

MUIbase

Une base de données relationnelle programmable
Version 2.9

27 mai 2010

Steffen Gutmann

ATTENTION cette traduction française correspond à la version 2.8 de MUIbase où certaines parties ne sont donc pas à jour, veuillez nous en excuser et vous reporter à la version anglaise (ou allemande).

Traduction française par Alexandre Balaban, Lionel Müller, Gilles Mathevet, Philippe Ferrucci et Pascal Marcelin 2007-2010

Copyright © 2010 Steffen Gutmann

Vous avez l'autorisation de faire et distribuer des copies de ce manuel à condition que la notification de copyright et la présente autorisation soient préservées sur toutes ces copies.

Table des matières

1	Conditions de copie de MUIbase	1
1.1	Faire un don	1
1.2	Distribution	1
1.3	Liste de discussion	1
1.4	Mise en garde	1
1.5	Parties externes	2
1.5.1	Versions Linux et Windows	2
1.5.2	Version Amiga	2
2	Bienvenu sur MUIbase	4
3	Pour commencer	5
3.1	Installation de MUIbase sur Linux	5
3.2	Installation de MUIbase sur Windows	5
3.3	Installation de MUIbase sur Amiga	6
3.4	Mise à jour d'une version précédente	6
3.5	Démarrez MUIbase	7
3.6	Quitter MUIbase	7
3.7	Conventions de nommage de fichiers sur Linux et Windows	7
4	Travaux dirigés	9
4.1	Comment fonctionne MUIbase	9
4.2	Pour commencer un Projet : l'éditeur de structure	9
4.3	Ajouter une table	9
4.4	Ajouter un Champ	10
4.5	Afficher le Projet	10
4.6	Ajouter deux champs de type référence	12
4.7	Ajouter des enregistrements	12
4.8	Filtre	13
4.9	Requêtes	13
4.10	Ajouter une Table avec un Mémo et un champ de type bouton	14
4.11	MUIbase, programmation pour faire une ascendance	15
4.12	Programmation dans MUIbase pour énumérer les enfants d'une personne	16
5	Concepts de base	19
5.1	Projets	19
5.2	Tables	19
5.3	Enregistrements	20
5.4	Champs	20
5.5	Types de champ	20

5.5.1	Champs texte	20
5.5.2	Champs entiers	21
5.5.3	Champs réels	21
5.5.4	Champs booléens	21
5.5.5	Champs choix	21
5.5.6	Champs date	22
5.5.7	Champs heure	22
5.5.8	Champs mémo	22
5.5.9	Champs Référence	22
5.5.10	Champs virtuels	22
5.5.11	Boutons	23
5.6	Tableau des types de champ	23
5.7	Consommation mémoire	24
5.8	Associations	24
5.8.1	Associations \message Character missing in OT1 encoding: LEFT-POINTING DOUBLE ANGLE QUOTATION MARK. Un à Un \message Character missing in OT1 encoding: RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK.	25
5.8.2	Associations \message Character missing in OT1 encoding: LEFT-POINTING DOUBLE ANGLE QUOTATION MARK. Un à plusieurs \message Character missing in OT1 encoding: RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK.	25
5.8.3	Associations \message Character missing in OT1 encoding: LEFT-POINTING DOUBLE ANGLE QUOTATION MARK. Plusieurs à plusieurs \message Character missing in OT1 encoding: RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK.....	25
5.9	Interface graphique	27
5.9.1	Fenêtres	27
5.9.2	Fiches	27
5.9.3	Panels	28
5.9.4	Objets de champ	28
5.9.5	Objets texte	28
5.9.6	Images	28
5.9.7	Objets d'espacement	28
5.9.8	Groupes	29
5.9.9	Objets balance	29
5.9.10	Registres	29

6	Gestion des projets	30
6.1	Format de fichier	30
6.2	Informations	30
6.3	Nouveau projet	30
6.4	Nettoyer le projet	30
6.5	Ouvrir le projet	31
6.6	Enregistrer le projet	31
6.7	Modes Administrateur et Utilisateur	32
6.8	Vérifier l'intégrité des données	32
6.9	Décharger les enregistrements	33
6.10	Fermer le Projet	33
7	Préférences	34
7.1	Réglages utilisateur	34
7.1.1	Formats	34
7.1.2	Editeur externe	34
7.1.3	Visionneuse externe	35
7.1.4	Boutons dans cycle de tabulation	35
7.1.5	Champ suivant via <Entrée>	35
7.2	Confirmer la sortie	36
7.3	MUI	36
7.4	Réglages dépendants du projet	36
7.4.1	Cache d'enregistrements	36
7.5	Confirmer la suppression d'enregistrement	37
7.5.1	Chemins relatifs au projet	37
7.5.2	Confirmer enregistrement et réorganisation	37
7.5.3	Code source du programme	37
7.5.4	Nettoyer les sources des programmes externes	38
7.5.5	Information de débogage	38
7.5.6	Fonctions obsolètes	38
7.5.7	Trier les déclencheurs	38
7.5.8	Répertoire d'inclusion	39
7.5.9	Fichier de sortie de programme	39
7.6	Enregistrer comme défaut	39
8	Edition d'enregistrement	41
8.1	Objet actif	41
8.2	Ajout d'enregistrement	41
8.3	Modification d'enregistrement	41
8.3.1	Champ texte avec bouton pop-up	41
8.3.2	Saisie de valeurs entières	42
8.3.3	Saisie de valeurs booléennes	42
8.3.4	Saisie de valeurs de choix	42
8.3.5	Saisie de valeurs de date	42
8.3.6	Saisie de valeurs horaires	42
8.3.7	Menu contextuel des Mémos	42
8.3.8	Saisie de valeur de type Référence	43

8.3.9	Saisie de valeur NIL	43
8.4	Suppression d'enregistrement	43
8.5	Parcourir les enregistrements	44
9	Filtre	45
9.1	Filtre d'enregistrement	45
9.1.1	Expression de filtrage	45
9.1.2	Modifier les filtres	45
9.1.3	Exemples de filtre	46
9.2	Filtre de référence	46
10	Ordre	47
10.1	Tri vide	47
10.2	Tri par champs	47
10.3	Tri par fonction	48
10.4	Changer le tri	48
10.5	Réordonner tous les enregistrements	49
11	Recherche	50
11.1	Boîte de recherche	50
11.2	Rechercher en avant/en arrière	50
11.3	Exemples de motif de recherche	51
12	Importer et Exporter	52
12.1	Format de fichier	52
12.2	Exemple de fichier d'import	52
12.3	Importer des enregistrements	53
12.4	Exporter des enregistrements	53
13	Traitement des données	55
13.1	Requêtes Select-from-where	55
13.2	Éditeur de requêtes	55
13.3	Impression de requêtes	56
13.4	Exemples de requêtes	58

14	Editeur de structure	60
14.1	Table Management	60
14.1.1	Création de tables	60
14.1.2	Modification de tables	61
14.1.3	Suppression de tables	61
14.1.4	Tri des tables	61
14.2	Gestion des champs	61
14.2.1	Création de champs	62
14.2.2	Type Specific Settings	62
14.2.3	Label Editor	63
14.2.4	Copying Attributes	64
14.2.5	Changing Attributes	64
14.2.6	Deleting Attributes	64
14.2.7	Sorting Attributes	65
14.3	Gestion de l'affichage	65
14.3.1	Champ d'affichage	65
14.3.2	Editeur de panel	66
14.3.3	Attribute Object Editor	67
14.3.4	Réglages liés au type	68
14.3.5	Text Editor	72
14.3.6	Image Editor	72
14.3.7	Space Editor	72
14.3.8	Group Editor	72
14.3.9	Register Group Editor	73
14.3.10	Window Editor	73
14.4	Export Structure	74
15	Programmation de MUIbase	75
15.1	Editeur de programme	75
15.2	Code source externe	75
15.3	Préprocesseur	76
15.3.1	#define	76
15.3.2	#undef	77
15.3.3	#include	77
15.3.4	#if	77
15.3.5	#ifdef	77
15.3.6	#ifndef	77
15.3.7	#elif	78
15.3.8	#else	78
15.3.9	#endif	78
15.4	Langage de programmation	78
15.4.1	Pourquoi Lisp ?	79
15.4.2	Syntaxe Lisp	79
15.4.3	Types de programmes	79
15.4.4	Conventions de nommage	80
15.4.5	Accéder aux enregistrements	80
15.4.6	Types de données pour programmer	81
15.4.7	Constantes	81

15.4.8	Typographie utilisée.....	83
15.5	Commandes de définition.....	83
15.5.1	DEFUN.....	83
15.5.2	DEFUN*.....	84
15.5.3	DEFVAR.....	84
15.5.4	DEFVAR*.....	84
15.6	Structures de contrôle.....	85
15.6.1	PROGN.....	85
15.6.2	PROG1.....	85
15.6.3	LET.....	85
15.6.4	SETQ.....	86
15.6.5	SETQ*.....	86
15.6.6	FUNCALL.....	87
15.6.7	APPLY.....	87
15.6.8	IF.....	87
15.6.9	CASE.....	87
15.6.10	COND.....	88
15.6.11	DOTIMES.....	88
15.6.12	DOLIST.....	89
15.6.13	DO.....	89
15.6.14	FOR ALL.....	90
15.6.15	NEXT.....	90
15.6.16	EXIT.....	91
15.6.17	RETURN.....	91
15.6.18	HALT.....	92
15.6.19	ERROR.....	92
15.7	Prédicats de typage.....	92
15.8	Fonctions de conversion de type.....	93
15.8.1	STR.....	93
15.8.2	MEMO.....	93
15.8.3	INT.....	94
15.8.4	REAL.....	94
15.8.5	DATE.....	95
15.8.6	TIME.....	95
15.9	Fonctions booléennes.....	96
15.9.1	AND.....	96
15.9.2	OR.....	96
15.9.3	NOT.....	96
15.10	Fonctions de comparaison.....	97
15.10.1	Opérateurs relationnels.....	97
15.10.2	CMP.....	97
15.10.3	CMP*.....	98
15.11	Fonctions mathématiques.....	98
15.11.1	Ajouter des valeurs.....	98
15.11.2	Soustraire des valeurs.....	98
15.11.3	1+.....	99
15.11.4	1-.....	99
15.11.5	Multiplier des valeurs.....	99

15.11.6	Diviser des valeurs	99
15.11.7	DIV	99
15.11.8	MOD	100
15.11.9	MAX	100
15.11.10	MIN	100
15.11.11	ABS	100
15.11.12	TRUNC	100
15.11.13	ROUND	100
15.11.14	RANDOM	101
15.12	Fonctions sur les chaînes	101
15.12.1	LEN	101
15.12.2	LEFTSTR	101
15.12.3	RIGHTSTR	101
15.12.4	MIDSTR	102
15.12.5	SETMIDSTR	102
15.12.6	INSMIDSTR	102
15.12.7	INDEXSTR	102
15.12.8	INDEXSTR*	102
15.12.9	INDEXBRK	103
15.12.10	INDEXBRK*	103
15.12.11	RINDEXSTR	103
15.12.12	RINDEXSTR*	103
15.12.13	RINDEXBRK	103
15.12.14	RINDEXBRK*	103
15.12.15	REPLACESTR	104
15.12.16	REPLACESTR*	104
15.12.17	REMCHARS	104
15.12.18	TRIMSTR	104
15.12.19	WORD	105
15.12.20	WORDS	105
15.12.21	STRTOLIST	105
15.12.22	LISTTOSTR	105
15.12.23	CONCAT	106
15.12.24	CONCAT2	106
15.12.25	COPYSTR	106
15.12.26	SHA1SUM	106
15.12.27	UPPER	107
15.12.28	LOWER	107
15.12.29	ASC	107
15.12.30	CHR	107
15.12.31	LIKE	107
15.12.32	SPRINTF	108
15.13	Fonctions sur les mémos	110
15.13.1	LINE	110
15.13.2	LINES	110
15.13.3	MEMOTOLIST	111
15.13.4	LISTTOMEMO	111
15.13.5	FILLMEMO	111

15.13.6	FORMATMEMO	112
15.13.7	INDENTMEMO	112
15.14	Fonctions sur la date et l'heure	112
15.14.1	DAY	112
15.14.2	MONTH	112
15.14.3	YEAR	112
15.14.4	DATEDMY	113
15.14.5	MONTHDAYS	113
15.14.6	YEARDAYS	113
15.14.7	ADDMONTH	113
15.14.8	ADDYEAR	114
15.14.9	TODAY	114
15.14.10	NOW	114
15.15	Liste des Fonctions	114
15.15.1	CONS	114
15.15.2	LIST	115
15.15.3	LENGTH	115
15.15.4	FIRST	115
15.15.5	REST	115
15.15.6	LAST	115
15.15.7	NTH	115
15.15.8	APPEND	116
15.15.9	REVERSE	116
15.15.10	MAPFIRST	116
15.15.11	SORTLIST	116
15.15.12	SORTLISTGT	117
15.16	Fonctions de demande de saisie	117
15.16.1	ASKFILE	117
15.16.2	ASKDIR	117
15.16.3	ASKSTR	118
15.16.4	ASKINT	118
15.16.5	ASKCHOICE	118
15.16.6	ASKCHOICESTR	119
15.16.7	ASKOPTIONS	120
15.16.8	ASKBUTTON	121
15.16.9	ASKMULTI	121
15.17	Fonctions d'entrée/sortie	123
15.17.1	FOPEN	123
15.17.2	FCLOSE	124
15.17.3	stdout	124
15.17.4	PRINT	124
15.17.5	PRINTF	124
15.17.6	FPRINTF	125
15.17.7	FERROR	125
15.17.8	FEOF	125
15.17.9	FSEEK	125
15.17.10	FTELL	126
15.17.11	FGETCHAR	126

15.17.12	FGETCHARS	126
15.17.13	FGETSTR	126
15.17.14	FGETMEMO	126
15.17.15	FPUTCHAR	127
15.17.16	FPUTSTR	127
15.17.17	FPUTMEMO	127
15.17.18	FFLUSH	127
15.18	Les fonctions sur les enregistrements	127
15.18.1	NEW	127
15.18.2	NEW*	128
15.18.3	DELETE	128
15.18.4	DELETE*	129
15.18.5	DELETEALL	129
15.18.6	GETMATCHFILTER	129
15.18.7	SETMATCHFILTER	129
15.18.8	GETISSORTED	130
15.18.9	SETISSORTED	130
15.18.10	RECNUM	130
15.18.11	COPYREC	131
15.19	Fonctions sur les champs	131
15.19.1	ATTRNAME	131
15.19.2	MAXLEN	131
15.19.3	GETLABELS	131
15.19.4	SETLABELS	132
15.20	Fonctions sur les tables	132
15.20.1	TABlename	132
15.20.2	GETORDERSTR	132
15.20.3	SETORDERSTR	133
15.20.4	REORDER	133
15.20.5	REORDERALL	134
15.20.6	GETFILTERACTIVE	134
15.20.7	SETFILTERACTIVE	134
15.20.8	GETFILTERSTR	134
15.20.9	SETFILTERSTR	134
15.20.10	RECORDS	135
15.20.11	RECORD	135
15.20.12	SELECT	135
15.21	Les fonctions GUI	136
15.21.1	SETCURSOR	136
15.21.2	GETWINDOWOPEN	137
15.21.3	SETWINDOWOPEN	137
15.21.4	GETVIRTUALLISTACTIVE	137
15.21.5	SETVIRTUALLISTACTIVE	137
15.22	Les fonctions projets	137
15.22.1	PROJECTNAME	137
15.22.2	CHANGES	138
15.22.3	GETADMINMODE	138
15.22.4	SETADMINMODE	138

15.22.5	ADMINPASSWORD	138
15.23	Les fonctions système	138
15.23.1	EDIT	139
15.23.2	EDIT*	139
15.23.3	VIEW	139
15.23.4	VIEW*	139
15.23.5	SYSTEM	139
15.23.6	SYSTEM*	140
15.23.7	STAT	140
15.23.8	TACKON	140
15.23.9	FILENAME	140
15.23.10	DIRNAME	141
15.23.11	MESSAGE	141
15.23.12	COMPLETEMAX	141
15.23.13	COMPLETEADD	141
15.23.14	COMPLETE	142
15.23.15	GC	142
15.23.16	PUBSCREEN	142
15.24	Les variables prédéfinies	142
15.25	Les constantes prédéfinies	143
15.26	Paramètres fonctionnels	143
15.27	Spécificateurs de type	144
15.28	Sémantique des expressions	145
15.29	Déclenchement de fonction	146
15.29.1	onOpen	146
15.29.2	onClose	146
15.29.3	onAdminMode	146
15.29.4	onChange	147
15.29.5	Déclencheur de création	147
15.29.6	Déclencheur de suppression	148
15.29.7	Fonction de comparaison	148
15.29.8	Déclencheur de champ	149
15.29.9	Programmation de champs virtuels	150
15.29.10	Fonction de calcul d'activation	150
15.29.11	Déclencheur de double-clic	151
15.29.12	Calculer les étiquettes des listes	152
15.29.13	Calculer les enregistrements référencés	152
15.30	Liste des fonctions obsolètes	152
16	Interface ARexx	154
16.1	Nom du port	154
16.2	Syntaxe des commandes	154
16.3	Codes retour	155
16.4	Quit	156
16.5	Hide	156
16.6	Show	156
16.7	Info	156
16.8	Help	156

16.9	Compile	156
16.10	Connect	157
16.11	Disconnect	157
16.12	Connections	158
16.13	Eval	158
16.14	Transaction	159
16.15	Commit	159
16.16	Rollback	160
Remerciements		161
Auteur		162
Index des fonctions		163
Index des concepts		166

1 Conditions de copie de MUIbase

MUIbase est Copyright © 1998-2010 Steffen Gutmann. Tous droits réservés.

MUIbase est un logiciel à sources ouverts distribué sous les termes de la licence générale publique GNU (GPL).

1.1 Faire un don

MUIbase est complètement gratuit. Cependant, si vous aimez le projet et que vous voulez le supporter, une donation sera toujours la bienvenue. Pour plus d'informations sur la façon de faire un don à MUIbase, veuillez s'il vous plaît visiter <http://www.sourceforge.net/projects/muibase>.

1.2 Distribution

Les binaires de la dernière version de MUIbase peuvent être téléchargés sur <http://muibase.sourceforge.net>. Tandis que le code source est quant à lui hébergé sur <http://www.sourceforge.net/projects/muibase>.

Vous avez le droit de distribuer MUIbase, mais pour cela, il faut que les archives de MUIbase soient exactement comme celles que vous avez reçues avec tous les fichiers associés inclus.

Aucune circonstance ne vous permet de percevoir des honoraires sur la copie ou le coût d'expédition pour distribuer les archives de MUIbase sans consentement écrit express de la part du détenteur des droits d'auteur.

1.3 Liste de discussion

Une liste de discussion spécifique à MUIbase se trouve sur <http://www.groups.yahoo.com/groups/muibase-1>. Venez nous rejoindre, vous êtes tous les bienvenus.

Veuillez envoyer les rapports de bogue ou les demandes d'évolution sur cette liste de discussion.

1.4 Mise en garde

CE LOGICIEL EST FOURNI PAR L'AUTEUR ET LES CONTRIBUANTS EN L'ETAT ET SANS IMPLIQUER LA MOINDRE GARANTIE. SONT ÉGALEMENT DÉMENTIES LES GARANTIES IMPLICITES DE LA VALEUR MARCHANDE ET L'ADAPTATION À UN USAGE PARTICULIER.

L'AUTEUR OU LES CONTRIBUANTS NE POURRONT EN AUCUN CAS ÊTRE TENUS POUR RESPONSABLES DES DOMMAGES DIRECTS, INDIRECTS, FORTUITS, SPÉCIAUX, EXEMPLAIRES OU CONSÉCUTIFS (COMPRENANT, MAIS NON LIMITÉ À, LA FOURNITURE DE BIENS OU SERVICES DE RECHANGE ; LA PERTE DE JOUISSANCE, DE DONNÉES OU DE BÉNÉFICES ; OU D'UNE INTERRUPTION DES ACTIVITÉS) SANS ÉGARD À LA CAUSE OU À LA THÉORIE DE RESPONSABILITÉ, QUELLE SOIT CONTRACTUELLE, STRICTE OU PRÉVUE PAR LA LOI (DONT LA NÉGLIGENCE OU TOUTE AUTRE FORME), DÉCOULANT DE QUELQUE FAÇON QUE CE SOIT DE L'UTILISATION DE CE LOGICIEL, MÊME

SI VOUS AVEZ ÉTÉ PRÉVENUS DE LA POSSIBILITÉ QUE DE TELS DOMMAGES SURVIENNENT.

1.5 Parties externes

MUIbase utilise des bibliothèques externes et d'autres matériels en fonction du système d'exploitation.

1.5.1 Versions Linux et Windows

Sous Linux et Windows MUIbase utilise la version 2.6 (ou supérieure) du Gimp Toolkit (GTK+), copyright © 2007-2008 The GTK+ Team. GTK est une trousse à outils permettant de développer des interfaces utilisateurs graphiques (GUI) et est distribué sous GNU Library General Public License (LGPL). Pour plus d'informations, voir le site <http://www.gtk.org>.

Le script d'installation de Windows pour MUIbase est basé sur le script d'installation de Gimp qui est copyright par Jernej Simoncic.

1.5.2 Version Amiga

La version Amiga de MUIbase utilise

MUI - MagicUserInterface

(c) Copyright 1992-2008 par Stefan Stuntz

MUI est un système pour générer et maintenir les interfaces graphiques utilisateurs. À l'aide d'un programme de préférences, l'utilisateur d'une application a la capacité de l'adapter selon son goût personnel.

MUI est distribué sous forme de shareware. Pour obtenir la distribution complète contenant un bon nombre d'exemples et plus d'informations sur l'enregistrement, recherchez s'il vous plaît le fichier appelé ~muiXXusr.lha~ (XX représente le dernier nombre de la version) dans vos tableaux de liens locaux ou sur les disques domaine public

Si vous voulez vous enregistrer directement, sentez-vous libre d'envoyer

DM 30.- ou US\$ 20.-

à

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

Support et enregistrement en ligne sont disponibles sur

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

Sur Amiga, MUIbase utilise NList.mcc, copyright © 2001-2008 NList Open Source Team. Se connecter sur <http://www.sourceforge.net/projects/nlist-classes> pour de plus amples informations ou la dernière version.

Sur Amiga, MUIbase utilise aussi BetterString.mcc, copyright © 2005-2009 BetterString Open Source Team. Se connecter sur <http://www.sourceforge.net/projects/bstring-mcc> pour de plus amples informations ou la dernière version.

Sur Amiga, MUIbase utilise également TextEditor.mcc, copyright © 2005-2009 TextEditor Open Source Team. Se connecter sur <http://www.sourceforge.net/projects/texteditor-mcc> pour de plus amples informations ou la dernière version.

Sur Amiga, MUIbase utilise aussi codesets.library, copyright © 2005-2009 codesets.library Open Source Team. Se connecter sur <http://www.sourceforge.net/projects/codesetslib> pour de plus amples informations ou la dernière version.

Quelques icônes utilisées dans MUIbase pour la distribution Amiga sont extraites du jeu d'icônes DefaultIcons et sont sous copyright © Michael-Wolfgang Hohmann et Angela Schmidt.

2 Bienvenu sur MUIbase

MUIbase est un système de base de données rapide, flexible et programmable comportant une interface graphique utilisateur (GUI). Il est destiné aux utilisateurs avancés voulant contrôler les données d'une manière confortable et puissante. MUIbase peut contrôler n'importe quel type de données, comme par exemple : adresses, collections de CD, de films ou de photos, votre arbre généalogique, votre revenu et vos dépenses et beaucoup plus. La puissance de MUIbase est liée à son interface graphique claire et puissante, ainsi qu'à ses possibilités de programmation. La programmation de MUIbase permet de traiter les données de diverses façons, par exemple : calculs automatiques sur les saisies de l'utilisateur, génération de rapports, importation et exportation de données, etc. Il peut, par exemple, être employé pour calculer le montant total de vos revenus, le montant total du temps enregistré sur un CD ou pour créer et imprimer automatiquement un publipostage à vos clients.

MUIbase offre les caractéristiques suivantes :

- Nombre illimité de projets, tables, champs et enregistrements.
- Les attributs peuvent être de type "chaîne", mémo (multi ligne texte), entier, réel, date, heure, booléen, choix (sélection d'un élément dans une liste), référence (faire référence facilement à un enregistrement d'une autre table), bouton (pour exécuter des programmes MUIbase) et virtuel (valeur calculée à la volée).
- Le type chaîne peut également contrôler des listes de chaînes, fichiers, et polices de caractères. Une chaîne peut se rapporter à une image externe qui sera montrée dans l'interface graphique utilisateur.
- Chargement dynamique des enregistrements. Les enregistrements qui ne sont pas nécessaires peuvent être effacés de la mémoire (lorsque par exemple la mémoire devient limitée).
- Programmation. Avec le langage de programmation facile et puissant de MUIbase, des tâches complexes peuvent être mises en application. Le langage inclus également le requêtage `SELECT FROM WHERE` pour une recherche confortable et rapide des données.
- Ordonnancement des enregistrements par n'importe quelle combinaison de champs.
- Fonctions de recherche et de filtrage flexibles et puissantes.
- Editeur de requête permettant de taper et manipuler des requêtes `SELECT FROM WHERE`. Les requêtes peuvent être sauveées et les résultats imprimés.
- Importation et exportation aisée.
- Documentation complète comprenant le manuel de l'utilisateur et de programmation en HTML et PDF (plus AmigaGuide sur Amiga)
- La version Amiga fournit une interface ARexx puissante pour accéder à MUIbase à partir d'autres programmes. L'interface ARexx offre un mécanisme de transaction semblable à d'autres bases de données relationnelles.
- L'indépendance du système d'exploitation. MUIbase est disponible pour Linux, Windows et Amiga. Le code source est disponible en tant que projet Source Forge.

3 Pour commencer

Ce chapitre décrit le matériel et le logiciel nécessaires pour exécuter MUIbase sur votre ordinateur, la procédure pour l'installer, le mettre à jour, ainsi que comment démarrer et quitter MUIbase.

3.1 Installation de MUIbase sur Linux

Vous pouvez installer et lancer MUIbase sur un ordinateur compatible Intel 386 (sous Linux). Votre système devrait avoir un serveur X11 et un GTK+ récents (version 2.6 ou supérieure) installé. L'espace exigé pour le disque dur est modéré (moins de 10 Mo).

MUIbase pour Linux i386 est distribué à la fois sous la forme de paquetage Debian (p. ex. 'muibase_2.5_i386.deb' pour Ubuntu et Debian) ou sous la forme d'archive rpm (p. ex. 'MUIbase-2.5.i386.rpm' pour Fedora ou Mandriva). Les deux versions peuvent être téléchargées à partir de la page d'accueil de MUIbase <http://muibase.sourceforge.net>.

Habituellement MUIbase s'installe ou se met à jour lui-même automatiquement après le téléchargement sur votre ordinateur. Dans le cas où l'installation ne se lancerait pas automatiquement, les commandes suivantes exécutées sous l'identité du super utilisateur (root) depuis un shell installent ou mettent à jour MUIbase sur un système Linux sous Debian :

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

où *muibase.deb* est le paquetage Debian téléchargé.

Sur un système basé sur Redhat (RPM) les commandes sont :

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

où *MUIbase.rpm* est le fichier rpm de MUIbase téléchargé,

Après une installation réussie vous devriez voir un nouvel élément dans l'arborescence du menu "Applications - Office" de (Gnome ou KDE) votre ordinateur de bureau. À ce point l'archive téléchargée n'est plus nécessaire et peut être supprimée.

3.2 Installation de MUIbase sur Windows

MUIbase pour Windows fonctionne sur les ordinateurs compatibles Intel 386 qui utilisent les versions 32bit de Windows 2000, Windows XP ou Windows Vista. MUIbase exige que les bibliothèques gtk-2.0 pour Windows soient installées sur votre ordinateur. Habituellement, dans la plupart des installations de Windows, ces bibliothèques ne sont pas présentes, à moins que vous ayez déjà installé d'autres logiciels les exigeant également (par exemple : Gimp). L'espace nécessaire pour installer MUIbase sur le disque dur est modéré (moins de 10 Mo).

Le fichier d'installation de MUIbase est distribué sous la forme d'un exécutable Windows, par exemple : 'MUIbase-2.5-setup.exe' que vous pouvez télécharger sur la page d'accueil de MUIbase <http://muibase.sourceforge.net>. Quand vous lancez le fichier d'installation, une vérification de votre système a lieu pour s'assurer que les bibliothèques GTK+ sont bien présentes. Dans le cas où elles ne seraient pas retrouvées sur votre ordinateur, un message s'afficherait vous incitant à télécharger ces bibliothèques sur Internet et à les installer. L'installation de MUIbase reprend alors et vous demandera plusieurs options comme où installer MUIbase et quelle est la configuration à utiliser. Au cas où

vous auriez une version précédente de MUIbase installée sur votre ordinateur, la procédure d'installation permet la mise à jour vers la nouvelle version.

Après une installation réussie, le menu 'Démarez' de Windows devrait contenir un nouveau raccourci pour démarrer MUIbase. Vous pouvez maintenant enlever le fichier d'installation de MUIbase, il n'est plus nécessaire.

3.3 Installation de MUIbase sur Amiga

MUIbase pour Amiga fonctionne sur la version 3.0 (ou supérieure) de l'OS Amiga et exige un processeur 68020 ou supérieur. En outre, la version 3.8 (ou supérieure) de MUI doit être présente sur votre système. Un minimum de 4 Mo de RAM et de 10 Mo d'espace libre sur votre disque dur sont nécessaires. On m'a signalé que le binaire m68k de MUIbase pour Amiga se lance également sur UAE, MorphOS et Amiga OS4.

Depuis MUIbase v2.3 des exécutables natifs pour MorphOS et Amiga OS4 sont disponibles dans l'archive distribuée.

Pour installer MUIbase sur votre ordinateur Amiga vous avez besoin du programme "Installer" de Commodore. On peut trouver cet outil sur Aminet dans le tiroir '/pub/aminet/util/misc'. Assurez-vous que vous utilisez bien la version 43.3 (ou supérieure), autrement le script d'installation pourrait échouer.

MUIbase est distribué sous la forme d'archives lha, par exemple 'MUIbase-2.5.lha' et peut être téléchargé à partir de la page d'accueil de MUIbase <http://muibase.sourceforge.net>. Pour installer MUIbase sur votre ordinateur, vous devez décompacter ces archives dans un tiroir provisoire. Ne pas les décompacter dans le tiroir cible !

Double cliquez sur le script d'installation de MUIbase 'Install-MUIbase' et suivez les instructions. Le script demande le tiroir où le logiciel devra être copié. Ne pas écrire le chemin d'accès du tiroir où vous avez décompacté les archives de MUIbase. Le script est également capable de mettre à jour une installation existante de MUIbase, il suffit pour cela de lui fournir le chemin d'accès du tiroir où vous avez précédemment installé MUIbase.

Après une installation réussie, vous trouverez sur votre système le (nouveau) tiroir contenant le programme MUIbase avec tous les fichiers nécessaires et quelques échantillons de projets. De plus, une assignation 'MUIbase:' se rapportant à ce tiroir a été rajoutée sur votre fichier système 'S:User-Startup'. Vous pouvez enlever les archives lha et les fichiers temporaires, ils ne sont plus nécessaires.

3.4 Mise à jour d'une version précédente

En installant MUIbase sur Linux, Windows ou Amiga, vous pouvez mettre à jour la version précédente de MUIbase ou la réinstaller comme décrit ci-dessus. Pendant la nouvelle installation, tous les fichiers nécessaires sont remplacés. Ce qui inclut tous les échantillons de projets dans le tiroir 'Demos'. Il n'est donc pas recommandé de placer vos propres projets dans ce tiroir, ni d'employer l'un de ces projets pour contrôler vos propres données : elles risqueraient d'être écrasées pendant la réinstallation ou la mise à jour de MUIbase.

Il est préférable de contrôler vos propres projets en les plaçant dans un tiroir séparé. Ce tiroir peut être à l'intérieur du tiroir où vous avez installé MUIbase car la procédure d'installation ne modifiera pas et ne supprimera pas les fichiers situés dans des tiroirs inconnus.

3.5 Démarrez MUIbase

MUIbase peut être démarré à partir de votre environnement graphique ou à partir d'une commande shell. Si vous lancez Gnome ou KDE sur Linux, vous pouvez démarrer MUIbase en choisissant l'article correspondant dans le menu 'Applications - Office'. Sur Windows vous trouverez MUIbase dans le menu 'Démarrer'. Sur le Workbench (Amiga), vous pouvez double-cliquer sur l'icône MUIbase ou sur l'icône du projet MUIbase, il se chargera automatiquement après avoir démarré MUIbase (vous pouvez également sélectionner plusieurs projets de MUIbase à l'aide de la touche shift puis double-cliquer sur le dernier).

En démarrant MUIbase à partir d'une commande shell (par exemple : 'xterm' sur Linux, 'CMD' sur Windows, 'CLI' sur Amiga) vous pouvez inclure des arguments dans la ligne de commande et charger les projets optionnels. Il y a deux formes de base pour démarrer MUIbase à partir d'une commande shell :

```
MUIbase [projet1 ...]  
MUIbase -n
```

La première forme démarre MUIbase et charge les projets optionnels spécifiés par leurs noms de fichiers *project1* La seconde forme démarre MUIbase sans interface utilisateur graphique. Ce qui peut être utile pour démarrer MUIbase comme serveur en tâche de fond et pour accéder à son interface externe (par exemple ARexx sur Amiga, voir [Chapitre 16 \[Interface ARexx\]](#), page 154).

3.6 Quitter MUIbase

Pour quitter MUIbase sélectionnez l'article du menu 'Projet - Quitter' ou fermez tous les projets ouverts.

3.7 Conventions de nommage de fichiers sur Linux et Windows

MUIbase pour Linux et Windows a quelques conventions spéciales pour les noms de fichiers qu'on ne trouve habituellement pas dans d'autres logiciels (sur ces systèmes).

Lorsque MUIbase démarre, une variable d'environnement 'MUIbase' se réfère au tiroir d'installation de MUIbase. Sur Linux, c'est le tiroir '/usr/share/MUIbase'. Sur Windows c'est le tiroir indiqué pendant l'installation du logiciel.

Lors de l'interprétation des noms de fichiers (entrés par l'utilisateur ou ceux présents dans cette documentation), un nom de fichier est analysé à la recherche de variables d'environnement. Si un nom de fichier contient le signe dollar '\$' alors les caractères suivants jusqu'au premier caractère non alphanumérique sont interprétés comme étant une variable d'environnement et, si cette variable d'environnement est positionnée, cette partie du nom de fichier est remplacée par le contenu de la variable d'environnement (autrement, la partie du nom de fichier, y compris le signe '\$', reste inchangée). Vous pouvez mettre des parenthèses autour de la variable d'environnement si elle contient des caractères non alphanumériques. Par exemple '\$HOME/fichier' est interprété comme le chemin d'accès où '\$HOME' a été remplacé par le nom de votre tiroir personnel.

En outre, les 'assignments' style Amiga sont manipulées de la façon suivante. Si un nom de fichier contient deux points ':' alors, tout ce qui se trouve avant les deux points est traité comme une variable d'environnement (appelée 'assignment') et est remplacé par son

contenu si la variable d'environnement est positionnée. Les '**Assignations**' devraient être au moins composées de 2 caractères, pour ne pas les confondre avec les noms des volumes Windows.

Ces conventions permettent, par exemple, de se référer à la base de données filmographique d'exemple qui se trouve à l'intérieur du tiroir d'installation de MUIbase en utilisant '**MUIbase:demos/Movie.mb**'.

Un autre exemple : lorsque vous avez positionné une variable d'environnement '**EXTERNAL_DATA**' se référant à un tiroir où vous stockez les données externes d'un de vos projets (comme des images, etc.). Vous pouvez alors accéder aux données à l'intérieur de ce tiroir en utilisant '**EXTERNAL_DATA:un-fichier**' et stocker de tels noms de fichier dans le projet de base de données.

4 Travaux dirigés

Ce chapitre contient une brève formation décrivant l'utilisation basique de MUIbase. Au fil de ces travaux dirigés un petit projet permettant la gestion de son arbre généalogique est développé. Après avoir appliqué toutes les étapes de ce cours, vous pourrez trouver le projet qui en résulte 'genealogie.mb' dans le tiroir 'Demos' de votre installation de MUIbase.

4.1 Comment fonctionne MUIbase

On peut dire que MUIbase fonctionne dans deux modes différents, l'édition d'enregistrement et l'édition de la structure.

Le mode d'édition d'enregistrement vous permet d'ajoutez, de changez ou de supprimez les enregistrements.

Le mode d'édition de structure vous laisse éditer la mise en page de votre base de données ainsi que les tables et champs qu'elle contient.

En outre, il y a l'éditeur de programme où vous saisissez les fonctions du programme qui seront exécutées automatiquement lors de l'entrée des données ou explicitement lors de l'appui sur un bouton du programme.

4.2 Pour commencer un Projet : l'éditeur de structure

Pour créer une base de données, vous devez d'abord définir son contenu. Pour cela, il faut ouvrir l'éditeur de structure en choisissant 'Editeur de structure' dans le menu 'Projet'. Voici les trois différentes sections que vous y trouverez :

'Tables' Ajouter, changer ou supprimer les tables de votre projet.

'Champs' Ajouter, changer ou supprimer les champs de la table actuellement sélectionnée.

'Affichage'
Indiquer la disposition graphique de votre base de données, c'est à dire la façon dont elle devra être montrée.

4.3 Ajouter une table

Tout d'abord nous avons besoin d'une table, appuyez sur le bouton 'Nouveau' situé sous la liste de la section 'Table'. Vous verrez alors la fenêtre 'Nouvelle table' s'ouvrir pour vous demander de saisir quelques données :

'Nom' Le nom de la table. Le nom doit commencer par une lettre majuscule et peut se composer d'un maximum de 20 caractères. Le nom peut être changé plus tard. Dans ce cours nous entrons le nom sur 'Personne' puisqu'il va contenir les noms de toutes les personnes dans cette base de données.

'Nombre d'enregistrements'
Une table peut soit contenir un seul enregistrement, soit un nombre illimité. Dans notre cas il faut le positionner illimité puisque nous allons ajouter plus d'une personne.

‘Déclencheurs’

L’ajout et la suppression des enregistrements peuvent être contrôlés par des fonctions utilisateur. C’est ici que vous pouvez indiquer la fonction à exécuter. Puisque nous n’avons encore écrit aucune fonction utilisateur, nous laissons les champs vides.

Quand c’est fait, appuyez juste sur le bouton ‘OK’ et vous aurez votre première table, ‘Personne’.

4.4 Ajouter un Champ

Maintenant nous avons besoin d’un champ de type chaîne pour cette table, appuyez sur le bouton ‘Nouveau’ dans la section Champs. Les champs ont également besoin de quelques réglages :

‘Nom’ Comme pour une table : la première lettre doit être majuscule et peut se composer au maximum de 20 caractères. Nous mettons le nom de ce champ à ‘Nom’ car il contiendra les noms des personnes que nous sommes sur le point d’ajouter.

‘Type’ Ici nous choisissons le type de champ désiré. Il en existe un certain nombre, mais pour ce champ nous choisirons d’utiliser le type Chaîne.

‘Longueur maximum’ Ici, vous pouvez définir le nombre maximum de caractères qu’un utilisateur pourra écrire dans la chaîne. Ici, nous avons choisi 30.

‘Valeur initiale’ Il est possible qu’un champ utilise une valeur initiale à chaque nouvel enregistrement que vous ajoutez, entrez ici cette valeur. Laissez cette ligne vide pour notre exemple.

‘Déclencheur’ Un champ peut également déclencher l’exécution d’une fonction utilisateur. Par exemple, lorsque vous entrez un nom, vous pouvez avoir une fonction utilisateur qui vérifie si ce nom existe déjà. Nous laissons ce champ vide.

4.5 Afficher le Projet

Après avoir appuyé sur le bouton ‘OK’ vous pourrez noter quelques changements dans la section d’affichage. Changez le choix du bouton en haut de la section d’affichage en optant pour ‘Fenêtre Principale’. Vous verrez ce que la fenêtre principale propose : actuellement seulement la table ‘Personne’. Si vous revenez de nouveau sur le choix du bouton en optant pour ‘fiche de table’, vous pourrez voir comment la table ‘Personne’ se présente. Actuellement elle affiche un panneau avec un champ.

Double-cliquer maintenant sur le ‘Panneau(Personne)’ au-dessus de la liste, dans la section d’affichage; une fenêtre doit s’ouvrir, vous permettant de définir la façon dont ce panneau doit être affiché :

‘Titre’ Le titre d’une table peut être différent de son nom. Notre table s’appelle ‘Personne’ mais ici nous pourrions aussi bien mettre ‘C’EST LA TABLE PERSONNE!’ si nous préférons.

‘Arrière plan’

L’arrière plan peut être modifié selon votre goût.

‘Gadgets’ Ici, nous pouvons définir les gadgets que nous désirons voir sur le panneau.

Appuyez sur le bouton ‘OK’ et double-cliquez sur ‘Nom’ dans la liste de la section d’affichage. La fenêtre ‘**affichage du champ**’ apparaît, elle contient les réglages sur la façon dont le champ ‘Nom’ va être montré.

‘Titre’ Les mêmes que pour le panneau. La chaîne que vous entrez ici est ce que vous voyez réellement lorsqu’on est en mode édition d’enregistrement.

‘Raccourci clavier’

Permet de placer une lettre qui peut être utilisée pour sauter vers ce champ, lorsqu’on est en mode édition d’enregistrement. Pour sauter vers un champ vous devez maintenir la touche **Alt** (Linux et Windows) ou la touche **Amiga** et appuyer sur la lettre.

‘Début’ Cochez la case début pour sauter vers ce champ toutes les fois qu’un nouvel enregistrement est ajouté. Dans notre cas nous voulons toujours ou la majeure partie du temps présenter d’abord le premier nom dans un nouvel enregistrement. Donc, cochez la case début.

‘Lecture seule’

A cocher si le champ est seulement lu. Ne pas s’en occuper.

‘Poids’

Décide quelle quantité de ce champ (en concurrence avec d’autres champs) devra être visible par rapport à l’espace disponible. Par exemple, si trois chaînes de 50 caractères se trouvent dans une fenêtre n’offrant qu’une place limitée de 100 caractères, ce nombre décidera combien d’espaces la chaîne obtiendra par rapport aux autres. Le laisser à 100.

‘Arrière-plan’

Les mêmes que pour le panneau.

‘Info bulle’

Ici, vous pouvez écrire n’importe quel texte qui vous paraît utile pour un utilisateur. Une bulle d’aide apparaît lorsque la souris est maintenue quelques secondes au-dessus d’un champ. Placez ceci ‘**si vous avez besoin d’aide, appelez l’auteur au 112**’.

Quittez l’éditeur de structure (choisissez ‘**quitter l’éditeur de structure**’ dans le menu ‘**Projet**’) et vous serez de nouveau en mode édition d’enregistrement où vous verrez à quoi ressemble vraiment la base de données. Vous verrez le titre qui correspond à la chaîne que vous avez choisie dans la section d’affichage du panneau. Le compteur d’enregistrement devrait indiquer ‘**#0/0**’ puisque nous n’avons pas encore ajouté d’enregistrement. A la suite, se trouve un bouton filtre et deux boutons flèches. Au-dessous du tout, vous devriez avoir le champ ‘Nom’ avec le texte que vous avez pu écrire dans la section d’affichage de ce champ. Si dans la section d’affichage, vous n’avez changé aucun texte, alors le panneau devrait s’appeler ‘**Personne**’, et le champ de type chaîne ‘Nom’. Déplacez la souris au-dessus de la chaîne champ ‘Nom’ et laissez la pendant quelques secondes. Si vous écrivez quelque chose dans la bulle d’aide, ce texte devrait paraître dans une fenêtre flottante.

4.6 Ajouter deux champs de type référence

Maintenant nous ajouterons deux champs référence. Les champs référence sont un peu différents des autres champs. Car leur nom pourrait impliquer qu'ils se réfèrent à des enregistrements. Ce sera plus compréhensible, lorsque dans un moment vous l'essayerez par vous-même.

Entrez dans l'éditeur de structure et ajoutez deux autres champs à la table '**Personne**'. Appuyez sur le bouton '**Nouveau**' dans la section champ, la fenêtre nouveau champ s'ouvre, nommée là '**Pere**', et changez le type en '**Référence**'. Un champ référence ne dispose que d'un réglage :

'Faire référence à'

Indique à quelle table le champ se rapporte. Il devrait déjà pointer vers '**Personne**'. Laissez-le comme ça, et appuyez sur le bouton '**Ok**'.

Ajoutez un autre champ en appuyant sur le bouton '**Nouveau**' dans la section champs, la fenêtre '**Nouveau champ**' s'ouvre. Nous nommons ce champ '**Mere**', positionnons son type de champ à une référence vers la table '**Personne**'.

Comme vous avez pu le noter, il y a maintenant trois champs dans la section Affichage. Cliquez sur '**Père**' puis sur les boutons haut et bas placés juste à gauche. Ce qui aura pour effet de changer l'affichage et le positionnement de '**Pere**' dans le mode édition d'enregistrement. Mettez '**Pere**' en haut, '**Nom**' au milieu, et '**Mere**' en bas.

Après, nous indiquerons quel contenu les champs référence '**Pere**' et '**Mere**' devraient afficher dans les enregistrements référencés. Double-cliquez sur '**Pere**' dans la section d'affichage puis cliquez sur '**Extras**'. Ici nous choisissons d'afficher le champ '**Nom**', appuyez sur le bouton '**Ok**' et répétons le procédé pour la '**Mere**'.

4.7 Ajouter des enregistrements

Maintenant nous devons ajouter quelques enregistrements. Quittez l'éditeur de structure et entrez dans le mode édition d'enregistrement. Pour ajouter un nouvel enregistrement, choisissez '**Nouvel enregistrement**' dans le menu '**Table**' (sur Linux et Windows vous trouverez ce menu en maintenant pressé le bouton droit de la souris à l'intérieur de la fiche de table). Le curseur devrait maintenant sauter automatiquement au champ que nous avons placé un peu plus tôt comme étant au '**Début**' dans la section d'affichage de l'éditeur de structure. Ajoutez deux enregistrements, un avec le nom de votre père dans '**Nom**' et un autre avec le nom de votre mère dans '**Nom**'. Ensuite, vous ajoutez un autre enregistrement avec votre propre nom dans '**Nom**'.

Maintenant il est temps de comprendre les champs référence. Cliquez sur le bouton de sélection dans '**Pere**' et vous obtiendrez la liste de tous les enregistrements auquel ce champ référence pourrait se rapporter. Choisissez le nom de votre père puis, suivre le même cheminement pour le bouton de sélection dans mère.

Maintenant, vous devriez avoir trois enregistrements, vous, votre père et votre mère. Dans votre enregistrement, le nom de votre père devrait être évidemment visible en haut dans '**Père**' et le nom de votre mère devrait être en bas dans '**Mère**'. Vous pouvez passer en revue les enregistrements en appuyant sur **Alt** ainsi que sur touches du clavier **Haut** et **Bas**.

Vous vous dites : Mais hé, mes parents aussi ont/avaient des parents ! Aussi, ajoutons encore quatre enregistrements, la troisième génération Ajoutez juste les enregistrements un par un en écrivant leurs noms dans 'Nom'. Si vous ne vous rappelez pas leurs noms écrivez juste 'le père de mon père', 'le père de ma mère' ou toute autre formule équivalente du même style. Vous pourrez alors naviguer dans tous les enregistrements et placer dans 'Pere' et 'Mere' les valeurs adéquates. Une fois cela fait, vous devriez avoir au moins sept enregistrements, votre enregistrement, l'enregistrement de vos parents et l'enregistrement de vos grands-parents.

4.8 Filtre

Puisque nous avons maintenant quelques enregistrements avec lesquels nous pouvons travailler, nous pourrions essayer la fonction filtre. Le filtre peut trier les enregistrements que vous ne voulez pas montrer, ils resteront dans la base de données mais ils ne seront simplement pas montrés.

Pour éditer le filtre choisissez 'Changer le filtre' dans le menu 'Table'. Une fenêtre comportant des opérateurs apparaîtra. Vous permettant de placer les conditions qu'un enregistrement doit remplir pour être affiché.

Dans ce petit exemple, nous utiliserons la commande LIKE, qui vous laisse faire la comparaison d'un champ avec un joker. Appuyez une fois sur le bouton LIKE vers la droite et double-cliquez sur 'Nom' dans la liste à gauche et le (LIKE Nom) devrait apparaître dans la chaîne juste au-dessus des boutons 'Ok' et 'Annuler'. Ensuite vous écrivez "*a*" de manière à ce que le texte entier devienne (LIKE Nom "*a*"). Ceci signifie que MUIbase devrait afficher tous les enregistrements qui contiennent la lettre 'a' n'importe où dans le 'Nom'.

Appuyez sur le bouton 'Ok' et vous pourrez noter que les enregistrements ne contenant pas de 'a' dans leur 'Nom', ne seront plus visibles. Puisque la lettre 'a' est très courante dans la plupart des langues et noms, Tous les enregistrements pourraient encore être affichés, mais vous pouvez essayer d'autres lettres pour faire fonctionner le filtre de façon plus visible.

Comme cité précédemment, le bouton 'F' sur le panneau indique si le filtre est ouvert ou fermé. Enfin, lorsque vous aurez fini vos essais, quittez le filtre de sorte que tous les enregistrements soient visibles.

4.9 Requêtes

Maintenant que nous avons un peu joué avec la fonction filtre, nous nous tournons vers le dispositif de requête de MUIbase. Des requêtes d'informations peuvent être utilisées et affichées dans une base de données selon certains critères.

Choisir 'Requêtes' dans le menu 'Programme', l'éditeur de requête s'ouvre. Maintenant, une fenêtre avec quelques boutons en haut et deux plus grands secteurs ci-dessous apparaissent. La chaîne en haut à gauche permet d'écrire le nom de la requête que vous voulez appeler.

'Exécuter'

Compile et lance la requête.

'Imprimer'

Imprime le résultat de la requête dans un fichier ou sur votre imprimante.

‘Charger et Enregistrer’

Vous permet de charger et enregistrer chacune des requêtes.

Le premier grand secteur permet d’écrire la requête. Le deuxième grand secteur permet d’afficher le résultat.

Maintenant produisons une liste de toutes les personnes que nous avons filtrées précédemment. Ecrire : **‘Personnes avec un A dans leur nom’** dans la chaîne en haut à gauche. C’est le titre pour cette requête. Dans le grand secteur supérieur, introduire au clavier :

```
SELECT Nom FROM Personne WHERE (LIKE Nom "*a*")
```

Maintenant, si vous lancez cette requête en appuyant sur le bouton **‘Exécuter’** il produira une liste de toutes les personnes avec la lettre **‘a’** dans leur nom. Essayez de changer la lettre pour voir différents résultats.

Il est temps de présenter la commande AND. Appuyez sur le bouton de sélection juste à gauche du bouton **‘Exécuter’** dans l’éditeur de requête. Choisissez alors **‘Nouveau’** et nommez-la **‘Personnes dont le nom contient les lettres a et s’**. Ecrivez alors

```
SELECT Nom FROM Personne WHERE
(AND (LIKE Nom "*a*") (LIKE Nom "*s*"))
```

Notez que nous employons toujours la commande LIKE pour le choix des enregistrements contenant les lettres **‘a’** ou **‘s’** dans leurs noms, mais la commande AND exige que LES DEUX critères LIKE soient vérifiés. Par conséquent, seul les enregistrements comportant LES DEUX lettres **‘a’** et **‘s’** dans leur nom sont affichés lorsque la requête est exécutée.

4.10 Ajouter une Table avec un Mémo et un champ de type bouton

Il s’agit de deux manières de choisir et de montrer la base de données. Une autre manière d’afficher les données peut être faite par un programme. Pour afficher les données, nous pouvons utiliser un type de champ appelé mémo.

