Fiasco

Release 2.2

Benutzerhandbuch

Nils Bandener
Dekanatsgasse 4
D-34369 Hofgeismar
Deutschland
eMail: nils@dinoex.sub.org

Nils Bandener: Fiasco Release 2.2 Reference Manual

Dokument-Version: 8.9 (6.8.98) Deutsch
Deutsche Version: 8.5 (6.8.98)

Inhaltsverzeichnis

Kapitel 1

Einleitung

Fiasco ist ein neuer Kandidat unter der Masse von Datenbanken für den Amiga. Ursprünglich wollte ich nur ein kleines Programm schreiben, das einem ein paar Vokabeln abhört, die man vorher eingegeben hat. Später kam dann noch die Möglichkeit hinzu, mehr als nur zwei Felder, also Frage und Antwort, zu erstellen. Ab einem gewissen Entwicklungsstadium war dieses Programm einer Datenbank schon so nahe, daß es nur ein paar Änderungen bedurfte, um daraus schon eine Datenbank zu machen. Seitdem (Januar 95) tüftele ich an diesem Meisterstück der Programmierkunst. ;-)

In den Grundlagen unterscheidet sich Fiasco kaum von anderen Datenbanken. Zwar unterstützt Fiasco keine hierarchischen Datenbankstrukturen (wie z.B. eine Art Datenbank in der Datenbank), unterstützt dafür aber Relationen [1]. Seit Release 2.0 unterstützt Fiasco Listview-Felder, die mehrere Einträge für ein Feld enthalten können.

Außerdem kann Fiasco über ARexx gesteuert werden. So können außerdem Programme einzelnen Feldern zugeordnet werden.

Fiasco benutzt Indizes, um die Reihenfolge der Records zu organisieren. Die Implementation der Sortier- und Filter-Funktionen benutzt auch Indizes.

Weiterhin muß Fiasco beim Start nicht die ganze Datenbank laden. Record-Daten werden nur von Disk geladen, wenn sie benötigt werden. Dies macht die Benutzung von Datenbanken möglich, die größer als das verfügbare RAM sind. Um die Zugriffszeit zu verringern, behält Fiasco die geladenen Records im RAM. Die RAM-Ausnutzung von Fiasco Datenbanken kann vom Benutzer kontrolliert werden.

[1] Ich habe bisher nur von relationalen Datenbanken gelesen, so kann ich nicht 100%ig garantieren, daß ich das Prinzip richtig implementiert habe. Auf jeden Fall macht Fiasco sowas ähnliches :-)

Die Maske von Fiasco wird nicht durch eine Grafikdatei definiert, sondern mit internen Grafiken erstellt. So können Fiasco-Masken mit jedem nicht-proportionalen Font benutzt werden. Die Feldtypen, die in einer Maske benutzt werden können, sind breit gesät. Am besten gefällt mir der Datatypes-Feldtyp, mit dem man Grafiken, Animationen, Texte usw. direkt in die Maske einbauen kann.

Neben der Maske kann man die Daten im Listen-Fenster anzeigen. Diese ist wie die Maske frei konfigurierbar. Die Daten können in der Liste jedoch nicht verändert werden[2].

Um nicht den Überblick über Ihre Daten zu verlieren, können Sie Fiascos Suchsystem (siehe Abschnitt 6) benutzen. Die Eingenschaften des Suchsystems sind unter anderem Suche mit mehreren Feldern, Joker, unscharfe Suche und Suche mit Formeln.

Außerdem stehen Sortier-, Filter-, und Zählfunktionen zur Verfügung, die allesamt mit dem Suchsystem verwandt sind und so - kennt man einmal das Suchsystem - keine größeren Probleme darstellen sollten.

1.1 Über diese Anleitung

Um mit Fiasco zu arbeiten, müssen sie nicht die gesamte Anleitung gelesen haben. Viele Funktionen von Fiasco wurden so gestaltet, daß sie so einfach und intuitiv wie möglich benutzt werden können. Um etwaige Mißverständnisse auszuschließen, sollten Sie dennoch Abschnitt 2.4 lesen.

Weiterhin unterstützt Fiasco Online-Hilfe. Das Drücken der Help Taste in einem Requester oder über einem Menüpunkt wird den passenden Abschnitt der AmigaGuide-Version dieser Anleitung anzeigen.

Dieses Handbuch enthält dennoch Beschreibungen und Tutorien für alle Funktionen und Systeme von Fiasco.

Um mehrere der fortgeschrittenen Funktionen von Fiasco zu benutzen, benötigen zusätzliches Wissen. In diesen Fällen ist die Lektüre der passenden Abschnitte in diesem Handbuch sehr empfehlenswert.

Das Kapitel Grundlagen einer Datenbank beschreibt die Prinzipien von Datenbanken und wie sie von Fiasco benutzt werden. Danach beschreibt Einfache Benutzung von Fiasco das Erstellen einer einfachen Datenbank und die Benutzung dieser. Fortgeschrittene Benutzung von Fiasco beschreibt die Eigenschaften von Fiasco, die nicht nötig, aber nützlich für Fiasco-Datenbanken sind.

[2] Achten Sie darauf, die beiden “Listen” von Fiasco nicht durcheinander zu bringen. Das Feld wird in dieser Dokumentation immer Listview und das Fenster Listen-Fenster genannt werden.

Während die oben aufgezählten Kapitel Informationen über Funktionen enthalten, die die Datenbank direkt kontrollieren, enthalten die nächsten Kapitel Erklärungen von zusätzlichen Funktionen, die nicht direkt mit der Datenbank-Struktur zusammenhängen. Um diese Kapitel zu verstehen, benötigen Sie keinerlei Wissen von Fortgeschrittene Benutzung von Fiasco oder von den anderen Kapiteln in diesem Teil.

Suchen in einer Datenbank beschreibt Fiascos Suchfunktion und alle verwanten Funktionen. Drucken einer Datenbank beschreibt die Druckfunktion und erklärt, wie Ausdrucke ohne Benutzung der eingebauten Druckfunktion erstellt werden können. Das Kapitel Import und Export beschreibt nicht etwa Getränke sondern die Import und Export Funktion von Fiasco, die beim Austausch von Daten mit anderen Programmen nützlich ist.

Die Kapitel Feldtypen, Fiascos GUI und Fiascos ARexx-Port sind hauptsächlich zum Nachschlagen gedacht.

1.2 Features

Fiasco kann das alles machen:

- Dynamisches Laden von Records. Fiasco liest Record-Daten nur wenn sie benötigt werden. So können Datenbanken, die größer als das verfügbare RAM sind benutzt werden.
- Unterstützung von mehreren Indizes für eine Datenbank. Diese Indizes können automatisch sortiert oder gefiltert werden.
- Mehrere Projekte können gleichzeitig im RAM sein. Die Anzahl dieser Projekte ist nur durch das freie RAM begrenzt.
- Die Masken verhalten sich genauso wie andere GUIs.
- Masken und Listen passen sich nicht-proportionalen Schriften an.
- Wahre Feldtypen-Vielfalt: Neben String-, Integer- und Float-Feldern gibt es noch Cycle-, Boolean-, Slider-, Datums-, Zeit-, Extern-, Datatypes- und Var String-Felder.
- Mit Datatypes-Feldern können Grafiken, etc. direkt in der Maske angezeigt werden.
- Flexible Such-Funktion, die Joker, unscharfe Suche und Formeln unterstützt.

- ARexx-Port, mit dem das Programm gesteuert oder erweitert werden kann. Der ARexx-Port wird auch benutzt, um einzelnen Feldern Programme zuzuordnen.
- Frei konfigurierbares "Benutzer-Menü", von dem CLI-Programme und ARexx-Scripts gestartet werden können.
- Sehr flexibles Listen-Fenster, in dem Spalten versteckt und in der Position und Größe verändert werden können.
- Einfaches Relation-Handling
- Import und Export von Datenbanken
- Flexible Druckfunktion

1.3 Neuigkeiten

Neue Features in Fiasco 2.2:

- Verbesserte Ersetzen-Funktion: Unterstützt nun mehrere Felder, deren Inhalt ersetzt werden soll, und Formeln zum Berechnen des neuen Inhalts.
- "CD-ROM-Modus" für Datenbanken implementiert.
- Der Feld-Requester checkt nun, ob eine Feld-ID für die Benutzung in Formeln geeignet ist und gibt eine Warnung aus, wenn dies nicht der Fall ist.
- Fiasco kann nun konfiguriert werden, einen Datei-Requester beim Start zu öffnen, wenn Fiasco ohne Argumente gestartet wurde.
- Neues Programm-Argument: Iconified startet Fiasco, ohne die GUI zu öffnen.
- Der Benutzer kann nun Start- und End-ARexx-Scripte für einzelne Datenbanken und das ganze Programm definieren.
- Neue ARexx-Kommandos: ActivateDBWindow aktiviert ein Fenster einer Datenbank, MenuControl kann die Menüs ausschalten. ReadSettings und SaveSettings lesen bzw. schreiben Fiascos Einstellungen.
- Neues Argument für SetField und GetField: ExtFormat benutzt erweiterte, z.B. lokalisierte, Formate.

- Neues Argument für SetField Stem: CreateListEntries macht es nun möglich, alte Listview-Einträge vollständig zu ersetzen.
- Neuer Formel-Operator: numentries(*listview*) gibt die Anzahl von Einträgen in einem Listview-Feld zurück.
- Neue Formel-Funktionen: numrecords() gibt die Anzahl von Records in der Datenbank und activerecord() die Nummer des aktiven Records in der Datenbank zurück.
- Geht nun robuster mit zerstörten Daten in Datenbank-Dateien um.

Behobene Fehler in Fiasco 2.2:

- Den berühmt-berüchtigten "Fehler 500"-Bug behoben.
- Beim Benutzen von Relationen wärend beide Datenbanken der Relation offen waren, konnte ein "Adding existing record" Fehler auftreten.
- Relations-Requester stürtzte ab, wenn entfernte Datenbank benutzerdefinierte Konstanten oder Funktionen enthielt.
- Wenn ein ARexx-Script, das von Fiasco gestarted wurde, mit Ctrl-C abgebrochen wurde, funktionierten Datatypes-Felder nicht mehr.
- GetAttr Fields gab auch schon gelöschte Felder zurück.
- Projekt/Speichern Als konnte manchmal Probleme mit Indizes haben.
- Button-Feld-Shortcuts funktionierten nicht, wenn kein Record aktiv war.
- Argumente-Attribut für Button-Felder funktionierte nicht mit CLI-Programmen.
- ARexx-Scripts ohne Argumente wurden von Fiasco mit einem leeren Paar Anführungszeichen gestartet.
- Die Import-Funktion konnte unter bestimmten Umständen hängen.
- Wegen einer uninitialisierten Variable konnte die Sortieren-Funktion unter bestimmten Umständen streiken.
- Die Sortieren-Funktion markierte eine Datenbank nicht als verändert.
- Das automatische Einsortieren sortierte einen Record, der eigentlich an das Ende eines Index gehört hätte, fälschlicherweise eine Stelle davor ein.

- SetField Stem funktionierte nicht korrekt mit automatischem sortierenden Indizes.
- Der Formel-Requester zeige keine Konstanten an.
- ActiveIndex Var und Find Var funktionierten nicht.

Neue Features in Fiasco 2.11:

- Druck-Elemente können nun die Daten zentrieren oder rechtsbündig ausrichten.
- Neue ARexx-Kommandos SetConstant, GetConstant, DeleteConstant, RecompileAllFormulas und FlushRecords.
- Neuer Formel-Operator current(), der die Nummer des Listview-Eintrags zurückgibt, dessen Wert gerade von dieser Formel berechnet wird.
- Export-Funktion filtert nun die ganzen Export-Strings.
- Ein Feld mit Popup-Gadget benutzt nun Shift + Shortcut, um den Popup-Requester zu öffnen.

Behobene Fehler in Fiasco 2.11:

- In Datenbanken mit virtuellen Feldern konnten nach der Benutzung der Such-Funktion (oder verwandten Funktionen) Internal Errors erzeugt werden.
- Nach dem Löschen vieler Records funktionierte der Record-Zugriff manchmal nicht korrekt.
- Nach Record/Alle Records Löschen und Save konnte Fiasco verwirrt werden.
- Ein von Vergl./Sortieren erzeugter Index konnte nicht mehr durch Vergl./Sortieren überschrieben werden.
- Vergl./Filter konnte Probleme verursachen, wenn die Datenbank automatisch gespeichert wurde.
- Ein fehlgeschlagenes Speichern konnte einen Absturz von Fiasco verursachen.
- Das Löschen von Indizes konnte die Erzeugung eines zweiten Standard.fidx zur Folge haben.

- Der Feld-Konvertieren-Requester erlaubte das Konvertieren in Slider-Listviews (!) anstatt in Datums-Listviews.
- Durch das Benutzen von Var String-Feldern konnten Fiasco die Signale ausgehen.
- Beim Hinzufügen von Records zu einer Datenbank ohne Records aber mit aktivierten Relationen konnten Enforcer-Hits erzeugt werden.
- Nach dem Überschreiben von Indizes mit Vergl./Sortieren oder Vergl./Filter konnten Internal Errors erzeugt werden.
- Die ARexx-Kommandos SetField und GetField benutzten fälschlicherweise Locale-Einstellungen.
- Die alten Such-Kommandos von Fiascos ARexx-Port öffneten in Fiasco 2.1 nur den Such-Requester.
- Das ARexx-Kommando ActiveRecord funktionierte mit relativen Bewegungen nicht (Next, Prev).
- Das ARexx-Kommando Export hatte eine falsche Argument-Schablone.
- Das Beenden von Fiasco vom ARexx-Port aus endete oft in einem Absturz.
- Checkbox-Gadgets in den Import/Export-Requestern wurden nicht korrekt aktualisiert.
- Oberfläche/Erstes Feld automatisch aktivieren würde ein Feld auch aktivieren, wenn es gesperrt war.

Neue Features in Fiasco 2.1:

- Unterstützung von Formeln für berechnete Feld-Inhalte und für komplexe Such-Abfragen.
- Neue Such-Funktion, die das Suchen mit mehreren Feldern und mit Formeln unterstützt.
- Verbesserte Suchmuster.
- Komplett überarbeiteter ARexx-Port, der nun mit dem Style-Guide übereinstimmt. Alte Scripts werden dennoch weiter funktionieren.

- Feldern können nun vordefinierte Werte zugewiesen werden, die mit einem Auswähl-Schalter an der rechten Seite des Feldes abgerufen werden können. Dieser Schalter kann auch benutzt werden, um ARexx-Scripte zu starten.
- Var String Felder können nun auch im Listen-Fenster angezeigt werden. Var String Felder aus Fiasco 2.0x Dateien werden als verborgen markiert und können mit dem Feld-Requester oder mit Liste/Spalte anzeigen angezeigt werden.
- Feld Entfernen überprüft nun, ob das zu löschende Feld noch benutzt wird.
- Die Index-History speichert die Reihenfolge, in der die Indizes aktiv waren. Mit Datenbank/Voriger aktiver Index kann man in der History rückwärts gehen.
- Besonderer Relations-Modus Nur Lesen, der nur die Relations-Daten liest, wenn der Schlüssel verändert wird, die Daten in der lokalen Datei speichert und die Daten nicht aktuell hält.
- Float, Datums und Zeit-Felder benutzen nun die Formatierungs-Einstellungen der locale.library wenn verfügbar.
- ARexx-Scripte von Buttons können nun Argumente haben und einen Consolen-Fenster öffnen.
- Neues Attribut für Listview-Felder: Nur Auswählen erlaubt das auswählen aber nicht das ändern von Einträgen.
- Float-Werte werden nun in einem 64 Bit Format gespeichert und sind so wesentlich präziser.
- Der Standard-Pfad von Datenbanken und der Ort des Fiasco-Hauptprogramms können nun angegeben werden.
- Neues Dateiformat für Einstellungen. Das alte Format kann weiterhin gelesen werden.

Behobene Fehler in Fiasco 2.1:

- Importieren in einen Index, der sortiert, würde den Index nicht neu sortieren.
- Wenn keine Records geladen waren (z.B. wenn eine Datenbank im Masken-Modus geladen wurde) und die Datenbank dann gespeichert wurde, wurde der aktive Index leer gespeichert.

- Das Löschen von Records in einer Datenbank die nie zuvor gespeichert wurde, bewirkte, daß falsche .frec-Dateien geschrieben wurden.
- Die Import-Funktion konnte ein Kontroll-Zeichen erkennen, obwohl keins angegeben war.
- Das Öffnen von zwei Datenbanken mit .fdat-Dateien und Relationen konnte Alerts bewirken.
- Nach Project/Sichtbar machen wurde das Service-Window nicht korrekt aktualisiert.

Neue Features in Fiasco 2.02:

- Die Fenster-Gadgets von Fiascos Masken-Fenster können nun im Optionen-Requester kontrolliert werden.

Behobene Fehler in Fiasco 2.02:

- Das Löschen eines Records, der seit dem letzten Speichern hinzugefügt wurde, konnte Fiasco verwirren.
- Das automatische Sortieren konnte die Record-Daten eines neu sortierten Records ändern.
- Speichern einer Datenbank nach Aufforderung im Index-Requester konnte Fiasco dazu bringen, falsche Index-Dateien zu benutzen.
- Einstellungen Speichern konnte abstürzen, wenn Service Fenster/Feste Position aktiv war.
- Var String Felder bemerkten Änderungen nicht, wenn nur die Länge des Inhalts verändert wurde.
- Mehrere kleine Bug-Fixes.

Neue Features in Fiasco 2.01:

- Die Editor-Einstellung unterstützt %s für das Angeben einer Position, an der der Dateiname eingesetzt werden soll. Dies ist insbesondere für GoldED-Benutzer nützlich, die das STICKY-Argument einsetzen sollten, um Probleme zu vermeiden. Beispiel: c:ged "%s" sticky.
- F_SetFieldCont wurde wesentlich beschleunigt.

Behobene Fehler in Fiasco 2.01:

- Automatisches Sortieren konnte falsch sortieren und Enforcer-Hits erzeugen.
- Automatisches Sortieren funktionierte nicht korrekt mit Absteigend.
- Wenn ein Record zu einer Datenbank hinzugefügt wurde, die keine Records enthielt und vorher ein Feld gelöscht wurde, konnten Enforcer-Hits erzeugt werden.
- Das Schließen einer Datenbank, die entfernte Datenbank einer Relation in einer Datenbank, die auch offen ist, war, konnte Enforcer-Hits oder Abstürze erzeugen.
- Das Hinzufügen von Relationen mit dem Relations-Requester funktionierte nicht auf Amigas ohne RAM nach $0100 0000.
- Wenn ein in der Liste nicht anzeigbares Feld in ein anzeigbares Feld konvertiert wurde, oder umgekehrt, wurde das Layout des Listen-Fensters nicht korrekt aktualisiert.
- Druck-Funktion schnitt den Inhalt von Var String-Feldern nach einer Leerzeile ab und konnte einzelne Zeichen verschlucken.
- Manchmal löschte Reorganisieren nicht alle unbenutzten Records.
- Das Löschen von Records, die nach dem letzten Speichern einer Datenbank hinzugefügt wurden, erzeugte Müll-Records in der Datenbank-Datei.
- Das Record-Argument von F_AddListEntry funktionierte nicht korrekt.
- F_MarkMatch funktionierte nicht verläßlich.
- Mehrere kleine Bug-Fixes.

Kapitel 2

Der Beginn

2.1 Voraussetzungen

Die minimalen Anforderungen von Fiasco sind ein Amiga mit OS 2.04 (37.175) und 1 MB RAM. Die optimale Konfiguration sind ein Amiga mit OS 3.x (39.x oder höher), ein 68020 Prozessor, 2 MB RAM und eine Festplatte.

Fiasco benutzt die gtlayout.library von Olaf Barthel für die GUI. Var String-Felder sind mit dem textfield.gadget von Mark Thomas implementiert. Die Dateien sind im Fiasco-Archiv enthalten.

Fiasco benötigt das ACTION_SET_FILE_SIZE Packet, das mit Amiga OS 2.0 eingeführt wurde. Sowohl das 2.0 ROM Filesystem als auch der 2.0 RAM Handler unterstützen dieses Paket. Es kann jedoch sein, daß manche Handler dies nicht unterstützen. In diesem Fall können Sie keine Fiasco-Projekte auf diesen Handlern speichern. Lesen sollte immer möglich sein.

Möglicherweise auch deshalb funktioniert Fiasco nicht mit Diskexpander und XFH. Sie werden Datenbanken lesen können, aber das Speichern funktioniert nicht. Wenn Sie Diskexpander benutzen, werden Fehler erzeugt. Beachten Sie, daß wenn Sie XFH benutzen, das System abstürzen kann.

Die Memory-Pool Funktionen von Amiga OS 3.0 und Amiga OS 3.1 geben nicht mehr benutzte Puddles nicht frei, bis der Pool gelöscht wird. Benutzen Sie SetPatch 40.16 (bereits bei WB 40.42), um dies zu beheben. Wenn Sie Amiga OS 2.0 oder Amiga OS 2.1 benutzen, brauchen Sie sich nicht darum zu kümmern.

