %f` auf Mac OS, `gvim -f %f` auf Linux, und `Ed %f` auf dem Amiga.

Unter Mac OS ist zu beachten, dass die Voreinstellung den externen Editor synchron startet, d.h. MUIbase wartet bis der Editor beendet wird (siehe die `-W` und `-n` Optionen in `open --help`). Dies ist erforderlich, wenn Sie Menüpunkt `Externer Editor` im Kontextmenü eines mehrzeiligen Textfeldes benutzen (siehe Abschnitt 9.3 [Datensätze verändern], Seite 45) oder den Befehl `EDIT` für die Programmierung von MUIbase verwenden (siehe Abschnitt 16.23.1 [EDIT], Seite 150).

Auf dem Amiga kann zusätzlich `%p` in der Befehlszeichenkette verwendet werden, welches bei Aufruf durch den Namen des Public-Screens, auf dem MUIbase läuft, ersetzt wird.

7.1.3 Externer Anzeiger

MUIbase verwendet einen externen Anzeiger, um den Inhalt von externen Dateien wie Bilder, Filme oder andere Dokumente, anzuzeigen. Zum Beispiel kann der Zeichenkettentyp verwendet werden, um Dateinamen in einer Tabelle zu speichern und diese mittels des externen Anzeigers darzustellen. Zum Eingeben des externen Anzeigers wird der Menüpunkt `Einstellungen - Externer Anzeiger` aufgerufen. Wie beim externen Editor (siehe Abschnitt 7.1.2 [Externer Editor], Seite 35) muss hier eine Befehlszeichenkette eingegeben werden.

Voreingestellt ist `%f` auf Windows (Verwendung von ShellExecute), `open %f` on Mac OS, `gnome-open %f` auf Linux, `Open %f` auf MorphOS und `Multiview %f` auf anderen Amiga-Systemen.

Unter Mac OS ist zu beachten, dass die Voreinstellung den externen Anzeiger immer asynchron startet, d.h. MUIbase wartet nicht bis der Anzeiger beendet wird. Falls Sie Mac OS 10.5 oder höher verwenden, so können Sie mittels `open -Wn %f` das synchrone Anzeigen ermöglichen. Dies ist erforderlich, wenn Sie den Befehl `VIEW` für die Programmierung von MUIbase verwenden (siehe Abschnitt 16.23.3 [VIEW], Seite 151).

Auf dem Amiga kann zusätzlich `%p` in der Befehlszeichenkette verwendet werden, welches bei Aufruf durch den Namen des Public-Screens, auf dem MUIbase läuft, ersetzt wird.

Ab MUIbase 4.0 wird der externe Anzeiger automatisch anhand Ihrer Systemeinstellungen auf Windows und Linux gewählt und diese Einstellung ist nicht verfügbar.

7.1.4 Extra-Knöpfe in Tab-Kette

In der grafischen Benutzeroberfläche können verschiedenen Extra-Knöpfe vorhanden sein. Unter Extra-Knöpfe fallen Popup-Knöpfe, z.B. Datei-, Zeichensatz- oder Listenansicht-Popups neben einem Zeichenkettenfeld, Auto-Anzeige- und Filter-Knöpfe von Referenzfeldern, sowie Anzeige-Knöpfe neben Zeichenkettenfeldern.

Diese Knöpfe sind normalerweise nicht in der Aktivierungskette eingebunden, was bedeutet, dass sie nicht mit der `Tab`-Taste erreichbar sind. Wird jedoch der Menüpunkt

‘Einstellungen - Extra-Knöpfe in Tab-Kette’ gesetzt, dann werden diese Knöpfe ebenfalls in die Tab-Kette eingefügt. Voreingestellt ist gesetzt.

Auf dem Amiga ist zu beachten, dass das Ändern des Status dieses Menüpunkts erst dann eine Auswirkung hat, wenn die Benutzeroberfläche neu generiert wird, z.B. durch das Wechseln zum Struktureditor und zurück zur Benutzeroberfläche.

7.1.5 Weiterspringen bei Enter

Ist der Cursor auf einem editierbaren einzeiligen Textfeld in der Benutzeroberfläche eines Projekts und wird die *Enter*-Taste gedrückt, so kann der Cursor entweder zum nächsten Feld springen oder im aktuellen Feld bleiben.

Ist Menüpunkt ‘Einstellungen - Weiterspringen bei <Enter>’ gesetzt, so springt der Cursor in das nächste Feld in der Benutzeroberfläche, ansonsten bleibt der Cursor auf dem aktuellen Feld. Voreingestellt ist gesetzt.

Zu beachten ist, dass in jedem Falle bei drücken von *Enter* der eingegebene Text bestätigt und gespeichert wird.

7.1.6 Beenden bestätigen

Wird versucht MUIbase zu beenden und es liegen ungespeicherte Projekte vor, so wird zuerst durch eine Sicherheitsabfrage um Erlaubnis gefragt. MUIbase beendet sich jedoch stillschweigend, falls alle Projekte schon gespeichert wurden.

Soll MUIbase immer eine Sicherheitsabfrage beim Beenden anzeigen, so muss der Menüpunkt ‘Einstellungen - Beenden bestätigen’ gesetzt sein. In diesem Fall erhält man immer eine Sicherheitsabfrage, wenn der Menüpunkt ‘Projekt - Beenden’ ausgewählt oder das letzte geöffnete Projekt geschlossen wird. Voreingestellt ist ungesetzt.

7.1.7 MUI

Nur für Amiga. Da MUIbase auf dem Amiga eine MUI-Anwendung ist, lassen sich auch die MUI-Voreinstellungen für diese Anwendung verändern, indem Menüpunkt ‘Einstellungen - MUI’ ausgewählt wird.

7.2 Projekteinstellungen

Die Projekteinstellungen beinhalten die Einstellungen, welche mit einem Projekt gespeichert werden. Diese Einstellungen werden im unteren Teil des Menüs ‘Einstellungen’ und im Menü ‘Programm’ angezeigt und können dort geändert werden (die Aufteilung in diese zwei Bereiche erhöht die Übersichtlichkeit der Menüs).

Die folgenden Punkte gehören zu den Projekteinstellungen. Bei jeder Änderung einer dieser Punkte wird der Zähler der Änderungen des Projekts erhöht.

7.2.1 Datensatzspeicher

MUIbase muss nicht alle Datensätze eines Projekts im Speicher halten. Stattdessen verwendet es einen Puffer, der nur eine kleine Anzahl von Datensätzen enthält. 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 benutzt MUIbase bis zu 2MB für die Datensätze beider Projekte.

MUIbase reserviert den Speicher nicht a priori am Stück, sondern verwendet ein dynamisches Allokierungsschema. Die festgelegte Puffergröße ist auch nur eine weiche Grenze und MUIbase kann diese gelegentlich überziehen.

Ist die Puffer einmal gefüllt oder erhält MUIbase keinen Speicher mehr, dann werden alle Datensätze aus dem Puffer entfernt. Dies bedeutet, dass unveränderte Datensätze einfach freigegeben und veränderte Datensätze zuerst auf Platte geschrieben und danach freigegeben werden (siehe Abschnitt 6.9 [Datensätze auslagern], Seite 34).

Je größer der Wert für den Puffer ist, desto größer ist der Geschwindigkeitszuwachs beim Zugriff auf die Datensätze, da mehr Datensätze im Speicher gehalten werden können und weniger Plattenzugriffe notwendig sind. Ist genug Speicher vorhanden, um alle Datensätze im Speicher zu halten, und eine Obergrenze festgelegt, die hoch genug ist (z.B. ‘unbegrenzt’), so arbeitet MUIbase mit optimaler Geschwindigkeit.

Voreingestellt ist ‘unbegrenzt’.

7.2.2 Datensätze löschen bestätigen

Der Menüpunkt ‘Einstellungen - Datensätze löschen bestätigen’ sollte gesetzt sein, wenn MUIbase bei jedem Löschen eines Datensatzes eine Sicherheitsabfrage ausführen soll. Ist er ungesetzt, so werden Datensätze stillschweigend gelöscht. Voreingestellt ist gesetzt.

7.2.3 Pfade relativ zu einem Projekt

Sollen externen Dateien (z.B. Dokumente oder Bilder) in einer Datenbank referenziert werden, so muss der jeweilige Dateiname im Projekt gespeichert werden. Zu diesem Zweck können absolute Pfade und Pfade mit Assign-Namen (siehe Abschnitt 3.8 [Konventionen für Dateinamen], Seite 8) verwendet werden. Eine andere Möglichkeit besteht darin, externe Dateien relativ zu dem Verzeichnis des Projekts abzulegen (Hinweis: dies ist nicht das Projektverzeichnis selbst, sondern das Verzeichnis, welches das Projekt enthält).

Ist Menüpunkt ‘Einstellungen - Pfade relativ zu Verzeichnis eines Projekts’ aktiviert, so wechselt MUIbase das aktuelle Verzeichnis zu dem Verzeichnis des Projekts. Dies bedeutet im Falle mehrerer gleichzeitig geöffneter Projekte, dass MUIbase Verzeichnisse wechselt, je nachdem welches Projekt gerade aktiv ist. Wird dieser Menüpunkt für ein Projekt aktiviert, so können Dateinamen relativ zu dem Verzeichnis des Projekts angegeben werden. Dies macht ein Projekt unabhängig davon, wo es im Dateisystem gespeichert wird.

Bleibt der Menüpunkt ungesetzt, so benutzt MUIbase das initiale Arbeitsverzeichnis bei Programmstart.

Voreingestellt ist ungesetzt.

7.2.4 Automatisches Neuladen bestätigen

Findet MUIbase eine aktualisierte Version Ihres Projekts im Dateisystem, so wird das Projekt automatisch neu geladen. Ist Menüpunkt ‘Einstellungen - Automatisches Neuladen bestätigen’ gesetzt, so erscheint zuerst eine Sicherheitsabfrage, um das Neuladen zu bestätigen. Falls ungesetzt, dann wird das Projekt stillschweigend geladen. Voreingestellt ist gesetzt.

7.2.5 Speichern & Umschichten bestätigen

Das Speichern und Umschichten eines Projekts kann je nach dem, wie groß das Projekt ist, etwas Zeit in Anspruch nehmen, Falls der Menüpunkt ‘Projekt - Speichern & Umschichten’ oder ‘Projekt - Speichern & Umschichten als’ ausgewählt wird, erscheint daher eine Sicherheitsabfrage, die diese Operationen erst bestätigen soll. Diese Abfrage erscheint nur dann, wenn der Menüpunkt ‘Einstellungen - Speichern & Umschichten bestätigen’ gesetzt ist. Es kann also durch das Nicht-Setzen des Menüpunktes die Abfrage deaktiviert werden. Voreingestellt ist gesetzt.

7.2.6 Programmquellen

Die Programmquellen eines Projekts können in Menüpunkt ‘Programm - Quellen’ auf ‘Intern’ oder ‘Extern’ gesetzt werden. Sind sie auf ‘Intern’ gesetzt, so kann der interne Editor benutzt werden, um das Program eines Projekts zu editieren und zu übersetzen. Dies ist die Voreinstellung. Falls Sie lieber einen externen Editor zum Programmieren verwenden wollen, so wählen Sie ‘Extern’ und geben den Namen einer neuen Datei ein, in welche MUIbase dann die Programmquellen schreibt. Sie können nun die Quelldatei in ihren Lieblingseditor laden und editieren. Für weitere Informationen über diese Funktion siehe Abschnitt 16.2 [Externe Programmquellen], Seite 83.

Bitte beachten Sie, dass beim Zurücksetzen von ‘Extern’ nach ‘Internal’ die letzte erfolgreich übersetzte Version des Programmes behalten wird.

7.2.7 Externe Programmquellen aufräumen

Menüpunkt ‘Einstellungen - Externe Programmquellen aufräumen’ bestimmt, ob die externe Programmquellendatei eines Projekts gelöscht werden soll, wenn das Projekt geschlossen wird oder wenn Menüpunkt ‘Programm - Quellen’ zurück auf ‘Intern’ gesetzt wird. Für mehr Informationen über diese Option siehe Abschnitt 16.2 [Externe Programmquellen], Seite 83. Voreingestellt ist gesetzt.

7.2.8 Programm-Debug-Information

Zum Kompilieren des Programms eines Projekts kann man wählen, ob Debug-Informationen in den ausführbaren Code eingebunden werden sollen. Wird ohne Debug-Informationen 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 Debug-Informationen kompiliert, dann erhält man auch die genaue Fehlerposition. Das Kompilieren mit Debug-Information ist ein bisschen langsamer, benötigt mehr Speicher und das resultierende Programm ist geringfügig langsamer.

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

7.2.9 Veraltete Funktionen

Seit MUIbase Version 2.7 sind ein paar Programmfunktionen veraltet (siehe Abschnitt 16.30 [Liste veralteter Funktionen], Seite 166. Die Funktionalität dieser veralteten Funktionen wurde ersetzt durch andere Mechanismen und die Funktionen arbeiten nicht mehr wie

erwartet. Wird ein Projekt geöffnet, das Programmfunktionen enthält, die nun als veraltet gelten, so kann gewählt werden, wie diese zu behandeln sind.

Menüpunkt **‘Programm - Veraltete Funktionen’** gibt an, was passieren soll, wenn eine veraltete Funktion aufgerufen wird. Wird **‘Schweigend ignorieren’** ausgewählt, so wird jeder Aufruf einer veralteten Funktion ignoriert und die Programmausführung fährt fort, indem über die Funktion gesprungen wird. **‘Warnung bei Aufruf’** öffnet bei jedem Aufruf einer veralteten Funktion ein Dialogfenster, das den Benutzer informiert und anbietet, die Programmausführung fortzusetzen oder mit einer Fehlermeldung abzubrechen. Dies ist die Voreinstellung. Wird **‘Behandlung als Fehler’** gewählt, so generiert jeder Aufruf einer veralteten Funktion einen Fehler und die Übersetzung von Programmen, welche veraltete Funktionen enthalten, schlägt mit einer entsprechenden Meldung fehl.

Es wird empfohlen **‘Behandlung als Fehler’** zu wählen, nachdem alle Aufrufe von veralteten Funktionen im Projektprogramm entfernt wurden. Um herauszufinden, ob ein Projekt veraltete Funktionen benutzt, wählt man **‘Behandlung als Fehler’** und übersetzt das Programm neu.

7.2.10 Trigger-Funktionen sortieren

Ist Menüpunkt **‘Einstellungen - Trigger-Funktionen sortieren’** gesetzt, so werden als Trigger verfügbare Funktionen für die Anzeige in den Fenstern für das Erstellen und Ändern von Tabellen (siehe Abschnitt 15.1.1 [Tabellen erstellen], Seite 67) und für das Erstellen und Ändern von Feldern (siehe Abschnitt 15.2.1 [Felder erstellen], Seite 69) alphabetisch sortiert. Andernfalls werden Funktionen in der Reihenfolge aufgelistet, in der sie im Program des Projekts vorkommen. Voreingestellt ist ungesetzt.

7.2.11 Programm-Include-Verzeichnis

Die Programmiermöglichkeiten von MUIbase erlauben es, externe Quellen in das Programm des Projekts einzubinden (siehe Abschnitt 16.3.3 [#include], Seite 85, für mehr Informationen). Der Menüpunkt **‘Programm - Include-Verzeichnis’** erlaubt das Setzen eines Verzeichnisses, in dem MUIbase nach solchen Include-Dateien sucht. Voreingestellt ist **‘MUIbase:Include’**.

7.2.12 Programm-Ausgabedatei

Beim Ausführen eines MUIbase-Programms werden sämtliche Ausgaben nach **‘stdout’** in eine Datei ausgegeben. Der Name dieser Datei kann in einem Dialogfenster eingegeben werden, das nach Auswählen des Menüpunkts **‘Programm - Ausgabedatei’** erscheint. Es kann hier auch angegeben werden, ob die Ausgabe an eine bestehende Datei angehängt, oder die Datei gelöscht und neu erzeugt werden soll.

Auf Windows, Mac OS und Linux kann die Ausgabe an ein externes Programm weitergeleitet werden, das die Daten verarbeitet. Hierfür muss das erste Zeichen das Pipe-Symbol **‘|’** gefolgt von dem externen Programm und seinen Parametern sein. Das externe Programm muss die Daten von der Standardeingabe lesen. Dies ermöglicht auf Linux zum Beispiel die folgenden Umleitungen:

- `|lpr` gibt die Ausgabe auf dem Drucker aus.
- `|mknod /tmp/pipe p; (xterm -e less -f /tmp/pipe &); cat > /tmp/pipe; rm /tmp/pipe` zeigt die Ausgabe mittels dem Programm **‘less’** in einem eigenen **‘xterm’**-Fenster an.

- `/dev/pts/0` schreibt die Ausgabe in ein Terminal, das mit `'pts/0'` verbunden ist. Auf manchen Arbeitsumgebungen, z.B. KDE, wacht ein Prozess über dieses Terminal, so dass die Ausgaben in einem Fenster landen.
- `/dev/stdout` gibt die Ausgaben an die Standardausgabe von MUIbase weiter. Dies ist die Voreinstellung.

Auf dem Amiga gibt es für die Umleitung spezielle Dateinamen, 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, von dem aus MUIbase gestartet wurde. Dies ist die Voreinstellung.

7.3 Als Voreinstellungen speichern

Die Projekteinstellungen können für jedes Projekt individuell angepasst werden, d.h. verschiedene Projekte können verschiedene Einstellungen haben. Für neue Projekte wird eine Voreinstellung verwendet. Diese Voreinstellung wird zusammen mit den Benutzereinstellungen in der Umgebung des Benutzers abgelegt (siehe Abschnitt 7.1 [Benutzereinstellungen], Seite 35).

Um die Voreinstellung Ihren Wünschen anzupassen, können die Einstellungen des aktuellen Projekts als Voreinstellung für neue Projekte durch Auswahl von Menüpunkt `'Einstellungen - Als Voreinstellung speichern'` gesichert werden.

8 Log

Dieses Kapitel beschreibt die Möglichkeiten von MUIbase Änderungen zu protokollieren, inklusive der ‘_Log’-Tabelle, wie das Protokollieren aktiviert wird, das Setzen einer Logbeschriftung für neue Logeinträge, wie das Log importiert, exportiert und geleert wird, und wie alle Logeinträge angesehen werden können.

8.1 Log-Tabelle

Für das Protokollieren von Änderungen verwaltet MUIbase eine interne Tabelle mit dem Namen ‘_Log’. Zu beachten ist der Unterstrich als erstes Zeichen, was bedeutet, dass diese Tabelle eine von MUIbase verwaltete Systemtabelle ist, die nicht geändert werden kann. Die Tabelle enthält die folgenden Felder, welche alle vom Typ Zeichenkette sind:

Name	Beschreibung
‘_Label’	ein Feld, welches mittels Menüpunkt ‘Log - Logbeschriftung setzen’ eingegeben oder durch eine Auslösefunktion mit dem Namen logLabel berechnet werden kann (siehe Abschnitt 8.3 [Logbeschriftung setzen], Seite 43).
‘_Date’	Datum des Logeintrags.
‘_Time’	Uhrzeit des Logeintrags.
‘_Action’	ausgeführte Aktion: entweder ‘New record’, ‘Delete record’ oder ‘Change value’.
‘_Table’	Name der Tabelle, in welcher die Änderungsaktion ausgeführt wurde.
‘_Record’ de zum Zeitpunkt	Nummer des Datensatzes, in welchem die Aktion ausgeführt wurde der Änderung. Zu beachten ist, dass die Datensatznummer sich ändern kann wenn Datensätze hinzugefügt oder gelöscht werden, oder wenn Datensätze in einer Tabelle neu sortiert werden.
‘_Description’ ziert werden	Beschreibung des Datensatzes, in welchem die Aktion ausgeführt wurde. Die Datensatzbeschriftung kann im Tabellenobjekt spezifiziert werden (siehe Abschnitt 15.3.2 [Tabellenobjekteditor], Seite 73).
‘_Attribute’ ses Feld	Falls die Aktion den Eintrag ‘Change value’ hat, dann enthält dieses Feld den Namen des Tabellenfelds, welches geändert wurde.
‘_OldValue’	Alter Wert des Datensatzfelds (für Aktion ‘Change value’).
‘_NewValue’	Neuer Wert des Datensatzfelds (für Aktion ‘Change value’).

Auf die ‘_Log’-Tabelle kann wie auf jede andere Tabelle eines Projekts zugegriffen werden. Z.B. kann eine Select-from-where-Abfrage

```
SELECT * from _Log
```

im Abfrageeditor ausgeführt werden, umn eine Liste aller Änderungen zu erhalten (siehe Abschnitt 14.1 [Select-from-where Abfragen], Seite 60, und Abschnitt 14.2 [Abfrageeditor], Seite 60).

8.2 Protokollieren aktivieren

Zu Anfang ist das Protokollieren in einem Projekt nicht aktiv. Um das Protokollieren von Änderungen zu starten, wählt man Menüpunkt **‘Log - Protokollieren aktiv’**. Der Zustand dieses Menüpunkts wird in der Projektdatei gespeichert, so dass, einmal aktiviert und gespeichert, Änderungen weiterhin protokolliert werden, wenn das Projekt wieder geöffnet wird. Man deaktiviert den Menüpunkt, um das Protokollieren zu stoppen.

Zu beachten ist, dass nur für die Tabellen und Felder Änderungen protokolliert werden, für welche die Eigenschaft **‘Protokolliere Änderungen’** gesetzt wurde (siehe Abschnitt 15.1.1 [Tabellen erstellen], Seite 67, und Abschnitt 15.2.1 [Felder erstellen], Seite 69). Die Änderungen, die protokolliert werden, sind das Anlegen eines neuen Datensatzes, das Löschen eines Datensatzes und das Ändern eines Felds in einem Datensatz. Für jede dieser Aktionen wird ein neuer Logeintrag erstellt und in die **‘_Log’**-Tabelle hinzugefügt.

8.3 Logbeschriftung setzen

Die **‘_Log’**-Tabelle hat ein Feld **‘_Label’**, das beliebig durch das Projekt gesetzt werden kann. Wählt man Menüpunkt **‘Log - Logbeschriftung setzen’**, so erscheint ein Dialogfenster für die Eingabe des Beschriftungstexts.

Es ist auch möglich die Beschriftung durch eine Programmfunktion anzugeben. Die Funktion muss den Namen `logLabel` haben und wird für jeden neuen Logeintrag aufgerufen. Die Funktion hat keine Argumente und der zurückgegebene Ausdruck wird in eine Zeichenkette umgewandelt und als Beschriftungstext verwendet. Für weitere Informationen über diese Auslösefunktion, siehe Abschnitt 16.29.6 [`logLabel`], Seite 159.

Zu beachten ist, dass falls eine Funktion `logLabel` vorhanden ist, der Menüpunkt **‘Log - Logbeschriftung setzen’** deaktiviert ist.

8.4 Log importieren

Es ist möglich Logeinträge aus einer externen Datei zu importieren. Dies kann nützlich sein, wenn ältere Logeinträge gelöscht wurden. Nach Wahl des Menüpunkts **‘Log - Importiere Log’** erscheint ein Fenster, das den Import von Datensätzen der **‘_Log’**-Tabelle ermöglicht. Siehe Abschnitt 13.3 [Datensätze importieren], Seite 58, für eine Beschreibung aller Importoptionen.

8.5 Log exportieren

Wird die **‘_Log’**-Tabelle groß, so kann es nützlich sein, ältere Logeinträge in eine externe Datei auszulagern. Um dies zu tun wählt man Menüpunkt **‘Log - Exportiere Log’**. Dies öffnet ein Fenster für den Export der Datensätze der **‘_Log’**-Tabelle in eine Datei. Siehe Abschnitt 13.4 [Datensätze exportieren], Seite 58, für eine Beschreibung aller Exportoptionen.

8.6 Log löschen

Um alle Logeinträge zu löschen, wählt man Menüpunkt **‘Log - Log löschen’**. Ein typischer Fall für das Löschen aller Logeinträge ist, wenn die **‘_Log’**-Tabelle zu groß wird. Stellen Sie

sicher, dass alle Einträge in eine externe Datei exportiert wurden, bevor das Log geleert wird.

8.7 Log ansehen

Wählt man Menüpunkt ‘Log - Log ansehen’, dann wird der Abfrageeditor mit einer vordefinierten Abfrage geöffnet, welche alle Logeinträge auflistet. Man kann die Abfrage modifizieren und den Titel ändern, und MUIbase merkt sich die Abfrage. Für weitere Informationen, siehe Abschnitt 14.2 [Abfrageeditor], Seite 60.

9 Datensatzbearbeitung

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

9.1 Aktive Objekte

MUIbase verwendet einen Cursor, um anzuzeigen, welches Objekt gerade aktiv ist. Ist das aktive Objekt ein Zeichenkettenobjekt, so erscheint ein normaler Textcursor. Andere Objekte erhalten einen besonderen Rahmen. Man kann das aktive Object durch drücken der Tasten *Tab* oder *Shift-Tab* wechseln. Wird die Taste *Help* oder *F1* gedrückt, so wird ein externer Anzeiger mit hilfreichen Informationen über das aktive Objekt geladen.

Die Tabelle, in der das aktive Objekt liegt, wird *aktive Tabelle* genannt. Das Panel einer Tabelle kann ebenfalls aktiviert werden. Dadurch wird garantiert, dass eine Tabelle jederzeit aktiviert werden kann, selbst wenn sie keine aktivierbaren Objekte enthält.

Unter Windows, Mac OS und Linux hat jede Tabelle ein Kontextmenü, welches Menüpunkte zur Manipulation der Tabelle enthält. Dieses Kontextmenü kann durch Drücken der rechten Maustaste innerhalb der Tabellenmaske (aber außerhalb anderer GUI-Objekte, die ebenfalls ein Kontextmenü besitzen) geöffnet werden.

Auf Mac OS und Amiga sind die Tabellen-Menüpunkte Bestandteil des globalen Menüs, welches am oberen Bildschirmrand zu finden ist.

9.2 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 15.1.1 [Tabellen erstellen], Seite 67), dann wird diese Auslösefunktion zum Erzeugen des Datensatzes aufgerufen. Mehr über diesen Mechanismus, siehe Abschnitt 16.29.8 [Auslösefunktion Neu], Seite 160.

9.3 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, Kalenderdaten, 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).

9.3.1 Zeichenkettenfelder mit einem Popup-Knopf

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

ein 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 weiterer kleiner Knopf erscheinen. Mit diesem Knopf wird ein externer Anzeiger gestartet, mit dem der Inhalt der Datei angezeigt werden kann, der im Zeichenkettenfeld angegeben ist.

9.3.2 Eingabe von Ganzzahlwerten

Bei der Eingabe von Ganzzahlen, kann man eine Octalzahlnotation (führende 0), eine Hexadezimalzahlnotation (führendes 0x) oder die gewöhnliche Dezimalzahlnotation benutzen.

9.3.3 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.

9.3.4 Eingabe von Auswahlwerten

Bei Auswahlfeldern kann ein Wert durch Anklicken oder mit den Cursortasten *Up* und *Down* zum Durchforsten aller Auswahlelemente ausgewählt werden.

9.3.5 Eingabe von Datumswerten

Datumswerte können in einem der Formate ‘DD.MM.YYYY’, ‘MM/DD/YYYY’ oder ‘YYYY-MM-DD’ 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, um das aktuelle Jahr zu verwenden.

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.

9.3.6 Eingabe von Zeitwerten

Für Zeitwerte wird das Eingabeformat im Struktureditor festgelegt (siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75). Mögliche Formate sind ‘HH:MM:SS’, ‘MM:SS’ oder ‘HH:MM’, wobei ‘HH’ die Stunden, ‘MM’ die Minuten und ‘SS’ die Sekunden darstellen.

Es ist auch möglich, Teile des Formats leer zu lassen, z.B. wird die Eingabe ‘6:30’ unter dem Format ‘HH:MM:SS’ automatisch auf ‘00:06:00’ ergänzt. Wird nur eine Zahl eingegeben, so wird dies bei den Formaten ‘HH:MM:SS’ und ‘MM:SS’ als die Anzahl Sekunden, bei ‘HH:MM’ als die Anzahl Minuten gewertet und die entsprechende Zeit errechnet.