Entrez dans l’éditeur de structure et appuyez sur le bouton **‘Nouveau’** dans la section table. Nommez la nouvelle table **‘Controle’** et placez son nombre d’enregistrements à **‘Exactement un’**. Pressez et maintenez le bouton de la souris sur la nouvelle table, puis déplacez le curseur juste un peu au-dessus du milieu de la table **‘Personne’** et enfin relâchez le bouton de la souris. Dans la section table, **‘Controle’** devrait maintenant être en haut, suivi par la **‘Personne’** ci-dessous.

Assurez-vous que la table **‘Controle’** est sélectionnée, et appuyez sur le bouton **‘Nouveau’** dans la section champ. Placez le type du nouveau champ sur **‘Mémo’** et donnez-lui le nom **‘Resultat’**. Appuyez sur le bouton **‘Ok’** puis ajoutez un autre champ à la table **‘Controle’** en appuyant de nouveau sur le bouton **‘Nouveau’** dans la section champ. Cette fois, choisissez le type **‘Bouton’** et appelé le **‘Ascendance’**.

Pour améliorer l’aspect de la base de données, dans la section affichage, cliquez une fois sur **‘Ascendance’** ainsi que sur le bouton **‘haut’**.

4.11 MUIbase, programmation pour faire une ascendance

Maintenant nous avons un bouton qui peut exécuter un programme et un mémo pour y afficher les données. Il est donc temps d'entrer dans l'éditeur de programme. Pour cela, il faut choisir 'Editer' à partir du menu 'Programme'. L'éditeur de programme à trois boutons :

'Compiler & Fermer'

Il fait juste cela. Il compile le programme et sort de l'éditeur de programme.

'Compiler'

Compile les programmes mais reste dans l'éditeur de programme.

'Rétablir'

Rétablit tous les changements pour revenir à l'état initial lors de votre entrée dans l'éditeur de programme.

Toutes les fonctions de programme que vous écrirez résideront dans cette fenêtre, nous devons les séparer entre elles. Dans MUIbase c'est la commande DEFUN qui le fait. Tout ce qui est entre les deux parenthèses fera partie, dans cet exemple, de la fonction `ascendance` :

```
(DEFUN ascendance ()
```

```
  ; C'est la parenthèse de la fin de DEFUN
)
```

À cet effet nous écrivons maintenant la première fonction qui produira, dans la base de données, un arbre généalogique de la personne courante et plaçons le résultat dans le champ 'Résultat'. Cette fonction `ascendance` comporte en fait trois fonctions :

- `ascendance` qui remplit 'Controle.Resultat' avec le résultat de l'appel à une autre fonction.
- `getascendance` qui rassemble l'ascendance dans une liste.
- `ascendance2memo` qui convertit cette liste en mémo.

```
  ; Le programme ascendance
```

```
(DEFUN ascendance ()
```

```
  (SETQ Controle.Resultat (ascendance2memo (ascendance Personne NIL) 0 3))
)
```

```
  ; Le programme getascendance
```

```
(DEFUN getascendance (personne:Personne niveau:INT)
```

```
  (IF (AND personne (OR (NULL niveau) (> niveau 0)))
```

```
    (LIST personne.Nom
```

```
      (getascendance personne.Pere (1- niveau))
```

```
      (getascendance personne.Mere (1- niveau))
```

```
    )
```

```
  )
```

```

)

; Le programme ascendance2memo

(DEFUN ascendance2memo (ascendance:LIST indent:INT niveau:INT)
  (IF (> niveau 0)
    (+
      (ascendance2memo (NTH 1 ascendance) (+ indent 8) (1- niveau))
      (IF pedigree (SPRINTF "%*s%s\n" indent "" (FIRST ascendance)) "\n")
      (ascendance2memo (NTH 2 ascendance) (+ indent 8) (1- niveau))
    )
    ""
  )
)

```

En écrivant ce programme, assurez-vous que toutes les parenthèses se trouvent où elles devraient être. Un trop grand nombre ou trop peu de parenthèses est l'erreur la plus commune lorsque MUIbase compile votre programme. Le message d'erreur de MUIbase serait alors probablement 'Erreur de syntaxe'. Appuyez sur le bouton 'Compiler & fermer' et la fenêtre se ferme, ce qui signifie qu'il n'y a eu aucune erreur lors de la compilation.

Ne vous inquiétez pas trop si au début, vous ne comprenez pas toutes les fonctions. Il y a un chapitre complet (voir voir [Chapitre 15 \[Programming MUIbase\], page 75](#)) contenant la liste de toutes les fonctions avec leur description détaillée.

Maintenant nous avons un programme à exécuter, mais d'abord nous devons assigner la fonction programme au bouton 'Ascendance'. Pour cela, il faut entrer dans l'éditeur de structure, choisir 'Controle' dans la section table et double-cliquer sur le champ 'Ascendance' dans la section champs. Cliquez alors le bouton de sélection 'Déclencheur'. Dans cette liste, toutes les fonctions de votre programme seront énumérées, à ce moment, il devrait y avoir trois fonctions : `ascendance`, `getascendance` et `ascendance2memo`. Double-cliquez sur `ascendance` comme la fonction du programme que le bouton 'Ascendance' déclenchera, puis appuyez sur le bouton 'Ok' et quittez l'éditeur de structure.

Maintenant si tout est fait correctement, l'appuie sur le bouton 'Ascendance' produira l'ascendance de la personne courante. Essayez de changer la personne pour voir différentes ascendances.

4.12 Programmation dans MUIbase pour énumérer les enfants d'une personne

Le prochain ajout sur cette base de données exige encore plus d'enregistrements, vous devriez ajouter vos frères et soeurs. Si vous n'en avez pas, écrivez 'Ma soeur fictive 1', 'Mon frère fictif 1' qui naturellement devront être placés de façon à avoir les même parents que vous.

Alors allez, dans l'éditeur de programme et écrivez le suivant (pour créer un autre programme) :

```

; Le programme enfants compte le nombre d'enfants pour une personne.
; D'abord nous définissons les variables, par exemple noms pour les noms des enfants

```

```

(DEFUN enfants ()
  (LET ( (noms "") (count 0) (parent Personne) )

    ; Pour tous les enregistrements dans la table Personne, faire ce qui suit :
    ; Si la variable parent apparaît comme père ou mère
    ; dans un ou plusieurs enregistrements alors :
    ;   ajoutez le nom à la variable des noms
    ;   incrémentez le compte de 1.

    (FOR ALL Personne DO
      (IF (OR (= parent Mere) (= parent Mere))
        (
          (SETQ noms (+ noms Nom "\n"))
          (SETQ nombre (1+ nombre))
        )
      )
    )

    ; Ensuite nous écrivons le résultat dans la Controle.Resultat.
    ; Si la personne courante n'a aucun enfant, écrire une chaîne.
    ; Si elle a des enfants alors écrire une autre chaîne.

    (SETQ Controle.Resultat
      (+ Personne.Nom (IF (> nombre 0)
        (+ " est fier d'être le parent de " (STR nombre) " enfant(s).")
        " n'a aucun enfants (pour le moment :-).")
      ))
    )

    ; Si les parents ont des enfants alors, ajoutez leurs noms.

    (IF (<> nombre 0)
      (SETQ Controle.Resultat
        (+ Controle.Resultat "\n\n"
          (IF (= nombre 1)
            "Le nom de l'enfants est :"
            "Les noms des enfants sont :")
          )
        "\n\n"
        noms
        )
      )
    )

    ; C'est la parenthèse de fin de la commande LET.
  )

```

```
; C'est la parenthèse de fin de DEFUN enfants
)
```

Pour créer des variables, nous utilisons la commande **LET**. Les variables créées avec la commande **LET** sont locales et seulement visibles entre les parenthèses ouvrante et fermante de cette commande '**LET**'. Ainsi, toutes les commandes voulant accéder à ces variables doivent être placées à l'intérieur des parenthèses.

Tout ce dont nous avons besoin pour exécuter notre programme que c'est un nouveau bouton programme. Par conséquent, entrez dans l'éditeur de structure et ajoutez un champ de type bouton dans la table '**Controle**', appelez-la '**Enfants**' et choisissez **enfants** comme étant fonction de programme à déclencher.

Pour apporter un certain ordre dans la fiche de la table '**Controle**' il est maintenant temps de présenter les groupes. Tous les objets peuvent être ordonnés dans des groupes alignés verticalement ou horizontalement.

Dans la section d'affichage, shift-cliquez sur '**Ascendant**' et sur '**Enfants**', ensuite cliquez sur le bouton '**Groupe**' sur la gauche. Maintenant les deux boutons de programme seront rassemblés et alignés verticalement dans un groupe. Cependant, comme nous voulons qu'il soit aligné horizontalement, double-cliquez sur le '**VGroup**' qui est apparu dans la section d'affichage. Une fenêtre s'ouvrira qui vous laissera modifier les réglages de ce groupe. Saisissez '**Programmes**' comme titre et vérifiez le bouton '**Horizontal**'.

À ce moment nous pouvons cacher le nom du champ '**Resultat**' de la table '**Controle**'. Double-cliquez sur le '**Resultat**' dans la section d'affichage et videz le champ titre. Ceci affichera le champ '**Resultat**' sans aucun titre.

Pour rendre les choses plus faciles, si nous ajoutons plus de programmes ou de champs dans la table '**Controle**', nous devrions placer '**Resultat**' et le groupe '**Programmes**' dans un groupe vertical. Soyez certain que vous avez seulement sélectionné '**Programmes**' et '**Resultat**' appuyez sur '**Groupe**'. Cela place '**Programmes**' et '**Resultat**' dans un groupe vertical.

Quittez l'éditeur de structure et jetez un coup d'oeil au résultat. Appuyez sur le bouton '**Les enfants de la personne**' pour voir le nombre et le nom des enfants de la personne courante.

Cet exemple peut très bien être étendu dans un programme généalogique complet. Les seules vraies limites sont votre imagination et la taille de votre disque dur.

5 Concepts de base

Avant de créer votre propre base de données et de commencer à y entrer des données, vous devez vous familiariser avec certaines notions de base de MUIbase.

5.1 Projets

Un projet MUIbase se compose de tous les éléments dont vous avez besoin pour gérer vos données. Cela va de l'interface graphique aux programmes écrits pour le projet en passant par la saisie des données.

Un projet peut être chargé, sauvegardé ou effacé du disque. Toute modification d'un projet n'est effectuée qu'en mémoire. Vous pouvez revenir à tout moment à l'état de la dernière sauvegarde en rechargeant le projet.

MUIbase est capable de gérer de multiples projets en parallèle. Ainsi il n'est pas nécessaire de démarrer MUIbase une deuxième fois pour charger un autre projet.

5.2 Tables

MUIbase organise les données en tables. Une table est constituée de lignes et de colonnes : une ligne correspond à un enregistrement alors qu'une colonne correspond à un champ.

Voici un exemple de structure pour un carnet d'adresses :

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

Il existe un genre de table particulier qui ne peut contenir qu'un seul enregistrement. Ce genre de table est très utile pour contrôler un projet de base de données. Par exemple, vous pouvez y mettre des boutons pour exécuter des tâches diverses ou un champ en lecture seule pour afficher une information concernant le projet.

Voici un exemple : supposez que vous ayez une base de données pour gérer votre compte bancaire dans laquelle vous entrez vos revenus et vos dépenses. Une table à un seul enregistrement pourrait contenir un champ en lecture seule de type Réel qui afficherait la différence.

Chaque table dispose de deux pointeurs d'enregistrement : un vers l'enregistrement actuellement affiché dans l'interface graphique (nommé *pointeur d'enregistrement GUI*) et un vers l'enregistrement courant lors de l'exécution d'un programme MUIbase (nommé *pointeur d'enregistrement de programme*).

Vous pouvez définir autant de tables que vous le désirez dans un projet MUIbase. Les tables peuvent être ajoutées, renommées ou effacées d'un projet.

5.3 Enregistrements

Un enregistrement correspond à une ligne d'une table. Il contient toutes les informations d'un élément de la table, par exemple dans une table gérant des adresses, un enregistrement correspond à une adresse.

Chaque enregistrement se voit assigner un numéro qui définit son emplacement dans la table. Ce numéro peut changer si vous ajoutez ou effacez des enregistrements.

Il existe pour chaque table un enregistrement appelé *initial record* avec des valeurs par défaut qui permet l'entrée de nouveaux enregistrements. L'enregistrement initial porte toujours le numéro d'enregistrement 0.

Les enregistrements d'une table peuvent être ajoutés, modifiés ou effacés. Ils ne sont pas forcément présent en mémoire mais sont chargés et stockés quand c'est nécessaire. Le nombre maximum d'enregistrements dans une table est limité par deux choses: Le numéro de l'enregistrement, est une valeur 32bits de type entier, ce qui donne en théorie un maximum de 4294967295 enregistrements. La deuxième limitation (et la plus contraignante) concerne la mémoire parce que chaque enregistrement génère un petit fichier qui est gardé en mémoire. Malgré tout, MUIbase est capable de travailler sur 10,000 enregistrements et plus.

5.4 Champs

Un champ correspond à une colonne d'une table. Il définit le type et l'apparence de la colonne correspondante.

Les champs d'une table peuvent être ajoutés, renommés ou effacés. Le nombre de champ par table n'est pas limité.

Pour chaque champ vous devez définir un type qui délimite le contenu de ce champ. Reportez-vous au paragraphe suivant pour avoir la liste des types de champs disponible.

5.5 Types de champ

Les champs peuvent être de type string, integer, real, bool, choice, date, time, memo, reference, virtual, ou button. Ces types sont décrits plus en détail ci-dessous.

Certains types de champ peuvent avoir une valeur spéciale appelée NIL. Cette valeur spéciale se traduit par une valeur non définie, par exemple pour un type date, elle est interprétée comme une date inconnue. La valeur NIL est similaire à la valeur NULL que l'on trouve dans d'autres systèmes de base de données.

Notez qu'une fois le type de champ défini, il ne peut plus être changé.

5.5.1 Champs texte

Les champs de type texte (en anglais String) contiennent une seule ligne de texte. Les champs textes sont les types de champ les plus souvent rencontrés dans un projet de base de données. Par exemple, une base de données relationnelle pourrait contenir le nom, la rue et la ville d'une personne chacun dans son propre champ texte.

Pour un champ texte, vous devez définir le nombre maximum de caractères autorisés. Ce nombre n'affecte pas directement la consommation en mémoire ou la place sur le disque parce que seul le contenu effectif du champ est stocké (d'autres systèmes de base de données appellent cette fonctionnalité texte compressé). En cas de besoin, ce nombre peut être modifié après avoir défini un champ texte.

Les champs texte peuvent également servir à stocker des polices de caractères et des noms de fichiers. Concernant les noms de fichiers, des afficheurs externes peuvent être appelés pour afficher leur contenu. De plus, une classe image interne permet de visualiser l'image d'un fichier.

Les champs texte ne peuvent pas prendre la valeur spéciale NIL.

5.5.2 Champs entiers

Les champs de type Entier (en anglais Integer) stockent des nombres entiers compris entre -2147483648 et 2147483647. Ils sont principalement utilisés pour stocker des quantités de toute nature, par exemple le nombre d'enfants d'une personne, le nombre de titres présents sur un CD.

Les champs entiers peuvent prendre la valeur spéciale NIL, ce qui équivaut à un nombre entier non défini.

5.5.3 Champs réels

Les champs de type Réel (en anglais Real) stockent des nombres à virgules compris entre -3.59e308 et +3.59e308. Ils sont utilisés pour stocker des nombres de toute nature, par exemple les sommes d'argent perçues ou dépensées.

Pour chaque champ réel, vous pouvez définir le nombre de chiffres après la virgule pour l'affichage, bien que la valeur stockée soit d'une précision absolue.

Les champs réels peuvent prendre la valeur spéciale NIL, ce qui équivaut à un nombre réel non défini.

5.5.4 Champs booléens

Les champs de type Booléen (en anglais Bool) stockent une seule information. Ils sont utilisés pour stocker des valeurs oui/non ou vrai/faux, par exemple dans un projet de gestion de commandes, un champ Booléen peut stocker l'information 'paiement effectué ?'.

Les champs Booléens ne peuvent avoir que TRUE (mot anglais pour VRAI) ou NIL comme valeur. Dans ce cas, NIL équivaut à la valeur FALSE (mot anglais pour FAUX).

5.5.5 Champs choix

Les champs de type Choix (en anglais Choice) enregistrent une donnée parmi une liste de données. Par exemple, pour un projet de carnet d'adresses un champ de type Choix peut être utilisé pour enregistrer le nom du pays, qui se trouve dans une liste de pays du genre 'USA', 'Canada', 'Allemagne' ou 'autres'.

Les champs de type Choix n'enregistrent pas le texte lui-même mais le numéro correspondant au texte dans la liste de données (index). Le nombre de données ainsi que les données peuvent être modifiées une fois que le champ a été créé. Cependant si vous modifiez un champ de type Choix, les valeurs des enregistrements existants ne seront pas modifiées pour refléter ce changement.

Les champs de type Choix ne peuvent pas prendre la valeur NIL.

5.5.6 Champs date

Les champs de type Date enregistrent des dates du calendrier. Par exemple, un champ de type Date pourrait stocker les dates d'anniversaire.

Pour visualiser ou enregistrer des champs de type Date, il faut que le format soit du style 'JJ.MM.AAAA', 'JJ/MM/AAAA' ou 'AAAA-MM-JJ', dans lequel 'JJ', 'MM' et 'AAAA' sont des valeurs à 2 et 4 chiffres qui correspondent respectivement au jour, au mois et à l'année.

Les dates peuvent prendre la valeur NIL, ce qui correspond à une date non définie.

5.5.7 Champs heure

Les champs de type heure (en anglais Time) enregistrent l'heure ou la durée. Par exemple, un champ de type heure pourrait enregistrer la durée des pistes d'un CD.

Pour visualiser ou enregistrer des champs de type heure, il faut que le format soit du style 'HH:MM:SS', 'MM:SS' ou 'HH:MM', dans lequel 'HH' représente les heures, 'MM' les minutes, et 'SS' les secondes. En interne, les heures sont stockées en nombre de secondes à partir de 0:00. Les heures peuvent dépasser 23:59:59 jusqu'au temps maximum de 596523:14:07, en revanche les valeurs négatives ne sont pas possibles.

Les heures peuvent prendre la valeur NIL, ce qui signifie une durée non définie.

5.5.8 Champs mémo

Les champs de type mémo (en anglais Memo) enregistrent du texte de longueur variable et sur plusieurs lignes. La longueur du texte s'effectue de manière dynamique ce qui veut dire que la mémoire est allouée d'après la longueur actuelle du texte. Par exemple dans un projet de gestion de films, un mémo pourrait enregistrer les résumés des films.

Les mémos ne peuvent pas prendre la valeur NIL.

5.5.9 Champs Référence

Les champs de type référence (en anglais Reference) sont particuliers et n'existent probablement pas dans d'autres systèmes de base de données. Les références stockent un pointeur vers un autre enregistrement. Cet autre enregistrement peut se trouver dans la même table que celle contenant la référence mais aussi dans une toute autre table.

Par exemple dans un projet de gestion généalogique, deux champs de type Référence pourraient stocker les pointeurs vers l'enregistrement respectivement du père et de la mère. Autre exemple dans un projet de gestion de CD et de chansons, une table contenant les chansons pourrait avoir une référence qui pointerait vers les enregistrements des CD correspondants.

Pour l'affichage d'une référence, n'importe quel champ de l'enregistrement référencé peut être utilisé. La valeur d'une référence peut être saisie en sélectionnant un enregistrement parmi la liste des enregistrements de la table référencée.

Les références peuvent prendre la valeur NIL, ce qui correspond à un pointeur vers l'enregistrement initial de la table "externe".

5.5.10 Champs virtuels

Les champs virtuels (en anglais Virtual) n'enregistrent pas d'information dans la base de donnée elle-même mais effectuent des calculs dynamiques quand c'est nécessaire.

Par exemple, dans un projet de gestion de factures où un champ réel contiendrait les sommes hors taxes, un champ virtuel pourrait calculer les sommes toutes taxes comprises. Le champ virtuel serait calculé à partir des valeurs hors taxes à chaque fois que l'on voudrait connaître sa valeur.

Il existe trois façons de visualiser des champs virtuels : booléen, texte et liste. Cela permet d'afficher la valeur d'un champ virtuel sous forme de valeur VRAI/FAUX, sous forme de texte sur une seule ligne en incluant nombres, dates et heures ou sous forme d'une liste de textes, par exemple pour afficher tous les titres d'un CD.

Les champs virtuels peuvent prendre la valeur NIL, ce qui correspond à FAUX dans le mode booléen, non défini dans le mode texte et vide dans le mode liste de textes.

5.5.11 Boutons

Les champs de type Bouton (en anglais Button) ne constituent pas vraiment un type de champ parce qu'ils ne stockent pas et n'affichent pas d'information. Les boutons sont plutôt utilisés pour lancer des programmes MUIbase.

5.6 Tableau des types de champ

La table suivante récapitule tous les types champ disponibles :

Type	Description	peut prendre NIL ?
String (Texte)	Pour stocker du texte de 1..999 caractères. Le texte peut servir également à stocker des noms de fichiers, des noms de poli	Non
Integer (Entier)	Pour stocker des nombres entiers.	Oui
Real (Réel)	Pour stocker des nombres réels.	Oui
Bool (Booléen)	Valeur booléenne VRAI ou NIL.	Oui (NIL = Faux)
Choice (Choix)	Un nombre parmi n nombres. Les nombres sont représentés par des étiquettes de texte.	Non
Date	Pour stocker des dates (1.1.0000- 31.12.9999).	Oui
Time (Heure)	Pour stocker des horaires (00:00:00-596523:14:07).	Oui
Memo (Mémo)	Texte sur plusieurs lignes de longueur illimitée.	Non
Reference	Pour stocker un pointeur vers un enregistrement dans une autre table.	Oui (NI

(Référence)

Virtual Pour afficher des résultats à partir d'un programme MUIbase. Oui
(Virtuel)

Button Pour lancer un programme de fonction. Non (non disponible)
(Bouton)

5.7 Consommation mémoire

Chaque type de champ nécessite une certaine quantité de mémoire pour stocker la valeur dans un enregistrement. À l'exception des types Virtuel et Bouton, tous les autres types ont besoin d'un en-tête de 2 octets pour le stockage interne de l'information. De plus, de l'espace spécifique à chaque type est nécessaire pour stocker la valeur actuelle. La table suivante montre les besoins en mémoire et en espace de stockage sur disque avec l'en-tête de 2 octets pour chaque type.

Type	besoin en Mémoire	besoin en espace disque
Texte	$2 + 4 + \text{longueur du texte} + 1$	$2 + \text{longueur du texte} + 1$
Entier	$2 + 4$	$2 + 4$
Réel	$2 + 8$	$2 + 8$
Booléen	$2 + 0$	$2 + 0$
Choix	$2 + 2$	$2 + 2$
Date	$2 + 4$	$2 + 4$
Heure	$2 + 4$	$2 + 4$
Mémo	$2 + 4 + \text{longueur du texte multi lignes} + 1$	$2 + \text{longueur du texte multi lignes}$
Référence	$2 + 4$	$2 + 4$
Virtuel	0	0
Bouton	0	0

Ici on entend par *longueur du texte* la longueur du texte à stocker et par *longueur du texte multi lignes* la taille du texte multi lignes à stocker.

5.8 Associations

Maintenant vous êtes en mesure de mettre vos informations sous forme de table avec enregistrements et champs. Mais peut être voudriez-vous associer différentes tables.

Par exemple, si vous répertoriez votre collection de CD, vous pourriez avoir besoin de deux tables, une pour les CD eux-mêmes et une autre pour les chansons se trouvant sur ces CD. Bien sûr vous pourriez également avoir toutes les chansons dans la table du CD mais dans ce cas, vous auriez un nombre fixe de titres par CD.

Maintenant que vous avez ces deux tables, il est nécessaire de relier chaque chanson au CD dont elle fait partie. Cette étape se nomme *association* entre les deux tables. Dans MUIbase, on utilise un champ de type Référence pour mettre en place une association.

Lors de la mise en place d'un champ de type Référence dans une table, vous créez automatiquement une association entre la table contenant ce champ et la table à laquelle il se réfère.

5.8.1 Associations Un à Un

Les associations un à un sont les plus simples où pour chaque enregistrement on a zéro ou un partenaire dans la même table ou dans une autre table.

Par exemple dans une base de données gérant vos acteurs favoris, vous pourriez utiliser un champ de type Référence nommé 'marié avec' qui montrerait la personne avec qui cet acteur est marié. Un acteur qui serait célibataire aurait une valeur NIL pour ce champ.

Evidemment, personne ne vous empêche d'associer le champ 'marié avec' de plusieurs acteurs avec la même personne. Quoi qu'il en soit en programmant MUIbase, il est possible de déceler ce genre de cas et d'agir en conséquence.

5.8.2 Associations Un à plusieurs

Les associations un à plusieurs sont utiles pour associer un ensemble d'enregistrements avec un enregistrement de la même table ou dans une autre table.

Par exemple, pour un projet de gestion de compte bancaire vous pourriez définir une table pour tous vos comptes bancaires et une autre table répertoriant toutes vos transactions. Maintenant vous voudriez sûrement savoir à quel compte appartient cette transaction donc vous créez, dans la table des transactions, un champ de type Référence associé à la table des comptes.

Les associations un à plusieurs sont les associations les plus couramment utilisées. Vous pouvez les utiliser pour gérer toute structure de type hiérarchisé, par exemple des CD avec les chansons ou des comptes bancaires avec les transactions ou encore des arbres généalogiques, etc.

Les associations un à plusieurs servent également de base à la réalisation d'associations plusieurs avec plusieurs comme vous le verrez dans le chapitre suivant.

5.8.3 Associations Plusieurs à plusieurs

Les associations plusieurs à plusieurs sont utilisées dans le cas où vous voudriez relier un ensemble d'enregistrements à un autre ensemble d'enregistrements.

Par exemple, dans un projet de gestion d'acteurs et de films, vous auriez deux tables, une pour les acteurs et une autre pour les films. Supposez que vous vouliez connaître les acteurs d'un film. Dans ce cas vous pourriez créer, dans la table des acteurs, un champ Référence vers la table des films. Mais de cette façon, vous ne pourriez avoir qu'un seul film référencé pour chaque acteur parce qu'il n'existe qu'un seul champ Référence dans la table des acteurs. Ce dont vous avez besoin c'est d'un nombre illimité de références depuis la table des acteurs vers la table des films.

Cette opération s'effectue en ajoutant une nouvelle table qui ne contient que deux champs référence, l'un pointant vers la table des acteurs tandis que l'autre pointera vers la table des films. À partir de maintenant vous pouvez établir des associations en ajoutant des nouveaux

enregistrements à cette table. Pour chaque association acteur-film, vous ajoutez un nouvel enregistrement et sélectionnez l'acteur et le film dans les champs Référence correspondants.

Pour savoir dans quels films a joué tel acteur, vous n'avez plus qu'à chercher dans la nouvelle table tous les enregistrements relatifs à cet acteur, ce qui montrera tous les films qui lui sont associés. MUIbase pourrait automatiquement effectuer ce genre de recherche et afficher le résultat dans une liste.

Les tables suivantes montrent comment relier un groupe d'acteurs avec un groupe de films.

	Titre	Pays

m1:	Batman	USA
m2:	Batman Returns	USA
m3:	Speechless	USA
m4:	Tequila Sunrise	USA
m5:	Mad Max	Australie
m6:	Braveheart	USA

	Nom

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

FilmRef	ActeurRef

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

À partir de ces tables, vous pouvez voir que Mel Gibson a joué dans les films Tequila Sunrise, Mad Max et Braveheart ou que dans le film Batman, les acteurs principaux sont Michael Keaton, Jack Nicholson, et Kim Basinger.

5.9 Interface graphique

MUIbase utilise une interface utilisateur (GUI ou graphical user interface) organisée de façon hiérarchique pour afficher le contenu des enregistrements et stocker des données. Chaque projet utilise sa propre fenêtre principale, qui contiendra les éléments graphiques ainsi que les fenêtres secondaires. Ces éléments graphiques sont aussi appelés *objets d’affichage*.

Une table est affichée dans son propre environnement visuel appelé *fiche*. Une fiche ne peut afficher qu’un seul enregistrement à la fois. Sa représentation ainsi que les champs qui composent la fiche sont modifiables par l’utilisateur.

Les chapitres suivants détaillent les éléments graphiques dont dispose MUIbase pour créer l’environnement visuel d’un projet.

5.9.1 Fenêtres

Les fenêtres sont utilisées pour diffuser les informations d’un projet vers différentes zones indépendantes.

Chaque projet a automatiquement sa propre fenêtre principale. En cas de besoin, par exemple si la taille de la fenêtre principale est trop grande, on peut créer des fenêtres supplémentaires, qui peuvent elles-mêmes avoir des fenêtres secondaires.

Pour chaque fenêtre secondaire, on peut placer un bouton dans la fenêtre du niveau supérieur afin de faciliter l’ouverture de la fenêtre secondaire. Ce bouton ressemble à un bouton de texte mais peut contenir un petit icône pour le distinguer des autres boutons.

La fenêtre principale ne peut avoir de fenêtre au niveau supérieur et est donc dépourvue de bouton fenêtre. Fermer une fenêtre principale équivaut à fermer le projet en cours.

Une fenêtre peut avoir d’autres éléments graphiques pour enfant. Une image par défaut est affichée si aucun enfant n’a été ajouté à la fenêtre.

5.9.2 Fiches

Une fiche permet d’afficher le contenu d’une table. On ne peut voir qu’un enregistrement de la table à la fois.

La fiche peut incorporer un panel pour contrôler la table (voir le paragraphe suivant). Les autres éléments graphiques tels que les objets de champ ou de texte peuvent être placés dans une fiche pour montrer le contenu de l'enregistrement.

Les fiches ne peuvent pas être placées à l'intérieur d'autres fiches car cela générerait une hiérarchie de fiches et par prolongement de tables. Pour mettre en place une hiérarchie de tables, on utilise une association de type 1:n entre deux tables.

5.9.3 Panels

Un panel est une zone rectangulaire de petite taille mais large, et qui est positionnée au bord supérieur d'une fiche. Un panel peut afficher un titre, par exemple le nom correspondant à la table, une paire de chiffres montrant le numéro de l'enregistrement actuel et le nombre total d'enregistrements, et plusieurs boutons pour contrôler la table, par exemple pour afficher l'enregistrement précédent ou suivant.

On ne peut définir qu'un seul panel par fiche. Si vous mettez un panel dans une fiche, cela se traduit par une bordure supplémentaire, sinon aucune bordure n'est affichée.

5.9.4 Objets de champ

Les objets de champ sont utilisés pour afficher le contenu d'un des champs d'un enregistrement.

Suivant le type de champ, l'élément graphique sera soit une zone de texte (pour les types texte, entier, réel, date et heure), une case à cocher (pour le type booléen), un bouton cyclique ou une série de boutons radio (pour le type choix), une zone d'édition (pour le type mémo), une liste (pour le type référence), un texte, une case à cocher ou une liste (pour le type virtuel) et enfin un bouton texte ou bouton image (pour le type bouton). Dans certains cas, l'élément graphique peut être une zone de texte si l'objet de champ est en lecture seule.

5.9.5 Objets texte

Les objets texte sont utilisés pour introduire ou décrire les autres éléments d'une fiche ou pour afficher un texte statique.

5.9.6 Images

Les images peuvent être affichées n'importe où dans une fenêtre. Une image peut être un motif, une zone colorée ou bien une image d'un fichier externe. La taille de l'image peut être fixe ou redimensionnable.

Une image est statique. Pour stocker des images dans une table, vous devez utiliser un champ de type string (voir [Section 5.5.1 \[String type\], page 20](#)).

5.9.7 Objets d'espacement

Les objets d'espacement sont utilisés pour insérer des espaces dans la présentation d'une fenêtre ou dans la fiche d'une table. Un objet d'espacement peut avoir une barre horizontale (ou verticale) pour délimiter d'autres éléments graphiques.

5.9.8 Groupes

Les éléments graphiques peuvent être classés en groupes horizontaux ou verticaux. Le placement des éléments se fait de gauche à droite (groupe horizontal) ou du haut vers le bas (groupe vertical).

Un groupe peut entourer ses éléments d'un cadre rectangulaire, peut afficher un titre en haut du groupe et peut insérer des espaces entre ses éléments.

5.9.9 Objets balance

Les objets balance peuvent être placés entre d'autres objets d'une fenêtre, d'une fiche ou d'un groupe. Un objet balance permet à l'utilisateur de contrôler les dimensions de ses éléments fils et par conséquent la place prise par chacun.

5.9.10 Registres

Un registre peut être utilisé pour disposer les éléments graphiques sur plusieurs pages mais avec une seule de ces pages visible à la fois. Cela est très utile lorsque l'interface utilisateur devient trop importante et que vous ne voulez pas l'étendre sur plusieurs fenêtres.

6 Gestion des projets

Dans ce chapitre, vous trouverez comment MUIbase organise des projets, comment les ouvrir, les sauver, les supprimer et les fermer, comment vérifier l'intégrité des données et comment télécharger les enregistrements.

6.1 Format de fichier

Un projet de MUIbase se compose de plusieurs fichiers stockés dans un tiroir contenant le projet. Ce tiroir est d'abord créé quand vous sauvez le projet. Ne faites pas de supposition au sujet de la structure du tiroir ou des noms de fichier à l'intérieur du tiroir. En particulier, n'enlevez pas et ne placez pas de fichiers ni de tiroirs supplémentaires dans ce tiroir ! Ils seraient supprimés lors d'une réorganisation ultérieure du projet.

Le tiroir contient un fichier appelé '**Structure.mb**' où sont stockées les descriptions de toutes les tables, champs, filtres, etc. Les en-têtes d'enregistrements sont également placés ici. Pour chaque table vous trouverez un fichier avec le nom de la table. C'est à l'intérieur que sont stockés tous les enregistrements d'une table. Il y a enfin et surtout un fichier appelé '**.lock**'. Ce fichier est utilisé pour verrouiller un projet, c'est-à-dire, que MUIbase commence par poser un verrou exclusif sur ce fichier puis ouvre les autres fichiers. Si le verrou échoue, MUIbase sait qu'il y a déjà une instance de MUIbase travaillant sur ce projet. Une seule instance de MUIbase est autorisée à travailler sur un projet donné à la fois, parce que les fichiers d'enregistrements sont ouverts en mode lecture/écriture et que nous ne voulons certainement pas avoir mélange dans les données des opérations d'écriture de deux ou plus instances de MUIbase.

6.2 Informations

MUIbase conserve quelques informations sur chaque projet. Sélectionnez le menu '**Projet - Information**' pour obtenir des informations sur le projet en cours. L'information obtenue comprend le nom du projet, le nombre de tables, le nombre d'enregistrements dans toutes les tables, et une valeur montrant le nombre d'octets que vous pourriez gagner si vous réorganisiez ce projet. Le gain n'est cependant qu'une estimation grossière et ne doit pas être considéré comme une valeur exacte. Cette valeur est loin d'être précise, particulièrement si vous avez fait beaucoup de changements sur la structure du projet (en ajoutant ou en enlevant des champs).

6.3 Nouveau projet

MUIbase peut manipuler plusieurs projets en même temps. Vous êtes seulement limités par la mémoire disponible. Pour démarrer un autre projet, choisissez le menu '**Projet - Nouveau**'. Une nouvelle fenêtre s'ouvre avec un projet sans nom vide. Vous pouvez maintenant définir la structure de ce projet (voir l'voir [Chapitre 14 \[Structure editor\], page 60](#)) ou charger un projet existant à partir du disque dur (voir voir [Section 6.5 \[Open project\], page 31](#)).

6.4 Nettoyer le projet

Pour remettre à zéro un projet sélectionnez le menu '**Projet - Nettoyer - Projet**'. Le projet en cours sera alors clôturé et remplacé par un projet vide. Lorsque vous démarrez MUIbase sans projets vous arrivez automatiquement dans cet état.

Par la sélection du menu ‘**Projet - Nettoyer - Enregistrements**’ vous démarrez un nouveau projet en utilisant la structure courante. Ce qui signifie que tout (excepté les données enregistrées du projet en cours) est employé pour le nouveau projet.

Si un projet en cours n’a pas été sauvé sur le disque dur et que vous avez sélectionné le menu ci-dessus, une fenêtre de sûreté vous demandera alors la confirmation de l’opération.

6.5 Ouvrir le projet

Pour charger un projet sélectionnez le menu ‘**Projet - Ouvrir - Projet**’ qui ouvrira une fenêtre vous permettant de choisir un projet. Il est également possible de charger uniquement la structure d’un projet, ce qui veut dire que le projet entier est chargé à l’exception des données enregistrées. Pour cela, choisir le menu ‘**Projet - Ouvrir - Structure**’.

Si le projet chargé a son programme source placé à l’extérieur, le fichier source externe sera créé après ouverture du projet (voir voir [Section 15.2 \[External program source\]](#), [page 75](#)).

Si vous chargez un projet ou la structure d’un projet alors que le projet courant n’a pas été sauvé, une fenêtre de sûreté s’ouvrira vous demandant de confirmer l’opération.

6.6 Enregistrer le projet

Tous les changements effectués sur un projet se font uniquement en mémoire ou temporairement stockés lors de déchargements d’enregistrements (voir voir [Section 6.9 \[Swap records\]](#), [page 33](#)). Aussi, si vous voulez les stocker de façon permanentes, vous devez enregistrer le projet sur votre disque dur, en choisissant le menu ‘**Projet - Enregistrer**’. Si votre projet n’a pas encore de nom, une requête de fichier vous demandant un nom sera tout d’abord ouverte.

La raison pour laquelle MUIbase ne sauve pas automatiquement un projet qui a changé, est que, de cette façon, c’est vous qui décidez lorsque vous voulez sauver un projet. Vous pouvez toujours retourner sur la dernière version sauvée de votre projet en choisissant le menu ‘**Projet - Rétablir la dernière sauvegarde**’. Ce mécanisme est semblable aux commandes ‘**COMMIT**’ et ‘**ROLLBACK**’ des systèmes de bases de données SQL.

Si vous enregistrez un projet, tous les enregistrements modifiés sont écrits sur le disque dur et le fichier ‘**Structure.mb**’ est recréé. Avant de créer le nouveau fichier ‘**Structure.mb**’, MUIbase renomme d’abord le fichier existant ‘**Structure.mb**’ en ‘**Structure.old**’ pour avoir une copie de sécurité au cas où l’opération échouerait.

Ce mécanisme garantit le chargement et la sauvegarde rapide des opérations, mais il n’est pas exempté de réorganisation. Si vous avez modifié beaucoup d’enregistrements alors l’emplacement physique où les enregistrements se trouvent et la fragmentation qui en résulte peut devenir désavantageuse. C’est pourquoi un menu ‘**Projet - Enregistrer & réorganiser**’ existe, pour réaliser l’opération d’enregistrement & de réorganisation. Cette opération peut prendre un certain temps selon le nombre et la taille de vos enregistrements. L’opération enregistrer & réorganiser crée un nouveau tiroir et réécrit tous les fichiers reliés à un projet. Une fois cette opération réalisée, l’ancien tiroir est supprimé.

Un autre bon moment pour programmer une réorganisation est lorsque vous avez fait des changements sur la structure des données d’un projet, par exemple après avoir ajouté un nouveau champ dans une table. Ces changements ne sont pas appliqués immédiatement

à tous les enregistrements parce que cela prendrait trop de temps de charger chaque enregistrement, pour le modifier, et le sauver de nouveau sur le disque. Par conséquent ces changements sont mis sur une liste interne ‘à faire’ qui est appliquée après le chargement d’un enregistrement. L’application de cette liste à un enregistrement ne prend que peu de temps. Cependant plus la liste devient longue, plus cela prend de temps. Réorganiser un projet oblige à appliquer cette liste ‘à faire’ à tous les enregistrements, donc si vous avez fait des modifications à la structure du projet alors sa réorganisation réduira le temps de chargement des enregistrements.

Vous pouvez également enregistrer et réorganiser un projet sous un nouveau nom, ce qui conserve l’ancien projet intact. Pour faire cela il suffit d’utiliser le menu ‘**Projet – Enregistrer & réorganiser en...**’ qui vous demande de saisir un nouveau nom de projet.

6.7 Modes Administrateur et Utilisateur

MUIbase travaille soit en mode administrateur (mode par défaut), soit en mode utilisateur. Vous pouvez basculer d’un mode à l’autre en sélectionnant le menu ‘**Projet – Passer en mode Administrateur**’ et ‘**Projet – Passer en mode Utilisateur**’. En mode Utilisateur, plusieurs éléments de menu sont désactivés et les éditeurs de structure, de programme et de requête ne sont pas disponibles. Ainsi seulement l’édition basique d’enregistrement est possible. En mode Administrateur toutes les opérations sont autorisées.

Un mot de passe Administrateur peut être configuré pour un projet en sélectionnant le menu ‘**Projet – Modifier le mot de passe Administrateur**’. Une fois configuré, le mot de passe doit être saisi pour passer en mode Administrateur sinon le basculement sera refusé laissant le projet en mode Utilisateur.

Lors de l’ouverture d’un projet ayant un mot de passe Administrateur configuré, le projet est démarré en mode Utilisateur, dans le cas contraire (aucun mot de passe Administrateur n’est configuré), il est démarré en mode Administrateur.

6.8 Vérifier l’intégrité des données

MUIbase peut vérifier si toutes les données de votre projet sont encore valides et non corrompues par des programmes de corrections de votre fichier projet ou par des accidents systèmes. Choisir le menu ‘**Projet – Vérifier l’intégrité**’ pour démarrer le processus.

Normalement, vous ne devriez jamais avoir besoin de cette fonctionnalité, MUIbase devrait toujours signaler que l’intégrité des données est parfaite. Mais si jamais il vous arrive que votre projet contienne des ‘**erreurs internes**’, c’est-à-dire, quelques enregistrements ne peuvent plus être chargés, vous pourrez réparer votre projet en employant ce menu.

MUIbase écrira un fichier journal où il notera tous les enregistrements affectés et vous pourrez ensuite enregistrer et réorganiser le projet. Dans ce journal, les enregistrements qui pourraient être corrompus ainsi que les enregistrements qui ne sont plus accessibles et qui donc pourraient avoir été supprimés sont énumérés par leur numéro d’enregistrement dans l’ancien projet (corrompu) et par leur numéro d’enregistrement (entre parenthèses) dans le projet réorganisé.

6.9 Décharger les enregistrements

MUIbase n'a pas besoin de conserver tous les enregistrements d'un projet en mémoire. Le chargement et la sauvegarde des projets sont beaucoup plus rapide. En chargeant un projet, un en-tête enregistrement est alloué pour chaque enregistrement. Les données elle-même sont chargées seulement une fois lorsque c'est nécessaire, par exemple quand elles sont affichées sur l'écran. Le nombre d'enregistrements est encore limité par la mémoire disponible puisque chaque en-tête enregistrement utilise quelques octets de mémoire.

Vous pouvez spécifier la taille de la mémoire que devrait utiliser MUIbase pour les enregistrements d'un projet. Choisir une des valeurs prédéfinies se trouvant dans le menu '**Préférences - cache d'enregistrements**' (voir voir [Section 7.4.1 \[Record memory\]](#), [page 36](#)). MUIbase ne préaffecte pas un bloc mémoire de la taille indiquée, il vérifie seulement de temps en temps si la quantité courante de mémoire allouée est plus grande que la valeur indiquée.

Si MUIbase manque de mémoire ou si la limite supérieure de la capacité mémoire a été atteinte alors MUIbase essaye de libérer autant de d'enregistrements de la mémoire que possible. Dans ce cas-là, MUIbase peut écrire les enregistrements modifiés sur le disque dur pour obtenir le maximum de mémoire disponible. Vous pouvez également forcer MUIbase à le faire en choisissant le menu '**Projet - Décharger les enregistrements**'.

MUIbase maintient une liste d'espace libre pour chaque fichier d'enregistrements. Si vous supprimez un enregistrement, l'emplacement dans le fichier d'enregistrements est ajouté à la liste d'espace libre. En outre, si vous changez un enregistrement et que celui-ci a besoin d'être écrit sur le disque dur, l'ancien emplacement dans le fichier est ajouté à la liste d'espace libre. Cependant MUIbase s'assure qu'un rechargement vous permettra toujours de retourner au point de la dernière opération de sauvegarde. MUIbase n'écrit pas sur les secteurs qui sont libres mais où un enregistrement existe toujours et qui pourra être accédé en rouvrant le projet.

6.10 Fermer le Projet

Quand vous êtes éditer un projet vous pouvez le fermer en sélectionnant le menu '**Projet - Fermer**'. Ce qui libère la mémoire et toutes les ressources appartenant au projet. Si le projet contient des changements qui n'ont pas encore été sauvés, une fenêtre de sécurité s'ouvrira vous demandant de sauver, de continuer ou d'annuler l'opération.

Pour la fermeture d'un projet vous pouvez également choisir le menu '**Projet - Sauver & Fermer**' qui sauve d'abord le projet s'il y a eu des changements avant de le clôturer.

7 Préférences

MUIbase offre à l'utilisateur un certain nombre d'éléments de préférence qu'il peut configurer selon ses goûts. Ce chapitre présente quels éléments de préférences sont disponibles et donne des informations générales sur la façon dont le système de préférence fonctionne.

Les éléments des préférences peuvent être divisés en réglages utilisateur et réglages projet.

7.1 Réglages utilisateur

Les réglages utilisateur comprennent les éléments de configuration dépendant de l'utilisateur, p. ex. la langue de l'utilisateur, son pays ou ses goûts. Les réglages utilisateur sont affichés et peuvent être configurés en haut du menu 'Préférences'.

Les réglages utilisateur sont stockés dans l'environnement de l'utilisateur. Sous Linux et Windows ils sont conservés dans le répertoire personnel de l'utilisateur dans '.MUIbase.prefs'. Sur Amiga ils sont stockés dans 'ENV:MUIbase.prefs' et 'ENVARC:MUIbase.prefs'.

Les réglages utilisateur contiennent les éléments de configuration listés ci-dessous. À chaque fois que l'un de ces éléments est modifié dans le menu 'Préférences', les réglages sont sauvegardés sur disque afin de les rendre permanents.

7.1.1 Formats

En sélectionnant le menu 'Préférences - Formats' vous pouvez spécifier les formats utilisés pour l'affichage ou l'impression des valeurs décimales et de type date. La sélection de ce menu ouvre une nouvelle fenêtre contenant les éléments suivants :

- un champ 'Format des décimaux' pour positionner le caractère marquant le début de la partie décimale. Vous pouvez choisir entre 'Point décimal' et 'Virgule décimale'.
- un champ 'Format des dates' pour spécifier comment les valeurs de type date sont affichées. Vous avez le choix entre 'Jour.Mois.Année', 'Mois/Jour/Année' et 'Année-Mois-Jour'.
- deux boutons 'Ok' et 'Annuler' pour quitter la fenêtre.

Les valeurs initiales pour les formats des décimaux et des dates sont déterminées en fonction des informations fournies par l'environnement de localisation ('locale') du système d'exploitation.

Lorsque vous avez terminé, pressez le bouton 'Ok' pour quitter la fenêtre et mettre à jour l'affichage.

7.1.2 Editeur externe

En plus de fournir un éditeur intégré via le système d'interface graphique utilisé, MUIbase propose également d'éditer les contenus textuels dans un éditeur externe (p.ex. le menu contextuel de l'éditeur interne propose un élément permettant d'appeler l'éditeur externe pour éditer le contenu). Le nom de l'éditeur ainsi que ses paramètres sont spécifiés en choisissant le menu 'Préférences - Editeur externe'. Vous devez saisir la ligne de commande à exécuter lors de l'appel de l'éditeur externe. La chaîne spéciale '%f' peut être utilisée pour

symboliser le nom du fichier et est remplacée avant l'exécution par le nom réel du fichier (ou un nom de fichier temporaire que MUIbase crée pour échanger les données textuelles).

Par exemple, sur Linux, vous pouvez spécifier '`emacs %f`' pour utiliser le puissant éditeur GNU Emacs ; sur Amiga, vous pouvez utiliser '`CED %f -keepio`' pour basculer vers le célèbre éditeur CED comme éditeur externe.

Par défaut la valeur est '`gvim -f %f`' sur Linux, '`Notepad %f`' sur Windows et '`Ed %f`' sur Amiga.

Sur Amiga, la séquence additionnelle '`%p`' peut-être utilisée dans la ligne de commande. Lors du lancement de l'éditeur externe, cette séquence est remplacée par le nom de l'écran publique sur lequel MUIbase est affiché.

7.1.3 Visionneuse externe

MUIbase utilise une visionneuse externe pour afficher le contenu des fichiers externes, p. ex. les images, les films et toute sorte de documents. Par exemple vous pouvez utiliser le type de donnée texte pour stocker des noms de fichier dans une table puis laisser MUIbase les afficher en utilisant la visionneuse externe. Pour spécifier cette visionneuse, utilisez le menu '**Préférences - Visionneuse externe**'. Comme pour l'éditeur externe (voir [Section 7.1.2 \[External editor\]](#), page 34), vous pouvez alors saisir une ligne de commande.

La valeur par défaut est '`gnome-open %f`' sur Linux, '`%f`' sur Windows (en utilisant ShellExecute) et '`Multiview %f`' sur Amiga.

Sur Amiga, la séquence additionnelle '`%p`' peut-être utilisée dans la ligne de commande. Lors du lancement de l'éditeur externe, cette séquence est remplacée par le nom de l'écran publique sur lequel MUIbase est affiché.

7.1.4 Boutons dans cycle de tabulation

Dans l'interface graphique spécifiée dans l'éditeur de structure, il est possible qu'il existe un certain nombre de boutons spéciaux. Par boutons spéciaux s'entendent les boutons déroulants, p. ex. les boutons de sélection de fichier ou de police attachés à un champ texte, les boutons d'affichage automatique et de filtrage d'un champ Référence, ainsi que les boutons de visualisation à côté d'un champ texte.

Ces boutons ne sont généralement pas inclus dans le cycle de tabulation, c'est à dire qu'il est impossible de les activer avec la touche **Tab**. Cependant en sélectionnant le menu '**Préférences - Boutons dans cycle de tabulation**' ces boutons seront tout de même inclus dans le cycle de tabulation. Cette option est activée par défaut.

Sur Amiga, notez que la modification de l'état de cette option n'a d'effet qu'après reconstruction de l'interface graphique, c'est à dire en basculant vers l'éditeur de structure puis en revenant dans l'interface graphique.

7.1.5 Champ suivant via <Entrée>

Lorsque le curseur est dans un champ éditable texte simple dans l'interface graphique d'un projet, presser la touche **Entrée** peut soit passer au prochain élément de l'interface graphique, soit laisser le curseur dans le champ texte courant.

En cochant l'élément de menu '**Préférences - Champ suivant via <Entrée>**' le curseur passera au prochain élément de l'interface graphique, dans le cas contraire il restera au champ texte courant. Par défaut ce menu est coché.

Veillez noter que dans tous les cas, presser **Entrée** confirme et mémorise le texte saisi.

7.2 Confirmer la sortie

Si vous tenez de quitter MUIbase et qu'il existe des projets non sauves, une requête de sécurité est ouverte pour vous demander de confirmer. En revanche si tous les projets sont sauves, MUIbase se termine en silence.

Afin de toujours forcer l'affichage d'une requête lors de la sortie de MUIbase, le menu '**Préférences - Confirmer la sortie**' doit être coché. Dans ce cas il est possible de toujours obtenir une requête de sécurité lors de la sélection du menu '**Projet - Quitter**' ou lorsque le dernier projet ouvert est fermé. Par défaut cette option est décochée.

7.3 MUI

Sur Amiga uniquement. Comme MUIbase pour Amiga est une application MUI, vous pouvez également indiquer vos préférences MUI pour cette application en utilisant le menu '**Préférences - MUI**'.

7.4 Réglages dépendants du projet

Les réglages projet contiennent les éléments de configuration stockés avec le projet. Ces éléments sont affichés et modifiables depuis la partie inférieure du menu '**Préférences**' et dans le menu '**Programme**' (la raison de l'étalement dans ces deux menus est de conserver les menus compacts et clairs).