Features und benötigte OS-Versionen:

Lokalisation:	Amiga OS 2.1 (38.x)
Screenmode-Requester:	Amiga OS 2.1 (38.x)
Online-Hilfe:	Amiga OS 3.0 (39.x) oder amigaguide.library v34
Datatypes-Felder:	Amiga OS 3.0 (39.x)

2.2 Installation

Wenn Sie den Installer haben, klicken Sie einfach zweimal auf das Installer Icon Ihrer Lieblingssprache im Verzeichnis "Install". Sie werden dann instruiert, was sie zu tun haben.

Falls Sie den Installer nicht besitzen, brauchen sie das "Fiasco" Verzeichnis nur irgendwo hin kopieren. Falls Sie wollen, können Sie die Catalogs nach locale:catalogs kopieren, was aber nicht notwendig ist. Weiterhin können Sie die unbenutzten Sprachen in "Documentation" löschen und die verbleibenden Dateien in die übergeordneten Verzeichnisse schieben. Die Dateien in "Development" und "Install" werden für den normalen Gebrauch nicht benötigt. Mit dieser Konfiguration kann Fiasco laufen. Falls sie einen 68020 besitzen, können sie zusätzlich die gtlayout.library im Hauptverzeichnis von Fiasco durch die gtlayout.library aus dem Verzeichnis libs/68020 ersetzen. Um die Library anderen Programmen zugänglich zu machen, muß sie ins Verzeichnis Libs: kopiert werden.

2.3 Fiasco starten

Sie können Fiasco von der Workbench oder der Shell starten. Die simpelsten Möglichkeiten sind in der Workbench einfach zweimal auf das Fiasco-Piktogramm zu klicken oder in der Shell einfach in das Verzeichnis von Fiasco zu wechseln und Fiasco einzugeben.

Diese Start-Argumente werden von Fiasco unterstützt:

From: Die Datenbanken die beim Start geladen werden sollen. Sie können mehrere Datenbanken angeben. Nur von der Shell unterstützt. Beim Start von der Workbench gibt es andere Möglichkeiten: Wenn Fiasco über ein Projekt-Icon einer Fiasco-Datenbank gestartet wird, wird diese Datenbank beim Start in Fiasco geladen. Sie können auch mehrere Projekt-Piktogramme auswählen und Fiasco danach über das Programm-Piktogramm starten. Dann wird Fiasco alle ausgewählten Projekte laden.

Config: Nach diesem Schlüsselwort können Sie eine Fiasco-Konfigurationsdatei angeben, die von Fiasco benutzt werden soll. Standardwert: env:fiasco.prefs. Beispiel: CONFIG=env:fiasco_24bit.prefs. Dies wird sowohl als Shell-Argument als auch als Workbench-Merkmal unterstützt.

Iconified: Wenn sie ICONIFIED angeben, wird Fiasco seine GUI nicht öffnen, sondern im verborgenen Status gestartet. Auf der Workbench wird dann ein Fiasco-AppIcon angezeigt, mit dem Fiasco's GUI geöffnet werden kann. Weitere Möglichkeiten, auf Fiasco Zugriff zu erhalten sind, Fiasco nochmal zu starten, oder den ARexx-Port zu benutzen. Dies wird sowohl als Shell-Argument als auch als Workbench-Merkmal unterstützt. Neu in Fiasco 2.2.

Bevor Sie Fiasco starten, sollten Sie allerdings sicherstellen, daß Fiasco genug Stack zur Verfügung steht. Für jetzt sind 8192 Bytes ein guter Wert. Wenn Sie Fiasco von der Workbench starten, können Sie die Stack-Größe im Informations-Requester für Fiascos Programm-Piktogramm und die Projekt-Piktogramme einstellen. In einer Shell müssen Sie das Stack-Kommando benutzen, um den Stack zu erhöhen.

Wenn Sie versuchen, Fiasco zu starten, während bereits ein Fiasco-Prozess läuft, wird Fiasco sich selbst bemerken und nicht ein zweites mal starten. Wenn Sie beim zweiten Start Datenbanken angegeben haben, wird der bereits laufende Fiasco-Prozess diese laden. Wenn keine angegeben wurden, wird Fiasco ein neues Fenster öffnen.

2.4 Quick Start

Dies sind die wichtigsten Dinge, die man wissen muß, um mit Fiasco zu arbeiten:

- Das Programm startet man am besten über das Programm- oder ein Projekt-Icon.
- Es gibt zwei Arbeits-Modi in Fiasco: Einen Modus, in dem man die Records verändern kann, und einen anderen, in dem man die Maske verändern kann. Man steuert sie über Kontrolle/Record-Modus bzw. Kontrolle/Masken-Modus.
- Das Service-Fenster erleichtert das Arbeiten mit Fiasco, wenn man sich noch nicht mit Shortcuts u.ä. auskennt. Mann öffnet es über Kontrolle/ServiceFenster. *Achtung*: Die Funktionen der Gadgets variieren in den Modi.

- Die Liste, die man mit Kontrolle/Listen-Fenster öffnen kann, kann verändert werden, indem man in die Titel-Zeile der Liste klickt. Klickt man einen Titel einmal an, so wird er aktiviert. Über das Menü Liste kann man nun diverse Sachen mit der Spalte machen. Wenn man am rechten Rand eines Titels klickt, und die Maustaste gedrückt hält, kann man die Breite einer Spalte verändern. Falls man in den restlichen Raum des Titels klickt, und die Maustaste gedrückt hält, kann man die Position der Spalte verändern.

- Bestimmte Einstellungen zum Projekt, z.B. zum Masken-Layout kann man über den Menüpunkt Datenbank/Optionen einstellen.

- Die Sortier- und Filter-Funktionen benutzten Indizes. Wenn Sie eine Datenbank sortieren oder filtern wird ein neuer Index erstellt und aktiviert. Um den alten Index zu aktivieren, benutzen Sie Datenbank/Voriger aktiver Index. Vollständige Kontrolle über alle Indizes haben Sie mit Datenbank/Indizes.

- Record/Löschen löscht Records nur aus dem Index. Um den Record endgültig loszuwerden müssen Sie Datenbank/Reorganisieren benutzen.

- Bei sonstigen Problemen kann man einfach das Menü anwählen und bevor man die rechte Maustaste losläßt, die Help-Taste drücken.

Kapitel 3

Grundlagen einer Datenbank

Eine Datenbank kann mit einer einfachen Kartei verglichen werden. Die meisten Implementationen von Datenbanken benutzen dieses Schema als Grundlage.

In Fiasco besteht ein Datenbank-Projekt aus zwei Komponenten: Auf der einen Seite die Daten, die durch Records aufgeteilt werden. Auf der anderen Seite die Maske, die die Struktur der Daten definiert.

Im Folgenden werden die grundlegenden Allgemeinen und die Fiasco-Spezifischen Prinzipien von Datenbanken beschrieben.

3.1 Records

Records stellen in einer Datenbank die Karteikarten einer Kartei dar. Ein Record ist also eine Sammlung verschiedener Aspekte zu einem Oberbegriff; z.B. zu einer Person Name, Adresse usw. In der Maske wird immer nur ein Record auf einmal angezeigt, in der Liste werden Records als Zeilen dargestellt.

Indizes geben Ihnen mehr Kontrolle über die Records. Eine Datenbank kann mehrere Indizes haben. Jeder Index bestimmt, welche Records überhaupt angezeigt werden, und wie diese Records geordnet sind. So kann eine Datenbank Records enthalten, die nicht sichbar sind, da sie nicht im aktiven Index sind.

3.2 Indizes

Indizes sind neu in Fiasco 2.0. Indizes werden benutzt, um die Reihenfolge der Records zu bestimmen, und um zu bestimmen, ob ein Record überhaupt angezeigt wird oder nicht.

Eine Datenbank kann mehrere Indizes enthalten.

Wenn Sie sich die Records als Karteikarten auffassen, sind Indizes zusätzliche Listen, die z.B. sagen "Karteikarte 16 ist an Position 1, Karte 5 an Position 2", usw. Die sogenannten Record-Nummern, die von Fiasco benutzt werden, sind in Wirklichkeit die Positionen der Index-Einträge. Deshalb kann sich mit jedem Index auch die Nummer jedes Records ändern.

Die neuen Sortier- und Filter-Funktionen benutzen ebenfalls Indizes. Diese Funktionen erzeugen einfach einen neuen Index, der wie gewünscht sortiert oder gefiltert ist. Fiasco erlaubt es außerdem, einen Index sortiert oder gefilter zu halten.

Normalerweise werden Records nur zum aktiven Index hinzugefügt. Wenn Sie wollen, daß ein anderer Index auch die neuen Records aufnimmt, müssen Sie die Option Neue Records automatisch hinzufügen im Index Ändern-Requester (siehe Abschnitt 10.6.7) aktivieren.

Mehr Informationen über Indizes können Sie in Abschnitt 5.3 finden.

3.3 Felder

Felder legen fest, was für Daten gespeichert werden. In Fiasco werden Felder in der Maske definiert. Felder sind die grundlegenden Elemente der Maske und der Liste.

Fiasco unterstützt mehrere Feldtypen. Mehr Informationen über Feldtypen und deren Features gibt es im Kapitel Feldtypen.

3.4 Maske

Die Maske ist die Darstellungsform, mit der Fiasco am meisten arbeitet. In einer Maske kann im Gegensatz zur Liste nur ein Record gleichzeitig angezeigt werden. Dafür ist diese Anzeige übersichtlicher und geordneter (falls man es richtig gemacht hat ;-), weshalb diese Darstellungsform auch manchmal als "Formular" bezeichnet wird.

Die Maske setzt sich aus sogenannten Feldern zusammen, die verschiedene Typen und somit verschiedene Erscheinungsformen haben.

Bei normalen Amiga-Programmen würde man die "Felder" als Gadgets bezeichnen.

Fiasco benutzt intern Gadgets als Felder, die optisch denen der gadtools.library entsprechen.

Fiasco-Masken passen sich automatisch an einen beliebigen Zeichensatz an, einzige Voraussetzung ist, daß er nicht proportional ist (z.B. topaz, courier).

Um eine Maske in Fiasco zu gestalten, muß man über den Menüpunkt Kontrolle/Masken-Modus in den Masken-Modus überwechseln. Nun kann man mit der Maus existierende Felder verschieben oder über das Feld Menü verschiedene andere Manipulationen ausführen. Mehr dazu kann im Kapitel 4.1 gefunden werden.

Abschnitt 10.1 enthält mehr Informationen über die Eigenschaften des Masken-Fensters.

3.5 Liste

Im Gegensatz zur Maske, kann eine Liste mehrere Records auf einmal anzeigen. In einer Liste werden Records durch Zeilen repräsentiert, während die Felder durch Spalten repräsentiert werden. Deshalb hat ein Record in der Liste wesentlich weniger Platz als in der Maske und das Layout der Liste ist im Gegensatz zur Maske stark eingeschränkt. Auf der anderen Seite ist die Liste sehr geeignet, um einen schnellen Überblick über alle Records der Datenbank bzw. des aktiven Records zu bekommen.

Abschnitt 10.2 enthält genaue Beschreibungen der Eigenschaften von Fiascos Listen-Fenster.

3.6 Editier-Modi in Fiasco

Fiasco unterteilt seine Arbeit in Modi. Wenn Sie Änderungen in der Maske vornehmen wollen, muß der Masken-Modus aktiv sein. Falls Sie Änderungen in den Record-Daten vornehmen wollen, der Record-Modus.

3.6.1 Record-Modus

In diesem Modus kann man Records hinzufügen, löschen und deren Inhalt verändern. Dieser Modus kann über den Menüpunkt Kontrolle/Record-Modus aktiviert werden. Wenn der Record-Modus aktiv ist, werden "Tapedeck"-Gadgets im Service-Fenster angezeigt und das Status-Gadget zeigt normalerweise die Nummer des aktiven Records und die Anzahl der Records an (z.B.: 78 / 92).

3.6.2 Masken Modus

In diesem Modus hat man die Möglichkeit, die Maske zu verändern, d.h. Felder neu zu erstellen, zu löschen und zu verschieben. Außerdem können hier Relationen erstellt und verändert werden. Dieser Modus kann über den Menüpunkt Kontrolle/Masken-Modus aktiviert werden. Wenn der Masken-Modus aktiv ist, wird ein Cycle-Gadget mit Feldtypen im Service-Fenster angezeigt und das Status-Gadget zeigt normalerweise die Koordinaten des Cursors in der Maske an (z.B. X: 10, Y: 5).

Kapitel 4

Einfache Benutzung von Fiasco

Und nun zur Praxis: Wenn Sie eine Datenbank in Fiasco erstellen möchten, müssen Sie zuerst die Maske und dann die Records erstellen. Fiasco erlaubt es Ihnen, in den meisten Fällen die Datenbank auf intuitive Weise zu erstellen.

4.1 Arbeiten in der Maske

Um eine Maske zu erstellen, muß man zuerst in den Masken-Modus indem man den Menüpunkt Kontrolle/Masken-Modus aktiviert. Nun erscheint in der Maske ein Cursor, mit dem man die Position des Feldes festlegen kann, das man erstellen will. Den Typ des Feldes, das erzeugt werden soll, kann man über das Menü Feld/Typ festlegen. Die verschiedenen Feldtypen werden im Kapitel Feldtypen erklärt. Weiterhin läßt der aktuelle Feldtyp sich noch über das Service-Fenster und per Tastaturshortcuts einstellen (siehe Abschnitt Menüs).

Nach diesen Einstellungen kann man mit dem Menüpunkt Feld/Feld Hinzufügen ein neues Feld erstellen. Zuerst öffnet sich der Feld-Requester. Die einzelnen Gadgets im Feld-Requester sind vom Feldtyp abhängig und bei den jeweiligen Typdokumentationen (siehe Abschnitt 9) erklärt. Es reicht übrigens nicht, nur Ok anzuklicken. Bestimmte Attribute, wie z.B. die ID müssen vom Benutzer immer neu angegeben werden. Sollten alle Attribute gültig sein, schließt sich der Requester und das jeweilige Feld erscheint in der Maske.

Nachträglich können alle Attribute des Feldes bis auf den Feldtypen an

sich (dazu später mehr) geändert werden. Die Position läßt sich einfach mit der Maus verändern. Wenn man ein Feld auswählt und die Maustaste gedrückt hält erscheint einmal ein gefülltes Rechteck, das mit der Maus mitgeht und ein anderes, das die genaue neue Position zeigt. Läßt man die linke Maustaste los, wird das Feld an der neuen Stelle abgelegt, soweit keine anderen Felder im Weg sind. Um das Ziehen des Feldes abzubrechen, müssen Sie die rechte Maustaste drücken, während Sie noch die linke Maustaste gedrückt halten. Sie können auch mehrere Felder ziehen, indem sie Shift drücken wenn Sie die Felder auswählen. Die Felder werden deselektiert werden, wenn Sie ein Feld auswählen, ohne Shift gedrückt zu halten.

Der Feld-Requester läßt sich auch wieder aufrufen, indem man entweder auf ein Feld doppelt klickt, oder ein Feld aktiviert und den Menüpunkt Feld/Aktives Feld ändern anwählt. Der Menüpunkt Feld/Benanntes Feld ändern zeigt eine Liste aller Felder an. Wenn Sie ein Feld ausgewählt haben, öffnet diese Funktion den Feld-Requester für das ausgewählte Funktion. Dies ist nützlich für das Ändern von verborgenen Feldern. Hier lassen sich wiederum alle Parameter frei verändern. Der ID-Wert sollte jedoch nur mit Vorsicht verändert werden, da andere Fiasco-Projekte oder ARexx-Scripts unter Umständen noch die alten Werte eingestellt haben und die Felder nicht mehr finden. Falls man die maximale Zeichenanzahl von String, Extern oder Datatypes-Feldern verkleinert, wird zuerst geprüft, ob Daten eventuell verloren gehen würden und eine Warnung ausgegeben.

Über den Menüpunkt Feld/Feld entfernen lassen sich Felder wieder entfernen. *Vorsicht:* Falls die Option Sicherheits-Requester in den Benutzer-Oberflächen-Einstellungen nicht aktiviert ist, werden alle in diesem Feld gespeicherten Daten sofort aus dem RAM gelöscht. Beim nächsten Abspeichern werden auch die auf der Diskette verbliebenen Daten dieses Feldes vernichtet. Wenn das Feld von irgendeinem anderen Fiasco-System benutzt wird, wie z. B. Indizes oder Formeln, werden Sie vor dem Löschen des Feldes gewarnt. Dabei ist die Sicherheits-Requester-Einstellung nicht ausschlaggebend.

Im Options-Requester (siehe Abschnitt 10.6.4) lassen sich weitere Parameter für das aktuelle Datenbank-Projekt angeben.

Sie können Felder auch gruppieren. Wenn ein gruppiertes Feld aktiviert wird, werden auch alle anderen Felder in der Gruppe aktiviert. Weiterhin können bestimmte Feld-Typen grafisch miteinander verschmelzen wenn sie gruppiert sind. Beispiele dafür sind Leisten- und Listview-Felder. Abschnitt 5.2 enthält mehr Informationen über Gruppen.

Der Menüpunkt Feld/Relation ändern arbeitet ähnlich wie Feld ändern. Hier kann man jedoch Parameter für Relationen einstellen. Kapitel 5.5 enthält weitere Informationen über Relationen.

Ist die Maske vollständig kann man wieder in den Record-Modus zurück-

gehen (Kontrolle/Record-Modus) und die eigentlichen Daten erstellen.

4.2 Arbeiten mit Records

Wenn Sie eine Maske mit ein paar Feldern haben, ist die Zeit gekommen, Records zu erstellen, um Daten zu speichern. Die einfachste Methode, einen Record zu erstellen, ist Record/Hinzufügen oder das Äquivalent Neu im Service-Fenster. Diese Prozedur erzeugt — wie schon der Name sagt — einen Record und aktiviert diesen. Die Felder im Record werden die Werte enthalten, die im Masken-Modus definiert wurden. Mit der Maus können Sie nun ein Feld aktivieren und den Inhalt verändern.

Ein anderer Weg, Records zu erstellen ist Record/Duplizieren. Diese Funktion erzeugt einen Record, der eine exakte Kopie des vorher aktiven Records ist. Alle Startwert-Attribute werden ignoriert!

Sie können Records auch auschneiden oder kopieren und diese an einer anderen Position wieder einfügen. Beachten Sie, daß Ausschneiden und Einfügen wie das Löschen und Neuerstellen eines Records funktioniert und den Record nicht etwa einfach verschiebt. Wenn also andere Indizes (siehe Abschnitt 3.2) den Record benutzen, der ausgeschnitten wurde, wird der Inhalt des Records konstant bleiben, selbst wenn Sie den Record wieder einfügen und den Inhalt verändern.

Wenn Sie einen Record nicht mehr brauchen, können Sie ihn mit Record/Entfernen oder Löschen im Service-Fenster aus dem Projekt entfernen. Wenn währenddessen Einstellungen/Sicherheits-Abfragen aktiv ist, werden Sie noch einmal gewarnt, bevor der Record tatsächlich gelöscht wird.

Beachten Sie, daß nur der Eintrag im aktiven Index gelöscht wird. Die Daten des Records werden in der Datenbank verbleiben. Um den Record endgültig zu löschen, müssen sie Datenbank/Reorganisieren benutzen.

Wenn Sie durch die erzeugten Records “hindurchblättern” wollen, können Sie die Menüs, das Service-Fenster, die Cursor-Tasten oder das Listen-Fenster benutzen[1]. Weil ich denke, daß die GUI-Teile intuitiv benutzt werden können, erkläre ich nur die Cursor-Tasten. Fangen wir mit der “Hoch”-Taste an. Bei Betätigung wird der dem aktuellen Record vorausgehende Record aktiviert. Die “Runter”-Taste aktiviert den nächsten Record. Diese Einteilung entspricht der Struktur des Listen-Fensters[2]. Diese Tasten kombiniert mit der Ctrl-Taste aktivieren die jeweiligen Extrema, also den ersten, bzw. den letzten Record.

[1] Bitte beschweren Sie sich nicht, daß dies zu viel ist ;-)
[2] Wie der aufmerksame Leser sicherlich sofort erkannt hat

Kapitel 5

Fortgeschrittene Benutzung von Fiasco

Das Wissen, das in diesem Kapitel beschrieben wird, ist zum Arbeiten mit Fiasco nicht notwendig. Es kann jedoch in vielen Fällen sehr hilfreich sein.

5.1 Ändern von Feldtypen

Wenn Sie ein Projekt erstellt haben, kann es nützlich werden, wenn man den Typ eines Felds ändern kann. Diese Situation kann dadurch entstehen, daß manche Inhalte eines Feldes sich in eine andere Richtung als ursprünglich erwartet entwickelt haben. Die Konvertierungs-Funktion kann auch beim Importieren von Dateien nützlich werden. Die Import-Funktion erzeugt nämlich ausschließlich String-Felder, während manche Daten jedoch auch in anderen Feldtypen besser aufgehoben wären.

Um den Konvertieren-Requester öffnen zu können, muß der Masken-Modus aktiv sein. Wenn dies so ist, aktivieren Sie das Feld, das Sie umwandeln möchten und wählen Feld/Feld Konvertieren auf. Der Kovertieren-Requester zeigt die ID des Feldes, den jetzigen Typ und die Typen, in die das Feld umgewandelt werden kann, an. Wenn Sie Alternatives Format aktivieren, wandelt die Funktion die Daten in einer anderen, normalerweise abstrakteren Form um. Dies wird nicht von allen Feldtypen unterstützt. Wenn Sie den neuen Typ ausgewählt haben und mit Ok weitergehen, wird der Umwandlungs-Vorgang gestartet.