9.3.7 Kontextmenü des mehrzeiligen Textfelds

Mehrzeilige Textfelder besitzen ein Kontextmenü, das weitere Editiermöglichkeiten anbietet:

- ‘Ausschneiden’, ‘Kopieren’, and ‘Einfügen’ erlauben den Datenaustausch mit der Zwischenablage.
- ‘Löschen’ entfernt den markiertext Text und ‘Alles markieren’ wählt den gesamten Text aus (Windows, Mac OS und Linux).
- ‘Löschen’ löscht den gesamten Inhalt des Textfelds (Amiga).

- ‘Rückgängig’ und ‘Wiederherstellen’ erlauben das Vor- und Zurückspringen der Änderungen, die am Textfeldinhalt gemacht wurden (nur Amiga).
- ‘Eingabemethoden’ und ‘Unicode-Steuerzeichen einfügen’ sind GTK-spezifische Menüpunkte (Windows, Mac OS und Linux). Diese sind in der GTK-Dokumentation beschrieben.
- Mit ‘Text laden’ und ‘Text speichern’ kann der Inhalt des mehrzeiligen Textfelds von einer Datei geladen bzw. gespeichert werden.
- ‘Externer Editor’ startet einen externen Editor zum Bearbeiten des mehrzeiligen Textfeldes. Siehe Abschnitt 7.1.2 [Externer Editor], Seite 35, für weitere Informationen zum externen Editor.

9.3.8 Kontextmenü des Select-from-where-Listenfelds

Virtuelle Felder des Typs ‘Liste’ sind mit einem Kontextmenü ausgestattet, das die folgende Einträge enthält:

- ‘Export als Text’ für das Exportieren der Liste in eine Textdatei (siehe Abschnitt 14.3 [Abfragen als Text exportieren], Seite 61).
- ‘Export als PDF’ (auf den meisten Systemen) für das Exportieren der Liste in eine PDF-Datei (siehe Abschnitt 14.4 [Abfragen als PDF exportieren], Seite 62).
- ‘Drucken’ für das Drucken der Liste (siehe Abschnitt 14.5 [Abfragen ausdrucken], Seite 62).

9.3.9 Eingabe von Beziehungswerten

Für Beziehungsfelder kann die Datensatzreferenz über eine Popup-Liste ausgewählt werden:

- Rechts vom Beziehungsfeld befindet sich ein Popup-Knopf, der auf Knopfdruck eine Liste von Datensätzen 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.
- Durch Tastatureingabe kann nach einem entsprechenden Eintrag in der referenzierten Tabelle gesucht werden. Nach dem ersten Tastendruck öffnet sich ein Eingabefeld, um weitere Zeichen für die Suche aufzunehmen. Nach jedem Tastendruck wird sofort eine Suche gestartet (ohne Beachtung von Groß-/Kleinschreibung) und der erste passende Eintrag wird ausgewählt. Die Suchmethode kann im Anzeigeobjekt des Feldes (siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75) unter dem Bereich ‘Schnellsuche’ spezifiziert werden. Es können die Zeichen ‘*’ zum Ersetzen beliebig vieler Zeichen und ‘?’ zum Ersetzen genau eines beliebigen Zeichens verwendet werden. Mittels den Cursorstasten *Down* und *Up* kann man zum nächsten, bzw. vorherigen passenden Eintrag springen. Der gefundene Eintrag wird durch Bestätigen mit *Enter* übernommen. Wird das Suchfeld auf andere Weise verlassen, z.B. durch Drücken von *Esc*, so bleibt der bisherige Wert erhalten.

Ein Beziehungsfeld kann auch gesetzt werden, in dem eine Zeile von einer Liste eines virtuellen Attributs auf das Beziehungsfeld gezogen wird (Drag & Drop). Stammt die Zeile von einem Datensatz der referenzierten Tabelle, so wird dieser Datensatz als neuer Beziehungsdatensatz verwendet.

9.3.10 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 [Tabelle der Feldtypen], Seite 24, für einen Überblick über alle Feldtypen.

9.4 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 7.2.2 [Datensätze löschen bestätigen], Seite 38).

Wurde eine Auslösefunktion zum Löschen von Datensätzen eingerichtet (siehe Abschnitt 15.1.1 [Tabellen erstellen], Seite 67), dann wird diese Auslösefunktion zum Löschen des Datensatzes ausgeführt. Für mehr Informationen zu diesem Mechanismus, siehe Abschnitt 16.29.9 [Auslösefunktion Löschen], Seite 161.

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.

9.5 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 gehen, 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 29). Das Panel kann auch zwei Pfeilknöpfe enthalten, um zum vorhergehenden und nächsten Datensatz zu springen.

Das Durchforsten der Datensätze lässt sich einfach mit den Cursorstasten *Up* und *Down* in Verbindung mit den Tasten *Shift*, *Alt*, *Command* und *Ctrl* durchführen. Alle möglichen Kombinationen sind im Menüpunkt ‘**Tabelle - Gehe zum Datensatz**’ aufgeführt. Auf Windows, Linux und Amiga sind die Tastenkombinationen:

	<i>Alt</i>	<i>Shift-Ctrl</i>	<i>Shift-Alt</i>
<i>Up</i>	Voriger Datensatz	Erster Datensatz	Springe zurück
<i>Down</i>	Nächster Datensatz	Letzter Datensatz	Springe vorwärts

Unter Mac OS sind die Tastenkombinationen:

	<i>Command</i>	<i>Ctrl-Command</i>	<i>Shift-Command</i>
<i>Up</i>	Voriger Datensatz	Erster Datensatz	Springe zurück
<i>Down</i>	Nächster Datensatz	Letzter Datensatz	Springe vorwärts

9.6 Alle Datensätze ansehen

Es ist möglich alle Datensätze einer Tabelle anzusehen, indem Menüpunkt 'Tabelle - Alle Datensätze ansehen' selektiert wird. Dies öffnet den Abfrageeditor mit einer voreingestellten Abfrage, welche alle Datensätze der aktuellen Tabelle auflistet. Man kann die Abfrage modifizieren und den Titel ändern, und MUIbase merkt sich die Abfrage. Für weitere Informationen, siehe Abschnitt 14.2 [Abfrageeditor], Seite 60.

10 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.

10.1 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.

10.1.1 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 er in die Tabellenmaske übernommen.

Jede Tabelle kann seinen eigenen Filterausdruck besitzen.

10.1.2 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 plaziert 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 Funktions-/Operatormenamen werden hier eingefügt. Man kann hier auch direkt einen Filterausdruck eingeben.
- zwei Knöpfe ‘Ok’ und ‘Abbrechen’, um das Fenster zu verlassen.

Nach der Festlegung des Filterausdruckes klickt man auf ‘Ok’, um das Fenster zu verlassen. Der eingegebene Ausdruck wird kompiliert und bei fehlerfreier Kompilation wird der Ausdruck für alle Datensätze ausgewertet. Die Datensätze, für die der Boolesche Ausdruck NIL ergibt, werden aus der Tabellenmaske ausgeblendet.

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, sofern er installiert wurde, ein- und ausgeschaltet werden. Nachdem ein Filterausdruck für eine Tabelle festgelegt wurde, wird der 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.

10.1.3 Filterbeispiele

Hier ein paar Beispiele für gültige Filterausdrücke:

- ‘NIL’ filtert alle Datensätze heraus (es wird also kein Datensatz mehr angezeigt).
- ‘TRUE’ filtert keinen Datensatz heraus (es werden alle Datensätze angezeigt).
- ‘0’ entspricht ‘TRUE’, ebenso wie alle Ausdrücke die einen Wert ungleich NIL liefern.
- ‘(> 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. Mehr dazu im Kapitel Kapitel 16 [MUIbase programmieren], Seite 83.

10.2 Referenzfilter

Beziehungsfelder besitzen auch einen Filter. Dies ist nützlich, wenn Tabellen hierarchisch aufgebaut werden sollen. Das Projekt ‘Albums’ ist ein Beispiel dafür.

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

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

Hinweis: Ein verschachteltes Löschen 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, da diese wenig Sinn machen.

11 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.

11.1 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.

11.2 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.

Für die Festlegung der Reihenfolge werden die folgenden Regeln 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 unabhängig von Groß-/Kleinschreibung durchgeführt)
Datum	NIL < 1.1.0000 < ... < 31.12.9999
Zeit	NIL < 00:00:00 < ... < 596523:14:07
Boolesch	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.

11.3 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 16.29.10 [Vergleichsfunktion], Seite 161, im Abschnitt über die Programmierung von MUIbase für mehr Details).

MUIbase bietet 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 16.29.10 [Vergleichsfunktion], Seite 161.

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.

11.4 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 plazierte. 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, so ist sie 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 keine Sortierung durchgeführt. Für mehr Informationen über die Anwendung der Vergleichsfunktion, einschließlich der Argumente, die ihr übergeben werden, siehe Abschnitt 16.29.10 [Vergleichsfunktion], Seite 161.

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. 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.

11.5 Neu sortieren aller Datensätze

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

12 Suchen

Zum Durchforsten von Datensätzen kann ein Suchfenster verwendet werden, um nach einem bestimmten Datensatz zu suchen. Nach Eingabe eines Suchmusters werden alle Datensätze durchsucht. Wird das Muster in einem Datensatz gefunden, so wird dieser in der Tabellenmaske angezeigt.

12.1 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 eine beliebige Anzahl von Zeichen (einschließlich keine Zeichen), wohingegen **‘?’** genau ein Zeichen ersetzt.
- ein Feld **‘GROSS/klein beachten?’**. Wenn markiert, dann verwendet die Suche einen Zeichenkettenvergleich der Groß-/Kleinschreibung beachtet, anderenfalls wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- 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 Sortierung der Datensätze ausnützt. Dies kann die Geschwindigkeit enorm steigern.

12.2 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.

12.3 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’.

13 Import und Export

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

13.1 Dateiformat

Zum Importieren von Datensätzen in MUIbase müssen alle Datensätze in einer einzelnen Textdatei vorliegen. Sollen Datensätze mehrerer Tabellen importiert werden, 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 Exportdialogen festgelegt werden. Da Datensatzfelder selbst auch diese Trennzeichen enthalten können, ist es möglich, diese mit doppelten Anführungszeichen 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. Taucht ein Name auf, der nicht in der Tabelle vorkommt, so wird eine Fehlermeldung angezeigt.
- 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 Boolesch muss das Feld entweder NIL oder TRUE (unabhängig von Groß-/Kleinschreibung) enthalten, anderenfalls wird eine Fehlermeldung ausgegeben. Für Felder vom Typ Auswahl müssen die genauen Auswahltexte angegeben werden (Groß-/Kleinschreibung wird beachtet). 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 umgeben werden.

13.2 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 (Boolesch)
- 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" ""
```

13.3 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 zweite Knopf startet den externen Anzeiger mit der angegebenen Datei und der dritte Knopf startet den externen Editor, um den Dateinhalt 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`, `\???` (Oktalzahl) oder `\x??` (Hexadezimalzahl) eingeben. Trennzeichen müssen 7-Bit-ASCII-Zeichen sein.
- Ein Feld ‘In Anführungszeichen’, das eingeschaltet werden kann, um anzugeben, dass die Felder mit doppelten Anführungszeichen umgeben sind.
- Ein Feld ‘Überschreibe Datensätze’, welches, wenn aktiviert, bestehende Datensätze mit den importierten Daten überschreibt. Dies kann nützlich sein, wenn bestehende Datensätze erhalten bleiben sollen (z.B. weil es Referenzen auf diese gibt) diese aber mit neuen Daten versorgt werden sollen.
- Zwei Knöpfe ‘Importieren’ und ‘Abbrechen’, um das Fenster zu verlassen.

Wird der Knopf ‘Importieren’ gedrückt, dann beginnt MUIbase die angegebene Datei einzuladen und alle gefundenen Datensätze zu importieren. Falls keine Fehler auftraten und neue Datensätze angelegt wurden, so fragt MUIbase nach, ob die neuen importierten Datensätze wirklich zur Tabelle hinzugefügt werden sollen. An dieser Stelle lässt sich der Vorgang noch abbrechen. Überschriebene Datensätze können jedoch nur durch Wiederherstellen des Projekts zurückgestellt werden.

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.

13.4 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 zur 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`, `\????` (Oktalzahl) oder `\x???` (Hexadezimalzahl) eingeben.
- Ein Feld `'In Anführungszeichen'`, das eingeschaltet werden kann, um anzugeben, dass die Felder mit doppelten Anführungszeichen umgeben 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 zu verlassen.

Nach einem Druck auf den Knopf `'Exportieren'` öffnet MUIbase die angegebene Datei und beschreibt sie mit den Datensätzen einschließlich der Kopfzeile mit den Feldnamen. 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 14 [Datenabfragen], Seite 60) verwenden oder eigene Exportfunktionen als MUIbase-Programm schreiben.

14 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 15 [Struktureditor], Seite 67) und im Kapitel über die Programmierung von MUIbase (siehe Kapitel 16 [MUIbase programmieren], Seite 83) beschrieben.

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

14.1 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 bestimmten Kriterien genügen, werden in die Ausgabe einbezogen. Die (unvollständige) Syntax einer Select-from-where Abfrage ist

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

wobei *explist* 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.

Für weitere Informationen zur Select-from-where Abfrage einschließlich ihrer vollständigen Syntax siehe Kapitel 16 [MUIbase programmieren], Seite 83, und für weitere Beispiele siehe Abschnitt 14.6 [Abfragebeispiele], Seite 64.

14.2 Abfrageeditor

Zum Eingeben und Ausführen von Abfragen öffnet man den Abfrageeditor über den Menüpunkt 'Programm - Abfragen'. Der Abfrageeditor kann mehrere Abfragen verwalten, 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 Auswahlfeld, welches es erlaubt die Abfrage einer Tabelle zuzuordnen. Wird eine Tabelle zugeordnet, so führt MUIbase diese Abfrage aus, wenn der Benutzer den Menüpunkt ‘Tabelle - Alle Datensätze anzeigen’ auswählt.
- ein Knopf ‘Ausführen’, der das Abfrageprogramm kompiliert, ausführt und das Ergebnis in der Ausgabe-Listenansicht ausgibt.
- ein Knopf ‘Export’, der ein Fenster (siehe Abschnitt 14.3 [Abfragen als Text exportieren], Seite 61) öffnet, um das Ergebnis der Abfrage in eine Textdatei zu exportieren.
- ein Knopf ‘PDF’ (auf den meisten Systemen), der ein Fenster (siehe Abschnitt 14.4 [Abfragen als PDF exportieren], Seite 62) öffnet, um das Ergebnis der Abfrage in eine PDF-Datei zu exportieren.
- ein Knopf ‘Drucken’, der ein Fenster (siehe Abschnitt 14.5 [Abfragen ausdrucken], Seite 62) öffnet, um Ergebnisse auszudrucken.
- ein Editorfeld zum Eingeben des Abfrageprogramms. Hier wird normalerweise eine Select-from-where Abfrage eingegeben. Es ist jedoch auch möglich, einen beliebigen 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. Ein Klick auf einen Spaltentitel bewirkt, dass die Liste nach dieser Spalte sortiert dargestellt wird. Wird auf denselben Titel erneut geklickt, dann wird die Sortierung umgedreht. Auf dem Amiga kann durch Klicken auf einen Spaltentitel bei gleichzeitigem Halten der *Shift*-Taste eine zweite Sortierspalte gesetzt werden. Wird auf einen Eintrag in der Liste doppelgeklickt und dieser Eintrag wurde aus einem Datensatz generiert, dann wird dieser in der zugehörigen Tabellenansicht angezeigt. Dies ist eine einfache Möglichkeit, zu einem bestimmten Datensatz in der 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.

14.3 Abfragen als Text exportieren

Das Ergebnis einer Select-From-Where-Abfrage kann durch drücken des ‘Export’-Knopfes in eine Textdatei ausgegeben werden. Dies öffnet ein Fenster mit folgender Struktur:

- Ein Zeichenkettenfeld zum Eingeben des Exportdateinamens. Rechts neben dem Feld gibt es einen Knopf, der zur 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`, `\???` (Oktalzahl) oder `\x??` (Hexadezimalzahl) eingeben.
- Ein Feld **‘In Anführungszeichen’**, das eingeschaltet werden kann, um anzugeben, dass die Felder mit doppelten Anführungszeichen umgeben werden sollen.
- Zwei Knöpfe **‘Exportieren’** und **‘Abbrechen’**, um das Fenster zu verlassen.

Nach einem Druck auf den Knopf **‘Exportieren’** öffnet MUIbase die angegebene Datei und beschreibt sie mit dem Abfrageergebnis einschließlich einer Kopfzeile mit dem Listenkopf. Die Felder werden in der Reihenfolge der Listenspalten ausgegeben.

14.4 Abfragen als PDF exportieren

Auf Windows, Mac OS X, Linux und MorphOS kann das Abfrageergebnis durch drücken des Knopfs **‘PDF’** in eine PDF-Datei exportiert werden.

Unter Windows, Mac OS X und Linux öffnet dies den Standard-Druckdialog, in welchem der Eintrag **‘In Datei drucken’** mit dem Format **‘PDF’** vorausgewählt ist. Der Dialog enthält eine zusätzliche Seite **‘Zeichensatz’**, in welchem ein Zeichensatz gewählt werden kann. Weiterhin kann hier das Verkleinern der Ausgabe ermöglicht werden, so dass das Ergebnis in die Seitenbreite des gewählten Papiers und der gewählten Orientierung passt. Nach drücken des Knopfs **‘Drucken’** wird das Abfrageergebnis in die angegebene Datei ausgegeben.

Auf MorphOS wird ein Fenster geöffnet, das die folgende Elemente enthält:

- Ein Zeichenkettenfeld zum Eingeben des Exportdateinamens. Rechts neben dem Feld gibt es einen Knopf, der zur Auswahl eines Dateinamens aus einem Dateiauswahlfenster dient.
- Ein Auswahlfeld zur Angabe der Papiergröße.
- Ein Auswahlfeld zur Angabe der Orientierung (**‘Portrait’** oder **‘Landschaft’**).
- Ein Auswahlfeld zur Auswahl des Zeichensatzes mit einem Zeichenkettenfeld für die Eingabe der Zeichensatzgröße. Es ist möglich Bruchteile einzugeben, z.B. **‘10.5’**.
- Ein Statusfeld, welches die Anzahl Seiten anzeigt und angibt, ob der Inhalt in die Seitenbreite passt.
- Zwei Knöpfe **‘Erzeuge PDF’** und **‘Abbruch’** zum Verlassen des Fensters.

Nach drücken des Knopfs **‘Erzeuge PDF’** öffnet MUIbase die angegebene Datei und beschreibt sie mit dem Abfrageergebnis einschließlich einer Kopfzeile mit dem Listenkopf. Die Felder werden in der Reihenfolge der Listenspalten ausgegeben.

14.5 Abfragen ausdrucken

Nachdem eine Abfrage durchgeführt wurde, kann das Ergebnis über den Knopf **‘Drucken’** ausgedruckt werden.

Auf Windows, Mac OS X und Linux öffnet dies den Standard-Druckdialog. Der Dialog enthält eine zusätzliche Seite **‘Zeichensatz’**, in welchem ein Zeichensatz gewählt werden

kann. Weiterhin kann hier das Verkleinern der Ausgabe ermöglicht werden, so dass das Ergebnis in die Seitenbreite des gewählten Papiers und der gewählten Orientierung passt. Nach drücken des Knopfs 'Drucken' wird das Abfrageergebnis auf dem gewählten Drucker ausgegeben.

Auf MorphOS öffnet der Knopf 'Drucken' ein ähnliches Fenster wie beim Exportieren als PDF (siehe Abschnitt 14.4 [Abfragen als PDF exportieren], Seite 62). Das Fenster enthält die folgenden Elemente:

- Ein Auswahlfeld zur Angabe der Papiergröße.
- Ein Auswahlfeld zur Angabe der Orientierung ('Portrait' oder 'Landschaft').
- Ein Auswahlfeld zur Auswahl des Zeichensatzes mit einem Zeichenkettenfeld für die Eingabe der Zeichensatzgröße. Es ist möglich Bruchteile einzugeben, z.B. '10.5'.
- Ein Statusfeld, welches die Anzahl Seiten anzeigt und angibt, ob der Inhalt in die Seitenbreite passt.
- Zwei Knöpfe 'Erzeuge PDF' und 'Abbruch' zum Verlassen des Fensters.

Nach drücken des Knopfs 'Erzeuge PDF' generiert MUIbase eine temporäre PDF-Datei und öffnet diese mit dem externen Anzeiger (siehe Abschnitt 7.1.3 [Externer Anzeiger], Seite 36). Man kann dann die Druckmöglichkeiten des externen Anzeigers verwenden, um das Abfrageergebnis auszudrucken.

Auf anderen Amiga-Systemen wird ein Druckfenster mit den folgenden Elementen geöffnet:

- 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). 'Custom' erlaubt die Eingabe einer beliebigen Zeichenkette, die zwischen den Feldern ausgegeben werden soll.
- 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 'Entwurf' erzeugt.
- ein Bereich 'Größe', in dem die Zeichengröße definiert wird. 'Pica' druckt mit großer Schrift (10 cpi), 'Elite' in mittelgroßer 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 eine Ausgabe auf dem Drucker sollte `l1pr` (Linux), bzw. `PRT:` (Amiga) im Zeichenkettenfeld eingegeben werden. Für weitere spezielle Dateinamen siehe Abschnitt 7.2.12 [Programm-Ausgabedatei], Seite 40.
- zwei Knöpfe ‘Ok’ und ‘Abbrechen’, um das Druckfenster zu verlassen.

Sind alle Einstellungen getan, so drückt man auf den Knopf ‘Ok’, um den Druckauftrag zu starten.

14.6 Abfragebeispiele

Um einen Eindruck von der Leistungsfähigkeit der Select-from-where Abfragen zu bekommen, folgen einige Beispiele¹.

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

¹ Im folgenden werden die originalen Beispiele verwendet, welche englische Bezeichnungen enthalten. Daher wird bei den Beschreibungen auch die deutsche Bezeichnung in Klammern dahintergesetzt, sofern diese vom Englischen abweicht.

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.)

```
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

15 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.

15.1 Tabellenverwaltung

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

15.1.1 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 Großbuchstaben 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 Datensätzen halten kann und **‘genau ein’** kann nur einen einzigen Datensatz halten. Letzteres ist manchmal nützlich, um das Projekt zu steuern (siehe Abschnitt 5.2 [Tabellen], 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 dieser Auslösefunktionen, einschließlich der übergebenen Argumente, siehe Abschnitt 16.29.8 [Auslösefunktion Neu], Seite 160, und Abschnitt 16.29.9 [Auslösefunktion Löschen], Seite 161.

- ein Feld ‘Zähle Änderungen’, welches, wenn gesetzt, das Anlegen und Löschen eines Datensatzes als eine Änderung des Projekts zählt. Wenn ungesetzt, so werden jegliche Änderungen in der Tabelle (oder in einem Feld der Tabelle) ignoriert.
- ein Feld ‘Protokolliere Änderungen’. Falls gesetzt, so wird jedes Anlegen und Löschen eines Datensatzes im Log des Projekts festgehalten. Andernfalls wird niemals eine Änderung in der Tabelle (oder in einem Feld der Tabelle) im Log notiert.
- 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 vorliegt, z.B. Eingabe eines falschen Namens, dann weist ein Hinweisfenster auf den Fehler hin. Tritt kein Fehler auf, dann schließt das Fenster und die neue Tabelle erscheint in der Tabellenliste des Struktureditors.

15.1.2 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 15.1.1 [Tabellen erstellen], Seite 67) 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.

15.1.3 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 Programmmeditor 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 Programmmeditor schließen.

15.1.4 Tabellen sortieren

Zum Sortieren der Tabellen im Bereich ‘Tabellen’ des Struktureditors hat man mehrere Möglichkeiten: mit Verschieben & Loslassen die Reihenfolge per Hand einstellen oder den Knopf ‘Sortieren’ unterhalb der Listenansicht drücken, der die Tabellen alphabetisch (‘Sortieren1’) oder anhand der Reihenfolge in der Anzeigenliste (‘Sortieren2’) sortiert.

15.2 Felderverwaltung

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

15.2.1 Felder erstellen

Um ein neues Feld für die aktive Tabelle zu erstellen, drückt man den Knopf **‘Neu’** im Bereich **‘Felder’**. Dies öffnet 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 [Feldtypen], Seite 21.
- einem Bereich unterhalb des Feldes **‘Typ’** zum Festlegen von typabhängigen Einstellungen. Mehr über diesen Bereich, siehe Abschnitt 15.2.2 [Typabhängige Einstellungen], Seite 69.
- 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 dieser Auslösefunktion, einschließlich der übergebenen Argumente, siehe Abschnitt 16.29.11 [Auslösefunktion Feld], Seite 162.
- einem Feld **‘Zähle Änderungen’**. Wenn ausgewählt, so wird jede Änderung des Feldes in einem Datensatz als eine Änderung des Projekts gezählt. Man deaktiviert dieses Feld, wenn Änderungen des Feldes ignoriert werden sollen. Zu beachten ist, dass dieses Feld nur dann anwählbar ist, wenn das Feld **‘Zähle Änderungen’** in der Tabelle ebenfalls gesetzt ist (siehe Abschnitt 15.1.1 [Tabellen erstellen], Seite 67).
- einem Feld **‘Protokolliere Änderungen’**, das, wenn gesetzt, jede Änderung des Feldes in einem Datensatz im Log des Projekts festhält. Zu beachten ist, dass dieses Feld nur dann anwählbar ist, wenn das Feld **‘Protokolliere Änderungen’** in der Tabelle ebenfalls gesetzt ist (siehe Abschnitt 15.1.1 [Tabellen erstellen], Seite 67).
- 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 vorliegt, z.B. Eingabe eines falschen Namens, dann weist ein Hinweisfenster auf den Fehler hin. Tritt kein Fehler auf, dann schließt das Fenster und das neue Feld erscheint in der Felderliste des Struktureditors.

15.2.2 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 davon 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 15.2.3 [Auswahltexteditor], Seite 70).
 - ein Auswahlfeld ‘Vorgabewert’ zur Festlegung des Wertes zum Initialisieren des Feldes.
- Für Beziehungsfelder enthält der typabhängige Bereich
 - eine Listenansicht, die alle Tabellen anzeigt, zu denen eine Beziehung hergestellt werden kann. Hier klickt man auf die Tabelle, zu der die Beziehung angelegt werden soll.
 - ein Feld ‘Auto Anzeige’. Wenn ausgewählt, dann wird in der referenzierte Tabelle immer automatisch der gerade referenzierte Datensatz angezeigt.
 - ein Feld ‘Filtern’. Wenn ausgewählt, ist der Referenzfilter des Feldes aktiviert. Siehe Abschnitt 10.2 [Referenzfilter], Seite 51, 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 der Funktion, die zur Berechnung des Feldwertes ausgeführt wird. Der angehängte Popup-Knopf kann zur Auswahl eines Namens aus einer Liste von Funktionsnamen verwendet werden. Mehr über die Anwendung dieser Auslösefunktion, siehe Abschnitt 16.29.12 [Virtuelle Felder programmieren], Seite 163.
- Mehrzeilige Textfelder und Knöpfe besitzen keine typabhängigen Einstellungen. Der Vorgabewert für mehrzeilige Texte ist eine leere Zeichenkette.

15.2.3 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 *Enter*-Taste gedrückt wird. Gibt es keinen aktiven Auswahltext, dann fügt *Enter* 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 zu sortieren.
- zwei Knöpfen ‘Ok’ und ‘Abbrechen’, um den Auswahltexteditor zu verlassen.

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

15.2.4 Felder kopieren

Falls viele gleichartige Felder benötigt werden sollen, dann kann ein Feld kopiert werden. Hierzu wird das gewünschte Feld ausgewählt und der Knopf ‘Kopieren’ unterhalb der Feldliste gedrückt. 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.

15.2.5 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 15.2.1 [Felder erstellen], Seite 69) 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 soll, ist es am besten, ein neues Feld mit dem gewünschten Typ zu erzeugen und die Datensatzinhalte des alten Feldes mit einem einfachen MUIbase-Programm im Abfrageeditor (siehe Abschnitt 14.2 [Abfrageeditor], Seite 60) 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. Eine flexiblere Variante 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 15.3.3 [Feldobjekteditor], Seite 75).
- Die referenzierte Tabelle eines Beziehungsfeldes kann nicht verändert werden.