Les éléments de configuration suivants appartiennent aux réglages projet. À chaque fois que l'un des ces éléments est modifié, le compteur de modification du projet est incrémenté.

7.4.1 Cache d'enregistrements

MUIbase n'a pas besoin de conserver tous les enregistrements d'un projet en mémoire. À la place il utilise un cache pour ne maintenir en mémoire qu'un petit nombre d'enregistrements. En choisissant une valeur dans le menu '**Préférences - Cache d'enregistrements**' il est possible de donner la taille de ce cache. Chaque projet possède son propre cache, donc si vous ouvrez deux projets chacun avec un cache de 1 MO, MUIbase utilisera jusqu'à 2 MO au total pour les enregistrements des deux projets.

MUIbase n'alloue pas cette mémoire a priori, mais utilise un schéma d'allocation dynamique. De plus la taille spécifiée du cache n'est qu'une limite logicielle et occasionnellement, MUIbase peut être amené à la dépasser.

Lorsque que le cache est plein ou si MUIbase vient à manquer de mémoire, tous les enregistrements sont supprimés du cache. Cela signifie que les enregistrements non modifiés sont simplement libérés tandis que ceux qui ont été modifiés sont d'abord écrits sur le disque puis libérés (voir [Section 6.9 \[Swap records\]](#), page 33).

En donnant à MUIbase une plus grande valeur pour le cache d'enregistrements, vous pourrez noter une augmentation de la rapidité d'accès aux enregistrements car un plus grand nombre d'enregistrements pourront tenir en mémoire ce qui diminue le nombre d'accès disque nécessaires. S'il y a suffisamment de mémoire pour contenir la totalité des enregistrements en mémoire et que vous avez indiqué une limite de cache suffisante (par exemple '**illimité**') alors MUIbase travaille en vitesse optimale.

Par défaut, la valeur est ‘illimité’.

7.5 Confirmer la suppression d’enregistrement

Vous devez cocher l’élément de menu ‘**Préférences - Confirmer la suppression d’enregistrement**’ si vous souhaitez que MUIbase ouvre une requête de sécurité vous demandant de confirmer à chaque fois que vous essayez de supprimer un enregistrement. Laissez l’élément décoché si les enregistrements doivent être supprimés silencieusement.

Par défaut, l’option est cochée.

7.5.1 Chemins relatifs au projet

Si vous voulez référencer des données externes (p. ex. des documents ou des images) alors le nom de fichier de ces données doit être stocké dans le projet. Pour cela vous pouvez utiliser des chemins absolus ou des chemins contenant des assignations (voir [Section 3.7 \[Filename conventions\]](#), page 7). Un autre moyen est de stocker les données externes relativement au répertoire du projet (note : ce n’est pas le répertoire du projet lui-même mais le répertoire contenant le projet).

En activant le menu ‘**Préférences - Chemins relatifs au répertoire du projet**’, MUIbase change le répertoire de travail pour celui contenant le projet. Cela signifie que si vous avez plusieurs projets ouverts, MUIbase change de répertoire en fonction du projet actif. Une fois que cet élément de menu a été activé pour un projet, les noms de fichiers peuvent être donnés de manière relative au répertoire du projet. Cela rend un projet indépendant de l’endroit où il est stocké sur le système de fichiers.

Si vous laissez ce menu décoché, MUIbase utilise le répertoire de travail actif au moment où le programme a été lancé.

Par défaut ce menu est décoché.

7.5.2 Confirmer enregistrement et réorganisation

L’opération d’enregistrement et réorganisation d’un projet peut prendre un certain temps en fonction de la taille du projet. C’est pour cela que lors de la sélection des menus ‘**Projet - Enregistrer & réorganiser**’ ou ‘**Projet - Enregistrer & réorganiser en**’, une requête de sécurité est ouverte pour demander la confirmation de l’opération. Cette requête n’apparaît que si le menu ‘**Préférences - Confirmer enregistrement et réorganisation**’ est coché, il est donc possible de la désactiver en décochant ce menu. Par défaut ce menu est coché.

7.5.3 Code source du programme

En utilisant le menu ‘**Programme - Code Source**’, vous pouvez définir si le code source du programme d’un projet est ‘Interne’ ou ‘Externe’. Lorsqu’il est défini comme ‘Interne’, il est possible d’utiliser l’éditeur intégré de MUIbase pour éditer et compiler le programme du projet. C’est la valeur par défaut. Si vous souhaitez utiliser un éditeur externe pour programmer, choisissez ‘Externe’ et saisissez le nom du nouveau fichier dans lequel MUIbase va ensuite stocker le code source du programme. Cela vous permet de charger et éditer le code source du programme dans votre éditeur favori. Pour plus d’informations sur la façon d’utiliser cette fonctionnalité voir [Section 15.2 \[External program source\]](#), page 75.

Notez qu'en basculant de 'Externe' à 'Interne', la dernière compilation réussie du programme est conservée.

7.5.4 Nettoyer les sources des programmes externes

Le menu 'Programme - Nettoyer les sources des programmes externes' indique si les fichiers sources externes des programmes doivent être effacés lors de la fermeture du projet ou de la modification du menu 'Programme - Code Source' vers 'Interne'. Voir [Section 15.2 \[External program source\], page 75](#), pour plus d'informations à propos des codes sources externes. Par défaut cette option est activée.

7.5.5 Information de débogage

Lors de la compilation d'un programme du projet, il est possible de choisir si les informations de débogage doivent être incluses dans l'exécutable ou non. Si vous compilez sans ces informations et qu'une erreur survient à l'exécution, une description d'erreur est générée, mais il n'y a pas d'indication sur l'endroit exact où est survenue cette erreur. Si vous compilez avec ces informations, vous obtenez en plus l'endroit exact de l'erreur. Compiler avec les informations de débogage est un peu plus long, nécessite plus de mémoire et le programme résultant est légèrement moins efficace.

Utilisez le menu 'Programme - Inclure information de débogage' pour activer ou désactiver les informations de débogage. Par défaut les informations sont activées. Lorsque vous changez cette option, n'oubliez pas de recompiler le programme du projet en sélectionnant le menu 'Programme - Compiler'.

7.5.6 Fonctions obsolètes

Depuis la version 2.7 de MUIbase quelques fonctions de programmation ont été rendues obsolètes (voir [Section 15.30 \[List of obsolete functions\], page 152](#)). Ces fonctions obsolètes ont été remplacées par d'autres mécanismes et elles ne fonctionnent plus comme attendu. Il est possible de choisir comment gérer le cas lors de l'ouverture de projets contenant des fonctions de programmation devenues obsolètes.

Le menu 'Programme - Fonctions obsolètes' détermine comment réagir lorsqu'une fonction obsolète est appelée. En choisissant 'Ignorer silencieusement' tout appel à une fonction obsolète sera ignoré et l'exécution du programme continue en sautant l'appel à la fonction. 'Avertir à l'appel' ouvre une boîte de dialogue informant l'utilisateur qu'une fonction obsolète est appelée et lui permet de poursuivre l'exécution ou de montrer l'emplacement de l'appel dans l'éditeur de programme. C'est l'action par défaut. Si 'Traiter comme une erreur' est sélectionné, tout appel à une fonction obsolète génère une erreur et la compilation de programmes contenant des fonctions obsolètes échoue avec un message d'erreur approprié.

Il est recommandé de choisir 'Traiter comme une erreur' après que tous les appels aux fonctions obsolètes ont été supprimés du programme d'un projet. Afin de trouver si votre projet contient des appels à des fonctions obsolètes choisissez 'Traiter comme une erreur' et recompilez le programme du projet.

7.5.7 Trier les déclencheurs

En cochant le menu 'Préférences - Trier les déclencheurs', les fonctions disponibles en tant que déclencheurs sont triées par ordre alphabétique lors de leur affichage dans les

boîtes de dialogue de création et de modification de table (voir [Section 14.1.1 \[Creating tables\]](#), page 60) et dans celles de création et modification de champ (voir [Section 14.2.1 \[Creating attributes\]](#), page 62). Dans le cas contraire, les fonctions sont affichées dans leur ordre d'apparition dans le programme du projet. Cette option est désactivée par défaut.

7.5.8 Répertoire d'inclusion

La fonction de programmation de MUIbase autorise l'inclusion dans le programme du projet de fichiers sources externes (voir [Section 15.3.3 \[#include\]](#), page 77 pour plus de détails). Le menu 'Programme - Répertoire d'inclusion' permet de spécifier le répertoire où MUIbase doit rechercher ces fichiers d'inclusion. La valeur par défaut est 'MUIbase:Include'.

7.5.9 Fichier de sortie de programme

Lors de l'exécution d'un programme MUIbase, toutes les sorties à destination de la sortie standard ('`stdout`') sont redirigées vers un fichier. Le nom de ce fichier peut être saisi dans une boîte de dialogue ouverte lorsque l'on sélectionne le menu 'Program - Fichier de sortie de programme'. Vous pouvez également spécifier si les sorties doivent être ajoutées en fin de fichier ou si le fichier doit être réinitialisé (i.e. effacé puis recréé) à la première sortie.

Sur Linux et Windows, vous pouvez diriger la sortie vers un programme externe pour en traiter les données. Pour cela, le premier caractère du nom de fichier doit être le symbole pipe '`|`' suivi du nom du programme externe et de ses arguments. Le programme externe doit lire les données à partir de son entrée standard ('`stdin`'). Par exemple, cela permet sous Linux les redirections suivantes :

- '`|lpr`' impression de la sortie sur l'imprimante par défaut.
- '`|mknod /tmp/pipe p; (xterm -e less -f /tmp/pipe &); cat > /tmp/pipe; rm /tmp/pipe`' affichage de la sortie en utilisant le programme 'less' dans sa propre fenêtre 'xterm'.
- '`/dev/pts/0`' redirection de la sortie sur le terminal connecté sur '`pts/0`'. Sur certains bureaux, p.ex. KDE, un démon est à l'écoute de ce terminal et la sortie est redirigée dans une fenêtre.
- '`/dev/stdout`' redirection de la sortie sur le canal de sortie de MUIbase. C'est la valeur par défaut.

Sur Amiga, il existe quelques noms spéciaux pour rediriger la sortie, p.ex. :

- '`PRT:`' impression de la sortie sur l'imprimante.
- '`CON:////MUIbase output/CLOSE/WAIT`' affichage de la sortie dans une fenêtre Shell.
- '`CONSOLE:`' redirection de la sortie vers la fenêtre Shell depuis laquelle MUIbase a été lancé. C'est la valeur par défaut.

7.6 Enregistrer comme défaut

Les réglages projet sont spécifiques à chaque projet, c.-à-d. des projets différents peuvent avoir des réglages différents. Pour les nouveaux projets des réglages par défaut sont choisis. Ces réglages par défaut sont stockés avec les réglages utilisateurs dans l'environnement de l'utilisateur (voir [Section 7.1 \[User settings\]](#), page 34).

Afin de personnaliser les réglages par défaut que vous souhaitez, vous pouvez stocker les réglages du projet courant comme étant les réglages par défaut des nouveaux projets en sélectionnant le menu **‘Préférences – Enregistrer comme défaut’**.

8 Edition d'enregistrement

Ce chapitre décrit comment ajouter, modifier, effacer et parcourir les enregistrement d'une table de base de données.

8.1 Objet actif

MUIbase utilise un curseur pour indiquer quel est l'objet actif. Si l'objet actif est un objet chaîne, le curseur texte habituel apparaît, les autres objets sont encadrés par un cadre spécial. Il est possible de cycliser les objets actifs par pression des touches **Tab** ou **Shift-Tab**. En pressant les touches **Help** ou **F1**, une visionneuse externe apparaît avec des informations utiles à propos de l'objet actif.

La table à laquelle l'objet actif appartient est appelée la *table courante*. Le panel d'une table peut devenir l'objet actif, cela assure qu'il est toujours possible de faire passer une table, en table courante, même si celle-ci ne contient aucun objet activable.

Sous Linux et Windows, chaque table dispose d'un menu contextuel avec les opérations permettant la manipulation de la table. Ce menu contextuel peut être ouvert en pressant le bouton droit de la souris n'importe où dans la fiche de la table (mais en dehors de tout autre élément graphique qui pourrait avoir son propre menu contextuel).

Sur Amiga les éléments de menu liés à la table font partie du menu global qui se trouve en haut de l'écran.

8.2 Ajout d'enregistrement

En sélectionnant le menu 'Table - Nouvel enregistrement', un nouvel enregistrement est alloué dans la table courante. Il est initialisé avec les valeurs initiales pour chacun des champs. Il est également possible de dupliquer l'enregistrement courant de la table courante en sélectionnant le menu 'Table - Dupliquer enregistrement'.

Si un déclencheur pour la création d'enregistrement a été installé (voir [Section 14.1.1 \[Creating tables\]](#), page 60) alors celui est appelé pour créer l'enregistrement. Pour plus de détails sur ce mécanisme, Voir [Section 15.29.5 \[New trigger\]](#), page 147.

8.3 Modification d'enregistrement

Pour modifier l'enregistrement courant d'une table, il suffit d'activer n'importe quel objet champ dans la fiche de la table et de saisir une nouvelle valeur. Pour les champs de type texte, entier, réel, date, heure et mémo il est possible d'utiliser les commandes d'édition habituelles.

Un objet champ peut être configuré en lecture seule, dans ce cas il n'est pas possible de changer sa valeur (exception : les champs texte avec un bouton pop-up).

8.3.1 Champ texte avec bouton pop-up

Si un champ texte dispose d'un bouton attaché alors il est possible de cliquer sur ce bouton pour obtenir une requête pour saisir le contenu de la chaîne, p. ex. une requête de fichier pour sélectionner un nom de fichier ou une liste de chaînes pour en sélectionner une. Le bouton pop-up peut toujours être utilisé pour positionner la valeur du champ texte même si celui-ci est en lecture seule.

À droite du champ texte un autre petit bouton peut être présent. Une pression sur ce bouton lance la visionneuse externe pour afficher le fichier spécifié dans le champ texte.

8.3.2 Saisie de valeurs entières

Lors de la saisie d'un nombre entier, il est possible d'utiliser une notation octale (préfixe 0) ou hexadécimale (préfixe 0x) en plus de la notation décimale classique.

8.3.3 Saisie de valeurs booléennes

Etat d'activation d'un champ booléen peut être inversé en cliquant avec le bouton gauche de la souris ou en pressant la barre d'espace si c'est l'objet actif.

8.3.4 Saisie de valeurs de choix

Pour les champs de type choix, il est possible de sélectionner une valeur en cliquant sur le champ ou en utilisant les touches du curseur *Haut* et *Bas* pour parcourir tous les labels des choix.

8.3.5 Saisie de valeurs de date

Les valeurs de date peuvent être saisies dans l'un des formats 'JJ.MM.AAAA', 'MM/JJ/AAAA' ou 'AAAA-MM-JJ', où 'JJ', 'MM' et 'AAAA' représentent des valeurs de deux ou quatre chiffres représentant respectivement le jour, le mois et l'année de la date. Il est possible d'omettre la valeur de l'année d'une date, dans ce cas, l'année courante est utilisée.

En insérant une unique valeur entière, il est possible de spécifier une date relative à la date courante, p. ex. en entrant '0' la date du jour est utilisée, de même en entrant '-1' la date de la veille.

8.3.6 Saisie de valeurs horaires

Le format de saisie de valeurs horaires est spécifié dans l'éditeur de structure (voir [Section 14.3.3 \[Attribute object editor\], page 67](#)). Les formats possibles sont 'HH:MM:SS', 'MM:SS' et 'HH:MM' où 'HH' représente les heures, 'MM' les minutes, et 'SS' les secondes.

Il est possible d'omettre certaines parties du format, p. ex. saisir '6:30' pour un format 'HH:MM:SS' est interprété comme '00:06:30'. Lors de la saisie d'un nombre unique, il est considéré respectivement comme le nombre de secondes pour les formats 'HH:MM:SS' et 'MM:SS' et comme le nombre de minutes pour le format 'HH:MM'.

8.3.7 Menu contextuel des Mémos

Les champs Mémo possèdent un menu contextuel offrant des possibilités d'édition supplémentaires :

- 'Couper', 'Copier', et 'Coller' permettent d'échanger des données avec le presse-papiers.
- 'Supprimer' efface le texte sélectionné, tandis que 'Tout sélectionner' permet de sélectionner tout le texte (Linux et Windows).
- 'Effacer' efface tout le texte du mémo (Amiga).
- 'Annuler' et 'Refaire' permet de défaire et refaire les modifications faites sur le contenu du mémo (uniquement Amiga).

- ‘Méthodes de saisie’ et ‘Insérer caractère de contrôle Unicode’ sont des éléments de menus spécifiques à GTK (Linux et Windows), se référer à la documentation GTK.
- Avec ‘Ouvrir texte’ et ‘Sauver texte’ il est possible de charger et sauver le contenu du mémo depuis/vers un fichier.
- ‘Editeur externe’ lance un éditeur externe pour éditer le mémo, pour plus d’informations sur l’éditeur externe voir Voir [Section 7.1.2 \[External editor\]](#), page 34.

8.3.8 Saisie de valeur de type Référence

For reference attributes the record reference can be entered through a pop-up list:

- To the right of a reference attribute you find a pop-up button which, if pressed, opens a list of records. Choose a record from the list to set the reference to this record, ‘Initial’ to set the reference to the NIL value, or ‘Current’ to set the reference to the current record of the referenced table.
- You can search for an entry in the referenced table by using the keyboard. After the first key press an input field is opened allowing to enter more characters for the search pattern. After each key press, a search is started immediately (case-insensitive) and the first matching entry is selected. The search method can be specified in the display object of the attribute (voir [Section 14.3.3 \[Attribute object editor\]](#), page 67) under the ‘Quick search’ category. You can use the characters ‘*’ for matching an arbitrary sequence of characters and ‘?’ for matching exactly one arbitrary character. Using the cursor keys *Down* and *Up*, the next respectively previous matching entry is selected. A selected entry is stored when acknowledged by pressing the *Enter* key. If you leave the search window by other means, e.g. pressing *Esc*, then the attribute stays unchanged and keeps its previous value.

8.3.9 Saisie de valeur NIL

Pour saisir la valeur NIL, il suffit de saisir une valeur invalide pour le type du champ, p. ex. entrer ‘xyz’ dans un champ entier, la valeur de ce champ est alors positionnée à NIL. A noter cependant que tous les types de champ ne supportent pas la valeur NIL. Voir [Section 5.6 \[Table of attribute types\]](#), page 23, pour un aperçu des types de champ.

8.4 Suppression d'enregistrement

Le menu ‘Table - Effacer l’enregistrement’ permet de supprimer l’enregistrement courant. Avant de supprimer l’enregistrement une requête de sécurité peut demander la confirmation. Il est possible d’activer et de désactiver cette requête dans les préférences de configuration (voir [Section 7.5 \[Record delete requester\]](#), page 37).

Si un déclencheur a été installé pour la suppression d’enregistrement (voir [Section 14.1.1 \[Creating tables\]](#), page 60) alors celui-ci est appelé pour supprimer l’enregistrement. Pour plus d’information sur ce mécanisme voir [Section 15.29.6 \[Delete trigger\]](#), page 148.

Il est également possible de supprimer tous les enregistrements d’une table en sélectionnant le menu ‘Table - Supprimer tous les enregistrements’. Seuls les enregistrements correspondant au filtre d’enregistrement de la table considérée sont supprimés. Avant la suppression, une confirmation peut apparaître s’elle a été activée. Aucun déclencheur n’est appelé lors de la suppression de tous les enregistrements.

8.5 Parcourir les enregistrements

Pour afficher d'autres enregistrements que celui en cours, il suffit de sélectionner l'un des sous-menus du menu 'Table - Atteindre l'enregistrement'. Il est possible d'aller à l'enregistrement précédent, suivant, le premier, le dernier, sauter plusieurs enregistrements en arrière ou en avant ou même de saisir le numéro de l'enregistrement que l'on désire afficher. Dans ce contexte, le numéro d'enregistrement est le numéro affiché dans le panel correspondant (voir [Section 5.9.3 \[Panels\], page 28](#)). Le panel peut également inclure deux boutons fléchés pour naviguer vers l'enregistrement précédent et suivant.

Le parcours d'enregistrements peut facilement être réalisé en utilisant les touches du curseur *Haut* et *Bas* en combinaison avec les touches *Shift*, *Alt*, et *Ctrl*. Toutes les combinaisons possibles sont listées dans les éléments du menu 'Table - Atteindre l'enregistrement' ainsi que dans le tableau suivant.

	<i>Alt</i>	<i>Shift-Ctrl</i>	<i>Shift-Alt</i>
<i>Up</i>	Enregistrement précédent	Premier enregistrement	Sauter en arrière
<i>Down</i>	Enregistrement suivant	Dernier enregistrement	Sauter en avant

9 Filtre

Des filtres peuvent être utilisés pour masquer des enregistrements. Ce chapitre décrit les différents types de filtre disponibles et la façon de les utiliser.

MUIbase connaît deux types de filtres : les filtres d'enregistrement et les filtres de référence.

9.1 Filtre d'enregistrement

Un filtre d'enregistrement peut être positionné sur une table pour filtrer les enregistrements qui ne vous intéressent pas. Les enregistrements filtrés sont exclus de l'affichage de la table et par conséquent l'utilisateur ne peut ni les voir, ni les parcourir.

9.1.1 Expression de filtrage

Un filtre est défini en donnant une expression booléenne qui peut contenir des appels à des fonctions de programmation MUIbase. Pour chaque enregistrement de la table sur laquelle est spécifié le filtre, cette expression est évaluée. Si elle retourne NULL (ou NIL) alors l'enregistrement est filtré, sinon l'enregistrement est accepté.

Chaque table peut avoir sa propre expression de filtrage.

9.1.2 Modifier les filtres

Pour modifier le filtre de la table courante, sélectionnez le menu 'Table - Changer le filtre'. Cela ouvrira une fenêtre contenant :

- dans le titre de la fenêtre, le nom de la table sur laquelle vous installez le filtre.
- une liste de tous les champs de la table pouvant être utilisés dans l'expression de filtrage. Cette liste est placée sur la gauche de la fenêtre. Si vous double-cliquez sur un nom alors celui-ci sera inséré dans l'expression de filtrage à la position courante du curseur.
- une banque de boutons reprenant les fonctions de programmation et les opérateurs de MUIbase est placée sur la droite de la fenêtre. Cliquez sur l'un des boutons pour entrer la fonction correspondante dans l'expression de filtrage. Veuillez noter que la liste présentée des fonctions et opérateurs n'est pas exhaustive, les autres fonctions de MUIbase qui sont pas présentes dans l'un des boutons doivent être saisies manuellement. Seules les fonctions MUIbase qui n'ont pas d'effet de bord peuvent être utilisées, p.ex. il n'est pas possible d'écrire des données dans un fichier au sein d'une expression de filtrage.
- un champ de saisie de chaîne pour entrer l'expression de filtrage. Les noms des champs, des fonctions et des opérateurs sont insérés ici. Vous pouvez également directement saisir votre expression de filtrage.
- deux boutons 'Ok' et 'Annuler' pour refermer la fenêtre.

Après avoir spécifié l'expression de filtrage, cliquez sur le bouton 'Ok' pour quitter la fenêtre. L'expression saisie est alors compilée, en cas de compilation réussie, l'expression est appliquée à tous les enregistrements. Ceux pour qui l'expression booléenne n'est pas vérifiée sont retirés de l'affichage de la table.

Si l'expression ne compile pas, vous obtiendrez un message d'erreur dans la barre de titre de la fenêtre.

Un filtre peut être activé ou désactivé en cliquant sur le bouton ‘F’ dans le panneau de la table (s’il celui-ci est activé). Après la spécification d’une expression de filtrage sur une table, le filtrage de cette table est automatiquement activé.

Si vous (ré)activez un filtre alors tous les enregistrements de la table sont analysés pour vérifier s’ils répondent ou non au filtre.

Lorsqu’un filtre est activé et que vous changez la valeur d’un champ (employé dans le filtre) dans un enregistrement de cette table, la correspondance avec le filtre de cet enregistrement n’est pas réévaluée et reste inchangée.

Si vous allouez un nouvel enregistrement dans une table ayant un filtre actif, aucune vérification n’est faite sur le nouvel enregistrement quant à sa correspondance avec le filtre, il a son état de vérification automatiquement positionné à VRAI.

9.1.3 Exemples de filtre

Voici quelques exemples d’expression de filtrage valides :

- ‘NIL’ filtre tous les enregistrements.
- ‘TRUE’ ne filtre aucun enregistrement (conserve tout).
- ‘0’ a le même effet que ‘TRUE’ car pour MUIbase toute expression différente de NIL est considérée VRAI.
- ‘(> Montant 100.0)’ n’affiche que les enregistrements dont le champ ‘Montant’ est supérieur à 100.0 (on suppose ici que la table dispose d’un champ ‘Montant’ de type décimal).
- ‘(NOT (LIKE Nom "*x*))’ filtre tous les enregistrements ayant la lettre ‘x’ dans le champ ‘Nom’ (de type chaîne).

Notez que le langage de programmation de MUIbase utilise une syntaxe similaire à celle de Lisp. Pour plus d’informations concernant le langage de programmation, voir [Chapitre 15 \[Programming MUIbase\]](#), page 75.

9.2 Filtre de référence

Les champs de type référence peuvent également avoir un comportement de filtre. Cela est utile par exemple pour construire une hiérarchie de tables. Comme exemple voyez le projet ‘Albums’.

Si le filtre d’un champ référence est activé alors les fonctionnalités suivantes le sont aussi :

1. L’utilisateur ne peut accéder qu’aux enregistrements de la table pour lesquels la référence pointe sur l’enregistrement courant de la table référencée.
2. Si la table référencée change d’enregistrement courant, un nouvel enregistrement courant est également recherché et activé dans la table du champ référence.
3. Lors de l’allocation d’un nouvel enregistrement, la référence est automatiquement positionnée à l’enregistrement courant de la table référencée.

Note: La suppression en cascade doit être implémentée manuellement (en utilisant un déclencheur de suppression).

N’utilisez pas les filtres de référence sur des graphes cycliques, p.ex. une table s’auto-référénçant, car cela n’a pas beaucoup de sens.

10 Ordre

Pour chaque table dans votre base de données vous pouvez indiquer dans quel ordre les enregistrements devraient être affichés. Ce chapitre décrit comment spécifier un tri et quelles conséquences il a.

10.1 Tri vide

Par défaut, chaque table crée récemment à un tri vide. Ce qui signifie que si vous écrivez un nouvel enregistrement, l'enregistrement créé est inséré à la position actuelle (c'est-à-dire, derrière l'enregistrement courant de la table). Si vous changez quelques champs d'un enregistrement, la position de l'enregistrement dans la table reste inchangée.

10.2 Tri par champs

C'est parfois utile pour assortir les enregistrements avec certains champs, comme par exemple avec le champ 'Nom' si la table en a un.

Dans MUIbase, pour chaque table, vous pouvez indiquer une liste de champs par lesquels ses enregistrements devraient être assortis. Tous les enregistrements sont d'abord assortis avec le premier champ de cette liste. Si deux enregistrements sont équivalents pour un champ, c'est le champ suivant dans la liste qui déterminera le tri. Pour chaque champ vous pouvez indiquer si les enregistrements assortis sont croissants ou décroissants.

Pour déterminer le tri des champs, les règles de la table suivante sont utilisées :

Type	Relation d'ordre
Integer Choice	NIL < MIN_INT < ... < -1 < 0 < 1 < ... < MAX_INT (les valeurs choisies sont traitées comme des nombres entiers)
Real	NIL < -HUGE_VAL < ... < -1.0 < 0.0 < 1.0 < ... < HUGE_VAL
String Memo	NIL < "" < ... < "a" < "AA" < "b" < ... (la comparaison d'une chaîne est un cas insensible)
Date	NIL < 1.1.0000 < ... < 31.12.9999
Time	NIL < 00:00:00 < ... < 596523:14:07
Bool	NIL < TRUE
Reference	NIL < any_record (les enregistrements ne sont pas comparables entre eux par le tri)

Si vous avez spécifié un tri pour une table, les enregistrements seront classés automatiquement à chaque fois que vous ajouterez un nouvel enregistrement ou que vous changerez un champ utilisé dans le tri d'un enregistrement.

10.3 Tri par fonction

Il est parfois utile d’avoir une commande de tri plus complexe qu’une simple liste de champs comme celle décrite dans la dernière section. Par exemple, une liste de champs ne peut pas contenir des champs de type Référence, il n’est donc pas possible de trier vos enregistrements selon un champ Référence. Pour la bonne et simple raison qu’avec les champs de type référence, MUIbase est incapable de maintenir à tout moment le tri de vos enregistrements (pour plus de détails, voir la voir [Section 15.29.7 \[Comparison function\]](#), page 148 dans le chapitre de programmation de MUIbase).

Cependant, MUIbase offre également la possibilité d’indiquer une fonction de comparaison pour assortir vos enregistrements. Vous pouvez indiquer n’importe quelle fonction que vous avez écrite en utilisant l’éditeur de programme de MUIbase. La fonction appellera et se positionnera sur les deux enregistrements avant de retourner la valeur que devrait refléter le tri des deux enregistrements. La fonction peut employer toutes les opérations pour comparer les enregistrements, ainsi elle peut par exemple comparer des enregistrements en utilisant des champs de référence. Pour plus d’information sur ce mécanisme, voir la voir [Section 15.29.7 \[Comparison function\]](#), page 148.

Si vous décidez d’employer une fonction de comparaison pour trier une table, veuillez noter que MUIbase ne peut pas toujours détecter lorsque les enregistrements de la table doivent être triés à nouveau puisque les dépendances sont inconnues.

Utilisez l’article du menu ‘Table – Réorganiser tous les enregistrements’ lorsque les enregistrements ne sont pas triés.

10.4 Changer le tri

Pour spécifier un tri, sélectionnez l’article de la table courante ‘Table – Changer le tri’. Ce qui ouvrira une fenêtre contenant

- Le nom de la table, dans le titre de la fenêtre.
- Le choix du champ ‘Type’ où vous indiquez si la table doit être triée en utilisant une ‘liste des champs’ ou en utilisant une ‘fonction de comparaison’. Selon le ‘Type’ choisi, vous pouvez saisir des données dans les articles suivants.

Pour un tri en utilisant une liste de champs, les articles suivants existent :

- Une liste de tous les champs de la table pouvant être employés dans la liste de tri. Cette liste est placée dans la partie gauche de la fenêtre. Si vous double-cliquez sur un nom alors le nom sera inséré dans la liste de tri (à la position actuelle du curseur).
- La liste courante des champs utilisés pour le tri. Cette liste est placée dans la partie droite de la fenêtre. L’article situé en haut de cette liste est le premier champ de la liste de tri. Vous pouvez réarranger les articles en les traînant sur d’autres positions de la liste. Pour ajouter d’autres champs, traînez-les de la liste de champ à la liste de tri. Vous pouvez enlever un champ de la liste de tri en traînant l’article hors de la liste de tri et en le laissant tomber sur la liste de champ.

Chaque entrée dans la liste de tri a un symbole avec une flèche se dirigeant vers le haut ou vers le bas du côté gauche. Vous pouvez changer l’état de la flèche en la double-cliquant. Si une flèche se dirige vers le haut le tri pour ce champ sera ascendant, si elle se dirige vers le bas, il sera descendant.

Pour un tri en utilisant une fonction de comparaison, il y a l'article suivant :

- Un champ 'Fonction de comparaison d'enregistrement' où vous pouvez entrer le nom de la fonction qui devraient être appelé pour comparer deux enregistrements de la table. Vous pouvez utiliser le bouton menu à la droite du champ 'Fonction de comparaison d'enregistrement' pour choisir un nom dans la liste de tous les noms de fonction. Si vous laissez le champ vide, il n'y aura pas de tri. Pour plus d'information sur la façon d'employer une fonction de comparaison, y compris les arguments qui lui sont passés, voir la voir [Section 15.29.7 \[Comparison function\]](#), page 148.

En outre, les boutons suivants existent :

- Un bouton 'Effacer' qui enlève tous les champs pour ne pas faire de tri.
- Deux boutons 'Ok' et 'Annuler' pour quitter la fenêtre.

Pour entrer un nouveau champ dans le classement de la liste, choisissez le 'Type' 'Liste des champs' et appuyez sur le bouton 'Effacer'. Utilisez alors le glisser/déposer comme décrit ci-dessus pour accumuler une nouvelle liste de champs. Si vous voulez avoir un tri vide, il suffit juste de ne pas ajouter de champs dans le tri de la liste.

Quand vous avez spécifié le tri, appuyez sur le bouton 'Ok'. MUIbase réorganisera alors à nouveau tous les enregistrements de la table.

10.5 Réordonner tous les enregistrements

Si vous pensez que certains enregistrements d'une table ne sont plus triés, par exemple en utilisant une fonction de comparaison pour le tri, alors vous pouvez choisir l'article du menu 'Table - Réordonner tous les enregistrements' pour trier tous les enregistrements.

11 Recherche

Pour parcourir les enregistrements, vous pouvez utiliser la boîte de recherche pour chercher un enregistrement particulier. La fonction de recherche utilise un motif de recherche (que vous devez fournir) et vérifie tous les enregistrements pour une correspondance avec ce motif. Lorsqu'elle en trouve, l'enregistrement est affiché dans la fiche de table.

11.1 Boîte de recherche

Le menu **'Table - Rechercher...'**, ouvre la boîte de recherche, qui contient les éléments suivants :

- un champ texte pour saisir le motif de recherche. Les caractères '*' et '?' peuvent être utilisés comme jokers. Le caractère '*' remplace n'importe quel nombre de n'importe quel caractère (y compris pas de caractère du tout), tandis que le caractère '?' remplace n'importe quel caractère mais exactement une fois.
- une option **'Sensible à la casse'** qui lorsqu'elle est cochée active la recherche de chaîne en différenciant les caractères majuscules des minuscules, sinon la recherche s'effectue sans discernement sur les majuscules et les minuscules.
- une option **'Tous les champs'**, qui lorsqu'elle est cochée, active la recherche de correspondances avec le motif spécifié dans tous les champs des enregistrements. Dans le cas contraire seul le champ actif au moment de l'ouverture de la boîte de recherche est testé. Dans le cas où l'objet actif au moment de l'ouverture de la boîte de recherche n'était pas un objet champ, celui-ci est vérifié et désactivé automatiquement.
- deux boutons radios pour la direction de la recherche, **'En avant'** et **'En Arrière'**.
- deux boutons radios pour déterminer à partir de quel enregistrement la recherche doit débuter, **'Premier/dernier enregistrement'** pour débuter la recherche du premier ou dernier enregistrement en fonction de la direction de la recherche, **'Enregistrement courant'** pour débuter la recherche à partir de l'enregistrement courant.
- deux boutons **'Rechercher'** et **'Annuler'** pour sortir de la fenêtre.

Une fois le motif de recherche saisi et la boîte de recherche fermée via le bouton **'Rechercher'**, MUIbase commence à rechercher un enregistrement correspondant. La comparaison d'un champ avec le motif de recherche est toujours réalisée en mode texte, les champs qui ne sont pas de type texte sont donc auparavant convertis en chaînes de caractères.

Si un enregistrement correspondant est trouvé, il est affiché en tant qu'enregistrement courant dans la fiche de table, dans le cas contraire, un message **motif non trouvé** est affiché.

Lorsque la recherche porte sur un champ qui est utilisé comme premier champ de tri et qu'elle ne commence pas par un joker ('*' or '?'), un algorithme de recherche amélioré (recherche dichotomique) est employé qui prend en compte l'ordre des enregistrements, ce qui augmente significativement la vitesse de recherche.

11.2 Rechercher en avant/en arrière

Deux autres menus permettent de rechercher l'occurrence suivante et précédente du motif de recherche. Sélectionnez le menu **'Table - Rechercher suivant'** pour naviguer vers

le prochain enregistrement correspondant au motif de recherche, et **‘Table - Rechercher précédent’** pour aller à l’enregistrement correspondant précédent.

11.3 Exemples de motif de recherche

voici quelques exemples de motif de recherche :

- **‘Lassie’** recherche les enregistrements ayant la chaîne **‘Lassie’** dans un des champs recherchés.
- **‘*x*’** recherche les enregistrements ayant la lettre **‘x’** dans un des champs recherchés.
- **‘????’** recherche les enregistrements ayant exactement quatre caractères dans un des champs recherchés, p. ex. un enregistrement avec une entrée **‘OVNI’**.

12 Importer et Exporter

Pour partager vos enregistrements avec d'autres systèmes de base de données, MUIbase offre un moyen d'importer et d'exporter les enregistrements depuis et vers d'autres bases de données. L'import et l'export sont réalisés par lecture et écriture de fichiers textes ASCII. De plus, les données que vous souhaitez importer doivent être dans un format particulier décrit dans la section suivante.

12.1 Format de fichier

Pour importer des enregistrements dans MUIbase, tous les enregistrements de la table doivent être disponibles dans un seul fichier texte. Si vous désirez importer des enregistrements de plusieurs tables, vous devez avoir plusieurs fichiers d'import (un par table).

Un fichier d'import est constitué de lignes et de colonnes. Les lignes sont séparées par un séparateur d'enregistrement, les colonnes par un séparateur de champs. Les délimiteurs peuvent être spécifiés dans les fenêtres d'import et d'export. Comme les champs des enregistrements eux-mêmes peuvent déjà contenir ces séparateurs, il est possible d'utiliser des guillemets autour des champs pour les protéger.

Un fichier d'import doit posséder la structure suivante.

- La première ligne contient le nom des champs, et pour chacun, un champ avec exactement le même nom doit exister dans la table dans laquelle sont importés les enregistrements. S'il existe un nom pour lequel il n'y a pas de champ correspondant dans la table, alors un message d'erreur est généré.
- Les lignes suivantes contiennent chacune un enregistrement. Comme tous les champs doivent être spécifiés au format texte, ils sont automatiquement convertis dans le type du champ de destination. Pour les champs de type booléen, le champ doit être soit NIL, soit TRUE (VRAI) (peu importe la casse), autrement un message d'erreur est généré. Pour les champs de type choix, le libellé (label) exact doit être spécifié (la casse est importante ici). Pour les champs de type référence, il faut indiquer le numéro de l'enregistrement référencé (à partir de 1). Pour tous les autres types, la valeur NIL est utilisée si le champ ne peut être converti vers le type requis.
- Lors de l'utilisation des guillemets, tous les champs des enregistrements, ainsi que ceux de la ligne contenant le nom des champs, doivent être protégés par des guillemets.

12.2 Exemple de fichier d'import

L'exemple suivant importe un fichier en utilisant `\n` et `\t` respectivement comme délimiteur d'enregistrement et délimiteur de champ, et des guillemets autour de tous les champs. Le fichier peut être importé dans une table ayant les champs suivants :

- Nom (texte)
- NbEnfants (entier)
- Femme (booléen)
- Travail (choix)
- Notes (mémo)

```
"Nom" "NbEnfants" "Femme" "Travail" "Notes"
"Janet Jackson" "???" "TRUE" "Musicienne" "Dernier CD: The velvet rope"
"Bernt Schiele" "???" "NIL" "Scientifique" "Centres de recherches :
Robotique, Autonomie et Vision par ordinateur"
"Gerhard" "0" "NIL" "Mécanique de précision" ""
```

12.3 Importer des enregistrements

Pour importer des enregistrements dans la table active, sélectionnez le menu ‘**Table – Importer des enregistrements...**’. Cela ouvre une fenêtre contenant les éléments suivants :

- un champ texte pour saisir le nom du fichier d’import. À droite de ce champ il y a trois boutons : le premier permet d’ouvrir une requête de sélection de fichier, le second lance la visionneuse externe pour inspecter le fichier saisi, et le troisième lance un éditeur pour éditer le contenu du fichier.
- deux champs texte pour saisir respectivement les délimiteurs d’enregistrement et de champ. Vous pouvez saisir un caractère ou un code de contrôle tel que `\n`, `\t`, `\f`, `\???` (code octal) ou `\x??` (code hexa). Les délimiteurs doivent être des caractères ASCII 7 bits.
- une case ‘**Guillemets**’ qui peut être cochée pour indiquer que les champs sont protégés par des guillemets.
- deux boutons ‘**Importer**’ et ‘**Annuler**’ pour fermer la fenêtre.

Si vous appuyez sur le bouton ‘**Importer**’, MUIbase va charger le fichier spécifié et importer tous les enregistrements qu’il va trouver. Si tout se passe correctement après le processus d’import, MUIbase demande si vous voulez vraiment ajouter les enregistrements importés dans la table. A ce point il est encore possible d’annuler l’opération.

Si une erreur survient lors de la lecture du fichier d’import, un message d’erreur est affiché.

Si vous avez besoin d’un mécanisme d’import plus sophistiqué, il est recommandé d’écrire sa propre routine d’import dans un programme MUIbase.

12.4 Exporter des enregistrements

Pour exporter les enregistrements de la table active, sélectionnez le menu ‘**Table – Exporter des enregistrements...**’. Cela ouvre une fenêtre contenant les éléments suivants :

- un champ texte pour saisir le nom du fichier d’export avec à droite un bouton pour ouvrir une requête de sélection de fichier.
- deux champs texte pour saisir respectivement les délimiteurs d’enregistrement et de champ. Vous pouvez saisir un caractère ou un code de contrôle tel que `\n`, `\t`, `\f`, `\???` (code octal) ou `\x??` (code hexa). Les délimiteurs doivent être des caractères ASCII 7bit.
- une case ‘**Guillemets**’ qui peut être cochée pour indiquer que les champs doivent être protégés par des guillemets.

- deux boutons ‘**Exporter**’ et ‘**Annuler**’ pour fermer la fenêtre.

Si vous appuyez sur le bouton ‘**Importer**’, MUIbase va ouvrir le fichier spécifié et y écrire les enregistrements en y incluant une ligne d’entête contenant le nom des champs. La fonction d’export écrit toujours tous les champs de la table dans le fichier d’export.

Pour un mécanisme d’export personnalisé, vous pouvez utiliser l’éditeur de requête de MUIbase (voir [Chapitre 13 \[Data retrieval\]](#), [page 55](#)) ou écrire votre propre routine d’export dans un programme MUIbase.

13 Traitement des données

Deux méthodes sont utilisables dans MUIbase pour le traitement des données : par programmation ou l'éditeur de requêtes.

La méthode par programmation vous permet de créer des boutons dans les masques de tables qui, sur pression, appelleront des fonctions programmées. L'utilisation de cette méthode est décrite dans le chapitre concernant l'éditeur de structure (voir [Chapitre 14 \[Structure editor\]](#), page 60) et dans le chapitre à propos de la programmation dans MUIbase (voir [Chapitre 15 \[Programming MUIbase\]](#), page 75).

Ce chapitre décrit l'utilisation de l'éditeur de requêtes, une fenêtre où vous pouvez entrer des requêtes et consulter le résultat dans une fenêtre défilante.

13.1 Requêtes Select-from-where

MUIbase utilise un système de requête du type select-from-where comme dans les bases de données SQL. La requête vous permet d'afficher le contenu des enregistrements d'une ou plusieurs tables. Seuls les enregistrements qui correspondent à un certain critère se retrouvent dans le résultat. La syntaxe (incomplète) d'une requête select-from-where est

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

où *exprlist* est une liste des expressions, séparées par des virgules, à afficher (souvent les noms d'attributs) ou une simple étoile * qui correspond à tous les attributs des tables spécifiées, *tablelist* est une liste de nom de tables, séparées par des virgules, desquelles les enregistrements sont examinés, *test-expr* est l'expression qui est testée pour chaque groupe d'enregistrements qui doit être inclu dans le résultat, et *orderlist* est une liste d'attributs, séparés par des virgules, qui déterminent l'ordre pour afficher le résultat. Notez que les champs WHERE et ORDER BY sont optionnels ; ce qui est indiqué par les crochets [].

Par exemple, la requête

```
SELECT * FROM table
```

affiche le contenu des attributs de tous les enregistrements de la table spécifiée.

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

affiche les valeurs du champ *attr1* dans tous les enregistrements de *table* dans laquelle le contenu du champ *attr2* contient le mot 'Madonna'.

Pour plus d'informations à propos des requêtes select-from-where et notamment sa syntaxe complète, voir [Chapitre 15 \[Programming MUIbase\]](#), page 75, pour plus d'exemples voir [Section 13.4 \[Query examples\]](#), page 58.

13.2 Éditeur de requêtes

Pour saisir et exécuter des requêtes, ouvrez l'éditeur de requêtes en sélectionnant le choix de menu 'Program - Queries'. L'éditeur de requêtes peut gérer plusieurs requêtes mais une seule est affichée. La fenêtre de l'éditeur de requêtes contient les éléments suivants :

- un champ de saisie de texte avec un bouton attaché. Le champ texte indique le nom de la requête en cours. En pressant le bouton, apparaissent une liste avec d'autres noms de requêtes ainsi que d'autres boutons. Vous pouvez sélectionner une des requêtes

affichées pour la rendre en faire la requête en cours, pressez le bouton ‘Nouveau’ pour créer une nouvelle requête, pressez le bouton ‘Dupliquer’ pour faire une copie de la requête sélectionnée, cliquez sur le bouton ‘Trier’ pour trier la liste des requêtes, ou pressez le bouton ‘Delete’ pour effacer la requête en cours. Pour quitter la fenêtre sans rien modifier, cliquez de nouveau sur le bouton des requetes.

- un bouton ‘Run’ qui compile et exécute le programme de requête et affiche le résultat dans le fenêtre déroulante.
- un bouton ‘Imprimer’ qui ouvre une requête (voir [Section 13.3 \[Printing queries\]](#), [page 56](#)) pour imprimer les résultats.
- deux boutons ‘Load’ et ‘Save’ pour charger et enregistrer le programme de requête courant. Si vous enregistrez une requête, le texte du programme est écrit au format ASCII et la première ligne contient le nom de la requête. La fonction de chargement utilise le même format au chargement d’un fichier. Il est aussi possible de charger et enregistrer des requêtes en utilisant les choix de menu dans le menu contextuel du champ de saisie. Cependant, ces éléments de menu enregistrent seulement le texte du programme et n’incluent pas le nom d’une requête.
- un champ de saisie pour taper le programme de requête. Vous saisissez ici une requête de type select-from-where. Cependant, it is also possible to enter any expression of MUIbase’ programming language. This can be useful if you want to do simple computations or update some fields of a table by using a simple program. Veuillez noter que MUIbase entoure automatiquement votre expression avec une paire de parenthèses. Vous pouvez donc ne pas utiliser les deux les plus éloignées.
- un affichage en liste qui montre les résultats de la requête en cour. L’affichage est formaté en lignes et colonnes. La ligne de titre contient les noms des champs de la requête select-from-where (le plus souvent les noms d’attribut). Sur Les lignes suivantes est affiché le résultat de la requête, un jeu d’enregistrement par ligne. Chaque valeur est affichée dans sa propre colonne. Si vous double-cliquez un élément de la liste, et que cet élément était généré directement à partir d’un enregistrement, alors cet enregistrement est affiché dans le masque de table correspondant. C’est une façon simple de d’aller à un certain enregistrement dans son masque de table.

L’éditeur de requête est non-modale. Cela signifie que vous pouvez laisser l’éditeur de requête ouvert alors que vous travaillez avec le reste de l’application. Vous pouvez à tout moment fermer l’éditeur de requête, en cliquant sur le bouton de fermeture sur la barre de titre.

13.3 Impression de requêtes

Après avoir exécuté une requête, vous pouvez imprimer le résultat dans un fichier ou sur une imprimante en cliquant sur le bouton ‘Imprimer’ dans l’éditeur de requêtes. Cela ouvre une requête d’impression qui contient les éléments suivants:

- un champ ‘Délimiteur’ où vous précisez comment les colonnes doivent être séparées. ‘Espaces’ place des caractères espaces entre chaque champs. La justification est faite à gauche ou à droite en fonction du type du champ (les nombres sont justifiés à gauche, le texte à droite). ‘Tabulations’ insère un caractère tabulation entre les colonnes. Cela peut être utile si vous voulez utiliser la fenêtre d’impression pour exporter des enreg-

istements (voir ci-dessous). ‘Personnaliser’ vous permet de préciser un délimiteur personnalisé à afficher entre les champs.

- un champ ‘Police’ où vous précisez quelle police de caractère doit être utilisée pour imprimer l’affichage. ‘NLQ’ signifie qualité proche d’une lettre qui doit imprimer dans une meilleure qualité que ‘Brouillon’.
- un champ ‘Taille’ où vous définissez la taille des caractères. ‘Pica’ imprime avec une police large (10 cpi), ‘Elite’ avec une police de taille moyenne (12 cpi) et ‘Condensé’ avec une petite police (17 cpi).
- un champ texte ‘Séquence d’initialisation’ qui vous permet de préciser votre imprimante. Le contenu de ce champ est directement envoyé à l’imprimante à l’ouverture de celle-ci. Par exemple, vous pouvez utiliser ‘\33c’ comme séquence d’initialisation qui réinitialise votre imprimante.
- un champ ‘Indentation’ où vous pouvez préciser le nombre d’espaces qui seront utilisés pour indenter chaque ligne en sortie.
- un champ ‘Entête’ qui, s’il est coché, imprime les noms de champs sur la première ligne.
- un champ ‘Codes d’échappement’. S’il n’est pas coché l’impression de tous les codes d’échappement est supprimé, donc les paramétrages des champs ‘Police’ et ‘Taille’ sont ignorés et le contenu de la ‘Séquence d’initialisation’ ne sont pas imprimés. Supprimer l’impression de tous les codes d’échappement est utile si vous voulez générer un fichier ASCII, par exemple pour exporter des enregistrements.
- un champ ‘Guillemets’ qui, s’il est coché, entoure chaque champ par des guillemets.
- un champ ‘Après impression’ où vous choisissez comment doit se terminer l’extraction. ‘Saut de page’ imprime un caractère saut de page \f. ‘Saut de ligne’ imprime un nombre de sauts de ligne \n. Le nombre de sauts de ligne peut être saisi dans le champ texte à la droite du bouton ‘Sauts de ligne’. ‘Rien’ n’imprime rien sur l’imprimante.
- un champ texte ‘Sortie’ avec un bouton attaché. Vous pouvez utiliser le bouton pour ouvrir une requête de fichier pour choisir un nom de fichier ou saisissez un nom de fichier dans le champ texte. Pour envoyer la sortie vers l’imprimante entrez ‘|lpr’ (Linux) ou ‘PRT:’ (Amiga). Pour d’autres noms de fichiers spéciaux, voir [Section 7.5.9 \[Program output file\], page 39](#).
- deux boutons ‘Ok’ et ‘Annuler’ pour fermer la fenêtre d’impression.

Quand vous en avez terminé avec tous les paramètres, cliquez sur le bouton ‘Ok’ pour lancer le travail d’impression.

La fenêtre d’impression peut aussi être utilisée pour exporter des enregistrements dans un fichier ASCII. Pour cela, précisez ‘Tabulations’ (ou ‘Personnalisé’) dans le champ ‘Délimiteur’, mettez à 0 le nombre d’espaces dans le champ ‘Indentation’, cochez ‘Entête’, décochez ‘Codes d’échappement’ pour supprimer les paramètres de police, taille et séquence d’initialisation, éventuellement cochez ‘Guillemets’ si vous voulez que les champs soient entourés de guillemets, cochez ‘Rien’ dans le champ ‘Après impression’, et entrez un nom de fichier dans le champ ‘Sortie’. Il se peut qu’utiliser l’éditeur de requête avec la fenêtre d’impression soit plus puissant pour exporter des enregistrements que la fonction import/export de MUIbase (voir [Chapitre 12 \[Import and Export\], page 52](#)). C’est dû au fait que n’importe quelle requête peut être utilisée dans l’éditeur alors que la fenêtre d’export utilise seulement une requête fixe.

13.4 Exemples de requêtes

Voici plusieurs exemples de requêtes pour vous donner une idée de la puissance des requêtes select-from-where.