Bitte beachten Sie, daß Fiasco nicht vor irgendwelchem Datenverlust, der durch die Umwandlung entstehen kann (z.B. String zu Integer) warnt. Wenn der neue Feldtyp zusätzliche Attribute benötigt (Der Extern-Typ

benötigt beispielsweise ein Programm), wird Fiasco den Feld-Requester danach öffnen. Andere Attribute werden Standardwerte benutzen. Bitte beachten Sie auch, daß wenn Sie ein Feld umwandeln und es danach wieder in den ursprünglichen Typ wandeln, die alten Attribute nicht benutzt werden.

Wenn ein Feld durch Relationen, Formeln oder Indizes (zum Sortieren oder Filtern) benutzt wird, werden diese Nutzer entfernt. Fiasco wird vorher noch eine Warnung ausgeben.

Informationen über die Ergebnisse vom Feldumwandeln von einem zum anderen Typ können aus der Feldtypen-Dokumentation bezogen werden. Text-, Button- und Leisten-Felder können nicht konvertiert werden. In anderen Fällen macht das Konvertieren von einem bestimmten Typ zu einem anderen nicht viel Sinn (z.B. Boolean zu Datatypes).

Wenn Sie ein Listview-Feld in ein Nicht-Listview-Feld oder umgekehrt konvertieren, wird Fiasco das Zeichen '| ' benutzen, um die Einträge eines Listviews zu trennen.

5.2 Gruppen

Eine Gruppe ist eine Gruppe von Feldern, die vom Benutzer ausgewählt wurde. Dies allein scheint allerdings nicht allzu nützlich. Die Vorteile von Gruppen liegen in den zusätzlichen Eigenschaften, die gruppierte Felder haben. Zum einen "kleben" gruppierte Felder im Masken-Modus zusammen. Das bedeutet, daß wenn Sie ein gruppiertes Feld aktivieren, die ganze Gruppe aktiviert wird. Wenn Sie ein gruppiertes Feld ziehen, wird die ganze Gruppe gezogen. Alle anderen Funktionen, die Wirkung auf alle ausgewählten Felder haben, werden auch Wirkung auf die gesamte Gruppe haben.

Der zweite Vorteil von Gruppen ist die verschmolzene Anzeige (und Funktion) gruppierter Felder. Manche Feldtypen teilen ihre Grafiken mit benachbarten Feldern in der Gruppe. Dies wird von diesen Feldtypen unterstützt:

- Leisten-Felder (siehe Abschnitt 9.16)
- Listview-Felder (siehe Abschnitt 9.17)

Wie diese Felder ihre Grafiken teilen wird in den Feld-Typ-Dokumentationen beschrieben.

Um eine Gruppe zu erstellen, selektieren Sie alle Felder, die in der Gruppe sein sollen, und wählen Feld/Gruppe Bilden. Um die Gruppe wieder aufzulösen, aktivieren Sie die Gruppe und wählen Feld/Gruppe Auflösen.

Wenn Sie versuchen, eine bereits bestehende Gruppe mit anderen Feldern oder anderen Gruppen zu gruppieren, wird eine neue große Gruppe

aller ausgewählten Felder erzeugt. Das Auflösen dieser Gruppe wird alle Gruppen, die in dieser Gruppe gruppiert waren mit auflösen. Alle Felder in dieser Gruppe werden also frei werden.

5.3 Benutzen von Indizes

Das Konzept und die einfache Benutzung von Indizes wird in den Abschnitten 3.2 und 6.7 beschrieben.

Das Index-System ist kann jedoch wesentlich mehr. Es erlaubt Ihnen, Indizes automatisch zu sortieren, zu filtern und zu aktualisieren. Die GUI des Index-Systems ist der Index-Requester, der mit Projekt/Indizes geöffnet werden kann. Hier können Sie einen Index aktivieren, indem Sie ihn auswählen und Ok anwählen. Sie können auch neue Indizes erstellen, Indizes ändern oder löschen.

Es kann sein, daß noch kein Index existiert. Das ist der Fall, wenn die Datenbank noch nicht gespeichert wrude oder im Format von Fiasco 1.x ist. Ein Standard-Index wird erstellt, wenn die Datenbank das erste Mal gespeichert wird.

Erstellen eines neuen Index

Wenn Sie einen neuen Index mit dem Neu Schalter erstellen, öffnet sich ein anderer Requester, der Neuer Index Requester (siehe Abschnitt 10.6.7).

Das oberste Gadget nimmt den Namen für den Index auf.

Die Checkbox Neue Records automatisch hinzufügen kontrolliert, ob Fiasco automatisch neue Records zu diesem Index hinzufügen soll. Wenn die Checkbox nicht aktiv ist, werden Records, die der Datenbank hinzugefügt wurden, währen ein *anderer* Index aktiv war, diesem Index *nicht* hinzugefügt. Wenn es aktiv ist, wird Fiasco automatisch diese Records diesem Index hinzufügen. Wenn die Sortier-Option (siehe unten) aktiv ist, wird dieser Record automatisch in den Index einsortiert. Wenn die Filter-Option (auch siehe unten) aktiv ist, wird der Record zuerst mit dem Filter getestet und nur dann zu dem Index hinzugefügt wenn er auf den Filter passt.

Wenn Sie die Schalter benutzen, können Sie einen Filter für den Index angeben oder die Sortierung angeben. Die Sortier- und Filter-Funktionen werden nur aktiviert wrrden, wenn Sie die Checkboxen neben dieser Schalter aktiviert haben. Die Sortierung wird immer automatisch vorgenommen, wenn Sie einen Record ändern oder hinzufügen. Der Filter wird nur während der Erzeugung des Index benutzt und wenn ein Record in einem anderen Index hinzugefügt wurde und Neue Records automatisch hinzufügen aktiv ist. Der Filter wird *nicht* benutzt, wenn ein Records hinzugefügt wurde

während ein Record hinzugefügt wurde während der Index aktiv war oder ein Record geändert wurde.

Das Listview-Gadget erlaubt es Ihnen, einen bereits bestehenden Index als Vorbild für den neuen Index auszuwählen. Nur die Records, die in dem ausgewählten Index vorhanden sind, werden für den neuen Index überprüft. Wenn Sie das Sortieren nicht benutzen, wird die Reihenfolge übernommen. Wenn Sie weder Sortieren, noch den Filter benutzen, wird eine Kopie des ausgewählten Index erzeugt. Wenn Sie einen Index auswählen, der bereits einen Filter benutzt, können Sie einen Filter erzeugen, der mehrere Bedingungen benutzt, die mit Und verküpft sind. Der besondere Eintrag ≪Kein Index≫ entspricht allen Records in der Datenbank, sogar denen, die in keinem Index benutzt werden.

Der neue Index wird erzeugt, nachdem Sie sowohl den Neuer Index-Requester als auch den Index-Requester mit Ok bestätigt haben.

Ändern eines Index

Wenn Sie einen Index Ändern, öffnet sich der Index Ändern-Requester, der dem Neuer Index-Requester ohne dem Index-Namen und dem Listview entspricht. Sie können den Filter, die Sortierung und die Neue Records automatisch hinzufügen Option ändern. Die Änderungen werden ausgeführt wenn Sie beide Requester mit Ok verlassen.

Löschen eines Index

Das Löschen gadget löscht den ausgewählten Index. Sie können den letzten Index jedoch nicht löschen, da Fiasco immer mindestens einen Record benötigt[1].

5.3.1 Die Index-History

Die Index-History ist ein System, daß sich die Reihenfolge merkt, in der die Indizes vom Benutzer aktiviert werden. Beispielsweise wird ein neuer Index aktiviert, wenn ein Filter erzeugt wird. Fiasco merkt sich jedoch, welcher Index vor diesem aktiv war. So können Sie den Filter einfach wieder ausschalten, indem Sie einen Schritt in der Index-History rückwärts gehen. Dies kann mit dem Menüpunkt Datenbank/Voriger aktiver Index geschehen. Um den Filter wieder zu aktivieren, können Sie Datenbank/Nächster aktiver Index benutzen, um den Schritt wieder vorwärts zu machen.

[1] Abgesehen von der Sitation, daß Sie eine Fiasco 1.x Datenbank geladen haben oder eine Datenbank neu erstellt haben.

5.4 Benutzen von Markierungen

Bei der fortgeschrittenen Benutzung von Datenbanken können Markierungen nützlich sein. Eine Markierung ist einfach ein kleiner Hinweis, der für einen Record gesetzt oder nicht gesetzt sein kann. Markierungen könnten mit Boolean oder ähnlichen Feldern simuliert werden, die Markierungen von Fiasco haben jedoch zusätzliche Vorteile. Zum Beispiel kann ein markierter Record einfach in der Liste von Fiasco entdeckt werden, da der Record farblich hervorgehoben wird. In der Maske können markierte Records durch ein "M" im Status-Gadget des Services-Fensters entdeckt werden.

Markierungen können mit Record/Markiere Record gesetzt und mit Record/Lösche Markierung wieder entfernt werden. Wenn Sie alle Markierungen in einem Projekt löschen wollen, steht Ihnen Record/Lösche alle Markierungen zur Verfügung. Um alle Markierungen zu setzen, benutzen Sie Record/Markiere alle Records. Wenn Sie alle Markierungen löschen und alle nicht markierten Records markieren wollen, können Sie dies mit Record/Markierungen umschalten machen.

Vergl./Markieren öffnet einen Such-Requester, der zum Markieren aller Records benutzt werden kann, die auf ein gegebenes Muster passen.

Markierungen werden in der Fiasco-Projekt-Datei mit den Record-Daten gespeichert und sind so unabhängig von Indizes.

5.5 Relationen

Relationen sind Felder, deren Inhalt nicht in der Datei des Projektes, aus dem die Relationen stammen, sondern in einer weiteren Projekt-Datei zu finden sind. Um zu wissen, aus welchem Record die Daten entnommen werden sollen, müssen beide Projekte ein Feld mit einem eindeutigen Identifikations-Inhalt haben, das als Schlüssel ("Key") bezeichnet wird.

Diese Methode beugt der Situation vor, daß in vielen verschiedenen Projekten die selben Daten gespeichert sind. So wird geholfen, Disketten- oder Platten-Speicherplatz zu sparen. Weiterhin brauchen Sie nur die Daten in einem Feld zu ändern, damit alle anderen "verwandten" Felder auch ihren Inhalt ändern.

Mit Fiasco müssen Sie Relationen in der Datenbank definieren, die in die Daten aus einer anderen Datenbank liest. Die Datenbank, in der die Relationen definiert wurden, wird im Weiteren als "lokal" bezeichnet, während die Datenbank, aus der die Daten gelesen werden als "entfernt"[2].

[2] Fiasco 1.x benutzte die Worte "hier" und "dort" zu diesem Zweck. Die neuen Namen wurden von Claudio Mazzuco vorgeschlagen. Vielen Dank :-) Beachten Sie, daß in diesem Zusammenhang dieses nichts mit Netzwerken zu tun hat.

5.5.1 Relations-Typen

Fiasco unterstützt diese Typen von Relationen:

1:N

Dies ist der einfachste Relations-Typ. Eine 1:N-Relation[3] liest die Daten jedes Feldtyps in der entfernten Datenbank über einen Schlüssel, der einen einfachen Typ haben muß; das Feld darf also kein Listview-Feld sein. In der entfernten Datenbank muß der Schlüssel eindeutig sein. Die gelesenen Daten wurden direkt in das passende Feld in der lokalen Datenbank kopiert.

N:Sum

Dies ist keine echte Relation, da die Daten nicht zurückgeschrieben werden können. Eine N:Sum-Relation[4] liest die Daten einfacher Feldtypen und versucht, die Summe aller Felder mit passenden Schlüsseln zu errechnen. Deshalb muß der Schlüssel auch nicht eindeutig sein.

N:L

Eine N:L-Relation[5] liest die Daten einfacher Feldtypen in der entfernten Datenbank. Für jeden Record in der entfernten Datenbank, in dem ein passender Schlüssel gefunden wird, fügt Fiasco einen Eintrag dem lokalen Listview-Feld hinzu und kopiert die Daten hinein. Deshalb muß der Schlüssel auch hier nicht eindeutig sein.

1:L

Eine 1:L-Relation ist der einzige Relations-Typ, der den lokalen Schlüssel in einem Listview-Feld benötigt. Der entfernte Schlüssel muß jedoch in einem einfachen Feld sein. Dieser Typ ist dem 1:N-Typ sehr ähnlich, Sie können jedoch mehrere lokale Schlüssel mit den Möglichkeiten des Listview-Feldtyps (siehe Abschnitt 9.17) angeben. Fiasco wird die eigentlichen Daten der entfernten Datenbank nach jedem Schlüssel durchsuchen und Einträge im lokalen Listview für die eigentlichen Daten mit dem gefunden Inhalt erstellen. Da es kein Listview im Listview gibt, müssen die entfernten eigentlichen Daten in einem einfachen Typ sein.

[3] In Fiasco 1.x wurde dieser Typ 1:1 genannt.
[4] In Fiasco 1.x wurde dieser Typ Sum:N genannt.
[5] Das “L” steht für Listview.

Nur Lesen

Nur Lesen ist ein besonderer Modus der Relations-Typen *1:N* und *1:L*. Wenn Nur Lesen aktiv ist, wird das Relations-System nur benutzt, um die Eingabe-Möglichkeiten zu verbesseren. So können Sie Daten von einer entfernten Datenbank mit einem eindeutigen Schlüssel kopieren. Die Daten werden jedoch in der Datei der lokalen Datenbank gespeichert und werden nicht länger zu der entfernten Datenbank aktuell gehalten.

5.5.2 Erstellen von Relationen

Dieser Abschnitt beschreibt das Erstellen von Relationen in Fiasco-Datenbanken. Er wurde ursprünglich geschrieben, um das Erstellen von Relationen des Typs 1:N zu beschreiben. Sie können jedoch auch andere Relations-Typen nach diesen Anweisungen erstellen, wenn Sie die Bedingungen für die Relations-Typen im Kopf behalten, die im vorherigen Abschitt beschrieben wurden.

Um in Fiasco Relationen nutzen zu können, muß man zuerst ein Projekt erzeugen, das die Quelle für ein anderes Projekt darstellt, das dann daraus die Daten liest. In dem entfernten Projekt müssen dann mindestens zwei Felder erstellt werden, eins für die eigentlichen Daten, das andere für den Schlüssel. Das Feld für den Schlüssel sollte vom Typ Integer sein, da dies am schnellsten ist und von Fiasco automatisch mit Schlüsseln versorgt werden kann. Für besondere Fälle erlaubt Fiasco jedoch auch, jeden anderen Feldtypen zu verwenden.

Um die Option zu aktivieren, die dem Schlüssel-Feld automatisch einen eindeutigen Schlüssel zuweist, muß man im Feld-Requester das Gadget Eindeutiger Schlüssel aktivieren. Bitte beachten Sie, daß der Schlüssel nur beim Erstellen eines Records erzeugt wird! Falls schon Records existierten, bevor diese Option aktiviert wurde, behalten diese ihren alten, und möglicherweise nicht eindeutigen Inhalt. Außerdem kann man später den Inhalt dieses Feldes beliebig verändern, ohne daß irgendeine Überprüfung vorgenommen wird.

Der Typ des zweiten Feldes ist relativ egal. Man sollte sich nur bei den Feldtypen String, Extern und Datatypes das Attribut Maximale Zeichen gut merken, da dieses im Feld des zweiten Projektes genau übereinstimmen muß.

Es wäre sinnvoll, wenn dieses Projekt bereits einige Records mit etwas Inhalt enthalten würde, damit nach dem Aktivieren der Relation sofort ein Effekt sichtbar wird.

Dieses Projekt muß jetzt abgespeichert werden und ein Neues muß erstellt werden.

Das zweite Projekt muß wiederum aus zwei Feldern bestehen, die in Typ und bei den Typen String, Extern und Datatypes im Attribut Maximale Zeichen übereinstimmen. Das Feld für den Schlüssel sollte jedoch *nicht* wieder Eindeutiger Schlüssel aktiviert haben, da dies keinen größeren Sinn machen würde.

Bevor nun die Relation aktiviert werden kann, sollte man das Projekt im selben Verzeichnis wie das erste abspeichern, damit im Relations-Requester keine absoluten, sondern relative – oder in diesem Fall keine Pfade – benutzt werden können.

Um nun die Relation erstellen zu können, müssen Sie das Feld, das *nicht* den Schlüssel beinhalten soll, aktivieren und dazu den Relations-Requester aufrufen (mit Feld/Relationen Ändern). Stellen Sie den Relations-Typ im oberen Cycle-Gadget ein. Nun sollte man zuerst den lokalen Schlüssel im Listview oben links auswählen. Danach muß die Relations-Datei mit dem Filerequester-Gadget ausgewählt werden. Der Requester sollte sich bereits im richtigen Verzeichnis befinden. Nachdem die Datei ausgewählt wurde, werden in den unteren beiden Listviews die möglichen Schlüssel und eigentlichen Felder in der entfernten Datenbank angezeigt. Hier sollte man auch die richtigen Felder auswählen und anschließend Ok anwählen. Hat man alles richtig gemacht werden jetzt die Relationen zu den bereits bestehenden Records geladen.

Eine Relationen-Checkliste, die diese Informationen in komprimierter Form enthält, ist ebenfalls verfügbar.

5.6 Virtuelle Felder

Die Daten von virtuellen Feldern werden nicht auf Disk gespeichert, sondern während des Ladens des Projektes berechnet. Wenn Sie ein Feld virtuell machen möchten, sollten sie die Virtuell Option im Feld-Requester aktivieren.

Fiasco benutzt zum Berechnen dieser Daten die Formel des Feldes. Wann immer ein Record gelesen wird, werden die Inhalte der virtuellen Felder mit dieser Formel berechnet. Mehr über Formeln in Kapitel 11.

Es ist auch möglich, ARexx-Scripts zur Berechnung der Inhalte von virtuellen Feldern zu benutzen. Aus Geschwindigkeits und anderen technischen Gründen wird hiervon jedoch abgeraten. Deshalb ist dieses Verfahren hier auch nicht länger Dokumentiert. Wenn Sie mehr über ARexx und virtuelle Felder erfahren wollen, benutzen Sie die Release 2.0x-Versionen dieses Dokuments.

Kapitel 6

Suchen in einer Datenbank

Fiascos Such-Funktion erlaubt Ihnen, nach spezifischen Daten in einer Datenbank zu suchen. Es gibt zwei Möglichkeiten, nach Daten zu suchen: Sie können entweder Such-Muster für ein oder mehrere Felder angeben, die alle auf die Feld-Inhalte passen müssen. Oder Sie können eine Formel angeben, die komplexe Untersuchungen an dem Inhalt eines Records vornehmen kann.

Der Such-Requester

Die wichtigste Schnittstelle zur Fiasco-Suchfunktion ist der Such-Requester. Er kann mit dem Menüpunkt Vergl./Suchen geöffnet werden. Weiterhin benutzen die Filter-, Zähl-, Ersetz- und Markieren-Funktionen bzw. Menüpunkte diesen Requester.

Um auszuwählen, ob Sie über Feld-Inhalte oder mit Formeln suchen wollen, benutzen Sie das Modus Cycle-Gadget im oberen Teil der Requesters.

Wenn Sie den Feld-Modus benutzen, zeigt der Requester zwei Listview-Gadgets an: Felder zeigt alle Felder in der aktiven Datenbank an. Wenn Sie ein Feld anklicken, wird es in das Listview mit dem Titel Suchen nach übernommen. Unter diesem Gadget können Sie das Such-Muster für das Feld angeben. Um das Feld wieder aus der Liste zu entfernen, benutzen Sie das Löschen Gadget.

Wenn Sie Mit Formel suchen, zeigt das Fenster nur ein String-Gadget für die Formel an. Das Picker-Gadget an der rechten Seite kann zum Öffnen des Formula-Requesters (siehe Abschnitt 10.6.15) benutzt werden.

Mit den Gadgets am unteren Rand kann man die Suche starten. Wenn ein passender Eintrag gefunden wird, wird der entsprechende Record automatisch aktiviert. Die Suche läßt sich hinterher ohne Umweg über den Suchrequester mit den Menüpunkten Vergl./weitersuchen und Vergl./rückwärts suchen fortsetzen.

Mehr über den Such-Requester kann in Abschitt 10.6.17 gefunden werden.

6.1 Suchen mit Feldern

Wenn Sie den Such-Modus Mit Feldern benutzen, müssen Sie ein oder mehrere Felder angeben, die untersucht werden sollen. Für jedes angegebene Feld müssen Sie außerdem ein Suchmuster angeben. Nur wenn der Inhalt aller angegebenen Felder mit den angegebenen Mustern übereinstimmt, wird ein Record von Fiasco als ein passender Record angesehen.

6.1.1 Muster

Suchmuster sind die Daten, die mit den Feld-Inhalten verglichen werden. Für die einzelnen Feld-Typen gibt es verschiedene Regeln für das Angeben von Mustern:

String, Var String, Extern, Datatypes: Der String selbst muß angegeben werden. Weiterhin können Joker (siehe Abschnitt 6.1.2) und unscharfe Suche (siehe Abschnitt 6.1.4) benutzt werden.

Integer, Slider, Float: Die Zahl selbst muß angegeben werden. Weiterhin können Joker für Zahlen (siehe Abschnitt 6.1.3) angegeben werden.