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

15.2.6 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.

15.2.7 Felder sortieren

Zum Sortieren der Felder im Bereich ‘Felder’ des Struktureditors gibt es mehrere Möglichkeiten. Zum einen lässt sich dies per Verschieben & Loslassen von einzelnen Feldern oder durch den Knopf ‘Sortieren’ unterhalb der Listenansicht zum alphabetischen Sortieren (‘Sortieren1’) oder zum Sortieren anhand der Reihenfolge in der Anzeigenliste (‘Sortieren2’) erledigen.

15.3 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.

15.3.1 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 (Doppel-)klicken auf das Pfeilsymbol geöffnet und geschlossen werden. Ein Doppelklick auf das Element selbst öffnet ein Fenster zum Verändern dessen Einstellungen. Alle Benutzeroberflächenelemente, die die gleiche übergeordnete Objekt besitzen, werden in der gleichen Weise ausgerichtet (entweder horizontal oder vertikal). Das übergeordnete Benutzeroberflächenobjekt bestimmt, wie das Layout dargestellt wird: Tabellen und Fenster ordnen ihre Elemente vertikal an, Gruppen ordnen sie nach den Einstellungen im Gruppeneditor an (siehe Abschnitt 15.3.8 [Gruppeneditor], Seite 81).
- 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 Benutzeroberfläche nicht sichtbar ist. Auf diese Weise kann man eine Tabelle verstecken. 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 ‘Hoch’ und ‘Runter’ zum Verschieben des aktiven Elements in der Anzeigelistenansicht nach oben bzw. unten.
- zwei Knöpfe ‘Hinein’ und ‘Heraus’, um das aktive Elemente 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 15.3.5 [Texteditor], Seite 80).
- ein Knopf ‘Bild’ zum Hinzufügen eines Bildobjekts (siehe Abschnitt 15.3.6 [Bildeditor], Seite 80).
- ein Knopf ‘Zwischenraum’ zum Einsetzen von Zwischenräumen zwischen anderen Objekten (siehe Abschnitt 15.3.7 [Zwischenraumeditor], Seite 80).
- 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 15.3.8 [Gruppeneditor], Seite 81, für mehr Informationen zum Einrichten eines Gruppenobjekts.
- ein Knopf ‘Karteikarten’ zum Hinzufügen einer neuen Karteikarten-Gruppe in die Anzeigelistenansicht. Wie bei Gruppenobjekten können mehrere Benutzeroberflächenelemente ausgewählt werden, die in die neue Karteikarten-Gruppe übernommen werden sollen. Für mehr Informationen, siehe Abschnitt 15.3.9 [Karteikarteneditor], Seite 81.
- ein Knopf ‘Fenster’ zum Hinzufügen eines neuen Fensters in die Anzeigelistenansicht. Auch hier können mehrere Benutzeroberflächenelemente ausgewählt werden, die in das neue Fenster übernommen werden sollen. Mehr zum Einrichten eines Fensters siehe Abschnitt 15.3.10 [Fenstereditor], Seite 81.

Mehr Informationen über die Benutzeroberflächenelemente einschließlich ihrer Anwendung siehe Abschnitt 5.9 [Benutzerschnittstelle], Seite 28.

15.3.2 Tabellenobjekteditor

Beim erstellen einer neuen Tabelle wird ein voreingestelltes Anzeigeobjekt erzeugt. Um die Einstellungen dieses Tabellenobjekts zu ändern, klickt man doppelt auf die Tabelle in der ‘Anzeige’-Liste und das Fenster ‘Tabellenanzeige’ erscheint. Dieses Fenster hat drei Bereiche, die in die Karteikarten ‘Allgemeines’, ‘Panel’ und ‘Datensatz’ eingeteilt sind.

Der Bereich ‘Allgemeines’ enthält die folgenden Elemente:

- ein numerisches Feld ‘Gewichtung’, um das Gewicht des Objekts festzulegen. Der Wert dieses Feldes gibt an, wieviel Platz bezogen auf andere Objekte die Tabelle im endgültigen Layout des Fensters erhält.
- ein Feld ‘Hintergrund’ mit einem Checkmark-Feld ‘Vorgabe’ zum Festlegen, wie der Hintergrund der Tabelle aussehen soll. Wird das Feld ‘Vorgabe’ aktiviert, dann wird ein Vorgabehintergrund ausgewählt. Anderenfalls kann auf den Knopf ‘Hintergrund’ geklickt werden, um eine alternative Hintergrundeinstellung vorzunehmen.
- ein Checkmark-Feld ‘Mit Panel’, das angibt, ob ein Panel im oberen Bereich der Tabelle angezeigt werden soll. Deselektiert man dieses Feld, dann wird kein Panel angezeigt. Für mehr Informationen über Panels, siehe Abschnitt 5.9.3 [Panels], Seite 29.

Der Bereich ‘Panel’ ist nur dann anwählbar, wenn das ‘Mit Panel’-Feld gesetzt wurde. Er enthält die folgenden 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 des Panelkopfes. Wird das Feld ‘Vorgabe’ aktiviert, dann wird ein Vorgabehintergrund ausgewählt. Anderenfalls kann auf den Knopf ‘Hintergrund’ geklickt werden, um eine alternative Hintergrundeinstellung vorzunehmen.
- 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 ‘Filter’, 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 deaktiviert, was bedeutet, dass auch kein Filterausdruck für die Tabelle angegeben werden kann. Mehr über Datensatzfilter, siehe Abschnitt 10.1 [Datensatzfilter], Seite 50.
- 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 deaktiviert.

Der Bereich ‘Datensatz’ dient zur Festlegung einer Datensatzbeschreibung. Die Datensatzbeschreibung wird beim Erstellen von Logeinträgen und in Popup-Listenansichten für die Auswahl eines Datensatzes verwendet. Sie sollte eine kurze aber eindeutige Identifizierung sein. Der Bereich enthält die folgenden Elemente:

- ein Auswahlfeld ‘Beschreibung’ welches entweder auf ‘Felder’ oder ‘Berechnet’ gesetzt werden kann.
- Ist ‘Beschreibung’ auf ‘Felder’ gesetzt, dann wird eine Liste von Feldern der Tabelle angezeigt. Man kann jedes der angezeigten Felder für die Beschreibung verwenden. Wird ‘Datensatznummer’ ausgewählt, dann wird die Datensatznummer eines Datensatzes

mitangezeigt. Es lassen sich mehrere Einträge in dieser Liste auswählen und die Reihenfolge der Felder kann durch Verschieben der Elemente festgelegt werden.

- Ist ‘**Beschreibung**’ auf ‘**Berechnet**’ gesetzt, dann wird ein Feld ‘**Berechne**’ angezeigt, in welchem man eine Funktion für die Berechnung der Datensatzbeschreibung eingibt. Die Funktion kann entweder einen einfachen Ausdruck oder eine Liste von Ausdrücken zurückgeben (siehe Abschnitt 16.29.14 [Berechne Datensatzbeschriftung], Seite 164).
- Ein Feld ‘**Verwende Mehrspaltenliste**’, welches eine Präferenz festlegt, wie die Felder in einer Listenanzeige angeordnet werden sollen.

Unterhalb des Karteikartenbereichs befinden sich zwei Knöpfe ‘**Ok**’ und ‘**Abbruch**’ zum Verlassen des Fensters. Wurden alle Änderungen durchgeführt, dann wird ‘**Ok**’ gedrückt, um das Fenster zu schließen.

15.3.3 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 ‘**Feldanzeige**’ öffnet sich. Dieses Fenster enthält verschiedene Elemente, die vom Felddatentyp abhängen. 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**’ 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 **Alt**-Taste (Windows, Mac OS und Linux), bzw. der **Amiga**-Taste verwendet wird, um das Objekt zu aktivieren.
- ein Feld ‘**Home**’ (Start). Wenn aktiviert, dann wird dieses Objekt zum Startobjekt. 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 verlieren alle anderen Objekte der gleichen Tabelle diese Eigenschaft.
- ein Feld ‘**Tab-Kette**’. Wenn aktiviert, dann ist das Objekt in der Fokus-Kette, bei welcher man mit der **Tab**-Taste von Element zu Element springen kann. Ist das Feld nicht gesetzt, dann wird das Objekt übersprungen.
- 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. Zu beachten ist aber, dass wenn ein Popup-Knopf dem Objekt hinzugefügt wird, der Inhalt dann dennoch durch die Auswahl im Popup verändert werden kann.
- ein Auswahlfeld ‘**Ausrichtung**’ 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 Bereich ‘Aktiv/inaktiv’. Ist ‘Immer aktiv’ gesetzt, so ist das Feldobjekt immer anwählbar unabhängig vom angezeigten Datensatz. ‘Inaktiv im Initial-DS’ deaktiviert das Objekt im initialen Datensatz, sonst ist es aktiv. Wird ‘Berechne aktiv’ gewählt, so kann rechts davon eine Funktion zur Berechnung des Aktivzustands eingegeben werden. Die Funktion wird ohne Argumente aufgerufen. Liefert die Funktion NIL zurück, so wird das Objekt deaktiviert, sonst aktiv. Im Falle, dass die Berechnungsfunktion leer gelassen wird oder nicht gefunden werden kann, wird das Objekt deaktiviert. Für mehr Informationen über die Anwendung dieser Auslösefunktion, siehe Abschnitt 16.29.13 [Compute enabled function], Seite 164.
- 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. Für die meisten Feldtypen betrifft der Wert dieses Feldes nur die horizontale Größe des Objekts, da die meisten Objekte feste Höhen haben. Für mehrzeilige Textfelder aber z.B. betrifft es auch die vertikale Größe.
- ein Zeichenkettenfeld ‘Zeichensatz’ 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 alternative HintergrundEinstellung vorzunehmen.
- 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.

15.3.4 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 in die Benutzeroberfläche plaziert, 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 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 ‘veränderbar’ aktiv, dann kann das Objekt in der Größe

verändert werden und das Objekt kann größer werden als die Ausmaße des Bildes. Mit `'fixiert'` wird die Größe des Objekts auf die des Bildes gesetzt. Ändert sich die Größe des Bildes von Datensatz zu Datensatz, dann ändert sich entsprechend auch die Größe des Objekts. Die Auswahl von `'scrollbar'` fügt zwei Rollbalken zum Objekt hinzu, um Bilder anzuzeigen, die größer sind als der sichtbare Ausschnitt des Objekts. Ist `'skaliert'` aktiviert, dann wird das Bild auf die Größe des Objekts skaliert. Mit `'aspekt-skaliert'` wird das Bild ebenfalls skaliert, dabei bleibt aber das Seitenverhältnis des Bildes bewahrt.

- 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.
 - 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 einer Liste ausgewählt werden kann. Die Liste der Zeichenketten wird rechts neben dem `'Listenansicht-Popup'`-Feld definiert. Die Auswahltexte können entweder `'Statisch'` oder `'Berechnet'` sein. Wird `'Statisch'` ausgewählt, so kann die Liste der Zeichenketten im Auswahltexteditor eingegeben werden, der über den Knopf `'Ändere Auswahltexte'` aufgerufen wird. Mehr zum Auswahltexteditor siehe Abschnitt 15.2.3 [Auswahltexteditor], Seite 70. Wird `'Berechnet'` ausgewählt, so kann im Feld `'Berechne'` eine Auslösefunktion eingegeben werden, die immer bei Drücken des Popup-Knopfes aufgerufen wird. Diese Funktion sollte einen Memotext, bei der jede Zeile einen Listeneintrag darstellt, oder NIL für eine leere Liste zurückgeben (siehe Abschnitt 16.29.16 [Berechne Listenansichts-Auswahltexte], Seite 165). Nur eines der Felder `'Dateiauswahl'`, `'Zeichensatzauswahl'` und `'Listenansicht-Popup'` kann aktiviert werden.
 - 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 Dateien 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 7.1.3 [Externer Anzeiger], Seite 36).
- 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 und die Festlegung auf eine horizontale Ausrichtung.
 - für Felder vom Typ Fließkommazahl gibt es ein Ganzzahlfeld `'Nachkommastellen'`, in dem die Anzahl der Nachkommastellen zum Darstellen der Fließkommazahlen eingegeben werden kann.
 - für Felder vom Typ Zeit gibt es ein Auswahlfeld `'Format'` für die Angabe, wie Zeitwerte

angezeigt und eingegen werden sollen. Man kann zwischen ‘HH:MM:SS’, ‘MM:SS’ und ‘HH:MM’ wählen. Bei ‘HH:MM’ wird die Anzeige der Sekunden unterdrückt und reine Zahlenwerte bei der Eingabe werden als die Anzahl Minuten gewertet.

- für Beziehungsfelder gibt es eine Seite ‘Extras’, die folgende Punkte enthält:
 - ein Bereich ‘Anzeige’ für das Festlegen einer Datensatzbeschriftung, welche den anzuzeigenden Inhalt eines referenzierten Datensatzes angibt. Siehe Abschnitt 15.3.2 [Tabellenobjekteditor], Seite 73, für Informationen wie man eine Datensatzbeschriftung festlegt. Zu beachten ist, dass das Auswahlfeld ‘Beschriftung’ auch eine Auswahl ‘Von Tabelle’ zulässt. Wenn ausgewählt, dann werden die Einstellungen der Tabelle für die Anzeige eines referenzierten Datensatzes verwendet.
 - ein Bereich ‘Popup’, in welchem festgelegt wird, welche Datensätze der Bezugstabelle im Popup zur Auswahl gestellt und wie diese angezeigt werden sollen. Wird ‘Datensätze’ auf ‘Alle’ gesetzt, so stehen alle Datensätze zur Verfügung. Ist ‘Datensätze’ auf ‘Filter’ gesetzt, so stehen nur die Datensätze zur Auswahl, die dem aktuell eingestellten Filter in der Bezugstabelle genügen. Wird ‘Datensätze’ auf ‘Berechnet’ gesetzt, so kann eine Funktion zur Berechnung der Menge der Datensätze im Feld ‘Berechne’ eingegeben werden. Diese Auslösefunktion muss eine Liste zurückgeben, welche nach dem Vorkommen von Datensätzen der Bezugstabelle untersucht wird. Jeder solche gefundene Datensatz wird in der Popup-Liste angezeigt. Andere Listenelemente werden ignoriert. Siehe Abschnitt 16.29.17 [Berechne Referenz-Datensätze], Seite 165, für weitere Informationen über diese Funktion. Zusätzlich zu den in ‘Datensätze’ spezifizierten Einträgen können der initiale Datensatz und der aktuell angezeigte Datensatz der Bezugstabelle mit in die Popup-Liste aufgenommen werden, Dies geschieht durch Setzen der Felder ‘Initial-Datensatz’ bzw. ‘Aktueller Datensatz’.
 - 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. Hierfür wird ggf. das Fenster, in dem die Tabelle eingesetzt ist, geöffnet und nach vorne geholt.
 - ein Feld ‘Auto Anzeige’, das - wenn ausgewählt - einen Knopf rechts neben das Referenzfeld plaziert, um die automatische Anzeige von referenzierten Datensätzen ein- und auszuschalten. Ist sie eingeschaltet, dann wird in der referenzierte Tabelle immer automatisch der gerade referenzierte Datensatz angezeigt.
 - ein Feld ‘Filter’, das - wenn aktiviert - einen Knopf rechts neben dem Beziehungsfeld ergänzt, mit dem der Referenzfilter für dieses Feld ein- und ausgeschaltet werden kann. Mehr zu Referenzfilter siehe Abschnitt 10.2 [Referenzfilter], Seite 51.
- für virtuelle Felder enthält der Feldobjekteditor eine ‘Extras’-Seite mit den folgenden Punkten:
 - ein Auswahlfeld ‘Art’, in dem festgelegt wird, wie der Inhalt des virtuellen Feldes dargestellt werden soll. Es lässt sich zwischen ‘Boolesch’, 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: 'Ausrichtung', 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, so werden weitere Felder 'Zeige Titel', 'Tab-Kette' und 'Bei Doppelklick' verfügbar. Ist 'Zeige Titel' aktiviert, dann wird die erste Zeile des Feldes als Titelzeile in der Listenansicht angezeigt. Anderenfalls wird keine Titelzeile angezeigt und die erste Zeile ignoriert. Wird 'Tab-Kette' aktiviert, dann ist das Objekt in der Fokus-Kette, bei welcher man mit der **Tab**-Taste von Element zu Element springen kann. Ist das Feld nicht gesetzt, dann wird das Objekt übersprungen. In 'Bei Doppelklick' kann angegeben werden, welche Aktion bei Doppelklick auf ein Listenelement ausgeführt werden soll. 'Tue nichts' ignoriert den Doppelklick. 'Zeige Datensatz' stellt den Datensatz, zu welchem das angewählten Feld gehört, in der zugehörigen Tabellenansicht dar. Bei Auswahl von 'Auslösefunktion' kann der Name einer Auslösefunktion weiter rechts eingegeben werden, die bei jedem Doppelklick ausgeführt werden soll. Mehr über die Anwendung dieser Auslösefunktion, einschließlich der übergebenen Argumente, siehe Abschnitt 16.29.15 [Auslösefunktion Doppelklick], Seite 164.
 - ein Feld 'Auto-Aktualisierung'. Ist es gesetzt, so wird das virtuelle Feld automatisch neu berechnet, sobald sich ein abhängiger Zustand ändert. Dies beinhaltet Wechseln zu einem anderen Datensatz in einer abhängigen Tabelle, Ändern eines abhängigen Datensatzfeldes, Anlegen oder Löschen eines Datensatzes in einer abhängigen Tabelle, Wechseln des Admin-/Benutzermodus, oder das Neuübersetzen des Projekt-Programms. Die Abhängigkeiten des virtuellen Feldes werden automatisch aus der Berechnungsfunktion des virtuellen Feldes abgeleitet. Mittels Menüpunkt 'Projekt - Struktur exportieren' können alle abgeleiteten Abhängigkeiten ausgegeben werden (hierzu muss ggf. das Programm neu übersetzt werden, um alle Abhängigkeiten zu aktualisieren, nachdem diese Option gesetzt wird). Zu beachten ist, dass das virtuelle Feld neu berechnet wird unabhängig von dem Datensatz, in welchem ein Feld geändert wurde, und unabhängig davon, ob das Feld in der Benutzeroberfläche oder während einer Programmausführung geändert wurde. Das automatische Aktualisieren von virtuellen Feldern wird aber normalerweise erst nach Abschluss aller Programmausführungen angestoßen. Ist 'Auto-Aktualisierung' ungesetzt, so wird der Wert des virtuellen Feldes nur dann berechnet, wenn er explizit abgefragt oder in einer Programmfunktion gesetzt wird.
- Für Knöpfe gibt es folgende zusätzlichen Felder:
 - ein Auswahlfeld 'Art', mit dem zwischen 'Textknopf' und 'Symbolknopf' gewählt werden kann.
 - 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.

15.3.5 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 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 Hintergrundes des Textobjektes.
- zwei Knöpfe **‘Ok’** und **‘Abbrechen’** zum Verlassen des Fensters

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

15.3.6 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 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 **‘veränderbar’** gewählt, dann kann das Bild in der Größe geändert werden. Bei **‘fixiert’** 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.

15.3.7 Zwischenraumeditor

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

- 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, wenn 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.

15.3.8 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 ‘Vert.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.

15.3.9 Karteikarteneditor

Man klickt doppelt auf ein Karteikartenobjekt, 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.
- Ein Bereich ‘Textmarken’ 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 15.2.3 [Labeleditor], Seite 70.
- zwei Knöpfe ‘Ok’ und ‘Abbrechen’ zum Verlassen des Fensters

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

15.3.10 Fenstereditor

Um die Einstellungen für ein Fensterobjekt zu ändern, wird doppelt auf das Fenstertobjekt geklickt. 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 ‘Id’ (nur für Amiga), das bis zu vier Zeichen aufnehmen kann und die MUI-Fenster-Id definiert. Wird eine Id eingegeben, so kann mittels der MUI-Funktion Window-Snapshot die aktuelle Fensterposition und -größe gespeichert werden. Zu beachten ist, dass die Id eindeutig für alle Fenster in MUIbase sein sollte. Wird eine Id aus versehen zweimal benutzt, so teilen sich die zugehörigen Fenster die gleichen Fensterausmaße! Ids, welche mit einem Unterstrich ‘_’ beginnen, sind für den internen

Gebrauch reserviert. Wird das Feld leer gelassen, so wird keine MUI-Fenster-Id gesetzt und die Fensterdimensionen werden in der Projektdatei gespeichert. Das Hauptfenster eines Projekts hat immer die ersten vier Zeichen des Projektnamens als Fenster-Id. Die Angabe einer Fenster-Id ist nützlich, wenn ein Projekt unter verschiedenen Rechnerkonfigurationen bearbeitet werden soll, da die Fensterpositionen dann aus der jeweiligen Konfiguration entnommen werden.

- ein Feld ‘Knopf’, das, wenn gesetzt, einen Knopf zum Öffnen des Fensters in das übergeordnete Fenster plaziert. Wird dieses Feld nicht gesetzt, so kann das Fenster nur von einem MUIbase-Programm aus mittels der Funktion SETWINDOWOPEN geöffnet werden (siehe Abschnitt 16.21.4 [SETWINDOWOPEN], Seite 148). Die nachfolgenden Felder spezifizieren das Aussehen des Fensterknopfes.
- ein Zeichenkettenfeld ‘Tastenkürzel’, in der die Taste zum Aktivieren des Fensterknopfes eingegeben wird.
- ein Bereich ‘Aktiv/inaktiv’. Ist ‘Immer aktiv’ gesetzt, so ist der Fensterknopf immer anwählbar. ‘Inaktiv im Initial-DS’ deaktiviert den Knopf, falls er innerhalb einer Tabelle plaziert wurde und die Tabelle den initialen Datensatz anzeigt, sonst ist der Fensterknopf aktiv. Wird ‘Berechne aktiv’ gewählt, so kann rechts davon eine Funktion zur Berechnung des Aktivzustands eingegeben werden. Die Funktion wird ohne Argumente aufgerufen. Liefert die Funktion NIL zurück, so wird der Knopf deaktiv, sonst aktiv. Im Falle, dass die Berechnungsfunktion leer gelassen wird oder nicht gefunden werden kann, wird der Fensterknopf deaktiviert. Für mehr Informationen über die Anwendung dieser Auslösefunktion, siehe Abschnitt 16.29.13 [Compute enabled function], Seite 164. Ist zusätzlich das Feld ‘Schließe Fenster wenn deaktiviert’ ausgewählt, so wird das Fenster automatisch geschlossen, wenn der Knopf deaktiviert 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 Hintergrundes des Fensterknopfes.
- zwei Knöpfe ‘Ok’ und ‘Abbrechen’ zum Verlassen des Fensters

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

15.4 Struktur exportieren

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 exportieren’ 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 dem jeweiligen Typ und der optionalen Auslösefunktion ausgegeben. Bei virtuellen Feldern werden außerdem die Abhängigkeiten für die automatische Aktualisierung aufgelistet.

16 MUIbase programmieren

Dieses Kapitel (das größte dieser Dokumentation) 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 (richtig funktionierende :-)) Programme geschrieben haben.

16.1 Programmeditor

Um ein Programm für ein Projekt einzugeben, öffnet man den Programmeditor über den Menüpunkt **‘Programm - Ändern’**. Sofern Sie die Einstellung Programmquellen intern (siehe Abschnitt 7.2.6 [Programmquellen], Seite 39) verwenden, öffnet dies 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 nicht-modales 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 eine Sicherheitsabfrage, die um Bestätigung fragt, ob das Fenster geschlossen werden darf.

Falls Sie Menüpunkt **‘Programm - Quellen’** auf **‘Extern’** gesetzt haben, so wird bei Wahl von **‘Programm - Ändern’** der externe Editor (siehe Abschnitt 7.1.2 [Externer Editor], Seite 35) mit dem Dateinamen der externen Quelldatei gestartet. Dies ermöglicht das Editieren der Programmquellen mit Ihrem Lieblingseditor (siehe Abschnitt 16.2 [Externe Programmquellen], Seite 83).

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 Include-Datei gemacht wurden und diese Änderungen in das Projektprogramm einfließen sollen.

16.2 Externe Programmquellen

Durch Wählen des Menüpunktes **‘Programm - Quellen - Extern’** und Eingabe eines Dateinamens können die Programmquellen eines Projekts extern verfügbar gemacht werden. Dies erlaubt es, die Quelldatei für die Programmierung in Ihren Lieblingseditor zu laden.

Bei erfolgreicher Übersetzung wird das übersetzte Programm als neues Projektprogramm übernommen und beim Aufruf von Auslösefunktionen verwendet. Wenn Sie ein Projekt

speichern, so wird das letzte erfolgreich übersetzte Program in der Projektdatei gespeichert. Von daher wird nach dem Speichern und Schließen eines Projekts die externe Programmquelldatei nicht mehr benötigt. Sie können in Menüpunkt ‘Einstellungen – Externe Programmquellen aufräumen’ einstellen, ob unbenötigte externe Quelldateien automatisch gelöscht werden sollen.

Der Zustand von Menüpunkt ‘Programm – Quellen’ wird mit einem Projekt gespeichert. Wenn Sie ein Projekt, welches die externe Quelldateien-Einstellung verwendet, erneut öffnen, so wird die externe Quelldatei nach dem Öffnen erzeugt. Falls die externe Datei bereits existiert und verschieden von der im Projekt gespeicherten Version ist, so erscheint eine Sicherheitsabfrage bevor die Datei überschrieben wird.

Auf dem Amiga kann aus dem Editor heraus der `compile`-Befehl an MUIbase’ ARexx-Port geschickt werden. MUIbase lädt daraufhin die externe Quelldatei, übersetzt sie, und liefert den Übersetzungsstatus mit einer optionalen Fehlermeldung zurück. Die Fehlermeldung enthält Dateiname, Zeile und Spalte, und eine Fehlerbeschreibung. Dies erlaubt es, auf die genau Fehlerstelle im Editor zu springen. Siehe Abschnitt 17.9 [ARexx compile], Seite 169, für weitere Details über Rückgabewerte und Fehlerformat.

16.3 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.

16.3.1 `#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`.

16.3.2 `#undef`

`#undef name`

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

Siehe auch `#define`, `#ifdef`, `#ifndef`.

16.3.3 `#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 Abschnitt 7.2.11 [Programm-Include-Verzeichnis], Seite 40) nach der zu ladenden Datei.

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.

16.3.4 `#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`.

16.3.5 `#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`.

16.3.6 `#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`.

16.3.7 `#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`.

16.3.8 `#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`.

16.3.9 `#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`.

16.4 Programmiersprache

MUIbase verwendet eine Programmiersprache mit einem 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`).

16.4.1 Warum Lisp?

Der Vorteil einer lisp-ähnlichen Sprache ist, dass man sowohl funktional als auch imperativ programmieren kann. Funktionale Sprachen sind in mathematischen Anwendungen weit verbreitet. Das Grundelement von funktionalen Sprachen ist die Verwendung von Ausdrücken. Funktionen werden in mathematischer Schreibweise definiert und häufig wird Rekursion benutzt.

Imperative Programmiersprachen (wie C, Pascal, Modula) benutzen eine Befehlsvorschrift (oder Zustandsänderungsvorschrift), wie etwas berechnet werden soll. Hier ist das Grundelement der Zustand (z.B. Variablen) und ein Programm berechnet dessen Ausgabe durch den Wechsel 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.

16.4.2 Lisp-Aufbau

Ein lisp-Ausdruck ist entweder eine Konstante, eine Variable oder ein Funktionsausdruck. Beim Funktionsaufruf verwendet Lisp eine Prefix-Notation. 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 aufgerufen, was bedeutet, dass zuerst die Parameter berechnet werden, bevor die Funktion aufgerufen wird.

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.

16.4.3 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 Programmmeditor (siehe Abschnitt 16.1 [Programmmeditor], Seite 83) 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 14.2 [Abfrageeditor], Seite 60) 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 Abschnitt 10.1.2 [Filter ändern], Seite 50) verändert.

16.4.4 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 (groß oder klein), Ziffern und Unterstriche sein. Andere Zeichen wie z.B. deutsche Umlaute sind nicht zulässig.

16.4.5 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.

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 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, 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 ‘Ralph’ 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 ‘Ralph’ und ‘Bar.Name’ liefert ‘Steffen’.