Supposez que nous ayons deux tables 'Personne' et 'Chien'. 'Personne' a les attributs suivants : 'Nom', un attribut entier 'Age', et deux attributs de référence 'Père' et 'Mère' qui se réfèrent aux enregistrements père et mère de la table 'Personne'. La table contient les enregistrements suivants :

	Nom	Age	Père	Mère
p1:	Steffen	26	p2	p3
p2:	Dieter	58	NIL	NIL
p3:	Marlies	56	NIL	NIL
p4:	Henning	57	NIL	NIL

'Chien' a un attribut texte 'Nom', un attribut de type choix 'Couleur' et un attribut de référence 'Propriétaire' qui se réfère au propriétaire dans la table 'Personne'. La table contient les enregistrements suivants :

	Nom	Couleur	Propriétaire
d1:	Boy	blanc	p3
d2:	Streuner	gris	NIL

Avec ces données, on peut exécuter les requêtes select-from-where suivantes :

```
SELECT * FROM Personne
```

donne :

Name	Age	Père	Mère
Steffen	26	Dieter	Marlies
Dieter	58		
Marlies	56		
Henning	57		

(Pour les attributs de référence le champ 'Nom' de l'enregistrement référencé est affiché.)

```
SELECT Nom "Enfant", Age,
       Père.Nom "Père", Père.Age "Age",
       Mère.Nom "Mère", Mère.Age "Age"
FROM Personne WHERE (AND Père Mère)
```

donne :

Enfant	Age	Père	Age	Mère	Age
Steffen	26	Dieter	58	Marlies	56


```
SELECT Nom, Couleur,
       (IF Propriétaire Propriétaire.Nom "Pas de propriétaire") "Propriétaire"
FROM Chien
```

donne :

Nom	Couleur	Propriétaire
Boy	blanc	Marlies
Streuner	gris	Pas de propriétaire

```
SELECT a.Nom, a.Age, b.Nom, b.Age FROM Personne a, Personne b
WHERE (> a.Age b.Age)
```

donne :

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

14 Editeur de structure

MUIbase dispose de deux modes de fonctionnement différents : le mode édition d'enregistrement où il est possible de saisir et parcourir les enregistrements, et le mode édition de structure où se définit la structure de la base, c'est à dire les tables, les champs et l'apparence d'un projet. Ce chapitre décrit l'éditeur de structure et explique comment gérer la structure d'un projet.

Pour basculer d'un mode édition d'enregistrement à celui d'édition de structure, il faut sélectionner l'élément '**Éditeur de structure**' dans le menu '**Projet**', ce qui ferme toutes les fenêtres et ouvre la fenêtre de l'éditeur de structure. Pour revenir au mode édition d'enregistrement, il faut sélectionner le menu '**Projet - Quitter l'éditeur de structure**' ou simplement fermer l'éditeur de structure en cliquant sur le gadget de fermeture dans la barre de titre de la fenêtre.

La fenêtre de l'éditeur de structure est divisée en trois parties : la partie supérieure gauche composée du groupe '**Tables**' servant à gérer les tables du projet, la partie inférieure gauche quant à elle occupée par le groupe '**Champs**' permet de gérer les champs d'une table, et pour finir la partie droite est occupée par un groupe '**Affichage**' pour gérer les éléments graphiques du projet (interface).

14.1 Table Management

Depuis le groupe '**Tables**' de l'éditeur de structure, vous pouvez créer, changer, supprimer et trier les tables.

14.1.1 Création de tables

Pour créer une nouvelle table, appuyez sur le bouton '**Nouveau**' du groupe '**Tables**' ce qui ouvrira la fenêtre '**Nouvelle table**' contenant :

- un champ de texte pour nommer la table. Chaque table doit avoir un nom unique commençant par une lettre majuscule suivie d'autres lettres, de chiffres ou de tirets bas. Les caractères non-ASCII (tréma, double point) ne sont pas autorisés. En revanche, il est tout à fait possible d'utiliser des caractères non-ASCII dans l'interface utilisateur de la table.
- un groupe '**Nombre d'enregistrements**' dans lequel vous définissez combien d'enregistrements comportera la table. '**Illimité**' signifie que la table peut contenir un nombre infini d'enregistrements, '**Exactement un**' signifie que la table ne peut comporter qu'un seul enregistrement. Ce champ est utile pour contrôler le projet (voir [Section 5.2 \[Tables\], page 19](#)).
- un groupe '**Déclencheurs**' dans lequel vous pouvez définir le nom de deux fonctions. Dans le champ '**Nouveau**' vous entrez le nom de la fonction à appeler à chaque fois que l'on veut créer un nouvel enregistrement, alors que le champ '**Suppression**' contient le nom de la fonction à appeler à chaque suppression d'enregistrement. Vous pouvez également utiliser les boutons pop-up situés à droite des champs de texte pour choisir à partir d'une liste de toutes les fonctions celle à appeler. Si vous laissez ces champs vides, alors ce sont les actions par défaut qui seront exécutées (les enregistrements sont créés automatiquement et les enregistrements sont effacés après une éventuelle requête de confirmation). Pour de plus amples informations concernant l'utilisation de

ces déclencheurs, ainsi que le passage d'arguments, voir [Section 15.29.5 \[New trigger\]](#), page 147 et [Section 15.29.6 \[Delete trigger\]](#), page 148.

- deux boutons 'Ok' et 'Annuler' pour fermer la fenêtre.

Une fois que vous avez effectué tous les réglages, pressez le bouton 'Ok' pour créer la nouvelle table. Si vous avez fait une erreur comme entrer un nom invalide, un message s'affichera et vous donnera des informations sur l'erreur. A l'inverse si tout se passe correctement, la fenêtre 'Nouvelle table' se fermera et la nouvelle table s'affichera dans la liste des tables de l'éditeur de structure.

14.1.2 Modification de tables

Après avoir créé une table, vous pouvez toujours la modifier. Il vous suffit de double cliquer sur le nom de la table et la fenêtre 'Modifier la table' apparaîtra. Cette fenêtre est similaire à celle de création de table (voir [Section 14.1.1 \[Creating tables\]](#), page 60) et vous permet de modifier les champs en saisissant une nouvelle valeur.

Quand vous avez effectué toutes vos modifications, appuyez sur le bouton 'Ok' pour fermer la fenêtre.

Vous ne pouvez changer le nombre d'enregistrements 'Illimité' en 'Exactement un' si la table contient déjà plus d'un enregistrement.

14.1.3 Suppression de tables

Pour supprimer une table, sélectionnez son nom dans la liste de l'éditeur de structure, puis cliquez sur le bouton 'Supprimer' sous la liste. Avant de supprimer définitivement la table, une requête de confirmation sera ouverte. Si vous confirmez en appuyant sur le bouton 'Supprimer', la table sera irrémédiablement supprimée.

Un problème se pose si la table est utilisée ailleurs dans un projet. Dans ce cas, la table ne peut pas être supprimée simplement, mais toutes les références à cette table doivent être éliminées du programme. Si la table à supprimer est utilisée dans un projet, l'éditeur apparaîtra et affichera la première référence de la table. Vous devez alors modifier le programme de manière à éliminer toute référence à la table. Après avoir éliminé une référence, vous pouvez aller à la prochaine en appuyant sur le bouton 'Compiler'. Vous pouvez à tout moment annuler l'opération en appuyant sur le bouton 'Rétablir' et fermer l'éditeur.

14.1.4 Tri des tables

Vous pouvez trier les tables dans le groupe 'Tables' de l'éditeur de structure de plusieurs façons. Vous pouvez les arranger manuellement par glisser/déposer ou vous pouvez utiliser le bouton 'Tri' sous la liste pour les classer par ordre alphabétique ('Tri 1') ou bien encore en fonction de l'ordre de la liste d'affichage ('Tri 2').

14.2 Gestion des champs

Avec le groupe 'Champs' de l'éditeur de structure, vous pouvez créer, copier, modifier, effacer et trier les champs de la table sélectionnée dans 'Tables'.

14.2.1 Création de champs

Pour créer un nouveau champ dans la table active, cliquez sur le bouton ‘Nouveau’ dans le groupe ‘Champs’. Cette action ouvrira la fenêtre ‘Nouveau champ’ qui contient les éléments suivants :

- un champ de texte pour entrer le nom du champ. Chaque champ d’une table doit avoir un nom unique et commencer par une lettre majuscule suivie d’autres lettres ou chiffres ou tiret bas. Les caractères non-ASCII (les trémas par exemple) ne sont pas autorisés. Cependant il est toujours possible d’utiliser pour les champs (dans l’interface utilisateur) n’importe quel caractère y compris non-ASCII.
- une liste ‘Type’ dans laquelle vous précisez le type de champ. Pour plus de précision sur les types de champ, consultez voir [Section 5.5 \[Attribute types\]](#), page 20.
- une section sous le champ ‘Type’ pour préciser les réglages spécifiques au type de champ choisi. Pour plus de précisions sur cette section, consultez voir [Section 14.2.2 \[Type specific settings\]](#), page 62.
- un champ ‘Déclencheur’ dans lequel vous donnez le nom d’une fonction qui sera appelée chaque fois que l’utilisateur voudra modifier le contenu du champ dans un enregistrement. Vous pouvez utiliser le bouton contextuel (ou bouton pop-up) à droite du champ de texte pour sélectionner un nom parmi une liste de noms de fonction. Si vous laissez ce champ vide, alors l’action par défaut sera exécutée, ce qui veut dire que la valeur entrée sera simplement stockée dans le champ. Pour plus de précision sur le déclenchement de fonction, y compris comment passer des arguments, consultez voir [Section 15.29.8 \[Attribute trigger\]](#), page 149.
- une case à cocher ‘Tenir compte des modifications’. Si elle est cochée, toute modification dans un champ d’un enregistrement, sera considérée comme une modification du projet. Décochez cette case si vous voulez ignorer les modifications de champ.
- deux boutons ‘Ok’ et ‘Annuler’ pour fermer la fenêtre.

Lorsque vous avez effectué vos réglages, appuyez sur le bouton ‘Ok’ pour créer le nouveau champ. Si vous avez fait une erreur comme entrer un nom invalide, un message s’affichera et vous donnera des informations sur l’erreur. A l’inverse si tout se passe correctement, la fenêtre ‘Nouveau champ’ se fermera et le nouveau champ s’affichera dans la liste des champs de l’éditeur de structure.

14.2.2 Type Specific Settings

Voici les modifications spécifiques aux types de champ que l’on peut effectuer:

- Les réglages spécifiques aux champs de type String sont les suivants
 - un champ de type Integer ‘Max length’ pour définir la longueur maximale pour ce champ.
 - un champ de type String ‘Valeur initiale’ pour définir la valeur par défaut pour ce champ. On peut entrer n’importe quel texte dans la limite de la longueur maximale.
- Les réglages spécifiques aux champs de type Integer sont les suivants
 - un champ ‘Valeur initiale’ dans lequel vous définissez la valeur par défaut pour ce champ. Vous avez le choix entre ‘NIL’ et ‘other’. Si vous utilisez ‘other’ vous devez spécifier une valeur initiale dans le champ de texte à droite.

- un champ de type String `'NIL string'` dans lequel vous définissez le texte qui sera affiché lorsque le champ contiendra la valeur NIL.
- Le réglage spécifique aux champs de type Bool comprend un champ `'Initial value'` dans lequel vous entrez la valeur par défaut, soit `'NIL'` ou soit `'TRUE'`.
- Les réglages spécifiques aux champs de type Choice sont les suivants
 - un bouton `'Edit labels'` pour ouvrir la fenêtre `'Edit labels'` et donnez un nom au champ de type Choice (voir [Section 14.2.3 \[Label editor\]](#), page 63).
 - un champ `'Valeur initiale'` dans lequel vous entrez la valeur par défaut.
- Les réglages spécifiques aux champs de type Reference sont les suivants
 - une liste de toutes les tables pour sélectionner celle vers laquelle pointer le champ. En cliquant sur la table, le champ ira vers la référence.
 - un champ `'Auto show'`. S'il est coché, la table de référence sera mise à jour automatiquement en fonction de l'enregistrement de référence, chaque fois que l'utilisateur passera à un autre enregistrement.
 - un champ `'Filter'`. S'il est coché, le filtre de référence de ce champ sera activé. Pour plus de précision, consultez Voir [Section 9.2 \[Reference filter\]](#), page 46.

Les champs de type Reference ont toujours NIL comme valeur par défaut.

- Le réglage spécifique aux champs de type Virtual concerne le champ de texte `'Compute'` dans lequel vous donnez le nom de la fonction à lancer pour calculer la valeur du champ. Vous pouvez vous servir du bouton pop-up pour sélectionner un nom parmi une liste de tous les noms de fonctions. Veuillez consulter voir [Section 15.29.9 \[Programming virtual attributes\]](#), page 150 pour en savoir plus sur l'utilisation de cette fonction.
- Les champs de type Memo et button n'ont pas de réglages propres. La valeur par défaut des champs de type Memo est une ligne de texte vierge.

14.2.3 Label Editor

L'éditeur d'étiquettes intervient à chaque fois que vous devez définir une liste comme par exemple une liste de noms pour un champ de type Choice. L'éditeur se compose d'une fenêtre qui contient :

- une liste montrant les étiquettes existantes. Cliquer sur une étiquette la sélectionne et elle sera visible dans le champ de texte sous la liste. Vous pouvez utiliser le glisser/déposer pour réarranger les étiquettes.
- un champ de texte `'Label'` qui montre l'étiquette sélectionnée et permet également de la modifier. Les modifications ne seront prises en compte que lorsque vous appuyez sur la touche **Enter**. S'il n'y a aucune étiquette de sélectionner, la pression sur **Enter** ajoutera une nouvelle étiquette à la liste.
- un bouton `'New'` qui désélectionne l'étiquette courante, ce qui permet d'ajouter de nouvelles entrées dans le champ de texte `'Label'`.
- un bouton `'Remove'` qui efface l'étiquette sélectionnée de la liste.
- un bouton `'Sort'` pour trier par ordre alphabétique la liste d'étiquettes.
- deux boutons `'Ok'` et `'Cancel'` pour quitter l'éditeur d'étiquettes.

Après avoir ajouté toutes les étiquettes ou les avoir changées, pressez le bouton `'Ok'` pour fermer la fenêtre.

14.2.4 Copying Attributes

Dans le cas où vous nécessitez des champs similaires, il vous est possible de copier un champ. Pour cela, sélectionnez le champ à dupliquer et cliquez sur le bouton ‘Copy’ sous la liste de champ. Cette action ouvre une requête ‘Copy attribute’ montrant les options du champ sélectionné. Modifiez certaines parties comme le nom puis cliquez sur ‘Ok’ pour générer une copie du champ.

14.2.5 Changing Attributes

Après la création d’un nouveau champ, il est toujours encore possible de modifier ses réglages. Pour cela, double-cliquez sur le nom du champ ce qui laissera apparaître la requête ‘Change attribute’. Cette requête est semblable à celle lors de la création d’un champ (voir [Section 14.2.1 \[Creating attributes\], page 62](#)) et permet de modifier certains paramètres. Les paramètres ne pouvant être modifiés, comme le type de champ, apparaissent grisés.

Les choses à prendre en compte lors d’un changement de champ sont les suivantes :

- Le type de champ ne peut être changé. Si vous devez changer le type d’un champ, il est préférable d’en créer un nouveau du type voulu et de copier les enregistrements du champ à remplacer vers le nouveau champ en entrant un programme MUIbase simple dans l’éditeur de requête (voir [Section 13.2 \[Query editor\], page 55](#)).
- Si vous modifiez la valeur par défaut d’un champ, alors uniquement les nouveaux enregistrements auront cette nouvelle valeur pour initialiser l’enregistrement.
- Certaines précautions sont à prendre pour les champs de type Choice lors de changement d’étiquettes. Les étiquettes sont uniquement utilisées pour visualiser le contenu du champ, mais en interne, ce sont des numéros qui sont stockés et utilisés comme index dans la liste d’étiquettes. Donc si vous modifiez l’ordre des étiquettes, vous ne modifiez pas le numéro en interne mais sa visualisation sous forme d’étiquettes ! Par conséquent, vous ne devriez pas changer l’ordre des étiquettes après avoir créé un champ de type Choice. Par contre l’ajout de nouvelles étiquettes à la fin de la liste ne pose pas ce genre de problème. Pour une plus grande flexibilité avec possibilité de changer l’ordre des étiquettes, utilisez plutôt un champ de type String associé à ‘List-view pop-up’ (voir [Section 14.3.3 \[Attribute object editor\], page 67](#)).
- La table de référence d’un champ référence ne peut pas être modifiée

Si vous en avez terminé avec les modifications, pressez le bouton ‘Ok’ pour fermer la requête.

14.2.6 Deleting Attributes

Pour effacer un champ, cliquez sur son nom dans la liste des champs de l’éditeur de structure et appuyez sur le bouton ‘Del’ situé sous la liste. Une requête de confirmation apparaîtra et si vous confirmez en appuyant à nouveau sur ‘Delete’, le champ sera définitivement effacé.

Un problème peut se poser lorsque le champ est utilisé ailleurs dans le projet. Dans ce cas, le champ ne peut être simplement effacé mais toutes les références à celui-ci doivent impérativement disparaître du programme. Si le champ à effacer est utilisé ailleurs, l’éditeur de programme apparaîtra et affichera la première occurrence de ce champ. Vous devrez alors modifier le programme de manière à ce qu’il ne contienne plus aucune référence à ce champ.

Après chaque suppression d'une référence, vous passez à la suivante en appuyant sur le bouton **'Compile'**. A tout moment vous pouvez annuler toute l'opération en appuyant sur **'Revert'** et en refermant l'éditeur de programmes.

14.2.7 Sorting Attributes

Le classement des champs dans la partie **'Attributes'** de l'éditeur de structure peut se faire de plusieurs façons. Vous pouvez le faire à la main par glisser/déposer, ou utiliser le bouton **'Sort'** sous la liste qui permet un tri par ordre alphabétique (**'Sort1'**), ou encore un tri suivant l'ordre de l'affichage de la liste (**'Sort2'**).

14.3 Gestion de l'affichage

Dans la partie **'Display'** de l'éditeur de structure, vous définissez l'agencement des éléments de la base de données dans l'interface utilisateur. Cette section comporte une partie choix, une partie liste et plusieurs boutons.

14.3.1 Champ d'affichage

Le champ d'affichage contient les éléments suivants :

- un élément de sélection avec deux réglages, **'Table mask'** et **'Main window'**. Dans **'Table mask'** vous définissez l'agencement des champs de la table active dans l'interface utilisateur. Dans **'Main window'** vous définissez l'arrangement des tables.
- une liste montrant la caractéristique actuelle de l'interface utilisateur. La liste se présente sous forme d'arborescence. Les éléments ayant une flèche à leur gauche sont des objets GUI composés pouvant être ouverts ou fermés en (double-) cliquant sur le symbole de la flèche. Un double clic sur l'élément lui-même ouvre une fenêtre permettant l'édition de ses réglages. Tous les objets GUI issus du même objet parent sont placés de manière identique (horizontalement ou verticalement). La représentation est déterminée par l'objet GUI parent : les éléments des tableaux, panels et fenêtres sont placés verticalement alors que les éléments des groupes sont placés en fonction des réglages définis par l'éditeur de groupe (voir [Section 14.3.8 \[Group editor\]](#), page 72).
- un bouton **'+' ('Add')** pour ajouter la table sélectionnée ou le champ sélectionné (suivant ce qui est affiché dans la partie choix) à la liste. En général, les tables et les champs sont ajoutés à la liste automatiquement après leur création.
- un bouton **'-' ('Rem')** pour retirer de la liste ce qui est sélectionné. Si vous retirez une table, la totalité du formulaire de la table sera enlevé de l'interface utilisateur. De ce fait, vous ne pourrez voir la table dans l'interface projet et pourrez cacher des tables entières. Idem pour les champs : retirez un champ équivaut à le cacher de l'interface projet.
- deux boutons **'Up'** et **'Down'** pour déplacer l'élément sélectionné vers le haut ou vers la bas dans la liste.
- deux boutons **'In'** et **'Out'** pour déplacer l'élément sélectionné vers un niveau hiérarchique plus élevé ou plus bas dans la liste.
- un bouton **'Panel'** pour ajouter un panel à la table. Voir [Section 14.3.2 \[Panel editor\]](#), page 66, pour plus de précisions concernant la mise en place d'un panel.
- un bouton **'Text'** pour ajouter un objet texte dans la liste (voir [Section 14.3.5 \[Text editor\]](#), page 72).

- un bouton ‘Image’ pour ajouter un objet image (voir [Section 14.3.6 \[Image editor\]](#), page 72).
- un bouton ‘Space’ pour insérer un espace entre les autres objets (voir [Section 14.3.7 \[Space editor\]](#), page 72).
- un bouton ‘Balance’ pour ajouter un objet balance dans la liste. Cet objet est très utile pour contrôler la taille des autres objets graphiques.
- un bouton ‘Group’ pour ajouter un objet groupe dans la liste. Avant d’appuyer sur ce bouton, vous pouvez sélectionner plusieurs éléments dans la liste pour les déplacer dans un nouveau groupe. Voir [Section 14.3.8 \[Group editor\]](#), page 72, pour plus de précisions concernant la mise en place d’un groupe.
- un bouton ‘Register’ pour ajouter un groupe enregistré dans la liste. Comme pour l’objet groupe, il est possible de sélectionner plusieurs objets graphiques afin de les déplacer vers le nouveau groupe enregistré. Pour plus de précisions concernant la mise en place d’un groupe enregistré, consultez voir [Section 14.3.9 \[Register group editor\]](#), page 73.
- un bouton ‘Window’ pour ajouter une nouvelle fenêtre dans la liste. Comme précédemment, il est possible de sélectionner plusieurs objets graphiques pour les déplacer dans la nouvelle fenêtre. Pour plus de précisions concernant la mise en place d’une fenêtre, consultez voir [Section 14.3.10 \[Window editor\]](#), page 73.

Pour plus de précisions concernant les éléments graphiques ainsi que la façon de les utiliser, consultez voir [Section 5.9 \[User interface\]](#), page 27.

14.3.2 Editeur de panel

En ajoutant un panel au formulaire d’une table ou en double cliquant sur un panel existant, une fenêtre ‘Panel’ apparaîtra avec les éléments suivants :

- un champ de texte ‘Title’ pour donner un titre à l’entête du panel.
- un champ de texte ‘Font’ avec un bouton pop-up pour choisir la police du titre. Si vous laissez le champ vide, la police par défaut sera utilisée.
- un champ ‘Background’ avec une case à cocher ‘Default’ pour définir l’image de fond de l’entête du panel. Si le champ ‘Default’ est coché, alors l’image de fond par défaut sera utilisée. Sinon vous pouvez cliquer sur ‘Background’ pour ouvrir une requête et en sélectionner une.
- un champ ‘Num/All’. S’il est coché, le numéro de l’enregistrement en cours et le nombre total d’enregistrements apparaîtront dans la partie droite de l’entête du panel.
- un champ ‘Filter’ qui, si coché, ajoute un bouton de filtrage à l’entête du panel et permet d’activer ou non le filtrage d’enregistrement de la table. S’il n’est pas coché, l’élément du menu ‘Table – Change filter’ sera aussi désactivé pour cette table et vous ne pourrez plus ajouter de filtre pour la table. Pour plus d’informations sur le filtrage des enregistrements, consultez voir [Section 9.1 \[Record filter\]](#), page 45.
- un champ ‘Arrows’ pour ajouter deux boutons flèches au masque de la table. Ces flèches permettent la navigation entre les enregistrements de la table. S’il n’est pas coché, vous ne pourrez pas naviguer dans les enregistrements et tous les sous-menus du menu ‘Goto record’ ainsi que les éléments ‘Search for’, ‘Search forward’, et ‘Search backward’ du menu ‘Table’ seront désactivés.

- deux boutons ‘Ok’ et ‘Cancel’ pour fermer la fenêtre.

Si vous en avez terminé avec les modifications, pressez le bouton ‘Ok’ pour fermer la fenêtre.

14.3.3 Attribute Object Editor

Lorsque vous ajoutez un champ à la liste, un objet graphique par défaut lui sera attribué. Pour modifier les réglages de cet objet, un double clic ouvre la fenêtre ‘Display attribute’. Le contenu de cette fenêtre dépend du type de champ. Les éléments suivants sont disponibles pour la plupart des types de champ :

- une partie texte ‘Title’ pour le titre qui sera affiché à côté de l’objet champ (ou à l’intérieur de l’objet si c’est un bouton). Si vous laissez cette partie vide, aucun titre ne sera affiché.
- une partie choix ‘Position’ pour définir la position du titre (s’il y en a un) par rapport un champ objet. Vous avez le choix entre ‘Left’, ‘Right’, ‘Top’ et ‘Bottom’.
- une partie de texte ‘Shortcut’ pour la saisie d’une lettre qui sera utilisée conjointement avec la touche *Alt* (pour Linux et Windows) ou la touche *Amiga* (pour AmigaOS) pour activer l’objet.
- une case ‘Home’. Si elle est cochée, l’objet de champ correspondant sera l’objet "home". Cet objet sera le placement du curseur à chaque fois qu’un enregistrement sera entré. C’est très pratique dans le cas où vous entrez vos données en commençant toujours par le même champ. Lorsque vous définissez un champ en tant qu’objet "home", tous les autres objets de champ de cette table perdront cette propriété.
- une case ‘Tab chain’. Si elle est cochée, l’objet fera partie de la chaîne focalisée (focus chain) qui utilise la touche *Tab* pour faire défiler les objets. Ne cochez pas cette cas si vous ne voulez pas inclure cet élément.
- une case ‘Read only’ qui, si elle est cochée, attribuera l’état +lecture seule à l’objet. Le signe "+" signifie que vous pouvez lire le contenu mais pas le modifier. À noter cependant que l’ajout d’un bouton popup à l’objet permet de changer son contenu pour l’un des choix proposés dans la liste déroulante.
- une partie choix ‘Alignment’ pour fixer la représentation du contenu d’un objet. Vous avez le choix entre ‘Center’, ‘Left’ et ‘Right’ pour respectivement afficher au centre, à gauche ou à droite.
- une zone ‘Activé/désactivé’. Si ‘Toujours activé’ est sélectionné alors l’objet est toujours activé indépendamment de l’enregistrement affiché. ‘Désactivé sur initial’ désactive l’objet pour l’enregistrement initial mais l’active dans tous les autres cas. Si ‘Calculer activation’ est choisi alors une fonction pour calculer l’état d’activation de l’objet peut être spécifiée sur la droite. La fonction ne requiert aucun argument. Si elle renvoie NIL alors l’objet est désactivé, sinon il est activé. Dans le cas où le déclencheur est laissé vide ou ne peut pas être trouvé, l’objet est désactivé. Pour plus d’informations sur la façon d’utiliser ce déclencheur, voir [Section 15.29.10 \[Compute enabled function\]](#), page 150.
- une partie numérique ‘Weight’ pour définir le poids de l’objet. Cette valeur définit la place occupée par cet objet dans la représentation finale par rapport aux autres objets. Pour la plupart des types de champs, cela n’affectera que la taille horizontale

parce que la plupart des objets ont une taille prédéfinie. Pour un champ de type Memo, cela affectera également la taille verticale.

- une partie **Font** pour choisir la police utilisée pour l’affichage du contenu du champ. La police par défaut sera utilisée si vous laissez ce champ vide.
- une partie **Background** avec une case à cocher **Default** pour définir la représentation du fond du champ. Si la case **Default** est cochée, un fond par défaut sera utilisé. Sinon vous cliquez sur **Background** pour ouvrir une fenêtre de sélection pour le fond.
- une partie édition **Bubble help** pour définir le texte qui sera affiché dans la bulle d’aide pour l’objet de champ.
- deux boutons **Ok** et **Cancel** pour fermer et quitter la fenêtre.

Une fois les réglages effectués, appuyez sur le bouton **Ok** pour fermer la fenêtre.

14.3.4 Réglages liés au type

Parmi les éléments ci-dessus, ceux qui suivent sont spécifiques au type de champ :

- Pour les champs de type texte, il existe une page **Extras** qui contient :
 - une case **Display picture** qui, si elle est cochée, joint une partie image au champ de type texte pour afficher une image dont le nom provient du contenu du champ. La partie image est mise au dessus du champ de texte. Si vous ne cochez pas cette case, les réglages des éléments **Title at string field**, **Hide string field** et **Size** sont superflus.
 - une case **Title at string field**. Si elle est cochée, le titre de l’objet de champ est placé à gauche de la partie texte pour ménager un espace plus important pour la partie image. Sinon, le titre sera placé à côté de la partie image.
 - une case **Hide string field** pour dissimuler la partie texte dans l’interface utilisateur. Si elle est cochée, seule la partie image sera visible.
 - une partie **Size** pour définir le redimensionnement de la partie image. Si **Resizable** est sélectionné, l’objet peut être redimensionné et pourrait devenir plus grand que la taille de l’image. **Fixed** adapte la taille de l’objet à la taille de l’image. Si la taille de l’image change en fonction des enregistrements, alors l’objet est également redimensionné à chaque fois. **Scrollable** ajoute deux barres de déplacement à l’objet pour voir les images qui seraient plus grandes que la zone visible. Si **Scaled** est sélectionné, la taille de l’image est redimensionnée de façon à s’adapter à la taille de l’objet affiché. **Aspect-scaled** permet également le redimensionnement mais en préservant les proportions originales.
 - une case **File pop-up** qui, si elle est cochée, rajoute un bouton pop-up du côté droit de la partie texte. Ce bouton permet d’afficher une requête pour sélectionner un nom de fichier.
 - une partie **Font pop-up** pour ajouter un bouton pop-up permettant l’ouverture d’une requête de sélection d’une police de caractères.
 - une case **List-view pop-up**. Si elle est cochée, un bouton pop-up sera ajouté à la droite de la partie texte permettant de choisir le texte parmi une liste. Le texte des étiquettes pour la liste déroulante est défini à droite du champ. Les éléments peuvent être soit **Statiques**, soit **Calculés**. Si **Statique** est choisi alors la liste de chaînes peut être saisie dans l’éditeur d’étiquettes qui s’ouvre après

un clic sur le bouton ‘Edit labels’. Pour plus de renseignements sur l’éditeur d’étiquettes, consultez voir [Section 14.2.3 \[Label editor\]](#), page 63. Si ‘Calculé’ est sélectionné alors un déclencheur peut être désigné dans le champ ‘Compute’, il sera exécuté à chaque fois que le bouton déroulant sera cliqué. Cette fonction doit retourner un mémo contenant une étiquette par ligne ou NIL pour une liste vide (voir [Section 15.29.12 \[Compute list-view labels\]](#), page 152). Seulement une seule des cases ‘File pop-up’, ‘Font pop-up’ et ‘List-view pop-up’ peut être sélectionnée à la fois.

- une case ‘View’ qui, si elle est cochée, rajoute un bouton à droite de la partie texte permettant de lancer un programme de visualisation externe avec le contenu du champ comme argument. C’est très pratique dans le cas où vous stockez les noms de fichiers dans le champ et voudriez voir le contenu d’un fichier via un programme de visualisation externe. Ce programme de visualisation est défini dans le menu ‘Preferences – External viewer’ (voir [Section 7.1.3 \[External viewer\]](#), page 35).
- pour les champs de type Choice, il existe une partie ‘Kind’ qui vous permet de choisir si le contenu du champ sera affiché par un ‘Cycle button’ ou bien par un groupe de ‘Radio buttons’. Si vous choisissez ‘Cycle button’ alors vous pouvez définir la position du titre parmi ‘Left’, ‘Right’, ‘Top’, ou ‘Button’. Si vous choisissez ‘Radio buttons’ alors les deux éléments ‘Frame’ et ‘Horizontal’ permettent d’avoir une bordure autour des boutons et de définir l’aspect horizontal.
- pour les champs de type Real il existe une partie integer ‘Num decimals’ dans laquelle vous définissez le nombre de chiffres après la virgule qui seront affichés.
- pour les champs de type Time il existe une partie choix ‘Format’ pour définir l’aspect visuel. Vous avez le choix entre ‘HH:MM:SS’, ‘MM:SS’ et ‘HH:MM’. Si vous choisissez ‘HH:MM’, les secondes ne seront pas affichées et les valeurs d’entrées seront considérées comme des minutes.
- pour les champs de type Reference il existe une page ‘Extras’ qui comportent les éléments suivants :
 - une liste ‘Affichage’ dans laquelle vous définissez la visualisation du contenu d’un enregistrement référencé. Il est possible de sélectionner simultanément plusieurs éléments de cette liste. Si vous choisissez ‘Record number’ alors le numéro de l’enregistrement sera inclus dans la visualisation. Les autres éléments sont les noms des champs dans la table de référence. Vous pouvez changer l’ordre par simple glisser/déposer.
 - une zone ‘Popup’ dans laquelle vous pouvez spécifier quels enregistrements de la table référencée doivent être disponibles dans la liste déroulante du champ référence et comment ils doivent être affichés. Si ‘Enregistrements’ est positionné sur ‘Tous’ alors tous les enregistrements sont disponibles. Si ‘Enregistrements’ est positionné sur ‘Correspondant au filtre’ alors seuls les enregistrements de la table référencée correspondant au filtre installé sont disponibles. Si ‘Enregistrements’ est positionné sur ‘Calculé’ alors une fonction pour calculer l’ensemble d’enregistrements peut être saisie dans le champ ‘Calculer’. Cette fonction déclencheur doit retourner une liste qui est parcourue à la recherche d’enregistrements de la table référencée. Ces enregistrements identifiés sont affichés dans la liste déroulante. Les éléments qui ne sont pas

des enregistrements sont ignorés. Voir [Section 15.29.13 \[Compute reference records\], page 152](#), pour plus d'informations sur cette fonction. En plus de ces enregistrements spécifiés par le réglage 'Enregistrements', l'enregistrement initial et l'enregistrement courant de la table référencée peuvent être ajoutés à la liste déroulante en cochant respectivement les options 'Enregistrement initial' et 'Enregistrement courant'. L'option 'Utiliser une liste multi-colonnes' décide si les champs sélectionnés dans la liste 'Affichage' sont affichés sur plusieurs colonnes ou si les éléments sont concaténés et séparés par le caractère '-'.

- une partie 'Quick search' dans laquelle vous définissez comment effectuer une recherche d'élément de référence par appui de touche au clavier (where you specify how the keyboard type-ahead search for the reference item should be performed). La recherche rapide permet de trouver des entrées dans la table de référence quand le champ de référence ou la liste popup associée est l'objet en cours (Quick search allows to find entries in the referenced table when the reference field or the associated popup list view is the active object (voir [Section 8.3 \[Changing records\], page 41](#)). Si la case 'Disabled' est cochée, aucune recherche n'est possible et la frappe au clavier est tout simplement ignorée. Si vous choisissez l'option 'In first order field' la recherche s'effectuera uniquement dans le champ défini comme numéro un dans l'ordre de la table de référence (searches only in the attribute that has been specified as the first one for ordering the referenced table (voir [Section 10.2 \[Order by attributes\], page 47](#))). Si l'élément 'In all fields' est sélectionné, la recherche s'effectue dans tous les champs de la table de référence. Pour finir, dans 'Prefix search only' vous choisissez si une recherche positive doit commencer par le filtre de recherche ou si ce filtre peut exister n'importe où dans le champ. Par défaut, la recherche se fera par préfixe dans le champ défini comme numéro un parce que ce type de recherche s'effectue de manière très efficace.
- une case 'Show'. Si elle est cochée, la création de l'objet graphique représentant la référence sera un bouton. En cliquant sur celui-ci, vous verrez l'enregistrement référencé dans le masque de la table de référence. Pour cela, la fenêtre contenant la table de référence s'ouvrira et sera amenée vers l'avant si nécessaire.
- une case 'Auto show' qui, si elle est cochée, ajoute un bouton à droite de la partie référence pour une visualisation automatique ou non du champ. En l'activant, la table de référence sera mise à jour automatiquement avec l'enregistrement référencé à chaque fois que l'utilisateur passera à un autre enregistrement.
- une case 'Filter' qui, si elle est cochée, ajoute un bouton à droite de la partie référence pour appliquer ou non une filtration de référence sur le champ. Voir [Section 9.2 \[Reference filter\], page 46](#), pour plus d'informations sur les filtres de référence.
- pour les champs de type Virtuel, l'éditeur d'objet de champ contient une page 'Extras' avec les options suivantes :
 - une partie choix 'Kind' dans laquelle vous définissez comment le contenu du champ devra être affiché. Vous pouvez opter pour 'Bool' qui utilise une zone à cocher pour afficher les valeurs booléennes, ou pour 'Text' qui utilise une zone de texte pour afficher une ligne de texte (y compris la date, l'heure et les valeurs numériques),

ou encore **'List'** qui utilise une liste pour afficher une liste de lignes (exemple : le resultat d'une requête genre select-from-where).

- en réglant l'élément **'Kind'** sur **'Text'**, deux nouveaux champs feront leur apparition : **'Alignment'** pour définir comment sera présenter le contenu du champ, et **'Num decimals'** pour fixer le nombre de chiffres affiché après la virgule pour le contenu un champ de type Real.
 - si le champ **'Kind'** est réglé sur **'List'** alors les champs suivants apparaîtront : **'Show titles'**, **'Tab chain'** et **'On double click'**. Si **'Show titles'** est coché, la première ligne du contenu du champ sera affichée dans la liste comme un intitulé de ligne. Sinon, aucun titre ne sera affiché et la première ligne sera omise. Si **'Tab chain'** est coché, l'objet fera parti de la chaîne active (focus chain??) que l'on peut faire tourner (that can be cycled through??) avec la touche **Tab**. Dans **'On double click'** vous définissez ce qu'il faut faire lorsque l'utilisateur double clique un élément d'une liste : **'Do nothing'** ignore le bouble clic, **'Show record'** affiche l'enregistrement correspondant dans la table correspondante. Si vous choisissez **'Call trigger'** alors vous pouvez taper le nom de la fonction à lancer dans la partie droite. Cette fonction sera lancée à chaque double clic. Pour plus d'information sur le lancement et le passage d'arguments à une fonction, veuillez consulter voir [Section 15.29.11 \[Double click trigger\], page 151](#).
 - une case **'Auto update'**. Si elle cochée, le champ virtuel est recalculé automatiquement à chaque changement d'un élément dépendant. Cela inclut le changement d'enregistrement dans une table dépendante, la modification d'un champ dépendant, l'ajout ou la suppression d'un enregistrement dans une table dépendante, basculer le mode utilisateur/administrateur du projet ou recompiler le programme du projet. Les dépendances d'un champ virtuel sont obtenues automatiquement depuis la fonction de calcul du champ virtuel. Utilisez le menu **'Projet - Exporter la structure...'** pour visualiser toutes les dépendances obtenues (il peut être nécessaire de recompiler le programme du projet afin de mettre à jour les dépendances lors de l'activation de cette fonction). Veuillez noter que le champ virtuel est recalculé quelque soit l'enregistrement dans lequel un champ dépendant est modifié et sans tenir compte si la valeur du champ dépendant est modifiée depuis l'interface utilisateur ou au cours de l'exécution du programme. Toutefois, la mise à jour automatique des champs virtuels se déroule généralement seulement après que toutes les autres exécutions du programme sont terminées. Si **'Auto update'** est décoché, la valeur du champ virtuel n'est calculée que lorsque cela est explicitement demandé ou qu'elle est modifiée dans une fonction du programme.
- Voici les options supplémentaires pour les boutons :
 - une partie choix **'Kind'** dans laquelle vous choisissez entre **'Text button'** et **'Image button'**.
 - si vous choisissez le type **'Text button'** les options suivantes seront disponibles : **'Title'**, **'Font'**, **'Background'** et **'Default'** pour respectivement donner un titre au bouton, choisir sa police d'affichage et les réglages pour le fond.
 - si vous choisissezle type **'Image button'** les options suivantes seront disponibles : **'Image'** pour choisir l'image et **'Size'** pour définir sa taille.

14.3.5 Text Editor

Lorsque vous ajoutez un objet de texte dans la liste d’affichage ou lorsque vous en modifiez un par double clic, la fenêtre ‘Text’ apparaîtra et proposera les options suivantes :

- une partie texte ‘Title’ pour définir le texte à afficher.
- une partie numérique ‘Weight’ pour définir la largeur de l’objet de texte.
- une partie choix ‘Font’ pour définir la police du texte. Si vous laissez le champ vide, la police par défaut sera utilisée.
- une partie ‘Background’ et une partie ‘Default’ pour définir les réglages du fond de l’objet de texte.
- deux boutons ‘Ok’ et ‘Cancel’ pour fermer la fenêtre.

Quand vous en avez terminé avec les réglages, cliquez sur le bouton ‘Ok’ pour fermer la fenêtre.

14.3.6 Image Editor

L’éditeur d’images apparaît lorsque vous ajoutez un nouvel objet image ou lorsque vous double cliquez sur une image existante. Il contient les options suivantes :

- une partie ‘Weight’ dans laquelle vous définissez la largeur de l’objet image dans la représentation finale.
- une partie ‘Image’ dans laquelle vous choisissez l’image à afficher.
- une partie ‘Size’ dans laquelle vous définissez le type de redimensionnement. Si vous choisissez ‘Resizable’ l’objet image sera redimensionnable, alors que si vous choisissez ‘Fixed’, la taille sera fixe.
- deux boutons ‘Ok’ et ‘Cancel’ pour fermer la fenêtre.

Quand vous en avez terminé avec les réglages, cliquez sur le bouton ‘Ok’ pour fermer la fenêtre.

14.3.7 Space Editor

Après avoir ajouté un objet d’espacement à la liste d’affichage, vous pourrez modifier ses paramètres en double cliquant sur celui-ci, ce qui ouvrira la fenêtre ‘Space’ comportant les options suivantes :

- un champ ‘Delimiter’ qui, s’il est sélectionné, affiche une barre horizontale ou verticale (en fonction de la représentation de l’objet parent) au milieu de l’objet d’espacement. Ceci est très utile pour séparer les différents éléments dans la fenêtre de représentation.
- un champ numérique ‘Weight’ pour fixer la largeur de l’objet.
- deux champs ‘Background’ et ‘Default’ pour les réglages de fond.
- deux boutons ‘Ok’ et ‘Cancel’ pour fermer la fenêtre.

Quand vous en avez terminé avec les réglages, cliquez sur le bouton ‘Ok’ pour fermer la fenêtre.

14.3.8 Group Editor

Après avoir ajouté un objet de groupe à la liste d’affichage, vous pourrez modifier ses paramètres en double cliquant sur celui-ci, ce qui ouvrira la fenêtre ‘Group’ comportant les options suivantes :

- une partie texte **'Title'** pour donner un titre, centré au-dessus du groupe. Si vous laissez ce champ vide, aucun titre ne sera affiché.
- une partie numérique **'Weight'** pour fixer la largeur de l'objet.
- deux champs **'Background'** et **'Default'** pour les réglages de fond.
- une case **'Border'** qui, si elle est sélectionnée, affichera un cadre autour du groupe.
- une case **'Horizontal'** qui, si elle est cochée, affichera le groupe de façon horizontale et sera défini comme **'HGroup'** dans la liste d'affichage. Sinon, le groupe sera organisé de façon verticale et sera défini comme **'VGroup'** dans la liste d'affichage.
- une case **'Spacing'** qui, si elle est sélectionnée, ajoute de l'espace entre les objets child du groupe. Sinon aucun espace supplémentaire ne sera ajouté entre les objets.
- deux boutons **'Ok'** et **'Cancel'** pour fermer la fenêtre.

Quand vous en avez terminé avec les réglages, cliquez sur le bouton **'Ok'** pour fermer la fenêtre.

14.3.9 Register Group Editor

Vous pouvez modifier les réglages d'un objet groupe enregistré en double-cliquant dessus, ce qui ouvrira la fenêtre **'Register group'** proposant les options suivantes :

- une partie numérique **'Weight'** pour fixer la largeur de cet objet.
- une partie **'Labels'** pour donner un nom à chaque page enregistrée. Il est recommandé d'avoir précisément le même nombre d'étiquettes que d'éléments présents dans le groupe enregistré. Pour en savoir plus sur l'édition des étiquettes, consultez voir [Section 14.2.3 \[Label editor\], page 63](#).
- deux boutons **'Ok'** et **'Cancel'** pour fermer la fenêtre.

Quand vous en avez terminé avec les réglages, cliquez sur le bouton **'Ok'** pour fermer la fenêtre.

14.3.10 Window Editor

Vous pouvez modifier les réglages d'un objet fenêtre en double-cliquant dessus, ce qui ouvrira l'éditeur de fenêtre et proposera les options suivantes :

- une partie texte **'Title'** dans laquelle vous entrez le texte qui sera affiché dans la barre de titre de la fenêtre ainsi que dans le bouton de fenêtre optionnel.
- une partie texte **'Id'** (seulement sur Amiga) qui peut contenir jusqu'à quatre caractères correspondant à l'id d'une fenêtre MUI. Si vous ajoutez un id, alors la taille et la position de la fenêtre pourront être sauvegardés grâce à l'option "figer" de MUI. Notez que chaque id défini dans MUIbase devra être unique. Si par accident vous utilisez un id existant, les fenêtres partageront les mêmes réglages. Les identifiants commençant par un souligné **'_'** sont réservés à un usage interne à MUIbase. Si vous laissez le champ vide, aucun id ne sera défini et les dimensions de la fenêtre seront stockés dans le fichier du projet. La fenêtre principale a toujours comme id les quatre premiers caractères du nom du projet. Définir un id pour une fenêtre est utile dans le cas où l'édition d'un projet se fait sous différentes configurations, parce que les coordonnées de la fenêtre sont mémorisées individuellement pour chaque configuration.

- une case ‘**Button**’ qui, si elle est cochée, ajoute un bouton pour ouvrir la fenêtre dans la fenêtre parent. Si cette case n’est pas cochée, la fenêtre ne peut être ouverte qu’à partir d’un programme MUIbase, via la fonction SETWINDOWOPEN (voir [Section 15.21.3 \[SETWINDOWOPEN\]](#), page 137). Les options suivantes définissent l’affichage du bouton de fenêtre :
- une partie texte ‘**Shortcut**’ dans laquelle vous définissez le raccourci permettant d’activer le bouton de fenêtre.
- une zone ‘**Enabled/disabled**’. Si ‘**Toujours activé**’ est sélectionné, alors le bouton de fenêtre est toujours actif. ‘**Désactivé sur initial**’ désactive le bouton s’il est dans une table et que celle-ci affiche son enregistrement initial, sinon le bouton est activé. Si ‘**Calculer activation**’ est choisi alors une fonction pour calculer l’état d’activation du bouton de fenêtre peut être spécifiée sur la droite. La fonction ne requiert aucun argument. Si elle renvoie NIL alors le bouton est désactivé, sinon il est activé. Dans le cas où le déclencheur n’est pas renseigné ou ne peut pas être trouvé, le bouton de fenêtre est désactivé. Pour plus d’informations sur la façon d’utiliser ce déclencheur, voir [Section 15.29.10 \[Compute enabled function\]](#), page 150. De plus si la case ‘**Fermer la fenêtre si désactivé**’ est cochée alors la fenêtre est automatiquement fermée lorsque le bouton est désactivé.
- une partie numérique ‘**Weight**’ dans laquelle vous définissez la largeur du bouton de fenêtre.
- une partie ‘**Font**’ pour définir la police du bouton de fenêtre. Si vous laissez ce champ vide, la police par défaut sera utilisée.
- deux champs ‘**Background**’ et ‘**Default**’ pour définir les réglages de fond du bouton de fenêtre.
- deux boutons ‘**Ok**’ et ‘**Cancel**’ pour fermer la fenêtre.

Quand vous en avez terminé avec les réglages, cliquez sur le bouton ‘**Ok**’ pour fermer la fenêtre.

14.4 Export Structure

Parfois, il peut être utile d’avoir une vue d’ensemble de toutes les tables et champs d’un projet, par exemple lorsque vous voulez écrire un programme MUIbase. Pour cela, vous sélectionnez l’option du menu ‘**Project - Export structure**’, ce qui vous demandera un nom de fichier dans lequel sera stocké la liste des tables et des champs. Vous obtiendrez en sortie le nom du projet suivi de toutes les tables qui y sont associées. Pour chaque table, vous aurez le détail de tous les champs avec leurs types et les fonctions optionnelles qu’elles lancent. Pour les champs de type Virtuel, vous aurez également les dépendances permettant la mise à jour automatique de la valeur des champs.

15 Programmation de MUIbase

Ce chapitre (le plus long de ce manuel) décrit le langage de programmation de MUIbase, ainsi que toutes les fonctions disponibles. En revanche, ce chapitre n'est pas conçu comme un guide général sur la programmation. Vous devez être familier avec les bases de la programmation et devez avoir déjà écrit quelques programmes simples (et fonctionnant correctement :-)).

15.1 Editeur de programme

Pour saisir un programme dans un projet, ouvrez l'éditeur de programme en sélectionnant le menu **'Programme - Editer'**. Si vous êtes configuré en code source en interne (voir [Section 7.5.3 \[Program source\], page 37](#)), cela ouvre la fenêtre **'Edition de programme'** qui contient :

- un champ éditeur de texte où vous éditez le programme du projet.
- un bouton **'Compiler & fermer'** pour compiler le programme et, en cas de réussite, sortir de l'éditeur de programme.
- un bouton **'Compiler'** pour compiler le programme. Si votre programme contient une erreur quelque part, alors une description de l'erreur est affichée dans le titre de la fenêtre et le curseur est positionné sur l'emplacement fautif.
- un bouton **'Rétablir'** qui annule tous les changements effectués depuis la dernière compilation réussie.

L'éditeur de programme est une fenêtre non modale, vous pouvez laisser la fenêtre ouverte et continuer à travailler avec le reste de l'application. Vous pouvez fermer l'éditeur à tout moment en cliquant sur le bouton de fermeture de sa fenêtre, si vous avez fait des changements depuis la dernière compilation réussie, alors une requête de sécurité vous demandera de confirmer la fermeture de la fenêtre.

Si vous avez configuré le menu **'Programme - Code Source'** sur **'Externe'** alors l'éditeur externe (voir [Section 7.1.2 \[External editor\], page 34](#)) est lancé avec le nom du fichier source externe lors de la sélection du menu **'Programme - Editer'**. Cela vous permet d'éditer le programme dans votre éditeur de texte favori (voir [Section 15.2 \[External program source\], page 75](#)).

Vous pouvez également compiler le programme d'un projet sans ouvrir l'éditeur de programme en sélectionnant le menu **'Programme - Compiler'**. Cela peut être utile par exemple si vous avez fait des modifications dans un fichier d'inclusion externe et que vous désirez incorporer ces changements dans le programme du projet.

15.2 Code source externe

En choisissant le menu **'Programme - Code Source - Externe'** et en saisissant un nom de fichier, il est possible de rendre le code source du programme d'un projet accessible de l'extérieur. Cela vous permet de charger le code source du programme dans votre éditeur de texte favori pour programmer.

Si la compilation réussit, le programme compilé est intégré en tant que programme du projet et est utilisé lors de l'exécution des déclencheurs. Lorsque de la sauvegarde d'un projet, la dernière version du programme compilée avec succès est stockée dans le projet.

Du coup, après la sauvegarde et la fermeture d'un projet, le fichier source externe n'est plus nécessaire. Il est possible d'indiquer si les fichiers sources externes inutiles doivent être effacés automatiquement en cochant le menu 'Préférences - Nettoyer les sources externes'.

L'état du menu 'Programme - Code source' est mémorisé avec le projet, ainsi lors de la réouverture d'un projet utilisant la fonctionnalité du code source externe, le fichier source externe est recréé automatiquement. Si le fichier externe existe déjà et est différent de la version stockée dans le projet, une requête de sécurité demande confirmation avant d'écraser le fichier.

Sur Amiga il est possible d'envoyer la commande `compile` au port ARexx de MUIbase depuis votre éditeur. MUIbase lit alors le fichier source externe, le compile et retourne le statut de la compilation ainsi qu'un éventuel message d'erreur contenant le nom du fichier, la ligne, la colonne et une description de l'erreur. Cela permet de positionner le curseur à l'emplacement exact où l'erreur de compilation s'est produite. Voir [Section 16.9 \[ARexx compile\]](#), page 156, pour plus de détails sur les valeurs de retour et le format des erreurs.

15.3 Préprocesseur

Les programmes MUIbase sont analysés par un préprocesseur de manière similaire à ce qui est fait sur les programmes C. Cette section décrit comment utiliser les directives du préprocesseur.