Boolean: true oder 1 geben ein Feld mit Haken an, false oder 0 ein Feld ohne Haken.

Cycle: Der Name des Labels oder die Nummer des Labels (Beginnend bei 0) muß angegeben werden.

Datum, Zeit: Das Datum oder die Zeit muß angegeben werden.

Der Such-Requester hat ein zusätzliches Feature, das das Angeben von Mustern für Felder einfacher macht. Mit dem Picker-Gadget an der rechten Seite des Muster-String-Gadgets können Sie einen Requester öffnen, der eine Liste der möglichen Muster für das Feld anzeigt. Wenn Sie eins auswählen und den Requester bestätigen wird es zum Muster-Gadget kopiert. Die Liste der möglichen Werte hängt auch vom Feld-Typ ab:

String, Integer, Float, Extern, Datat., Datum, Zeit, Var String:
Die vordefinierten Werte.

Slider: Die Maximum- und Minimum-Werte des Sliders.

Boolean: true und false

Cycle: Alle Labels des Cycle-Feldes.

Bitte beachten Sie auch, daß Sie die eingebaute Suchfunktion *nicht* benutzen können, um nach Inhalten von Dateien zu suchen, die beispielsweise in Datatypes- oder Extern-Feldern angegeben sind. Sie können nur nach dem Datei-Namen suchen.

Die von einem Feldtyp unterstützten Muster sind auch nochmal bei den Dokumentationen der jeweiligen Feldtypen aufgeführt.

6.1.2 Joker

Neben Eingaben im Klartext ist auch bei bestimmten Feldtypen die Benutzung von Jokern möglich. String-, Extern- und Datatypes-Felder erlauben die Benutzung von Jokern, die auch von Amiga DOS benutzt werden.

Diese Joker sind möglich:

?	Ein einzelnes unbekanntes Zeichen.
#	Steht für den folgenden Ausdruck null oder mehr mal.
(ab| cd)	Steht für einen der Pukte getrennt von einem | .
~	Negiert den folgenden Ausdruck.
[abc]	Zeichen-Klasse: Steht für jedes der Zeichen in der Klasse.
[~ bc]	Steht für jeder der Zeichen, das nicht in der Klasse ist.
[a-z]	Zeichen-Bereich.
%	Steht für null Zeichen.
*	Synonym für #?. Muß in den Datenbank-Einst. angestellt werden.

Wenn Sie die Zeichen, die als Joker benutzt werden, als normale Zeichen im String benutzen wollen, müssen Sie ein einzelnes Anführungszeichen (', Alt-Ä) dem Zeichen voranstellen.

Wenn Sie Unscharfe Suche (siehe Abschnitt 6.1.4) benutzen, sind nur die Muster ? und #? verfügbar.

Beispiele:

?iasco würde nach Aiasco, Biasco, Ciasco, 1iasco, usw. suchen. ???? würde nach Einträgen suchen, die 4 Buchstaben lang sind. #? steht für null oder mehr unbekannte Zeichen. A#? könnte also Amiga, Afrika, A, ABCD bedeuten. ?#? sucht nach allen Einträgen, die nicht leer sind.

6.1.3 Joker für Zahlen

Integer-, Slider- und Floatfelder unterstützen auch zusätzliche Muster. Folgende sind möglich: >, <, >=, <=, !=. Das Argument muß nach dem Muster gegeben werden. > sucht nur nach Zahlen größer als x, >= nur nach Zahlen größer oder gleich x, < nur nach Zahlen kleiner als x und <= nur nach Zahlen kleiner oder gleich x. != sucht nur nach Zahlen, die ungleich x sind. Ein Muster wie ==, also "gleich" gibt es nicht, da dies durch den Wert ohne jedes Muster dargestellt wird.

6.1.4 Unscharfe Suche

Die unscharfe Suche bietet die Möglichkeit, nach Strings zu suchen, wobei nicht nur nach gleichen Einträgen gesucht wird, sondern nach ähnlichen Einträgen. Wie tolerant die Funktion dabei ist, kann man unter Faktor einstellen. 0 entspricht der niedrigsten Toleranz, dies gleicht der normalen Suche, ist jedoch etwas langsamer. 100 bedeutet, daß fast alle Einträge als "ähnlich" angesehen werden.

Dieses Feature kann nur mit Feld-Typen benutzt werden, die Strings enthalten, also String, Var String, Extern und Datatypes Felder.

Wenn Sie unscharfe Suche benutzen, sind nur die Muster ? und #? verfügbar.

Die Zählfunktion eignet sich hervorragend, um mit der Unscharfen Suche etwas zu experimentieren.

6.2 Suchen mit Formeln

Auch wenn die GUI für den Such-Modus Mit Formeln einfacher als die GUI des anderen Modus aussieht, ist dieser Such-Modus der bei weitem mächtigere. Die im String-Gadget angegebene Formel wird zum Untersuchen jedes Records benutzt. Ein Record wird von Fiasco als passend angesehen, wenn die Formel als Ergebnis einen wahren Wert hat. In Fiasco-Formeln ist dies ein Wert ungleich Null.

Mehr über Fiasco-Formeln kann in Kapitel 11 gefunden werden.

Beispiele:

size > 5
sucht nach allen Records, in denen das Feld size einen Wert größer als 5 hat.

size >= 5 && size <= 10
sucht nach allen Records, in denen size einen Wert in dem Bereich von 5 bis 10 (inklusive) hat.
important && stricmp(name, "Meier") == 0
sucht nach allen Records, in denen das Boolean-Feld important einen Haken hat und das Feld name den String Meier enthält.
datediff(currentdate(), date) <= 10 && datediff(currentdate(), date) >= 0
sucht nach allen Records in denen das Feld date ein Datum enthält, das in den nächsten zehn Tagen liegt.

6.3 Such-Informationen

Such-Infos können benutzt werden, um zeitweise Such-Kriterien zu speichern. Im Such-Requester kann das Name-Gadget benutzt werden, um einen Namen für das Search-Info anzugeben. Die Such-Kriterien werden gesichert, wenn Sie einen der positivien Buttons am unteren Rand des Such-Requesters wählen. Um die gesicherten Kriterien zurückzubekommen, müssen Sie den Picker-Button des Name-Gadgets anklicken und die gewünschten Such-Infos auswählen. Die Kriteren werden dann in den Such-Requester übernommen.

Wenn Sie ein Such-Info mit einem bereits existierenden Namen benutzen, wird das alte überschrieben. Die Such-Infos bleiben bis die Datenbank geschlossen wird gültig.

6.4 Suchen mit ARexx

Auch mit Fiasco's ARexx-Port (siehe Abschnitt 12) können Suchen durchgeführt werden. Die ARexx-Kommandos zum Suchen benutzen Such-Infos (siehe Abschnitt 6.3) um die Such-Kriterien zwischenzuspeichern.

Deshalb müß ein Such-Info vorhanden sein, bevor mit der Such begonnen werden kann. Dieses Such-Infos kann mit NewSearchInfo Name/K,Fields/S,Formula/K,Var/K erzeugt werden. Wenn Sie Name nicht angeben, wird Fiasco einen Namen auswählen, der noch nicht exisitiert und ihn in Result zurückgeben. Dieser Name muß in den Argumenten aller folgenden Such-Kommandos angegeben werden, die sich auf dieses Such-Info beziehen.

Wenn Sie mit einer Formel suchen wollen, müssen Sie diese gleich bei den Argumenten für NewSearchInfo angeben.

Wenn Sie mit Feldern suchen wollen, müssen Sie mit dem Kommando SetSearchField SearchInfo/K/A,FieldID/K/A,Blur/K/N,Pattern/K/A/F ein

Feld angeben, nach dem gesucht werden soll. Sie können dieses Kommando mehrmals aufrufen, um mehrere Felder anzugeben.

Nun können Sie das Kommando Find SearchInfo/K/A,Record/K/N,Reverse/S,All=Stem/K,Var/K benutzen, um nach passenden Records zu suchen. Einmal können Sie das Record-Argument benutzen, um mit jedem Aufruf des Kommandos nach dem nächsten passenden Record zu suchen. Dieses Argument gibt den Record an, nach dem die Suche beginnt. Wenn Sie also Record 0 angeben, wird die Suche beim ersten Record starten. Wenn ein passender Record gefunden wird, wird die Number in Result zurückgegeben. Um die Suche fortzusetzen, können Sie Find mit der Record-Nummer des gefundenen Records aufrufen. Im gegensatz zur GUI-Such-Funktion, werden gefundene Records nicht aktiviert.

Die andere Möglichkeit hängt mit dem Argument All zusammen. Fiasco wird alle Records durchsuchen und die Nummern der gefundenen Records in der Stem-Variablen, die nach All angegeben wurde, ablegen. Die Anzahl der gefundenen Records wird in *Stem.*count abgelegt. Bitte benutzen Sie diese Möglichkeit nicht, wenn eine sehr große Anzahl von Übereinstimmungen zu erwarten ist. Dies würde zu Zeit- und Speicher-Intensiv werden.

6.5 Zählen

Über den Menüpunkt Vergl./Zählen öffnet man einen dem Suchrequester sehr ähnlichen Requester, der sich genaugenommen nur durch Titel und die Gadgets am unteren Rand unterscheidet. Hier kann man wiederum Suchmuster, Bezugsfelder und die Toleranz für die unscharfe Suche angeben. Falls man dann Ok wählt, werden die Übereinstimmungen gezählt und das Ergebnis angezeigt. Natürlich werden auch hier Muster und Toleranz mit einbezogen und beeinflussen so das Ergebnis. So kann man sich auch mal das Verhalten des Toleranz-Faktors genauer betrachten.

6.6 Ersetzen

Vergl./Ersetzen erlaubt es, bestimmte Werte durch einen anderen zu ersetzen. Sie müssen wie immer die Kriterien für die Suche angeben.

Zusätzlich müssen Sie mindestens ein Feld angeben, dessen Inhalt durch einen neuen Inhalt ersetzt wird, wenn ein passender Record gefunden wird. Dieser Ersatz kann entweder ein fester Wert (mit Ersetzen durch Wert) oder das Ergebis einer Fiasco-Formel (Ersetzen durch Ergebnis von Formel) sein.

Mit dem zweiten Modus können Sie Aktualisierungen oder Korrekturen an Ihren Daten vornehmen. Ein Beispiel dafür ist eine Änderung ei-

ner Telefon-Vorwahl oder eine allgemeine Änderung des Vorwahl-Systems. Wenn die Vorwahl 0123 in 0456 würde, müßten Sie die Suchkriterien für die Telefonnummer auf 0123#? setzen. Setzen Sie dann die Ersatz-Formel für die Telefonnummer auf strcat("0456", strmid(test, 4, -1)). Diese Formel schneidet die ersten vier Zeichen der Telefonnummer ab und ersetzt diese durch 0456.

Falls das Gadget Bestätigen aktiviert ist, wird ihnen immer der jeweilige Record gezeigt und Sie werden gefragt, ob sie wirklich den alten Wert ersetzen wollen.

Achtung: Mit einem falschen Muster (z.B. #?) kann man schnell viele Einträge löschen bzw. unbrauchbar machen!

6.7 Filter

Fiascos Filter erlaubt es, nur die Records anzuzeigen, die auf ein bestimmtes Muster passen. Mit Vergl./Filter können sie den Filter-Requester öffnen, der eine ähnliche Struktur wie der Such-Requester besitzt. Wenn Sie die benötigten Parameter eingegeben haben und Ok anklicken, zeigt Fiasco nur noch die Records an, die auf das angegebene Muster passen.

Mit Record/Nächster, Record/Voriger oder deren Äquivalenten können Sie nun diese kleinere Datenbank durchgehen.

Da ein Filter ein Index ist, können Sie einen Namen für den Index im Filter-Requester angeben.

Um den Filter auszuschalten, können Sie die Index-History benutzen. Der Menüpunkt Datenbank/Voriger aktiver Index aktiviert den Index, der aktiv gewesen ist, bevor der Filter aktiv war.

Wenn Sie einen neuen Filter erstellen, während ein anderer Filter noch aktiv ist (wenn Sie z.B. nicht den alten Index im Index-Requester aktiviert haben), werden nur die Records untersucht, die bereits im alten Filter enthalten waren. So können Sie Filter erstellen, deren Bedingungen mit einem Und logisch verknüpft sind.

Der Filter wird gleich nach dem Bestätigen des Requesters erstellt. Das bedeutet, daß neu hinzugefügte Records auch angezeigt werden, auch wenn sie nicht mit dem Filter übereinstimmen. Um zu erreichen, daß auch die neuen Records gefilter werden, muß man zuerst den alten Index auswählen, und dann den Filter neu bauen.

Abschnitt 5.3 enthält mehr Informationen über die fortgeschrittene Benutzung von Filtern.

Kapitel 7

Drucken einer Datenbank

Wenn Sie einen Ausdruck einer Datenbank erstellen möchten, können Sie das auf verschiedene Weisen anstellen. Die interne Druckfunktion ist der komfortabelste Weg. Um die Qualität der Ausdrucke zu steigern, kann TeX mit der Druckfunktion kombiniert werden. Wenn Sie einen Ausdruck erstellen möchten, der nicht mit der Druckfunktion erstellt werden kann, können Sie ein ARexx-Script benutzen.

7.1 Interne Druckfunktion

Der Menüpunkt Projekt/Drucken öffnet das Druck-Fenster von Fiasco. Das Fenster arbeitet ähnlich zu einem Fiasco-Projekt im Masken-Modus. Es enthält Elemente, die mit der Maus angeordnet werden können. Im endgültigen Ausdruck werden alle Records auf diese Weise dargestellt.

Wenn Sie das Druck-Fenster von Fiasco öffnen, versucht Fiasco eine Datei zu öffnen, die die Standard-Druckmaske des Projektes enthält. Der Dateiname für solche Dateien ist *Projekt_Name*.fpr, wobei *Projekt_Name* der Dateiname des Projektes ohne .fdb ist. Wenn die Datei nicht gefunden wird, erstellt Fiasco ein Layout, das zur reellen Maske paßt.

Wenn Sie die Datenbank in Form der Liste ausdrucken wollen, sollten Sie den Menüpunkt Projekt/Von Liste anwählen. Dies wird eine Druckmaske passend zu der echten Liste erstellen. Mit Projekt/Von Maske bekommen sie das Layout der Maske.

Mit Projekt/Drucken können Sie dann das Projekt mit diesem Layout ausdrucken.

7.1.1 Die Druckmaske

Die Druckmaske besteht aus drei Teilen: Dem Kopfteil, dem Hauptteil[1] und dem Fußteil. Der Kopfteil wird vor allen anderen Teilen gedruckt. Der Hauptteil wird für jeden Record einmal gedruckt. Er kann[2] Verweise auf Felder des Projektes enthalten. Diese Verweise werden während des Druckens durch Feldinhalte der Records ersetzt. Der Fußteil wird nach allen anderen Daten gedruckt.

Das Druckfenster zeigt nur einen dieser Teile auf einmal an. Das Kontrolle-Menü dient dazu, einen Teil zum Anzeigen auszuwählen.

Der Gebrauch des Druckfensters ist dem Gebrauch des Projektfensters im Masken-Modus sehr ähnlich. Um ein Element (vergleichbar mit Feldern in der Projekt-Maske) zu erstellen wählen Sie einen Typ mit Element/Typ aus und rufen Element/Hinzufügen aus oder drücken Return. Abhängig vom Typ erscheint ein Requester, der ein paar Optionen für die Felder abfragt. Fiasco unterstützt diese Element-Typen:

- Feld
- Text
- Seitenumbruch

Feld-Elemente können nur im Hauptteil benutzt werden. Mit ihnen können Feldinhalte in den Ausdruck eingefügt werden. Der Requester für Feld-Elemente enthält Gadgets für das Auswählen des Feldes, um Breite, Druckstil, wie Fett, Kursiv oder Unterstrichen, einzustellen oder Abschneiden zu aktivieren. Abschneiden kontrolliert, ob ein Eintrag weiter als die vorgegebene Breite werden darf. Wenn Abschneiden aktiv ist, wird jeder Eintrag, der länger als die Elementbreite ist, verkleinert, um in die Breite zu passen. Wenn Abschneiden nicht aktiv ist, werden die folgenden Einträge verschoben.

Text-Elemente sind Text-Feldern der Datenbank-Maske ähnlich. Mit Text-Elementen kann fester Text in die Druck-Maske eingefügt werden. Sie unterstützen die Druckstile Fett, Kursiv und Unterstrichen. Text-Elemente sind die wichtigsten Elemente in den Kopf- und Fußteilen.

Seitenumbruch-Elemente schließen eine Seite ab. Das bedeutet, daß alle Daten nach einem Seitenumbruch auf einer neuen Seite gedruckt werden. Seitenumbruch-Elemente haben keine veränderbaren Optionen. Deshalb öffnet sich kein Requester nach dem Hinzufügen eines solchen Elementes. Element Ändern ist wirkungslos. Aufgrund der besonderen Bedeutung

[1] Auch wenn Kopf und Haupt Synonyme sind, bezeichnen die Begriffe hier zwei verschiedene Teile ;-)

[2] und sollte

ist die Breite von Seitenumbruch-Elementen “unendlich”. Seitenumbruch-Elemente erscheinen als horizontale Linie in der Maske.

7.1.2 Druckmasken-Dateien

Projekt/Speichern und Projekt/Speichern als des Druckfensters erzeugen Dateien, die die Struktur der Druckmaske enthalten. Diese Dateien können später wieder geöffnet werden, um eine bestimmte Struktur zurückzuerhalten. Wenn Sie in der Zwischenzeit ein Feld in der Datenbank gelöscht haben oder eine Feld-ID geändert haben, kann die Druckmasken-Datei Verweise auf “nichts” haben. Wenn Sie eine solche Datei versuchen zu öffnen, wird Fiasco versuchen, diese Verweise zurückzukriegen. Für diesen Zweck zeigt Fiasco die Feld-ID an, die nicht gefunden wurde, und eine Liste aller Felder des aktiven Projektes. Wenn Sie eins auswählen und Ok anklicken wird der Verweis auf das ausgewählte Feld verändert. Wenn Sie den Requester Abbrechen, wird das Element gelöscht.

Auf diese Weise können Sie einfach Druckmasken auf andere Projekte anpassen. Laden Sie einfach die Datenbank, öffnen das Druckfenster und laden die Druckmaske. Nun können Sie alle Elemente auf die passenden Felder der neuen Datenbank einstellen.

Neben des Layouts enthalten Druckmasken-Dateien die Einstellungen, die im Druckoptionen-Requester (siehe Abschnitt 10.6.28) gemacht wurden.

7.2 Drucken mit TeX

Um hochqualitative Ausdrucke zu erstellen, können Sie TeX benutzen. TeX ist eine Art Programmiersprache, die zum Erstellen von gedruckten Dokumenten benutzt werden kann. TeX wurde ursprünglich von Donald E. Knuth entwickelt. Eine frei kopierbare TeX-Implementation für den Amiga ist PasTeX, das auf beispielsweise auf der Meeting Pearls III CD-ROM gefunden werden kann.

Die Druck-Funktion von Fiasco unterstützt Drucken mit TeX mit dem Umweg über ARexx (siehe Abschnitt 12).

Wenn Sie die Option Mit ARexx drucken im Druckoptionen-Requester (siehe Abschnitt 10.6.28) aktivieren, wird die Funktion des Drucken-Menüpunktes des Druckfensters geändert: Nachdem der Ausdruck erstellt ist, ruft Fiasco das ARexx-Script ARexx/ARexxPrint.rexx mit dem Namen der erstellten Datei auf. Das Script kann dann TeX aufrufen, um die Datei zu übersetzen und auszudrucken. Daher darf die Datei nicht nach PRT: geschrieben werden. Sie sollten Drucke nach im Druckoptionen-Requester

auf eine temporäre Datei setzen, beispielsweise T:FiascoPrintOut.tex. Das Script sollte so oder ähnlich aussehen:

```
/* ARexxPrint.rexx
 * FÜr PasTeX
 */

/* Argumente holen
 */
Parse Arg File

Address Command

File = strip(File,,'"')

/*  Virtex aufrufen
 */
'virtex' '"' || File || '"'

/*  Erstelle Name der DVI-Datei
 */
dotpos = lastpos(".", File)

if dotpos ~= 0 then
    DVIFile = substr(File, 1, dotpos-1) || ".dvi"
else
    DVIFile = File || ".dvi"

/*  DVIPrint aufrufen
 */
'dviprint' '"' || DVIFile  || '"'

/* Temporäre Dateien löschen
 */
call delete(file)
call delete(dvifile)
```

Wenn Sie mit TeX drucken wollen, müssen Sie die Druckmaske in einer TeX-Kompatiblen Weise gestalten. Zum Beispiel müssen Sie einen Text wie \documentstylearticle oder Ähnliches in den Kopfteil setzen, wenn Sie mit LaTeX arbeiten. Weiterhin darf die Datei keine Steuerzeichen enthal-

ten. Druckstil-Attribute und Seitenumbruch-Elemente können also nicht benutzt werden. Die Fiasco-Distribution enthält mehrere Beispiele dafür.

7.3 Drucken mit ARexx

"Drucken mit ARexx" ist ein sehr umfassendes Thema. Dieser Abschnitt soll nur einen groben Eindruck davon geben, was getan werden kann und wie.