16.4.6 Datentypen zum Programmieren

Die Programmiersprache von MUIbase kennt die folgenden Datentypen:

Typ	Beschreibung
Boolesch	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 Programmierarten unterstützen den Wert NIL.

16.4.7 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. Werte, die mit 0 beginnen, werden als Octalzahlen interpretiert, Werte, die mit 0x beginnen, als Hexadezimalzahlen.

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	<p>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:</p> <table> <tr> <td><code>\n</code></td> <td>neue Zeile (nl)</td> </tr> <tr> <td><code>\t</code></td> <td>horizontaler Tabulator (ht)</td> </tr> <tr> <td><code>\v</code></td> <td>vertikaler Tabulator (vt)</td> </tr> <tr> <td><code>\b</code></td> <td>Rückschritt (bs)</td> </tr> <tr> <td><code>\r</code></td> <td>Wagenrücklauf (cr)</td> </tr> <tr> <td><code>\f</code></td> <td>Seitenumbruch (ff)</td> </tr> <tr> <td><code>\\</code></td> <td>der Backslash selbst</td> </tr> <tr> <td><code>\"</code></td> <td>doppeltes Anführungszeichen</td> </tr> <tr> <td><code>\e</code></td> <td>Escapezeichen 033</td> </tr> <tr> <td><code>\nnn</code></td> <td>Zeichen mit dem Oktalcode <i>nnn</i></td> </tr> <tr> <td><code>\xnn</code></td> <td>Zeichen mit dem Hexcode <i>nn</i></td> </tr> </table>	<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>
<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	Wie Zeichenkettenkonstanten.																						
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' die Stunden, 'MM' die Minuten und 'SS' die Sekunden repräsentieren.																						

Für andere vordefinierte Konstanten, siehe Abschnitt 16.25 [Pre-defined constants], Seite 154.

16.4.8 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.

16.5 Befehle definieren

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

16.5.1 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 16.4.4 [Namenskonventionen], Seite 88).

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 16.27 [Typdeklarierer], Seite 156).

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 15.1.1 [Tabellen erstellen], Seite 67, und Abschnitt 15.2.1 [Felder erstellen], Seite 69).

Dieser Befehl ist nur für Projektprogramme verfügbar.

Siehe auch `DEFUN*`, `DEFVAR`.

16.5.2 `DEFUN*`

`DEFUN*` ist die Stern-Variante von `DEFUN` und hat den selben Effekt wie `DEFUN` (siehe Abschnitt 16.5.1 [`DEFUN`], Seite 91). 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`.

16.5.3 `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 16.4.4 [Nameskonventionen], Seite 88).

Man kann Typdeklarierer an den Variablennamen anhängen (siehe Abschnitt 16.27 [Typdeklarierer], Seite 156).

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

Nach Beenden einer Trigger-Funktion (wenn MUIbase zum Benutzer-Interface zurückschaltet) geht der Inhalt aller globaler Variablen verloren. Diese werden beim nächsten Aufruf einer Auslösefunktion neu mit ihrem Vorgabewert `expr` initialisiert. Ist dies nicht erwünscht, so kann der Befehl `DEFVAR*` (siehe Abschnitt 16.5.4 [`DEFVAR*`], Seite 93) verwendet werden, welcher es erlaubt, den Inhalt globaler Variablen zwischen den Aufrufen zu speichern.

Bitte verwenden Sie globale Variablen sparsam (oder gar nicht). Alle globalen Variablen müssen bei jedem externen Aufruf einer Auslösefunktion neu initialisiert (und `expr` ausgewertet, wenn gegeben) werden.

Beispiel: ‘`(DEFVAR x 42)`’ definiert eine globale Variable ‘`x`’ mit dem Wert 42.

Es gibt einige vordefinierte Variablen in MUIbase (siehe Abschnitt 16.24 [Vordefinierte Variablen], Seite 154).

Siehe auch `DEFVAR*`, `DEFUN`, `DEFUN*`, `LET`.

16.5.4 DEFVAR*

```
(DEFVAR* var [expr])
```

DEFVAR* hat den gleichen Effekt wie der DEFVAR-Befehl (siehe Abschnitt 16.5.3 [DEFVAR], Seite 92) nur dass eine mit DEFVAR* definierte Variable nicht ihren Inhalt bei Programmende verliert.

Bei dem ersten Programmaufruf wird *var* mit dem Ausdruck *expr* oder NIL (falls *expr* fehlt) initialisiert. In allen weiteren Aufrufen wird *expr* nicht nocheinmal ausgewertet sondern der Wert von *var* aus dem letzten Aufruf verwendet. Dadurch ist es möglich, Informationen von einem Programmaufruf zum nächsten zu übermitteln, ohne Daten in externen Dateien oder in einer Datenbank-Tabelle zu speichern. Zu beachten ist aber, dass alle mit DEFVAR* definierten Variablen bei jeder Programmübersetzung ihren Wert verlieren. Falls Daten permanent gespeichert werden sollen, verwenden Sie hierfür ein (vorzugsweise verstecktes) Feld in einer Tabelle.

Siehe auch DEFVAR, DEFUN, DEFUN*, LET.

16.6 Programmsteuerungsfunktionen

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

16.6.1 PROGN

Um mehrere Ausdrücke der Reihe nach auszuführen, wird der PROGN-Aufruf 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.

16.6.2 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.

16.6.3 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 16.4.4 [Namenskonventionen], Seite 88).

Im Falle von (var *expr*) wird die neue Variable mit dem gegebenen Ausdruck initialisiert. Im anderen Fall ist die neue Variable auf NIL gesetzt.

Es ist auch möglich, Typdeklarierer an die Variablen zu hängen (siehe Abschnitt 16.27 [Typdeklarierer], Seite 156).

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.

16.6.4 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 Oberflächen-Datensatzzeigers: '(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 [Tabellen], 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, 2Tables, Tabellen, 2Semantics of expressions, Aufbau von Ausdrücken.

16.6.5 SETQ*

SETQ* ist die Stern-Variante von SETQ (siehe Abschnitt 16.6.4 [SETQ], Seite 94) und hat ähnliche Auswirkungen. Der Unterschied ist, dass SETQ* beim Zuweisen eines Wertes zu einer Variable die Auslösefunktion dieses Feldes aufruft (siehe Abschnitt 16.29.11 [Auslösefunktion Feld], Seite 162), 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.

16.6.6 FUNCALL

FUNCALL ruft eine Funktion mit Argumenten auf.

```
(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 funktionale Ausdrücke, siehe Abschnitt 16.26 [Funktionale Parameter], Seite 155.

Siehe auch APPLY.

16.6.7 APPLY

APPLY wendet eine Funktion auf eine Argumentenliste an.

```
(APPLY fun-expr [expr ...] list-expr)
```

Wendet die Funktion *fun-expr* auf eine Liste an, die dadurch erzeugt wird, indem die Argumente *expr* ... mittels cons vorne an *list-expr* angehängt werden. Anders gesagt wird die Funktion *fun-expr* mit den Argumenten *expr* ... und *list-expr*, ersetzt durch seine Listenelemente, aufgerufen.

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. Das letzte Argument *list-expr* muss eine gültige Liste oder NIL sein, sonst wird eine Fehlermeldung erzeugt. Stimmt die Anzahl der Parameter nicht, so wird ein Fehler gemeldet.

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

Mehr Informationen über funktionale Ausdrücke, siehe Abschnitt 16.26 [Funktionale Parameter], Seite 155.

Beispiel: ‘(APPLY + 4 (LIST 1 2 3))’ liefert 10.

Siehe auch FUNCALL.

16.6.8 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.

16.6.9 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 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.

16.6.10 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.

16.6.11 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. Der Name muss mit einem Kleinbuchstaben beginnen, gefolgt von weiteren Zeichen, Ziffern und Unterstrich-Zeichen. (siehe Abschnitt 16.4.4 [Namenskonventionen], Seite 88).

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.

16.6.12 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 16.4.4 [Namenskonventionen], Seite 88).

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.

16.6.13 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 Zählerweite.

Des Weiteren ist *term-expr* der Abbruchbedingungsausdruck, *result-expr ...* sind die Rückgabewertausdrücke (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 Rückgabewertausdrücke 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.

16.6.14 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 16.20.12 [SELECT], Seite 146, 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.

16.6.15 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.

16.6.16 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.

16.6.17 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.

16.6.18 HALT

HALT kann verwendet werden, um die Programmausführung abzubrechen.

(HALT)

stoppt stillschweigend die Programmausführung.

Siehe auch ERROR, EXIT, RETURN.

16.6.19 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 16.12.34 [SPRINTF], Seite 118) aus *fmt* und den optionalen Parametern *arg* ... erzeugt.

Siehe auch HALT, SPRINTF.

16.7 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
(STRP <i>expr</i>)	TRUE wenn <i>expr</i> vom Typ Zeichenkette ist, sonst NIL.
(MEMOP <i>expr</i>)	TRUE wenn <i>expr</i> vom Typ mehrzeiliger Text ist, sonst NIL.
(INTP <i>expr</i>)	TRUE wenn <i>expr</i> vom Typ Ganzzahl ist, sonst NIL.
(REALP <i>expr</i>)	TRUE wenn <i>expr</i> vom Typ Fließkommazahl ist, sonst NIL.
(DATEP <i>expr</i>)	TRUE wenn <i>expr</i> vom Typ Datum ist, sonst NIL.
(TIMEP <i>expr</i>)	TRUE wenn <i>expr</i> vom Typ Zeit ist, sonst NIL.
(NULL <i>expr</i>)	TRUE wenn <i>expr</i> vom Typ NIL (leere Liste) ist, sonst NIL.
(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.

16.8 Typumwandlungsfunktionen

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

16.8.1 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.
Boolesch	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.

Siehe auch MEMO, SPRINTF.

16.8.2 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 16.8.1 [STR], Seite 101) auf, liefert aber einen mehrzeiligen Text statt einer Zeichenkette.

Siehe auch STR.

16.8.3 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. Beginnt die Zeichenkette mit 0, so wird sie als Oktalzahl interpretiert, beginnt sie mit 0x, als Hexadezimalzahl. Führende und folgende Leerzeichen werden ignoriert. Stellt die Zeichenkette keine Ganzzahl dar, so wird NIL geliefert.
mehrzeiliger Text	Analog zu Zeichenkette.
Ganzzahl	Der Wert selbst.
Fließkommazahl	Wenn der Wert im Ganzzahlbereich liegt, wird der gerundetete Fließkommawert geliefert, anderenfalls NIL.
Auswahltext	Die interne Nummer (beginnend bei 0) des Auswahltextes.
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 wird abgebrochen.

Siehe auch REAL, ASC.

16.8.4 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 16.8.3 [INT], Seite 102) auf, liefert aber einen Wert vom Typ Fließkommazahl anstatt einer Ganzzahl.

Siehe auch INT.

16.8.5 DATE

DATE wird verwendet, um einen Ausdruck in ein Datum umzuwandeln.

(DATE *expr*)

wandelt den gegebenen Ausdruck in ein Datum. Mögliche Umwandlungen sind:

Typ	Ergebniswert
Zeichenkette	Wenn die gesamte Zeichenkette einen Datum darstellt, dann wird die Zeichenkette in ein Datum umgewandelt. Führende und folgende Leerzeichen werden ignoriert. Stellt es kein Datum dar, wird NIL geliefert.
mehrzeiliger Text	Analog zu Zeichenkette.
Ganzzahl	Ein Datumswert wird erzeugt, der die gegebene Ganzzahl als 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 zu Ganzzahl.
Datum	Der Wert selbst.
NIL	NIL
Andere	Eine Fehlermeldung wird erzeugt und die Programmausführung wird abgebrochen.

Siehe auch DATEDMY.

16.8.6 TIME

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

(TIME *expr*)

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

Typ	Ergebniswert
-----	--------------

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 zu Zeichenkette.
Ganzzahl	Ein Zeitwert wird erzeugt, der die gegebene Ganzzahl als Anzahl der Sekunden seit 00:00:00 darstellt.
Fließkommazahl	Analog zu Ganzzahl.
Zeit	Der Wert selbst.
NIL	NIL
Andere	Eine Fehlermeldung wird erzeugt und die Programmausführung wird abgebrochen.

16.9 Boolesche Funktionen

Dieser Abschnitt listet die booleschen Operatoren auf.

16.9.1 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.

16.9.2 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.

16.9.3 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.

16.10 Vergleichsfunktionen

In diesem Abschnitt findet man Funktionen zum Vergleich von Werten vor.

16.10.1 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 < ... < 596523:14:07
Boolesch	NIL < TRUE
Datensatz	NIL < jeder_Datensatz (Datensätze selbst können nicht mit < verglichen werden) NIL <* rec1 <* rec2 (Reihenfolge festgelegt durch Benutzer)

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.

Siehe auch CMP, CMP*, LIKE.

16.10.2 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 16.10.1 [Relationsoperatoren], Seite 105) verwendet.

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

Beispiel: '(CMP "Bike" "bIKE")' liefert -1.

Siehe auch CMP*, 2Relational operators, Relationsoperatoren.

16.10.3 CMP*

CMP* ist die Stern-Version von CMP. Der Unterschied ist, dass CMP* eine erweiterte Sortierung wie bei den Relationsoperatoren (siehe Abschnitt 16.10.1 [Relationsoperatoren], Seite 105) verwendet, bei denen Zeichenketten zeichengrößenunabhängig und Datensätze gemäß ihrer benutzerdefinierten Datensatzreihenfolge verglichen werden.

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

Siehe auch CMP, 2Relational operators, Relationsoperatoren.

16.10.4 MAX

MAX liefert den Ausdruck mit dem größten Wert.

(MAX [*expr ...*])

Liefert den größten Wert der Argumente *expr ...*. Werden keine Argumente angegeben, so wird NIL zurückgegeben. Zum Bestimmen des größten Ausdrucks wird die einfache (ohne Stern) Sortierrelation wie bei den Relationaloperatoren (siehe Abschnitt 16.10.1 [Relationsoperatoren], Seite 105) verwendet.

Siehe auch MAX*, MIN, 2Relational operators, Relationsoperatoren.

16.10.5 MAX*

MAX* ist die Stern-Version von MAX. Der Unterschied ist, dass MAX* eine erweiterte Sortierung wie bei den Relationsoperatoren (siehe Abschnitt 16.10.1 [Relationsoperatoren], Seite 105) verwendet, bei denen Zeichenketten zeichengrößenunabhängig und Datensätze gemäß ihrer benutzerdefinierten Datensatzreihenfolge verglichen werden.

Beispiel: '(MAX* "x" "Y")' ergibt "Y".

Siehe auch MAX, MIN*, 2Relational operators, Relationsoperatoren.

16.10.6 MIN

MIN liefert den Ausdruck mit dem kleinsten Wert.

(MIN [*expr* ...])

Liefert den kleinsten Wert der Argumente *expr* ... Werden keine Argumente angegeben, so wird NIL zurückgegeben. Zum Bestimmen des kleinsten Ausdrucks wird die einfache (ohne Stern) Sortierrelation wie bei den Relationaloperatoren (siehe Abschnitt 16.10.1 [Relationsoperatoren], Seite 105) verwendet.

Siehe auch MIN*, MAX, 2Relational operators, Relationsoperatoren.

16.10.7 MIN*

MIN* ist die Stern-Version von MIN. Der Unterschied ist, dass MIN* eine erweiterte Sortierung wie bei den Relationsoperatoren (siehe Abschnitt 16.10.1 [Relationsoperatoren], Seite 105) verwendet, bei denen Zeichenketten zeichengrößenunabhängig und Datensätze gemäß ihrer benutzerdefinierten Datensatzreihenfolge verglichen werden.

Beispiel: '(MIN* "x" "Y")' ergibt "x".

Siehe auch MIN, MAX*, 2Relational operators, Relationsoperatoren.

16.11 Mathematik-Funktionen

Einige mathematische Funktionen werden hier aufgelistet.

16.11.1 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 oder andere Zeitwerte, dann wird die Summe der Ganz-/Fließkommazahlen (als Anzahl Sekunden aufgefasst) und Zeitwerte zu *expr* addiert. Ist das zurückgelieferte Ergebnis außerhalb des gültigen Bereichs (kleiner als 00:00:00 oder größer als 596523:14:07), dann ist das Ergebnis NIL.

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)	26:59:59

Siehe auch 2sub,-, 1+, 2mul,*, CONCAT, CONCAT2,ADDMONTH, ADDYEAR.

16.11.2 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 16.11.1 [Werte addieren], Seite 107, 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 2add,+, 1-.

16.11.3 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 2add,+, 1-.

16.11.4 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 2sub,-, 1+.

16.11.5 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 2add,+, 2fdiv,/.

16.11.6 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 `2mul,*`, `DIV`, `MOD`.

16.11.7 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 `2fdiv,/`, `MOD`.

16.11.8 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`.

16.11.9 ABS

`ABS` berechnet den absoluten Wert eines Ausdrucks.

`(ABS expr)`

Liefert den absoluten Wert von *expr* (Ganz- oder Fließkommazahl). Ist *expr* NIL, dann wird NIL geliefert.

16.11.10 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`.

16.11.11 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 390.0.

Siehe auch `TRUNC`.

16.11.12 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.

Beispiele:

Beispiel	Bedeutung
(RANDOM 10)	liefert einen Wert von 0 bis 9,
(RANDOM 10.0)	liefert einen Wert von 0.0 bis 9.99999...

16.11.13 POW

POW berechnet die Potenz von Zahlen.

(POW *x y*)

Liefert den Wert der Fließkommazahl *x* potenziert mit Fließkommazahl *y*. Ist *x* or *y* NIL, oder ist *x* negativ und *y* keine Ganzzahl, so wird NIL zurückgegeben.

Beispiel: '(POW 2 3)' ergibt 8.

Siehe auch SQRT, EXP.

16.11.14 SQRT

SQRT berechnet die Wurzel einer Zahl.

(SQRT *x*)

Gibt die Quadratwurzel der Fließkommazahl *x* zurück. Ist *x* NIL oder ein negativer Wert, so wird NIL zurückgegeben.

Siehe auch POW.

16.11.15 EXP

EXP berechnet die Exponentialfunktion.

(EXP *x*)

Liefert den Wert der Basis des natürlichen Logarithmus potenziert mit Fließkommazahl *x*. Ist *x* NIL, so wird NIL zurückgegeben.

Siehe auch POW, LOG.

16.11.16 LOG

LOG berechnet den natürlichen Logarithmus einer Zahl.

(LOG *x*)

Liefert den natürlichen Logarithmus der Fließkommazahl *x*. Ist *x* NIL oder keine positive Zahl, so wird NIL zurückgegeben.

Siehe auch EXP.

16.12 Zeichenkettenfunktionen

Dieser Abschnitt behandelt Funktionen für Zeichenketten.

16.12.1 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.

16.12.2 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.

16.12.3 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.

16.12.4 MIDSTR

MIDSTR extrahiert eine Teilzeichenkette aus einer Zeichenkette.

(MIDSTR *str pos len*)

Liefert einen Teil einer gegebenen Zeichenkette mit höchstens *len* Zeichen. Die Teilzeichenkette beginnt an der Stelle *pos* (beginnend bei Null). Ist *len* NIL, dann wird die komplette Teilzeichenkette, welche bei *pos* beginnt, zurückgegeben. Falls *str* NIL oder *len* negativ ist, dann wird NIL zurückgeliefert. Ist *pos* außerhalb des gültigen Bereichs (negativ oder größer als die Zeichenkettenlänge), dann wird NIL geliefert.

Beispiel: '(MIDSTR "Hallo Welt!" 3 5)' liefert "lo We".

Siehe auch LEFTSTR, RIGHTSTR, WORD, LINE, SETMIDSTR, INSMIDSTR.

16.12.5 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.

16.12.6 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 eines der Parameter NIL oder ist *index* außerhalb des gültigen Bereichs, dann wird NIL zurückgeliefert.

Beispiel: ‘(INSMIDSTR "Hallo Welt!" 6 "MUIbase-")’ liefert "Hallo MUIbase-Welt!".

Siehe auch SETMIDSTR, REPLACESTR.

16.12.7 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*.

16.12.8 INDEXSTR*

INDEXSTR* hat den selben Effekt als INDEXSTR (siehe Abschnitt 16.12.7 [INDEXSTR], Seite 112), außer dass der Zeichenkettenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch INDEXSTR, RINDEXSTR, RINDEXSTR*, INDEXBRK, INDEXBRK*.

16.12.9 INDEXBRK

INDEXBRK sucht nach dem ersten Vorkommen eines Zeichens in einer Zeichenkette.

(INDEXBRK *str brkstr*)

Sucht nach dem ersten Vorkommen eines Zeichens von *brkstr* 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*.

16.12.10 INDEXBRK*

INDEXBRK* hat den selben Effekt wie INDEXBRK (siehe Abschnitt 16.12.9 [INDEXBRK], Seite 112), außer dass der Zeichenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch INDEXBRK, RINDEXBRK, RINDEXBRK*, INDEXSTR, INDEXSTR*.

16.12.11 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.

Siehe auch RINDEXSTR*, INDEXSTR, INDEXSTR*, RINDEXBRK, RINDEXBRK*.

16.12.12 RINDEXSTR*

RINDEXSTR* hat den selben Effekt als RINDEXSTR (siehe Abschnitt 16.12.11 [RINDEXSTR], Seite 113), außer dass der Zeichenkettenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch RINDEXSTR, INDEXSTR, INDEXSTR*, RINDEXBRK, RINDEXBRK*.

16.12.13 RINDEXBRK

RINDEXBRK sucht nach dem letzten Vorkommen eines Zeichens in einer Zeichenkette.

(RINDEXBRK *str brkstr*)

Sucht nach dem letzten Vorkommen eines Zeichens von *brkstr* 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.

Siehe auch RINDEXBRK*, INDEXBRK, INDEXBRK*, RINDEXSTR, RINDEXSTR*.

16.12.14 RINDEXBRK*

RINDEXBRK* hat den selben Effekt wie RINDEXBRK (siehe Abschnitt 16.12.13 [RINDEXBRK], Seite 113), außer dass der Zeichenvergleich nicht auf Groß-/Kleinschreibung achtet.

Siehe auch RINDEXBRK, INDEXBRK, INDEXBRK*, RINDEXSTR, RINDEXSTR*.

16.12.15 REPLACESTR

REPLACESTR ersetzt Teilzeichenketten durch andere Zeichenketten.

(REPLACESTR *str [substr1 replacestr1 ...]*)

Ersetzt alle Vorkommen von *substr1* in *str* durch *replacestr1*. Fortführend werden mit der nächsten Such- und Ersetzungszeichenkette weitere Teilzeichenketten in der neu erhaltenen Zeichenkette ersetzt bis alle Argumente abgearbeitet sind. Zu beachten ist, dass die Anzahl der Argumente ungerade sein muss. Argumente an geraden Positionen geben die Such-Teilzeichenketten an, jeweils gefolgt von der Ersetzungszeichenkette. Aufgrund der Tatsache, dass das Ergebnis einer Ersetzung für die darauf folgende Ersetzung verwendet wird, können Mehrfacheretzungen auftreten. Dies sollte bei Verwendung dieser Funktion beachtet werden. Das Ändern der Reihenfolge der Argumente kann helfen, Konflikte aufzulösen, da Ersetzungen immer von links nach rechts durchgeführt werden.

Wenn eine der Zeichenketten NIL ist oder einer der Such-Teilzeichenketten leer ist, dann wird NIL zurückgeliefert.

Beispiel: ‘(REPLACESTR "black is white" "black" "white" "white "black")’ results to "black is black".

Siehe auch REPLACESTR*, SETMIDSTR, INSMIDSTR, REMCHARS.

16.12.16 REPLACESTR*

REPLACESTR* hat den selben Effekt wie REPLACESTR (siehe Abschnitt 16.12.15 [REPLACESTR], Seite 113), außer dass der Zeichenvergleich für die Suche nach Teilzeichenketten nicht auf Groß-/Kleinschreibung achtet.

Siehe auch REPLACESTR, SETMIDSTR, INSMIDSTR, REMCHARS.

16.12.17 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.

16.12.18 TRIMSTR

TRIMSTR entfernt führende und abschließende Zeichen von einer Zeichenkette.

(TRIMSTR *str [front back]*)

Liefert eine Kopie von *str*, in der führenden und abschließenden Zeichen entfernt wurden. Bei Aufruf mit nur einem Argument werden Leerzeichen, Seiten- und Zeilenumbrüche, Carriage Return, sowie horizontale und vertikale Tabulatoren entfernt. Werden drei Argumente verwendet, so gibt *front* die führenden und *back* die abschließenden Zeichen an, die entfernt werden sollen. TRIMSTR kann nicht mit zwei Argumenten aufgerufen werden.

Ist eines der Argumente *str*, *front* oder *back* NIL, so wird NIL zurückgegeben.

Beispiel: (TRIMSTR " Ich fuhr Selma's Fahrrad zu Schrott. ") liefert "Ich fuhr Selma's Fahrrad zu Schrott.", (TRIMSTR "007 " "0" " \f\n\r\t\v") ergibt "7".

Siehe auch REMCHARS.

16.12.19 WORD

WORD liefert ein Wort einer Zeichenkette.

(WORD *str num*)

Liefert das *num*-te Wort (beginnend bei Null) aus der gegebenen Zeichenkette. Wörter sind nicht-leere Teilzeichenketten, die durch ein oder mehrere nicht-brechbare Leerzeichen (z.B. Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen) getrennt sind.

Ist *str* oder *num* gleich NIL oder ist *num* außerhalb des gültigen Bereichs (negativ oder größer gleich der Anzahl Wörter in der Zeichenkette), dann wird NIL zurückgeliefert.

Beispiel: ‘(WORD "Deshalb lieb ich Selma mein Fahrrad." 3)’ liefert "Selma".

Siehe auch WORDS, FIELD, LINE, LEFTSTR, RIGHTSTR, MIDSTR, STRTOLIST.

16.12.20 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 nicht-leere Teilzeichenketten, die durch ein oder mehrere nicht-brechbare Leerzeichen (z.B. Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen) getrennt sind.

Beispiel: '(WORDS "In Wirklichkeit ist es aber nicht mein Fahrrad.")' liefert 8.

Siehe auch WORD, FIELDS, LINES, LEN.

16.12.21 FIELD

FIELD liefert ein Feld in einer Zeichenkette.

(FIELD *str num [sep [quotes]]*)

Liefert das *num*-te Feld (beginnend bei Null) in der gegebenen Zeichenkette. Felder sind Teilzeichenketten, die durch genau ein Trennzeichen getrennt sind. Ein Feld kann eine leere Zeichenkette sein. Argument *sep* enthält die Trennzeichen, durch die Felder getrennt sind. Ist *sep* NIL oder wird es nicht angegeben, so werden nicht-brechbare Leerzeichen (z.B. Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen) verwendet. Wird *quotes* angegeben und ist nicht NIL, so können Felder mit doppelten Anführungszeichen umgeben sein und Trennzeichen enthalten. Die doppelten Anführungszeichen werden entfernt bevor das Feld zurückgegeben wird.

Ist *str* oder *num* gleich NIL oder ist *num* außerhalb des gültigen Bereichs (negativ oder größer gleich der Anzahl Wörter in der Zeichenkette), dann wird NIL zurückgeliefert.

Beispiel: '(FIELD "Mein Name ist \"Darth Vader\"" 3 " " TRUE)' liefert "Darth Vader".

Siehe auch FIELDS, WORD, LEFTSTR, RIGHTSTR, MIDSTR, STRTOLIST.

16.12.22 FIELDS

FIELDS zählt die Anzahl der Felder in einer Zeichenkette.

(FIELDS *str [sep [quotes]]*)

Liefert die Anzahl Felder in der gegebenen Zeichenkette oder NIL, wenn *str* NIL ist. Felder sind Teilzeichenketten, die durch genau ein Trennzeichen getrennt sind. Ein Feld kann eine leere Zeichenkette sein. Argument *sep* enthält die Trennzeichen, durch die Felder getrennt sind. Ist *sep* NIL oder wird es nicht angegeben, so werden nicht-brechbare Leerzeichen (z.B. Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen) verwendet. Wird *quotes* angegeben und ist nicht NIL, so können Felder mit doppelten Anführungszeichen umgeben sein und Trennzeichen enthalten.

Siehe auch FIELD, WORDS, LEN.

16.12.23 STRTOLIST

STRTOLIST wandelt eine Zeichenkette in eine Liste von Zeichenketten um.

(STRTOLIST *str [sep]*)

Erzeugt eine Liste von Zeichenketten, welche durch das Aufteilen der Zeichenkette *str* an den Stellen der Trennungs-Sequenz *sep* entsteht. Wird *sep* nicht angegeben, so wird

das Tab-Zeichen "\t" verwendet. Ist *sep* die leere Zeichenkette "", so wird eine Liste aller Zeichen der Zeichenkette generiert.

Ist *str* oder *sep* NIL, so wird NIL zurückgegeben.

Beispiele

‘(STRTOLIST "Ich\tmag\tJapan.")’ ergibt ("Ich" "mag" "Japan.").

‘(STRTOLIST "Name|Straße|Stadt" "|")’ ergibt ("Name" "Straße" "Stadt").

‘(STRTOLIST "abc" "")’ ergibt ("a" "b" "c").

Siehe auch MEMOTOLIST, LISTTOSTR.

16.12.24 LISTTOSTR

LISTTOSTR wandelt eine Liste von Elementen in eine Zeichenkette um.

(LISTTOSTR *list* [*sep*])

Wandelt die gegebene Liste von Elementen in eine Zeichenkette um, indem die Zeichenketten-Repräsentationen jedes Elements getrennt durch die Sequenz *sep* aneinanderghängt werden. Wird *sep* nicht angegeben, so wird das Tab-Zeichen "\t" verwendet. Ist *list* oder *sep* NIL, so wird NIL zurückgegeben.

Beispiele

‘(LISTTOSTR (LIST "Peter ist" 18 "Jahre alt"))’ ergibt "Peter ist\t18\tJahre alt".

‘(LISTTOSTR (LIST "Name" "Straße" "Stadt") "|")’ ergibt "Name|Straße|Stadt".

Siehe auch LISTTOMEMO, CONCAT, CONCAT2, STRTOLIST.

16.12.25 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, 2add,+, LISTTOSTR, COPYSTR, SPRINTF.

16.12.26 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, 2add,+, LISTTOSTR, COPYSTR, SPRINTF.

16.12.27 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, 2add,+, SPRINTF.

16.12.28 SHA1SUM

SHA1SUM berechnet den SHA1-Hash einer Zeichenkette.

(SHA1SUM *str*)

Gibt eine Zeichenkette zurück, welche den SHA1-Hash der gegebenen Zeichenkette enthält. Ist *str* NIL, so wird NIL zurückgegeben.

Beispiel: '(SHA1SUM "flower, sun and beach")' ergibt "47b6c496493c512b40e042337c128d85ecf15ba4".

Siehe auch ADMINPASSWORD, Beispielprojekt Users.mb.

16.12.29 UPPER

UPPER wandelt eine Zeichenkette in Großbuchstaben.

(UPPER *str*)

Liefert eine Kopie der gegebenen Zeichenkette, in der alle Zeichen in Großbuchstaben umgewandelt wurden. 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.".

Siehe auch LOWER.

16.12.30 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.

16.12.31 ASC

ASC wandelt ein Zeichen in die interne Zahlendarstellung um.

(ASC *str*)

Liefert den internen Zahlenwert des ersten Zeichens von *str*. Auf Windows, Mac OS und Linux ist dies die Unicode-Darstellung. Auf Amiga ist es der 8-bit-Zahlenwert in der vom System vorgegebenen Zeichen-Kodierung. Ist *str* leer, wird 0 geliefert. Ist *str* NIL, dann wird NIL geliefert.

Beispiel: (ASC "A") liefert 65.

Siehe auch CHR, INT.

16.12.32 CHR

CHR wandelt einen Zahlenwert in ein Zeichen um.

(CHR *int*)

Liefert eine Zeichenkette, die das Zeichen mit dem Zahlen-Code *int* enthält. Auf Linux und Windows wird *int* als Unicode-Zeichen interpretiert. Auf Amiga ist *int* der 8-bit Zahlenwert in der vom System vorgegebenen Zeichenkodierung. Ist *int* gleich 0, dann wird eine leere Zeichenkette geliefert. Ist *int* NIL oder nicht im Bereich der zulässigen Zeichenwerte, so wird NIL geliefert.

Beispiel: '(CHR 99)' liefert "c".

Siehe auch ASC, STR.

16.12.33 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 (inklusive der leeren Zeichenkette) darstellt. Der Zeichenkettenvergleich wird ohne Beachtung der Groß-/Kleinschreibung durchgeführt.

Beispiel: '(LIKE "Silke war für ein Jahr in Frankreich." "*Jahr*")' liefert TRUE.

Siehe auch 2Comparison functions, Vergleichsfunktionen.

16.12.34 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`

wobei

- 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 führenden Nullen aufgefüllt.
- Das optionale Feld *precision* legt entweder die maximale Anzahl von auszugebenden Zeichen für die Typen Zeichenkette, Boolesch, Datum und Zeit oder die Anzahl der Ziffern nach dem Dezimalpunkt zur Ausgabe eines Fließkommawertes fest.

- Das Feld *type* legt den gewünschten Type des Parameters fest, den `PRINTF` umwandeln soll, wie etwa Zeichenkette, Ganzzahl, Fliekkommazahl etc.

Zu beachten ist, dass alle Felder außer *type* optional sind. Die folgenden Tabellen listen die gültigen Optionen für diese Felder auf.

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.
- 0: Für Zahlen wird linksbündig mit führenden Nullen anstatt mit Leerzeichen aufgefüllt.

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 oder führenden Nullen 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 Zeichenketten-, Boolesche, Datums- und Zeit-Werte ist *n* die maximale Anzahl der auszugebenden Zeichen vom umgewandelten Element. Für Umwandlungen von Fließkommazahlen legt *n* die Anzahl der Nachkommastellen fest (Umwandlungen 'e' und 'f') oder die Anzahl signifikanter Stellen (Umwandlung 'g'). Für Ganzzahlkonvertierungen wird dieser Wert ignoriert.
- .*: Die Genauigkeit wird in der Parameterliste als Ganz- oder Fließkommazahl vor dem eigentlichen Umwandlungsparameter mitgeliefert. Der Wert ist beschränkt auf 0 bis 999.

Typenfeld *type*

- b: Wandelt einen Booleschen Parameter nach "TRUE" (wahr) oder "NIL" (falsch).
- i: Wandelt eine Ganzzahl in Dezimalzahlnotation um.
- o: Wandelt eine Ganzzahl in Octalzahlnotation um.
- x: Wandelt eine Ganzzahl in Hexadezimalzahlnotation um. Es werden Kleinbuchstaben 'abcdef' verwendet.

- X: Wandelt eine Ganzzahl in Hexadezimalzahlnotation um. Es werden Großbuchstaben ‘ABCDEF’ verwendet.
- 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.
- g: Wandelt eine Fließkommazahl entweder mittels Stil ‘e’ oder ‘f’ um, je nach Anzahl der benötigten Stellen, um die Zahl darzustellen. Ist die Genauigkeit nicht festgelegt, so werden 15 signifikante Stellen verwendet. Nachfolgende Nullen mit der Ausnahme einer einzelnen Null nach dem Dezimalpunkt werden unterdrückt.
- s: Schreibt eine Zeichenkette bis zum Ende der Zeichenkette oder so viele 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)	" 3 "
(SPRINTF "%f" 12)	"12.00"
(SPRINTF "%10e" 12.0)	" 1.20e+01"
(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, 2add,+,CONCAT, CONCAT2, COPYSTR.

16.13 Funktionen für mehrzeilige Texte

Dieser Abschnitt behandelt Funktionen für mehrzeilige Texte.

16.13.1 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.

16.13.2 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.

16.13.3 MEMOTOLIST

MEMOTOLIST wandelt einen mehrzeiligen Text in eine Liste von Zeichenketten um.

(MEMOTOLIST *memo* [*expandstr*])

Wandelt den gegebenen mehrzeiligen Text in eine Liste um. Ist *memo* gleich NIL, dann wird NIL geliefert, anderenfalls wird eine Liste erzeugt, in der jedes Element eine Zeile des mehrzeiligen Textes enthält.

Wird *expandstr* angegeben und ist nicht NIL, so wird die resultierende Liste von Zeichenketten weiter unterteilt, indem die Funktion STRTOLIST auf jedes Listenelement angewendet wird. Dies ergibt eine Liste von Listen von Zeichenketten.

Beispiele

‘(MEMOTOLIST "Meine Versicherung\nzahlt für\ndas kaputte Fahrrad.")’ liefert ("Meine Versicherung" "zahlt für" "das kaputte Fahrrad.").

‘(MEMOTOLIST "Hier ist\tein mehr-spaltiges\nBeispiel." TRUE)’ liefert (("Hier ist" "ein mehr-spaltiges") ("Beispiel")).

Siehe auch STRTOLIST, LISTTOMEMO.

16.13.4 LISTTOMEMO

LISTTOMEMO wandelt eine Liste in einen mehrzeiligen Text um.

(LISTTOMEMO *list*)

Wandelt eine gegebene Liste in einen mehrzeiligen Text um. Ist *list* gleich NIL, dann wird NIL zurückgegeben, anderenfalls wird ein mehrzeiliger Text erzeugt, dessen einzelne Zeilen die Zeichenkettendarstellung des entsprechenden Listenelements enthalten. Falls ein Listenelement eine Unterliste enthält, so wird LISTTOSTR (siehe Abschnitt 16.12.24 [LISTTOSTR], Seite 116) auf die Unterliste angewandt bevor die resultierende Zeichenkette in den mehrzeiligen Text aufgenommen wird.

Beispiele

‘(LISTTOMEMO (LIST "Silke" "leiht mir" "'mein' Fahrrad" "bis zum" 01.09.1998))’
liefert: "Silke\nleiht mir\n'mein' Fahrrad\nbis zum\n01.09.1998".

‘(LISTTOMEMO (LIST (LIST "Name" "Geburtstag") (LIST "Steffen" 28.11.1968)))’
liefert: "Name\tGeburtstag\nSteffen\t28.11.1968".

Siehe auch LISTTOSTR, MEMOTOLIST.

16.13.5 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 ist¹.

Siehe auch FORMATMEMO, INDENTMEMO.

16.13.6 FORMATMEMO

FORMATMEMO formatiert einen mehrzeiligen Text.

(FORMATMEMO *memo width [fill [singlelinesec]]*)

Formatiert *memo* in einen mehrzeiligen Text mit Zeilen, die nicht länger als *width* Zeichen sind. Wird *fill* angegeben und ist 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. Wird *singlelinesec* angegeben und ist nicht NIL, dann gehören alle Zeichen in dieser Zeile zu dem Abschnitt. Andernfalls zählen alle Zeichen in dieser und den folgenden Zeilen zu dem Abschnitt, bis eine Zeile gefunden wird, die mit einem Leerzeichen beginnt. Der gesamte Abschnitt wird wortweise formatiert, dies bedeutet, dass so viele Wörter in eine Zeile gesetzt werden, wie dafür vorhanden Platz ist. Die restlichen Wörter kommen dann in eine neue Zeile, usw.

¹ 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.

Siehe auch FILLMEMO, INDENTMEMO.

16.13.7 INDENTMEMO

INDENTMEMO rückt einen mehrzeiligen Text ein, in dem links Leerzeichen eingefügt werden.

(INDENTMEMO *memo indent*)

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.

16.14 Datum- und Zeitfunktionen

Dieser Abschnitt behandelt Funktionen für Datum und Zeit.

16.14.1 DAY

DAY extrahiert den Tag eines Datums.

(DAY *date*)

Liefert eine Ganzzahl, welche den Tag des gegebenen Datums repräsentiert. Ist *date* NIL, so wird NIL zurückgegeben.

Siehe auch MONTH, YEAR, DATEDMY.

16.14.2 MONTH

MONTH extrahiert den Monat eines Datums.

(MONTH *date*)

Liefert eine Ganzzahl, welche den Monat des gegebenen Datums repräsentiert. Ist *date* NIL, so wird NIL zurückgegeben.

Siehe auch DAY, YEAR, DATEDMY, MONTHDAYS.

16.14.3 YEAR

YEAR extrahiert das Jahr eines Datums.

(YEAR *date*)

Liefert eine Ganzzahl, welche das Jahr des gegebenen Datums repräsentiert. Ist *date* NIL, so wird NIL zurückgegeben.

Siehe auch DAY, MONTH, DATEDMY, YEARDAYS.

16.14.4 DATEDMY

DATEDMY erstellt ein Datum aus Tag, Monat und Jahr.

(DATEDMY *day month year*)

Erstellt ein Datum aus dem gegebenen Tag, Monat und Jahr. Ist eines von *day*, *month*, oder *year* NIL oder außerhalb des gültigen Bereichs, oder ist das resultierende Datum ungültig, so wird NIL geliefert.

Beispiel: '(DATEDMY 28 11 1968)' ergibt den 28. November, 1968.

Siehe auch DATE, TODAY, DAY, MONTH, YEAR.

16.14.5 MONTHDAYS

MONTHDAYS bestimmt die Anzahl Tage eines Monats.

(MONTHDAYS *month year*)

Liefert die Anzahl Tage des gegebenen Monats und Jahrs als Ganzzahlwert. Ist *month* NIL oder außerhalb des gültigen Bereichs (kleiner als 1 oder größer als 12), so wird NIL zurückgegeben. Ist *year* NIL, so wird ein Nicht-Schaltjahr für die Berechnung der Anzahl Tage betrachtet. Falls *year* ungültig ist (kleiner als 0 oder größer als 9999), so wird NIL geliefert.

Beispiele: '(MONTHDAYS 2 2004)' liefert 29, '(MONTHDAYS 2 NIL)' ergibt 28.

Siehe auch YEARDAYS, MONTH.

16.14.6 YEARDAYS

YEARDAYS bestimmt die Anzahl Tage eines Jahres.

(YEARDAYS *year*)

Liefert die Anzahl Tage des gegebenen Jahrs als Ganzzahlwert. Ist *year* NIL oder außerhalb des gültigen Bereichs (kleiner als 0 oder größer als 9999), so wird NIL zurückgegeben.

Beispiele: '(YEARDAYS 2004)' liefert 366, '(YEARDAYS 2005)' ergibt 365.

Siehe auch MONTHDAYS, YEAR.

16.14.7 ADDMONTH

ADDMONTH addiert eine Anzahl Monate zu einem Datum.

(ADDMONTH *date months*)

Liefert ein Datum, bei welchem die gegebene Anzahl Monate zu dem gegebenen Datum addiert wurde. Negative Werte für *months* subtrahieren Monate. Falls *date* oder *months* NIL sind, oder das resultierende Datum ungültig ist, so wird NIL geliefert.

ADDMONTH behandelt einen Über- oder Unterlauf des Monatfelds, indem das Jahresfeld entsprechend angepasst wird. Falls das Tagfeld die maximale Anzahl Tage des resultierenden Monats überschreitet, so wird es auf den maximal erlaubten Tag reduziert.

Beispiele: '(ADDMONTH 30.01.2004 1)' ergibt 29.02.2004, '(ADDMONTH 30.01.2004 -1)' ergibt 30.12.2003.

Siehe auch ADDYEAR, 2add,+.

16.14.8 ADDYEAR

ADDEAR addiert eine Anzahl Jahre zu einem Datum.

(ADDEAR *date years*)

Liefert ein Datum, bei welchem die gegebene Anzahl Jahre zu dem gegebenen Datum addiert wurde. Negative Werte für *years* subtrahieren Jahre. Falls *date* oder *years* NIL sind, oder das resultierende Datum ungültig ist, so wird NIL geliefert.

ADDEAR dekrementiert das Tagfeld um 1, falls *date* den 29. Februar repräsentiert und das resultierende Jahr kein Schaltjahr ist.

Beispiele: '(ADDEAR 29.02.2004 1)' ergibt 28.02.2005, '(ADDMONTH 04.02.2004 -1962)' ergibt 04.02.0042.

Siehe auch ADDMONTH, 2add,+.

16.14.9 TODAY

TODAY liefert das heutige Datum.

(TODAY)

Liefert das heutige Datum als Datumswert.

Siehe auch NOW.

16.14.10 NOW

NOW liefert die aktuelle Uhrzeit.

(NOW)

Liefert die aktuelle Uhrzeit als Zeitwert.

Siehe auch TODAY.

16.15 Listenfunktionen

Dieser Abschnitt listet Funktionen zum Verarbeiten von Listen auf.

16.15.1 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.

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 16.20.12 [SELECT], Seite 146). 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.

16.15.2 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.

16.15.3 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.

16.15.4 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.

16.15.5 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.

16.15.6 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.

16.15.7 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.

16.15.8 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.

16.15.9 REVERSE

REVERSE kehrt eine Liste um.

(REVERSE *list*)

liefert die umgekehrte Liste.

Beispiel: ‘(REVERSE (list 1 2 3))’ liefert (3 2 1).

16.15.10 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.

Beispiele

Ausdruck	Wert
(MAPFIRST 1+ (LIST 1 2 3))	(2 3 4)
(MAPFIRST + (LIST 1 2 3) (LIST 2 3))	(3 5 NIL)

16.15.11 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.

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.

16.15.12 SORTLISTGT

SORTLISTGT sortiert die Elemente einer Liste.