Toutes les directives débutent par le symbole dièse `#` qui doit également être le premier caractère de la ligne, des caractères d'espacement ou de tabulation peuvent cependant apparaître après le `#`.

15.3.1 `#define`

`#define nom chaîne`

Définit un nouveau symbole ayant le nom et le contenu spécifiés. La *chaîne* peut être n'importe quel texte incluant des espaces et se termine à la fin de la ligne. Dans le cas où *chaîne* ne pourrait pas tenir sur une seule ligne, il est possible d'utiliser les lignes suivantes en utilisant le caractère anti-slash `\` à la fin de chacune des lignes (sauf la dernière). Si le symbole *nom* apparaît dans la suite du code source il est alors remplacé par la valeur *chaîne*.

Exemple: '(PRINTF "X vaut %i" X)' afficher 'X vaut 1' (Les occurrences de *nom* dans les chaînes ne sont pas modifiées.)

Le remplacement des symboles définis est réalisé syntaxiquement, ce qui signifie que vous pouvez remplacer des symboles par n'importe quel texte, p. ex. vous pouvez définir votre propre syntaxe comme le montre l'exemple suivant :

```
#define BEGIN (
#define END )

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

La chaîne de substitution d'une définition peut faire référence à d'autres symboles définis par la directive **#define** ce qui autorise des définitions imbriquées. Cependant un maximum de 16 définitions imbriquées est autorisé.

Voir aussi **#undef**, **#ifdef**, **#ifndef**.

15.3.2 **#undef**

#undef *nom*

Supprime la définition du symbole *nom*. Si *nom* n'est pas défini, rien ne se passe.

Voir aussi **#define**, **#ifdef**, **#ifndef**.

15.3.3 **#include**

#include *fichier*

Lit le contenu de *fichier* (une chaîne encadrée par des guillemets). MUIbase recherche le fichier à charger dans le répertoire courant et dans \$ le répertoire indiqué dans les préférences (voir [Section 7.5.8 \[Program include directory\]](#), page 39). Le contenu du fichier est alors traité par le compilateur comme s'il faisait partie du code source courant.

Un fichier externe peut inclure un ou plusieurs autres fichiers externes, avec cependant une limite maximum de 16 directives **#include** imbriquées. Pour éviter d'inclure plusieurs fois les fichiers il est possible d'utiliser la compilation conditionnelle.

Lors de l'externalisation de code sources, il convient d'être attentif : le débogage et la localisation des erreurs sont plus difficiles dans les fichiers externes. Il est préférable de n'externaliser dans des fichiers séparés que les codes sources bien testés et indépendants du projet.

15.3.4 **#if**

#if *expr-const*

Si le résultat de l'expression constante spécifiée est différent de NIL alors le texte présent jusqu'à la directive **#else**, **#elif** ou **#endif** correspondante est utilisé pour la compilation, sinon (c.-à-d. la valeur de l'expression est NIL) alors le texte jusqu'à la directive **#else**, **#elif** ou **#endif** correspondante est ignoré pour la compilation.

Pour le moment il n'est possible d'utiliser que TRUE et NIL comme expression constante.

Voir aussi **#ifdef**, **#ifndef**, **#elif**, **#else**, **#endif**.

15.3.5 **#ifdef**

#ifdef *nom*

Si le symbole spécifié a été défini par une directive **#define** alors le texte suivant jusqu'à la directive **#else**, **#elif** ou **#endif** correspondante est pris en compte dans la compilation, sinon il est ignoré.

Voir aussi **#if**, **#ifndef**, **#elif**, **#else**, **#endif**.

15.3.6 **#ifndef**

#ifndef *nom*

Si le symbole spécifié n'a pas été défini par une directive `#define` alors le texte suivant jusqu'à la directive `#else`, `#elif` ou `#endif` correspondante est pris en compte dans la compilation, sinon il est ignoré.

Voir aussi `#if`, `#ifdef`, `#elif`, `#else`, `#endif`.

15.3.7 `#elif`

`#elif expr-const`

N'importe quel nombre de directives `#elif` peut apparaître entre une directive `#if`, `#ifdef` ou `#ifndef` et sa directive `#else` ou `#endif` correspondante. Les lignes suivant la directive `#elif` sont prises en compte dans la compilation seulement si toutes les conditions suivantes sont remplies :

- L'expression constante dans la directive `#if` précédente a été évaluée à NIL, le symbole de la directive `#ifdef` précédente n'était pas défini ou le symbole de la directive `#ifndef` précédente était défini.
- L'expression constante de chacune des directives `#elif` intermédiaires a été évaluée à NIL.
- L'expression constante a été évaluée comme différente de NIL.

Si toutes les conditions précédentes sont remplies alors les directives `#elif` et `#else` suivantes sont ignorées jusqu'au `#endif` correspondant.

Voir aussi `#if`, `#ifdef`, `#ifndef`, `#else`, `#endif`.

15.3.8 `#else`

`#else`

Inverse le sens de la directive conditionnelle qui été en effet. Si la directive conditionnelle précédente indiquait que les lignes devaient être prises en compte, alors les lignes entre le `#else` et le `#endif` correspondant sont ignorées. Si la directive conditionnelle précédente indiquait que les lignes devaient être ignorées, alors les lignes suivantes sont prises en compte dans la compilation.

Les directives conditionnelles et les directives `#else` correspondantes peuvent être imbriquées jusqu'à un niveau maximum de 16 directives conditionnelles imbriquées.

Voir aussi `#if`, `#ifdef`, `#ifndef`, `#elif`, `#endif`.

15.3.9 `#endif`

`#endif`

Termine une section de lignes introduites par l'une des directives de compilation conditionnelle `#if`, `#ifdef` ou `#ifndef`. Chacune de ces directives doit avoir un `#endif` correspondant.

Voir aussi `#if`, `#ifdef`, `#ifndef`, `#elif`, `#else`.

15.4 Langage de programmation

MUIbase utilise un langage de programmation dont la syntaxe est proche du lisp. En fait, plusieurs constructions et fonctions ont été adoptées du lisp standard. Cependant, MUIbase n'est pas totalement compatible avec du lisp standard. De nombreuses fonctions manquent

(p. ex. les commandes de destruction) et le sens de certaines autres commandes est différent (p. ex. la commande `return`).

15.4.1 Pourquoi Lisp ?

L'avantage d'un langage "à la" Lisp est qu'il est possible de programmer à la fois de manière fonctionnelle et impérative. Les langages fonctionnels sont de plus en plus populaires dans les applications mathématiques. Le concept de base des langages fonctionnels est l'utilisation des expressions. Les fonctions sont définies de "manière mathématique" et la récursivité est beaucoup utilisée.

Les langages de programmation impératifs (p. ex. C, Pascal, Modula) utilisent une description impérative sur la façon de traiter les choses. Ici, c'est l'état qui est le concept de base (p. ex. variables) et un programme calcule ses sorties en passant d'un état à un autre (p. ex. en assignant des valeurs à des variables).

Lisp combine ces deux techniques et par conséquent permet de choisir la façon dont on souhaite implémenter les choses. On utilise alors la technique qui correspond le mieux à un problème spécifique ou celle que l'on préfère.

15.4.2 Syntaxe Lisp

Une expression lisp est soit une constante, une variable ou une application de fonction. Pour appeler une fonction, lisp utilise une notation préfixée. La fonction et ses paramètres sont encadrés par des parenthèses. Par exemple, pour ajouter deux valeurs `a` et `b`, on écrit :

(+ a b)

Toutes les expressions retournent une valeur, ainsi dans l'exemple précédent la somme de `a` et `b` renvoyée. Les expressions peuvent être imbriquées, c'est à dire qu'il est possible de placer une expression en tant que sous expression d'une autre.

L'évaluation des fonctions est effectuée en utilisant le principe d'appel par valeur, cela signifie que les arguments sont évalués avant d'appeler la fonction.

Sauf si indication contraire, toutes les fonctions sont strictes, c'est à dire que *tous* les arguments de la fonction sont évalués avant que la fonction ne soit appelée. Certaines fonctions cependant sont non strictes, p. ex. `IF`, `AND` et `OR`. Ces fonctions peuvent ne pas évaluer tous leurs arguments.

15.4.3 Types de programmes

MUIbase distingue trois types de programmes. Le premier est le programme du projet, dans un programme de ce type, il est possible de fonctions et des variables globales. Les fonctions peuvent être utilisées comme déclencheurs sur les champs. La saisie d'un programme de projet est réalisée dans l'éditeur de programme (voir [Section 15.1 \[Program editor\]](#), page 75).

Le deuxième type correspond aux programmes de requête où il n'est possible de saisir que des expressions. Une expression est autorisée à référencer des variables globales et à appeler des fonctions définies dans le programme du projet. Cependant, il est impossible de définir de nouvelles fonctions ou variables globales dans un tel programme. La saisie d'un programme de requête par l'intermédiaire de l'éditeur de requête (voir [Section 13.2 \[Query editor\]](#), page 55).

Le troisième type représente les expressions de filtrage. Ici, il uniquement possible de saisir des expressions contenant des appels à des fonctions MUIbase prédéfinies. Toutes

les fonctions prédéfinies ne sont pas disponibles, seules celles n’ayant pas d’effet de bord, p. ex. vous ne pouvez pas utiliser une fonction qui écrit des données dans un fichier. Les expressions de filtrage sont saisies dans la fenêtre de modification de filtrage (voir [Section 9.1.2 \[Changing filters\], page 45](#)).

15.4.4 Conventions de nommage

Dans un programme MUIbase, il est possible de définir des symboles tels que des fonctions et des variables globales ou locales. Les noms de ces symboles doivent suivre les conventions suivantes :

- Le premier caractère d’un nom doit être une lettre minuscule, cela différencie les symboles de programme des noms de table et de champs.
- Les caractères suivants peuvent être n’importe quelle lettre, chiffre ou caractère de soulignement. Les autres caractères, comme les umlauts allemands ne sont pas autorisés.

15.4.5 Accéder aux enregistrements

Pour accéder aux tables et aux champs dans un programme MUIbase, vous devez spécifier un chemin pour les atteindre. Un chemin est une liste de composants séparés par des points, où chaque composant est le nom d’une table ou d’un champ.

Les chemins peuvent être soit relatifs, soit absolus. Les chemins absolus sont reconnaissables par le fait que leur premier composant est un nom de table, suivi par une liste de champs se terminant par le champ à atteindre. P. ex. le chemin absolu ‘**Personne.Nom**’ accède au champ ‘**Nom**’ de l’enregistrement courant de la table ‘**Personne**’, de même le chemin absolu ‘**Personne.Pere.Nom**’ accède à l’attribut ‘**Nom**’ de l’enregistrement référencé par le champ ‘**Pere**’ (un champ de type Référence vers la table ‘**Personne**’).

Les chemins relatifs possèdent déjà une table courante à laquelle ils se rapportent. Par exemple dans une expression de filtrage, la table courante est la table sur laquelle porte le filtre. Le chemin relatif pour un champ de la table courante est simplement son nom. Pour les champs qui ne sont pas directement accessibles via la table courante, mais indirectement via des champs de référence, les mêmes règles que pour les chemins absolus s’appliquent.

Il n’est pas toujours évident si un chemin donné est relatif ou absolu p. ex. supposons que nous soyons en train d’écrire une expression de filtrage pour la table ‘**Foo**’ qui dispose d’un champ ‘**Bar**’ mais qu’il existe également une table ‘**Bar**’ ; dans ce cas saisir ‘**Bar**’ est ambigu : de quoi parle t’on, de la table ou du champ ? Pour cette raison, les chemins sont tout d’abord traités comme étant relatifs. Si aucun champ n’est trouvé de cette manière, alors le chemin est traité comme étant absolu, dans notre exemple, l’attribut sera préféré.

Et si nous désirons accéder à la table dans notre exemple ? Dans ce cas, le chemin doit être considéré comme absolu, pour indiquer qu’un chemin est absolu il faut ajouter deux doubles points devant le chemin. Dans notre exemple il faudrait taper ‘**::Bar**’ pour accéder à la table.

Pour mieux comprendre les chemins et leur sémantique, considérons dans l’exemple précédent que le champ ‘**Bar**’ de la table ‘**Foo**’ est une référence vers la table ‘**Bar**’ qui contient un attribut ‘**Nom**’. Maintenant, il est possible d’accéder au champ ‘**Nom**’ en tapant ‘**Bar.Nom**’ ou ‘**::Bar.Nom**’. Les deux expressions ont une signification différente. ‘**::Bar.Nom**’ indique de prendre l’enregistrement courant de la ‘**Bar**’ et de retourner la

valeur du champ ‘Nom’ de cet enregistrement, tandis que ‘Bar.Nom’ prend l’enregistrement courant de la table ‘Foo’, extrait du champ ‘Bar’ l’enregistrement référencé et récupère la valeur de son champ ‘Nom’.

Pour donner un exemple encore plus complet, considérons que la table ‘Bar’ dispose de deux enregistrements. L’un contient ‘Ralph’ et l’autre contient ‘Steffen’ dans le champ ‘Nom’. Le premier enregistrement est l’enregistrement courant. De plus, la table ‘Foo’ dispose d’un enregistrement (que l’on considère comme étant le courant) dont le champ ‘Bar’ référence le deuxième enregistrement de la table ‘Bar’. Maintenant ‘::Bar.Nom’ est évalué en ‘Ralph’ et ‘Bar.Nom’ en ‘Steffen’.

15.4.6 Types de données pour programmer

Le langage de programmation de MUIbase connaît les types de données suivants :

Type	Description
Bool	toutes les expressions, les expressions différentes de NIL sont considérées comme TRUE (NDT: VRAI).
Integer	entier long sur 32 bits, les valeurs de choix sont converties automatiquement en entiers
Real	double sur 64 bits
String	chaîne de caractères de longueur arbitraire
Memo	comme une chaîne, mais répartie sur plusieurs lignes
Date	valeur de date
Time	valeur horaire
Record	pointeur sur un enregistrement
File	descripteur de fichier pour la lecture/écriture
List	liste d’éléments, NIL est la liste vide.

Tous les types de données supportent la valeur NIL.

15.4.7 Constantes

Le langage de programmation de MUIbase gère les expressions constantes qui peuvent être saisies en fonction de son type :

Type	Description
Integer	Les constantes entières dans l’intervalle -2147483648 2147483647 peuvent être indiquées telles quelles. Les

valeurs débutant par 0 sont interprétées comme des nombres octaux, tandis que celles débutant par 0x comme des nombres hexadécimaux.

Real Les constantes réelles dans l'intervalle -3.59e308 3.59e308 peuvent être spécifiées comme habituellement, au format scientifique ou non. En l'absence de point décimal le nombre peut être traité comme un entier au lieu d'un réel.

String Les chaînes constantes sont toute suite de caractères encadrée par des guillemets, p. ex. "chaîne exemple". Entre les guillemets, tout caractère peut apparaître hormis les caractères de contrôle et les retours à la ligne. Cependant des codes d'échappement spéciaux permettent de saisir ces caractères :

<code>\n</code>	retour à la ligne (nl)
<code>\t</code>	tabulation horizontale (ht)
<code>\v</code>	tabulation verticale (vt)
<code>\b</code>	retour arrière (bs)
<code>\r</code>	retour chariot (cr)
<code>\f</code>	saut de page (ff)
<code>\\</code>	un caractère backslash
<code>\"</code>	guillemet
<code>\e</code>	code échappement 033
<code>\nnn</code>	caractère ayant le code octal <i>nnn</i>
<code>\xnn</code>	caractère ayant le code hexa <i>nn</i>

Memo Comme pour les chaînes (String).

Date Les valeurs constantes de date peuvent être spécifiées dans l'un des formats 'JJ.MM.AAAA', 'MM/JJ/AAAA' ou 'AAAA-MM-JJ', où 'JJ', 'MM' et 'AAAA' sont des valeurs de deux ou quatre chiffres représentant respectivement le jour, le mois et l'année de la date.

Time Les valeurs constantes horaires peuvent être spécifiées au format 'HH:MM:SS', où 'HH' représente les heures, 'MM' les minutes et 'SS' les secondes.

Pour quelques autres valeurs constantes prédéfinies, voir [Section 15.25 \[Pre-defined constants\]](#), page 143.

15.4.8 Typographie utilisée

Le reste de ce chapitre est consacré à la description de toutes les commandes et fonctions disponibles pour programmer dans MUIbase. La syntaxe suivante est utilisée pour la description des commandes :

- le texte entre crochets `[]` est optionnel. Si vous omettez le texte entre crochet, une valeur par défaut est utilisée.
- du texte séparé par une barre verticale `|` indique plusieurs options, p. ex. `'a | b'` signifie que vous pouvez spécifier soit `'a'` soit `'b'`.
- le texte écrit dans une police telle que *var* indique un paramètre qui est remplacé par une autre expression.
- les points de suspension `...` indiquent que d'autres expressions peuvent suivre.
- tout autre texte est obligatoire.

15.5 Commandes de définition

Cette section liste les commandes permettant de définir des fonctions et des variables globales. Ces commandes ne sont disponibles dans les programmes de projet.

15.5.1 DEFUN

DEFUN définit une fonction ayant le nom spécifié, une liste d'arguments passés à la fonction et une liste d'expressions à évaluer.

```
(DEFUN nom (liste-var) expr ...)
```

Le nom de la fonction doit commencer par une lettre minuscule, suivi d'autres lettres, chiffres ou caractères de soulignement (voir [Section 15.4.4 \[Name conventions\]](#), page 80).

Les paramètres *liste-var* spécifient les arguments de la fonction :

```
liste-var: var1 ...
```

où *var1* ... sont les noms des arguments. Ces noms doivent suivre les mêmes règles de nommage que ceux des fonctions.

Il est également possible de spécifier les types des arguments (voir [Section 15.27 \[Type specifiers\]](#), page 144).

La fonction exécute les expressions *expr*, ... une par une et retourne la valeur de la dernière. La fonction peut appeler d'autres fonctions y compris elle-même. L'appel d'une fonction définie par l'utilisateur est identique à l'appel d'une fonction prédéfinie.

par exemple pour compter le nombre d'arguments de la liste, vous pouvez définir la fonction suivante :

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

Les fonctions définies par DEFUN sont affichées dans les listes déroulantes des fenêtres liées aux tables et aux champs (voir [Section 14.1.1 \[Creating tables\]](#), page 60) et [Section 14.2.1 \[Creating attributes\]](#), page 62).

Cette commande n'est disponible que dans les programmes de projet.

Voir aussi DEFUN*, DEFVAR.

15.5.2 DEFUN*

DEFUN* est la version étoilée de DEFUN et a le même effet que DEFUN (voir [Section 15.5.1 \[DEFUN\]](#), page 83). La seule différence réside dans le fait que les fonctions définies avec DEFUN* ne sont pas affichées dans les listes déroulantes de création et de modification de table et de champ. Cependant, il est toujours possible d'entrer le nom de la fonction dans les champs texte correspondants.

Cette commande est uniquement disponible dans les programmes de projet.

Voir aussi DEFUN, DEFVAR.

15.5.3 DEFVAR

(DEFVAR *var* [*expr*])

Définit une variable globale ayant comme valeur initiale *expr* ou NIL si *expr* est absent. Les noms des variables doivent commencer par une lettre minuscule suivie de lettres, chiffres ou caractères souligné (voir [Section 15.4.4 \[Name conventions\]](#), page 80).

Il est également possible d'ajouter un spécificateur de type au nom de la variable (voir [Section 15.27 \[Type specifiers\]](#), page 144).

DEFVAR est uniquement disponible dans les programmes de projet. Toutes les commandes DEFVAR doivent être placées avant la définition de toutes les fonctions.

Après l'exécution d'un déclencheur (lorsque MUIbase rend la main à l'interface graphique), toutes les variables globales perdent leur contenu. Elles sont réinitialisées avec leur valeur initiale *expr* lors de la prochaine invocation d'un déclencheur. Si ce n'est pas voulu, il faut utiliser la commande DEFVAR* (voir [Section 15.5.4 \[DEFVAR*\]](#), page 84) qui permet de préserver la valeur des variables globales entre les appels de programme.

Il est conseillé de limiter (ou d'éviter complètement) l'utilisation des variables globales, chacune doit être initialisée (et *expr* doit être évalué s'il est fourni) à chaque fois qu'un déclencheur est appelé de l'extérieur.

Exemple: '(DEFVAR x 42)' définit une variable globale 'x' ayant la valeur 42.

Il existe quelques variables globales prédéfinies dans MUIbase (voir [Section 15.24 \[Pre-defined variables\]](#), page 142).

Voir aussi DEFVAR*, DEFUN, DEFUN*, LET.

15.5.4 DEFVAR*

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

DEFVAR* a le même effet que la commande DEFVAR (voir [Section 15.5.3 \[DEFVAR\]](#), page 84) sauf qu'une variable définie avec DEFVAR* ne perd pas sa valeur à la fin du programme.

Lors de la première invocation du programme, *var* est initialisée avec *expr* ou NIL si *expr* est omis. Les appels suivants du programme ne réévalueront pas *expr*, mais utiliseront la valeur de *var* de l'appel précédent. De cette manière, il est possible de transférer l'information d'un appel de programme à un autre sans avoir à stocker les données dans un

fichier externe ou une table de la base de données. Il faut cependant noter que toutes les variables globales définies avec `DEFVAR*` perdent leur valeur lorsque le programme du projet est recompilé. Pour conserver de manière permanente des informations, il est préférable d'utiliser un champ (éventuellement caché) d'une table.

Voir aussi `DEFVAR`, `DEFUN`, `DEFUN*`, `LET`.

15.6 Structures de contrôle

Cette section liste les fonctions contrôlant le programme, p. ex. les fonctions définissant les variables locales, les fonctions de boucle, les fonctions d'exécution conditionnelle, et d'autres.

15.6.1 PROGN

Pour évaluer plusieurs expressions l'une à la suite de l'autre, il est possible d'utiliser la construction `PROGN`.

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

exécute *expr* ... une à une. Renvoie le résultat de la dernière expression (ou `NIL` si aucune expression n'a été spécifiée). En Lisp cette construction est connue comme `(PROGN [expr ...])`.

Exemple: `'(1 2 3 4)'` retourne 4.

Voir aussi `PROG1`.

15.6.2 PROG1

Une autre manière, hormis la fonction `PROGN`, pour évaluer plusieurs expressions séquentiellement est l'expression `PROG1`.

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

exécute *expr* ... et renvoie la valeur de la première expression (ou `NIL` si aucune expression n'a été spécifiée).

Exemple: `'(PROG1 1 2 3 4)'` retourne 1.

Voir aussi `PROGN`.

15.6.3 LET

`LET` définit un nouveau bloc de variables locales. Ceci est utile, p. ex. pour définir les variables locales d'une fonction. La syntaxe est

```
(LET (liste-var) expr ...)
```

où *liste-var* est une liste de variables locales.

```
liste-var : varspec ...
```

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

Ici *var* est le nom d'une variable et doit commencer par une lettre minuscule, suivie d'autres lettres, chiffres ou caractères soulignés (voir [Section 15.4.4 \[Name conventions\]](#), [page 80](#)).

Dans le cas de l'utilisation de `(var expr)`, la nouvelle variable est initialisée par l'expression spécifiée. Dans le cas contraire, la nouvelle variable est positionnée à `NIL`.

Il est également possible d'ajouter des spécificateurs de type aux variables (voir [Section 15.27 \[Type specifiers\]](#), page 144).

Après l'initialisation de toutes les variables les expressions de la liste `expr ...` sont évaluées et la valeur de la dernière est renvoyée.

Par exemple, l'expression LET suivante

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

renvoie 1.

Voir aussi DOTIMES, DOLIST, DO, DEFVAR.

15.6.4 SETQ

La fonction SETQ affecte des valeurs à des variables, des champs et des tables.

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

Affecte à *lvalue1* la valeur de l'expression *expr*. Les points de suspension indiquent la possibilité d'affecter d'autres *parties gauches* (NDT: en anglais *lvalue* est l'abréviation de *left value*, c'est à dire littéralement *valeur de gauche* on utilise plutôt *partie gauche*). Une partie gauche est soit une variable, un champ d'une table ou une table. Dans le cas d'une variable, elle doit avoir été préalablement définie (p. ex. en utilisant l'expression LET).

Affecter la valeur à une table correspond à positionner son pointeur de programme ou d'interface : '(SETQ Table expr)' positionne le pointeur d'enregistrement du programme de Table à la valeur expr, '(SETQ Table* expr)' positionne son pointeur d'enregistrement de l'interface et rafraîchit l'affichage. Pour plus d'informations à propos des pointeurs d'enregistrement de programme et d'interface, voir [Section 5.2 \[Tables\]](#), page 19.

SETQ retourne la valeur de la dernière expression.

Exemple: '(SETQ a 1 b 2)' affecte 1 à la variable 'a', 2 à la variable 'b' et renvoie 2.

Voir aussi SETQ*, LET, DEFVAR, Tables, Sémantique des expressions.

15.6.5 SETQ*

SETQ* est la version étoilée de SETQ (voir [Section 15.6.4 \[SETQ\]](#), page 86) et a les mêmes effets. La différence réside dans le fait que lors de l'affectation à un champ, SETQ* appelle le déclencheur de ce champ (voir [Section 15.29.8 \[Attribute trigger\]](#), page 149) au lieu de lui affecter directement la valeur. Dans le cas où aucun déclencheur ne serait associé à un champ, SETQ* se comporte comme SETQ et affecte simplement la valeur au champ.

Exemple: '(SETQ* Table.Attr 0)' appelle le déclencheur de 'Table.Attr' avec un argument à 0.

Attention : Avec cette fonction il est possible d'écrire des boucles infinies, p. ex. si un déclencheur est défini pour un champ et que ce déclencheur utilise SETQ* pour affecter la valeur.

Voir aussi SETQ*, LET, DEFVAR.

15.6.6 FUNCALL

FUNCALL est utilisé pour appeler une fonction avec des arguments.

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

Appelle la fonction *fonc-expr* avec les arguments spécifiés. L'expression *fonc-expr* peut être n'importe quelle expression dont la valeur est une fonction utilisateur ou prédéfinie, p. ex. une variable contenant la fonction à appeler. Si le nombre d'argument est incorrect, un message d'erreur est généré.

FUNCALL renvoie la valeur de retour de l'appel à la fonction ou NIL si *fonc-expr* est NIL.

Pour plus d'informations à propos des expressions fonctionnelles, voir [Section 15.26 \[Functional parameters\]](#), page 143.

Voir aussi APPLY.

15.6.7 APPLY

APPLY est utilisé pour appliquer une fonction à une liste d'arguments.

```
(APPLY fonc-expr [expr ...] expr-liste)
```

Applique la fonction *fonc-expr* à une liste créée en associant les arguments *expr* ... à *expr-liste*. En d'autres termes : appelle la fonction *fonc-expr* avec les arguments *expr* ... et *expr-liste* étendue à ses éléments de liste.

L'expression *fonc-expr* peut être n'importe quelle expression dont la valeur est une fonction utilisateur ou prédéfinie, p. ex. une variable contenant la fonction à appeler. Le dernier argument, *expr-liste*, doit être une liste valide ou NIL, dans le cas contraire un message d'erreur est généré. Si le nombre d'arguments n'est pas correct, une erreur survient.

APPLY renvoie la valeur de retour de l'appel de la fonction ou NIL si *fonc-expr* est NIL.

Pour plus d'informations à propos des expressions fonctionnelles, voir [Section 15.26 \[Functional parameters\]](#), page 143.

Exemple: '(APPLY + 4 (LIST 1 2 3))' renvoie 10.

Voir aussi FUNCALL.

15.6.8 IF

IF est un opérateur conditionnel.

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

L'expression *expr1* est évaluée, si son résultat n'est pas NIL, alors la valeur de *expr2* est retournée, sinon c'est la valeur de *expr3* (ou NIL en cas d'absence).

Cette fonction n'est pas stricte, c'est à dire qu'une seule des expressions *expr2* ou *expr3* est évaluée.

Voir aussi CASE, COND.

15.6.9 CASE

CASE est similaire à l'instruction `switch` du langage C.

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

Ici *expr* est l'expression de sélection et *choix* ... des paires composées de :

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

où *valeur* est une instruction ou une liste d'instructions et *expr ...* les expressions à exécuter si l'une des expressions du choix est remplie.

L'expression **CASE** évalue d'abord *expr*. Puis chaque paire de choix est testée pour vérifier si elle (ou l'une des expressions de la liste) correspond à l'expression évaluée. Si une expression de choix valide est trouvée, alors les expressions correspondantes sont exécutées et la valeur de la dernière expression est renvoyée. Si aucun choix ne convient, **NIL** est retourné.

Exemple: '(CASE 1 ((2 3 4) 1) (1 2))' retourne 2.

Voir aussi IF, COND.

15.6.10 COND

COND est, comme **IF**, un opérateur conditionnel.

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

COND teste la première expression de chaque liste une à une. Pour la première qui ne renvoie par **NIL**, les expressions associées *expr ...* sont évaluées et la valeur de la dernière est retournée.

Si toutes les expressions testées retournent **NIL**, alors la valeur de retour de **COND** sera également **NIL**.

Exemple

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

retourne "1 < 2".

Voir aussi IF, CASE.

15.6.11 DOTIMES

Pour des boucles simples, la commande **DOTIMES** est utilisable.

```
(DOTIMES (nom expr-entière [expr-résultat ...]) [expr-boucle ...])
```

Ici, *nom* est le nom d'une nouvelle variable qui sera utilisée dans la boucle. Le nom doit commencer par une lettre minuscule, suivie par d'autres lettres, chiffres ou caractères souligné (voir [Section 15.4.4 \[Name conventions\]](#), page 80).

Le nombre fois que la boucle sera exécutée est indiqué par *expr-entière*. Dans *expr-résultat ...* il est possible de donner des expressions qui seront exécutées après la dernière boucle. *expr-boucle* correspond au corps de la boucle, c'est à dire les expressions qui sont évaluées dans chaque exécution de la boucle.

Avant d'exécuter la boucle, **DOTIMES** calcule la valeur de *expr-entière* pour déterminer le nombre de fois que la boucle sera exécutée. *expr-entière* n'est évaluée qu'une seule fois au début de la boucle et doit retourner une valeur entière. Ensuite, **DOTIMES** modifiera la valeur de la variable de boucle à chaque exécution de la boucle de 0 à *expr-entière*-1. Tout d'abord, la variable est initialisée à zéro puis comparée à la valeur de *expr* pour voir si elle

est déjà supérieure ou égale. Si *expr-entière* est négative ou NIL ou si elle est supérieure ou égale à la valeur de *expr* alors la boucle se termine et les expressions de résultat sont évaluées. Dans le cas contraire, les expressions de la boucle sont évaluées et la variable est incrémentée de un, puis l'exécution revient au test d'arrêt et, éventuellement exécute d'autres boucles.

L'expression DOTIMES retourne la valeur de la dernière expression de résultat ou NIL si aucune expression de résultat n'a été donnée.

Exemple

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

Affiche les nombres de 0 à 49 et retourne la valeur 50.

Voir aussi DOLIST, DO, FOR ALL, LET.

15.6.12 DOLIST

Pour boucler sur des listes, l'expression DOLIST peut être utilisée.

```
(DOLIST (nom expr-liste [expr-résultat ...]) [expr-boucle ...])
```

Ici, *nom* est le nom d'une nouvelle variable qui sera utilisée dans la boucle. Le nom doit commencer par une lettre minuscule, suivie par d'autres lettres, chiffres ou caractères souligné (voir [Section 15.4.4 \[Name conventions\]](#), page 80).

expr-liste spécifie la liste sur laquelle la boucle doit s'exécuter, *expr-résultat ...* sont des expressions qui seront évaluées après la dernière boucle, et *expr-boucle ...* correspond au corps de la boucle.

Avant d'exécuter la boucle, DOLIST calcule la valeur de *expr-liste*. Cette expression n'est évaluée qu'une seule fois au démarrage de la boucle et doit retourner une valeur de type liste. Ensuite, DOTIMES positionnera la variable de boucle à chacun des noeuds de la liste, un à chaque exécution de la boucle. Tout d'abord, la variable est initialisée au premier noeud de la liste. Si la liste est vide (NIL) alors la boucle se termine et les expressions de résultat sont évaluées. Dans le cas contraire, les expressions de la boucle sont évaluées et la variable est positionnée sur le noeud suivant de la liste, puis l'exécution revient au test d'arrêt et, éventuellement exécute d'autres boucles.

L'expression DOLIST retourne la valeur de la dernière expression de résultat ou NIL si aucune expression de résultat n'a été donnée.

Exemple

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

Affiche tous les enregistrements de la table 'Comptes' et retourne NIL.

Voir aussi DOTIMES, DO, FOR ALL, LET.

15.6.13 DO

L'expression DO permet de programmer des boucles arbitraires.

```
(DO ([binding ...]) (expr-terminaison [expr-résultat ...]) [expr-boucle ...])
```

où *binding ...* are the variable bindings, each of which is either:

- un nouveau nom de variable (qui sera initialisé à NIL)

- une liste de la forme : (*nom init [pas]*) où *nom* est le nom de la nouvelle variable, *init* est la valeur initiale de cette variable et *pas* est l'expression de pas (NDT: c'est à dire que c'est elle qui modifie la valeur de la variable d'une exécution de boucle à une autre).

De plus *expr-terminaison* est l'expression du test d'arrêt, *expr-résultat* . . . les expressions du résultat (NIL en cas d'absence) et *expr-boucle* . . . correspond au corps de la boucle.

L'expression **DO** commence par initialiser toutes les variables locales avec leur valeur initiale, ensuite elle examine les expressions de terminaison. Si celles si retournent TRUE (NDT: TRUE en anglais) la boucle est arrêtée et les expressions de résultat sont évaluées, et la valeur de la dernière est retournée. Dans le cas contraire, la boucle (*expr-loop* . . .) est exécutée et chaque variable est mise à jour par la valeur de son expression de pas. Ensuite l'exécution au test d'arrêt et ainsi de suite.

Exemple

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

Affiche les valeurs 0, 1, 2, 3 et 4 et renvoie la valeur 5. Bien sûr c'est une façon un peu compliquée d'écrire une simple boucle **FOR**. Par ailleurs il existe une version plus simple, l'expression **DOTIMES**.

Voir aussi **DOTIMES**, **DOLIST**, **FOR ALL**, **LET**.

15.6.14 FOR ALL

L'expression **FOR ALL** est utilisée pour boucler sur une liste d'enregistrements.

```
(FOR ALL liste-tables [WHERE expr-sélection] [ORDER BY list-ordre] DO expr ...)
```

Ici, *table-list* est une liste de tables séparées par des virgules, *expr-sélection* une expression à valider pour chaque enregistrement, *list-ordre* une liste d'expressions séparées par des virgules pour ordonner les enregistrements, et *expr* . . . les expressions à exécuter pour chaque enregistrement.

FOR ALL commence par générer la liste de tous les enregistrements pour lesquels le corps de la boucle devra être exécuté. C'est réalisé de la même manière que dans l'expression **SELECT**. voir [Section 15.20.12 \[SELECT\], page 135](#) pour plus d'informations sur la façon dont cette liste est générée. Pour chacun des éléments de cette liste, le corps de la boucle *expr* . . . est exécuté.

Par exemple, faire la somme d'un attribut d'une table peut être réalisé de la manière suivante :

```
(SETQ sum 0)
(FOR ALL Comptes DO
  (SETQ sum (+ sum Comptes.Solde))
)
```

L'expression **FOR ALL** renvoie NIL.

Voir aussi **SELECT**, **DOTIMES**, **DOLIST**, **DO**.

15.6.15 NEXT

NEXT peut être utilisé pour contrôler les boucles **DOTIMES**, **DOLIST**, **DO** et **FOR ALL**.

Un appel à **NEXT** dans le corps d'une boucle fera passer à la prochaine itération de la boucle. Cela peut être utilisé pour sauter des itérations de boucle sans intérêt, comme dans l'exemple suivant :

```
(FOR ALL Table DO
  (IF pas-intéressé-par-enregistrement-courant (NEXT))
  ...
)
```

Voir aussi **EXIT**, **DOTIMES**, **DOLIST**, **DO**, **FOR ALL**.

15.6.16 EXIT

EXIT peut être utilisé pour terminer une boucle.

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

A l'intérieur du corps d'une boucle, **EXIT** termine la boucle, exécute les éventuelles expressions *expr* ..., et renvoie la valeur de la dernière expression (ou **NIL** en cas d'absence) comme valeur de retour pour la boucle. Les éventuelles expressions de résultat de la boucle comme par exemple dans

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

ne sont pas exécutées.

Il est par exemple possible d'utiliser la fonction **EXIT** pour terminer une boucle **FOR ALL** lorsque l'enregistrement recherché a été trouvé :

```
(FOR ALL Table DO
  (IF intéressé-par-enregistrement-courant (EXIT Table))
  ...
)
```

Voir aussi **NEXT**, **RETURN**, **HALT**, **DOTIMES**, **DOLIST**, **DO**, **FOR ALL**.

15.6.17 RETURN

Lors de la définition d'une fonction, il est possible de rendre la main à l'appelant en utilisant la commande **RETURN**.

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

arrête l'exécution de la fonction, exécute les expressions *expr* ... optionnelles, et renvoie la valeur de la dernière (ou **NIL** en cas d'absence).

Exemple

```
(DEFUN find-record (nom)
  (FOR ALL Table DO
    (IF (= Nom nom) (RETURN Table))
  )
)
```

Cet exemple recherche un enregistrement dont le champ **Nom** correspond au nom spécifié. La fonction renvoie le premier enregistrement correspondant ou **NIL** si aucun enregistrement n'est trouvé.

Voir aussi **HALT**, **EXIT**.

15.6.18 HALT

HALT peut être utilisé pour terminer l'exécution du programme.

(HALT)

stoppe l'exécution du programme silencieusement.

Voir aussi ERROR, EXIT, RETURN.

15.6.19 ERROR

Pour avorter l'exécution du programme avec un message d'erreur, il est possible d'utiliser la fonction ERROR.

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

stoppe l'exécution du programme et affiche une fenêtre avec un message d'erreur. Ce message est généré à partir de *fmt* et des arguments optionnels *arg* ... comme avec la fonction SPRINTF (voir [Section 15.12.32 \[SPRINTF\]](#), page 108).

Voir aussi HALT, SPRINTF.

15.7 Prédicats de typage

Pour chaque type il existe un prédicat qui retourne TRUE si l'expression fournie est du type spécifié ou NIL dans le cas contraire. Ces prédicats sont :

Prédicat	Description
(STRP <i>expr</i>)	TRUE si <i>expr</i> est de type chaîne, NIL sinon.
(MEMOP <i>expr</i>)	TRUE si <i>expr</i> est de type mémoire, NIL sinon.
(INTP <i>expr</i>)	TRUE si <i>expr</i> est de type entier, NIL sinon.
(REALP <i>expr</i>)	TRUE si <i>expr</i> est de type réel, NIL sinon.
(DATEP <i>expr</i>)	TRUE si <i>expr</i> est de type date, NIL sinon.
(TIMEP <i>expr</i>)	TRUE si <i>expr</i> est de type heure, NIL sinon.
(NULL <i>expr</i>)	TRUE si <i>expr</i> est NIL (une liste vide), NIL sinon.
(CONSP <i>expr</i>)	TRUE si <i>expr</i> n'est pas la liste vide, NIL sinon.
(LISTP <i>expr</i>)	TRUE si <i>expr</i> est une liste (peut être NIL), NIL sinon.
(RECP <i>table expr</i>)	TRUE si <i>expr</i> pointe sur un enregistrement de la table spécifiée. Si <i>expr</i> est NIL, TRUE est renvoyé (enregistrement initial). Si <i>table</i> est NIL, vérification si <i>expr</i> est un pointeur d'enregistrement vers l'une des tables.

15.8 Fonctions de conversion de type

Cette section liste les fonctions de conversion d'un type vers un autre.

15.8.1 STR

STR est utilisé pour convertir une expression vers une représentation en chaîne.

(STR *expr*)

convertit *expr* en une représentation chaîne. Le type de *expr* détermine la conversion :

Type	Chaîne retournée
String	La chaîne elle-même.
Memo	Le texte complet du mémo dans une chaîne.
Integer	La représentation de la valeur entière.
Real	La représentation de la valeur réelle. Si <i>expr</i> est un champ de table, alors le nombre de décimales spécifié pour ce champ est utilisé, sinon 2 décimales sont produites.
Choice	Le label associé au choix.
Date	La représentation de la date en chaîne.
Time	La représentation de l'heure en chaîne.
Bool	La chaîne "TRUE"
NIL	La chaîne nil définie par l'utilisateur si <i>expr</i> est un champ, la chaîne "NIL" sinon.
Record	La représentation du numéro d'enregistrement en chaîne.
Autres	La représentation de l'adresse interne du pointeur en chaîne.

Voir aussi MEMO, SPRINTF.

15.8.2 MEMO

MEMO est utilisé pour convertir une expression en mémo.

(MEMO *expr*)

convertit *expr* en mémo. Elle traite l'expression de la même manière que la fonction STR (voir [Section 15.8.1 \[STR\]](#), page 93) mais retourne un mémo plutôt qu'une chaîne.

Voir aussi STR.

15.8.3 INT

INT est utilisé pour convertir une expression en entier.

(INT *expr*)

convertit *expr* en valeur entière. Les conversions possibles sont :

Type	Valeur retournée
String	Si la chaîne complète représente une valeur entière valide, elle est convertie en entier. Une chaîne débutant par un 0 est interprétée comme un nombre octal, une débutant par 0x comme un nombre hexadécimal. Les espaces au début et à la fin sont ignorés. Si la chaîne ne représente pas une valeur entière, NIL est retourné.
Memo	Idem que pour le type String.
Integer	La valeur elle-même.
Real	Si la valeur est comprise dans l'intervalle de valeurs autorisé pour les entiers, alors la valeur réelle est arrondie avant d'être retournée, sinon NIL est renvoyé.
Choice	Le numéro interne (à partir de 0) du label courant.
Date	Le nombre de jours depuis 01.01.0000.
Time	le nombre de secondes depuis 00:00:00.
Record	Le numéro d'enregistrement.
NIL	NIL
Autre	Un message d'erreur est généré et l'exécution du programme avortée.

Voir aussi REAL, ASC.

15.8.4 REAL

REAL est utilisé pour convertir une expression en une valeur de type réel.

(REAL *expr*)

convertit *expr* en un réel. L'expression est traitée de la même manière qu'avec la fonction INT (voir [Section 15.8.3 \[INT\], page 94](#)) mais renvoie une valeur de type réel au lieu d'un entier.

Voir aussi INT.

15.8.5 DATE

DATE est utilisé pour convertir une expression en une date.

(DATE *expr*)

converti l'expression spécifiée en une valeur de type date. Les conversions possibles sont :

Type	Valeur retournée
String	Si la totalité de la chaîne représente une date, alors la chaîne est convertie en une valeur de type date. Les espaces au début et à la fin sont ignorés. Dans le cas contraire, NIL est renvoyé.
Memo	Idem que pour le type String.
Integer	Une date est générée en utilisant l'entier comme étant le nombre de jours depuis 01.01.0000. Si la valeur entière est trop grande (c.-à-d. la date serait supérieure au 31.12.9999) ou négative alors NIL est retourné.
Real	Idem que pour le type Integer.
Date	La date elle-même.
NIL	NIL
Autre	Un message d'erreur est généré et l'exécution du programme avortée.

Voir aussi DATEDMY.

15.8.6 TIME

TIME est utilisé pour convertir une expression en une heure.

(TIME *expr*)

convertit l'expression spécifiée en une valeur de type date. Les conversions possibles sont :

Type	Valeur retournée
String	Si la totalité de la chaîne représente une heure, alors la chaîne est convertie en une valeur de type heure. Les espaces au début et à la fin sont ignorés. Dans le cas contraire, NIL est renvoyé.
Memo	Idem que pour le type String.
Integer	Une heure est générée en utilisant l'entier comme étant le nombre de secondes depuis 00:00:00.

Real	Idem que pour le type Integer.
Time	L'heure elle-même.
NIL	NIL
Autre	Un message d'erreur est généré et l'exécution du programme avortée.

15.9 Fonctions booléennes

Cette section liste les opérateurs booléens.

15.9.1 AND

AND vérifie que tous ses arguments sont vrais (TRUE).

(AND [*expr* ...])

vérifie *expr* ... une à une jusqu'à ce qu'une expression retourne NIL. Si toutes les expressions sont différentes de NIL alors la valeur de la dernière expression est renvoyée, sinon NIL est retourné.

Cette fonction n'est pas stricte, ce qui signifie que ses arguments peuvent ne pas être évalués, p. ex. dans '(AND NIL (+ 1 2))' l'expression '(+ 1 2)' n'est pas évaluée puisqu'une valeur NIL a déjà été traitée, en revanche dans '(AND (+ 1 2) NIL)' l'expression '(+ 1 2)' sera bien évaluée.

Voir aussi OR, NOT.

15.9.2 OR

OR vérifie que tous les arguments sont NIL.

(OR [*expr* ...])

vérifie *expr* ... une à une jusqu'à ce qu'une expression retourne une valeur différente de NIL. Retourne la valeur de la première expression différente de NIL ou NIL si toutes les expressions sont NIL.

Cette fonction n'est pas stricte, ce qui signifie que ses arguments peuvent ne pas être évalués, p. ex. dans '(OR TRUE (+ 1 2))' l'expression '(+ 1 2)' n'est pas évaluée puisqu'une valeur différente de NIL a déjà été traitée, en revanche dans '(OR (+ 1 2) TRUE)' l'expression '(+ 1 2)' sera effectivement évaluée.

Voir aussi AND, NOT.

15.9.3 NOT

NOT est utilisé pour inverser la valeur d'une expression booléenne.

(NOT *expr*)

retourne TRUE si *expr* est NIL, NIL dans le cas contraire.

Voir aussi AND, OR.

15.10 Fonctions de comparaison

Cette section présente les fonctions de comparaison de valeur.

15.10.1 Opérateurs relationnels

Pour comparer deux valeurs dans un programme MUIbase, il faut utiliser

`(op expr1 expr2)`

où *op* est parmi {=, <, >, >=, <=, =*, <*>, <*, >*, >=*, <=*}. L'étoile est utilisée pour les comparaisons spéciales (p. ex. les chaînes sont comparées en tenant compte de la casse, les enregistrements en utilisant l'ordre défini par l'utilisateur).

Le tableau suivant indique les règles utilisées pour comparer deux valeurs dans un programme MUIbase.

Type	Relation d'ordre
Entier	<code>NIL < MIN_INT < ... < -1 < 0 < 1 < ... < MAX_INT</code>
Réel	<code>NIL < -HUGE_VAL < ... < -1.0 < 0.0 < 1.0 < ... < HUGE_VAL</code>
Chaîne	<code>NIL < "" < "Z" < "a" < "aa" < "b" < ...</code> (sensible à la casse) <code>NIL < * "" < * "a" < * "AA" < * "b" < ...</code> (insensible à la casse)
Mémo	Identique aux chaînes
Date	<code>NIL < 1.1.0000 < ... < 31.12.9999</code>
Heure	<code>NIL < 00:00:00 < ... < 596523:14:07</code>
Booléen	<code>NIL < TRUE</code>
Enregistrement	<code>NIL < un_enregistrement</code> (les enregistrements ne sont pas comparables avec <) <code>NIL < * rec1 < * rec2</code> (l'ordre définit par l'utilisateur)

Voir aussi `CMP`, `CMP*`, `LIKE`.

15.10.2 CMP

`CMP` retourne un entier représentant l'ordre de ses arguments.

`(CMP expr1 expr2)`

retourne une valeur inférieure à 0 si *expr1* est plus petite que *expr2*, 0 si *expr1* est égale à *expr2*, et une valeur supérieure à 0 si *expr1* est plus grand que *expr2*. Pour déterminer l'ordre, la relation d'ordre simple (non étoilée) est utilisée comme dans les opérateurs relationnels (voir [Section 15.10.1 \[Relational operators\]](#), page 97).

Ne supposez pas que la valeur retournée sera toujours -1, 0 ou 1 !

Exemple: `'(CMP "Velo" "vELO")'` renvoie -1.

Voir aussi `CMP*`, Opérateurs relationnels.

15.10.3 CMP*

CMP* est la version étoilée de **CMP**. La différence est que **CMP*** utilise un ordre étendu comme défini dans les opérateurs relationnels (voir [Section 15.10.1 \[Relational operators\], page 97](#)) où les chaînes sont comparées sans tenir compte de la casse et les enregistrements sont comparés en utilisant l'ordre défini par l'utilisateur.

Exemple: '(CMP* "Velo" "vELO")' renvoie 0.

Voir aussi **CMP**, Opérateurs relationnels.

15.11 Fonctions mathématiques

Ici quelques fonctions mathématiques sont listées.

15.11.1 Ajouter des valeurs

Pour ajouter des valeurs, utilisez