Ein Weg des Druckens mit ARexx (siehe Abschnitt 12) wurde bereits im Abschnitt Drucken mit TeX erklärt. Sie können ARexxPrint.rexx auch für andere Zwecke "mißbrauchen". So können sie beispielsweise ein Script benutzen, das die Daten für eigene Zwecke liest oder die Daten in Ihr Textverarbeitungsprogramm liest.

Wenn Sie komplexere Ausdrucke erstellen möchten, die nicht mit der internen Druckfunktion von Fiasco erstellt werden können, müssen Sie auf ARexx allein zurückgreifen. Ein solches ARexx-Script muß durch die ganze Datenbank gehen und die benötigten Daten mit GetField holen. Danach kann es mit den Daten machen was es will.

7.3.1 GraphPrint.rexx

Die Fiasco-Distribution enthält ein komplexes Beispiel für ein Script, wie es im vorigen Abschnitt beschrieben wurde. Das Script GraphPrint.rexx liegt im Verzeichnis ARexx und kann mit dem GraphDemo-Projekt benutzt werden. Es kann jedoch auch mit jedem anderen Projekt benutzt werden, das die erforderlichen Daten enthält. Das Script liest Daten aus dem Projekt und erstellt ein X/Y Diagramm der Daten. Es paßt sich automatisch an verschiedene Wertebereiche an. Das Script benutzt LaTex und die eepic Erweiterung für den Ausdruck. Das bedeutet, daß Sie ein Special Host-Programm im Hintergrund während des Druckens laufen haben müssen. Da das Script viele mathematischen Operationen ausführt, benutzt es die rexxmathlib.library, die nicht mit Fiasco mitgeliefert wird.

Um GraphPrint.rexx zu starten, müssen Sie den Graphic-Schalter im GraphDemo/Fragments-Project anklicken. Wenn Sie das Script mit einem anderen Projekt benutzen möchten, müssen Sie dies nur aktivieren und dann das Script über Workbench oder Shell starten. Bevor der Ausdruck beginnt werden Ihnen mehrere Fragen gestellt. Sie müssen angeben, welche Felder benutzt werden sollten. Sie können auch angeben, ob die die TeX-Datei direkt ausdrucken oder ansehen möchten oder die Daten in eine angebene Datei zu schreiben. Danach erschein das Advanced Options-Menü. Um nichts zu verändern klicken Sie einfach auf Continue. Edit Scale Base

erlaubt es Ihnen, einen Wert anzugeben, der vom Script als ein Basis-Wert für die Skala einer Achse benutzt wird. Wenn Sie beispielsweise 5 benutzen (was voreingestellt ist), werden Sie eine Skale von 5, 10, 15, etc. oder Vielfachen davon bekommen. Wenn Sie 2 benutzen kriegen sie 2, 4, 6, etc. oder Vielfache. Edit Origin läßt Sie auswählen, ob das Diagramm beim Punkt (0;0) oder an einem Punkt beginnt, der am Bestem zum Projekt paßt.

Kapitel 8

Import und Export

Die Import und Export-Funktionen von Fiasco machen es möglich, Daten von anderen Datenbank-Programmen in Fiasco zu laden und Daten von Fiasco für andere Programme lesbar zu machen.

Solche Import/Export-Dateien enthalten die Daten kodiert im ASCII-Format. Die Felder oder Records sind durch besondere Zeichen markiert, die in der Import/Export-Funktion von Fiasco frei definiert werden können.

An den Anfänger: Um die Import/Export Funktion von Fiasco nutzen zu können, müssen Sie ein paar Details kennen. Im Folgenden wird die Struktur von Import/Export-Dateien beschrieben. Wenn Sie dieses bereits kennen, brauchen Sie das nicht zu lesen. Der Abschnitt danach beschreibt die besonderen Escape-Sequenzen, die von Fiasco benutzt werden. Obwohl andere Datenbanken ein ähnliches Schema benutzen können, sollten Sie sich dies genau durchlesen, da die gesamte Import/Export-Funktion von Fiasco darauf beruht.

8.1 Struktur von Import/Export-Dateien

Die hier benutzen Namen beziehen sich auf die Import/Export-Requester. Bitte beachten Sie, daß manchen Markierungs-Zeichen leer sein könne. Um die Datei benutzen zu können, müssen Sie mindestens entweder Feld Start/Feld Ende oder Feld Trenner und entweder Record Start/Record Ende oder Record Trenner angeben.

Die Import-Funktion von anderen Programmen kann jedoch möglicherweise auch verwirrt werden, auch wenn diese Regeln eingehalten werden. Wenn Sie Daten exportien, filtert Fiasco die Zeichen aus den Daten aus, die als Kontroll-Zeichen benutzt werden. Wenn ein Feld beispielsweise 1,2

enthält und das , als Kontroll-Zeichen benutzt wird, wird Fiasco 12 exportieren.

Record Start	
Feld Start	
Feld Daten	Inhalt des Feldes in ASCII-Format
Feld Ende	
Feld Seperator	Trennt zwei Felder, nicht nach dem letzten Feld eines Records
...	
Feld Start	
Feld Daten	
Feld Ende	
Record Ende	
Record Seperator	Trennt zwei Records, nicht nach dem letzten Record einer Datei.
...	
Record Start	
...	(siehe oben)
Record Ende	
Ende der Datei	

Wenn Sie Erster Record enthält IDs aktivieren, werden die IDs der Felder im ersten Record gespeichert, als ob die Felder wären.

Wenn Sie Listview-Felder exportieren, werden die Einträge der Listviews mit den Zeichen getrennt, die in Listview/Trenner angegeben sind.

Ein Beispiel einer Import/Export-Datei

Record Start und Record Ende sind leer. Record Seperator ist eine Newline. Feld Start und Feld Ende sind Anführungszeichen. Feld Seperator ist ein Komma. Der erste Record enthält die IDs der Felder. Beachten Sie das leere Feld im letzten Record.

```
"Name","FirstName","Rank","Ship"
"Picard","Jean-Luc","Captain","U.S.S. Enterprise"
"Riker","William Thomas","Commander","U.S.S. Enterprise"
"Data","","Lieutenant Cmdr.","U.S.S. Enterprise"
```

8.2 Besondere Zeichen

Oft können die Zeichen fürs Markieren der Felder und Records nicht als einfacher Text eingegeben werden. Beispielsweise kann man für das Newline-Zeichen nicht einfach die Return-Taste betätigen. Stattdessen muß dieses Zeichen als Escape-Sequenz eingegeben werden. Fiasco unterstützt

Escape-Sequenzen ähnlich denen der Programmiersprache "C". Die Escape-Sequenzen werden von einem \ eingeleitet. Diese werden unterstützt:

\n	Newline-Zeichen, ASCII 10
\f	Formfeed-Zeichen, ASCII 12
\r	Return-Zeichen, ASCII 13
\t	Horizontaler Tabulator, ASCII 9
\v	Vertikaler Tabulator, ASCII 11
Nummer	Zeichen mit angegebenem ASCII Code
Zeichen	Zeichen direkt übernommen

Die letzte Option (\ + *Zeichen*) macht es möglich, Zeichen zu benutzen, die für Escape-Sequencen reserviert sind (z.B. \\ für einen \).

In der Import-Fuktion können auch Zeichen-Klassen angegeben werden. Zeichen-Klassen werden in Fiasco mit einem # markiert. Diese werden unterstützt:

#p	Druckbares Zeichen.
#a	Druckbares ASCII-Zeichen ohne internationale Zeichen.
#c	Kontrol-Zeichen. Nicht Druckbar

Export unterstützt das Einfügen einiger zusätzlicher Informationen in der Export-Datei. Diese Kommandos werden durch ein % markiert. Diese werden unterstützt:

%f	ID des Feldes.
%r	Nummer des Records

8.3 Importieren von Daten

Der Import-Requester ist die GUI-Schnittstelle zur Import-Funktion von Fiasco. Sie können den Requester mit Projekt/Importieren öffnen. Die zu importierende Datei muß in Datei angegeben werden. Nachdem Sie dies getan haben müssen Sie die Struktur der Datei im Requester angeben. Wenn Sie die Datei gerade aus einer anderen Datenbank exportiert haben und die Struktur-Parameter immer noch kennen, können Sie sie einfach übernehmen. Wenn dies nicht der Fall ist, können Sie die Datei mit dem View-Schalter betrachten. Fiasco wird entweder More oder Multiview starten, um die Datei anzuzeigen. Wenn die Datei eine standardmäßige Struktur hat, sollte es nicht zu schwer sein, die Parameter zu erkennen.

Normalerweise sind Record Start und Record Ende leer und Record Separator ist \n. Feld Start und Feld Ende sind oft leer oder Anführungszeichen ("). Gewöhnliche Werte für Feld Separator sind ein Komma (,) oder ein Tabulator (\t).

Zeilen Überspringen definiert die Zeichen, die einen Kommentar *am Beginn einer Zeile* markieren. Dies kann auch benutzt werden, um Formatierungs-Informationen zu überspringen, die Fiasco nicht auswerten

kann. Mit Am Start überspringen kann man Kommentare oder Ähnliches am Anfang der Datei überspringen. Max. Felder kann zum Definieren einer Record-Ende-Markierung benutzt werden, wenn weder Record Trenner noch Record Ende zur Verfügung stehen.

Akitivieren Sie Erster Record enthält IDs, wenn der erste Record der Datei nicht aus echten Daten, sondern den IDs der jeweiligen Felder besteht. Wenn dies aktiv ist, wird Fiasco die IDs benutzen, um entweder neue Felder zu erstellen, oder um die bereits bestehenden Felder zu benutzen.

Die Optionen Neue Felder anhängen und Altes Projekt überschreiben kontrollieren, ob Sie ein Projekt aktualisieren wollen, oder ein neues erstellen wollen. Wenn Sie ein neues Projekt erstellen wollen, sollten Sie beide Optionen aktivieren.

Wenn Sie die Einstellungen, die Sie gemacht haben, in der Zukunft weiterbenutzen wollen, können Sie diese mit Save speichern. Die Einstellungen können dann mit Load wieder geladen werden. Die Fiasco-Distribution enthält bereits mehrere vorgefertigte Einstellungen.

Um den Import-Vorgang zu starten, müssen Sie einfach auf Ok klicken. *Achtung:* Wenn die Import-Datei zu groß ist oder die Struktur-Parameter falsch sind, kann Fiasco allen freien Speicher an sich reißen. Fiasco hat zwar keine größeren Probleme bei RAM-Mangel, aber andere Programme können Probleme haben.

Wenn alles gut ging, schließt sich der Import-Requester und das neue Projekt wird aktiviert. Sie werden wahrscheinlich zuerst im Masken-Modus das Layout der Maske verändern, das von Import nur sehr einfach angelegt wurde. Wenn Sie Erster Record enthält IDs nicht aktivert hatten, sollten Sie die IDs passend zu den Feldinhalten ändern. Zusätzlich sollten Sie Text-Felder erzeugen, um den existierenden Feldern eine gewisse Beschreibung zukommen zu lassen. Nun haben Sie schon eine schön formatierte Maske. Alle Felder sind bis jetzt jedoch noch String-Felder. Nun sollten Sie untersuchen, ob manche Felder Integer-, Cycle- oder oder andere Felder sein können. Der Typ dieser Felder kann dann mit der Umwandlungs-Funktion (siehe Abschnitt 5.1) von Fiasco verändert werden. Im Beispiel, das in Sektion 8.1, gegeben wurde, kann das Rank-Feld zu einem Cycle-Feld umgewandelt werden.

Wenn Sie diese Schritte erledigt haben, sollten Sie das Projekt unter einem passenden Namen speichern.

8.4 Exportieren von Daten

Das Exportieren von Daten ist aus der Sicht von Fiasco viel weniger kompliziert als das Importieren. Normalerweise können Sie die Standard-

Parameter von Fiasco, d.h. kein Record Start und Record Ende, ein Newline für Record Trenner, Anführungszeichen für Feld Start und Feld Ende und ein Komma für Feld Trenner. Wenn Sie diese Parameter benutzen, müssen Sie beachten, daß die Daten keinerlei Anführungstriche enthalten. Zusätzlich müssen Sie darauf achten, daß das Programm, das die Daten lesen soll diese Konfiguration auch unterstützt.

Wenn Sie Erster Record enthält IDs anwählen, wird Fiasco einen zusätzlichen Record am Anfang der Datei erzeugen, der die IDs der Felder statt den eigenlichen Daten enthält. Die Datei wird keine andere Formatierungs-Information enthalten.

Wenn Sie Nur Markierte Records anwählen, werden nur die Records geschrieben, bei denen die Markierung gesetzt ist.

Um das Exportieren zu starten, klicken Sie auf Ok.

8.5 Aktualisieren von Datenbanken mit Im/Export

Fiascos Import- und Export-Funktion kann auch benutzt werden, um automatisch Daten, die in anderen Datenbanken erfaßt wurden, in eine bereits existierende Datenbank einzufügen. Um dies zu tun, müssen Sie eine Datenbank in einem Export-Format haben, das Fiasco lesen kann. Der erste Record dieser exportierten Datenbank muß die Feld-IDs der folgenden Daten beinhalten. Diese IDs müssen mit den IDs in der bereits existierenden Fiasco-Datenbank, in die die Daten importiert werden sollen, übereinstimmen. Wenn Sie beispielsweise ein Feld mit der ID Name in der existierenden Fiasco-Datenbank haben, muß die Feld-ID im entsprechenden Feld in der exportierten Datenbank auch Name sein. Wenn dies nicht der Fall ist, können Sie mit einem Text-Editor einfach die IDs in der exportierten Datenbank anpassen[1]. Die bereits bestehende Datenbank kann Felder enthalten, die nicht in der zu importierenden Datei sind. Wenn die Datei Felder enthält, die nicht in der Fiasco-Datenbank sind, wird Fiascos Import-Funktion ein neues Feld für diese Daten erzeugen.

Um das Importieren zu beginnen, müssen Sie die passenden Einstellungen für die Struktur der Datei im Import-Requester vornehmen. Weiterhin muß dort Erster Record enthält IDs aktiv sein (da Fiasco sonst nicht die Daten den Feldern zuordnen kann). Neue Felder anhängen und Altes Projekt überschreiben müssen ausgeschaltet sein.

Diese Methode kann sowohl bei Daten die schon von Fiasco exportiert

[1] Beachten Sie, daß manche Editoren bestimmte Zeichen (z.B. Tabulatoren) von selbst ändern können. Der Workbench-MEmacs macht dies nicht.

wurden als auch bei Daten die von jedem anderen Programm erzeugt wurden, so lange sie in einem Format vorliegen, das von Fiasco gelesen werden kann.

Kapitel 9

Feldtypen

Felder sind die eigentlichen Speicher für Daten. Dazu reichen eigentlich die Grundtypen “String” und “Zahl”. Alle anderen Typen sind mehr oder weniger Modifikationen dieser Grundtypen, die die Arbeit mit der Datenbank erleichtern.

Manche Feldtypen können in Listview-Feldern (siehe Abschnitt 9.17) benutzt werden. Diese Felder können mehrere Einträge enthalten, die hinzgefügt oder gelöscht werden können. Jeder Eintrag kann wie ein normales Feld dieses Typs genutzt werden.

Fiasco unterstützt zur Zeit folgende Feldtypen:

- String
- Integer
- Float
- Boolean
- Cycle
- Slider
- Datum
- Zeit
- Extern
- Datatypes
- Var String

- Text
- Button
- Leiste
- Listview

Die Beschreibungen für die Feldtypen basieren auf den Standard-Attributen, die im nächsten Abschnitt behandelt werden. Jeder Feldtyp kann neue Attribute definieren, ein Attribut verändern oder es sogar nicht unterstützen.

9.1 Standard-Attribute

Diese Attribute werden normalerweise von einem Feldtyp unterstützt:

Struktur/ID: Hier muß man einen String angeben, der für die Identifikation des Feldes dient. Dieser String darf nur einmal in einem Projekt vorkommen. Im Masken-Modus wird dieser String in den Feldern angezeigt. Weiterhin wird er in der Kopf-Zeile der Liste, beim Suchen, dem Relations-Requester usw. angezeigt. Um das Feld über seine ID von Formeln oder ARexx-Scripts aus anzusprechen, treffen weitere Einschränkungen betreffend des Formats der ID zu: Die ID darf keine Leerzeichen enthalten und darf nicht mit einer Zahl beginnen.

Struktur/Virtuell: Der Inhalt des Feldes wird nicht auf Diskette gespeichert, sondern jedesmal, wenn das Projekt geladen wird, neu berechnet. Dies wird mit den Startwert-Attributen und dem Formel-Attribut oder dem ARexx-Script-Attribut erledigt. Bitte beachten Sie, daß diese Felder dieselbe Menge an RAM wie normale Felder beanspruchen.

Struktur/Listview:
Das Feld wird ein Listview-Feld (siehe Abschnitt 9.17). Dieses Attribut kann nur geändert werden, wenn das Feld erstellt wird. Um das Listview-Attribut später zu ändern, müssen Sie die Feld-Umwandlungs-Funktion (siehe Abschnitt 5.1) benutzen. Da Listview-Felder nicht im Listen-Fenster angezeigt werden können, werden die Listen-Fenster Attribute gesperrt, wenn dieses Attribut aktiv ist.

Masken-Fenster/Breite: bestimmt die Breite in Zeichen, mit der das Feld in der Maske erscheint.

Masken-Fenster/Höhe: bestimmt die Höhe in Zeichen, mit der das Feld in der Maske erscheint.

Masken-Fenster/Tastatur-Abkürzung: Hier können sie ein Zeichen angeben. Wenn Sie im Record-Modus dieses Zeichen auf der Tastatur drücken, wird das Feld aktiviert oder geändert. Dieses Zeichen mit Shift kann das Feld in die andere Richtung ändern.

Masken-Fenster/Ausrichtung: Kontrolliert, ob der Inhalt des Feldes linksbündig, rechtsbündig oder zentriert angezeigt wird.

Masken-Fenster/Nur Lesbar: Der Inhalt des Feldes wird in einer zurückgelegten Box angezeigt, die nicht aktiviert oder verändert werden kann.

Masken-Fenster/Verborgen: Wenn diese Option aktiv ist, wird das Feld in der Maske nicht sichtbar. Um den Feldrequester für verborgene Felder zu öffnen, müssen Sie den Menüpunkt Feld/Benanntes Feld Ändern benutzen.

Listen-Fenster/Breite: Gibt die Breite des Feldes im Listen-Fenster an. Die Breite wird in Zeichen gemessen. Sie können diesen Wert auch direkt im Listen-Fenster (siehe Abschnitt 3.5) ändern.

Listen-Fenster/Ausrichtung: Kontrolliert, ob der Inhalt des Feldes im Listen-Felder linksbündig, rechtsbündig oder zentriert angezeigt wird.

Listen-Fenster/Verborgen: Die Spalte für das Feld wird nicht angezeigt, wenn diese Option aktiv ist. Sie können diese Option auch direkt im Listen-Fenster (siehe Abschnitt 3.5) ändern.

Startwert/Eigener Wert: Hier kann man einen Wert angeben, der beim Neuerstellen eines Records in dieses Feld kopiert wird.

Startwert/Alter Wert: Wenn man einen Record neu erstellt, wird der Wert des alten Records übernommen. Wenn Alter Wert aktiv ist, wird Eigener Wert ignoriert. Dies ist insbesondere beim Eingeben vieler ähnlicher Datensätze nützlich.

Vordefinierte Werte: Dieses Feature erlaubt es Ihnen, einen Auswahl-Schalter dem Feld in der Maske hinzuzufügen. Dieser Schalter kann benutzt werden, um aus einer Liste von vordefinierten Werten einen auszuwählen, auf den das Feld dann gesetzt wird. Anklicken des Vordefinierte Werte-Schalters im Feldrequester bewirkt, daß ein neuer Requester geöffnet wird, in dem Sie die Einstellungen für dieses Attribut

machen können. Entweder können Sie eine Liste von möglichen Werten für das Feld angeben oder ein ARexx-Script angeben, das ausgeführt werden soll, wenn der Auswahl-Schalter betätigt wird. Dieses Script kann zum Beispiel einen Datei-Requester öffnen und mit SetField das entsprechende Feld auf die ausgewählte Datei setzten. Die Liste der vordefinierten Werte kann auch vom Such-Requester aus mit dem Auswahl-Schalter an der rechten Seite des Muster-Gadgets aufgerufen werden.

Programmierung/Formel: Sie können hier eine Formel angeben, die benutzt wird, um den Inhalt des Feldes zu berechnen. Die Formel kann auf andere Felder zugreifen, um die Berechnung zu machen. Wann immer eines dieser Felder geändert wird, oder ein komplett neuer Record hinzugefügt wird, wird die Formel erneut berechnet. Dieses Attribut kann sehr gut mit dem Virtuell-Attribut benutzt werden. Mehr über Formeln im Kapitel 11.

Programmierung/ARexx-Script: Hier läßt sich ein ARexx-Script angeben, das aktiviert wird, wenn ein neuer Record erstellt wird, bzw. der Inhalt des jeweiligen Feldes geändert wird. Abhängig vom Inhalt des Scripts kann der eingestellte Startwert von seinem eigentlichen Sinn abweichen. Auf das Script wird relativ von dem Verzeichnis zugegriffen, in dem das Projekt liegt. Wenn die Aufgabe des Scripts auch mit Formeln gemacht werden kann, ist es sehr empfehlenswert, auch Formeln zu benutzen.