```
(SORTLISTGT gfunc list)
```

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.

16.16 Benutzereingabefunktionen

Zum Abfragen von Benutzereingaben können folgende Funktionen verwendet werden:

16.16.1 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* (Boolesch) 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.

16.16.2 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 wie in ASKFILE (siehe Abschnitt 16.16.1 [ASKFILE], Seite 128).

ASKDIR liefert den eingegebenen Verzeichnisnamen als Zeichenkette oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Siehe auch ASKFILE, ASKSTR.

16.16.3 ASKSTR

ASKSTR fragt den Benutzer nach einer Zeichenkette.

(ASKSTR *title oktext default maxlen [secret]*)

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. Wird *secret* angegeben und ist nicht NIL, so wird die eingegebene Zeichnkette unsichtbar gemacht, indem ein Aufzählungszeichen für jedes Zeichen der Zeichenkette angezeigt wird.

ASKSTR liefert die eingegebene Zeichenkette oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Siehe auch ASKFILE, ASKDIR, ASKCHOICESTR, ASKINT.

16.16.4 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 einen unbeschränkten Vorgabebereich.

ASKINT liefert die eingegebene Ganzzahl oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Siehe auch ASKSTR.

16.16.5 ASKCHOICE

ASKCHOICE fragt den Benutzer nach einer Auswahl aus mehreren Elementen.

```
(ASKCHOICE title oktext choices default [titles])
```

Öffnet ein Eingabefenster, das den Benutzer nach einem Element aus einer Liste von Elementen fragt. Der Fenstertitel und der Text des 'Ok'-Knopfes können mit *title* und *oktext* (Zeichenketten oder NIL für Vorgabewerte) gesetzt werden. In *choices* wird eine Liste der Auswahllemente angegeben. Es kann ein mehr-spaltiges Format verwendet werden, indem jedes Auswahllement als eine Liste von Unterelementen angegeben wird. Eine Vorauswahl kann in *default* als Index in die Liste der Auswahllemente (beginnend mit 0 für das erste Element) getroffen werden. Für keine Vorauswahl wird hier NIL angegeben. Wird das optionale Argument *titles* angegeben und ist nicht NIL, so wird ein Listenkopf mit dem Inhalt von *titles* angezeigt. Für ein mehr-spaltiges Format kann *titles* als eine Liste von Spaltenüberschriften angegeben werden.

Sowohl *choices* als auch *titles* können durch einen mehrzeiligen Text und eine Zeichenkette (anstatt von Listen) angegeben werden. In diesem Fall werden diese automatisch durch Aufrufen von (MEMOTOLIST *choices* TRUE) (siehe Abschnitt 16.13.3 [MEMOTOLIST], Seite 121), bzw. (STRTOLIST *titles*) (siehe Abschnitt 16.12.23 [STRTOLIST], Seite 115) in Listen umgewandelt.

ASKCHOICE liefert den Index des gewählten Elements oder NIL, wenn der Benutzer das Fenster mit Abbrechen verlassen hat.

Beispiele

```
(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)))
    )
  )
)
```

Nehmen wir an, Sie wollen den Benutzer einen bestimmten Datensatz einer Tabelle auswählen lassen. Die Tabelle soll 'Artikel' heißen und die Felder 'Name', 'Nummer', and

‘Preis’ enthalten. Das folgende Programm-Fragment zeigt, wie ASKCHOICE dazu verwendet werden kann, um einen Datensatz mit Preis größer 10 und nach Name geordnet auszuwählen:

```
(LET ((query (SELECT Artikel, Name, Nummer, Preis from Artikel
  WHERE (> Preis 10) ORDER BY Name))
  (recs (MAPFIRST FIRST (REST query))) ; Datensatz-Zeiger
  (items (MAPFIRST REST (REST query))) ; Auswahl
  (titles (REST (FIRST query))) ; Überschriften
  (index (ASKCHOICE "Auswahl" "Ok" items NIL titles))
  (rec (NTH index recs)))
  ; jetzt enthält rec den ausgewählten Datensatz (oder NIL bei Abbruch)
)
```

Siehe auch ASKCHOICESTR, ASKOPTIONS.

16.16.6 ASKCHOICESTR

ASKCHOICESTR fragt den Benutzer nach einer Zeichenkette und bietet vorgegebene an.

```
(ASKCHOICESTR title oktext strings default [titles])
```

Öffnet ein Eingabefenster, das dem Benutzer erlaubt, eine Zeichenkette aus mehreren auszuwählen oder eine beliebige Zeichenkette im separaten Zeichenkettenfeld einzugeben. Der Fenstertitel und der Text des ‘Ok’-Knopfes können mit *title* und *oktext* (Zeichenketten oder NIL für Vorgabewerte) gesetzt werden. In *strings* wird eine Liste der Auswahlelemente angegeben. Es kann ein mehr-spaltiges Format verwendet werden, indem jedes Auswahlelement als eine Liste von Unterelementen angegeben wird. Der Vorgabewert des Zeichenkettenfeldes kann mit *default* (Zeichenkette oder NIL für ein leeres Feld) gesetzt werden. Wird das optionale Argument *titles* angegeben und ist nicht NIL, so wird ein Listenkopf mit dem Inhalt von *titles* angezeigt. Für ein mehr-spaltiges Format kann *titles* als eine Liste von Spaltenüberschriften angegeben werden.

Sowohl *strings* als auch *titles* können durch einen mehrzeiligen Text und eine Zeichenkette (anstatt von Listen) angegeben werden. In diesem Fall werden diese automatisch durch Aufrufen von (MEMOTOLIST *strings* TRUE) (siehe Abschnitt 16.13.3 [MEMOTOLIST], Seite 121), bzw. (STRTOLIST *titles*) (siehe Abschnitt 16.12.23 [STRTOLIST], Seite 115) in Listen umgewandelt.

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.

16.16.7 ASKOPTIONS

ASKOPTIONS erlaubt dem Benutzer mehreren Optionen aus einer Liste auszuwählen.

```
(ASKOPTIONS title oktext options selected
[titles])
```

Öffnet ein Eingabefenster, das dem Benutzer erlaubt, mehrere Optionen aus einer Liste auszuwählen. Der Fenstertitel und der Text des 'Ok'-Knopfes können mit *title* und *oktext* (Zeichenketten oder NIL für Vorgabewerte) gesetzt werden. In *options* wird eine Liste der Optionen angegeben. Es kann ein mehr-spaltiges Format verwendet werden, indem jede Option als eine Liste von Unterelementen angegeben wird. Die Vorgabeauswahl der Optionen lässt sich in *selected* festlegen, welche eine Liste von Index-Zahlen enthält, die jeweils den entsprechenden Eintrag in *options* angibt, die vorab ausgewählt werden soll. Für keine Vorauswahl wird NIL anstatt einer Liste von Index-Zahlen angegeben. Wird das optionale Argument *titles* angegeben und ist nicht NIL, so wird ein Listenkopf mit dem Inhalt von *titles* angezeigt. Für ein mehr-spaltiges Format kann *titles* als eine Liste von Spaltenüberschriften angegeben werden.

Sowohl *options* als auch *titles* können durch einen mehrzeiligen Text und eine Zeichenkette (anstatt von Listen) angegeben werden. In diesem Fall werden diese automatisch durch Aufrufen von (MEMOTOLIST *options* TRUE) (siehe Abschnitt 16.13.3 [MEMOTOLIST], Seite 121), bzw. (STRTOLIST *titles*) (siehe Abschnitt 16.12.23 [STRTOLIST], Seite 115) in Listen umgewandelt.

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 Entwickler und Übersetzer die gleiche Musik von Faithless :-)

16.16.8 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 *canceltext* 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 Computer"))) 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" "Ich finde MUIbase spitze!" NIL NIL)
```

Siehe auch ASKCHOICE.

16.16.9 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 title "String" initial [help [secret]]) zum Bearbeiten einer Textzeile,
(LIST title "Memo" initial [help])           zum Bearbeiten von mehrzeiligen Texten,
(LIST title "Integer" initial [help])        zum Bearbeiten einer Ganzzahl,
(LIST title "Real" initial [help])           zum Bearbeiten einer Fließkommazahl,
(LIST title "Date" initial [help])           zum Bearbeiten eines Datums,
(LIST title "Time" initial [help])          zum Bearbeiten einer Zeit,
(LIST title "Bool" initial [help])          zum Bearbeiten eines Booleschen Wertes,
(LIST title "Choice" initial
  (LIST choice ...) [help]
)
) für ein Auswahlfeld.
(LIST title "ChoiceList" initial
  (LIST choice ...) [help]
```

```

)                               zum Auswählen eines Elements
                               aus einer Liste.