```
(+ expr ...)
```

Renvoie la somme des arguments *expr* ... Si l'un des arguments est **NIL** alors le résultat est **NIL**. Si les valeurs sont de type réel ou entier, alors la valeur du résultat sera de type réel ou entier.

Il est également possible d'ajouter des chaînes ou des mémos. Dans ce cas le résultat est la concaténation des chaînes ou des mémos.

Si *expr* est de type date et que le reste des arguments est de type entier/réel alors la somme des entiers/réels est interprétée comme un nombre de jours et est ajoutée à *expr*. Si la date résultante est hors limite (inférieure à 1.1.0000 ou supérieure à 31.12.9999) alors le résultat est **NIL**.

Si *expr* est de type heure et que le reste des arguments est de type entier, réel ou heure, alors la somme des entiers/réels (interprétée comme un nombre secondes) ainsi que les valeurs horaires sont ajoutées à *expr*. Si l'heure résultante est hors limite (inférieur à 00:00:00 ou supérieure à 596523:14:07) alors le résultat est **NIL**.

Exemples

Expression	Valeur
(+ 1 2 3)	6
(+ 5 1.0)	6.0
(+ "Hello" " " "world!")	"Hello world!"
(+ 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

Voir aussi -, 1+, *, **CONCAT**, **CONCAT2**, **ADDMONTH**, **ADDYEAR**.

15.11.2 Soustraire des valeurs

Pour soustraire des valeurs, utilisez

`(- expr1 expr2 ...)`

Soustrait la somme de *expr2* ... à *expr1*. Les mêmes règles que pour l'ajout de valeurs (voir [Section 15.11.1 \[add\]](#), [page 98](#)) s'appliquent sauf que les chaînes et les mémos ne peuvent se soustraire.

`(- expr)` a une signification particulière : cela retourne la valeur négative de *expr* (entier ou réel), p. ex. `'(- (+ 1 2))'` retourne -3.

Voir aussi +, 1-.

15.11.3 1+

1+ incrémente de un une expression entière ou réelle.

`(1+ expr)`

Retourne la valeur de *expr* (entier ou réel) augmentée de un. Si *expr* est NIL alors le résultat est également NIL.

Voir aussi +, 1-.

15.11.4 1-

1- décrémente de un une expression entière ou réelle.

`(1- expr)`

Retourne la valeur de *expr* (entier ou réel) diminuée de un. Si *expr* est NIL alors le résultat est également NIL.

Voir aussi -, 1+.

15.11.5 Multiplier des valeurs

Pour multiplier des valeurs entières/réelles, utilisez :

`(* expr ...)`

Retourne la multiplication des valeurs entières/réelles *expr* Si tous les arguments sont entiers alors un entier est retourné, sinon le résultat est de type réel.

Voir aussi +, /.

15.11.6 Diviser des valeurs

Pour diviser des valeurs entières/réelles, utilisez :

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

Divise *expr1* par le résultat de la multiplication du reste des arguments. Retourne une valeur réelle, NIL est renvoyé en cas de division par zéro.

Voir aussi *, DIV, MOD.

15.11.7 DIV

DIV permet d'effectuer des divisions entières.

`(DIV int1 int2)`

Retourne le résultat de la division entière de *int1* par *int2*. Par exemple, `'(DIV 5 3)'` renvoie 1.

Voir aussi /, MOD.

15.11.8 MOD

MOD permet de calculer le modulo.

(MOD *int1 int2*)

Retourne *int1* modulo *int2*. Par exemple, ‘(MOD 5 3)’ renvoie 2.

Voir aussi DIV.

15.11.9 MAX

MAX retourne l’argument ayant la plus grande valeur.

(MAX *expr ...*)

Retourne la valeur maximum parmi les arguments *expr ...* (tous entiers ou réels). Si l’une des expressions est NIL alors le résultat sera NIL.

Voir aussi MIN.

15.11.10 MIN

MIN retourne l’argument ayant la plus petite valeur.

(MIN *expr ...*)

Retourne la valeur minimale parmi les arguments *expr ...* (tous entiers ou réels). Si l’une des expressions est NIL alors le résultat sera NIL.

Voir aussi MAX.

15.11.11 ABS

ABS calcule la valeur absolue d’une expression.

(ABS *expr*)

Retourne la valeur absolue de *expr* (entier ou réel). Si *expr* vaut NIL alors le résultat est NIL.

15.11.12 TRUNC

TRUNC tronque les décimales d’une valeur réelle.

(TRUNC *reel*)

Retourne le plus grand entier (sous forme réelle) qui n’est pas plus grand que le nombre réel spécifié. Si *reel* vaut NIL, le résultat est NIL.

Exemples: ‘(TRUNC 26.1)’ renvoie 26, ‘(TRUNC -1.2)’ renvoie -2.

Voir aussi ROUND.

15.11.13 ROUND

ROUND arrondit une valeur réelle.

(ROUND *reel décimales*)

Renvoie le nombre réel spécifié arrondi à *décimales* chiffres après la virgule. Si *reel* ou *décimales* sont NIL, le résultat est NIL.

Exemples: ‘(ROUND 70.70859 2)’ renvoie 70.71, ‘(ROUND 392.36 -1)’ renvoie 390.0.

Voir aussi TRUNC.

15.11.14 RANDOM

RANDOM permet de générer des nombres aléatoires.

(RANDOM *expr*)

Renvoie un nombre aléatoire. Au premier appel, le générateur de nombre aléatoire est initialisé avec une valeur générée à partir de l'heure courante. RANDOM génère un nombre dans l'intervalle 0 ... *expr*, en excluant la valeur *expr*. Le type de retour dépend de celui de *expr* : entier ou réel. Si *expr* est NIL, le résultat est NIL.

Exemples:

Exemple	Signification
(RANDOM 10)	retourne une valeur entre 0 et 9 compris,
(RANDOM 10.0)	retourne une valeur entre 0.0 et 9.99999... compris

15.12 Fonctions sur les chaînes

Cette section concerne les fonctions ayant trait aux chaînes de caractères.

15.12.1 LEN

LEN calcule la longueur d'une chaîne.

(LEN *str*)

Retourne la longueur de la chaîne spécifiée ou NIL si *str* est NIL.

Voir aussi WORDS, LINES, MAXLEN.

15.12.2 LEFTSTR

LEFTSTR extrait une sous-chaîne de la chaîne.

(LEFTSTR *str long*)

Retourne la partie gauche de la chaîne spécifiée avec au plus *long* caractères. Si *str* ou *long* sont NIL ou si *long* est négatif alors NIL est retourné.

Exemple: '(LEFTSTR "Hello world!" 5)' retourne "Hello".

Voir aussi RIGHTSTR, MIDSTR, WORD, LINE.

15.12.3 RIGHTSTR

RIGHTSTR extrait une sous-chaîne de la chaîne.

(RIGHTSTR *str long*)

Retourne la partie droite de la chaîne spécifiée avec au plus *long* caractères. Si *str* ou *long* sont NIL ou si *long* est négatif alors NIL est retourné.

Exemple: '(RIGHTSTR "Hello world!" 6)' retourne "world!".

Voir aussi LEFTSTR, MIDSTR, WORD, LINE.

15.12.4 MIDSTR

MIDSTR extrait une sous-chaîne de la chaîne.

`(MIDSTR str pos long)`

Retourne une partie de la chaîne spécifiée avec au plus *long* caractères. La sous-chaîne débute à la position *pos* (à partir de zéro). Si *long* est NIL alors toute la fin de la chaîne à partir de la position *pos* est retournée. Si *str* est NIL ou si *long* est négatif alors NIL est retourné. Si *pos* est hors limite, c.-à-d. négatif ou supérieur à la longueur de la chaîne, NIL est retourné.

Exemple: ‘`(MIDSTR "Hello world!" 3 5)`’ retourne "lo wo".

Voir aussi LEFTSTR, RIGHTSTR, WORD, LINE, SETMIDSTR, INSMIDSTR.

15.12.5 SETMIDSTR

SETMIDSTR remplace une partie de la chaîne.

`(SETMIDSTR str index set)`

Retourne une copie de la chaîne *str* où la sous-chaîne débutant à *index* est remplacée par la chaîne *set*. La longueur de la chaîne retournée est plus supérieure ou égale à la celle de *str*. Si l’un des arguments est NIL ou si *index* est hors limite, le résultat est NIL.

Exemple: ‘`(SETMIDSTR "Salut monde!" 6 "Mélanie!")`’ retourne "Salut Mélanie!".

Voir aussi INSMIDSTR, REPLACESTR.

15.12.6 INSMIDSTR

INSMIDSTR permet d’insérer une chaîne dans une autre.

`(INSMIDSTR str index insert)`

Retourne une copie de la chaîne *str* où la chaîne *insert* a été insérée à l’index spécifié. Si l’un des arguments est NIL ou si *index* est hors limite, le résultat est NIL.

Exemple: ‘`(INSMIDSTR "Salut le monde !" 14 " de MUIbase")`’ retourne "Salut le monde de MUIbase !".

Voir aussi SETMIDSTR, REPLACESTR.

15.12.7 INDEXSTR

INDEXSTR cherche la première occurrence d’une chaîne dans une autre.

`(INDEXSTR str substr)`

Recherche la première occurrence de la chaîne *substr* dans *str* avec une comparaison des chaînes sensible à la casse. Retourne un index (à partir de 0) de la première occurrence de la sous-chaîne dans *str* ou NIL si la sous-chaîne n’est pas présente. Si l’un des arguments est NIL alors le résultat est NIL.

Exemple: ‘`(INDEXSTR "Salut monde !" "monde")`’ retourne 6.

Voir aussi INDEXSTR*, RINDEXSTR, RINDEXSTR*, INDEXBRK, INDEXBRK*.

15.12.8 INDEXSTR*

INDEXSTR* a le même effet que INDEXSTR (voir [Section 15.12.7 \[INDEXSTR\]](#), page 102) sauf que la comparaison de chaîne n’est pas sensible à la casse.

Voir aussi INDEXSTR, RINDEXSTR, RINDEXSTR*, INDEXBRK, INDEXBRK*.

15.12.9 INDEXBRK

INDEXBRK cherche la première occurrence d'un caractère dans une chaîne.

`(INDEXBRK str listecar)`

Recherche la première occurrence d'un des caractères contenus dans *listecar* dans la chaîne *str* avec une comparaison sensible à la casse. Retourne l'index (à partir de 0) du premier caractère trouvé dans *str* ou NIL si aucun caractère n'est trouvé. Si l'un des arguments est NIL, le résultat est NIL.

Exemple: `'(INDEXBRK "Salut monde!" "aeiou")'` retourne 1.

Voir aussi INDEXBRK*, RINDEXBRK, RINDEXBRK*, INDEXSTR, INDEXSTR*.

15.12.10 INDEXBRK*

INDEXBRK* a le même effet que INDEXBRK (voir [Section 15.12.9 \[INDEXBRK\], page 103](#)) sauf que la comparaison de chaîne n'est pas sensible à la casse.

Voir aussi INDEXBRK, RINDEXBRK, RINDEXBRK*, INDEXSTR, INDEXSTR*.

15.12.11 RINDEXSTR

RINDEXSTR recherche la dernière occurrence d'une chaîne dans une autre.

`(RINDEXSTR str substr)`

Recherche la dernière occurrence de *substr* dans *str*, avec une comparaison de chaîne sensible à la casse. Retourne l'index (à partir de 0) de la sous-chaîne dans *str* ou NIL si la sous-chaîne n'est pas présente. Si l'un des arguments est NIL, le résultat est NIL.

Exemple: `'(RINDEXSTR "Doo itashimashite." "shi")'` returns 11.

Voir aussi RINDEXSTR*, INDEXSTR, INDEXSTR*, RINDEXBRK, RINDEXBRK*.

15.12.12 RINDEXSTR*

RINDEXSTR* a le même effet que RINDEXSTR (voir [Section 15.12.11 \[RINDEXSTR\], page 103](#)) sauf que la comparaison n'est pas sensible à la casse.

Voir aussi RINDEXSTR, INDEXSTR, INDEXSTR*, RINDEXBRK, RINDEXBRK*.

15.12.13 RINDEXBRK

RINDEXBRK recherche la dernière occurrence d'un caractère dans une chaîne.

`(RINDEXBRK str listecar)`

Recherche la dernière occurrence de l'un des caractères de *listecar* dans *str* avec une comparaison de chaîne sensible à la casse. Retourne l'index (à partir de 0) du dernier caractère trouvé dans *str* ou NIL si aucun caractère n'a été trouvé. Si l'un des arguments est NIL alors le résultat est NIL.

Exemple: `'(RINDEXBRK "Konnichiwa" "chk")'` retourne 6.

Voir aussi RINDEXBRK*, INDEXBRK, INDEXBRK*, RINDEXSTR, RINDEXSTR*.

15.12.14 RINDEXBRK*

RINDEXBRK* a le même effet que RINDEXBRK (voir [Section 15.12.13 \[RINDEXBRK\], page 103](#)) sauf que la comparaison n'est pas sensible à la casse.

Voir aussi RINDEXBRK, INDEXBRK, INDEXBRK*, RINDEXSTR, RINDEXSTR*.

15.12.15 REPLACESTR

REPLACESTR remplace des sous-chaînes par d'autres.

(REPLACESTR *str* [*recherché1 remplaçant1* ...])

Remplace toutes les occurrences de *recherché1* dans *str* par *remplaçant1* et poursuit par le remplacement de sous-chaînes dans la nouvelle chaîne en utilisant la paire suivante de chaînes recherchée et remplaçante jusqu'au traitement complet des arguments. Noter que le nombre d'arguments doit être impair et que les arguments aux positions paires spécifient des chaînes recherchées suivies de la chaîne de remplacement correspondante. Du fait que le résultat d'un remplacement est utilisé lors du remplacement suivant, des remplacements multiples peuvent être effectués, ce qui doit être pris en compte lors de l'utilisation de cette fonction. Un ordre différent des arguments peut aider la résolution de conflits puisque les remplacements sont effectués de gauches à droite.

Si l'une des chaînes est NIL ou que l'une des chaînes de recherche est vide, le résultat est NIL.

Exemple: '(REPLACESTR "noir c'est blanc" "noir" "blanc" "blanc "noir")' retourne "noir c'est noir".

Voir aussi REPLACESTR*, SETMIDSTR, INSMIDSTR, REMCHARS.

15.12.16 REPLACESTR*

REPLACESTR* a le même effet que REPLACESTR (voir [Section 15.12.15 \[REPLACESTR\]](#), [page 104](#)) sauf que la comparaison n'est pas sensible à la casse.

Voir aussi REPLACESTR, SETMIDSTR, INSMIDSTR, REMCHARS.

15.12.17 REMCHARS

REMCHARS supprime des caractères d'une chaîne.

(REMCHARS *str car-à-supprimer*)

Retourne une copie de *str* dans laquelle tous les caractères de *car-à-supprimer* ont été supprimés. Si *str* ou *car-à-supprimer* sont NIL, le résultat est NIL.

Exemple: '(REMCHARS *votre-chaîne* " \t\n")' supprime tous les espaces, tabulations et retour chariot de *votre-chaîne*.

Voir aussi REPLACESTR, TRIMSTR.

15.12.18 TRIMSTR

TRIMSTR supprime des caractères au début et à la fin d'une chaîne.

(TRIMSTR *str* [*début fin*])

Retourne une copie de *str* où les caractères de début et de fin ont été supprimés. Appelé avec un seul argument, les espaces, sauts de page, sauts de ligne, retours chariot et tabulations horizontales et verticales sont enlevés. Appelé avec trois arguments, *début* et *fin* spécifient respectivement les caractères à supprimer au début et à la fin de la chaîne. Il faut remarquer que TRIMSTR ne peut pas être appelé avec deux arguments.

Si l'un des arguments *str*, *début* ou *fin* est NIL alors le résultat est NIL.

Exemple: (TRIMSTR " J'ai cassé le vélo de Selma. ") retourne "J'ai cassé le vélo de Selma.", (TRIMSTR "007 " "0" " \f\n\r\t\v") retourne "7".

Voir aussi REMCHARS.

15.12.19 WORD

WORD retourne un mot contenu dans une chaîne.

(WORD *str num*)

Retourne le *num*ème mot (à partir de zéro) de la chaîne spécifiée. Les mots d'une chaîne sont des sous-chaînes non vides séparées par des caractères blancs (p. ex. espace, tabulation ou retour à la ligne).

Si *str* ou *num* sont NIL ou si *num* est hors limite, c'est à dire négatif ou supérieur au nombre de mots, alors le résultat est NIL.

Exemple: '(WORD "Du coup je prête mon vélo à Selma." 6)' retourne "vélo".

Voir aussi WORDS, LINE, LEFTSTR, RIGHTSTR, MIDSTR.

15.12.20 WORDS

WORDS compte le nombre de mots contenus dans une chaîne.

(WORDS *str*)

Retourne le nombre de mots contenus dans la chaîne spécifiée ou NIL si *str* est NIL. Les mots sont des sous-chaînes non vides séparées par des caractères blancs (p. ex. espace, tabulation ou retour à la ligne).

Exemple: '(WORDS "En fait, ce n'était pas vraiment mon vélo.")' retourne 8.

Voir aussi WORD, LINES, LEN.

15.12.21 STRTOLIST

STRTOLIST convertit une chaîne en une liste de sous-chaînes.

(STRTOLIST *str [sep]*)

Crée une liste de sous-chaînes en découpant la chaîne *str* au niveau des occurrences de la séquence de séparation *sep*. Si *sep* n'est pas spécifié alors le caractère tabulation "\t" est utilisé. Si *sep* est la chaîne vide "" alors la liste de tous les caractères contenus dans la chaîne est retournée.

Si *str* ou *sep* sont NIL alors NIL est retourné.

Exemples

'(STRTOLIST "J'aime\tle\tJapon.")' retourne ("J'aime" "le" "Japon.").

'(STRTOLIST "Nom|Rue|Ville" "|")' retourne ("Nom" "Rue" "Ville").

'(STRTOLIST "abc" "")' retourne ("a" "b" "c").

Voir aussi MEMOTOLIST, LISTTOSTR.

15.12.22 LISTTOSTR

LISTTOSTR convertit une liste d'éléments en une chaîne.

(LISTTOSTR *liste [sep]*)

Convertit la liste spécifiée d'éléments en une chaîne par concaténation des représentations textuelles de chaque élément de la liste séparées par la séquence *sep*. Si *sep* n'est pas spécifié alors le caractère tabulation "\t" est utilisé. Si *list* ou *sep* sont NIL alors NIL est retourné.

Exemples

‘(LISTTOSTR (LIST "Pierre a" 18 "ans"))’ retourne "Pierre a\t18\tans".

‘(LISTTOSTR (LIST "Nom" "Rue" "Ville") "|")’ retourne "Nom|Rue|Ville".

Voir aussi LISTTOMEMO, CONCAT, CONCAT2, STRTOLIST.

15.12.23 CONCAT

CONCAT concatène des chaînes.

```
(CONCAT [str ...])
```

Retourne la concaténation de la liste des chaînes donnée avec des espaces entre chacune. Si l’une des chaînes est NIL ou si la liste est vide, le résultat est NIL.

Exemple: ‘(CONCAT "Je" "pensais" "que" "c’était" "un" "vélo" "abandonné.")’ retourne "Je pensais que c’était un vélo abandonné."

Voir aussi CONCAT2, +, LISTTOSTR, COPYSTR, SPRINTF.

15.12.24 CONCAT2

CONCAT2 concatène des chaînes.

```
(CONCAT2 insert [str ...])
```

Retourne la concaténation de la liste des chaînes donnée avec la chaîne *insert* entre chacune. Si *insert* ou l’une des chaînes est NIL ou si la liste est vide, le résultat est NIL.

Exemple: ‘(CONCAT2 "!" "Mais" "ça" "ne" "l’était" "pas")’ retourne "Mais! ça! ne! l’était! pas!".

Voir aussi CONCAT, +, LISTTOSTR, COPYSTR, SPRINTF.

15.12.25 COPYSTR

COPYSTR crée des copies d’une chaîne.

```
(COPYSTR str num)
```

Retourne une chaîne constituée de *num* fois la chaîne *str*. Si *str* est NIL, *num* est NIL ou inférieur à zéro, le résultat est NIL.

Exemple: ‘(COPYSTR "+-" 5)’ retourne "+-+-+--+".

Voir aussi CONCAT, CONCAT2, +, SPRINTF.

15.12.26 SHA1SUM

SHA1SUM calcule la valeur de hachage SHA1 d’une chaîne.

```
(SHA1SUM str)
```

Retourne une chaîne contenant la valeur de hachage SHA1 de la chaîne spécifiée. Si *str* est NIL alors NIL est retourné.

Exemple: ‘(SHA1SUM "flower, sun and beach")’ retourne "47b6c496493c512b40e042337c128d85ecf15ba4"

Avoir aussi ADMINPASSWORD, demo ‘Users.mb’.

15.12.27 UPPER

UPPER convertit une chaîne en majuscules.

(UPPER *str*)

Retourne une copie de la chaîne donnée où tous les caractères ont été convertis en majuscules. Si *str* est NIL, le résultat est NIL.

Exemple: ‘(UPPER "Selma a trouvé une lettre attachée à mon vélo.")’ retourne "SELMA A TROUVÉ UNE LETTRE ATTACHÉE à MON VELO."

Voir aussi LOWER.

15.12.28 LOWER

LOWER convertit en minuscules une chaîne.

(LOWER *str*)

Retourne une copie de la chaîne donnée où tous les caractères ont été convertis en minuscules. Si *str* est NIL, le résultat est NIL.

Exemple: ‘(LOWER "La lettre était de Silke.")’ retourne "la lettre était de silke."

Voir aussi UPPER.

15.12.29 ASC

ASC convertit un caractère en sa représentation interne entière.

(ASC *str*)

Retourne le code entier interne du premier caractère de *str*. Sur Linux et Windows, il s’agit d’une représentation unicode. Sur Amiga, il s’agit du code entier 8 bits dans le jeu de caractères configuré dans le système. Si *str* est vide, le résultat est 0. Si *str* est NIL, le résultat est NIL.

Exemple: (ASC "A") retourne 65.

Voir aussi CHR, INT.

15.12.30 CHR

CHR convertit une valeur entière en caractère.

(CHR *entier*)

Retourne une chaîne contenant le caractère de code entier *entier*. Sur Linux et Windows *entier* est interprété comme un caractère unicode. Sur Amiga, *entier* est un entier 8 bits dans le jeu de caractères configuré dans le système. Si *entier* vaut 0, une chaîne vide est renvoyée. Si *entier* est NIL ou en dehors de l’intervalle des caractères valides, le résultat est NIL.

Exemple: ‘(CHR 67)’ retourne "C".

Voir aussi ASC, STR.

15.12.31 LIKE

LIKE compare des chaînes.

(LIKE *str1 str2*)

Retourne TRUE si *str1* correspond à *str2*, NIL sinon. La chaîne *str2* peut contenir les caractères jokers '?' (pour remplacer exactement un caractère) et '*' (pour remplacer une suite quelconque de caractères). La comparaison est sensible à la casse.

Exemple: '(LIKE "Silke est allé en France pendant un an." "*France*")' retourne TRUE.

Voir aussi Fonctions de comparaison.

15.12.32 SPRINTF

SPRINTF formate une chaîne à partir de diverses données.

(SPRINTF *fmt* [*expr* ...])

SPRINTF prend une série d'arguments, les convertit en chaîne et retourne l'information formatée dans une seule chaîne. La chaîne *fmt* détermine exactement ce qui est écrit dans la chaîne retournée et peut contenir deux types d'éléments : des caractères ordinaires qui sont toujours copiés tels quels et des spécificateurs de conversion qui indique à SPRINTF de prendre en compte des arguments depuis la liste et de les formater. Les spécificateurs de conversion débutent toujours par un caractère '%'.
 Les spécificateurs de conversion sont toujours de la forme :

%[drapeaux][largeur][.precision]type

où

- Le champ optionnel *drapeaux* contrôle la justification, le caractère de signe des valeurs numériques, le point décimal et les espaces de fin.
- Le champ optionnel *largeur* indique le nombre minimum de caractères à afficher (la largeur du champ), avec un bourrage par des blancs ou des zéros.
- Le champ optionnel *précision* spécifie soit le plus grand nombre de caractères à restituer pour les types chaîne, booléen, date et heure, soit le nombre de décimales à restituer après le séparateur décimal pour le type réel.
- Le champ *type* indique le type effectif de l'argument que SPRINTF va convertir : texte, entier, réel, etc.

Noter que tous les champs ci-dessus sont optionnels à l'exception de *type*. Les tableaux suivants listent les options valides pour ces champs.

Champ drapeaux

- | | |
|-----------------|--|
| -: | Le résultat est justifié à gauche, avec un bourrage par la droite avec des blancs. Par défaut lorsque '-' n'est pas spécifié, le résultat est justifié à droite avec un bourrage à gauche par des '0' ou des blancs. |
| +: | Le résultat possède toujours un caractère représentant son signe s'il s'agit d'une conversion numérique. |
| 0: | Pour les nombres justifiés à gauche, le remplissage est effectué avec des zéros au lieu d'espaces. |
| <i>espace</i> : | Les nombres positifs débutent par un espace à la place d'un caractère '+', en revanche les valeurs négatives sont toujours préfixées d'un '-'. |

Champ largeur

- n*: *n* au minimum sont affichés. Si la conversion dispose de moins de *n* caractères, un remplissage par des espaces ou des zéros non-significatifs est effectué.
- **: La valeur de la largeur est fournie dans la liste d'arguments en tant que valeur entière ou réelle, avant l'argument à convertir. Cette valeur est limitée à l'intervalle 0-999.

Champ précision

- .n*: Pour les valeurs de type texte, booléen, date et heure, *n* est le nombre maximum de caractères de l'élément converti à restituer. Pour les conversions de valeurs réelles, *n* spécifie le nombre de chiffres après la virgule (conversions 'f' et 'e') ou le nombre de chiffres significatifs (conversion 'g'). Dans les conversions d'entiers, ce champ est ignoré.
- .**: La valeur de la précision est fournie dans la liste des arguments sous forme de valeur entière ou réelle, avant l'argument à convertir. Cette valeur est limitée à l'intervalle 0-999.

Champ type

- b*: Convertit une valeur booléenne en "TRUE" ou "NIL".
- i*: Convertit une valeur entière en une notation décimale signée.
- o*: Convertit une valeur entière en une notation octale non signée.
- x*: Convertit une valeur entière en une notation hexadécimale non signée, en utilisant les lettres minuscules 'abcdef'.
- X*: Convertit une valeur entière en une notation hexadécimale non signée, en utilisant les lettres majuscules 'ABCDEF'.
- e*: Convertit une valeur réelle en utilisant le format [-]d.ddde+dd (NDT: c.-à-d. la notation scientifique). Exactement un chiffre avant le point décimal, suivi d'un 'e', suivi d'un exposant. Le nombre de chiffres après le point décimal est déterminé par le champ précision ou est de 2 si aucune précision n'a été spécifiée. Le point décimal n'apparaît pas si la précision est de 0.
- f*: Convertit une valeur réelle en utilisant le format [-]ddd.ddd. Le nombre de chiffres après le point décimal est déterminé par le champ précision ou est de 2 si aucune précision n'est spécifiée. Le point décimal n'apparaît pas si la précision est de 0.
- g*: Convertit une valeur réelle en utilisant le style 'e' ou 'f' en fonction du nombre de chiffres représentant la valeur. Les zéros terminaux sont supprimés de la sortie, à l'exception de celui qui suit directement le point décimal.
- s*: Ecrit la chaîne jusqu'à ce que soit la fin de la chaîne soit atteinte, ou jusqu'à ce que le nombre de caractères spécifié par le champ précision soit atteint.
- d*: Convertit une valeur de date.

t: Convertit une valeur horaire.

%%: Ecrit le caractère '%' sans qu'aucun argument ne soit converti.

SPRINTF retourne la chaîne formatée ou NIL si *fmt* est NIL.

Exemples

Appel	Résultat
(SPRINTF "Hello")	"Hello"
(SPRINTF "%s" "Hello")	"Hello"
(SPRINTF "%10s" "Hello")	" Hello"
(SPRINTF "%-10.10s" "Hello")	"Hello "
(SPRINTF "%010.3s" "Hello")	" Hello"
(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 "He%s %5.5s!" "llo"	
"world champion ship")	"Hello world!"

Voir aussi PRINTF, FPRINTF, STR, +, CONCAT, CONCAT2, COPYSTR.

15.13 Fonctions sur les mémos

Cette section traite des fonctions de manipulation des mémos.

15.13.1 LINE

LINE extrait une ligne d'un mémo.

(LINE *memo num*)

Retourne la *num*ème ligne (à partir de zéro) du mémo spécifié. La chaîne contenant la ligne ne contient pas de caractère de retour à la ligne à la fin. Si *memo* ou *num* sont NIL ou si *num* est hors limite, c.-à-d. négatif ou supérieur ou égal au nombre de lignes, le résultat est NIL.

Voir aussi LINES, WORD.

15.13.2 LINES

LINES retourne le nombre de lignes dans un mémo.

(LINES *memo*)

Retourne le nombre de lignes du mémo spécifié ou NIL si *memo* est NIL.

Voir aussi LINE, WORDS, LEN.

15.13.3 MEMOTOLIST

MEMOTOLIST convertit un mémo en une liste de chaînes.

```
(MEMOTOLIST memo [expandstr])
```

Convertit le mémo spécifié en liste. Si *memo* est NIL, alors le résultat est NIL, sinon, une liste est générée dans laquelle chaque élément contient une ligne du mémo.

Si *expandstr* est spécifié et n'est pas NIL alors la liste de chaînes retournée est ensuite traitée en appliquant STRTOLIST à chaque élément de la liste. Ce qui donne une liste de listes de chaînes.

Exemples

‘(MEMOTOLIST "Mon assurance\nprend en charge\nle vélo cassé.")’ retourne ("Mon assurance" "prend en charge" "le vélo cassé.").

‘(MEMOTOLIST "Voici\tun exemple\nmulti-colonnes." TRUE)’ retourne (("Voici" "un exemple") ("multi-colonnes.")).

Voir aussi STRTOLIST, LISTTOMEMO.

15.13.4 LISTTOMEMO

LISTTOMEMO convertit une liste en mémo.

```
(LISTTOMEMO list)
```

Convertit la liste spécifiée en mémo. Si *list* est NIL, alors le résultat est NIL, sinon un mémo est généré où chaque ligne est constituée de la conversion textuelle de l'élément correspondant de la liste. Si un élément de la liste est une sous-liste, alors LISTTOSTR (voir [Section 15.12.22 \[LISTTOSTR\], page 105](#)) lui est appliqué avant d'intégrer la chaîne résultat dans le mémo.

Exemples

‘(LISTTOMEMO (LIST "Silke" "me prête" " mon vélo" "jusqu'au" 01.09.1998))’ retourne "Silke\nme prête\n mon vélo\njusqu'au\n01.09.1998".

‘(LISTTOMEMO (LIST (LIST "Nom" "Date naissance") (LIST "Steffen" 28.11.1968)))’ retourne "Nom\tDate naissance\nSteffen\t28.11.1968".

Voir aussi LISTTOSTR, MEMOTOLIST.

15.13.5 FILLMEMO

FILLMEMO remplit un mémo avec les résultats d'expressions.

```
(FILLMEMO memo)
```

Crée une copie du mémo spécifié où toutes les sous-chaînes de la forme ‘\$(*expr*)’ sont remplacées par leur résultat après leur évaluation.

Exemple: ‘(FILLMEMO "(+ 1 1) donne \$(+ 1 1).")’ retourne "(+ 1 1) donne 2."

Le déboguage et l'identification n'étant pas aisée ici, il est préférable de n'utiliser que des expressions simples dans le mémo.

Voir aussi FORMATMEMO, INDENTMEMO.

15.13.6 FORMATMEMO

FORMATMEMO formate un mémo.

(FORMATMEMO *memo longueur [bourrage [secmonoligne]]*)

Formate *memo* de manière à ce qu'il devienne un mémo avec des lignes ayant moins de *longueur* caractères. Si *bourrage* n'est pas NIL des espaces de bourrage sont utilisés pour remplir les lignes jusqu'à *longueur* caractères. Le mémo est traité section par section, une section débute au premier caractère non blanc. Si *secmonoligne* est spécifié et différent de NIL alors tous les caractères jusqu'à la fin de cette ligne sont considérés comme faisant partie de la section. Sinon tous les caractères de cette ligne et des suivantes sont considérés dans la section jusqu'à rencontrer une ligne débutant par un caractère blanc. La section est ensuite formatée mot à mot, c'est à dire qu'autant de mots que possible sont mis dans une ligne. Les mots restants sont insérés dans la ligne suivante etc.

Voir aussi FILLMEMO, INDENTMEMO.

15.13.7 INDENTMEMO

INDENTMEMO indente un mémo en mettant des espaces à gauche.

(INDENTMEMO *memo indentation*)

Retourne une copie du mémo spécifié où chaque ligne est indentée par *indentation* caractères espace. Si *memo* ou *indentation* sont NIL, alors le résultat est NIL. Si *indentation* est négatif, une valeur de 0 est utilisée.

Voir aussi FILLMEMO, FORMATMEMO.

15.14 Fonctions sur la date et l'heure

Cette section concerne les fonctions de manipulation de valeurs de type date et heure.

15.14.1 DAY

DAY extrait le jour d'une date.

(DAY *date*)

Renvoie un nombre entier représentant le jour de la date spécifiée. Si *date* est NIL alors NIL est retourné.

Voir aussi MONTH, YEAR, DATEDMY.

15.14.2 MONTH

MONTH extrait le mois d'une date.

(MONTH *date*)

Renvoie un nombre entier représentant le mois de la date spécifiée. Si *date* est NIL alors NIL est retourné.

Voir aussi DAY, YEAR, DATEDMY, MONTHDAYS.

15.14.3 YEAR

YEAR extrait l'année d'une date.

(YEAR *date*)

Renvoie un nombre entier représentant l'année de la date spécifiée. Si *date* est NIL alors NIL est retourné.

Voir aussi DAY, MONTH, DATEDMY, YEARDAYS.

15.14.4 DATEDMY

DATEDMY crée une date à partir du jour, du mois et de l'année.

(DATEDMY *jour mois année*)

Crée une date à partir des données du jour, du mois et de l'année. Si *jour*, *mois* ou *année* sont NIL, hors limite ou si le résultat de la date n'est pas valide, alors NIL est retourné.

Exemple: '(DATEDMY 28 11 1968)' donne le 28 novembre 1968.

Voir aussi DATE, TODAY, DAY, MONTH, YEAR.

15.14.5 MONTHDAYS

MONTHDAYS donne le nombre de jours d'un mois.

(MONTHDAYS *mois année*)

Renvoie le nombre de jours à partir des données du mois et de l'année. Si *mois* est NIL ou hors limite (inférieur à 1 ou supérieur à 12) alors NIL est retourné. Si *année* est NIL alors une année non bissextile est supposée pour calculer le nombre de jours. Si *année* est invalide (inférieure à 0 ou supérieure à 9999) alors NIL est retourné.

Exemples: '(MONTHDAYS 2 2004)' donne 29, '(MONTHDAYS 2 NIL)' donne 28.

Voir aussi YEARDAYS, MONTH.

15.14.6 YEARDAYS

YEARDAYS donne le nombre de jours d'une année.

(YEARDAYS *année*)

Renvoie le nombre de jours contenus dans l'année spécifiée. Si *année* est NIL ou hors limite (inférieure à 0 ou supérieure à 9999) alors NIL est retourné.

Exemples: '(YEARDAYS 2004)' donne 366, '(YEARDAYS 2005)' donne 365.

Voir aussi MONTHDAYS, YEAR.

15.14.7 ADDMONTH

ADDMONTH ajoute un certain nombre de mois à une date.

(ADDMONTH *date mois*)

Renvoie une valeur date où le nombre de mois spécifié a été ajouté à la date initiale. Les valeurs négatives pour *mois* soustraient les mois. Si *date* ou *mois* sont NIL ou si le résultat de la date est invalide, alors NIL est retourné.

ADDMONTH gère le débordement ou l'épuisement du champ mois en ajustant l'année en conséquence. Dans le cas où le champ jour dépasserait le nombre maximum de jours pour le mois résultant, il est décrémenté au nombre maximum de jours autorisé pour ce mois.

Exemples: '(ADDMONTH 30.01.2004 1)' donne 29.02.2004, '(ADDMONTH 30.01.2004 -1)' donne 30.12.2003.

Voir aussi ADDYEAR, +.

15.14.8 ADDYEAR

ADDDYEAR ajoute un certain nombre d'années à une date.

(ADDDYEAR *date années*)

Renvoie une valeur date où le nombre d'années spécifié a été ajouté à la date initiale. Les valeurs négatives pour *années* soustraient des années. Si *date* ou *années* sont NIL ou si le résultat de la date est invalide, alors NIL est retourné.

ADDDYEAR décrémente le jour de 1 au cas où la *date* représenterait le 29 février et si l'année résultante n'est pas bissextile.

Exemples: '(ADDDYEAR 29.02.2004 1)' donne 28.02.2005, '(ADDDMONTH 04.02.2004 -1962)' donne 04.02.0042.

Voir aussi ADDMONTH, +.

15.14.9 TODAY

TODAY renvoie la date du jour.

(TODAY)

Renvoie la date du jour en tant que valeur date.

Voir aussi NOW, DATEDMY.

15.14.10 NOW

NOW renvoie l'heure courante.

(NOW)

Renvoie l'heure courante en tant que valeur de l'heure.

Voir aussi TODAY.

15.15 Liste des Fonctions

Cette section énumère les fonctions pour le traitement des listes.

15.15.1 CONS

CONS établit une paire d'expressions.

(CONS *elem list*)

Construit une nouvelle liste. Le premier élément de la nouvelle liste est *elem*, le reste sont les éléments de la *list* (qui devraient être une liste ou NIL). La liste *list* n'est pas copiée, seul un pointeur est utilisé pour la référencer !

Exemple: '(CONS 1 (CONS 2 NIL))' donne (1 2).

Les éléments d'une liste peuvent être de n'importe quel type, par exemple il est également possible d'avoir une liste de listes (par exemple voir voir [Section 15.20.12 \[SELECT\], page 135](#)). Le constructeur CONS peut également être employé pour établir des paires d'éléments, par exemple '(CONS 1 2)' est la paire avec les deux nombres entiers 1 et 2.

Voir aussi LIST, FIRST, REST.

15.15.2 LIST

LIST produit une liste en dehors de ses arguments.

(LIST [*elem* ...])

Prend les arguments *elem* ... pour produire une liste. Ce qui équivaut à appeler (CONS *elem* (CONS ... NIL)). Notez que NIL reste seul pour une liste vide.

Voir aussi CONS, LENGTH.

15.15.3 LENGTH

LENGTH détermine la longueur d'une liste.

(LENGTH *list*)

renvoie la longueur des données de la liste.

Exemple: '(LENGTH (LIST "a" 2 42 3))' donne 4.

Voir aussi LIST.

15.15.4 FIRST

FIRST extrait le premier élément dans une liste.

(FIRST *list*)

renvoie le premier élément des données de la liste. Si *list* est vide (NIL) alors NIL est retourné.

Voir aussi REST, LAST, NTH, CONS.

15.15.5 REST

REST renvoie la sous-liste après le premier élément d'une liste.

(REST *list*)

renvoie le reste des données de la liste (la liste sans premier élément). Si *list* est vide (NIL) alors NIL est retourné.

Exemple: '(REST (LIST 1 2 3))' donne (2 3).

Voir aussi FIRST, CONS.

15.15.6 LAST

LAST extrait le dernier élément dans une liste.

(LAST *list*)

Renvoie le dernier élément des données de la liste ou NIL si *list* est NIL.

Voir aussi FIRST, NTH.

15.15.7 NTH

NTH extrait le N ième élément d'une liste.

(NTH *n list*)

Renvoie *n*-ième élément des données de la liste (commençant par 0) ou NIL si l'élément n'existe pas.

Voir aussi FIRST, LAST.

15.15.8 APPEND

APPEND enchaîner les listes.

```
(APPEND [list ...])
```

renvoie la concaténation de *list*

Exemple: ‘(APPEND (list 1 2) (list 3 4) (list 5))’ donne (1 2 3 4 5).

Voir aussi LIST.

15.15.9 REVERSE

REVERSE renverse une liste.

```
(REVERSE list)
```

renvoie la liste renversée.

Exemple: ‘(REVERSE (list 1 2 3))’ donne (3 2 1).

15.15.10 MAPFIRST

MAPFIRST applique une fonction à tous les éléments de la liste.

```
(MAPFIRST func list [...])
```

Établit une liste dont les éléments sont le résultat de l’application d’une fonction appelant un par un les arguments des éléments d’une liste donnée. La longueur de la liste retournée est la longueur de la plus longue liste indiquée. Si une des listes indiquées est trop courte alors la liste est rallongée par des éléments NIL.

Exemples

Expression	Value
(MAPFIRST 1+ (LIST 1 2 3))	(2 3 4)
(MAPFIRST + (LIST 1 2 3) (LIST 2 3))	(3 5 NIL)

15.15.11 SORTLIST

SORTLIST Trie les éléments d’une liste.

```
(SORTLIST func list)
```

Renvoie une copie de la liste indiquée qui a été triée en utilisant la fonction *func* pour le tri. La fonction pour le tri doit prendre deux arguments un pour chaque élément et renvoyer une valeur de nombre entier inférieure à zéro si le premier élément est plus petit que le second, une valeur supérieure à zéro si le premier élément est plus grand que le second, une valeur de zéro si les deux éléments sont égaux.

Exemple pour une chaîne texte comparant la fonction utilisable pour le trie :

```
(DEFUN cmp_str (x y)
  (COND
    ((< x y) -1)
    ((> x y) 1)
    (TRUE 0)
```

```
)
)
```

Maintenant vous pouvez trier une liste en appelant :

```
(SORTLIST cmp_str (LIST "salut" "excellent" "grand" "ok"))
```

qui retourne ("excellent" "grand" "ok" "salut").

Voir aussi SORTLISTGT, MAPFIRST.

15.15.12 SORTLISTGT

SORTLIST tri les éléments de la liste

```
(SORTLISTGT gtfunc list)
```

Comme SORTLIST mais ici vous indiquez une fonction de tri qui renvoie une valeur différente de NIL si le premier et le second élément sont supérieur, et NIL autrement.

Exemple: ‘(SORTLISTGT > (LIST "salut" "excellent" "grand" "ok"))’ donne ("excellent" "grand" "salut" "ok").

Voir aussi SORTLIST, MAPFIRST.

15.16 Fonctions de demande de saisie

Pour demander des informations à l'utilisateur, les fonctions suivantes peuvent être utilisées :

15.16.1 ASKFILE

ASKFILE demande à l'utilisateur d'écrire un nom de fichier.

```
(ASKFILE title oktext default savemode)
```

Ouvre un sélecteur de fichier pour saisir le nom de fichier. Le titre de la fenêtre peut être placé dans *title*, le texte du bouton ‘Ok’ dans *oktext*, et le nom du fichier initial dans *default*. Vous pouvez spécifier NIL pour l'un d'entre eux afin d'utiliser des valeurs par défauts. Le dernier argument *savemode* (booléen) permet d'ouvrir le sélecteur de fichier en mode sauvegarde. Ce mode devrait être utilisé pour demander de sauver quelque chose sous un nom de fichier.

ASKFILE renvoie le nom de fichier écrit dans une chaîne texte ou NIL dans le cas où l'utilisateur aurait annulé le sélecteur de fichier.

Voir aussi ASKDIR, ASKSTR.

15.16.2 ASKDIR

ASKDIR demande à l'utilisateur d'écrire le nom d'un tiroir.

```
(ASKDIR title oktext default savemode)
```

Ouvre le sélecteur de fichier pour saisir le nom du tiroir. Les arguments sont utilisés de la même façon ASKFILE (voir [Section 15.16.1 \[ASKFILE\]](#), page 117).

ASKDIR renvoie le nom du tiroir écrit dans une chaîne texte ou NIL dans le cas où l'utilisateur aurait annulé le sélecteur de fichier.

Voir aussi ASKFILE, ASKSTR.

15.16.3 ASKSTR

ASKSTR demande à l'utilisateur d'écrire une chaîne texte.

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

Ouvre une requête de saisie de chaîne texte. Le titre de la fenêtre, le texte du bouton 'Ok', et la valeur initiale peuvent être placés dans *title*, *oktext*, et *default* respectivement (chaîne texte ou NIL pour des valeurs par défaut), *maxlen* détermine le nombre maximum de caractères que l'utilisateur peut écrire. Si *secret* est spécifié et n'est pas NIL alors la chaîne saisie est rendue invisible en affichant un symbole rond pour chaque caractère.

ASKSTR renvoie la chaîne texte entrée ou NIL au cas où l'utilisateur l'annulerait.

Voir aussi ASKFILE, ASKDIR, ASKCHOICESTR, ASKINT.

15.16.4 ASKINT

ASKINT incite l'utilisateur à écrire un nombre entier.

```
(ASKINT title oktext default min max)
```

Ouvre une requête de saisie de nombre entier. Le titre de fenêtre et le texte du bouton 'Ok' peuvent être spécifiés dans *title* et *oktext* (chaîne texte ou NIL pour des valeurs par défaut). Dans *default* vous passez la valeur initiale du nombre entier ou NIL pour démarrer avec un champ d'édition vide. Dans *min* et *max* vous pouvez placer la gamme des nombres entiers. Les valeurs entrées en dehors de cet intervalle sont automatiquement rejetées. Utilisez NIL pour les valeurs par défaut de min et max.

ASKINT renvoie le nombre entier saisi ou NIL si l'utilisateur annule la requête.

Voir aussi ASKSTR.

15.16.5 ASKCHOICE

ASKCHOICE Invite l'utilisateur à choisir un élément parmi d'autres.

```
(ASKCHOICE titre oktext choix default [titres])
```

Ouvre une requête de saisie demandant à l'utilisateur de choisir un élément dans une liste. Vous pouvez placer le titre de la fenêtre et le texte du bouton 'Ok' dans *titre* et *oktext* (chaîne texte ou NIL pour des valeurs par défaut). Dans *choix* vous indiquez une liste de choix sous forme de mémo avec un choix par ligne. Il est possible d'utiliser un format multi-colonnes en séparant les colonnes par le caractère tabulation "\t". La valeur initiale peut être placée dans *default*, il s'agit de l'index de la ligne dans le mémo (en commençant par l'index 0 pour le premier élément). Utilisez NIL pour ne pas spécifier de valeur initiale. Si l'argument optionnel *titres* est spécifié et n'est pas NIL, alors un entête de colonne est affiché dans la liste en utilisant les titres contenus dans *titres*. Dans le cas de colonnes multiples, séparez les titres de colonnes par des tabulations.

choix et *titres* peuvent tous les deux être représentés par des listes plutôt que par un mémo ou une chaîne. Dans ce cas ils sont convertis au format voulu par un appel automatique à LISTTOMEMO (voir [Section 15.13.4 \[LISTTOMEMO\]](#), [page 111](#)) ou LISTTOSTR (voir [Section 15.12.22 \[LISTTOSTR\]](#), [page 105](#)) respectivement.

ASKCHOICE renvoie l'index de l'article choisi ou NIL si l'utilisateur a annulé la requête.

Exemple

```
(LET ((items (LIST "Première entrée" 2 3.14 "Dernière entrée"))) index)
  (SETQ index (ASKCHOICE "Choisissez un article" "Ok" items NIL))
  (IF index
    (PRINTF "l'utilisateur a choisi l'article numérique %i avec le contenu <%s>\n"
      index (STR (NTH index items)))
    )
  )
)
```

Considérez le cas où vous voulez demander à l'utilisateur de sélectionner un enregistrement particulier dans une table. La table doit se nommer 'Article' et posséder les champs 'Nom', 'Quantite', et 'Prix'. Le bout de code suivant montre comment utiliser ASKCHOICE pour sélectionner un enregistrement avec un prix inférieur à 10 et ordonnés par leur Nom :

```
(LET ((requete (SELECT Article, Nome, Quantite, Prx from Article
  WHERE (> Prix 10) ORDER BY Nom)))
  (enregs (MAPFIRST FIRST (REST query))) ; pointeurs d'enregistrement
  (elements (MAPFIRST REST (REST query))) ; choic
  (titres (REST (FIRST query))) ; titres
  (index (ASKCHOICE "Sélectionnez" "Ok" items NIL titles))
  (elem (NTH index enregs)))
  ; maintenant elem contient l'enregistrement choisi (ou NIL si annulation)
)
```

Voir aussi ASKCHOICESTR, ASKOPTIONS.

15.16.6 ASKCHOICESTR

ASKCHOICESTR Invite l'utilisateur à entrer la valeur d'une chaîne texte parmi celles qui ont été prédéfinies.

```
(ASKCHOICESTR titre oktext chaines default [titres])
```

Ouvre une requête de saisie demandant à l'utilisateur de choisir une chaîne texte dans une liste ou d'en saisir une dans un champ texte séparé. Vous pouvez placer le titre de fenêtre et le texte du bouton 'Ok' dans *titre* et *oktext* (chaîne texte ou NIL pour des valeurs par défaut). Dans *chaines* vous indiquez la liste de choix sous forme de mémo avec un choix par ligne. Vous pouvez utiliser un format multi-colonnes en séparant les colonnes par le caractère tabulation "\t". La valeur initiale du champ texte peut être placée dans *default* (chaîne texte ou NIL pour un champ texte vide). Si l'argument optionnel *titres* est spécifié et n'est pas NIL, alors un entête de colonne est affiché dans la liste en utilisant les titres contenus dans *titres*. Dans le cas de colonnes multiples, séparez les titres de colonnes par des tabulations.

chaines et *titres* peuvent tous les deux être représentés par des listes plutôt que par un mémo ou une chaîne. Dans ce cas ils sont convertis au format voulu par un appel automatique à LISTTOMEMO (voir [Section 15.13.4 \[LISTTOMEMO\], page 111](#)) ou LISTTOSTR (voir [Section 15.12.22 \[LISTTOSTR\], page 105](#)) respectivement.

ASKCHOICESTR renvoie la chaîne texte choisie ou NIL si l'utilisateur a fermé la fenêtre.

Exemple

```
(LET ((chaines (LIST "Claudia" "Mats" "Ralphie"))) prefere)
  (SETQ prefere
    (ASKCHOICESTR "Qui préférez-vous?" "Ok" chaines
      "Mes Colley !")
  )
  )
  (IF prefere (PRINTF "l'utilisateur a choisi <%s>\n" prefere))
)
```

Voir aussi ASKCHOICE, ASKOPTIONS.

15.16.7 ASKOPTIONS

ASKOPTIONS Invite l'utilisateur à sélectionner plusieurs éléments dans une liste.

```
(ASKOPTIONS titre oktext options selectionnes [titres])
```

Ouvre une requête de saisie demandant à l'utilisateur de sélectionner une ou plusieurs options dans une liste. Vous pouvez placer le titre de la fenêtre et le texte du bouton 'Ok' dans *titre* et *oktext* (chaîne texte ou NIL pour des valeurs par défaut). Dans *options* vous indiquez la liste d'options sous forme de mémo avec une option par ligne. Vous pouvez utiliser un format multi-colonnes en séparant les colonnes par le caractère tabulation "\t". L'état initial peut être spécifié dans *selectionnes* sous forme de liste de nombres entiers spécifiant l'index dont la ligne correspondante dans *options* qui doit être sélectionnée au début. Utilisez NIL pour tous les éléments ne devant pas être présélectionnés.

Si l'argument optionnel *titres* est spécifié et n'est pas NIL, alors un entête de colonne est affiché dans la liste en utilisant les titres contenus dans *titres*. Dans le cas de colonnes multiples, séparez les titres de colonnes par des tabulations.

options et *titres* peuvent tous les deux être représentés par des listes plutôt que par un mémo ou une chaîne. Dans ce cas ils sont convertis au format voulu par un appel automatique à LISTTOMEMO (voir [Section 15.13.4 \[LISTTOMEMO\], page 111](#)) ou LISTTOSTR (voir [Section 15.12.22 \[LISTTOSTR\], page 105](#)) respectivement.

ASKOPTIONS renvoie une liste de nombres entiers correspondant à l'index des éléments sélectionnés ou NIL dans le cas où l'utilisateur aurait annulé la requête ou n'aurait pas choisi d'élément.

Exemple

```
(LET ((options (LIST "Salva Mea" "Insomnie" "n'attendez pas" "7 jours & 1 semaine")))
  (choisis (LIST 0 1 3))
)
(SETQ choisis (ASKOPTIONS "Titre de musique choisi" "Ok" options choisis))
(IF choisis
  (
    (PRINTF "L'utilisateur a choisi les articles suivants :\n")
    (DOLIST (i choisis)
      (PRINTF "\t num: %i contenu : <%s>\n" i (STR (NTH i options)))
    )
  )
)
```

```
)
)
```

15.16.8 ASKBUTTON

ASKBUTTON invite l'utilisateur à appuyer sur un bouton.

```
(ASKBUTTON title text buttons canceltext)
```

Ouvre une fenêtre de dialogue avec le titre (chaîne texte ou NIL pour un titre par défaut) et une description (chaîne texte ou NIL s'il n'y en a pas). La fonction attend jusqu'à ce que l'utilisateur appuie sur un des boutons indiqués dans *buttons* (liste de chaînes) ou le bouton 'Annuler'. Le texte du bouton d'annulation peut être spécifié dans *canceltext*. Si ici vous indiquez NIL un texte par défaut basé sur le nombre de boutons que vous avez indiqué est employé.

ASKBUTTON renvoie le numéro du bouton sélectionné (à partir de 0, pour le bouton à l'extrême gauche) ou NIL si l'utilisateur appuie sur le bouton 'Annuler'.

Exemples

```
(LET ((boutons (LIST "à la maison" "Dans le lit" "Devant mon ordinateur"))) index)
  (SETQ index (ASKBUTTON "répondez s'il vous plaît :"
    "Où vous voulez être demain ?" boutons "ne sait pas"))
  )
  (IF index
    (PRINTF "l'utilisateur a choisi : <%s>\n" (NTH index boutons))
    )
  )
```

```
(ASKBUTTON "Info" "MUIbase est grand !" NIL NIL)
```

Voir aussi ASKCHOICE.

15.16.9 ASKMULTI

ASKMULTI incite l'utilisateur à écrire divers genres d'informations.