Durch Dehnungswerte (siehe Abschnitt 10.1.1) ungleich 0 können die Dimensions-Attribute verzerrt werden.

9.2 Feldtypen

9.3 Der String-Feldtyp

In einem String-Feld lassen sich Zeichenketten begrenzter Länge eingeben.

Neue Attribute:

Struktur/Maximale Zeichen: bestimmt die maximale Anzahl von Zeichen, die in das Feld eingegeben werden dürfen. Dieses Attribut wirkt sich direkt auf die Größe der Datei aus.

Geänderte Attribute:

Masken-Fenster/Höhe: Dieses Attribut ist nur aktiv, wenn das Listview-Attribut aktiv ist.

Suchäquivalent:

entspricht dem Feldinhalt.

Unterstützte Suchmuster:

?	-	Ein unbekanntes Zeichen.
#?	-	Null oder mehr unbekannte Zeichen.

Umwandlung in ein String-Feld:

Alle Felder können ohne Datenverlust in String-Felder umgewandelt werden. Alternative Formate sind in Klammern angegeben, wenn sie unterstützt werden.
Zusätzliche Anmerkungen:

Boolean	-	"Checked" wird TRUE (1), sonst FALSE (0)
Cycle	-	Label (Labelnummer) umgewandelt
Slider	-	Level umgewandelt
Date	-	Datum im aktuellen Locale-Format umgewandelt
Time	-	Zeit im aktuellen Locale-Format umgewandelt

9.4 Der Integer-Feldtyp

In einem Integer-Feld lassen sich Ganzzahlen im Bereich von -2.147.483.348 bis 2.147.483.347 eingeben.

Neue Attribute:

Struktur/Maximale Zeichen: bestimmt die maximale Anzahl von Zeichen, die in das Feld eingegeben werden dürfen.

Startwert/Eindeutiger Schlüssel: legt in diesem Feld bei Neuerstellung eines Records eine – für diese Datenbank – einmalige Identifikations-Nummer ab. Bitte beachten Sie, daß diese frei geändert werden kann und so die Eindeutigkeit nicht gewährleistet ist.

Geänderte Attribute:

Masken-Fenster/Höhe: Dieses Attribut ist nur aktiv wenn das Listview-Attribut aktiv ist.

Suchäquivalent:

entspricht dem Feldinhalt.

Unterstützte Suchmuster:

>	-	größer als
<	-	kleiner als
>=	-	größer oder gleich
<=	-	kleiner oder gleich
!=	-	ungleich

Umwandlung in ein Integer-Feld:

Integer-Felder akzeptieren nur den numerischen Teil der Quelldaten. Wenn die Quelldaten mit einem nicht numerischen Teil beginnen, wird das Feld 0 enthalten.
Zusätzliche Bemerkungen:

Float	-	Ganzzahliger Teil umgewandelt
Boolean	-	"Checked" wird 1, sonst 0
Cycle	-	Nummer des Labels umgewandelt
Slider	-	Level umgewandelt

9.5 Der Float-Feldtyp

In ein Float-Feld können reelle Zahlen eingegeben werden.

Neue Attribute:

Präzision: Anzahl der darzustellenden Nachkommastellen.

Geänderte Attribute:

Masken-Fenster/Höhe: Dieses Attribut ist nur aktiv, wenn das Listview-Attribut aktiv ist.

Suchäquivalent:

entspricht dem Feldinhalt

Umwandlung in ein Float-Feld:

Float-Felder akzeptieren nur den numerischen Teil der Quelldaten. Wenn die Quelldaten mit einem nicht numerischen Teil beginnen, wird das Feld 0 enthalten.
Zusätzliche Bemerkungen:
Boolean - "Checked" wird 1.0, sonst 0.0
Cycle - Nummer des Labels umgewandelt

Bemerkungen:

Seit Fiasco 2.1 wird für Float-Felder intern ein 64-Bit-Format benutzt. Diese Eigenschaft erlaubt nun eine ziemlich hohe Präzision für Float-Felder. Fiasco-Datenbanken in einem älteren Format werden während des nächsten Speichern automatisch konvertiert.

9.6 Der Boolean-Feldtyp

Ein Boolean-Feldtyp nimmt Wahrheitswerte auf, d.h. entweder Ja/True oder Nein/False. Ein Boolean-Feld erscheint in der Maske als ein "Checkbox-Gadget".

Geänderte Attribute:

Struktur/Listview: nicht anwendbar.

Masken-Fenster/Ausrichtung: nicht anwendbar.

Masken-Fenster/Breite: ist immer 3.

Masken-Fenster/Höhe: ist immer 1.

Vordefinierte Werte: nicht anwendbar.

Suchäquivalent:

TRUE oder 1 - Gesetzter Haken
FALSE oder 0 - Kein Haken

Umwandlung in ein Boolean-Feld:

Boolean-Felder nehmen alle Zahlen ungleich 0 und TRUE als gesetzt. Alle anderen Werte werden nicht gesetzt.

Bemerkungen:

Unter Amiga OS 2.0 kann das Aussehen dieses Feldes leicht kaputt wirken, da die Grafiken noch nicht skalierbar sind. Unter Amiga OS 3.0 oder höher werden die Grafiken korrekt an die Größe der Schrift angepaßt.

9.7 Der Cycle-Feldtyp

Der Cycle-Feldtyp bietet mehrere Auswahlmöglichkeiten aus einer selbst definierten Liste. Der Größte Vorteil beim Cycle-Feldtyp ist die Speicherersparnis. Es werden maximal 65536 Auswahlmöglichkeiten unterstützt. (Ich hoffe, daß das reicht ;-) Ein Cycle-Feld erscheint in der Maske als ein "Cycle-Gadget" (wie der Name schon sagt).

Neue Attribute:

Labels: Eine Liste aller Auswahlmöglichkeiten. Ein Eintrag ist Pflicht, ab zweien wird das Cycle-Feld sinnvoll.

Geänderte Attribute:

Struktur/Listview: nicht anwendbar.

Masken-Fenster/Ausrichtung: immer zentriert.

Masken-Fenster/Höhe: immer 1.

Vordefinierte Werte: nicht anwendbar.

Suchäquivalent:

entweder die Nummer des Labels von Null zählend, oder der Eintrag im Klartext (nicht vertippen!)

Umwandlung in ein Cycle-Feld:

Die Werte werden in Labels umgewandelt. Wenn es gleiche Werte gibt, werden diese das selbe Label benutzen. Daten gehen dabei nicht verloren.
Zusätzliche Bemerkungen:
Boolean - "Gesetzt" wird TRUE(1), sonst FALSE(0)

9.8 Der Slider-Feldtyp

Ein Slider-Feld ist eine Abwandlung des Integer-Typs. Mit ihm können Ganzzahlen grafisch dargestellt und verändert werden. Der Bereich der Zahlen ist auf -32.768 bis 32.767 begrenzt, kann jedoch durch mehrere Attribute beeinflußt werden.

Neue Attribute:

Struktur/Min. Wert: gibt den niedrigsten Wert der dargestellt wird an. Dieser Wert entspricht der Stellung des "Knobs" am linken bzw. am oberen Rand des Feldes.

Struktur/Max. Wert: gibt den höchsten Wert der dargestellt wird an. Dieser Wert entspricht der Stellung des "Knobs" am rechten bzw. am unteren Rand des Feldes.

Masken-Fenster/Format: ist ein Formatstring im Stil der Programmiersprache C. Er baut sich wie folgt auf:

%[-][0][ZahlenFeld][.Maximum][l]Format

- -: Die Zahl wird linksbündig im Feld geschrieben, normal rechts
- 0: Das Feld wird mit Nullen aufgefüllt. also: 1 wird 001
- ZahlenFeld: Die minimale Feldbreite
- Maximum: nur für Strings, nicht wichtig hier
- l: Zeigt an, daß die Zahl 32 bit breit ist, muß hier immer angegeben werden.
- Format:
  c - Char, das ASCII-Zeichen für die Zahl wird ausgegeben
  d - Die Zahl wird ausgegeben.
  u - Die Zahl wird vorzeichenlos ausgegeben.
  x - Die Zahl wird im Hexadezimal-Format ausgegeben.
  Es gibt außerdem die Format-Zeichen b und s. Da diese Adressen als Argumente benötigen, produzieren sie in diesem Fall nur Müll!

Die Formatierung erfolgt mit der Exec-Funktion RawDoFmt().

Masken-Fenster/Max. Format-Länge: gibt die maximale Länge der Format-Ausgabe an, dieser Bereich wird bei der Breite angerechnet, d.h. je größer MaxFormatLen, desto kleiner das eigentliche Feld.

Geänderte Attribute:

Struktur/Listview: nicht anwendbar.

Masken-Fenster/Ausrichtung: nicht anwendbar.

Masken-Fenster/Höhe: immer 1.

Vordefinierte Werte: nicht anwendbar.

Suchäquivalent:

Die eigentliche Zahl.

Unterstützte Suchmuster:

>	-	größer als
<	-	kleiner als
>=	-	größer oder gleich
<=	-	kleiner oder gleich
!=	-	ungleich

Umwandlung in ein Slider-Feld:

Integer-Felder akzeptieren nur den numerischen Teil der Quelldaten. Wenn die Quelldaten mit einem nicht numerischen Teil beginnen, wird das Feld 0 enthalten. Nach der Umwandlung sollten sie die Max./Min. Attribute überprüfen.

9.9 Der Datums-Feldtyp

In ein Datums-Feld kann ein Datum eingegeben werden. Dabei wird das Datum nach den Angaben in den aktiven Locale-Einstellungen formatiert. Wenn die locale.library nicht verfügbar ist, wird das Format TT.MM.JJJJ benutzt.

Neue Attribute:

Startwert/Aktuelles Datum: Beim Neuerstellen eines Records wird das aktuelle Datum in dieses Feld eingefügt.

Geänderte Attribute:

Mask Window/Height: Dieses Attribut ist nur aktiv wenn das Listview-Attribut aktiv ist.

Suchäquivalent:

entspricht dem Feldinhalt

Umwandlung in ein Datums-Feld:

Datum-Felder benötigen die Daten im aktuellen Locale-Format. Die einzelnen Teile müssen Zahlen sein. Wenn die Werte nicht numerisch sind, wird der Teil ??.

Bemerkungen:

Auch wenn es mit den meisten Formaten möglich ist, ein zweistelliges Jahr anzugeben, wird empfohlen, ein vielstelliges Jahr zu benutzen, um jegliche Probleme zu vermeiden, die die DOS-Welt zur Zeit hat.

9.10 Der Zeit-Feldtyp

In ein Zeit-Feld kann eine Uhrzeit eingegeben werden. Dabei wird die Zeit nach den Angaben in den aktiven Locale-Einstellungen formatiert. Wenn die locale.library nicht verfügbar ist, wird das Format HH:MM:SS benutzt.

Neue Attribute:

Struktur/Zeitspannen-Format: Normalerweise wird das Format der locale.library für die Zeit benutzt. In manchen Ländern jedoch, (insbesondere in den USA) ist dieses Format jedoch nicht brauchbar, um Zeitspannen anzuzeigen. Wenn dieses Attribut aktiv ist, wird immer das Format HH:MM:SS benutzt. Weiterhin wird die Bereichsüberprüfung für die Stunde ausgeschaltet. So können Sie auch eine Zeit länger als 24 Studen eingeben.

Startwert/Aktuelle Zeit: Beim Neuerstellen eines Records wird die aktuelle Zeit in dieses Feld eingefügt.

Geänderte Attribute:

Mask Window/Höhe: Dieses Attribut ist nur aktiv wenn das Listview-Attribut aktiv ist.

Suchäquivalent:

entsprichten dem Feldinhalt.

Umwandlung in ein Zeit-Feld:

Zeit-Felder benötigen die Daten im Format HH:MM:SS. Jedes Element muß eine Zahl sein. Wenn ein Element nicht numerisch ist, wird es 0.

9.11 Der Extern-Feldtyp

Ein Extern-Feld nimmt einen String (meistens Dateinamen) auf, der auf Wunsch an ein benutzerdefinierbares Programm als Argument weitergeleitet wird. So kann man weitere Daten zu einem Record definieren und abrufen.

Neue Attribute:

Struktur/Maximale Zeichen: bestimmt die maximale Länge des Dateinamens. Dieses Attribut wirkt sich direkt auf die Dateigröße aus.

Kommando/Kommando: ist der Name eines Programms, mit dem die Daten verarbeitet werden können. Die Zeichenfolge `%s` wird bei Aufruf durch den Inhalt des Feldes ersetzt. Läßt man `%s` weg, so wird kein Argument übergeben. (Beispiel: `C:ED %s`)

Kommando/Stack: Gibt die Größe des Stacks für das Kommando an.

Masken-Fenster/DateiReq Gadget: Aktivieren Sie dieses Attribut, um ein weiteres Gadget an der linken Seite des Feldes zu haben, das einen Filerequester öffnet, um den Inhalt zu verändern. Dies hat natürlich nur Sinn, wenn das Kommando Dateinamen als Argumente will.

Geänderte Attribute:

Struktur/Listview: nicht anwendbar.

Masken-Fenster/Ausrichtung: nicht anwendbar.

Masken-Fenster/Höhe: immer 1.

Vordefinierte Werte: nicht anwendbar.

Suchäquivalent:

entspricht dem Feldinhalt.

Umwandlung in ein Extern-Feld

Alle Feldtypen können in Extern-Felder ohne Verlust von Daten umgewandelt werden. Sie müssen jedoch ein Programm angeben, das diese Daten benutzen kann.
Zusätzliche Bemerkungen:

Boolean	-	TRUE "Gesetzt" wird TRUE (1), sonst FALSE (0)
Cycle	-	Label (Labelnummer) umgewandelt

Bemerkungen:

Die Programme werden mit dem AmigaDOS-Funktion System() aufgerufen. Für Aus- oder Eingabeoperationen wird ein Consolen-Fenster geöffnet.

9.12 Der Datatypes-Feldtyp

Ein Datatypes-Feld ist Extern-Feldern ähnlich. Der Unterschied ist, daß die Datatypes-Library benutzt wird (und somit kann man dieses Feld erst ab OS 3.0 benutzen) und daß die Daten direkt in der Maske angezeigt werden. Durch die Datatypes-Library ist dieses Feld auch universell einsetzbar und frei erweiterbar. Ein "Popup"-Gadget an der unteren linken Seite des Feldes macht es möglich, den Inhalt des Feldes mit einem Filerequester zu verändern. Wenn ein Fehler in Zusammenhang mit dem Feld auftritt, wird dieser Fehler im Feld angezeigt.

Neue Attribute:

Struktur/Maximale Zeichen: bestimmt die maximale Länge des Dateinamens. Dieses Attribut wirkt sich direkt auf die Dateigröße aus.

Masken-Fenster/Optionen/Dateinamen anzeigen: Wenn diese Option aktiv ist, wird der Dateiname am unteren Rand des Feldes angezeigt. Wenn sie sie ausschalten, können Sie den Inhalt des Feldes nicht verändern. Mit dem ARexx-Script RequestDT.frx und einem Button ist es möglich, den Inhalt des Feldes indirekt zu verändern.

Masken-Fenster/Optionen/Scroll-Leisten: Kontrolliert, ob Scroller zur Verfügung gestellt werden sollen, mit denen der Inhalt des Feldes verschoben werden kann. Ohne Scroller kann immer nur die linke obere Ecke der Daten betrachtet werden. (Dies ist nicht ganz wahr. Manche Datentypen unterstützen das verschieben ihres Inhaltes, wenn man in den Bereich der Anzeige klickt und dabei die Maus in die

Richtung des verborgenen Inhalts zieht. Ein Beispiel ist der Bild Datentyp)

Masken-Fenster/Optionen/Speichern Gadget: Wenn Sie diese Option aktivieren, werden Sie einen zweiten Schalter unter dem Datatypes-Feld kriegen, der mit einem S markiert sein wird. Wenn Sie diesen anwählen, wird ein Filerequester erscheinen, der Sie auffordert, eine Datei auszuwählen, in dem die gerade angezeigten Daten im IFF-Format gespeichert werden.

Masken-Fenster/Optionen/Rahmen: Wenn diese Option aktiv ist, wird Fiasco einen Rahmen um das Feld zeichnen. Deaktivieren sie diese Option nicht zu oft, da es dann keine sichtbaren Elemente gibt, die das Feld abgrenzen.

Masken-Fenster/Optionen/Laden Verzögern: Wenn Sie diese Option aktivieren, wird die Datei nicht sofort geladen, wenn der Record aktiviert wurden. Stattdessen wird der Text Deferred in dem Feld angezeigt werden. Nur wenn Sie das String-Gadget aktivieren und Return drücken, werden die Daten geladen und angezeigt werden.

Masken-Fenster/Optionen/Bilder/Skalierung: Hier können Sie die Skalierung von Bildern in diesem Feld kontrollieren. Wenn der Inhalt des Feldes kein Bild ist, hat dieses Attribut keinen Effekt. Sie können auswählen, ob überhaupt keine Skalierung vorgenommen werden soll, oder nur wenn das Bild größer als das Feld ist oder immer skaliert werden soll.

Masken-Fenster/Optionen/Bilder/Größe bei Skalierung:
Wählen Sie hier die Art der Skalierung. Volle Größe wird das Bild genau in das Feld einpassen und kann so die Proportionen des Bildes verändern. Proportional klein benutzt eine Größe, die das Feld in eine Richtung ausfüllt und vollständig in das Feld paßt. Weiterhin werden die Proportionen des Bildes beibehalten. Proportional groß benutzt die minimale Größe, die das Feld völlig ausfüllt, während auch hier die Proportionen beibehalten werden.

Masken-Fenster/Optionen/Texte/Wort-Umbruch: Aktivieren Sie diese Option, damit das Datatypes-System bei Texten automatisch Wortumbrüche passend zur Feldgröße vornimmt.

Masken-Fenster/Optionen/Verschiedenes/Sofort Spielen:
Wählen Sie diese Option an, um das Spielen der Daten sofort nach dem aktivieren des Records zu starten. Wenn diese Option aktiv ist,

darf Laden Verzögern nicht aktiv sein. Dieses Attribut hat natürtlich keine Auswirkungen, wenn der Datentyp ein Abspielen unterstützt[1]. Zur Zeit unterstützen der Animations- und der Sound-Datentyp Abspielen.

Masken-Fenster/Optionen/Verschiedenes/Inhalt zentrieren:
Zentriert das Datatypes-Objekt im Feld.

Geänderte Attribute:

Struktur/Listview: nicht anwendbar.

Masken-Fenster/Ausrichtung: nicht anwendbar.

Vordefinierte Werte: nicht anwendbar.

Suchäquivalent:

Entspricht dem Dateinamen; Inhaltsbezogene Suche ist nicht möglich.

Umwandlung in ein Datatypes-Feld:

Alle Feldtypen können in Datatypes-Felder ohne Verlust von Daten umgewandelt werden. Sie das Datatypes-System benötigt jedoch korrekte Dateinamen.
Zusätzliche Bemerkungen:

Boolean	-	TRUE "Gesetzt" wird TRUE(1), sonst FALSE(0)
Cycle	-	Label (Labelnummer) umgewandelt

Bemerkungen:

Der AmigaGuide- und der Animations-Datentyp scheinen mit relativ kleinen Feldern nicht zurechtzukommen.

Der AmigaGuide-Datentyp hinterläßt manchmal nach dem Scrollen grafischen Müll.

HAM- und EHB-Bilder können nicht in Datatypes-Feldern dargestellt werden.

Weil die Daten bei jedem Wechsel des Records neu geladen werden müssen, kann sich dieser Vorgang verlangsamen. Um dies zu vermeiden, benutzen Sie Laden Verzögern!

[1] Sofort Spielen ist mit dem Senden einer STM_PLAY Methode implementiert. Dieses Attribut benutzt nicht das in Amiga OS 3.1 neue Tag DTA_Immediate. Daher funktioniert dieses Attribut auch mit 3.0.

9.13 Der Var String-Feldtyp

Var String-Felder nehmen Strings mit variabler Länge auf. String-Felder haben im Kontrast dazu eine maximale Länge. Var String-Felder können mehrere Zeilen enthalten.

Neue Attribute:

Masken-Fenster/Scroll-Balken: Erzeugt an der rechten Seite des Feldes einen Scroll-Balken. Hiermit können Sie den Inhalt des Feldes scrollen.

Geänderte Attribute:

Struktur/Listview: nicht anwendbar.

Masken-Fenster/Ausrichtung: nicht unterstützt.

Vordefinierte Werte: nicht anwendbar.

Such-Äquivalent:

Entspricht dem Feldinhalt.

Unterstützte Suchmuster:

? = Ein unbekanntes Zeichen.
#? = Keine oder mehr unbekannte Zeichen.

Umwandlung in ein Var String-Feld:

Jedes Feld kann ohne Datenverlust in ein Var String-Feld umgewandelt werden. Alternative Formate, sofern unterstützt, sind in Klammern angegeben. Zusätzliche Bemerkungen:

Boolean	-	"Checked" wird TRUE (1), sonst FALSE (0)
Cycle	-	Label (Label-Nummer) wird umgewandelt
Slider	-	Wert wird umgewandelt

Bemerkungen:

Var String-Felder sind mit dem textfield.gadget von Mark Thomas implementiert. Var String-Felder haben Ausschneide- und Einfüge-Funktionen: Mit der Maus können Sie Textteile markieren. Dieser Text kann mit *A* X

ausgeschnitten und mit *A* C kopiert werden. Text aus dem Clipboard kann an der aktiven Cursor-Position mit *A* V eingefügt werden. Diese Tastatur-Kombinationen sind nur aktiv, wenn das Feld aktiv ist, also der Cursor sichtbar ist oder Text markiert ist (dies funktioniert auch mit nur lesbaren Feldern). Ansonsten sind diese Kombinationen anderen Fiasco-Funktionen zugeordnet.