(LIST title "Options" initial
  (LIST choice ...) [help]
)                               zum Auswählen mehrerer Elemente
                               aus einer Liste.

non-list-expr                  für statischen Text

```

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 vorbelegt sein sollen. Das optionale Hilfsfeld (eine Zeichenkette) kann verwendet werden, um dem Benutzer mehr Informationen über die Verwendung des Feldes mitzugeben. Für Zeichenkettenfelder kann ein weitere Parameter `'secret'` angegeben werden. Ist dieser nicht NIL, so wird die eingegebene Zeichnkette unsichtbar gemacht, indem ein Aufzählungszeichen für jedes Zeichen der Zeichenkette angezeigt wird.

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.

16.17 E/A-Funktionen

Dieser Abschnitt listet die Funktionen und Variablen zur Dateiein- und ausgabe (z.B. drucken) auf.

16.17.1 FOPEN

FOPEN öffnet eine Datei zum Lesen/Schreiben.

(FOPEN *filename mode [encoding]*)

Öffnet eine Datei mit dem Dateinamen *filename* (Zeichenkette). Das Argument *mode* (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.

Der optionale Parameter *encoding* gibt die Textkodierung der Datei an und kann eine der folgenden Zeichenketten sein:

`"none"`: Keine Interpretation von Zeichen wird vorgenommen. Dies ist für Binärdateien.

`"UTF-8"`: Text ist in UTF-8 kodiert. Lesen und Schreiben wandelt von/in UTF-8 um.

`"locale"`:

Text ist in der Systemlokalisierung kodiert. Auf Windows ist dies die System-Codepage. Unter Mac OS und Linux ist dies die Einstellung in den LANG- und LC_*-Umgebungsvariablen (siehe man locale). Beim Amiga ist es die voreingestellte 8-Bit-Kodierung.

`"8-bit"`: Text ist in der voreingestellten 8-Bit-Kodierung abgelegt. Auf Windows, Mac OS und Linux ist dies die ISO-8859-1-Kodierung (Latin 1). Auf dem Amiga ist es die im System voreingestellte 8-Bit-Kodierung (gleich wie `'locale'`).

`"auto"`: Die Kodierung wird automatisch ermittelt. Ist die Datei lesbar, so wird die Kodierung wie folgt ermittelt: Ist der gesamte Inhalt UTF-8 konform, so wird `"UTF-8"` verwendet. Sonst, wenn die Systemlokalisierung nicht UTF-8 ist, so wird `"locale"` angenommen. Ansonsten wird `"8-bit"` verwendet. Beim Schreiben wird, sofern die Kodierung noch nicht festgelegt wurde, zuerst versucht in der Systemlokalisierung zu schreiben. Sofern es keine Umwandlungsfehler gab, wird `"locale"` verwendet, ansonsten wird `"UTF-8"` verwendet.

Wird kein *encoding*-Parameter angegeben, so wird `"auto"` verwendet.

FOPEN liefert bei Erfolg einen Dateihandler. Schlug er fehl, wird NIL geliefert. Sind *filename*, *mode* oder *encoding* NIL, dann wird NIL zurückgeliefert.

Beispiele

`(FOPEN "index.html" "w" "utf-8")` öffnet und liefert einen Dateihandler zum Schreiben der Datei `'index.html'`, welche in UTF-8 kodiert wird.

`(FOPEN "output.txt" "a+")` öffnet die Datei `'output.txt'` zum Anhängen und verwendet die in der Datei bereits vorhandene Textkodierung. Zu beachten ist, dass wenn nur `"a"` als Mode verwendet wird, MUIbase die Datei möglicherweise nicht lesen kann und

von daher die vorhandene Textkodierung nicht bestimmen kann. In diesem Fall wird die Kodierung beim Schreiben festgelegt (und ist möglicherweise eine andere als die bereits bestehende).

Siehe auch FCLOSE, stdout, FFLUSH.

16.17.2 FCLOSE

FCLOSE schließt eine Datei.

(FCLOSE *file*)

Schließt die gegebene Datei und liefert 0 bei Erfolg oder NIL, wenn ein Fehler auftrat. Ist *file* NIL, dann wird 0 geliefert (kein 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.

16.17.3 stdout

Die globale Variable `stdout` trägt den Dateihandler zur Standardausgabe von MUIbase. Der Ausgabedateinamen kann im Menüpunkt ‘Programm - Ausgabedatei’ gesetzt werden (siehe Abschnitt 7.2.12 [Programm-Ausgabedatei], Seite 40).

Die Datei wird beim ersten Zugriff auf diese Variable (z.B. durch den Aufruf von ‘(FPRINTF stdout ...)’ oder durch den Aufruf von ‘(PRINTF ...)’) geöffnet. Die Datei wird nicht vor der Programmausführung geöffnet. Dies verhindert das Öffnen der Datei, wenn keine Ausgabe erzeugt wird, z.B. wenn nur Berechnungen und Änderungen an einigen Datensätzen durchgeführt werden sollen.

Beim Öffnen der Datei wird als Mode-Parameter entweder “w” oder “a+” in Abhängigkeit der ‘Anhängen’-Einstellung in Menüpunkt ‘Programm - Ausgabedatei’ verwendet. Als Kodierung wird “auto” eingesetzt.

Wenn MUIbase die Programmausgabedatei nicht öffnen kann, dann wird die Ausführung unterbrochen und eine Fehlermeldung ausgegeben.

Siehe auch FOPEN, PRINTF.

16.17.4 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.

16.17.5 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 16.12.34 [SPRINTF], Seite 118).

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`.

16.17.6 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 16.12.34 [SPRINTF], Seite 118).

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.

16.17.7 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.

16.17.8 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.

16.17.9 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.

Zu beachten ist, dass nach einer Lese-Operation, der Aufruf von FSEEK mit *whence* als SEEK_CUR nur für die Zeichenkodierung "none" unterstützt wird.

Siehe auch FTELL, FOPEN, 2Pre-defined constants, Vordefinierte Konstanten.

16.17.10 FTELL

FTELL liefert die Schreib-/Lese-Position der Datei.

(FTELL *file*)

Ermittelt die aktuelle Schreib-/Lese-Position relativ zum Anfang der gegebenen Datei und liefert sie als Ganzzahl. Tritt ein Fehler auf oder ist 'file' NIL, dann wird NIL geliefert.

Zu beachten ist, dass nach einer Lese-Operation, der Aufruf von FTELL nur für die Zeichenkodierung "none" unterstützt wird.

Siehe auch FSEEK, FOPEN, FEOF.

16.17.11 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.

16.17.12 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.

16.17.13 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.

16.17.14 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.

16.17.15 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*

file NIL, dann passiert nichts. Liefert *str* oder NIL, wenn ein Ausgabefehler auftrat.

Siehe auch FPUTSTR, FGETCHAR.

16.17.16 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.

16.17.17 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.

16.17.18 FFLUSH

FFLUSH leert den Schreibpuffer in eine Datei.

(FFLUSH *file*)

Schreibt den Schreibpuffer der gegebenen Datei. Liefert 0 bei Erfolg, NIL bei einem Fehler. Ist *file* NIL, dann wird 0 zurückgegeben (kein Fehler).

Siehe auch FOPEN, FCLOSE.

16.18 Datensatzfunktionen

Dieser Abschnitt behandelt Funktionen für Datensätze.

16.18.1 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 [Tabellen], 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.

Siehe auch NEW*, DELETE, 2Tables, Tabellen.

16.18.2 NEW*

NEW* ist die Version von NEW (siehe Abschnitt 16.18.1 [NEW], Seite 139) mit dem Stern.

```
(NEW* table init)
```

NEW* prüft, ob eine Auslösefunktion für die gegebene Tabelle (siehe Abschnitt 16.29.8 [Auslösefunktion Neu], Seite 160) 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*.

16.18.3 DELETE

DELETE löscht einen Datensatz einer Tabelle.

```
(DELETE table confirm)
```

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 [Tabellen], Seite 20) der gegebenen Tabelle auf NIL.

Beispiel: '(DELETE table NIL)' löscht stillschweigend den aktuellen Datensatz der gegebenen Tabelle.

Siehe auch DELETE*, DELETEALL, NEW, 2Tables, Tabellen.

16.18.4 DELETE*

DELETE* ist die Version von DELETE (siehe Abschnitt 16.18.3 [DELETE], Seite 139) mit dem Stern.

```
(DELETE* table confirm)
```

DELETE* prüft, ob eine Auslösefunktion für die gegebene Tabelle (siehe Abschnitt 16.29.9 [Auslösefunktion Löschen], Seite 161) definiert wurde. Ist eine vorhanden, dann wird diese zum Löschen des Datensatzes ausgeführt und dessen Ergebnis zurückgeliefert. Das Argument *confirm* 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*.

16.18.5 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 dir 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, 2Tables, Tabellen.

16.18.6 GETMATCHFILTER

GETMATCHFILTER liefert den Status der Filterübereinstimmung eines Datensatzes.

```
(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.

16.18.7 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.

16.18.8 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, 2Comparison function, Vergleichsfunktion.

16.18.9 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 16.20.4 [REORDER], Seite 144) 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 16.29.10 [Vergleichsfunktion], Seite 161.

Siehe auch GETISSORTED, SETMATCHFILTER, REORDER, GETORDERSTR, SETORDERSTR, 2Comparison function, Vergleichsfunktion.

16.18.10 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, Zeichketten 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.

16.18.11 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.

16.19 Feldfunktionen

Dieser Abschnitt behandelt Funktionen für Felder einer Tabelle.

16.19.1 ATTRNAME

ATTRNAME liefert den Namen des Feldes.

(ATTRNAME *attr*)

Liefert eine Zeichenkette mit dem Namen des angegebenen Feldes.

Siehe auch TABLENAME

16.19.2 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.

16.19.3 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 Abschnitt 15.2.2 [Typabhängige Einstellungen], Seite 69) eingegebenen Texte zurückgegeben, Bei Zeichenkettenfeldern werden die statischen Auswahltexte zurückgegeben, die für das Listenansicht-Popup (siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75) eingegeben wurden (zu beachten ist, dass diese Funktion nur für statische Auswahltexte sinnvoll ist).

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 16.13.3 [MEMOTOLIST], Seite 121) in eine Liste umwandeln.

Siehe auch SETLABELS.

16.19.4 SETLABELS

SETLABELS wird verwendet, um die Auswahltexte eines Zeichenkettenfeldes zu setzen.

(SETLABELS *attr str*)

Setzt die statischen Auswahltexte des Zeichenkettenfeldes *attr* auf die Auswahlfelder, die im Parameter *str* aufgelistet sind. Der Parameter *str* enthält für jeden Auswahltext eine Zeile. Die Auswahltexte ersetzen diejenigen, die in dem Listenansicht-Popup des Feldobjekteditors (siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75) eingegeben wurden. Zu beachten ist, dass diese Funktion nur für statische Auswahltexte sinnvoll ist.

SETLABELS liefert den Wert des Parameters *str*.

Beispiel: '(SETLABELS Table.String "Mein Haus\nist\ndein Haus")' setzt die statischen Listenansicht-Auswahltexte des gegebenen Zeichenkettenfeldes auf 'Mein Haus', 'ist' und 'dein Haus'.

Hinweis: Man kann eine Liste von Auswahltexten durch den Aufruf von LISTTOMEMO in das benötigte Zeichenkettenformat umwandeln.

Siehe auch GETLABELS.

16.20 Tabellenfunktionen

16.20.1 TABLENAME

TABLENAME liefert den Namen einer Tabelle.

(TABLENAME *table*)

Liefert eine Zeichenkette, die den Namen der angegebenen Tabelle enthält.

Siehe auch ATTRNAME

16.20.2 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, 2Comparison function, Vergleichsfunktion.

16.20.3 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, andernfalls 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 16.19.1 [ATTRNAME], Seite 142) 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, 2Comparison function, Vergleichsfunktion.

16.20.4 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 16.29.10 [Vergleichsfunktion], Seite 161.

Siehe auch REORDERALL, GETORDERSTR, SETORDERSTR, GETISSORTED, SETISSORTED, Order, 2Comparison function, Vergleichsfunktion.

16.20.5 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, 2Order, Sortieren, 2Comparison function, Vergleichsfunktion.

16.20.6 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.

16.20.7 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.

16.20.8 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.

16.20.9 SETFILTERSTR

SETFILTERSTR setzt den Datensatzfilterausdruck einer Tabelle.

(SETFILTERSTR *table filter-str*)

Setzt den Datensatzfilterausdruck der gegebenen Tabelle auf den Ausdruck im Parameter *filter-str* (hierbei wird der Ausdruck als Zeichenkette angegeben, also nicht der Ausdruck

selbst!). 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:

```
(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.

16.20.10 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.

16.20.11 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.

Es ist darauf zu achten, dass Datensatznummern bei 1 beginnen und die Datensatznummer 0 für den Vorgabedatensatz verwendet wird.

Siehe auch RECORDS, RECNUM.

16.20.12 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 Bezeichner (Identifier) angegeben werden. 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.

Funktionsweise

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.

Beispiele

Siehe Abschnitt 14.6 [Abfragebeispiele], Seite 64, für einige Beispiele mit der Funktion SELECT.

Siehe auch FOR ALL.

16.21 Oberflächenfunktionen

Dieser Abschnitt beschreibt die Funktionen zum Verändern von Benutzeroberflächenelementen.

16.21.1 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.

Siehe auch SETVIRTUALLISTACTIVE.

16.21.2 SETBGPEN

SETBGPEN setzt den Hintergrundstift eines GUI-Elements.

(SETBGPEN *attr pen*)

Setzt den Stift zum Zeichnen des Hintergrunds des durch *attr* gegebenen GUI-Elements. Für *pen* kann eine hexadezimaler Ganzzahl, welche eine Farbe im RGB-Format (rot, grün, blau) spezifiziert, oder eine der PEN_*-Konstanten verwendet werden. Ist *pen* NIL dann wird ein Standardhintergrund gesetzt.

SETBGPEN gibt den Wert des neuen Hintergrundstiftes zurück.

Beispiel: '(SETBGPEN Control.Status 0xFF0000)' setzt einen roten Hintergrund für das GUI-Element des Feldes 'Status' in der Tabelle 'Table'. Der selbe Effekt kann durch Verwenden von '(SETBGPEN Control.Status PEN_RED)' erreicht werden.

Siehe auch Pre-defined constants.

16.21.3 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.

16.21.4 SETWINDOWOPEN

SETWINDOWOPEN öffnet und schließt ein Fenster.

(SETWINDOWOPEN *attr-or-table open*)

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.

16.21.5 GETVIRTUALLISTACTIVE

GETVIRTUALLISTACTIVE liefert den Index der aktiven Zeile eines virtuellen Feldes, welches für die Anzeige die 'Listen'-Art verwendet.

(GETVIRTUALLISTACTIVE *virtual-attr*)

Liefert den logischen Index (beginnend mit 1) der gerade aktuellen Zeile des gegebenen virtuellen Feldes. Der logische Index ist die Zeile in der ursprünglichen Liste, die beim Setzen des virtuellen Attributs verwendet wurde. Diese kann von der angezeigten Ordnung abweichen, falls der Benutzer die Liste umsortiert hat, z.B. durch Klicken auf einen Spaltentitel. Falls das Bedienelement von *virtual-attr* nicht sichtbar ist oder nicht die 'Listen'-Art für die Anzeige verwendet oder falls keine Zeile aktiv ist, so wird NIL zurückgegeben.

Siehe auch SETVIRTUALLISTACTIVE.

16.21.6 SETVIRTUALLISTACTIVE

SETVIRTUALLISTACTIVE setzt die aktive Zeile eines virtuellen Feldes, welche die ‘Listen’-Art für die Anzeige verwendet.

(SETVIRTUALLISTACTIVE *virtual-attr num*)

Setzt die aktive Zeile des gegebenen virtuellen Feldes auf die *num*-te logische Zeile (begannt mit 1). Der logische Zeile ist die Zeile in der ursprünglichen Liste, die beim Setzen des virtuellen Attributs verwendet wurde, und kann von der angezeigten Ordnung abweichen, falls der Benutzer die Liste umsortiert hat, z.B. durch Klicken auf einen Spaltentitel.

Liefert *num* oder NIL, falls das Bedienelement von *virtual-attr* nicht sichtbar ist oder nicht die ‘Listen’-Art für die Anzeige verwendet oder falls *num* außerhalb des gültigen Bereiches ist (kleiner als 1 oder größer der Anzahl Zeilen).

SETVIRTUALLISTACTIVE setzt nicht den Cursor auf das Benutzerelement des Feldes. Dies kann mit SETCURSOR (siehe Abschnitt 16.21.1 [SETCURSOR], Seite 147) erreicht werden.

Siehe auch GETVIRTUALLISTACTIVE, SETCURSOR.

16.22 Projektfunktionen

Dieser Abschnitt listet Funktionen auf, die mit Projekten zu tun haben.

16.22.1 PROJECTNAME

PROJECTNAME liefert den Projektnamen.

(PROJECTNAME)

PROJECTNAME liefert den Namen des aktuellen Projekts als Zeichenkette oder NIL, wenn noch kein Name definiert wurde. Der Projektname ist der Pfadname des Projektverzeichnisses im Dateisystem.

Siehe auch CHANGES.

16.22.2 PREPARECHANGE

PREPARECHANGE bereitet das Projekt für eine Änderung vor.

(PREPARECHANGE)

Dieser Befehl erwirbt einen Änderungszugriff auf das Projekt. Dies ist nützlich, wenn auf ein Projekt mit mehreren MUIbase-Instanzen zugegriffen wird, welche möglicherweise auf verschiedenen Rechnern laufen. Wurde ein Änderungszugriff erteilt, so kann keine andere MUIbase-Instanz diesen ebenfalls erhalten, bis der Zugriff wieder freigegeben wird. Der Änderungszugriff wird freigegeben, wenn das Projektprogramm endet ohne tatsächlich eine Änderung vorzunehmen, oder wenn das Projekt gespeichert wird. Für weitere Informationen über den geteilten Zugriff auf ein Projekt, siehe Abschnitt 6.1 [Dateiformat], Seite 31.

Bei Erfolg liefert PREPARECHANGE NIL zurück. Schlägt der Erwerb des Änderungszugriffs fehl, so wird das Programm beendet und eine entsprechende Fehlermeldung generiert.

Siehe auch CHANGES.

16.22.3 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 PREPARECHANGE, PROJECTNAME.

16.22.4 GETADMINMODE

GETADMINMODE gibt an, ob sich das aktuelle Projekt im Admin- oder Benutzermodus befindet.

(GETADMINMODE)

Liefert TRUE, falls sich das aktuelle Projekt im Admin-Modus befindet, NIL andernfalls.

Siehe auch SETADMINMODE, ADMINPASSWORD, onAdminMode.

16.22.5 SETADMINMODE

SETADMINMODE versetzt das aktuelle Projekt in den Admin- oder Benutzermodus.

(SETADMINMODE *admin*)

Ist *admin* NIL, so wird das aktuelle Projekt in den Benutzermodus, andernfalls in den Admin-Modus versetzt. Zu bemerken ist, dass keine Passwortabfrage stattfindet, wenn ein Projekt über diese Funktion vom Benutzer- in der Admin-Modus versetzt wird.

Liefert TRUE, falls das Projekt in den Admin-Modus versetzt wurde, oder NIL falls es in den Benutzermodus versetzt wurde.

Siehe auch GETADMINMODE, ADMINPASSWORD, onAdminMode, Beispielprojekt `Users.mb`.

16.22.6 ADMINPASSWORD

ADMINPASSWORD liefert das Admin-Passwort als SHA1-Hash.

(ADMINPASSWORD)

Liefert eine Zeichenkette, welche den SHA1-Hash des Admin-Passworts des aktuellen Projekts enthält. Wurde kein Admin-Passwort gesetzt, so wird NIL zurückgegeben.

Siehe auch GETADMINMODE, SETADMINMODE, SHA1SUM, Beispielprojekt `Users.mb`.

16.23 Systemfunktionen

Dieser Abschnitt listet Funktionen auf, die auf das Betriebssystem zugreifen.

16.23.1 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 7.1.2 [Externer Editor], Seite 35) eingestellt werden. EDIT startet den externen

Editor synchron, das bedeutet, die Funktion wartet bis der Benutzer den Editor beendet hat.

EDIT gibt den Rückgabewert des externen Editors als Ganzzahl zurück.

Siehe auch EDIT*, VIEW, SYSTEM.

16.23.2 EDIT*

EDIT* ist die Stern-Version von EDIT und hat den selben Effekt wie EDIT (siehe Abschnitt 16.23.1 [EDIT], Seite 150). Der einzige Unterschied ist, dass EDIT* den externen Editor asynchron startet und sofort wieder zurückkehrt.

EDIT* liefert 0, wenn der Editor erfolgreich gestartet werden konnte, anderenfalls wird eine Ganzzahl ungleich 0 zurückgegeben, welche einen system-abhängigen Fehler beschreibt.

Siehe auch EDIT, VIEW*, SYSTEM*.

16.23.3 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 7.1.3 [Externer Anzeiger], Seite 36) eingestellt werden. VIEW startet den externen Anzeiger synchron, das bedeutet, die Funktion wartet bis der Benutzer den Anzeiger beendet hat. Zu beachten ist aber, dass auf manchen Systemen sofort zurückgekehrt wird, falls bereits eine Instanz des Anzeigers gestartet wurde.

VIEW gibt den Rückgabewert des externen Anzeigers als Ganzzahl zurück.

Siehe auch VIEW*, EDIT, SYSTEM.

16.23.4 VIEW*

VIEW* ist die Stern-Version von VIEW und hat den selben Effekt wie VIEW (siehe Abschnitt 16.23.3 [VIEW], Seite 151). Der einzige Unterschied ist, dass VIEW* den externen Anzeiger asynchron startet und sofort wieder zurückkehrt.

VIEW* liefert 0, wenn der Anzeiger erfolgreich gestartet werden konnte, anderenfalls wird eine Ganzzahl ungleich 0 zurückgegeben, welche einen system-abhängigen Fehler beschreibt.

Siehe auch VIEW, EDIT*, SYSTEM*.

16.23.5 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 16.12.34 [SPRINTF], Seite 118) erzeugt. Die Befehlszeile wird mit einem Kommandozeilen-Interpreter ausgeführt (ShellExecute auf Windows, /bin/sh auf Mac OS und Linux, User-Shell auf Amiga). SYSTEM wartet, bis das aufgerufene Programm beendet wurde.

SYSTEM gibt den Rückgabewert des ausgeführten Befehls als Ganzzahl zurück.

Siehe auch EDIT, VIEW, SYSTEM*.

16.23.6 SYSTEM*

SYSTEM* ist die Stern-Version von **SYSTEM** und hat den selben Effekt wie **SYSTEM** (siehe Abschnitt 16.23.5 [SYSTEM], Seite 151). Der einzige Unterschied ist, dass **SYSTEM*** die Befehlszeile asynchron ausführt und sofort wieder zurückkehrt.

SYSTEM* liefert 0, wenn das Ausführen der Befehlszeile erfolgreich gestartet werden konnte, anderenfalls wird eine Ganzzahl ungleich 0 zurückgegeben, welche einen system-abhängigen Fehler beschreibt.

Siehe auch **SYSTEM**, **EDIT***, **VIEW***.

16.23.7 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.

16.23.8 TACKON

TACKON erzeugt einen Pfadnamen.

(**TACKON** *dirname* [*component* ...])

Verknüpft *dirname* und alle Komponenten in [*component* ...] zu einem Pfadnamen. **TACKON** weiß, wie es mit speziellen Zeichen, die als Pfadseparator dienen, am Ende jedes Arguments umzugehen hat. Es liefert den Pfadnamen als Zeichenkette oder **NIL**, falls eines der Argumente **NIL** ist. Zu beachten ist, dass **TACKON** nicht untersucht, ob der resultierende Pfad auch tatsächlich zu einer Datei oder einem Verzeichnis im Dateisystem gehört.

Beispiel: '(**TACKON** "Sys:System" "CLI")' liefert "Sys:System/CLI".

Siehe auch **FILENAME**, **DIRNAME**.

16.23.9 FILENAME

FILENAME extrahiert den Dateinamen aus einem Pfadnamen.

(**FILENAME** *path*)

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**.

16.23.10 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*.

16.23.11 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 16.12.34 [*SPRINTF*], Seite 118) erzeugt.

MESSAGE liefert die formatierte Titelzeichenkette.

Beispiel: '(*MESSAGE* "6 * 7 = %i" (* 6 7))'.

Siehe auch *PRINT*, *PRINTF*.

16.23.12 COMPLETEMAX

COMPLETEMAX setzt die maximale Anzahl Schritte der Fortschrittsanzeige.

```
(COMPLETEMAX steps)
```

Setzt die maximale Anzahl Schritte für die Anzeige des Fortschritts eines MUIbase-Programms. Voreingestellt sind 100 Schritte (falls diese Funktion nicht aufgerufen wird). Das Argument *steps* muss eine Ganzzahl sein. Ist *steps* *NIL* oder 0 so wird keine Fortschrittsleiste angezeigt. Die Fortschrittsleiste ist Teil des Pause/Abbrechen-Fensters, welches bei Ausführen eines MUIbase-Programms nach einer kurzen Verzögerung erscheint.

COMPLETEMAX gibt das Argument *steps* zurück.

Siehe auch *COMPLETEADD*, *COMPLETE*.

16.23.13 COMPLETEADD

COMPLETEADD erhöht die Fortschrittsanzeige.

```
(COMPLETEADD add)
```

Addiert die Ganzzahl *add* auf den aktuellen Fortschrittswert. Zu Anfang ist der Fortschrittswert 0. Ist *add* *NIL*, so wird der Fortschrittswert auf 0 zurückgesetzt und die Fortschrittsanzeige unterdrückt.

COMPLETEADD gibt das Argument *add* zurück.

Beispiel:

```
(SETQ num ...)
(COMPLETEMAX num)
(DOTIMES (i num)
  (COMPLETEADD 1)
)
```

Siehe auch *COMPLETEMAX*, *COMPLETE*.

16.23.14 COMPLETE

COMPLETE setzt die Fortschrittsanzeige.

```
(COMPLETE cur)
```

Setzt den aktuellen Fortschrittswert auf die Ganzzahl *cur*. Ist *cur* NIL oder 0, so wird keine Fortschrittsleiste angezeigt.

COMPLETE gibt das Argument *cur* zurück.

Beispiel:

```
(COMPLETE 10)
...
(COMPLETE 50)
...
(COMPLETE 100)
```

Siehe auch COMPLETEMAX, COMPLETEADD.

16.23.15 GC

GC erzwingt das Aufräumen des Speichers.

```
(GC)
```

Erzwingt das Aufräumen des Speichers und liefert NIL. Im Normalfall wird das Aufräumen automatisch von Zeit zu Zeit durchgeführt.

16.23.16 PUBSCREEN

PUBSCREEN liefert den Namen des Public-Screens.

```
(PUBSCREEN)
```

Auf dem Amiga gibt PUBSCREEN den Namen des Public-Screens, auf dem MUIbase läuft, zurück oder NIL, falls der Screen nicht öffentlich ist.

Auf anderen Systemen liefert PUBSCREEN NIL zurück.

16.24 Vordefinierte Variablen

MUIbase kennt einige vordefinierte globale Variablen.

Momentan existiert nur eine einzige globale Variable: `stdout` (siehe Abschnitt 16.17.3 [stdout], Seite 135).

16.25 Vordefinierte Konstanten

Die folgenden vordefinierten Konstanten können in jedem Ausdruck bei der Programmierung verwendet werden:

Name	Typ	Wert	Bemerkung
NIL	jeder	NIL	
TRUE	Boolesch	TRUE	
RESET	Zeichenkette	"\33c"	
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	Kleinster Ganzzahlwert
HUGE_VAL	Fließkommazahl	1.797693e+308	Größte absolute Fließkommazahl
PI	Fließkommazahl	3.14159265359	
OSTYPE	Zeichenkette	<OS-Typ>	"Windows", "MacOSX", "Unix" oder "Amiga"
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
TMPDIR	Zeichenkette	Temporäres Verzeichnis abhängig vom System	
PEN_WHITE	Ganzzahl	0xFFFFFFFF	
PEN_BLACK	Ganzzahl	0x000000	
PEN_RED	Ganzzahl	0xFF0000	
PEN_GREEN	Ganzzahl	0x00FF00	
PEN_BLUE	Ganzzahl	0x0000FF	
PEN_CYAN	Ganzzahl	0x00FFFF	
PEN_MAGENTA	Ganzzahl	0xFF00FF	
PEN_YELLOW	Ganzzahl	0xFFFF00	

Siehe Abschnitt 16.4.7 [Konstanten], Seite 89, für weitere Informationen über Konstanten. Zum Definieren eigener Konstanten benutzt man die Preprozessor-Anweisung `#define` (siehe Abschnitt 16.3.1 [`#define`], Seite 84).

16.26 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 16.6.6 [FUNCALL], Seite 95) verwendet werden.

Beispiel:

```
(DEFUN map (l fun)           # Parameter: Liste und Funktion
  (LET (res)                # lokale Variable res, vorbelegt mit NIL
    (DOLIST (i l)           # jedes Element der Reihe nach
      (SETQ res
        (CONS (FUNCALL fun i) res)  # Funktion aufrufen 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`, `APPLY`, `MAPFIRST`.

16.27 Typdeklarierer

Es ist möglich, den Typ einer Variable durch Anfügen eines Typdeklarierers hinter dem Namen festzulegen. Die folgenden Typdeklarierer existieren:

Deklarierer Beschreibung

<code>:INT</code>	für Ganzzahlen
<code>:REAL</code>	für Fließkommazahlen
<code>:STR</code>	für Zeichenketten
<code>:MEMO</code>	für mehrzeilige Zeichenketten
<code>:DATE</code>	für Datumswerte
<code>:TIME</code>	für Zeitwerte
<code>:LIST</code>	für Listen
<code>:FILE</code>	für Dateihandler
<code>:FUNC</code>	für Funktionen jedes Typs
<code>:table</code>	für Datensatzzeiger auf <code>table</code>

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.

16.28 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 16.4.2 [Lisp-Aufbau], Seite 87, Abschnitt 16.9.1 [AND], Seite 104, Abschnitt 16.9.2 [OR], Seite 104, and Abschnitt 16.6.8 [IF], Seite 95.

([expr ...])

Errechnet *expr ...* und liefert den Wert des letzten Ausdrucks (siehe Abschnitt 16.6.1 [PROGN], Seite 93). 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 Beziehungfeldes zu ermitteln (der ein Datensatzzeiger ist) und verwendet diesen Datensatz, um den Wert von ‘Attribute’ zu erhalten.

var

Liefert den Inhalt der globalen oder lokalen Variable *var*. Globale Variablen können durch DEFVAR (siehe Abschnitt 16.5.3 [DEFVAR], Seite 92), lokale Variablen z.B. durch LET (siehe Abschnitt 16.6.3 [LET], Seite 93) definiert werden.

var.AttrPath

Verwendet den Datensatzzeiger von *var*, um den Wert des gegebenen Feldes zu ermitteln.