```
(ASKMULTI title oktext itemlist)
```

ASKMULTI est une requête de saisie à usages multiples. Il ouvre une requête avec le titre indiqué, un ensemble d'objets GUI pour l'édition des données, et deux boutons ('Ok' et 'Annuler') pour fermer la requête. Le texte du bouton 'Ok' peut être placé dans *oktext* (chaîne texte ou NIL pour le texte de défaut). L'ensemble des objets GUI sont spécifiés dans *itemlist* qui est une liste où chaque élément a l'une des formes suivantes :

```
(LIST title "String" initial [help [secret]]) pour éditer une ligne de texte,
(LIST title "Memo" initial [help])           pour éditer plusieurs lignes de texte,
(LIST title "Integer" initial [help])         pour éditer un nombre entier,
(LIST title "Real" initial [help])             pour éditer un réel,
(LIST title "Date" initial [help])            pour éditer une date,
(LIST title "Time" initial [help])            pour éditer une heure,
(LIST title "Bool" initial [help])            pour un champ booléen,
(LIST title "Choice" initial
  (LIST choice ...) [help])
```

```

)                                pour un champ choix.
(LIST title "ChoiceList" initial
  (LIST choice ...) [help]
)                                pour choisir un élément dans une liste.
(LIST title "Options" initial
  (LIST option ...) [help]
)                                pour choisir plusieurs éléments dans une liste.
non-list-expr                    pour le texte statique

```

Le titre (chaîne texte ou NIL s'il n'y a pas de titre) sera affiché à la gauche de l'objet GUI. Si la valeur initiale est NIL la valeur par défaut est utilisée (par exemple un champ texte vide). Pour le choix de la valeur initiale des champs, l'index doit être (démarrer avec 0) l'entrée active initiale, parce que la valeur initiale d'une liste de champs peut être NIL (aucun article n'est activé), et pour les options des champs la valeur initiale doit être une liste de nombres entiers représentant les index (démarrant avec 0) des articles qui ont initialement été sélectionnés. Le champ facultatif d'aide (chaîne texte) peut être utilisé pour fournir plus d'information à l'utilisateur au sujet de l'utilisation du champ. Pour les champs texte un paramètre additionnel 'secret' peut être spécifié. S'il n'est pas NIL alors le contenu du champ texte est rendu invisible en affichant un symbole rond pour chacun de ses caractères.

ASKMULTI renvoie une liste de résultats que l'utilisateur a édités et validés en appuyant sur le bouton 'Ok'. Chaque valeur de résultat d'un champ a le même format que celui de la valeur initiale, par exemple pour le choix d'une liste de champs la valeur résultat est l'index de l'article choisi (ou NIL si aucun article n'a été choisi) ou pour une option de champ la valeur résultat est une liste de nombres entiers représentant les index des articles choisis. Pour le texte statique une valeur NIL est retournée.

Par exemple si vous avez indiqué un champ date, un champ texte statique, le choix d'un champ texte, une option de champ et une chaîne champ de texte avec la valeur initiale "world", et que l'utilisateur a écrit 11.11.1999, choisi l'entrée avec l'index numéro 2, choisi le 3ème et 4ème article dans l'option champ, et à gauche laissé le champ de la chaîne texte intact alors la fonction renvoie la liste (11.11.1999 NIL 2 (3 4) "world").

Si l'utilisateur a annulé la requête, NIL est retourné.

Exemple

```

(ASKMULTI "Veuillez répondre :" NIL (LIST
  (LIST "N_om" "String" "")
  (LIST "_Naissance" "Date" NIL)
  (LIST "_Sexe" "Choice" 0 (LIST "homme" "femme"))
  (LIST "Possède _voiture ?" "Bool" NIL)
  (LIST "_Aime" "Options" (LIST 0 2)
    (LIST "Bière" "Vin" "Whisky" "Wodka" "Schnaps"))
  ))
)

```

Veuillez regarder également le projet 'AskDemo.mb' pour d'autres exemples.

15.17 Fonctions d'entrée/sortie

Cette section liste les fonctions et les variables d'entrées/sortie (par exemple l'impression) des données.

15.17.1 FOPEN

FOPEN ouvre un fichier en lecture/écriture.

`(FOPEN nomfichier mode [encodage])`

Ouvre le fichier spécifié par *nomfichier* (string). Le paramètre *mode* (string) contrôle le mode d'accès. Utilisez `"w"` pour ouvrir un fichier en écriture, `"a"` pour y ajouter des données ou `"r"` pour y lire des données. Vous pouvez aussi utiliser d'autres drapeaux (ou combinaison de drapeaux) comme `"r+"` pour lire et écrire. Aucun contrôle de validité des drapeaux n'est effectué. Si le fichier ne peut être ouvert, l'instruction renvoie NIL.

Le paramètre optionnel *encodage* contrôle l'encodage du fichier et peut prendre l'une des chaînes suivantes pour valeur~:

- `"none"`: Aucune interprétation de caractère n'est effectuée. Utilisez ceci pour les fichiers binaires.
- `"UTF-8"`: Le texte est encodé en UTF-8. La lecture et l'écriture convertissent depuis/en UTF-8.
- `"locale"`:
Le texte est encodé selon les réglages de votre système. Sur Linux, c'est le contenu des variables d'environnement `LANG` et `LC_*`, voir `man locale`. Sur Windows, c'est la page de code du système. Sur Amiga, c'est l'encodage à 8-bit par défaut.
- `"8-bit"`: Le texte est encodé selon l'encodage à 8-bit par défaut. Sur Linux et Windows, il s'agit de l'encodage ISO-8859-1 (latin 1). Sur Amiga, il s'agit de l'encodage à 8-bit par défaut du système (comme `'locale'`).
- `"auto"`: L'encodage est détecté automatiquement. Si le fichier est lisible alors l'encodage est déterminé comme suit: Si l'intégralité du contenu est conforme à UTF-8 alors `"UTF-8"` est utilisé. Sinon, si la locale du système n'est pas UTF-8 alors `"locale"` est utilisé. Autrement, c'est `"8-bit"` qui est utilisé. Lors de l'écriture, si l'encodage n'a pas encore été déterminé, alors la première locale du système rencontrée est essayée. Si aucune erreur de conversion ne se produit, c'est `"locale"` qui est utilisé, sinon `"UTF-8"`.

Si aucun paramètre *encodage* n'est spécifié, c'est `"auto"` qui est utilisé.

En cas de succès, FOPEN renvoie un descripteur de fichier. En cas d'échec, c'est NIL qui est renvoyé. Si *nomfichier*, *mode* ou *encodage* sont à NIL, c'est NIL qui est renvoyé.

Exemples

`'(FOPEN "index.html" "w" "utf-8")'` ouvre et renvoi un descripteur de fichier pour écrire dans le fichier `'index.html'` selon l'encodage UTF-8.

`'(FOPEN "output.txt" "a+")'` ouvre le fichier `'output.txt'` pour y ajouter des données dans le même encodage que celui du fichier. Remarquez que si vous spécifiez uniquement

"a" pour mode, MUIbase peut ne pas être en mesure de lire le fichier et de déterminer son encodage. Dans ce cas, l'encodage sera déterminé au moment de l'écriture (et pourrait être différent de celui de la première partie du fichier).

Voir aussi FCLOSE, stdout, FFLUSH.

15.17.2 FCLOSE

FCLOSE ferme un fichier.

(FCLOSE *file*)

Ferme le fichier donné. Renvoie 0 en cas de succès, NIL si une erreur s'est produite. Si *file* est NIL alors 0 est retourné (aucune erreur). Après la fermeture du fichier l'accès au descripteur de fichier est une opération illégale et interrompt l'exécution du programme avec un message d'erreur.

Voir aussi FOPEN, FFLUSH.

15.17.3 stdout

La variable globale `stdout` conserve le descripteur de fichier vers la sortie standard de MUIbase. Le nom du fichier de sortie peut être réglé depuis le menu 'Programme - fichier de sortie' (voir [Section 7.5.9 \[Program output file\]](#), page 39).

Le fichier de sortie est ouvert au premier accès à cette variable (par exemple en appelant '(FPRINTF stdout ...)') ou '(PRINTF ...)'. Le fichier n'est ouvert automatiquement au démarrage de MUIbase, afin d'éviter une ouverture si aucune sortie n'est générée. Par exemple lorsque vous voulez seulement effectuer quelques calculs et changer le contenu de certains enregistrements.

Lors de l'ouverture du fichier de sortie, le paramètre de mode peut être soit "w", soit "a+" en fonction du réglage 'Ajouter' du menu 'Programme - Fichier de sortie'. L'encodage est réglé à "auto".

Si MUIbase ne peut pas ouvrir le fichier de sortie le programme s'arrête et un message d'erreur est produit.

Voir aussi FOPEN, PRINTF.

15.17.4 PRINT

PRINT converti une expression en chaîne et l'affiche.

(PRINT *elem*)

Convertit la valeur de *elem* en chaîne lisible et l'imprime dans `stdout`. Cette fonction existe principalement pour le débogage (la mise au point).

Voir aussi PRINTF, stdout.

15.17.5 PRINTF

PRINTF imprime une chaîne formater.

(PRINTF *format* [*expr* ...])

Formate une chaîne texte en utilisant les données du format chaîne et arguments pour l'imprimer dans `stdout`. Le formatage est fait comme dans `SPRINTF` (voir [Section 15.12.32 \[SPRINTF\]](#), page 108).

`PRINTF` renvoie le nombre de caractères en sortie ou `NIL` en cas d'échec. If *format* is `NIL` then `NIL` is returned.

Exemple: `'(PRINTF "%i jours et %i semaine" 7 1)'` imprime la chaîne texte "7 jours et 1 semaine" dans `stdout` et renvoie 17.

Voir aussi `PRINT`, `FPRINTF`, `stdout`.

15.17.6 FPRINTF

`FPRINTF` imprime une chaîne texte formatée dans un fichier.

`(FPRINTF file format [expr ...])`

Formate une chaîne texte en utilisant les données du format chaîne et arguments pour l'imprimer dans le fichier spécifié. Le formatage est fait comme dans `SPRINTF` (voir [Section 15.12.32 \[SPRINTF\]](#), page 108).

`FPRINTF` renvoie le nombre de caractères en sortie ou `NIL` en cas d'échec. Si *file* est `NIL` alors `FPRINTF` renvoie toujours le nombre de caractères potentiellement écrits mais rien n'est réellement écrit en sortie. Si *format* est `NIL` alors `NIL` est retourné.

Voir aussi `PRINTF`, `FOPEN`.

15.17.7 FERROR

`FERROR` vérifie si un fichier produit une erreur d'entrée/sortie.

`(FERROR file)`

renvoie `TRUE` si une erreur pour les données d'un fichier donné s'est produite, `NIL` autrement. Si *'file'* est `NIL`, `NIL` est retourné.

Voir aussi `FEOF`, `FOPEN`, `FCLOSE`.

15.17.8 FEOF

`FEOF` vérifie l'état de fin de fichier.

`(FEOF file)`

Examine l'indicateur de fin de fichier d'un fichier donné et renvoie `TRUE` s'il est placé. autrement `NIL` est retourné. Si le *'file'* est `NIL`, `NIL` est retourné.

Voir aussi `FERROR`, `FTELL`, `FOPEN`, `FCLOSE`.

15.17.9 FSEEK

`FSEEK` place la position de lecture/écriture d'un fichier.

`(FSEEK file offset whence)`

Place la position de lecture/écriture pour un fichier donné. La nouvelle position, mesurée en octets, est obtenue en ajoutant des *offset* octets à la position spécifiée par *whence*. Si *whence* est placé sur `SEEK_SET`, `SEEK_CUR` ou `SEEK_END`, *offset* se reporte respectivement au début du fichier, à la position actuelle ou à la fin de fichier.

En cas de succès, `FSEEK` renvoie 0. Sinon `NIL` est retourné et la position dans le fichier reste inchangée. Si *lfile*, *offset* ou *whence* sont `NIL` ou si *whence* n'est pas une des constantes `SEEK_SET`, `SEEK_CUR` ou `SEEK_END`, alors `NIL` est retourné.

Veuillez noter qu'après une lecture l'appel à `FSEEK` avec une valeur *whence* pour `SEEK_BUR` n'est supporté que pour l'encodage `"none"`.

Voir aussi `FTELL`, `FOPEN`, constantes prédéfinies.

15.17.10 FTELL

FTELL renvoie la position de lecture/écriture d'un fichier.

(FTELL *file*)

Détermine la position de lecture/écriture courante d'un fichier donné par rapport au début du fichier et la renvoie comme un nombre entier. Si une erreur se produit ou si le 'file' est NIL, alors NIL est retourné.

Veuillez noter qu'après une lecture l'appel à FSEEK avec une valeur *whence* pour SEEK_BUR n'est supporté que pour l'encodage "none".

Voir aussi FSEEK, FOPEN, FEOF.

15.17.11 FGETCHAR

FGETCHAR lit un caractère à partir d'un fichier.

(FGETCHAR *file*)

Renvoie le prochain caractère à partir d'un fichier donné de la chaîne texte ou NIL si *file* est NIL, si la fin du fichier a été atteinte ou si une erreur s'est produite. Si le prochain caractère est un caractère nul, une chaîne texte vide est retournée.

Voir aussi FGETCHARS, FGETSTR, FPUTCHAR.

15.17.12 FGETCHARS

FGETCHARS lit des caractères à partir d'un fichier.

(FGETCHARS *num file*)

renvoie une chaîne texte contenant les *num* prochains caractères depuis un fichier donné Si la fin du fichier a été atteinte avant la lecture des *num* caractères ou si un caractère nul a été lu alors les caractères déjà lus sont retournés. Si *num* ou *file* sont NIL, si *num* est négative, si la fin du fichier a été atteinte avant de lire le premier caractère ou si une erreur de lecture c'est alors produite, NIL est retourné.

Voir aussi FGETCHAR, FGETSTR.

15.17.13 FGETSTR

FGETSTR lit une chaîne texte à partir d'un fichier.

(FGETSTR *file*)

renvoie la prochaine ligne d'un fichier donné comme une chaîne texte ou NIL si le *file* est NIL, la fin du fichier a été atteint, ou une erreur s'est produite. La fin d'une ligne est détectée soit par une nouvelle ligne de caractères, soit par la lecture d'un caractère nul , soit si la fin du fichier est détectée. Dans l'un ou l'autre cas la chaîne texte ne contient aucune nouvelle ligne de caractères.

Voir également FGETCHAR, FGETCHARS, FGETMEMO, FPUTSTR.

15.17.14 FGETMEMO

FGETMEMO lit un mémo à partir d'un fichier.

(FGETMEMO *file*)

renvoie un mémo qui contient le contenu d'un fichier donné jusqu'au prochain caractère nul ou jusqu'à la fin du fichier. Si la *file* est NIL, la fin du fichier a été atteinte avant de lire tous les caractères, ou une erreur s'est produite alors NIL est retourné.

Voir aussi FGETSTR, FPUTMEMO.

15.17.15 FPUTCHAR

FPUTCHAR écrit un caractère dans un fichier.

(FPUTCHAR *str file*)

Écrit le premier caractère de la *str* dans un fichier donné. Si la *str* est vide, un caractère nul est écrit, si la *str* ou la *file* sont NIL, rien ne se produit. Renvoie la *str* ou NIL au cas où une erreur de sortie se produirait.

Voir aussi FPUTSTR, FGETCHAR.

15.17.16 FPUTSTR

FPUTSTR écrit une chaîne texte dans un fichier.

(FPUTSTR *str file*)

Imprime la *str* ainsi qu'une nouvelle ligne de caractères dans un fichier donné. Si la *str* ou la *file* sont NIL, rien ne se produit. Renvoie la *str* ou NIL au cas où une erreur de sortie se produirait.

Voir aussi FPUTCHAR, FPUTMEMO, FGETSTR.

15.17.17 FPUTMEMO

FPUTMEMO écrit un mémo dans un fichier.

(FPUTMEMO *memo file*)

Imprime la *memo* dans un fichier donné. Si la *memo* ou la *file* sont NIL, rien ne se produit. Renvoie la *memo* ou NIL au cas où une erreur de sortie se produirait.

Voir aussi FPUTSTR, FGETMEMO.

15.17.18 FFLUSH

FFLUSH purge les données en attente vers un fichier.

(FFLUSH *fichier*)

Purge toutes les données en attente d'écriture pour le fichier spécifié. Renvoie 0 en cas de succès, NIL si une erreur se produit. Si la *fichier* est NIL alors 0 est retourné (aucune erreur).

Voir aussi FOPEN, FCLOSE.

15.18 Les fonctions sur les enregistrements

Cette section énumère les fonctions qui traitent des enregistrements.

15.18.1 NEW

NEW alloue un nouvel enregistrement pour une table.

`(NEW table init)`

Alloue un nouvel enregistrement pour une table donnée. Le paramètre de la *init* spécifie l'enregistrement qui doit être utilisé pour initialiser le nouvel enregistrement. La valeur NIL représente l'enregistrement initial.

NEW renvoie un indicateur d'enregistrement pour le nouvel enregistrement.

La fonction NEW a un effet secondaire qui initialise le programme pointant les enregistrements d'une table donnée (voir les voir [Section 5.2 \[Tables\], page 19](#)) vers un nouvel enregistrement.

Exemple: '`(NEW table NIL)`' alloue un nouvel enregistrement dans une table donnée et l'initialise avec l'enregistrement initial.

Voir aussi NEW*, DELETE, Tables.

15.18.2 NEW*

NEW* est une version améliorée de NEW (voir [Section 15.18.1 \[NEW\], page 127](#)).

`(NEW* table init)`

NEW* vérifie si vous avez spécifié un déclencheur de '**Création**' pour la table donnée (voir [Section 15.29.5 \[New trigger\], page 147](#)). Si c'est le cas, cette fonction est appelée pour allouer l'enregistrement et son résultat est retourné. Le paramètre *init* peut être utilisé pour spécifier l'enregistrement avec lequel le nouvel enregistrement devrait être initialisé (utilisez NIL pour un enregistrement initial).

Si aucun déclencheur n'a été spécifié, la fonction se comporte comme la fonction NEW.

Avertissement : Avec cette fonction il est possible d'écrire des boucles sans fin, par exemple si vous avez défini un déclencheur de '**Création**' pour une table et qu'il appelle NEW* pour allouer les enregistrements.

Voir aussi NEW, DELETE*.

15.18.3 DELETE

DELETE supprime les enregistrements dans une table.

`(DELETE table requester)`

Supprime les enregistrements du programme en cours d'une table donnée après une demande de confirmation facultative. Le premier argument indique la table dans laquelle les enregistrements du programme en cours devraient être supprimés, le deuxième argument est une expression booléenne. Si c'est NIL alors l'enregistrement est supprimé sans confirmation, si ce n'est pas NIL alors l'état du menu préférences '**Confirmer la suppression des enregistrements**' est vérifié. S'il n'est pas coché, l'enregistrement est supprimé sans confirmation, autrement une fenêtre de confirmation standard apparaît vous demandant si vous voulez vraiment effacer cet enregistrement. Si l'utilisateur clique sur le bouton annuler, l'enregistrement ne sera pas supprimé.

Le code renvoyé par la fonction DELETE reflète l'action choisie. S'il renvoie TRUE alors l'enregistrement a été supprimé, sinon (l'utilisateur a annulé l'opération) NIL est retourné.

A propos de la suppression, DELETE place le pointeur du programme d'enregistrement (voir les voir [Section 5.2 \[Tables\], page 19](#)) de la table spécifiée sur NIL.

Exemple: '`(DELETE table NIL)`' supprime, sans confirmation, l'enregistrement dans la table de données courante.

Voir aussi DELETE*, DELETEALL, NEW, Tables.

15.18.4 DELETE*

DELETE* est une version améliorée de DELETE (voir [Section 15.18.3 \[DELETE\]](#), page 128).

(DELETE* *table requester*)

DELETE* vérifie si vous avez spécifié un déclencheur de ‘Suppression’ pour la table donnée (Voir voir [Section 15.29.6 \[Delete trigger\]](#), page 148). Si c’est le cas, alors cette fonction est appelée pour supprimer l’enregistrement et son résultat est retourné. Le paramètre *requester* peut être utilisé pour indiquer si le déclencheur appelle la fenêtre de confirmation avant de supprimer l’enregistrement.

Si aucun déclencheur n’a été spécifié, la fonction se comporte comme la fonction DELETE

Avertissement : Avec cette fonction il est possible d’écrire des boucles sans fin, par exemple si vous avez défini un déclencheur de ‘Suppression’ pour une table et qu’il appelle DELETE* pour supprimer l’enregistrement.

Voir aussi DELETE, DELETEALL, NEW*.

15.18.5 DELETEALL

DELETEALL supprime tous les enregistrements d’une table.

(DELETEALL *table* [*])

Supprime tous les enregistrements de la table indiquée. si vous ajoutez une étoile derrière le nom de la table seul les enregistrements du filtre de la table encours seront supprimés. Il n’y a pas de fenêtre de confirmation avant de supprimer les enregistrements.

DELETEALL renvoie TRUE en cas de succès pour la suppression de tous les enregistrements, sinon NIL est retourné. Si la *table* est NIL alors NIL est retourné.

Exemple: ‘(DELETEALL *table**)’ supprime, en utilisant le filtre de la table, tous les enregistrements dans la table de données.

Voir aussi DELETE, Tables.

15.18.6 GETMATCHFILTER

GETMATCHFILTER renvoie l’état du filtre d’enregistrement.

(GETMATCHFILTER *rec*)

Renvoie TRUE si les enregistrements dispose d’un filtre dans la table, NIL dans les autres cas. Si le filtre de la table courante n’est pas actif, alors TRUE est retourné. Si *rec* est NIL (l’enregistrement initial) alors NIL est retourné.

Voir aussi SETMATCHFILTER, GETISSORTED, GETFILTERSTR, SETFILTERSTR.

15.18.7 SETMATCHFILTER

SETMATCHFILTER place l’état du filtre d’un enregistrement.

(SETMATCHFILTER *rec on*)

Change l’état du filtre de l’enregistrement indiqué dans la valeur *on*. SETMATCHFILTER renvoie le nouvel état du filtre de l’enregistrement donné. Le nouvel état peut être différent

de ce qui était prévu parce que le réglage de l'état du filtre vers NIL fonctionne seulement quand le filtre de la table courante correspondante est en activité, sinon TRUE est retourné. Appelez `SETMATCHFILTER` avec la valeur NIL pour `rec` (l'enregistrement initial) renverra toujours NIL.

Voir aussi `GETMATCHFILTER`, `SETISSORTED`, `GETFILTERSTR`, `SETFILTERSTR`.

15.18.8 GETISSORTED

`GETISSORTED` renvoie l'état du type d'enregistrement.

`(GETISSORTED rec)`

Renvoie TRUE si l'enregistrement spécifie le type de tri qui a été défini pour sa table, NIL dans les autres cas. Si la `rec` est NIL alors NIL est retourné.

Voir aussi `SETISSORTED`, `GETMATCHFILTER`, `REORDER`, `GETORDERSTR`, `SETORDERSTR`, `Comparison function`.

15.18.9 SETISSORTED

`SETISSORTED` modifie l'indicateur de tri de l'enregistrement.

`(SETISSORTED rec on)`

Change l'indicateur de tri de l'enregistrement spécifié par la valeur `on`. Utilisez cette fonction si vous pensez que certains enregistrements sont dans le bon ordre (`on = TRUE`) ou s'ils doivent être réordonnés (`on = NIL`). Le réordonnancement de tous les enregistrements qui ne sont pas triés peut être réalisé en appelant la fonction `REORDER` (voir voir [Section 15.20.4 \[REORDER\]](#), page 133).

`SETISSORTED` renvoie la nouvelle valeur de l'indicateur pour l'enregistrement donné. Appeler `SETISSORTED` avec la valeur NIL pour `rec` (l'enregistrement initial) renverra NIL.

Pour un exemple sur la façon d'employer cette fonction, voir la voir [Section 15.29.7 \[Comparison function\]](#), page 148.

Voir aussi `GETISSORTED`, `SETMATCHFILTER`, `REORDER`, `GETORDERSTR`, `SETORDERSTR`, `Fonction de comparaison`.

15.18.10 RECNUM

`RECNUM` renvoie le numéro d'enregistrement d'un enregistrement.

`(RECNUM record)`

Renvoie le numéro d'enregistrement d'un enregistrement donné. Veuillez noter que la numérotation des enregistrements est différente de celle utilisée par exemple pour les listes. Pour les listes, les chaînes texte et autres, le compte commence à zéro. Toutefois, pour les enregistrements, il commence par 1 pour le premier enregistrement. Le numéro 0 est réservé à l'enregistrement initial. Cela semble être contradictoire avec le reste des fonctions de programmation de MUIbase, mais prend ici vraiment son sens lorsque les numéros d'enregistrement sont également utilisés dans la fenêtre d'affichage.

Voir aussi `RECORDS`, `INT`.

15.18.11 COPYREC

COPYREC copie les enregistrements.

(COPYREC *rec source*)

Copie le contenu de l'enregistrement *source* dans l'enregistrer *rec*. Si *source* est NIL alors *rec* est initialisé à la valeur de l'enregistrement initial. Si *rec* est NIL alors, un message d'erreur est généré.

COPYREC renvoie *rec*.

Voir aussi NEW.

15.19 Fonctions sur les champs

Cette section énumère les fonctions qui travaillent sur les champs d'une table.

15.19.1 ATTRNAME

ATTRNAME renvoie le nom d'un champ.

(ATTRNAME *attr*)

Renvoie une chaîne contenant le nom d'un champ indiqué.

Voir aussi TABLENAME

15.19.2 MAXLEN

MAXLEN renvoie la taille maximum du champ d'une chaîne.

(MAXLEN *string-attr*)

Renvoie le nombre maximum des caractères que le champ d'une chaîne texte donné peut contenir.

Voir aussi LEN.

15.19.3 GETLABELS

GETLABELS renvoie toutes les étiquettes d'un choix ou d'une chaîne.

(GETLABELS *attr*)

Renvoie les étiquettes d'un champ choix ou chaîne donné. Dans le cas d'un champ choix les étiquettes que vous avez entrées dans la fenêtre de saisie du champ (voir [Section 14.2.2 \[Type specific settings\], page 62](#)) sont retournées, dans le cas du champ chaîne les étiquettes statiques que vous avez entrées dans la liste déroulante (voir [Section 14.3.3 \[Attribute object editor\], page 67](#)) sont retournées (notez que cette fonction n'est utile que pour des étiquettes statiques).

Les étiquettes sont retournées dans une seule chaîne et sont séparées par un caractère de retour à la ligne.

Par exemple, si vous considérez que vous avez un champ choix avec les étiquettes 'Voiture', 'Maison', et 'Huile'. En appelant GETLABELS sur ce champ vous obtiendrez le résultat "Voiture\nMaison\nHuile" dans une chaîne.

Note: vous pouvez facilement convertir la chaîne résultat en une liste en appelant MEMOTOLIST (voir [Section 15.13.3 \[MEMOTOLIST\], page 111](#)) avec cette chaîne.

Voir aussi SETLABELS.

15.19.4 SETLABELS

SETLABELS est utilisé pour placer les étiquettes d'un champ dans une chaîne.

`(SETLABELS attr str)`

Définit les étiquettes statiques du champ chaîne *attr* à partir des étiquettes listées dans l'argument *str*. L'argument *str* se compose de lignes qui comportent une étiquette. Les étiquettes remplacent celles que vous avez entrées dans la liste de l'éditeur de champ objet (voir [Section 14.3.3 \[Attribute object editor\]](#), page 67). Notez que cette fonction n'est utile que pour des étiquettes statiques.

SETLABELS renvoie la valeur de l'argument *str*.

Exemple: `'(SETLABELS Table.String "Ma maison\nest\nvotre maison")'` définit les étiquettes statiques dans la liste d'affichage et indique le champ de la chaîne texte 'Ma maison', 'est', et 'votre maison'.

Note: vous pouvez facilement convertir une liste d'étiquettes dans le format requis par une chaîne texte en appelant LISTTOMEMO sur la liste.

Voir aussi GETLABELS.

15.20 Fonctions sur les tables

15.20.1 TABLENAME

TABLENAME renvoie le nom d'une table.

`(TABLENAME table)`

Renvoie une chaîne texte contenant le nom de la table indiquée.

Voir aussi ATTRNAME

15.20.2 GETORDERSTR

GETORDERSTR renvoie un enregistrement trié dans une table.

`(GETORDERSTR table)`

Renvoie l'expression courante de tri pour les données d'une table. Si la table utilise une liste de champs pour le tri, alors la chaîne retournée contiendra les noms des champs séparés par des espaces. Chaque nom de champ est précédé par le signe '+' ou '-' indiquant un tri croissant ou décroissant.

Si la table est triée par une fonction de comparaison alors le nom de cette fonction est retourné.

Une chaîne texte vide signifie qu'il n'y a aucun tri.

Exemple

Considérez une table 'Personne' qui est triée par les champs 'Nom' (croissant), 'Ville' (croissant), et 'Anniversaire' (décroissant). Alors, `'(ORDERSTR Personne)'` fournira le résultat de la chaîne texte "+Nom +Ville -Anniversaire".

Voir aussi SETORDERSTR, REORDER, REORDERALL, GETISSORTED, SETISSORTED, Order, Fonction de comparaison.

15.20.3 SETORDERSTR

SETORDERSTR place un enregistrement trié dans une table.

```
(SETORDERSTR table order)
```

Place le tri dans les données de la table selon le champ *Tri*. Le champ *Tri* peut contenir la liste des noms de champs ou le nom d'une fonction de comparaison.

Pour trier en utilisant une liste de champs, le champ *Tri* doit contenir les noms des champs séparés par un nombre d'espaces, tabulations ou une nouvelle ligne. Chaque nom de champ peut être précédé par le signe '+' ou a '-' indiquant un tri croissant ou décroissant. Si vous omettez ce signe, alors le tri croissant est assumé.

Pour trier en utilisant une fonction de comparaison, le champ *Tri* doit contenir le nom de la fonction.

SETORDERSTR renvoie TRUE s'il a pu placer le nouveau tri, NIL autrement, par exemple si un champ inconnu a été indiqué ou si le type du champ n'est pas permis pour le tri. Si vous indiquez NIL pour l'argument *Tri* alors, rien ne se produit et NIL est retourné.

Note: Pour construire le champ tri vous ne devriez pas écrire directement les noms des champs dans un champ parce que quand vous changerez un nom de champ, le champ tri ne sera pas mis à jour. Il vaut mieux utiliser la fonction ATTRNAME (voir voir [Section 15.19.1 \[ATTRNAME\], page 131](#)) qui permet de copier le nom d'un champ dans le champ tri.

Exemple

Si l'on considère une table 'Personne' avec les champs 'Nom', 'Ville', et 'Anniversaire'. alors, '(SETORDERSTR Personne (SPRINTF "+%s" (ATTRNAME Personne.Nom)))' placera le tri dans la table 'Personne' en utilisant le 'Nom' comme champ de tri (croissant).

Voir aussi GETORDERSTR, REORDER, REORDERALL, GETISSORTED, SETISSORTED, Order, Fonction de comparaison.

15.20.4 REORDER

REORDER réordonne tous les enregistrements non triés dans le bon ordre.

```
(REORDER table)
```

Examine tous les enregistrements d'une table de données pour les réordonner et les réinsérer à leur bonne position. Après réinsertion d'un enregistrement réordonné l'état de tri de l'enregistrement est placé sur TRUE, par conséquent l'état de tri de tous les enregistrements renvoyés par REORDER est TRUE.

REORDER renvoie NIL.

Habituellement vous devez seulement appeler cette fonction lorsque vous employez une fonction de comparaison pour définir le tri d'une table. Les tris définis par une liste de champs sont automatiques, c'est-à-dire, un enregistrement est réordonné automatiquement lorsque c'est nécessaire.

Pour un exemple sur la façon d'utiliser cette fonction, voir la voir [Section 15.29.7 \[Comparison function\], page 148](#).

Voir aussi REORDERALL, GETORDERSTR, SETORDERSTR, GETISSORTED, SETISSORTED, Order, Fonction de comparaison.

15.20.5 REORDERALL

REORDERALL réordonne tous les enregistrements d'une table.

(REORDERALL *table*)

Réordonne toutes les données enregistrées d'une table en plaçant l'état de tri de tous les enregistrements sur NIL et en appelant REORDER pour tout réordonner.

REORDERALL renvoie NIL.

Voir aussi REORDER, GETORDERSTR, SETORDERSTR, GETISSORTED, SETISSORTED, Order, Fonction de comparaison.

15.20.6 GETFILTERACTIVE

GETFILTERACTIVE renvoie l'état du filtre de la table.

(GETFILTERACTIVE *table*)

Renvoie TRUE si le filtre de la table courante spécifiée est activé et NIL dans les autres cas.

Voir aussi SETFILTERACTIVE, GETFILTERSTR, GETMATCHFILTER.

15.20.7 SETFILTERACTIVE

SETFILTERACTIVE Place l'état du filtre de la table.

(SETFILTERACTIVE *table bool*)

Place l'état du filtre de la table spécifiée. Si *bool* est différent de NIL alors le filtre est activé, sinon il est désactivé.

SETFILTERACTIVE renvoie le nouvel état du filtre. Si vous activez le filtre, le nouvel état attendu ne sera pas là, mais une erreur se produit et le filtre ne peut pas être activé. Cependant le filtre se désactive toujours avec succès.

Voir aussi GETFILTERACTIVE, SETFILTERSTR, SETMATCHFILTER.

15.20.8 GETFILTERSTR

GETFILTERSTR renvoie l'expression de l'enregistrement dans le filtre d'une table.

(GETFILTERSTR *table*)

Renvoie l'expression de l'enregistrement pour le filtre d'une table spécifiée dans champs. Un champ vide signifie qu'aucune expression de filtrage n'a été placée pour cette table.

Voir aussi SETFILTERSTR, GETFILTERACTIVE, GETMATCHFILTER.

15.20.9 SETFILTERSTR

SETFILTERSTR place l'expression de l'enregistrement dans le filtre d'une table.

(SETFILTERSTR *table filter-str*)

Place l'expression de l'enregistrement pour le filtre d'une table spécifiée par l'expression dans l'argument *filter-str* (qui doit être un champ et pas l'expression réelle elle-même !). Si le filtre d'une table donnée est actuellement activé alors, la nouvelle expression de filtrage est appliquée directement à tous les enregistrements et l'état du filtre de tous les enregistrements sont recalculés.

SETFILTERSTR renvoie TRUE s'il a pu compiler les données filtrées par le champ de l'expression, sinon NIL est retourné. Notez que vous obtenez seulement le résultat de la

compilation. Si le filtre d'une table donnée est actuellement activé et recalculé tous les états du filtre correspondant aux enregistrements échoueront et ne seront donc pas notés dans le résultat de cette fonction. Pour placer une nouvelle expression de filtrage il est recommandé de procéder de la manière suivante :

```
(SETFILTERACTIVE Table NIL)                ; réussit toujours.
(IF (NOT (SETFILTERSTR Table filter-string))
  (ERROR "ne peut pas placer le champ filtre pour %s!" (TABLENAME Table))
)
(IF (NOT (SETFILTERACTIVE Table TRUE))
  (ERROR "ne peut pas activer le filtre pour %s!" (TABLENAME Table))
)
```

Si SETFILTERSTR est appelé avec une valeur NIL pour l'argument *filter-str* et que rien ne se produit, NIL est retourné.

Exemple: '(SETFILTERSTR Table "> Value 0.0)")'.

Voir aussi GETFILTERSTR, SETFILTERACTIVE, SETMATCHFILTER.

15.20.10 RECORDS

RECORDS renvoie le nombre d'enregistrements dans une table.

```
(RECORDS table)
```

Renvoie le nombre d'enregistrements dans une table donnée. Vous pouvez apposer une étoile sur le nom d'une table pour compter le nombre d'enregistrements identique au filtre de la table.

Voir aussi RECORD, RECNUM.

15.20.11 RECORD

RECORD renvoie l'indicateur d'enregistrement pour le nombre de données enregistrées.

```
(RECORD table num)
```

renvoie l'indicateur d'enregistrement vers *num* enregistré dans les données de la table ou NIL si l'enregistrement avec ce nombre n'existe pas. Vous pouvez ajouter une étoile sur le nom d'une table pour obtenir un *num* enregistrement identique à l'enregistrement filtré de la table.

Veuillez noter que les nombres d'enregistrements commencent par 1 et que le numéro d'enregistrement 0 est utilisé pour l'enregistrement initial.

Voir aussi RECORDS, RECNUM.

15.20.12 SELECT

SELECT extrait et renvoie diverses données à partir des enregistrements.

```
(SELECT [DISTINCT] exprlist FROM tablelist
  [WHERE where-expr] [ORDER BY orderlist])
```

Ici *exprlist* est soit une simple étoile '*' soit une liste d'expressions avec des titres facultatifs séparés par des virgules :

```
exprlist:      * | expr "titre", ...
```

et *tablelist* est une liste de noms de table :

```
tablelist:      table [*] [ident], ...
```

Pour chaque table dans la liste de table vous pouvez indiquer une marque. Ce qui peut être très utile si une table se produit plus d'une fois dans la liste de table (voir l'exemple de comparaison des âges ci-dessous). Si vous ajoutez une étoile à une table alors seul les enregistrements identiques au filtre actuellement défini dans cette table seront examinés.

orderlist a la syntaxe suivante :

```
orderlist:      expr [ASC | DESC], ...
```

Ici, *expr*, ... peuvent être des expressions arbitraires ou des nombres de champs. Par exemple '(SELECT Nom FROM ... ORDER BY 1)' triera le résultat pour le champ 'Nom'. Vous pouvez spécifier ASC ou DESC pour un tri croissant ou décroissant. Si aucun d'eux n'est présent, le tri croissant est assumé.

Comment ça fonctionne

Choisir à partir d'une question construite (mathématique) le résultat produit par toutes les tables dans une liste de table (il examine les séries d'enregistrements dans la *table*, ...) et contrôle où se trouve l'expression (s'il y en a). Si le pointeur de l'expression renvoie TRUE comme résultat (ou s'il n'y a aucun pointeur sur l'expression) alors la liste est une construction dont les éléments sont calculés par une liste d'expression dans la partie sélectionnée. Si vous avez spécifié une simple étoile pour une liste d'expression alors qu'une liste contient les valeurs de tous les champs appartenant aux tables dans une liste de table (excepté les champs virtuels et les boutons).

Le résultat de la question est une liste de listes. La première entrée d'une liste contient le titre d'une chaîne, les autres contiennent les valeurs à partir de la liste dans les enregistrements correspondant.

Exemples

Voir les Voir [Section 13.4 \[Query examples\]](#), page 58, pour quelques exemples utilisant la fonction SELECT .

Voir aussi FOR ALL.

15.21 Les fonctions GUI

Cette section décrit les fonctions pour manipuler les éléments d'une interface graphique.

15.21.1 SETCURSOR

SETCURSOR place le curseur sur un élément graphique.

```
(SETCURSOR attr-or-table)
```

Place le curseur sur un objet graphique représentant un champ ou une table. La fonction ouvre également la fenêtre où le champ/table réside si la fenêtre n'était pas déjà ouverte.

SETCURSOR renvoie TRUE si tout va bien (la fenêtre a pu être ouverte) ou NIL en cas d'échec.

Voir aussi SETVIRTUALLISTACTIVE.

15.21.2 GETWINDOWOPEN

GETWINDOWOPEN renvoie l'état d'ouverture d'une fenêtre.

(GETWINDOWOPEN *attr-or-table*)

Renvoie l'état d'ouverture d'une fenêtre où réside le champ/table.

Voir aussi SETWINDOWOPEN.

15.21.3 SETWINDOWOPEN

SETWINDOWOPEN ouvre ou ferme une fenêtre.

(SETWINDOWOPEN *attr-or-table open*)

Ouvre ou ferme la fenêtre dans laquelle réside le champ/table. Si *open* n'est pas NIL alors la fenêtre est ouverte, sinon elle est fermée. Vous ne pouvez pas fermer la fenêtre principale d'un projet.

SETWINDOWOPEN renvoie le nouvel état d'ouverture de la fenêtre.

Voir aussi GETWINDOWOPEN.

15.21.4 GETVIRTUALLISTACTIVE

GETVIRTUALLISTACTIVE renvoie l'index de la ligne active d'un champ virtuel utilisant une 'Liste' son affichage.

(GETVIRTUALLISTACTIVE *virtual-attr*)

Renvoie l'index (commençant par 1) de la ligne active courante avec le champ virtuel spécifié. Si l'élément GUI de *virtual-attr* n'est pas visible, s'il n'emploie pas de 'Liste' pour son affichage, ou si aucune ligne n'est activée, alors NIL est retourné.

Voir aussi SETVIRTUALLISTACTIVE.

15.21.5 SETVIRTUALLISTACTIVE

SETVIRTUALLISTACTIVE Modifie la ligne active d'un champ virtuel utilisant une 'Liste' pour son affichage.

(SETVIRTUALLISTACTIVE *virtual-attr num*)

Modifie la ligne active du champ virtuel sur la *num*ème ligne (commençant à 1). Renvoie *num*, ou NIL si l'élément GUI de *virtual-attr* n'est pas visible, n'utilise pas de 'Liste' pour son affichage ou si *num* est hors limite (inférieure à 1 ou plus grand que le nombre de lignes).

SETVIRTUALLISTACTIVE ne place pas le curseur sur l'élément graphique du champ, utilisez pour cela SETCURSOR (voir voir [Section 15.21.1 \[SETCURSOR\], page 136](#)).

Voir aussi GETVIRTUALLISTACTIVE, SETCURSOR.

15.22 Les fonctions projets

Cette section énumère les fonctions traitant des projets.

15.22.1 PROJECTNAME

PROJECTNAME renvoie le nom de projet.

(PROJECTNAME)

PROJECTNAME renvoie le nom du projet en cours dans une chaîne ou NIL si aucun nom n'a encore été défini. Le nom du projet est le chemin du répertoire-projet du système de fichier.

Voir aussi CHANGES.

15.22.2 CHANGES

CHANGES renvoie le nombre de changements du projet en cours.

(CHANGES)

Renvoie un nombre entier contenant le nombre de changement depuis la dernière opération de sauvegarde du projet en cours.

Voir aussi PROJECTNAME.

15.22.3 GETADMINMODE

GETADMINMODE tells whether the current project is in admin mode GETADMINMODE indique si le projet courant est en mode administrateur ou utilisateur.

(GETADMINMODE)

Retourne TRUE si le projet courant est en mode administrateur, NIL sinon.

Voir aussi SETADMINMODE, ADMINPASSWORD, onAdminMode.

15.22.4 SETADMINMODE

SETADMINMODE bascule le projet courant vers le mode administrateur ou utilisateur.

(SETADMINMODE *admin*)

Si *admin* est NIL alors le projet courant est basculé en mode utilisateur, dans le cas contraire il passe en mode administrateur. Notez qu'il n'y a pas de fenêtre d'identification lors du passage du mode utilisateur vers le mode administrateur en utilisant cette fonction.

Retourne TRUE si le projet a été basculé en mode administrateur, ou NIL s'il est basculé en mode utilisateur.

Voir aussi GETADMINMODE, ADMINPASSWORD, onAdminMode, démo 'Users.mb'.

15.22.5 ADMINPASSWORD

ADMINPASSWORD obtient le mot de passe administrateur sous forme de chaîne de hachage SHA1.

(ADMINPASSWORD)

Retourne une chaîne représentant la valeur de hachage SHA1 du mot de passe administrateur du projet courant. Si aucun mot de passe administrateur n'a été configuré, NIL est retourné.

Voir aussi GETADMINMODE, SETADMINMODE, SHA1SUM, démo 'Users.mb'.

15.23 Les fonctions système

Cette section énumère les fonctions faisant appel au système d'exploitation.

15.23.1 EDIT

EDIT Lancement d'un éditeur externe.

(EDIT *filename*)

Démarrez un éditeur externe pour éditer un fichier spécifié. L'éditeur externe peut être placé dans le menu 'Préférences - Configurer l'éditeur externe' (voir voir [Section 7.1.2 \[External editor\], page 34](#)). EDIT démarre l'éditeur externe synchroniquement, ce qui veut dire qu'il attend jusqu'à ce que l'utilisateur sorte de l'éditeur.

EDIT renvoie le code return de l'éditeur externe comme nombre entier.

Voir aussi EDIT*, VIEW, SYSTEM.

15.23.2 EDIT*

EDIT* est la version étoilée de EDIT et a le même effet que EDIT (voir voir [Section 15.23.1 \[EDIT\], page 139](#)). La seule différence est que EDIT* démarre un éditeur externe de façon asynchrone, ainsi la fonction rend la main immédiatement.

EDIT* renvoie 0 en cas de succès lors du démarrage de l'éditeur, sinon il renvoie la valeur d'un nombre entier différent de zéro représentant un code d'erreur système spécifique.

Voir aussi EDIT, VIEW*, SYSTEM*.

15.23.3 VIEW

VIEW lance la visionneuse externe.

(VIEW *filename*)

Démarre la visionneuse externe pour afficher le fichier spécifié. La visionneuse externe peut être placée dans menu 'Préférences - Configurer la visionneuse' (voir voir [Section 7.1.3 \[External viewer\], page 35](#)). VIEW démarre la visionneuse externe synchroniquement, c'est-à-dire qu'elle attend jusqu'à ce que l'utilisateur sorte de la visionneuse. Notez que sur quelques systèmes, l'appel pourrait immédiatement être renvoyé si un exemple de la visionneuse fonctionne déjà.

VIEW* renvoie le code renvoyé par la visionneuse externe comme un nombre entier.

Voir aussi VIEW*, EDIT, SYSTEM.

15.23.4 VIEW*

VIEW* est la version étoilée de VIEW et a les mêmes effets que VIEW (voir [Section 15.23.3 \[VIEW\], page 139](#)). La seule différence est que VIEW* démarre la visionneuse externe de façon asynchrone, ainsi la fonction rend la main immédiatement.

VIEW* renvoie 0 s'il réussit à démarrer la visionneuse, sinon il renvoie la valeur d'un nombre entier différent de zéro représentant un code d'erreur système spécifique.

Voir aussi VIEW, EDIT*, SYSTEM*.

15.23.5 SYSTEM

SYSTEM appelle un programme externe.

(SYSTEM *fmt* [*arg* ...])

Appelle un programme externe. La ligne de commande pour appeler le programme est spécifiée par *fmt* et des arguments facultatifs comme dans la fonction `SPRINTF` (voir

Section 15.12.32 [SPRINTF], page 108). Pour interpréter la ligne de commande, le shell du système est utilisé (/bin/sh sur Linux, ShellExecute sur Windows, le shell utilisateur sur Amiga). **SYSTEM** attend jusqu'à ce que le programme appelé soit terminé.

SYSTEM renvoie le code retour de la commande exécutée sous forme de nombre entier.

Voir aussi **SYSTEM***, **EDIT**, **VIEW**.

15.23.6 **SYSTEM***

SYSTEM* est la version étoilée de **SYSTEM** et a les mêmes effets que **SYSTEM** (voir Section 15.23.5 [SYSTEM], page 139). La seule différence est que **SYSTEM*** exécute la ligne de commande de façon asynchrone, ainsi la fonction rend la main immédiatement.

SYSTEM* renvoie 0 s'il réussit à lancer l'exécution de la ligne de commande, sinon il renvoie un nombre entier différent de zéro représentant un code d'erreur spécifique au système.

Voir aussi **SYSTEM**, **EDIT***, **VIEW***.

15.23.7 **STAT**

STAT examine un nom de fichier.

(**STAT** *filename*)

Vérifie si le fichier dont le nom a été indiqué existe dans le système. **STAT** renvoie NIL si le fichier ne peut pas être trouvé, 0 si le fichier existe et est un tiroir, et un nombre entier supérieur à 0 si le fichier existe et est un fichier normal.

15.23.8 **TACKON**

TACKON crée le chemin d'accès.

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

Fusionne *dirname* et tous les composants de [*component* ...] pour obtenir un chemin d'accès. **TACKON** sait traiter les caractères spéciaux utilisés comme séparateurs à la fin de chaque élément. Il renvoie le chemin d'accès dans un champ ou NIL si aucun des arguments est NIL. Notez que **TACKON** n'effectue aucun contrôle sur l'existence ou non du chemin d'accès résultant.

Exemple: '(**TACKON** "Sys:System" "CLI")' donne "Sys:System/CLI".

Voir aussi **FILENAME**, **DIRNAME**.

15.23.9 **FILENAME**

FILENAME extrait le nom de fichier à partir du chemin d'accès.

(**FILENAME** *path*)

Extrait le dernier composant du chemin d'accès spécifié. Aucune vérification n'est faite si le *path* spécifié se rapporte réellement à un fichier, ainsi il est également possible d'utiliser **FILENAME** pour obtenir le nom d'un sous-répertoire. **FILENAME** renvoie son résultat sous forme de chaîne ou NIL si *path* est NIL.

Exemple: '(**FILENAME** "Sys:System/CLI")' donne "CLI".

Voir aussi **DIRNAME**, **TACKON**.

15.23.10 DIRNAME

DIRNAME extrait une partie correspondant au répertoire depuis un chemin d'accès.

(DIRNAME *path*)

Extrait la partie correspondant au répertoire du chemin d'accès spécifié. Aucune vérification n'est faite si *path* se rapporte effectivement à un fichier, ainsi il est également possible d'utiliser DIRNAME pour obtenir le répertoire parent. DIRNAME renvoie son résultat sous forme de chaîne ou NIL si *path* est NIL.

Exemple: '(DIRNAME "Sys:System/CLI")' donne "Sys:System".

Voir aussi FILENAME, TACKON.

15.23.11 MESSAGE

MESSAGE affiche un message à l'utilisateur.

(MESSAGE *fmt* [*arg* ...])

Positionne le titre de la fenêtre pause/abort (si elle est ouverte). Le titre est créé à partir de *fmt* et des arguments facultatifs comme dans la fonction SPRINTF (voir [Section 15.12.32 \[SPRINTF\]](#), page 108).

MESSAGE renvoie le champ titre formaté.

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

Voir aussi PRINT, PRINTF.

15.23.12 COMPLETEMAX

COMPLETEMAX spécifie le nombre maximum d'étapes de progression.

(COMPLETEMAX *steps*)

Spécifie le nombre maximum d'étapes pour montrer à l'utilisateur la progression de votre programme MUIbase. Le nombre d'étapes par défaut (si vous n'appellez pas cette fonction) est de 100. La valeur de l'argument *steps* doit être un nombre entier. Si *steps* est NIL ou 0 alors aucune barre de progression n'est affichée. La barre de progression fait partie de la fenêtre pause/arrêt et surgit après un bref délai en exécutant un programme MUIbase.

COMPLETEMAX renvoie son argument *steps*.

Voir aussi COMPLETEADD, COMPLETE.

15.23.13 COMPLETEADD

COMPLETEADD incrémente l'état de progression.

(COMPLETEADD *add*)

Ajoute le nombre entier contenu dans *add* à la valeur courante de la progression. La valeur initiale de progression est placée sur 0. Si *add* est NIL alors la valeur de progression est remise à 0 et aucune barre de progression n'est montrée.

COMPLETEADD renvoie son argument *add*.

Exemple :

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

Voir aussi COMPLETMAX, COMPLETE.

15.23.14 COMPLETE

COMPLETE modifie l'état de progression.

```
(COMPLETE cur)
```

Positionne la valeur courante de progression sur la valeur entière *cur*. Si *cur* est NIL ou 0 alors aucune barre de progression n'est montrée.

COMPLETE renvoie son argument *cur*.

Exemple :

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

Voir aussi COMPLETMAX, COMPLETEADD.

15.23.15 GC

GC force le passage du ramasse-miettes .

```
(GC)
```

Force le passage du ramasse-miettes et renvoie NIL. Normalement le ramasse-miettes est exécuté automatiquement de manière régulière.

15.23.16 PUBSCREEN

PUBSCREEN retourne le nom de l'écran publique.