Relativ große Var String-Felder können das Springen zwischen Records verlangsamen.

Beginnend mit Fiasco 2.1 kann der Inhalt von Var String-Feldern als einzelne Zeile im Listen-Fenster angezeigt werden. Zeilenumbrüche im Inhalt werden für das Listen-Fenster zu Leerzeichen umgewandelt. Beachten Sie jedoch, daß Var String-Felder mit einem relativ langen Inhalt den Grafik-Aufbau des Listen-Fensters erheblich verlangsamen können. Somit sollten Sie solche Felder im Listen-Fenster verbergen.

Seit der Einführung der Var String-Felder in Fiasco 2.00 konnten bis Fiasco 2.02 (inklusive) keine Var String-Felder im Listen-Fenster angezeigt werden. Wenn Sie Datenbanken laden, die mit diesen Versionen erzeugt wurden mit Fiasco 2.1 oder höher laden, werden Var String-Felder als Verborgen im Listen-Fenster markiert. Um diese wieder anzuzeigen, schalten Sie das entsprechende Attribut einfach aus.

9.14 Der Text-Feldtyp

Der Text-Feldtyp ist eigentlich gar kein echtes Feld, er dient lediglich zur Beschriftung der Maske.

Dieser Feldtyp unterstützt keine Standard-Attribute.

Unterstützte Attribute:

Text: ist der Text, der in die Maske geschrieben wird.

Stift: gibt die Farbe an, in der der Text geschrieben werden soll. Es sind möglich: Normal (normalerweise Schwarz), Hervorgehoben (normalerweise Weiß). Die Farben lassen sich über den Palette-Preferences-Editor be- einflussen.

Fett: Ist dieses Attribut aktiv, wird der Text Fett geschrieben. Ein Unterstrich ("_") vor einem Zeichen bewirkt, daß dieses Zeichen unterstichen gezeichnet wird. So können Sie Tastatur-Abkürzungen eines Feldes darstellen.

Kursiv: Läßt den Text kursiv schreiben.

Unterstrichen: Läßt den Text unterstreichen.

Suchäquivalent:

Nach einem Text-Feld kann aufgrund Inhaltfreiheit nicht gesucht werden.

Umwandlung in ein Text-Feld:

Felder können nicht in Text-Felder umgewandelt werden.

9.15 Der Button-Feldtyp

Buttons sind keine echten Felder, sie dienen nur dazu, einen Schalter in die Maske zu setzen, mit dem eine benutzerdefinierbare Aktion gestartet werden kann.

Dieser Feldtyp unterstützt nur die Standard-Attribute Breite und Tastatur-Kombination.

Unterstützte Attribute:

Text: wird im Button angezeigt.

Typ: Wählen Sie hier aus, ob der Button ein CLI- oder ein ARexx-Programm ausführen soll. CLI-Programme können normale Programme, Kommandos oder Scripts (mit dem "s"-Attribut) sein. ARexx-Programme müssen ARexx Scripts sein.

Kommando: Wählen Sie hier das Programm aus, das bei Betätigung des Buttons ausgeführt werden soll.

Stack: Hier können Sie die Stack-Größe für das Programm angeben. Der Standard ist 4096. Wenn Sie zu wenig Stack für ein Programm angeben, wird es abstürzen.

Ausgabe-Fenster: erlaubt es Ihnen, einen I/O-Strom für das Programm zu definieren. Dies kann ein Konsolen-Fenster (CON:), der Drucker (PRT:), eine einfache Datei, oder, wenn Sie keinerlei Ausgabe wünschen NIL: sein. Seit Fiasco 2.1 ist dies auch für ARexx-Scripts implementiert.

Suchäquivalent:

Nach einen Button-Feld kann nicht gesucht werden.

Umwandlung in ein Button-Feld:

Felder können nicht in Button-Felder umgewandelt werden.

9.16 Der Leisten-Feldtyp

Leisten sind keine echten Felder, sie dienen nur dazu, eine sichtbare Abgrenzung in die Maske zu setzen.

Dieser Feldtyp unterstützt keine Standard-Attribute.

Unterstützte Attribute:

Breite/Höhe: Die Höhe oder die Breite der Leiste; abhängig von Ausrichtung.

Ausrichtung: bestimmt, ob die Leiste horizontal oder vertikal in die Maske gesetzt wird.

Suchäquivalent:

Nach einen Leisten-Feld kann nicht gesucht werden.

Umwandlung in ein Button-Feld:

Felder können nicht in Leisten-Felder umgewandelt werden.

Besonderes Verhalten in Gruppen:

Leisten-Felder sind gesellige Felder; mehrere Leisten in einer Gruppe, die sich gegeseitig berühren (natürlich nur aus technischen Gründen) verschmelzen optisch. Diese Eingenschaft erlaubt es, Kästen zu erstellen, indem man vier Leisten ein einem Rechteck erstellt.

9.17 Der Listview-Feldtyp

Der Listview-Feldtyp ist kein echter Feldtyp, sondern eine Abänderung für Feldtypen.

Dieser Feldtyp wird z.Zt. von fünf Feldtypen unterstützt:

- String
- Integer
- Float

- Datum
- Zeit

Damit ein Feld ein Listview-Feld wird, müssen Sie die Listview Option beim Erstellen eines Feldes im Feld-Requester aktivieren. Der Feld Ändern-Requester, der mit Feld/Feld Ädern geöffnet werden kann, erlaubt es *nicht*, diese Option zu verändern, da die Änderung davon eine Umwandlung des Feldes benötigen würde. Die Feld Konvertieren Funktion erledigt diese Aufgabe.

Listview-Felder können eine unbestimmte Anzahl von Einträgen aufnehmen. Diese Einträge entsprechen den einfachen Feldern. Ein Eintrag kann durch Anklicken im Listview aktiviert werden. Der aktive Eintrag kann im Gadget unter dem Listenteil geändert werden. Um einen Eintrag hinzuzufügen, müssen Sie auf das '+' Gadget klicken. Das '–' Gadget löscht den aktiven Eintrag.

Geänderte Attribute:

Masken-Fenster/Höhe: Das Höhe-Attribut wird verfügbar, wenn die Listview-Option aktiv ist.

Listen-Fenster: Wenn die Listview-Option aktiv ist, werden alle Listen-Fenster Attribute gesperrt, da Listviews nicht im Listen-Fenster dargestellt werden können.

Startwert: Dies ist der Wert, der in neu hinzugefüge Einträge eingesetzt wird. Listviews in neuen Records sind immer leer.

Predefined Values: Nicht anwendbar.

Suchäquivalent:

Sie können nicht nach Listview-Feldern suchen.

Umwandlung in ein Listview-Feld:

Wenn Sie ein Listview-Feld in ein Listview-Feld eines anderen Typs umwandeln, wird jeder Eintrag des Listviews so umgewandelt, wie es in der Beschreibung für den neuen Feldtyp angegeben ist.

Wenn Sie ein Nicht-Listview-Feld in ein Listview-Feld umwandeln oder umgekehrt, wird das Zeichen '| ' benutzt, um die Einträge voneinander zu trennen. Da dieses Zeichen in numerischen Feldern nicht verfügbar sind, macht diese Umwandling nicht viel Sinn (die erste Zahl wird allerdings umgewandelt).

Besonderes Verhalten in Gruppen:

Listviews, die gruppiert sind (siehe Abschnitt 5.2), werden immer die Einträge mit der gleichen Nummer aktivieren. Wenn Sie also in einem gruppierten Listview den zweiten Eintrag aktivieren, werden alle Listviews in der Gruppe auch den zweiten Eintrag aktivieren. Weiterhin werden auch die Neu und Lösch-Funktionen von Listviews auf alle Listviews in einer Gruppe angewendet.

Wenn Sie gruppierte Listviews in der Maske nebeneinandersetzen, die Listviews dieselbe Höhe und den selben Zustand des Nur Lesbar Attributs haben, werden diese Listviews ihre optische Erscheinung teilen.

Bemerkungen:

Wenn Sie ein Listview-Feld zu schmal machen, wird Fiasco die +/– Gadgets entfernen, um mehr Platz zu haben. Um die Funktion diese Gadgets wiederzubekommen, können Sie entweder das Feld mit einem anderen Listview-Feld gruppieren oder die ARexx-Kommandos AddLVFieldEntry und DeleteLVFieldEntry benutzen.

Kapitel 10

Benutzeroberfläche von Fiasco

Das einzige, womit Fiasco sich normalerweise nach dem Start präsentiert, ist ein leeres Fenster. Durch Pull-Down-Menüs kann man nun dem Programm seine Wünsche mitteilen. Für diejenigen, die Menüs nicht so mögen, können über den Menüpunkt Kontrolle/Service-Fenster ein weiteres Fenster öffnen, mit dem sich die meisten Aktionen auch ohne Menü ausführen lassen. Die dritte Möglichkeit, mit Fiasco zu kommunizieren, sind Tastatur-Shortcuts.

Fiasco unterstützt Menü-Help. Mit Menü-Help können Sie während Sie ein Menü aktiviert haben durch Drücken der Help-Taste eine Beschreibung der Funktion in einem separaten AmigaGuide-Fenster abrufen. (Dieses Feature benötigt die amigaguide.library, die Teil des OS seit Release 3.0 ist. Wenn Sie noch 2.0 oder 2.1 benutzen, können Sie die Library von der PD kriegen).

Alle Requester, die Fiasco benutzt, haben eine grundlegende Struktur. Am unteren Rand befinden sich die Gadgets, mit denen man den Requester bestätigt. Meistens sind dies Ok, was die Einstellungen im Requester ausführt und Abbrechen bzw. Cancel, was die neu gemachten Einstellungen im Requester verwirft. Ok kann mit der Return-Taste abgekürzt werden und Cancel kann mit Esc abgekürzt werden. Das Closegadget oben links macht das selbe wie Cancel oder Esc.

10.1 Das Masken-Fenster

Die Arbeitsweise des Masken-Fensters ist in Fiascos Editier-Modi verschieden.

Masken-Modus

Die Hauptkontrolle im Masken-Modus ist der Cursor. Der Cursor kann einmal mit den Cursor-Tasten bewegt werden. Wenn eine Cursor-Taste ohne eine andere Taste gedrückt wird, wird der Cursor um eine Einheit in der Richtung der Cursor-Taste bewegt. Wenn Sie die Cursor-Taste mit Shift drücken, wird der Cursor an das entsprechende Ende des Fenster bewegt. Wenn Sie also Shift und Cursor Rechts drücken, wird der Cursor am rechten Rand des Fensters positioniert. Wenn Sie eine Cursor-Taste mit Ctrl kombinieren, wird der Cursor zum Extremum der Maske in der Richtung bewegt. Die Cursor-Rechts und -Runter-Tasten bewegen den Cursor dann zu der Position des letzten Feldes in dieser Richtung, die Cursor-Links und -Hoch-Tasten bewegen den Cursor zu der X bzw. Y Position 0.

Wenn Sie ein Feld mit dem Cursor erreichen, wird das Feld aktiviert.

Wenn Sie mit der Maus einen Platz im Masken-Fenster anklicken, wird der Cursor an diese Stelle bewegt. Wenn Sie auf ein Feld klicken, wird dies aktiviert. Sie können mehrere Felder aktivieren, indem Sie Shift gedrückt halten, während Sie die Felder aktivieren.

Die aktivierten Felder können mit der Maus verschoben werden. Halten Sie die linke Maustaste gedrückt, während Sie die Maus bewegen. Wenn Sie die Maustaste loslassen, werden die Felder dort fallengelassen, wo Sie sie hingezogen haben. Wenn Sie die rechte Maustaste während des Ziehens drücken, wird das Ziehen abgebrochen und die Felder werden an den alten Ort zurückgelegt.

Wenn Sie auf ein Feld zweimal kurz hintereinander klicken, wird der Feld-Requester für dieses Feld geöffnet.

Record-Modus

Im Record-Modus verhalten sich alle Felder in der Maske wie normale Gadgets in normalen Anwendungen. Felder können auch eine Tastatur-Abkürzung haben. Die Feld-Dokumentationen enthalten mehr Informationen darüber.

Wenn Sie in einem String-, Integer-, Float-, Datums-, Zeit-, Extern-, Datatypes- oder Var String-Feld die Tabulator Taste drücken, wird das nächste Feld aktiviert.

Wenn Sie die Enter Taste zum Weiterspringen vorziehen, wie es unter Fiasco 1.x war, müssen Sie die Benuteroberflächen Einstellungen (siehe Abschnitt 10.6.24) ändern.

10.1.1 Maskendehnung

Normalerweise sitzen die Felder in einer Fiasco-Maske sich dicht auf der Pelle. Dies ist a) nicht schön anzusehen, b) haben eigentliche alle anderen "normalen" GUIs ein paar Pixel zwischen den Gadgets Platz. Man könnte zwar jeweils eine Zeile zwischen den Feldern frei lassen, dies würde jedoch bald zu Platzproblemen führen. Deswegen gibt Fiasco die Möglichkeit, zwischen den einzelnen Zeilen und Spalten pixelweise Raum zu lassen.

Diese Werte können im Options-Requester in den Gadgets Maskendehnung X / Y angegeben werden.

Ein Problem bei dieser Methode die Maske zu "dehnen" ist, daß auch die Felder größer werden als in den Feld-Requestern angegeben wurde. Dies fällt insbesondere in der horizontalen Richtung auf, da die meisten Fiasco-Felder sich nur in diese Richtung ausdehnen. Das größte Problem ergibt sich bei Text-Feldern, die normalerweise nur so breit sind, wie der Text es erfordert. Nun sind sie aber breiter und der Text muß zentriert werden.

Um diese Probleme zu vermeiden, empfehle ich, daß als X-Wert Null angegeben wird und in dieser Richtung weiterhin einzelne Spalten als Trennung benutzt werden. In Y-Richtung liefert 4 die besten Ergebnisse.

10.2 Das Listen-Fenster

Das Listen-Fenster kann mit Kontrolle/Listen-Fenster geöffnet werden. Der obere Teil des Listen-Fensters ist der Kopf der Liste. Er zeigt die IDs der Felder an, die durch die Spalten repräsentiert werden. Die Zeilen der Liste repräsentieren die Records; der aktive Record wird mit einer starken Unterlegung markiert. Markierte Records werden mit einer dünnen Unterlegung angezeigt. Fiascos Listen-Fenster zeigt nur die Records an, die im aktiven Index sind. Der Nachfolger eines Records wird unter diesem angezeigt. Wenn Sie auf eine Record-Zeile klicken, wird dieser Record aktiviert. Zum Ändern eines Records müssen Sie jedoch die Maske benutzen.

Die Scroll-Balken des Fensters können zum scrollen des Inhaltes benutzt werden. Der vertikale Scroll-Balken scrollt durch alle Records, der horizontale scrollt durch alle Felder in der Liste.

Das Layout einer Liste, was Position und Breite von Spalten bedeutet, wird automatisch vorgenommen. Sie können das Layout jedoch auch direkt im Listen-Fenster und den Feld-Requestern im Masken-Modus kontrollieren.

Um die Position einer Spalte zu ändern, müssen Sie die Mitte des Spalten-Kopfes anklicken. Lassen Sie die Maustaste nicht los. Zwei Linien werden an der aktuellen Position der Spalte erscheinen. Nun können Sie

die Spalte über eine andere Spalte ziehen. Wenn Sie die Maustaste loslassen, setzt Fiasco die Spalte so nah wie möglich an den neuen Platz. Spalten, die von der Spalte überlappt werden, werden nach rechts verschoben. Der alte Platz der Spalte wird durch Verschieben von Spalten rechts davon nach links aufgefüllt.

Die Breite einer Spalte kann auch mit der Maus geändert werden. Wenn Sie den rechten Rand eines Spalten-Kopfes anklicken und die Maustaste nicht loslassen, wird eine Linie erscheinen. Nun können Sie diese mit der Maus verschieben. Der Ort, wo Sie die Maustaste loslassen, wird der neue rechte Rand der Spalte werden. Die Positionen der Felder rechts von der Spalte werden angepasst. Sie können auch den Feld-Requester des Feldes benuten, um die Breite der Spalte zu verändern. Das Attribut Listen-Fenster/Breite (siehe Abschnitt 9.1) dient diesem Zweck.

Sie können auch kontrollieren, ob ein Feld als Spalte in der Liste erscheint oder nicht. Normalerweise wird ein Feld automatisch in die Liste eingefügt, wenn es erstellt wird.[1] Um eine Spalte zu verbergen, aktivieren Sie den Kopf der Liste, indem Sie ihn anklicken, und wählen den Menüpunkt Liste/Spalte verbergen. Sie können eine Spalte auch verbergen, indem Sie das Listen-Fenster/Verborgen (siehe Abschnitt 9.1)-Attribut im Feld-Requester aktivieren. Um es wieder hervorzuholen, können Sie den Menüpunkt Liste/Spalte anzeigen (siehe Abschnitt 10.4.52) benutzen oder das Listen-Fenster/Verborgen-Attribut deaktivieren.

Der Menüpunkt Liste/Alle Spalten anzeigen macht alle verborgenen Spalten wieder sichtbar.

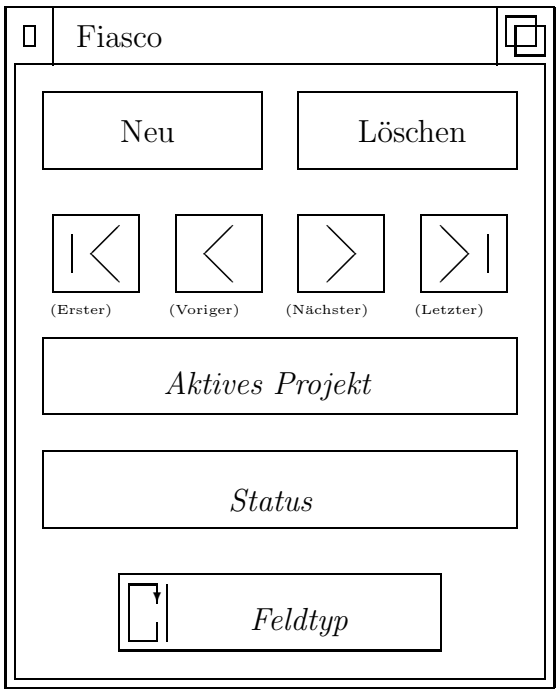
Liste/Liste neu berechnen berechnet die Positionen und Dimensionen aller Spalten neu. Dieser Menüpunkt ist vergleichbar mit Inhalt Aufräumen der Workbench. Spalten, die verborgen sind, werden verborgen gehalten.

10.3 Das Service-Fenster

Das Service-Fenster läßt sich über den Menüpunkt Kontrolle/Service-Fenster öffnen bzw. schließen. Wenn Sie wollen, daß Fiasco bei jedem Programmstart das Service-Fenster automatisch öffnet, brauchen Sie nur die Option Service-Fenster/Beim Start Öffnen in den Benutzeroberflächen Einstellungen (siehe Abschnitt 10.6.24) aktivieren. Wenn Service-Fenster/Fester Postion im gleichen Requester inaktiv ist, sucht Fiasco beim Öffnen immer nach einem freien Platz auf dem Bildschirm, um das Fenster zu öffnen. Sonst wird die Position des Fensters beim Speichern der Einstellungen benutzt.

Das Service-Fenster sieht ungefähr so aus:

[1] Wenn es überhaupt in der Liste dargestellt werden kann.



Die Schalter des Service-Fensters sind im Folgenden beschrieben.

10.3.1 Neu

Falls sich das aktive Projekt im Record-Modus befindet, wird ein neuer Record erstellt. Ist der Modus der Masken-Modus, wird ein neues Feld des eingestellten Feld-Typs an der aktuellen Cursorposition erstellt.

Äquivalent zu dem Menüpunkt:
Records/Hinzufügen im Record-Modus
bzw.
Felder/Feld hinzufügen... im Masken-Modus

10.3.2 Löschen

Falls das aktuelle Projekt im Record-Modus ist, wird der aktuelle Record gelöscht. Ist der Masken-Modus aktiv, wird das aktive Feld gelöscht (falls keins aktiv ist, wird auch keins gelöscht). *Achtung:* Das Löschen geschieht normalerweise ohne Sicherheitsabfrage!

Äquivalent zu dem Menüpunkt:
Records/Löschen im Record-Modus

bzw.
Felder/Feld Entfernen im Masken-Modus

10.3.3 Erster

Falls das aktuelle Projekt im Record-Modus ist, wird Record Nr.1 aktiviert.

Äquivalent zu dem Menüpunkt:
Record/Erster

10.3.4 Voriger

Falls das aktuelle Projekt im Record-Modus ist, wird der dem aktiven Record vorausgehende Record aktiviert.

Äquivalent zu dem Menüpunkt:
Record/Voriger

10.3.5 Nächster

Falls das aktuelle Projekt im Record-Modus ist, wird der dem aktiven Record folgende Record aktiviert.