16.29 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.

16.29.1 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 `onReload`, `onClose`, `onAdminMode`, `onChange`, Beispielprojekt `Trigger.mb`.

16.29.2 onReload

Ähnlich zu `onOpen` ruft MUIbase die Funktion `onReload` auf, wenn das Projekt neu geladen wird, z.B. durch Auswahl des Menüpunkts ‘Projekt - Neu laden’. Die Funktion wird ohne Argumente aufgerufen.

Die Funktion `onReload` wird allerdings nur dann aufgerufen, wenn keine strukturellen Änderungen am Projekt vorliegen, d.h. keine Tabellen oder Felder wurden geändert, hinzugefügt, oder gelöscht, das Projektprogramm wurde nicht verändert und auch keine anderen Änderungen an der Projektstruktur wurden vorgenommen. In solch einem Falle wird ein Neuladen des Projekts als eine neue Öffnung des Projekts betrachtet und die Funktion `onOpen` aufgerufen.

Siehe auch `onOpen`, `onClose`, `onAdminMode`, `onChange`.

16.29.3 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`.

16.29.4 onAdminMode

Wird ein Projekt in den Admin- oder Benutzermodus versetzt und eine Funktion mit dem Namen `onAdminMode` existiert im Projektprogramm, so wird diese Funktion aufgerufen. Die Funktion erhält ein Argument `admin`, welches angibt, ob sich das aktuelle Projekt im Admin- (`admin` ist nicht NIL) oder Benutzermodus (`admin` ist NIL) befindet.

Beispiel

```
(DEFUN onAdminMode (admin)
  (IF admin
    (ASKBUTTON NIL "Im Admin-Modus" NIL NIL)
    (ASKBUTTON NIL "Im Benutzermodus" NIL NIL)
  )
)
```

Siehe auch `onOpen`, `onChange`, `SETADMINMODE`, Beispielprojekt `Users.mb`.

16.29.5 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`, `onAdminMode`, `logLabel`, Beispielprojekt `Trigger.mb`.

16.29.6 logLabel

Beim Erstellen eines neuen Logeintrags sucht MUIbase im Programm des Projekts nach einer Funktion mit dem Namen `logLabel`. Ist diese vorhanden, so wird sie ohne Argumen-

te aufgerufen. Der zurückgegebene Wert wird in eine Zeichenkette umgewandelt und für das ‘_Label’-Feld des neuen Logeintrags verwendet (siehe Abschnitt 8.3 [Logbeschriftung setzen], Seite 43).

Beispiel

```
(DEFUN logLabel ()
  Control.CurrentUser.Name
)
```

Das obige Beispiel wurde dem Projekt ‘Users.mb’ entnommen. Hier wird die Beschriftung auf den Namen des aktuellen Benutzers gesetzt. Dies bedeutet, dass alle Änderungen des aktuellen Benutzers mit seinem oder ihrem Namen beschriftet werden. Dadurch ist es später möglich, zu identifizieren, welcher Benutzer welche Änderungen vorgenommen hat.

Siehe auch onChange, Beispielprojekt Users.mb.

16.29.7 mainWindowTitle

Enthält das Programm eines Projekts eine Funktion mit dem Namen mainWindowTitle, so wird das Resultat dieser Funktion zum Setzen des Fenstertitels des Projekt-Hauptfensters verwendet. Die Funktion wird ohne Argumente aufgerufen und muss eine Zeichenkette zurückliefern. Der Fenstertitel wird automatisch Neuberechnet, sobald ein in der Funktion verwendetes Feld seinen Wert ändert.

Falls mainWindowTitle nicht vorhanden ist, so zeigt der Fenstertitel den Namen des Projekts an.

In jedem Fall markiert MUIbase ein Projekt, welches nicht gespeicherte Änderungen enthält, mit einem vorgestellten ‘*’-Zeichen im Fenstertitel.

Siehe auch Beispielprojekt Trigger.mb.

16.29.8 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 15.1.1 [Tabellen erstellen], Seite 67) 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 16.18.1 [NEW], Seite 139) 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 16.18.2 [NEW*], Seite 139) aufruft.

Beispiel einer Auslösefunktion Neu

```
(DEFUN newRecord (init)
  (PROG1 ; zum Rückgeben des Ergebnisses von NEW
```

```

        (NEW Table init)
        ...
    )
)

```

Siehe auch NEW, NEW*, 2Delete trigger, Auslösefunktion Löschen.

16.29.9 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 15.1.1 [Tabellen erstellen], Seite 67) 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 16.18.3 [DELETE], Seite 139) zum Löschen des Datensatzes aufrufen.

Die Auslösefunktion ‘Delete’ wird auch aufgerufen, wenn ein MUIbase-Programm die Funktion DELETE* (siehe Abschnitt 16.18.4 [DELETE*], Seite 140) aufruft.

Beispiel einer Auslösefunktion Löschen

```

(DEFUN deleteRecord (confirm)
  (DELETE Table confirm)
)

```

Siehe auch DELETE, DELETE*, 2New trigger, Auslösefunktion Neu.

16.29.10 Vergleichsfunktion

Um eine Sortierung von Datensätzen einer Tabelle zu definieren, kann eine Vergleichsfunktion verwendet werden. Siehe Abschnitt 11.4 [Sortierung ändern], Seite 53, für Informationen wie eine solche Funktion für eine Tabelle spezifiziert werden kann. Die Vergleichsfunktion erhält zwei Datensatzzeiger als Argumente und liefert eine Ganzzahl zurück, 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 2Order,Sortieren, GETISSORTED, SETISSORTED,REORDER, REORDERALL, GETORDERSTR, SETORDERSTR, Demo Order.mb.

16.29.11 Auslösefunktion Feld

Im Feldfenster (siehe Abschnitt 15.2.1 [Felder erstellen], Seite 69) kann eine Auslösefunktion definiert werden, die immer dann aufgerufen wird, denn 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 16.6.4 [SETQ], Seite 94) 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 16.6.5 [SETQ*], Seite 94) 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*

16.29.12 Virtuelle Felder programmieren

In MUIbase sind virtuelle Felder besondere Felder, deren Inhalt immer dann berechnet wird, wenn er benötigt wird. Wird z.B. zu einem anderen Datensatz gewechselt, indem man auf einen der Pfeile in der Pannelleiste einer Tabelle klickt, so wird ein virtuelles Feld in dieser Tabelle automatisch neu berechnet und angezeigt (die entsprechenden Einstellungen für das virtuelle Feld vorausgesetzt, siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75). Zum Berechnen des Wertes wird die Auslösefunktion 'Berechne' des Feldes aufgerufen. Diese Auslösefunktion kann im Feldfenster (siehe Abschnitt 15.2.2 [Typabhängige Einstellungen], Seite 69) 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 ()
  virtual-attr
)
```

Man kann auch den virtuellen Wert auf einen beliebigen 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 2Virtual, Virtuelles Feld, Beispielprojekt `Movie.db`.

16.29.13 Berechne-Aktiv-Funktion

Für Feldobjekte und Fensterknöpfe kann eine Auslösefunktion angegeben werden, welche den Aktivzustand des Objekts berechnet. Siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75, und Abschnitt 15.3.10 [Fenstereditor], Seite 81, für Informationen wie diese Auslösefunktion angegeben wird.

Die Auslösefunktion wird ohne Argumente aufgerufen. Gibt die Funktion NIL zurück, so wird das Objekt deaktiv, sonst wird es aktiv.

Zum Beispiel kann die Berechne-Aktiv-Funktion für ein Objekt, das aktiv sein soll, wenn ein bestimmtes virtuelles 'Listen'-Feld eine aktive Zeile hat, wie folgt aussehen:

```
(DEFUN enableObject ()
  (GETVIRTUALLISTACTIVE virtual-list-attr)
)
```

Siehe auch 2Attribute object editor,Feldobjekteditor,2Window editor,Fenstereditor, Beispielprojekt `Users.mb`.

16.29.14 Berechne Datensatzbeschreibung

Für Tabellenobjekte und Feldobjekte vom Typ Referenz kann eine Auslösefunktion zur Berechnung einer Datensatzbeschriftung eingegeben werden (siehe Abschnitt 15.3.2 [Tabelleobjekteditor], Seite 73, und Abschnitt 15.3.3 [Attributobjekteditor], Seite 75).

Die Auslösefunktion wird ohne Argumente aufgerufen. Sie sollte einen einfachen Ausdruck oder eine Liste von Ausdrücken zurückgeben.

Zum Beispiel, hat eine Tabelle 'Person' die Felder 'Name' und 'Birthday', so könnte die Funktion so aussehen:

```
(DEFUN personDescription ()
  (LIST Person.Name Person.Birthday)
)
```

Siehe auch 2Table object editor,Tabellenobjekteditor,2Attribute object editor,Feldobjekteditor.

16.29.15 Auslösefunktion Doppelklick

Für virtuelle Attribute, welche die Listenanzeige verwenden, kann eine Auslösefunktion angegeben werden, welche immer dann aufgerufen wird, wenn der Benutzer auf ein Listenfeld doppelt klickt. Für Informationen wie eine solche Auslösefunktion für ein virtuelles Attribut angegeben werden kann, siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75.

Die Auslösefunktion wird mit drei Argumenten aufgerufen. Das erste Argument gibt die Zeilennummer beginnend mit 1 für die erste Zeile an (Zeile 0 enthält den Listenkopf). Das zweite Argument enthält die Spaltennummer beginnend mit 0. Das dritte Argument ist ein Zeiger auf den Datensatz, aus dem das Listenfeld erzeugt wurde, oder NIL, falls der Eintrag nicht direkt aus einem Datensatz stammt. Der Rückgabewert der Funktion wird ignoriert.

Ein typisches Beispiel für eine Doppelklick-Auslösefunktion ist das folgende.

```
(DEFUN doubleClickTrigger (row col rec:Table)
  ...
)
```

Hier wurde *rec* als Datensatzzeiger für Tabelle *Table* deklariert. Dies ermöglicht den direkten Zugriff von *rec* auf Felder in *Table*.

Falls es mehr als eine Tabelle gibt, auf die sich das Datensatzargument beziehen könnte, so ist folgende Konstruktion, welche Typprädikate zu verschiedenen Tabellen einsetzt, nützlich.

```
(DEFUN doubleClickTrigger (row col rec)
  (COND
    ((RECP Table1 rec) (SETQ Table1 rec) ...)
    ((RECP Table2 rec) (SETQ Table2 rec) ...)
    ...
  )
)
```

Es ist möglich, dass das vom Benutzer angeklickte Listenfeld zu keinem Datensatz gehört. In diesem Fall kann das dritte Argument ignoriert und auf das Listenelement wie in folgendem Beispiel zugegriffen werden.

```
(DEFUN doubleClickTrigger (row col)
  (PRINT (NTH col (NTH row virtual-attr)))
)
```

Siehe auch 2Attribute object editor,Feldobjekteditor,Beispielprojekt *Movie.mb*.

16.29.16 Berechne Listenansichts-Auswahltexte

Für Zeichenkettenfelder kann neben dem GUI-Element ein Listenansicht-Popup-Knopf platziert werden, welcher bei Drücken eine Liste von Auswahltexten anzeigt, aus welcher der Benutzer wählen kann. Die Auswahltexte in dieser Liste können statisch sein, oder aber durch eine Auslösefunktion berechnet werden. Siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75, für mehr Informationen, wie zwischen statischen und berechneten Auswahltexten ausgewählt werden kann und wie die Auslösefunktion angegeben wird.

Die Auslösefunktion zur Berechnung der Auswahltexte hat keine Argumente. Sie gibt einen Memo-Text, bei der jede Zeile eine Beschriftung enthält, oder NIL für keine Beschriftungen zurück.

Zum Beispiel kann die Berechnungsfunktion wie folgt aussehen:

```
(DEFUN computeLabels ()
  "Tokio\nMünchen\nLos Angeles\nRom"
)
```

Siehe auch 2Compute reference records,Berechne Referenz-Datensätze,2Attribute object editor,Feldobjekteditor,

16.29.17 Berechne Referenz-Datensätze

Für Referenz-Felder wird neben dem GUI-Element für gewöhnlich ein Popup-Knopf angebracht, welcher bei Drücken eine Liste von Datensätzen anzeigt, aus welcher der Benutzer wählen kann. Die Liste dieser Datensätze kann durch eine Auslösefunktion berechnet werden. Siehe Abschnitt 15.3.3 [Feldobjekteditor], Seite 75, für mehr Informationen, wie die Auslösefunktion für Referenz-Felder angegeben wird.

Die Auslösefunktion zur Berechnung der Datensatzliste hat keine Argumente. Sie gibt eine Liste zurück, welche nach Datensätzen der referenzierten Tabelle durchsucht wird. Jeder solche gefundene Datensatz wird in die anzuzeigende Liste mit aufgenommen. Einträge, die keine Datensätze sind, werden stillschweigend ignoriert.

Eine typische Funktion zur Berechnung der Referenz-Datensatzliste ist das folgende Beispiel. Gegeben sei ein Projekt mit einer Tabelle 'Person', welches ein Boolesches Feld 'Female' enthält. Dann zeigt die folgende Berechnungsfunktion nur die weiblichen Personen im Referenz-Popup an:

```
(DEFUN computeFemaleRecords ()
  (SELECT Person FROM Person WHERE Female)
)
```

Siehe auch `2Compute list-view labels`, `Berechne Listenansicht-Auswahltexte`, `2Attribute object editor`, `Feldobjekteditor`,

16.30 Liste veralteter Funktionen

Die folgenden Funktionen sind seit MUIbase Version 2.7 veraltet.

- GETDISABLED
- SETDISABLED
- GETWINDOWDISABLED
- SETWINDOWDISABLED

Veraltete Funktionen arbeiten nicht mehr wie erwartet und ein Aufruf wird entweder ignoriert (was einer Null-Operation entspricht), ein Warn-Dialog wird geöffnet, oder ein Fehler wird generiert. Menüpunkt 'Programm - Veraltete Funktionen' legt das genaue Vorgehen fest (siehe Abschnitt 7.2.9 [Veraltete Funktionen], Seite 39).

Es wird empfohlen, alle veralteten Funktionen im Projekt-Programm zu entfernen, und die Funktionalität über die Aktiv/Inaktiv-Einstellung der Feldobjekte und Fensterknöpfe zu realisieren (siehe Abschnitt 16.29.13 [Berechne-Aktiv-Funktion], Seite 164).

17 ARexx Schnittstelle

Die ARexx-Schnittstelle ist nur in der MUIbase-Version für Amiga verfügbar.

ARexx ist eine Standardschnittstelle für Amiga-Software, um Zugriff auf Funktionen und Daten eines Programms nach außen für ein anderes Programm bereitzustellen. MUIbase bietet eine solche ARexx-Schnittstelle mit einem kleinen aber wohldefinierten Befehlssatz an, welcher es externer Software erlaubt, den vollen Zugriff über MUIbase zu erhalten. Weiterhin stellt die ARexx-Schnittstelle von MUIbase einen Transaktionsmechanismus ähnlich zu anderen relationalen Datenbanken bereit.

Beispiel-ARexx-Befehlsdateien für MUIbase findet man im Verzeichnis 'rexx'.

17.1 Portname

Der Portname des MUIbase ARexx-Ports lautet 'MUIbase.n' wobei *n* ein Zähler beginnend mit 1 ist. Normalerweise, d.h. wenn Sie MUIbase nur einmal starten, lautet der Portname 'MUIbase.1'.

Sie benötigen den Portnamen für den ARexx-Befehl `address`, welcher aufgerufen werden muss, bevor irgendein ARexx-Befehl von MUIbase aufgerufen wird. Das folgende Programmfragment zeigt wie man das Vorhandensein des MUIbase ARexx-Ports feststellen kann, MUIbase ggf. startet und dann den Port adressiert.

```
if ~show(ports, MUIbase.1) then
do
  address command 'run <nil: >nil: MUIbase:MUIbase -n'
  address command 'waitforport MUIbase.1'
end

address MUIbase.1
```

Siehe auch Beispiel-ARexx-Befehlsdatei `address.rexx`.

17.2 Befehlsaufbau

Nachdem der ARexx-Port von MUIbase adressiert wurde, stehen alle MUIbase ARexx-Befehle zur Verfügung. Der Aufbau der Befehle ist ähnlich wie der vieler anderer Implementierungen:

```
cmd [arg1 ...]
```

wobei *cmd* einer der Befehle in den folgenden Teilen dieses Kapitels ist und *arg1 ...* optionale Argumente für den Befehl sind.

Da der ARexx-Übersetzer Befehlszeilen auswertet bevor sie an MUIbase gesendet werden, ist es manchmal nützlich, einige oder alle Argumente durch Anführungszeichen zu schützen. Es wird empfohlen einfache Anführungszeichen (') hierfür zu verwenden. Hierdurch können doppelte Anführungszeichen (") in den Argumenten, z.B. für Zeichenkettenkonstanten eingesetzt werden. Werden Argumente mit einfachen Anführungszeichen geschützt, so ist immer noch möglich, Werte von ARexx-Variablen mit in die Befehlszeile zu integrieren, indem um die ARexx-Variablen wiederum einzelne Anführungszeichen gesetzt werden. Hierzu ein Beispiel, welches den ARexx-Befehl `eval` verwendet.

```
search = ...
```

```
eval handle 'select Name from Person where (like Name "*"search'*')'
```

Siehe auch 2ARexx eval, Eval.

17.3 Rückgabewerte

Nach dem Aufruf eines MUIbase ARexx-Befehls werden verschiedenen ARexx-Variablen mit dem Ergebnis des Aufrufs aktualisiert. Um das Lesen aller Resultate eines Befehls zu ermöglichen, sollten Sie die ARexx-Option `results` aktivieren. Dies erreicht man mit der folgenden Zeile am Anfang einer ARexx-Befehlsdatei.

```
options results
```

Es gibt 3 ARexx-Variablen, welche von der MUIbase-ARexx-Schnittstelle gesetzt werden können: `rc`, `results` und `lasterror`. Variable `rc` wird immer gesetzt und reflektiert den Erfolg oder Misserfolg eines Befehls. War ein Befehl erfolgreich, so enthält `results` das eigentliche Resultat des Befehls, wogegen im Falle eines Misserfolges, `lasterror` zusätzliche Information enthalten kann, die den Fehler genauer beschreibt.

Für die Variable `rc` existieren die folgenden Fehlerwerte:

Rückgabewert Bedeutung

0	Erfolg. Variable <code>result</code> enthält das Ergebnis.
-1	Implementierungsfehler. Sollte nie auftreten.
-2	Speicherplatzmangel.
-3	Unbekannter ARexx-Befehl.
-4	Syntax-Fehler
<= -10	Anderer Fehler. Fehlerbeschreibung steht in <code>lasterror</code> .
-12	Übersetzungsfehler (nur für <code>compile</code> -Befehl).

Man beachte, dass nur für `rc <= -10` die Variable `lasterror` eine gültige Fehlerbeschreibung enthält. Weitere Rückgabewerte sind für zukünftige Versionen vorbehalten, um eine genauere Fehlerbehandlung zu ermöglichen.

Hier ist ein typisches Programmfragment, das zeigt wie man das Ergebnis eines ARexx-Befehls untersuchen kann.

```
eval handle 'select * from Accounts'
if (rc == 0) then
  say result
else if (rc == -1) then
  say "Implementation error"
else if (rc == -2) then
  say "Out of memory"
else if (rc == -3) then
  say "Unknown command"
else if (rc == -4) then
  say "Command syntax error"
else if (rc <= -10) then
```

```

    say lasterror
else
    say "Error: " rc

```

17.4 Quit

Der Befehl `quit` beendet MUIbase. Siehe auch die MUI-Dokumentation.

17.5 Hide

Der Befehl `hide` schließt (ikonifiziert) alle offenen MUIbase-Fenster. Siehe auch die MUI-Dokumentation.

17.6 Show

Mit `show` werden alle zuvor geschlossenen (ikonifizierten) Fenster wieder geöffnet. Siehe auch die MUI-Dokumentation.

17.7 Info

Der Befehl `info` liefert Informationen über Titel, Autor, Copyright, Beschreibung, Version, ARexx-Port und Bildschirm einer MUI-Applikation.

Befehl	Wert von <i>result</i>
<code>info title</code>	Titel der Applikation
<code>info author</code>	Autor der Applikation
<code>info copyright</code>	Copyright-Nachricht
<code>info description</code>	Kurzbeschreibung
<code>info version</code>	Versionsnummer
<code>info base</code>	Name des ARexx-Ports
<code>info screen</code>	Name des Bildschirms

Siehe auch die MUI-Dokumentation.

17.8 Help

Der Befehl `help` gibt alle verfügbaren ARexx-Befehle einer MUI-Applikation in eine Datei aus.

```

help filename

```

Die ARexx-Befehle werden in der AmigaDos-Befehlssyntax ausgegeben. Siehe die MUI-Dokumentation für weitere Informationen und das AmigaDos-Handbuch für die Befehlssyntax.

17.9 Compile

Der `compile`-Befehl übersetzt eine externe Programmquelldatei.

```

compile source [update]

```

Der Befehl übersetzt die externe Programmquelldatei des Projekts, dessen externe Programmquelldatei auf die gleiche Datei wie *source* zeigt. Bei erfolgreicher Übersetzung wird 0 zurückgegeben und, falls **update** angegeben wurde, die externe Quelldatei aktualisiert. Das Aktualisieren der Quelldatei erlaubt es die MUIbase-Schlüsselwörter zu "verschönern" (*pretty print*). Ein erfolgreich übersetztes Programm wird als Projektprogramm übernommen und bei Aufruf von Auslösefunktionen verwendet.

Schlägt die Übersetzung fehl, so wird der Fehlerwert -12 zurückgeliefert und die Variable *lasterror* auf eine 4 Zeilen lange Zeichenkette gesetzt:

- Die erste Zeile enthält den Dateinamen, in welcher der Fehler auftrat.
- Die zweite Zeile enthält die Fehlerzeile beginnend mit 1.
- Die dritte Zeile enthält die Fehlerspalte beginnend mit 1.
- Die vierte Zeile beschreibt den Fehler im Klartext.

Bitte beachten Sie, dass ein Projekt bereits geöffnet sein muss bevor der **compile**-Befehl zum Übersetzen seiner externen Programmquelle benutzt werden kann. Im Falle, dass kein Projekt, dessen externe Programmquelle mit *source* übereinstimmt, gefunden werden kann, wird ein Fehlerwert ≤ -10 (aber ungleich -12) zurückgegeben und *lasterror* entsprechend gesetzt.

17.10 Connect

Der **connect**-Befehl öffnet einen Kommunikationskanal zu einem MUIbase-Projekt.

```
connect project-name [GUI]
```

Der Befehl überprüft zunächst, ob das durch **project-name** angegebene Projekt bereits geöffnet wurde und lädt es gegebenenfalls. Ein Projekt wird immer nur einmal geöffnet und mehrfache Verbindungen zum gleichen Projekt teilen sich den Zugriff auf die Datenbank. Es wird ein eindeutiges Handle zur Kommunikation bestimmt. Das Kommunikations-Handle ist eine Ganzzahl ungleich Null und wird später für den Zugriff auf die Datenbank benötigt. Falls das Schlüsselwort **GUI** der Kommandozeile angehängt wurde, so wird zusätzlich beim Öffnen des Projekts die graphische Benutzerschnittstelle des Projekts geöffnet. Andernfalls wird keine graphische Benutzerschnittstelle angezeigt und ARexx-Befehle werden im Hintergrund ohne direkte Benutzerinteraktion verarbeitet.

Falls das Öffnen des Projekts erfolgreich verlief, gibt der Befehl eine 0 zurück und setzt die ARexx-Variable *result* auf den Wert des Kommunikations-Handles.

Beispiel: `'connect "MUIbase:Demos/Movies.mb"'` öffnet eine Verbindung zur Beispiel-Filmdatenbank.

Siehe auch **2ARexx disconnect, Disconnect, 2ARexx connections, Connections, 2ARexx return codes, Rückgabewerte**.

17.11 Disconnect

Der **disconnect**-Befehl schließt eine bestehende Verbindung.

```
disconnect handle
```

Schließt die durch *handle* angegebene Datenbankverbindung. Falls es die einzige Verbindung zu diesem Projekt war und kein graphisches Benutzerinterface für das Projekt geöffnet

wurde, so wird das Projekt geschlossen und aus dem Speicher entfernt. Andernfalls bleibt das Projekt geöffnet.

Beispiel: 'disconnect 1' schließt die Verbindung zum Datenbank-Projekt mit Handle-Wert 1.

Siehe auch 2ARexx connect,Connect, 2ARexx connections,Connections,2ARexx return codes,Rückgabewerte.

17.12 Connections

Um herauszufinden, welche Verbindungen zu Projekten bestehen, dient der Befehl `connections`.

```
connections
```

Falls erfolgreich, gibt `connections` den Wert 0 zurück und setzt die ARexx-Variable `result` auf eine lesbare Zeichenkette, bei der jede Zeile eine Verbindung bestehend aus Handle-Wert und Projektnamen enthält.

Beispiel: die Variable `result` könnte nach Ausführen des `connections`-Befehls wie folgt aussehen:

```
3 MUIbase:Demos/Accounts.mb
5 MUIbase:Demos/Movies.mb
6 MUIbase:Demos/Movies.mb
7 MUIbase:Demos/Movies.mb
```

Siehe auch 2ARexx connect,Connect, 2ARexx disconnect,Disconnect,2ARexx return codes,Rückgabewerte.

17.13 Eval

Der wichtigste Befehl des MUIbase-ARexx-Ports um Daten abzufragen und zu aktualisieren, ist der `eval`-Befehl.