```
(PUBSCREEN)
```

Sur Amiga, PUBSCREEN retourne le nom de l'écran public sur lequel est affiché MUIbase, oU NIL si l'écran n'est pas publique.

Sur tous les autres systèmes, PUBSCREEN retourne NIL.

15.24 Les variables prédéfinies

MUIbase connaît quelques variables globales prédéfinies.

Dans la version en cours il existe seulement une variable globale : **stdout** (voir [Section 15.17.3 \[stdout\]](#), page 124).

15.25 Les constantes prédéfinies

Les constantes prédéfinies suivantes peuvent être employées dans toute expression pour la programmation.

Nom	Type	Valeur	Commentaire
NIL	tous	NIL	
TRUE	booléen	TRUE	
RESET	texte	"\33c"	
NORMAL	texte	"\33[0m"	
ITON	texte	"\33[3m"	
ITOFF	texte	"\33[23m"	
ULON	texte	"\33[4m"	
ULOFF	texte	"\33[24m"	
BFON	texte	"\33[1m"	
BFOFF	texte	"\33[22m"	
ELITEON	texte	"\33[2w"	
ELITEOFF	texte	"\33[1w"	
CONDON	texte	"\33[4w"	
CONDOFF	texte	"\33[3w"	
WIDEON	texte	"\33[6w"	
WIDEOFF	texte	"\33[5w"	
NLQON	texte	"\33[2\"z"	
NLQOFF	texte	"\33[1\"z"	
INT_MAX	entier	2147483647	Valeur entière maximum
INT_MIN	entier	-2147483648	Valeur entière minimum
HUGE_VAL	réel	1.797693e+308	Valeur réelle absolue maximum
PI	réel	3.14159265359	
OSTYPE	texte	<Type OS>	"Unix", "Windows" ou "Amiga"
OSVER	entier	<Version OS>	
OSREV	entier	<Révision OS>	
MBVER	entier	<Version MUIbase>	
MBREV	entier	<Révision MUIbase>	
LANGUAGE	texte	dépendant	Langue par défaut
SEEK_SET	entier	cf. stdio.h	Seek depuis début de fichier
SEEK_CUR	entier	cf. stdio.h	Seek depuis position courante
SEEK_END	entier	cf. stdio.h	Seek depuis fin de fichier

Voir [Section 15.4.7 \[Constants\], page 81](#), pour plus d'informations sur les constantes. Pour définir vos propres constantes, utilisez l'instruction du préprocesseur `#define` (voir [Section 15.3.1 \[#define\], page 76](#)).

15.26 Paramètres fonctionnels

Vous pouvez passer une fonction comme argument à une autre fonction. C'est utile pour définir des fonctions évoluées, comme par exemple pour trier ou construire une liste.

Pour appeler une fonction qui a été passée dans un argument vous devez utiliser la fonction `FUNCALL` (voir [Section 15.6.6 \[FUNCALL\]](#), page 87).

Exemple :

```
(DEFUN map (l fun)                # arguments: list and function
  (LET (res)                      # local variable res, initialized with NIL
    (DOLIST (i l)                 # for all items one by one
      (SETQ res
        (CONS (FUNCALL fun i) res) # calls function and
      )                          # build new list
    )
    (REVERSE res)                # we need to reverse the new list
  )
)
```

Vous pouvez maintenant employer la fonction `map` par exemple pour incrémenter tous les articles d'une liste de nombres entiers :

`'(map (LIST 1 2 3 4) 1+)'` donne `(2 3 4 5)`.

Voir aussi `FUNCALL`, `APPLY`, `MAPFIRST`.

15.27 Spécificateurs de type

Il est possible d'indiquer le type d'une variable en ajoutant un spécificateurs de type derrière son nom. Les spécificateurs de type suivants existent :

Spécificateur Description

<code>:INT</code>	pour les entiers
<code>:REAL</code>	pour les réels
<code>:STR</code>	pour les chaînes
<code>:MEMO</code>	pour les mémos
<code>:DATE</code>	pour les dates
<code>:TIME</code>	pour les heures
<code>:LIST</code>	pour les listes
<code>:FILE</code>	pour les descripteurs de fichier
<code>:FUNC</code>	pour les fonctions de n'importe quel type
<code>:table</code>	pour les pointeurs d'enregistrement sur <i>table</i>

un spécificateur de type s'accroche au nom de variable comme dans l'exemple suivant :

```
(LET (x:INT (y:REAL 0.0) z) ...)
```

L'exemple définit trois nouvelles variables `'x'`, `'y'` et `'z'`, où `'x'` est de type entier et initialisé avec `NIL`, `'y'` est de type réel et initialisé avec `0.0`, et `'z'` est une variable sans type initialisé avec `NIL`.

L'avantage des spécificateurs de type est que le compilateur peut détecter plus d'erreurs de typage, p. ex. si vous avez une fonction :

```
(DEFUN foo (x:INT) ...)
```

et que vous l'appellez avec `'(foo "bar")'`, le compilateur produira un message d'erreur. Cependant, si vous appelez `'foo'` avec une valeur non typée, p. ex. `'(foo (FIRST list))'` alors aucune vérification d'erreur de typage ne peut être faite lors de la compilation puisque le type de `'(FIRST list)'` est inconnu à ce moment.

Pour des raisons de performance, aucune vérification de typage n'est actuellement faite lors de l'exécution. Cela pourrait être implémenté, mais ajouterait une légère surcharge inutile puisque de toute façon un mauvais type produira tôt ou tard une erreur de typage.

Les spécificateurs de type pour les pointeurs d'enregistrements ont une autre fonction utile. Si vous étiquetez une variable comme pointeur d'enregistrement vers une table alors vous pourrez accéder à tous les champs de cette table en employant le nom de la variable au lieu de celui de la table dans le chemin d'accès au champ. Par exemple si vous avez une table `'Foo'` avec un champ `'Bar'`, et que vous définissez une variable `'foo'` comme :

```
(LET (foo:Foo)
```

alors vous pourrez afficher le champ `'Bar'` du troisième enregistrement en utilisant :

```
(SETQ foo (RECORD Foo 3)) (PRINT foo.Bar)
```

Notez que dans une expression `select-from-where`, les variables définies dans la liste ont automatiquement un type pointeur d'enregistrement vers la table correspondante.

15.28 Sémantique des expressions

La sémantique des expressions est très importante pour la compréhension des programmes. Cette section énumère la sémantique selon des expressions syntaxiques.

(func [expr ...])

Évalue *expr ...* puis appelle la fonction *func* (appel par valeur). Renvoie la valeur de retour de la fonction appelée. Dans MUIbase il existe quelques fonctions non strictes, par exemple. `AND`, `OR` et `IF`. Ces fonctions peuvent ne pas évaluer toutes les expressions. Pour plus d'informations sur les fonctions non strictes, voir [Section 15.4.2 \[Lisp syntax\]](#), page 79, [Section 15.9.1 \[AND\]](#), page 96, [Section 15.9.2 \[OR\]](#), page 96, et [Section 15.6.8 \[IF\]](#), page 87.

([expr ...])

Évalue *expr ...* et renvoie la valeur de la dernière expression (voir [Section 15.6.1 \[PROGN\]](#), page 85). Une expression vide `()` s'évalue en `NIL`.

Table

Renvoie le pointeur d'enregistrement du programme pour la table.

*Table**

Renvoie le pointeur d'enregistrement de l'interface pour la table de données.

AttrPath

Renvoie le contenu du champ spécifié. Le chemin du champ spécifie quel enregistrement est utilisé pour extraire la valeur du champ. Par exemple `'Table.Champ'` utilise l'enregistrement du programme `'Table'` pour extraire la valeur du champ, `'Table.ChampRéférence.Champ'` utilise l'enregistrement du programme `'Table'` pour extraire la valeur du champ de référence (qui est un pointeur d'enregistrement) et utilise alors cet enregistrement pour extraire la valeur `'Champ'`.

var

Renvoie le contenu de la variable globale ou locale *var*. Les variables globales peuvent être définies avec **DEFVAR** (voir [Section 15.5.3 \[DEFVAR\]](#), page 84), et les variables locales par exemple avec **LET** (voir [Section 15.6.3 \[LET\]](#), page 85).

var.AttrPath

Utilise le pointeur d'enregistrement *var* pour déterminer la valeur du champ spécifié.

15.29 Déclenchement de fonction

Pour exécuter automatiquement des programmes MUIbase vous pouvez spécifier des fonctions déclencheurs sur les projets, les tables et les champs qui sont appelés dans des cas spécifiques. Cette section énumère toutes les possibilités de déclenchement.

15.29.1 onOpen

Après l'ouverture d'un projet, MUIbase recherche dans le programme du projet une fonction appelée **onOpen**. Si une telle fonction existe alors elle est appelée sans aucun argument.

Exemple

```
(DEFUN onOpen ()
  (ASKBUTTON NIL "Merci de m'ouvrir !" NIL NIL)
)
```

Voir aussi **onClose**, **onAdminMode**, **onChange**, la démo 'Trigger.mb'.

15.29.2 onClose

Avant la fermeture d'un projet, MUIbase recherche dans le programme du projet une fonction appelée **onClose**. Si une telle fonction existe alors elle est appelée sans argument. Dans la version courante le résultat de ce déclencheur est ignoré et le projet est fermé sans se soucier de sa valeur de retour.

Si vous effectuez des modifications du projet dans la fonction **onClose**, alors MUIbase vous demandera d'abord de sauvegarder le projet avant de le fermer réellement. Si vous utilisez le menu 'Projet - Sauver & Fermer' pour la fermeture d'un projet, le déclenchement sera appelé avant de sauver le projet, ainsi les changements seront sauvés automatiquement.

Exemple

```
(DEFUN onClose ()
  (ASKBUTTON NIL "Au revoir !" NIL NIL)
)
```

Voir aussi **onOpen**, **onChange**, demo 'Trigger.mb'.

15.29.3 onAdminMode

À chaque fois qu'un projet passe en mode administrateur ou utilisateur et qu'une fonction appelée **onAdminMode** existe dans un programme projet alors cette fonction est appelée. La fonction reçoit un argument *admin* indiquant si le projet est en mode administrateur (*admin* n'est pas NIL) ou en mode utilisateur (*admin* est NIL).

Exemple

```
(DEFUN onAdminMode (admin)
  (IF admin
    (ASKBUTTON NIL "Maintenant en mode administrateur" NIL NIL)
    (ASKBUTTON NIL "Revenu en mode utilisateur" NIL NIL)
  )
)
```

Voir aussi `onOpen`, `onChange`, `SETADMINMODE`, la démo `'Users.mb'`.

15.29.4 onChange

A chaque fois que l'utilisateur fait des changements sur un projet ou après avoir enregistré un projet, MUIbase recherche dans le programme du projet une fonction appelée `onChange`. Si une telle fonction existe, elle est exécutée sans argument. Ce qui peut être utilisé pour compter les changements faits par l'utilisateur sur un projet.

Exemple

```
(DEFUN onChange ()
  (SETQ Control.NumChanges (CHANGES))
)
```

Dans l'exemple ci-dessus `'Control.NumChanges'` pourrait être un champ virtuel quelque part dans une table de type `'Exactement un enregistrement'` pour afficher le nombre de modifications du projet.

Voir aussi `onOpen`, `onClose`, `onAdminMode`, la démo `'Trigger.mb'`.

15.29.5 Déclencheur de création

Quand l'utilisateur veut allouer un nouvel enregistrement en sélectionnant l'un des éléments du menu `'Nouvel enregistrement'` ou `'Dupliquer l'enregistrement'` et qu'un déclencheur de `'Création'` a été spécifié pour cette table, cette fonction de déclenchement est exécutée. Le déclencheur de `'Création'` peut être spécifié dans la fenêtre de saisie des tables (voir [Section 14.1.1 \[Creating tables\], page 60](#)).

Le déclencheur reçoit NIL ou un pointeur d'enregistrement en tant que premier et seul argument. NIL signifie que l'utilisateur veut allouer un nouvel enregistrement tandis qu'un pointeur d'enregistrement signifie que l'utilisateur veut dupliquer cet enregistrement. Si le déclencheur possède plus d'un argument alors ceux-ci seront initialisés avec NIL. Le déclencheur doit allouer le nouvel enregistrement en appelant la fonction `NEW` (voir [Section 15.18.1 \[NEW\], page 127](#)). Le résultat retourné par le déclencheur sera examiné et s'il renvoie un pointeur d'enregistrement alors l'enregistrement sera affiché.

Le déclencheur de `'Création'` est également exécuté lorsqu'un programme MUIbase appelle la fonction `NEW*` (voir [Section 15.18.2 \[NEW*\], page 128](#)).

Exemple de déclencheur de création

```
(DEFUN nouvelEnregistrement (init)
  (PROG1
    (NEW Table init)
    ...
  ) ; pour renvoyer le résultat de NEW
)
```

```
)
)
```

Voir aussi NEW, NEW*, Delete trigger.

15.29.6 Déclencheur de suppression

Lorsque l'utilisateur veut supprimer un enregistrement en sélectionnant l'élément du menu 'Supprimer l'enregistrement' et qu'un déclencheur de 'Suppression' a été spécifié pour cette table, alors cette fonction de déclenchement est exécutée. Le déclencheur de 'Suppression' peut être spécifié dans la fenêtre de saisie des tables (voir [Section 14.1.1 \[Creating tables\]](#), page 60).

Le déclencheur reçoit un argument booléen comme seul argument. Si l'argument est différent de NIL alors la fonction demande à l'utilisateur s'il veut vraiment supprimer l'enregistrement. Si c'est le cas, le déclencheur doit appeler DELETE (voir [Section 15.18.3 \[DELETE\]](#), page 128) pour supprimer l'enregistrement.

Le déclencheur de 'Suppression' est également appelé lorsqu'un programme MUIbase appelle la fonction DELETE* (voir [Section 15.18.4 \[DELETE*\]](#), page 129).

Exemple de déclencheur de suppression

```
(DEFUN supprimerEnregistrement (confirmation)
  (DELETE Table confirmation)
)
```

Voir aussi DELETE, DELETE*, New trigger.

15.29.7 Fonction de comparaison

Pour trier les enregistrements d'une table vous pouvez utiliser une fonction de comparaison. Voir [Section 10.4 \[Changing orders\]](#), page 48, pour savoir comment doit être spécifiée une telle fonction pour une table. La fonction prend deux pointeurs d'enregistrements en arguments et renvoie une valeur entière reflétant l'ordre de tri des deux enregistrements. La fonction de comparaison doit renvoyer une valeur inférieure à 0 si son premier argument est plus petit que le second, 0 s'ils sont équivalents et une valeur supérieure à 0 si le premier argument est plus grand que le second.

Par exemple, si vous avez une table 'Personnes' avec un champ de type chaîne 'Nom' alors vous pourriez utiliser la fonction suivante pour comparer deux enregistrements :

```
(DEFUN cmpPersonnes (rec1:Personnes rec2:Personnes)
  (CMP rec1.Nom rec2.Nom)
)
```

Ceci triera tous les enregistrements selon le champ 'Nom' en utilisant la comparaison de chaîne sensible à la casse. Notez qu'en utilisant une liste de champs vous ne pourriez pas obtenir le même tri car, dans le cas des listes de champs, une comparaison de chaînes sensible à la casse est effectuée.

En utilisant une fonction de comparaison vous pouvez définir des relations d'ordre très complexes. Faites attention à ne pas créer de fonctions récursives qui s'appelleraient elles-mêmes. MUIbase arrêtera l'exécution du programme et vous affichera un message d'erreur si vous essayez de faire cela. En outre, vous ne devez pas utiliser de commandes à effets de bord, par exemple positionner la valeur d'un champ.

En utilisant une fonction de comparaison, MUIbase ne sait pas toujours lorsqu'il doit réordonner les enregistrements. Par exemple si l'on considère dans l'exemple précédent une nouvelle table 'Jouets' possédant un champ de type chaîne 'Nom' et une référence 'Proprietaire' vers 'Personnes' et la fonction suivante pour comparer les enregistrements :

```
(DEFUN cmpJouets (rec1:Jouets rec2:Jouets)
  (CMP* rec1.Proprietaire rec2.Proprietaire)
)
```

Cette fonction utilise le tri des 'Personnes' pour déterminer l'ordre de tri des enregistrements, par conséquent les enregistrements de 'Jouets' sont classés selon le tri des 'Personnes'.

Maintenant si l'utilisateur modifie un enregistrement de la table 'Personnes' et que la position (dans le tri) de cet enregistrement change, alors tous les enregistrements de dans 'Jouets' se rapportant à cet enregistrement ont besoin d'être réordonnés. Cependant, MUIbase ne connaît pas cette dépendance.

En plus d'utiliser le menu 'Table - Réordonner tous les enregistrements' pour rétrier les enregistrements de la table 'Jouets', vous pouvez mettre en place une réorganisation automatique en spécifiant le déclencheur de champ suivant sur le champ 'Nom' de la table 'Personnes':

```
(DEFUN setNom (nouvelleValeur)
  (SETQ Personnes.Nom nouvelleValeur)
  (FOR ALL Jouets WHERE (= Jouets.Proprietaire Personnes) DO
    (SETISSORTED Jouets NIL)
  )
  (REORDER Jouets)
)
```

La fonction supprime l'information de tri pour tous les enregistrements se rapportant à l'enregistrement courant de la table 'Personnes' et réordonné alors tous les enregistrements non triés de la table 'Jouets'.

Voir aussi Order, CMP, GETISSORTED, SETISSORTED, REORDER, REORDER-ALL, GETORDERSTR, SETORDERSTR, la démo 'Order.mb'.

15.29.8 Déclencheur de champ

Dans la fenêtre de création de champs (voir [Section 14.2.1 \[Creating attributes\], page 62](#)) vous pouvez définir une fonction qui sera déclenchée toutes les fois que l'utilisateur voudra modifier le contenu d'un champ : le déclencheur de champ.

Si vous avez défini une telle fonction pour un champ et que l'utilisateur modifie la valeur de ce champ alors le contenu de l'enregistrement ne sera pas automatiquement mis à jour avec la nouvelle valeur. Au lieu de cela la valeur est passée en paramètre au déclencheur. Le déclencheur peut alors vérifier la valeur et éventuellement la refuser. Pour stocker la valeur dans un enregistrement vous devez utiliser la fonction SETQ.

Le déclencheur doit renvoyer le résultat de l'appel à SETQ ou l'ancienne valeur du champ s'il décide de refuser la nouvelle.

Le déclencheur est également appelé lorsqu'un programme MUIbase fait appel à la fonction SETQ* (voir [Section 15.6.5 \[SETQ*\], page 86](#)) pour positionner la valeur d'un champ.

Exemple de déclencheur de champ

```
(DEFUN setMontant (montant)
  (IF une-expression
    (SETQ Table.Montant montant)
    (ASKBUTTON NIL "Valeur Invalide !" NIL NIL)
  )
  Table.Montant
)
; retourne la valeur courante

Voir aussi SETQ*
```

15.29.9 Programmation de champs virtuels

Dans MUIbase les champs virtuels sont des champs spéciaux qui calculent leur valeur à chaque fois que c'est nécessaire. Par exemple si vous passez à un autre enregistrement en cliquant sur le bouton flèche dans le panneau de contrôle d'une table. Le champ virtuel de cette table sera alors automatiquement recalculé et affiché (les réglages appropriés pour le champ virtuel étant fournis, voir [Section 14.3.3 \[Attribute object editor\], page 67](#)). Pour calculer la valeur du champ, le déclencheur de 'Calcul' est appelé. Ce déclencheur peut être spécifié dans la fenêtre de saisie de champ (voir [Section 14.2.2 \[Type specific settings\], page 62](#)). La valeur renvoyée par ce déclencheur définit la valeur du champ virtuel. Si vous ne spécifiez aucun déclencheur de 'Calcul' pour un champ virtuel, alors la valeur du champ est NIL

Vous pouvez également déclencher le calcul d'un champ virtuel en y accédant simplement à partir d'un programme MUIbase. Ainsi par exemple, si vous avez un bouton qui doit calculer la valeur d'un champ virtuel, vous devrez seulement spécifier une fonction pour le bouton comme celle qui suit :

```
(DEFUN buttonHook ()
  champ-virtuel
)
```

Vous pouvez également donner à un champ virtuel n'importe quelle valeur en utilisant la fonction SETQ :

```
(SETQ champ-virtuel expr)
```

Cependant si vous accédez au champ virtuel après l'appel de SETQ, la valeur du champ virtuel sera recalculée.

Il n'y a pas de mécanisme de cache de la valeur d'un champ virtuel parce qu'il n'est pas facile de savoir quand la valeur doit ou ne doit pas être recalculée. Par conséquent, il est préférable d'accéder rarement aux champs virtuels et d'en mémoriser la valeur dans des variables locales pour un usage ultérieur.

Pour un exemple sur la façon d'utiliser les champs virtuels veuillez consulter la démo 'Movie.mb'.

Voir aussi Champs virtuels, la démo 'Movie.mb'.

15.29.10 Fonction de calcul d'activation

Pour les objets champs et les boutons de fenêtre il est possible de spécifier une fonction pour calculer l'état d'activation de l'objet. Voir [Section 14.3.3 \[Attribute object editor\], page 67](#) et [Section 14.3.10 \[Window editor\], page 73](#), pour savoir comment spécifier ce déclencheur.

La fonction déclencheur est appelée sans argument. Elle doit renvoyer NIL pour désactiver l'objet et toute autre valeur pour l'activer.

Par exemple la fonction d'activation d'un objet qui est activé lorsqu'un champ virtuel de type 'Liste' a un élément sélectionné ressemblerait à :

```
(DEFUN activeObjet ()
  (GETVIRTUALLISTACTIVE champ-liste-virtuel)
)
```

Voir aussi Éditeur d'objet champ, Éditeur de fenêtre, la démo 'Users.mb'.

15.29.11 Déclencheur de double-clic

Pour les champs virtuels utilisant un affichage de type liste pour leurs contenus, il est possible de spécifier une fonction exécutée toutes les fois que l'utilisateur double clique sur un élément de la liste. Voir [Section 14.3.3 \[Attribute object editor\], page 67](#), pour savoir comment spécifier ce déclencheur sur un objet champ virtuel.

Ces déclencheurs sont appelés avec trois arguments. Le premier argument contient le numéro de ligne du champ cliqué, commençant par 1 pour la première ligne (la ligne 0 se rapporte à l'en-tête de la liste). Le deuxième argument contient le numéro de colonne en commençant par 0. Le troisième argument est un pointeur d'enregistrement à partir auquel correspond l'élément de la liste ou NIL s'il n'a pas été produit directement à partir d'un enregistrement. Le code retour de la fonction est ignoré.

L'exemple typique de déclencheur de double-clic est le suivant :

```
(DEFUN declencheurDoubleClic (lig col enr:Table)
  ...
)
```

Ici *enr* est déclaré en tant que pointeur d'enregistrement vers la table *Table*. De cette manière il est possible d'accéder aux champs de *Table* à partir de *enr*.

Au cas où l'argument enregistrement pourrait appartenir à plusieurs tables, la construction suivante utilisant les prédicats de typage pour différencier les tables peut être utile.

```
(DEFUN doubleClickTrigger (lig col enr)
  (COND
    ((RECP Table1 enr) (SETQ Table1 enr) ...)
    ((RECP Table2 enr) (SETQ Table2 enr) ...)
    ...
  )
)
```

L'élément de la liste sur lequel l'utilisateur a cliqué peut ne se rapporter à aucun enregistrement. Dans ce cas-là, le troisième argument peut être ignoré et l'accès à l'élément de la liste se fait comme dans l'exemple suivant :

```
(DEFUN declencheurDoubleClic (lig col)
  (PRINT (NTH col (NTH lig attr-virtuel)))
)
```

Voir aussi Éditeur d'objet champ, la démo 'Movie.mb'.

15.29.12 Calculer les étiquettes des listes

Pour les champs chaînes, l'objet graphique peut contenir une liste déroulante permettant à l'utilisateur de choisir parmi une liste de chaînes. Les étiquettes de cette liste peuvent être statiques ou être calculées dynamiquement par un déclencheur. Voir [Section 14.3.3 \[Attribute object editor\], page 67](#), pour avoir des informations sur la façon de choisir entre des étiquettes statiques ou dynamiques et de spécifier la fonction déclencheur.

Le déclencheur pour le calcul des étiquettes ne requiert aucun argument. Elle doit retourner un mémo avec une étiquette par ligne ou NIL pour aucune étiquette.

Par exemple la fonction de calcul pourrait ressembler à ceci :

```
(DEFUN calculerEtiquettes ()
  "Tokyo\nMunich\nLos Angeles\nRome"
)
```

Voir aussi Calculer les enregistrements référencés, Éditeur d'objet champ.

15.29.13 Calculer les enregistrements référencés

Pour les champs référence, l'objet graphique possède généralement un bouton permettant d'ouvrir une liste d'enregistrements parmi lesquels l'utilisateur peut faire son choix. La liste de ces enregistrements peut être calculée par un déclencheur. Voir [Section 14.3.3 \[Attribute object editor\], page 67](#), pour avoir des informations sur la manière de spécifier le déclencheur pour les champs références. on how to specify

La fonction de calcul de la liste d'enregistrements ne requiert aucun argument. Elle doit retourner une liste devant contenir des enregistrements de la table référencée. Tout enregistrement de cette table est ajouté à la liste affichée. Les éléments qui ne sont pas des enregistrements de la table référencée sont ignorés silencieusement.

L'exemple suivant illustre une fonction typique de calcul d'enregistrements référencés. Disons qu'un projet contient une table 'Personne' avec un champ booléen 'Femme'. Alors la fonction de calcul suivante n'affiche que les personnes féminines dans la liste déroulante :

```
(DEFUN calculerEnregistrementsFemme ()
  (SELECT Personen FROM Personne WHERE Femme)
)
```

Voir aussi Calculer les étiquettes des listes, Éditeur d'objet champ.

15.30 Liste des fonctions obsolètes

Les fonctions suivantes sont obsolètes depuis la version 2.7 de MUIbase.

- GETDISABLED
- SETDISABLED
- GETWINDOWDISABLED
- SETWINDOWDISABLED

Les fonctions obsolètes ne fonctionnent plus comme attendu et leur appel est soit ignoré (donnant une non-opération), soit ouvre une fenêtre d'avertissement, soit cause une erreur selon le réglage du menu 'Programme - Fonctions obsolètes' (voir [Section 7.5.6 \[Obsolete functions\], page 38](#)).

Il est recommandé d'enlever ces fonctions des programmes et d'implémenter la fonctionnalité en utilisant le réglage activé/désactivé des objets champ et des boutons de fenêtre (voir [Section 15.29.10 \[Compute enabled function\]](#), page 150).

16 Interface ARexx

L'interface ARexx n'existe que dans la version Amiga de MUIbase.

ARexx est une interface standard pour des programmes Amiga et qui permet d'accéder aux fonctions et données d'autres programmes. MUIbase fournit un port ARexx avec un nombre de commandes restreint mais néanmoins suffisant pour permettre à un programme externe d'effectuer des opérations comme si c'était MUIbase lui-même qui les effectuaient. De plus l'interface ARexx de MUIbase possède un mécanisme de transaction similaire à d'autres bases de données relationnelles.

Des exemples de scripts ARexx pour MUIbase sont disponibles dans le tiroir 'rex'.

16.1 Nom du port

Le nom du port ARexx dans MUIbase est nommé 'MUIbase.*n*' où *n* est un compteur qui démarre à 1. Si vous lancez une seule fois MUIbase, le nom du port sera 'MUIbase.1'.

Vous devez déclarer le nom du port ARexx via `address` avant de lancer l'une des commandes ARexx de MUIbase. Le bout de programme suivant vous montre comment vérifier la présence du port ARexx dans MUIbase, comment lancer MUIbase le cas échéant ainsi que comment interagir avec le port.

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

address MUIbase.1
```

Consultez également l'exemple de script ARexx 'address.rex'.

16.2 Syntaxe des commandes

Après avoir contacté le port ARexx de MUIbase, vous pourrez lancer n'importe quelle commande ARexx de MUIbase. La syntaxe est similaire à celle d'autres implémentations :

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

où *cmd* est une des commandes décrites un peu plus bas dans ce chapitre, et *arg1* ... sont des arguments optionnels de cette commande.

Puisque l'interpréteur ARexx évalue la ligne de commande avant de l'envoyer à MUIbase, il peut être utile de mettre entre guillemets certains, voire tous les arguments. Il est recommandé d'utiliser des guillemets simples (') pour les arguments qui ne seront plus utilisés ultérieurement par l'interpréteur ARexx. Ainsi vous pouvez toujours utiliser des guillemets doubles (") pour des arguments, par exemple pour des constantes de texte. De plus, vous pouvez intégrer la valeur des variables ARexx en leur enlevant leurs guillemets. Voici un exemple utilisant la commande MUIbase `eval` :

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

Consultez également Eval.

16.3 Codes retour

Après avoir lancé l'une des commandes ARexx de MUIbase, plusieurs variables ARexx sont mises à jour avec le résultat de la commande. Pour afficher tous les résultats d'une commande, vous devez activer l'option de résultats ARexx en ajoutant les lignes suivantes au début de votre script ARexx :

```
options results
```

Il existe 3 variables ARexx qui peuvent être définies par l'interface ARexx de MUIbase : *rc*, *results* et *lasterror*. La variable *rc* est toujours définie et montre le succès ou l'échec d'une commande. Si une commande est lancée avec succès, la variable *results* est initialisé avec le résultat de la commande alors que dans le cas d'un échec de la commande, la variable *lasterror* peut contenir des informations supplémentaires décrivant l'erreur.

Pour la variable *rc* il existe les codes de retour suivants :

Code retour Signification

0	Succès. La variable <i>result</i> contient le résultat actuel.
-1	Erreur d'implémentation. Ne devrait jamais se produire.
-2	Manque de mémoire.
-3	Commande ARexx inconnue.
-4	Erreur de syntaxe.
<= -10	Autre erreur. Une description de l'erreur est donnée par <i>lasterror</i> .
-12	Erreur de compilation (uniquement pour la commande <i>compile</i>).

Dans le cas de retour *rc* <= -10, la variable *lasterror* contient une description valide de l'erreur. Des codes d'erreur supplémentaires pourraient être ajoutés dans le futur de façon à avoir un rapport d'erreur plus détaillé.

Voici un bout de code typique qui montre comment évaluer le résultat d'une commande ARexx de MUIbase :

```
eval handle 'select * from Accounts'
if (rc == 0) then
    say result
else if (rc == -1) then
    say "Erreur d'implémentation"
else if (rc == -2) then
    say "Manque de mémoire"
else if (rc == -3) then
    say "Commande inconnue"
else if (rc == -4) then
    say "Erreur de syntaxe"
else if (rc <= -10) then
    say lasterror
else
    say "Erreur: " rc
```

16.4 Quit

La commande `quit` provoque la fermeture du programme MUIbase. Veuillez également consulter la documentation de MUI.

16.5 Hide

La commande `hide` iconifie toutes les fenêtres ouvertes de MUIbase. Veuillez également consulter la documentation de MUI.

16.6 Show

La commande `show` désiconifie MUIbase et rouvre les fenêtres. Veuillez également consulter la documentation de MUI.

16.7 Info

La commande `info` retourne les renseignements suivants à propos de l'application MUI : titre, concepteur, droits de copie, description, version, port et écran.

Commande	Valeur de <i>result</i>
<code>info title</code>	Titre de l'application
<code>info author</code>	Concepteur de l'application
<code>info copyright</code>	Message sur les droits de copie
<code>info description</code>	Description
<code>info version</code>	Numéro de version
<code>info base</code>	Nom du port ARexx
<code>info screen</code>	Nom de l'écran public

Veuillez également consulter la documentation de MUI.

16.8 Help

La commande `help` écrit dans un fichier toutes les commandes ARexx disponibles de l'application MUI.

`help nom-fichier`

Les commandes ARexx sont affichées en respectant la syntaxe l'interpréteur de commandes AmigaDOS. Veuillez également consulter la documentation de MUI ainsi que le manuel de l'AmigaDOS pour la syntaxe de la ligne de commande.

16.9 Compile

La commande `compile` compile le source d'un programme externe.

`compile source [update]`

Cette commande compile le fichier source d'un programme externe au projet et dont le nom de fichier spécifié par commande `source`. En cas de succès, la commande retourne la

valeur 0, et si **update** est spécifié, le fichier source externe est réécrit. La mise à jour du fichier source permet la mise en évidence des mots clefs MUIbase. Un programme compilé avec succès est considéré comme le programme du projet et utilisé lors de l'exécution de déclencheurs.

Dans le cas d'un échec lors de la compilation, le code d'erreur -12 est retourné et *lasterror* est défini sur 4 lignes de texte :

- La première ligne contient le nom du fichier qui a généré l'erreur.
- La deuxième ligne contient le numéro de la ligne où l'erreur s'est produite en commençant à 1.
- La troisième ligne contient le numéro de la colonne où l'erreur s'est produite en commençant à 1.
- La quatrième ligne décrit l'erreur sous forme de texte compréhensible.

Un projet doit absolument être déjà ouvert avant de lancer la commande **compile** pour compiler son source externe. Si aucun projet externalisant son source dans le fichier spécifié par *source* n'est trouvé, un code d'erreur ≤ -10 (mais différent de -12) est retourné et *lasterror* est défini.

16.10 Connect

La commande **connect** ouvre la communication vers un projet MUIbase.

connect *nom-projet* [GUI]

La commande vérifie en premier lieu si le projet défini par **nom-projet** est déjà ouvert et l'ouvre le cas échéant. Un projet n'est lancé qu'une seule fois et les diverses connections vers le même projet partagent l'accès à la base de données. Ensuite un handle de communication (ou identifiant unique) est généré. Un handle de communication est une valeur entière non nulle. Si le mot-clé **GUI** fait parti de la ligne de commande, alors l'interface graphique du projet MUIbase est également lancée. Sinon aucune interface graphique n'apparaît ce qui permet d'exécuter des commandes ARexx de MUIbase en tâche de fond sans l'intervention directe de l'utilisateur.

En cas de succès dans l'exécution d'une commande, la valeur 0 est retournée et *result* prend la valeur du handle.

Exemple : `'connect "MUIbase:Demos/Movies.mb"'` établit une connexion vers la base de données d'exemple des films.

Consultez également Disconnect, Connexions, Codes retour.

16.11 Disconnect

La commande **disconnect** termine une communication existante.

disconnect *handle*

Termine la connexion à la base de donnée fournie par *handle*. Si c'est la seule connexion vers le projet référencé par *handle* et si le projet ne dispose d'aucune interface graphique, alors le projet peut être fermé et déchargé de la mémoire. Sinon le projet reste ouvert.

Exemple: `'disconnect 1'` termine la connexion ayant le handle 1.

Consultez également Connect, Connexions, Codes retour.

16.12 Connections

La commande `connections` donne des renseignements sur les connexions existantes.

```
connections
```

En cas de succès de la commande, `connections` retourne la valeur 0 et définit *result* sous forme de texte compréhensible dont chaque ligne est une connexion avec une valeur de handle et un nom de projet.

Exemple: la variable *result* après un appel à `connections` pourrait se présenter sous la forme suivante :

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

Consultez également `Connect`, `Disconnect`, `Codes retour`.

16.13 Eval

L'interface principale du port Arexx de MUIbase permettant la récupération et la mise à jour des données est gérée par la commande `eval`.

```
eval handle commande-lisp
```

La commande `eval` exécute la commande spécifiée par *commande-lisp* (écrite dans le langage lisp de MUIbase) sur le projet correspondant à *handle*. Un handle peut être obtenu via la commande `connect`. La commande *commande-lisp* peut être n'importe quelle expression du langage de programmation de MUIbase. La parenthèse la plus externe d'une expression est optionnelle et peut éventuellement être omise. Il est recommandé d'entourer *commande-lisp* par des guillemets simples comme décrit dans [Section 16.2 \[Syntaxe des commandes\]](#), page 154.

En cas de succès, `eval` retourne la valeur 0 et définit *result* comme une représentation textuelle de la valeur de retour de *commande-lisp*. La représentation textuelle est faite de façon à avoir une idée sur le type de donnée retourné, par exemple un texte est entouré par des guillemets doubles alors qu'une liste est entourée de parenthèses dont les éléments sont séparés par des espaces ou des caractères retour chariot. Si vous voulez avoir un format spécifique, utilisez votre propre format dans la commande lisp spécifiée.

Si vous modifiez la base de données via la commande `eval` sans avoir auparavant lancé une transaction (voir [Section 16.14 \[Transaction\]](#), page 159) les changements seront alors automatiquement rendus permanents (validation automatique). A l'inverse si vous avez lancé une transaction avant d'appeler `eval` les changements seront gardés en mémoire jusqu'à ce que la commande `commit` les rende permanents ou que la commande `rollback` les annule.

Exemple:

```
options results
address MUIbase.1
connect "MUIbase:Demos/Movie.mb"
if (rc == 0) then
do
    handle = result
```

```

        eval handle 'select Titre, Réalisateur from Movies'
    end
    if (rc == 0) then
        say result
    
```

Le résultat de l'exemple ci-dessus pourrait ressembler à ceci :

```

( ( "Titre" "Réalisateur" )
  ( "Batman" "Tim Burton" )
  ( "Batman Returns" "Tim Burton" )
  ( "Chérie, vote pour moi" "Ron Underwood" )
  ( "Tequila Sunrise" "Robert Towne" )
  ( "Mad Max" "George Miller (II)" )
  ( "Braveheart" "Mel Gibson" )
  ( "2010 - L'année du premier contact (L'odyssée continue)" "Peter Hyams" ) )
    
```

Consultez également `Connect`, `Syntaxe des commandes`, `Codes retour`, `Transaction`, `Commit`, l'exemple de script ARexx `'movies.rexx'`.

16.14 Transaction

Le port ARexx de MUIbase permet d'effectuer des transactions dans la base de données. Une transaction consiste en un ensemble d'opérations appliquées sur la base de données permettant la modification des données. Une transaction peut être soit exécutée et rendue permanente définitivement (validation), soit au contraire annulée à n'importe quel point de la transaction (annulation). On lance une transaction avec la commande suivante :

```
transaction handle
```

où *handle* fait référence à un projet obtenu par la commande `connect` (voir [Section 16.10 \[Connect\]](#), page 157).

Après lancement de la commande `transaction` vous pouvez mettre autant de commandes `eval` que vous le désirez sans affecter la base de données. Pourtant il va vous falloir décider à un moment si vous voulez rendre les modifications permanentes (voir [Section 16.15 \[Commit\]](#), page 159) ou revenir en arrière, avant le lancement de la commande `transaction` (voir [Section 16.16 \[Rollback\]](#), page 160).

Après lancement de la commande `transaction` l'accès au projet correspondant est rendu exclusif au *handle* spécifié. Ainsi, les autres programmes voulant accéder à la base de données (y compris l'utilisateur en passant par l'interface graphique) sont bloqués (ou différés dans le cas d'une autre connexion ARexx) jusqu'à ce que l'accès exclusif soit levé, en exécutant la commande `commit` ou `rollback`.

Habituellement, la commande `transaction` retourne la valeur 0. Si une autre connexion ARexx requiert l'accès exclusif au même projet spécifié par *handle*, la demande est bloquée jusqu'à ce que l'autre connexion se termine soit en validant, soit en annulant la transaction.

Consultez également `Eval`, `Commit`, `Rollback`, `Codes retour`.

16.15 Commit

La commande `commit` s'utilise à la fin d'une transaction pour valider les changements de manière définitive.

`commit handle`

La commande `commit` met fin à une transaction (voir [Section 16.14 \[Transaction\], page 159](#)) en sauvegardant le projet auquel fait référence *handle*. La valeur 0 est retournée en cas de succès de la commande `commit`. Si aucune transaction n'a été lancée avant l'appel de la commande `commit` ou s'il se produit une erreur, une valeur différente de 0 est retournée.

Consultez également Rollback, Transaction, Codes retour.

16.16 Rollback

Pour annuler les changements d'une transaction, utilisez la commande `rollback`.

`rollback handle`

Tous les changements effectués dans un projet référencé par *handle* depuis le début de la transaction en cours (voir [Section 16.14 \[Transaction\], page 159](#)) sont annulés et le projet revient à l'état où il se trouvait avant le lancement de la transaction. Une valeur de 0 indique le succès de la commande `rollback`.

Consultez également Commit, Transaction, Codes retour.

Remerciements

Merci à :

- Ralph Reuchlein (Ralphie) pour les rapports de bug, les idées et suggestions, ainsi que pour la traduction allemande du manuel de MUIbase.

Ralphie a également créé et maintient la page web de MUIbase <http://muibase.sourceforge.net>.

- Pascal Marcelin pour sa foi en MUIbase et l'Amiga, et pour le premier serveur web se connectant à MUIbase via son port ARexx.
- Christoph Poelzl et Sébastien Poelzl pour les graphismes utilisés dans MUIbase, pour le jeu d'icônes PNG et pour les tests sous MorphOS.
- Alexandre Balaban pour la traduction française du manuel utilisateur (beaucoup aidé par Lionel Muller et Gilles Mathevet) et pour le portage de MUIbase sur AmigaOS 4.
- Ilkka Lehtoranta pour avoir terminé le portage MorphOS.
- Thomas Fricke et Adrian Maleska pour les divers graphismes améliorant l'apparence de MUIbase.
- Martin Merz pour les icônes Masons.
- Mats Granstrom pour les bêta-tests de MUIbase et pour l'écriture du tutoriel.
- Henning Thilemann, Joseph Durchalet, André Schenk, Klaus Gessner, et Oliver Roberts pour les idées et les bêta-tests.
- Jernej Simoncic pour l'autorisation d'utiliser son script d'installation Windows de The Gimp comme base pour celui de MUIbase pour Windows.

Auteur

MUIbase est développé par :

Steffen Gutmann
3-3-15 Shirokane 301, Minato-ku,
Tokyo 108-0024
Japon

Email: muibase@yahoo.com

Index des fonctions

#

#define	76
#elif	78
#else	78
#endif	78
#if	77
#ifdef	77
#ifndef	77
#include	77
#undef	77

*

*	99
---------	----

+

+	98
---------	----

-

-	98
---------	----

/

/	99
---------	----

<

<	97
<*	97
<=	97
<=*	97
<>	97
<>*	97

=

=	97
=*	97

>

>	97
>*	97
>=	97
>=*	97

1

1+	99
1-	99

A

ABS	100
ADDMONTH	113
ADDYEAR	114
ADMINPASSWORD	138
AND	96
APPEND	116
APPLY	87
ASC	107
ASKBUTTON	121
ASKCHOICE	118
ASKCHOICESTR	119
ASKDIR	117
ASKFILE	117
ASKINT	118
ASKMULTI	121
ASKOPTIONS	120
ASKSTR	118
ATTRNAME	131

C

CASE	87
CHANGES	138
CHR	107
CMP	97
CMP*	98
COMPLETE	142
COMPLETEADD	141
COMPLETMAX	141
CONCAT	106
CONCAT2	106
COND	88
CONS	114
CONSP	92
COPYREC	131
COPYSTR	106

D

DATE	95
DATEDMY	113
DATEP	92
DAY	112
DEFUN	83
DEFUN*	84
DEFVAR	84
DEFVAR*	84
DELETE	128
DELETE*	129
DELETEALL	129
DIRNAME	141
DIV	99
DO	89

DOLIST	89
DOTIMES	88

E

EDIT	139
EDIT*	139
ERROR	92
EXIT	91

F

FCLOSE	124
FEOF	125
FERROR	125
FFLUSH	127
FGETCHAR	126
FGETCHARS	126
FGETMEMO	126
FGETSTR	126
FILENAME	140
FILLMEMO	111
FIRST	115
FOPEN	123
FOR ALL	90
FORMATMEMO	112
FPRINTF	125
FPUTCHAR	127
FPUTMEMO	127
FPUTSTR	127
FSEEK	125
FTELL	126
FUNCALL	87

G

GC	142
GETADMINMODE	138
GETDISABLED	152
GETFILTERACTIVE	134
GETFILTERSTR	134
GETISSORTED	130
GETLABELS	131
GETMATCHFILTER	129
GETORDERSTR	132
GETVIRTUALLISTACTIVE	137
GETWINDOWDISABLED	152
GETWINDOWOPEN	137

H

HALT	92
------------	----

I

IF	87
INDENTMEMO	112
INDEXBRK	103

INDEXBRK*	103
INDEXSTR	102
INDEXSTR*	102
INSMIDSTR	102
INT	94
INTP	92

L

LAST	115
LEFTSTR	101
LEN	101
LENGTH	115
LET	85
LIKE	107
LINE	110
LINES	110
LIST	115
LISTP	92
LISTTOMEMO	111
LISTTOSTR	105
LOWER	107

M

MAPFIRST	116
MAX	100
MAXLEN	131
MEMO	93
MEMOP	92
MEMOTOLIST	111
MESSAGE	141
MIDSTR	102
MIN	100
MOD	100
MONTH	112
MONTHDAYS	113

N

NEW	127
NEW*	128
NEXT	90
NOT	96
NOW	114
NTH	115
NULL	92

O

onAdminMode	146
onChange	147
onClose	146
onOpen	146
OR	96

P

PRINT	124
PRINTF	124
PROG1	85
PROGN	85
PROJECTNAME	137
PUBSCREEN	142

R

RANDOM	101
REAL	94
REALP	92
RECNUM	130
RECORD	135
RECORDS	135
RECP	92
REMCCHARS	104
REORDER	133
REORDERALL	134
REPLACESTR	104
REPLACESTR*	104
REST	115
RETURN	91
REVERSE	116
RIGHTSTR	101
RINDEXBRK	103
RINDEXBRK*	103
RINDEXSTR	103
RINDEXSTR*	103
ROUND	100

S

SELECT	135
SETADMINMODE	138
SETCURSOR	136
SETDISABLED	152
SETFILTERACTIVE	134
SETFILTERSTR	134
SETISSORTED	130
SETLABELS	132
SETMATCHFILTER	129
SETMIDSTR	102
SETORDERSTR	133

SETQ	86
SETQ*	86
SETVIRTUALLISTACTIVE	137
SETWINDOWDISABLED	152
SETWINDOWOPEN	137
SHA1SUM	106
SORTLIST	116
SORTLISTGT	117
SPRINTF	108
STAT	140
stdout	124
STR	93
STRP	92
STRTOLIST	105
SYSTEM	139
SYSTEM*	140

T

TABlename	132
TACKON	140
TIME	95
TIMEP	92
TODAY	114
TRIMSTR	104
TRUNC	100

U

UPPER	107
-------------	-----

V

VIEW	139
VIEW*	139

W

WORD	105
WORDS	105

Y

YEAR	112
YEARDAYS	113

Index des concepts

A

Accéder aux enregistrements	80
Affichage en tableau	55
ARexx	154
ARexx commit	159
ARexx compile	156
ARexx connect	157
ARexx connections	158
ARexx disconnect	157
ARexx eval	158
ARexx help	156
ARexx hide	156
ARexx info	156
ARexx quit	156
ARexx rollback	160
ARexx show	156
ARexx transaction	159
Associations	24
Associations Plusieurs à plusieurs	25
Associations Un à plusieurs	25
Associations Un à Un	25
Associations 1:1	25
Associations 1:n	25
Associations n:m	25
Attribute object editor	67
Auteur	162

B

Base de données corrompue	32
BetterString	2
Boîte de recherche	50
Boutons	23
Boutons dans cycle de tabulation	35
Buttons	23

C

Cache d'enregistrements	36
Calculer les enregistrements référencés	152
Calculer les étiquettes des listes	152
Champ d'affichage	65
Champ suivant via <Entrée>	35
Champs	20
Champs Bool	21
Champs booléens	21
Champs Choice	21
Champs choix	21
Champs date	22
Champs entiers	21
Champs fichier	20
Champs heure	22
Champs image	20
Champs Integer	21

Champs mémo	22
Champs Memo	22
Champs police	20
Champs real	21
Champs réels	21
Champs Référence	22
Champs String	20
Champs texte	20
Champs Time	22
Champs virtual	22
Champs virtuels	22
Change order	48
Changing attributes	64
Chemins relatifs	37
Code source du programme	37
Code source externe	75
Codes retour ARexx	155
Commandes de définition	83
Confirmer enregistrement et réorganisation	37
Confirmer la sortie	36
Confirmer la suppression d'enregistrement	37
Consommation mémoire	24
Constantes	81
Conventions de nommage	7
Conventions de nommage des symboles dans les programmes	80
Copier MUIbase	1
Copying attributes	64
Création de champs	62
Création de tables	60

D

Décharger les enregistrements	33
Déclenchement de fonction	146
Déclencheur de champ	149
Déclencheur de création	147
Déclencheur de double-clic	151
Déclencheur de suppression	148
Deleting attributes	64
Démarrez MUIbase	7
Distribution	1
Dupliquer un enregistrement	41

E

Editeur de panel	66
Editeur de programme	75
Éditeur de requêtes	55
Editeur de structure	60
Editeur externe	34
Edition d'enregistrement	41
Enregistrements	20
Enregistrer comme défaut	39

Enregistrer le projet	31
Erreurs internes à la base de données	32
Exemples de filtre	46
Exemples de motif de recherche	51
Exemples de requêtes	58
Export structure	74
Exporter des enregistrements	53
Expression de filtrage	45

F

Faire un don	1
Fenêtre principale	27
Fenêtres	27
Fermer le Projet	33
Fiches	27
Fichier de sortie de programme	39
Filtre	45
Filtre d'enregistrement	45
filtre de référence	46
Fonction de calcul d'activation	150
Fonction de comparaison	148
Fonctions booléennes	96
Fonctions d'entrée/sortie	123
Fonctions de comparaison	97
Fonctions de conversion de type	93
Fonctions de demande de saisie	117
Fonctions de manipulation des dates	112
Fonctions de manipulation des heures	112
Fonctions mathématiques	98
Fonctions obsolètes	38, 152
Fonctions sur les chaînes	101
Fonctions sur les champs	131
Fonctions sur les mémos	110
Fonctions sur les tables	132
Format de fichier	30
Formats	34
Formet de fichier pour l'import et l'export	52

G

Gestion de l'affichage	65
Gestion des champs	61
Group editor	72
Groupes	29

I

Icônes	2
Image editor	72
Images	28
Import (Exemple de fichier)	52
Importer des enregistrements	53
Importer et Exporter	52
Impression de requêtes	56
Information de débogage	38
Informations	30
Initial record	20

Installation de MUIbase sur Amiga	6
Installation de MUIbase sur Linux	5
Installation de MUIbase sur Windows	5
Interface graphique	27

L

Label editor	63
Langage de programmation	78
Les constantes prédéfinies	143
Les fonctions GUI	136
Les fonctions projets	137
Les fonctions sur les enregistrements	127
Les fonctions système	138
Les variables prédéfinies	142
Liste de discussion	1
Liste des fonctions	114
Liste des fonctions obsolètes	152

M

Menu contextuel des Mémos	42
Mise à jour d'une version précédente	6
Mise en garde	1
Mode Administrateur	32
Mode Utilisateur	32
Modification de tables	61
Modifier les filtres	45
Mot de passe administrateur	32
MUI	2

N

Nettoyer le projet	30
Nettoyer les sources des programmes externes ..	38
NList	2
Nom de port ARexx	154
Nouveau champ	62
Nouveau projet	30
Nouvel enregistrement	41
Nouvelle table	60

O

Objet actif	41
Objets balance	29
Objets d'espacement	28
Objets de champ	28
Objets texte	28
onAdminMode	146
onChange	147
onClose	146
onOpen	146
Opérateurs relationnels	97
Ordre	47
Ouvrir le projet	31

P

Panels.....	28
Paramètres fonctionnels	143
Parcourir les enregistrements	44
Parties externes	2
Pourquoi Lisp ?	79
Prédicats de typage.....	92
Préférences	34
Préférences MUI.....	36
Préprocesseur	76
Programmation.....	75
Programmation de champs virtuels	150
Projets.....	19

Q

Quitter MUIbase	7
-----------------------	---

R

Recherche	50
Rechercher en avant/en arrière.....	50
Register group editor	73
Registres	29
Réglages dépendants du projet.....	36
Réglages utilisateur	34
Remerciements	161
Reorder all records	49
Réorganisation	31
Répertoire d'inclusion	39
Requêtes Select-from-where.....	55

S

Saisie de valeur de type Référence.....	43
Saisie de valeur NIL	43
Saisie de valeurs booléennes	42
Saisie de valeurs de choix	42
Saisie de valeurs de date	42
Saisie de valeurs entières	42
Saisie de valeurs horaires	42

Sémantique des expressions.....	145
Sorting attributes.....	65
Space editor	72
Spécificateurs de type	144
Structures de contrôle	85
Suppression d'enregistrement	43
Suppression de tables.....	61
Syntaxe des commandes ARexx.....	154
Syntaxe Lisp	79

T

Table courante	41
Table management	60
Tables.....	19
Text editor	72
TextEditor	2
Traitement des données.....	55
Travaux dirigés	9
Tri des tables.....	61
Tri vide	47
Trier les déclencheurs.....	38
Type specific settings.....	62
Type specific settings for attribute objects	68
Types de champ.....	20
Types de champs (tableau)	23
Types de données pour programmer	81
Types de programmes	79
Typographie utilisée	83

V

Vérifier l'intégrité des données	32
Version Amiga.....	2
Version Linux	2
Version Windows	2
Visionneuse externe.....	35

W

Window editor	73
---------------------	----