```
eval handle lisp-cmd
```

Der `eval`-Befehl wertet die angegebene Anweisung `lisp-cmd` (welche in der MUIbase-Lisp-Sprache geschrieben sein muss) in dem durch `handle` angegebene Projekt aus. Ein Kommunikationskanal `handle` kann durch Aufrufen des `connect`-Befehls erhalten werden. Die Anweisung `lisp-cmd` kann ein beliebiger Ausdruck der MUIbase-Programmiersprache sein. Optional können die äußersten Klammern um den Ausdruck herum weggelassen werden. Es wird empfohlen, `lisp-cmp` mit einfachen Anführungszeichen wie in Abschnitt 17.2 [Befehlsaufbau], Seite 167. beschrieben zu versehen.

Falls die Anweisung erfolgreich ausgeführt werden konnte, so gibt `eval` eine 0 zurück und setzt die ARexx-Variable `result` auf eine Zeichenkettenrepräsentation des Ergebniswertes von `lisp-cmd`. Die Zeichenkette repräsentiert dabei das Ergebnis in einer solchen Weise, dass es möglich ist, die Art der zurückgegebenen Daten herauszufinden. Beispielsweise werden Zeichenketten mit doppelten Anführungszeichen und Listen mit Klammern versehen, wobei die Listenelemente durch Leerzeichen und Zeilenvorschübe getrennt werden. Falls Sie ein bestimmtes Zeichenketten-Format erzwingen möchten, so bietet es sich an, diese Format selber innerhalb der Lisp-Anweisung zu erzeugen.

Falls eine Veränderung des Projekts innerhalb des `eval`-Befehls vorgenommen wurde und keine Transaktion (siehe Abschnitt 17.14 [Transaction], Seite 172) gestartet wurde, so

werden die Änderungen automatisch gespeichert (auto commit). Andernfalls, d.h. es wurde eine Transaktion gestartet, werden die Änderungen im Speicher gehalten, bis ein `commit`-Befehl alle Änderungen speichert, oder ein `rollback`-Befehl alle Änderungen zurücknimmt.

Beispiel:

```
options results
address MUIbase.1
connect "MUIbase:Demos/Movie.mb"
if rc = 0 then
do
    handle = result
    eval handle 'select Title, Director from Movies'
end
if rc = 0 then
    say result
```

Die Ausgabe dieses Beispielprogramms könnte wie folgt aussehen:

```
( ( "Title" "Director" )
  ( "Batman" "Tim Burton" )
  ( "Batman Returns" "Tim Burton" )
  ( "Speechless" "Ron Underwood" )
  ( "Tequila Sunrise" "Robert Towne" )
  ( "Mad Max" "George Miller (II)" )
  ( "Braveheart" "Mel Gibson" )
  ( "2010" "Peter Hyams" ) )
```

Siehe auch 2ARexx connect, Connect, 2ARexx command syntax, Befehlsaufbau, 2ARexx return codes, Rückgabewerte, 2ARexx transaction, Transaction, 2ARexx commit, Commit, Beispiel-ARexx-Befehlsdatei `movies.rexx`.

17.14 Transaction

Der ARexx-Port von MUIbase erlaubt es Transaktionen auf einer Datenbank auszuführen. Eine Transaktion ist eine Menge von auf einer Datenbank operierender Befehlen, welche möglicherweise die Datenbank verändern und die entweder alle komplett ausgeführt werden mit einem abschließenden Speichern aller Änderungen (`commit`), oder zu einem beliebigen Zeitpunkt innerhalb der Transaktion abgebrochen werden können (`rollback`). Eine Transaktion wird durch den folgenden Befehl gestartet:

```
transaction handle
```

wobei *handle* auf ein Projekt verweist, das zuvor mit dem `connect`-Befehl geöffnet wurde (siehe Abschnitt 17.10 [Connect], Seite 170).

Nach dem Ausführen eines `transaction`-Befehls können beliebig viele `eval`-Befehle folgen ohne dass die Datenbank tatsächlich verändert wird. Irgendwann jedoch müssen Sie entscheiden, ob die Änderungen gespeichert werden sollen (siehe Abschnitt 17.15 [Commit], Seite 173), oder ob zu dem Zustand vor Ausführen des `transaction`-Befehls zurückgekehrt werden soll (siehe Abschnitt 17.16 [Rollback], Seite 173).

Nach Ausführen des `transaction`-Befehls erhält das zugehörige Kommunikations-Handle einen exklusiven Zugriff auf das Projekt. Andere Programme, welche auf die

Datenbank zugreifen wollen, inkl. der Benutzerschnittstelle, werden blockiert (bzw. verzögert im Falle einer ARexx-Verbindung), bis der exklusive Zugriff durch einen `commit`- oder `rollback`-Befehl wieder freigegeben wird.

Normalerweise gibt der `transaction`-Befehl den Wert 0 zurück. Falls ein anderes ARexx-Programm bereits exklusiven Zugriff zu dem Projekt hat, auf welches `handle` verweist, so wird die Ausführung blockiert bis die andere Verbindung den Zugriff wieder freigibt.

Siehe auch 2ARexx `eval, Eval`, 2ARexx `commit, Commit`, 2ARexx `rollback, Rollback`, 2ARexx `return codes, Rückgabewerte`.

17.15 Commit

Der `commit`-Befehl wird am Ende einer Transaktion aufgerufen, um alle getätigten Änderungen eines Projekts zu speichern.

```
commit handle
```

Der `commit`-Befehl beendet eine Transaktion (siehe Abschnitt 17.14 [Transaction], Seite 172) indem das Projekt, auf das `handle` verweist, gespeichert wird.

Falls erfolgreich gibt `commit` den Wert 0 zurück. Falls keine Transaktion gestartet wurde vor Ausführen von `commit` oder falls ein anderer Fehler auftritt, so wird ein Wert ungleich 0 zurückgegeben.

Siehe auch 2ARexx `rollback, Rollback`, 2ARexx `transaction, Transaction`, 2ARexx `return codes, Rückgabewerte`.

17.16 Rollback

Um alle Änderungen einer Transaktion zu verwerfen, wird der `rollback`-Befehl verwendet.

```
rollback handle
```

Alle Änderungen, die an dem Projekt, auf welches `handle` verweist, seit dem Start der aktuellen Transaktion (siehe Abschnitt 17.14 [Transaction], Seite 172) vorgenommen wurden, werden verworfen und der Zustand des Projekts wird auf den Zustand wie vor Ausführen der Transaktion gebracht.

Falls erfolgreich so gibt `rollback` den Wert 0 zurück.

Siehe auch 2ARexx `commit, Commit`, 2ARexx `transaction, Transaction`, 2ARexx `return codes, Rückgabewerte`.

Menüs

Menü Projekt

Information

Information über ein Projekt, siehe Abschnitt 6.2 [Info], Seite 31.

Neu Ein neues Projekt starten, siehe Abschnitt 6.3 [New project], Seite 31.

Leeren - Projekt

Projekt zurücksetzen, siehe Abschnitt 6.4 [Clear project], Seite 32.

Leeren - Datensätze

Alle Datensätze löschen, siehe Abschnitt 6.4 [Clear project], Seite 32.

Öffnen - Projekt

Projekt laden, siehe Abschnitt 6.5 [Open project], Seite 32.

Öffnen - Demo

Demoprojekt laden, siehe Abschnitt 6.5 [Open project], Seite 32.

Neu laden Neue Version von Disk laden, siehe Abschnitt 6.5 [Open project], Seite 32.

Zurück zum Gespeicherten

Zuletzt gespeicherten Zustand wieder laden, siehe Abschnitt 6.6 [Save project], Seite 32.

Speichern Projekt auf Platte speichern, siehe Abschnitt 6.6 [Save project], Seite 32.

Speichern & Umschichten

Projekt speichern und umschichten, siehe Abschnitt 6.6 [Save project], Seite 32.

Speichern & Umschichten als

Projekt unter neuem Namen speichern und umschichten, siehe Abschnitt 6.6 [Save project], Seite 32.

Zum Admin-Modus wechseln

Zugriffsbeschränkung auf Projektstruktur, siehe Abschnitt 6.7 [Admin and user mode], Seite 33.

Admin-Passwort ändern

Authentifizierung für Admin-Modus, siehe Abschnitt 6.7 [Admin and user mode], Seite 33.

Überprüfe Datenintegrität

Wenn das Projekt jemals beschädigt sein sollte, siehe Abschnitt 6.8 [Check data integrity], Seite 33.

Datensätze auslagern

Lagert alle Datensätze auf Platte aus, siehe Abschnitt 6.9 [Swap records], Seite 34.

Struktureditor

Öffnet den Struktureditor, siehe Kapitel 15 [Structure editor], Seite 67.

Struktur exportieren

Übersicht über alle Tabellen und Felder, siehe Abschnitt 15.4 [Export structure], Seite 82.

Schließen Wenn das Projekt beendet ist, siehe Abschnitt 6.10 [Close project], Seite 34.

Speichern & Schließen

Speichert und schließt das Projekt, siehe Abschnitt 6.10 [Close project], Seite 34.

Beenden MUIbase verlassen, siehe Kapitel 3 [Getting started], Seite 6.

Menü Einstellungen

Benutzereinstellungen

Globale Einstellungen in der Benutzerumgebung, siehe Abschnitt 7.1 [User settings], Seite 35.

Formate Fließkommazahl- und Datumsformat, siehe Abschnitt 7.1.1 [Formats], Seite 35.

Externer Editor

Festlegen des externen Editors, siehe Abschnitt 7.1.2 [External editor], Seite 35.

Externer Anzeiger

Festlegen des externen Anzeigers, siehe Abschnitt 7.1.3 [External viewer], Seite 36.

Extra-Knöpfe in Tab-Kette

Einbinden der Extra-Knöpfe in die Aktivierungskette, siehe Abschnitt 7.1.4 [Extra buttons in Tab chain], Seite 36.

Weiterspringen bei <Enter>

Zum nächsten Feld bei Drücken der *Enter*-Taste, siehe Abschnitt 7.1.5 [Advance on Enter], Seite 37.

Beenden bestätigen

Sicherheitsabfrage beim Beenden von MUIbase, siehe Abschnitt 7.1.6 [Confirm quit], Seite 37.

MUI MUI's Einstellungen, siehe Abschnitt 7.1.7 [MUI], Seite 37.

Projekteinstellungen

Einstellungen, die mit einem Projekt gespeichert werden, siehe Abschnitt 7.2 [Project settings], Seite 37.

Datensatzspeicher

Größe des Datensatzspeichers, siehe Abschnitt 7.2.1 [Record memory], Seite 37.

Datensätze löschen bestätigen

Sicherheitsabfrage beim Löschen von Datensätzen, siehe Abschnitt 7.2.2 [Confirm delete record], Seite 38.

Pfade relativ zu einem Projekt

Wie relative Pfadnamen behandelt werden, siehe Abschnitt 7.2.3 [Paths relative to project], Seite 38.

Automatisches Neuladen bestätigen

Sicherheitsabfrage beim automatischem Neuladen eines Projekts, siehe Abschnitt 7.2.4 [Confirm auto reload], Seite 38.

Speichern&Umschichten bestätigen

Sicherheitsabfrage beim Speichern&Umschichten, siehe Abschnitt 7.2.5 [Confirm save & reorg], Seite 39.

Als Voreinstellungen speichern

Speichert die Projekteinstellungen für neue Projekte, siehe Abschnitt 7.3 [Save as default], Seite 41.

Menü Log

Protokollieren aktiv

Wie das Protokollieren gestartet und gestoppt wird, siehe Abschnitt 8.2 [Activate logging], Seite 43.

Logbeschriftung setzen

Setzen einer Logbeschriftung für neue Logeinträge, siehe Abschnitt 8.3 [Set log label], Seite 43.

Log importieren

Importieren von Logeinträgen aus einer Datei, siehe Abschnitt 8.4 [Import log], Seite 43.

Log exportieren

Exportieren von Logeinträgen in eine Datei, siehe Abschnitt 8.5 [Export log], Seite 43.

Log löschen

Löschen aller Logeinträge, siehe Abschnitt 8.6 [Clear log], Seite 43.

Log ansehen

Wie alle Logeinträge angesehen werden können, siehe Abschnitt 8.7 [View log], Seite 44.

Menü Tabelle

Neuer Datensatz

Neuen Datensatz hinzufügen, siehe Abschnitt 9.2 [Adding records], Seite 45.

Datensatz kopieren

Datensatz kopieren, siehe Abschnitt 9.2 [Adding records], Seite 45.

Datensatz löschen

Wenn ein Datensatz nicht mehr benötigt wird, siehe Abschnitt 9.4 [Deleting records], Seite 48.

Lösche alle Datensätze

Wenn von vorne begonnen werden soll, siehe Abschnitt 9.4 [Deleting records], Seite 48.

Gehe zum Datensatz

Datensätze durchforsten, siehe Abschnitt 9.5 [Browsing records], Seite 48.

Ändere Filter

Filterausdruck festlegen, siehe Abschnitt 10.1.2 [Changing filters], Seite 50.

Ändere Sortierung

Sortierung festlegen, siehe Abschnitt 11.4 [Changing orders], Seite 53.

Neu sortieren aller Datensätze

Falls Datensätze jemals unsortiert sein sollten, siehe Abschnitt 11.5 [Reorder all records], Seite 54.

Suche nach

Wie nach einem Datensatz gesucht wird, siehe Abschnitt 12.1 [Search dialog], Seite 55.

Suche vorwärts

Gehe zum nächsten passenden Datensatz, siehe Abschnitt 12.2 [Forward/backward search], Seite 55.

Suche rückwärts

Gehe zum vorhergehenden passenden Datensatz, siehe Abschnitt 12.2 [Forward/backward search], Seite 55.

Importiere Datensätze

Wie Datensätze importiert werden, siehe Abschnitt 13.3 [Importing records], Seite 58.

Exportiere Datensätze

Wie Datensätze exportiert werden, siehe Abschnitt 13.4 [Exporting records], Seite 58.

Alle Datensätze ansehen

Auflisten aller Datensätze einer Tabelle, siehe Abschnitt 9.6 [View all records], Seite 49.

Menü Programm

Ändern Wo ein MUIbase-Programm eingegeben wird, siehe Abschnitt 16.1 [Program editor], Seite 83.

Kompilieren

Ein Programm kompilieren, siehe Abschnitt 16.1 [Program editor], Seite 83.

Programmquellen

Interne oder externe Programmquellen, siehe Abschnitt 7.2.6 [Program source], Seite 39.

Externe Programmquellen aufräumen

Löschen unbenötigter externer Programmquellen, siehe Abschnitt 7.2.7 [Cleanup external program source], Seite 39.

Debug-Information einbinden

Mit oder ohne Debug-Information kompilieren, siehe Abschnitt 7.2.8 [Program debug information], Seite 39.

Veraltete Funktionen

Behandlung von Aufrufen veralteter Funktionen, siehe Abschnitt 7.2.9 [Obsolete functions], Seite 39.

Trigger-Funktionen sortieren

Sortierung der Funktionen in Popup-Fenstern, siehe Abschnitt 7.2.10 [Sort trigger functions], Seite 40.

Include-Verzeichnis

Wo nach externen Include-Dateien gesucht wird, siehe Abschnitt 7.2.11 [Program include directory], Seite 40.

Ausgabedatei

Wohin die Programmausgabe geht, siehe Abschnitt 7.2.12 [Program output file], Seite 40.

Abfragen Öffnet das Abfragenfenster, siehe Abschnitt 14.2 [Query editor], Seite 60.

Menu Hilfe

Inhalt Diese Anleitung.

Über Über MUIbase, siehe Kapitel 1 [Copying], Seite 1.

Über MUI

Über das Magic User Interface, siehe Abschnitt 1.5 [Third party material], Seite 2.

Anerkennung

Dank geht an:

- Ralph Reuchlein (Ralphie) für Fehlerreports, Ideen und Verbesserungen, und für die deutsche übersetzung des MUIbase-Handbuchs.
Ralphie erstellte und verwaltet auch die MUIbase Homepage <http://muibase.sourceforge.net>.
- Pascal Marcellin für seinen Glauben in MUIbase und Amiga, und für den ersten Web-Server, der mittels ARexx mit MUIbase kommuniziert.
- Christoph Pözl und Sébastien Pözl für die Grafiken in MUIbase, das PNG-Icon-Set und für das Testen auf MorphOS.
- Alexandre Balaban für die französische Übersetzung des Benutzerhandbuchs (mit großer Hilfe von Lionel Muller und Gilles Mathevet) und für die Portierung von MUIbase auf AmigaOS 4.
- Stéphane Aulery für die Fortsetzung der französischen Übersetzung und der Aktualisierung der Webseiten.
- Harold Kinds für das neue ImportSample-Projekt.
- Ilkka Lehtoranta für die Fertigstellung der MorphOS-Portierung.
- Thomas Fricke und Adrian Maleska für verschiedene Grafiken, welche das Erscheinungsbild von MUIbase verbessern.
- Martin Merz für seine Mason-Piktogramme.
- Mats Granstrom für das Beta-Testen und das Schreiben des Tutorials.
- John Hertig, Harold Kinds, Magnus Hammarstrom, Henning Thilemann, Joseph Durchalet, André Schenk, Klaus Gessner und Oliver Roberts für Ideen und das Beta-Testen.
- Jernej Simoncic für die Erlaubnis, sein Windows-Installationsskript für The Gimp als Grundlage für das Installationsskript für MUIbase für Windows zu verwenden.

Autor

MUIbase wurde entwickelt von:

Steffen Gutmann

Email: muibase@yahoo.com

Die über 10000-zeilige 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

Funktionsverzeichnis

#

#define	84
#elif	86
#else	86
#endif	86
#if	85
#ifdef	85
#ifndef	85
#include	85
#undef	85

*

*	108
---	-----

+

+	107
---	-----

-

-	108
---	-----

/

/	109
---	-----

<

<	105
<*	105
<=	105
<=*	105
<>	105
<>*	105

=

=	105
=*	105

>

>	105
>*	105
>=	105
>=*	105

1

1+	108
1-	108

A

ABS	109
ADDMONTH	124
ADDYEAR	124
ADMINPASSWORD	150
AND	104
APPEND	126
APPLY	95
ASC	117
ASKBUTTON	131
ASKCHOICE	129
ASKCHOICESTR	130
ASKDIR	128
ASKFILE	128
ASKINT	129
ASKMULTI	132
ASKOPTIONS	131
ASKSTR	128
ATTRNAME	142

C

CASE	96
CHANGES	150
CHR	118
CMP	106
CMP*	106
COMPLETE	154
COMPLETEADD	153
COMPLETEMAX	153
CONCAT	116
CONCAT2	116
COND	96
CONS	125
CONSP	100
COPYREC	142
COPYSTR	117

D

DATE	103
DATEDMY	123
DATEP	100
DAY	123
DEFUN	91
DEFUN*	92
DEFVAR	92
DEFVAR*	93
DELETE	139
DELETE*	140
DELETEALL	140
DIRNAME	152
DIV	109
DO	98

DOLIST 97
 DOTIMES 96

E

EDIT 150
 EDIT* 151
 ERROR 100
 EXIT 99
 EXP 110

F

FCLOSE 135
 FEOF 136
 FERROR 136
 FFLUSH 138
 FGETCHAR 137
 FGETCHARS 137
 FGETMEMO 138
 FGETSTR 137
 FIELD 115
 FIELDS 115
 FILENAME 152
 FILLMEMO 122
 FIRST 126
 FOPEN 134
 FOR ALL 98
 FORMATMEMO 122
 FPRINTF 136
 FPUTCHAR 138
 FPUTMEMO 138
 FPUTSTR 138
 FSEEK 136
 FTELL 137
 FUNCALL 95

G

GC 154
 GETADMINMODE 150
 GETDISABLED 166
 GETFILTERACTIVE 145
 GETFILTERSTR 145
 GETISSORTED 141
 GETLABELS 142
 GETMATCHFILTER 140
 GETORDERSTR 143
 GETVIRTUALLISTACTIVE 148
 GETWINDOWDISABLED 166
 GETWINDOWOPEN 148

H

HALT 100

I

IF 95
 INDENTMEMO 123
 INDEXBRK 112
 INDEXBRK* 112
 INDEXSTR 112
 INDEXSTR* 112
 INSMIDSTR 112
 INT 102
 INTP 100

L

LAST 126
 LEFTSTR 111
 LEN 111
 LENGTH 125
 LET 93
 LIKE 118
 LINE 121
 LINES 121
 LIST 125
 LISTP 100
 LISTTOMEMO 122
 LISTTOSTR 116
 LOG 110
 logLabel 159
 LOWER 117

M

mainWindowTitle 160
 MAPFIRST 127
 MAX 106
 MAX* 106
 MAXLEN 142
 MEMO 102
 MEMOP 100
 MEMOTOLIST 121
 MESSAGE 153
 MIDSTR 111
 MIN 107
 MIN* 107
 MOD 109
 MONTH 123
 MONTHDAYS 124

N

NEW 139
 NEW* 139
 NEXT 99
 NOT 105
 NOW 125
 NTH 126
 NULL 100

O

onAdminMode	159
onChange	159
onClose	158
onOpen	158
onReload	158
OR	104

P

POW	110
PREPARECHANGE	149
PRINT	135
PRINTF	135
PROG1	93
PROGN	93
PROJECTNAME	149
PUBSCREEN	154

R

RANDOM	110
REAL	103
REALP	100
RECNUM	141
RECORD	146
RECORDS	146
RECP	100
REMHARS	114
REORDER	144
REORDERALL	145
REPLACESTR	113
REPLACESTR*	114
REST	126
RETURN	99
REVERSE	126
RIGHTSTR	111
RINDEXBRK	113
RINDEXBRK*	113
RINDEXSTR	113
RINDEXSTR*	113
ROUND	109

S

SELECT	146
SETADMINMODE	150
SETBGPEN	148
SETCURSOR	147
SETDISABLED	166
SETFILTERACTIVE	145
SETFILTERSTR	145
SETISSORTED	141
SETLABELS	143

SETMATCHFILTER	141
SETMIDSTR	111
SETORDERSTR	144
SETQ	94
SETQ*	94
SETVIRTUALLISTACTIVE	149
SETWINDOWDISABLED	166
SETWINDOWOPEN	148
SHA1SUM	117
SORTLIST	127
SORTLISTGT	127
SPRINTF	118
SQRT	110
STAT	152
stdout	135
STR	101
STRP	100
STRTOLIST	115
SYSTEM	151
SYSTEM*	152

T

TABlename	143
TACKON	152
TIME	103
TIMEP	100
TODAY	125
TRIMSTR	114
TRUNC	109

U

UPPER	117
-------------	-----

V

VIEW	151
VIEW*	151

W

WORD	114
WORDS	115

Y

YEAR	123
YEARDAYS	124

Stichwortverzeichnis

1

1:1-Beziehungen	26
1:n-Beziehungen	26

A

Abfragebeispiele	64
Abfrageeditor	60
Abfragen als PDF exportieren	62
Abfragen als Text exportieren	61
Abfragen ausdrucken	62
Admin-Modus	33
Admin-Passwort	33
Aktive Objekte	45
Aktive Tabellen	45
Aktualisieren einer bereits vorhandenen MUIbase Version	7
Alle Datensätze ansehen	49
Als Voreinstellungen speichern	41
Amiga-Version	2
Anerkennung	179
Anzeigebereich	72
Anzeigeverwaltung	72
ARexx	167
ARexx Befehlsaufbau	167
ARexx Commit	173
ARexx Compile	169
ARexx Connect	170
ARexx Connections	171
ARexx Disconnect	170
ARexx Eval	171
ARexx Help	169
ARexx Hide	169
ARexx Info	169
ARexx Portname	167
ARexx Quit	169
ARexx Rückgabewerte	168
ARexx Rollback	173
ARexx Show	169
ARexx Transaction	172
Aufbau von Ausdrücken	157
Ausgabedatei	40
Auslösefunktion Doppelklick	164
Auslösefunktion Feld	162
Auslösefunktion Löschen	161
Auslösefunktion Neu	160
Auslösefunktionen	158
Auswahlfelder	22
Auswahltexteditor	70
Automatisches Neuladen bestätigen	38
Autor	180

B

Beenden bestätigen	37
Befehle definieren	91
Befehlsaufbau	91
Beispiel-Importdatei	57
Benutzereingabefunktionen	128
Benutzereinstellungen	35
Benutzermodus	33
Benutzerschnittstelle	28
Berechne Datensatzbeschreibung	164
Berechne Listenansichts-Auswahltexte	165
Berechne Referenz-Datensätze	165
Berechne-Aktiv-Funktion	164
Beschädigte Datenbank	33
BetterString	2
Beziehungen	25
Beziehungsfelder	23
Bildeditor	80
Bilder	29
Bildfelder	22
Boolesche Felder	22
Boolesche Funktionen	104

D

Dankeschön	179
Dateiformat	31
Dateiformat für Import und Export	57
Dateinamenfelder	22
Datenabfragen	60
Datensätze	21
Datensätze auslagern	34
Datensätze durchforsten	48
Datensätze exportieren	58
Datensätze importieren	58
Datensätze kopieren	45
Datensätze löschen bestätigen	38
Datensätze vervielfältigen	45
Datensatzbearbeitung	45
Datensatzfilter	50
Datensatzfunktionen	138
Datensatzinhalte ansprechen	88
Datensatzspeicher	37
Datentypen zum Programmieren	89
Datumfunktionen	123
Datumfelder	23
Debug-Information	39
Delete record	48

E

E/A-Funktionen	134
Eingabe von Auswahlwerten	46
Eingabe von Beziehungswerten	47
Eingabe von Booleschen Werten	46
Eingabe von Datumswerten	46
Eingabe von Ganzzahlwerten	46
Eingabe von NIL-Werten	48
Eingabe von Zeitwerten	46
Eins-zu-Eins-Beziehungen	26
Eins-zu-Mehrfach-Beziehungen	26
Einstellungen	35
Email-Verteiler	1
Externe Programmquellen	83
Externe Programmquellen aufräumen	39
Externer Anzeiger	36
Externer Editor	35
Extra-Knöpfe die Tab-Kette	36

F

Felder	21
Felder ändern	71
Felder erstellen	69
Felder kopieren	71
Felder löschen	72
Felder sortieren	72
Feldverwaltung	69
Feldfunktionen	142
Feldobjekte	29
Feldobjekteditor	75
Feldtypen	21
Feldtypen (Tabelle)	24
Fenster	28
Fenstereditor	81
Filter	50
Filter ändern	50
Filterausdruck	50
Filterbeispiele	51
Fließkommazahlfelder	22
Formate	35
Fremde Hilfsmittel	2
Funktionale Parameter	155
Funktionen für mehrzeilige Texte	121

G

Ganzzahlfelder	22
Gewichtungsobjekte	30
Gruppen	30
Gruppeneditor	81

H

Hauptfenster	28
--------------------	----

I

Icons	2
Import und Export	57
Importdatei-Beispiel	57
Information	31
Integrität der Daten prüfen	33
Interne Fehler in der Datenbank	33

K

Karteikarten-Gruppen	30
Karteikarteneditor	81
Keine Sortierung	52
Knöpfe	24
Konstanten	89
Kontextmenü des mehrzeiligen Textfelds	46
Kontextmenü des Select-from-where-Listenfelds	47
Konventionen für Dateinamen	8
Kopieren von MUIbase	1

L

Linux-Version	2
Lisp-Aufbau	87
Liste veralteter Funktionen	166
Listenfunktionen	125
Lizens	1
Log	42
Log ansehen	44
Log exportieren	43
Log importieren	43
Log löschen	43
Log-Tabelle	42
Logbeschriftung setzen	43
logLabel	159

M

Mac-OS-Version	2
mainWindowTitle	160
Masken	29
Mathematik-Funktionen	107
Mehrfach-zu-Mehrfach-Beziehungen	26
Mehrzeilige Textfelder	23
Menü Einstellungen	175
Menü Log	176
Menü Programm	177
Menü Projekt	174
Menü Tabelle	176
Menüs	174
Menu Hilfe	178
MUI	2, 37
MUI-Voreinsteller	37
MUIbase beenden	8
MUIbase für Amiga installieren	7
MUIbase für Linux installieren	6
MUIbase für Mac OS installieren	6

MUIbase für Windows installieren	6
MUIbase starten	8

N

n:m-Beziehungen	26
Namenskonventionen for program symbols	88
Neu sortieren aller Datensätze	54
Neue Tabelle	67
Neuer Datensatz	45
Neues Feld	69
Neues Projekt	31
NList	2

O

Oberflächenfunktionen	147
onAdminMode	159
onChange	159
onClose	158
onOpen	158
onReload	158

P

Panels	29
Programm-Ausgabedatei	40
Programm-Debug-Information	39
Programm-Include-Verzeichnis	40
Programmarten	87
Programmeditor	83
Programmieren	83
Programmiersprache	86
Programmquellen	39
Programmsteuerungsfunktionen	93
Projekt öffnen	32
Projekt leeren	32
Projekt schließen	34
Projekt speichern	32
Projekte	20
Projekteinstellungen	37
Projektfunktionen	149
Protokollieren aktivieren	43

R

Referenzfilter	51
Relationsoperatoren	105
Relative Pfade	38
Reorganisieren	32

S

Select-from-where Abfragen	60
Sortieren	52
Sortierung ändern	53
Speichern & Umschichten bestätigen	39
Speicherverbrauch	25
Spenden	1
Struktur exportieren	82
Struktureditor	67
Suchen	55
Suchfenster	55
Suchmusterbeispiele	56
Systemfunktionen	150

T

Tabellenobjekteditor	73
Tabellarische Ansicht	60
Tabellen	20
Tabellen ändern	68
Tabellen erstellen	67
Tabellen löschen	68
Tabellen sortieren	68
Tabellenfunktionen	143
Tabellenverwaltung	67
Texteditor	80
TextEditor	2
Textobjekte	29
Trigger-Funktionen sortieren	40
Tutorial	10
Typabhängige Einstellungen	69
Typabhängige Einstellungen für Felder	76
Typaussagen	100
Typdeklarierer	156
Typumwandlungsfunktionen	101

V

Veraltete Funktionen	39, 166
Vergleichsfunktion	161
Vergleichsfunktionen	105
Verzichtserklärung	1
Virtuelle Felder programmieren	163
Virtuelles Feld	23
Vordefinierte Konstanten	154
Vordefinierte Variablen	154
Vorgabedatensatz	21
Vorverarbeitung	84
Vorwärts/Rückwärts suchen	55

W

Warum Lisp?	87
Weiterspringen bei Enter	37
Windows-Version	2

Z

Zeichenketten	21
Zeichenkettenfunktionen	111
Zeichensatznamenfelder	22
Zeitfelder	23
Zeitfunktionen	123
Zwischenraumeditor	80
Zwischenraumobjekte	30