Äquivalent zu dem Menüpunkt:
Record/Nächster

10.3.6 Letzter

Falls das aktuelle Projekt im Record-Modus ist, wird der letzte Record aktiviert.

Äquivalent zu dem Menüpunkt:
Record/Letzter

10.3.7 Aktives Projekt

Hier wird der Name des aktuellen Projektes angezeigt. Achtung: Sollten sich zwei Projekte nur im Pfad, nicht im Namen voneinander unterscheiden, wird beidesmal dasselbe angezeigt.

Ein anderes Projekt aktiviert man, indem man das jeweilige Fenster aktiviert.

10.3.8 Status

Hier werden Statusinformationen zum aktiven Projekt angezeigt.

Im Record-Modus ist dies:
Nummer des aktiven Records/Anzahl aller Records
Die Zahlen können variieren, falls ein Filter aktiv ist.
Im Mask-Modus wird folgendes angezeigt:
X: *X-Position des Cursors*, Y: *Y-Position des Cursors*

10.3.9 Feldtyp

Hier stellt man ein, was für ein Feldtyp beim nächsten Erzeugen eines Feldes erzeugt wird. Äquivalent mit Felder/Feldtyp.

10.4 Menüs

Die wichtigste Schnittstelle zu den Funktionen von Fiasco sind die Pull-Down-Menüs der Projekt-Fenster von Fiasco. Die Funktionen aller Menüpunkte (von Links nach Rechts) sind im Anschluß beschrieben.

10.4.1 Projekt/Neu (Project/New)

Shortcut: *A* N

Erzeugt ein neues Projekt mit dazugehörigem Masken-Fenster. Es enthält dann weder Records noch Felder. Nun können Sie entweder eine neue Datenbank erstellen, oder mit Öffnen eine bereits existierende laden.
Querverweise: Projekt/Öffnen, Projekt/Neu Öffnen

10.4.2 Projekt/Leeren (Project/Erase)

Shortcut: *A* Z

Löscht jegliche Daten aus dem aktuellen Projekt, das Projekt wird hinterher in einem Zustand wie nach dem Programmstart oder Projekt/Neu sein. Falls die Daten des Projektes seit dem letzten Abspeichern verändert wurden, wird ein vorher ein Sicherheitsrequester angezeigt.

10.4.3 Projekt/Öffnen... (Project/Open)

Shortcut: *A* O

Öffnet einen Datei-Requester, mit dem man ein Fiasco-Projekt zum Laden auswählen kann. Die Daten werden im aktiven Fenster angezeigt. Ein

eventuell bereits in diesem Fenster geladenes Projekt wird aus dem Speicher entfernt. Um versehentlichen Datenverlust zu vermeiden, werden Sie gefragt, ob sie wirklich laden wollen, falls das Projekt seit dem letzten Speichern verändert wurde.

10.4.4 Projekt/Neu Öffnen... (Project/Open new)

Shortcut: *A* L

Öffnet einen Datei-Requester und läd die ausgewählte Datenbank in ein automatisch erzeugtes neues Projekt-Fenster.

10.4.5 Projekt/Speichern (Project/Save)

Shortcut: *A* S

Mit Speichern werden die Daten des aktuellen Projektes unter dem selben Namen gespeichert, mit dem es auch geladen wurde. Falls Sie das Projekt unter einem anderen Namen speichern möchten, wählen sie Speichern Als (siehe Abschnitt 10.4.6).

10.4.6 Projekt/Speichern als... (Project/Save As)

Shortcut: *A* A

Hier können Sie das aktuelle Projekt unter einem anderen Namen als dem Namen, unter dem Sie es geladen haben, speichern. Der Name wird mit einem Requester der ASL-Library erfragt und nach dem Speichern beibehalten.

10.4.7 Projekt/Importieren... (Project/Import)

Shortcut: *A* I

Öffnet den Import-Requester (siehe Abschnitt 10.6.1), die GUI-Schnittstelle der Import-Funktion von Fiasco. Sie können die Import-Funktion benutzen, um Daten aus fremden Datenbank-Programmen in Fiasco einzulesen.

10.4.8 Projekt/Exportieren... (Project/Export)

Shortcut: *A* E

Öffnet den Export-Requester (siehe Abschnitt 10.6.2), die GUI-Schnittstelle zur Export-Funktion von Fiasco. Die Export-Funktion dient zum Speichern von Fiasco-Daten in einem Format, das von anderen Datenbank-Programmen gelesen werden kann.

10.4.9 Projekt/Drucken... (Project/Print)

Shortcut: *A* P

Dieser Menüpunkt öffnet das Druckfenster, die Hauptschnittstelle zur Druckfunktion (siehe Abschnitt 7) von Fiasco. Sie können hier ein Layout erstellen und dieses ausdrucken.

10.4.10 Projekt/Verbergen (Project/Hide)

Shortcut: *A* H

Schließt alle Fenster des aktiven Projektes. Die Projekt-Daten werden jedoch nicht freigegeben. Wenn das Fenster das letzte offene Fenster war, wird Fiasco seinen eigenen Bildschirm schließen oder den öffentlichen Bildschirm freigeben. Ein Icon wird dann auf der Workbench angezeigt.

Um ein Projekt wieder zu öffnen, können Sie Projekt/Sichtbar machen benutzen, wenn ein anderes Fenster noch offen ist, oder auf das Fiasco-Icon auf der Workbench doppelklicken. In beiden Fällen wird der Project Anzeigen-Requester geöffnet, der es Ihnen erlaubt, eines der verborgenen Projekte zu öffnen.

Eine andere Möglichkeit, Zugriff auf Fiasco zu bekommen, wenn alle Fenster zu sind, ist Fiasco nochmal zu starten. Fiasco wird dann ein leeres Projekt-Fenster öffnen, von dem Sie Zugriff auf den Projekt/Sichtbar machen Menüpunkt haben. Dies ist beispielsweise dann nützlich wenn die Workbench nicht läuft und Fiasco so kein Icon erzeugen kann.

10.4.11 Projekt/Sichtbar machen... (Project/Reveal)

Shortcut: *A* ^

Öffnet den Projekt Anzeigen-Requester (siehe Abschnitt 10.6.3), mit dem Sie ein mit Projekt/Verbergen verborgenes Projekt wieder öffnen können. Wenn Sie alle Fiasco-Fenster schließen, haben Sie keinen Zugriff auf diesen Menüpunkt. Deshalb erzeugt Fiasco ein Icon auf der Workbench. Wenn Sie darauf doppelklicken, wird dieselbe Funktion aktiviert.

Querverweise: Projekt/Verbergen

10.4.12 Projekt/Über Fiasco... (Project/About)

Shortcut: *A* ?

Hiermit wird ein kleiner Requester angezeigt, der Auskunft über Version, Copyright und ein paar Systeminterna gibt.

10.4.13 Projekt/Beenden (Project/Quit)

Shortcut: *A* Q

Dieser Menüpunkt schließt das aktive Projekt. Falls es in der Zwischenzeit verändert und noch nicht gespeichert wurde, werden Sie über einen Requester gefragt, ob sie wirklich dieses Projekt schließen möchten. Falls dies das letzte Projekt war, das Fiasco offen hatte, wird Fiasco beendet.

10.4.14 Datenbank/Optionen... (Database/Options)

Shortcut: *A* $

Über diesen Menüpunkt können Sie den Options-Requester öffnen, in dem sie projektspezifische Optionen einstellen können. Dies sind:

- Maskendehnung (siehe Abschnitt 10.1.1)
- Name des Autors sowie Anmerkungen
- Projekt-Fenster
- Diskettenzugriffszeit für das Lesen von Records
- RAM-Auslastung durch die Datenbank

Vor Fiasco 2.1 befand sich dieser Menüpunkt im Menü Projekt.

10.4.15 Datenbank/Statistik... (Database/Statistic)

kein Shortcut

Zeigt einige Informationen über RAM-Benutztung, etc. für das aktive Projekt im Statistik-Requester (siehe Abschnitt 10.6.5) an.

Vor Fiasco 2.1 befand sich dieser Menüpunkt im Menü Projekt.

10.4.16 Datenbank/Indizes... (Database/Indices)

Shortcut: *A* *

Dieser Menüpunkt öffnet den Indizes Requester (siehe Abschnitt 10.6.6), der zum Auswählen, Erzeugen oder Löschen eines Index benutzt werden kann.

Vor Fiasco 2.1 befand sich dieser Menüpunkt im Menü Projekt.

10.4.17 Datenbank/Voriger aktiver Index (Database/Prev)

Kein Shortcut

Geht einen Schritt in der Index-History (siehe Abschnitt 5.3.1) zurück. Das bedeutet, daß dieser Menüpunkt den Index wieder aktiviert, der aktiv war, bevor der augenblicklich aktive Index aktiviert wurde. Um diesen Index wieder zu aktivieren, müssen Sie mit dem Menü-Punkt Nächster aktiver Index wieder einen Schritt vorwärts machen.

10.4.18 Datenbank/Nächster aktiver Index

Kein Shortcut

Geht einen Schritt in der Index-History (siehe Abschnitt 5.3.1) wieder vorwärts, nachdem mit Voriger aktiver Index einer rückwärts gemacht wurde.

10.4.19 Datenbank/Reorganisieren... (Database/Reorganize)

Kein Shortcut

Dieser Menüpunkt ist ähnlich zu Projekt/Speichern. Der Unterschied liegt darin, daß Reorganisieren die ganze Datenbank-Datei nur schreibt und alle

Records endgültig löscht, die in keinem Index der Datenbank mehr benutzt werden. Records,indexRecords!Gelöschte die mit der Löschen-Funktion gelöscht werden, werden nur aus dem Index entfernt. Die Daten werden bis zur nächsten Reorganisation in der Datenbank bleiben.

Bevor Fiasco die Reorganisation beginnt, werden Sie noch in einem Requester darauf hingewiesen, wieviele Records gelöscht werden.

Wenn es Relationen gibt, die nach einem Schlüssel suchen, der in diesen Records definiert ist, werden diese nach einer Reorganisation den Schlüssel nicht mehr finden.

Vor Fiasco 2.1 befand sich dieser Menüpunkt im Menü Projekt.

10.4.20 Datenbank/Relationen neuladen (Database/Reload Rels)

Kein Shortcut

Hier kann man Fiasco instruieren, alle Relationen im aktuellen Projekt neu zu laden, wie es bereits beim Laden des Projektes geschehen ist. Dies ist insbesondere für diejenigen nützlich, die den Menüpunkt Relationen aktualisieren in den Datenbank-Einstellungen deaktiviert haben, einige Schlüsselworte neu eingegeben bzw. verändert haben und nun die Ergebnisse sehen wollen.

Vor Fiasco 2.1 befand sich dieser Menüpunkt im Menü Projekt.

10.4.21 Datenbank/Funktionen... (Database/Functions)

Kein Shortcut

Öffnet den Funktions-Requester (siehe Abschnitt 10.6.8), der zum Verändern der benutzerdefinierten Funktionen (siehe Abschnitt 11.1.5) der aktiven Datenbank benutzt werden kann.

10.4.22 Datenbank/Konstanten... (Database/Constants)

Kein Shortcut

Öffnet den Konstanten-Requester (siehe Abschnitt 10.6.9), der zum Verändern der benutzerdefinierten Konstanten (siehe Abschnitt 11.1.3) der aktiven Datenbank benutzt werden kann.

10.4.23 Record/Hinzufügen (Record/Add Record)

Shortcut: +

Fügt der Record-Liste des aktuellen Projektes einen neuen Record hinzu. Die einzelnen Felder enthalten nun entweder nichts, oder falls dies im Masken-Modus so eingestellt wurde, einen Startwert. Falls das Listen-Fenster offen ist, erscheint auch dort ein entsprechender Eintrag.

Falls ein Filter aktiv ist, wird der Record automatisch für gültig erklärt, egal was in den einzelnen Feldern steht. Damit neuerstellte Records korrekt in den Filter miteinbezogen werden, müssen Sie den Filter neu aufbauen.

Dieser Menüpunkt ist nur im Record-Modus anwählbar.

Querverweise: Record/Löschen

10.4.24 Record/Duplizieren (Record/Duplicate Record)

Shortcut: *A* 2

Kopiert den aktuellen Record auf das Haar genau in einen neuen Record. Jegliche Startwert-Vorgaben werden ignoriert. Selbst ein Feld, das das Attribut Eindeutiger Schlüssel gesetzt hat, enthält denselben Wert wie der alte Record, wodurch die Schlüssel nicht mehr eindeutig sind.

10.4.25 Record/Löschen (Record/Delete Record)

Shortcut: *A* -

Entfernt den aktiven Record aus dem aktiven Index. Die Daten werden jedoch in der Datenbank-Datei verbleiben. Um diese endgültig zu löschen, dürfen keine anderen Indizes die Daten benutzen. Wenn dies der Fall ist, entfernt Projekt/Reorganisieren (siehe Abschnitt 10.4.19) die Daten aus der Datei.

Bevor Sie die Daten mit Projekt/Reorganisieren löschen, können Sie sie noch wiederherstellen. Erzeugen Sie einen neuen Index, der den besonderen Eintrag ≪Kein Index≫ als Index-Vorbild benutzt. Der neu gebaute Index enthält dann alle Records der Datenbank.

Dieser Menüpunkt ist nur im Record-Modus anwählbar.

Falls die Option Sicherheits-Requester in den Benuteroberflächen-Einstellungen aktiv ist, erscheint zuvor noch eine Sicherheitsabfrage, ob dieser Record wirklich gelöscht werden soll.

Querverweise: Record/Hinzufügen, Record/Alle löschen, Projekt/Reorganisieren

10.4.26 Record/Alle löschen (Record/Delete all Records)

Shortcut: *A* @

Löscht alle Records des aktiven Projektes. Die Maske wird von dieser Funktion nicht angetastet.

Record/Alle Löschen darf nur im Record-Mode angewählt werden.

Bitte beachten Sie: Im Gegensatz zu den Funktionen Record Löschen und Feld Löschen öffnet dieser Punkt keinen Requester, wenn der Menüpunkt Einstellungen/Sicherheits-Requester aktiv ist. Vielmehr öffnet dieser Menüpunkt einen Standard-Warn-Requester, wenn das Projekt verändert wurde.

Querverweise: Record/Löschen

10.4.27 Record/Ausschneiden (Record/Cut Record)

Shortcut: *A* X

Kopiert den aktuellen Record ins Clipboard und entfernt den Record aus dem aktiven Index. Danach kann der Record mit Record/Einfügen (siehe Abschnitt 10.4.29) wieder in das Projekt eingefügt werden.

Diese Funktion kann nur im Record-Modus aufgerufen werden.

Querverweise: Record/Kopieren, Record/Einfügen, Abschnitt Clipboard-Unterstützung von Fiasco

10.4.28 Record/Kopieren (Record/Copy Record)

Shortcut: *A* C

Kopiert den aktuellen Record ins Clipboard. Danach können Sie mit Record/Einfügen den Record wieder in ein Projekt einfügen.

Diese Funktion darf nur im Record-Modus aufgerufen werden.

Querverweise: Record/Ausschneiden, Record/Einfügen, Abschnitt Clipboard-Unterstützung von Fiasco

10.4.29 Record/Einfügen (Record/Paste Record)

Shortcut: *A* P

Erzeugt einen neuen Record und fügt den Inhalt des Clipboards in diesen Record ein. Normalerweise sollten Sie Record/Ausschneiden (siehe Abschnitt 10.4.27) oder Record/Kopieren (siehe Abschnitt 10.4.28) aufrufen, bevor Sie diese Funktion benutzen.

Diese Funktion kann nur im Record-Modus aufgerufen werden.
Querverweise: Record/Ausschneiden, Record/Kopieren, Abschnitt Clipboard-Unterstützung von Fiasco

10.4.30 Record/Voriger (Record/Previous)

Shortcut: Cursor up

Aktiviert den dem aktuellen Record *vorgehenden* Record. Falls der aktuelle Record der erste in der Liste ist, wird die Anzeige "gebeebt".

Die Reihenfolge von Records wird vom aktiven Index bestimmt.

Der Tastatur-Shortcut wurde in Hinblick auf die Darstellung in der Liste gewählt, in der der vorgehende Record über dem aktuellen liegt.

Dieser Menüpunkt ist nur im Record-Modus anwählbar.
Querverweise: Nächster, Erster, Letzter, Gehe zu, Vergl./Rückwärts suchen

10.4.31 Record/Nächster (Record/Next)

Shortcut: Cursor down

Aktiviert den Record nach dem *aktuellen* Record. Falls der aktuelle Record der letzte in der Liste ist, wird die Anzeige "gebeebt".

Die Reihenfolge von Records wird vom aktiven Index bestimmt.

Der Tastatur-Shortcut wurde in Hinblick auf die Darstellung in der Liste gewählt, in der der nächste Record unter dem aktuellen liegt.

Dieser Menüpunkt ist nur im Record-Modus anwählbar.
Querverweise: Voriger, Erster, Letzter, Gehe zu, Vergl./Weitersuchen

10.4.32 Record/Erster (Record/First Record)

Shortcut: Ctrl Cursor up

Aktiviert den *ersten* Record des aktuellen Projektes.

Die Reihenfolge von Records wird vom aktiven Index bestimmt.

Dieser Menüpunkt ist nur im Record-Modus anwählbar.
Querverweise: Nächster, Voriger, Letzter, Gehe zu

10.4.33 Record/Letzter (Record/Last Record)

Shortcut: Ctrl Cursor down

Aktiviert den *letzten* Record des aktuellen Projektes.
Die Reihenfolge von Records wird vom aktiven Index bestimmt.
Dieser Menüpunkt ist nur im Record-Modus anwählbar.
Querverweise: Nächster, Voriger, Erster, Gehe zu

10.4.34 Record/Gehe zu... (Record/Goto)

Shortcut: *A* G

Öffnet den Goto-Requester mit dem man über die Nummer eines Records diesen erreichen kann. Bitte beachten Sie, daß die Nummern sich durch Einfügen oder Löschen von Records verändern können und von Index zu Index unterscheiden.
Dieser Menüpunkt ist nur im Record-Modus anwählbar.
Querverweise: Nächster, Voriger, Erster, Letzter

10.4.35 Record/Markiere Record (Record/Mark Record)

Shortcut: *A* .

Markiert den aktuellen Record. Wenn ein Record markiert ist, wird er in der Liste hervorgehoben angezeigt und das Zeichen "M" wird im Service-Fenster angezeigt, wenn dieser Record aktiv ist.
Dieser Menüpunkt ist nur im Record-Modus anwählbar.
Querverweise: Lösche Markierung, Markiere alle Records, Lösche alle Markierungen

10.4.36 Record/Lösche Markierung (Record/Unmark Record)

Shortcut: *A* :

Entfernt die Markierung des aktuellen Records. Der Record wird danach wieder normal dargestellt.

Dieser Menüpunkt kann nur im Record-Modus angwählt werden.
Querverweise: Markiere Record, Markiere alle Records, Lösche alle Markierungen

10.4.37 Record/Markiere alle Records (Record/Mark all Records)

Shortcut: *A* ,

Markiert alle Records des aktiven Projektes. Bitte beachten Sie, daß die alten Markierungen der Records überschrieben werden.

Dieser Menüpunkt kann nur im Record-Modus angewählt werden.
Querverweise: Markiere Record, Lösche Markierung, Lösche alle Markierungen, Markierungen umschalten

10.4.38 Record/Lösche alle Markierungen (Record/Unmark all Records)

Shortcut: *A* ;

Löscht die Markierungen aller Records im aktiven Projekt. Bitte beachten Sie, daß die alten Markierungen der Records dabei verlorengehen.

Dieser Menüpunkt kann nur im Record-Modus angewählt werden.
Querverweise: Markiere Record, Lösche alle Markierungen, Markiere alle Records, Markierungen umschalten

10.4.39 Record/Markierungen umschalten (Record/Toggle all Marks)

Kein Shortcut

Schaltet die Markierungen aller Records des aktiven Projektes um. Das heißt, daß ein markierter Record ein nicht markierter wird und ein nicht markierter wird ein markierter. Die ursprünglichen Markierungen kann man durch wiederholtes Anwählen dieser Funktion wiederherstellen.

Dieser Menüpunkt kann nur im Record-Modus angewählt werden.
Querverweise: Markiere Record, Lösche Markierung, Markiere alle Records, Lösche alle Markierungen

10.4.40 Feld/Feldtyp (Field/Fieldtype)

Wählen Sie in diesem Untermenü den aktuellen Feldtypen aus. Dieser wird beim Erstellen von Feldern benutzt. Das Gadget am unteren Rand des Service Fensters hat dieselbe Funktion.
Diese Feldtypen sind verfügbar (mit Shortcut):

String	Ctrl S
Integer	Ctrl I
Float	Ctrl F
Boolean	Ctrl B
Cycle	Ctrl C
Slider	Ctrl S
Datum	Ctrl A
Zeit	Ctrl M
Extern	Ctrl E
Datatypes	Ctrl D
Var String	Ctrl V
Text	Ctrl T
Button	Ctrl U
Leiste	Ctrl R

10.4.41 Feld/Feld Hinzufügen... (Field/Add Field)

Shortcut: Return

Öffnet den Feld-Requester (siehe Abschnitt 10.6.11) für den aktiven Feldtyp und fügt das so erstellte Feld an der Cursorposition ein.

Dieser Menüpunkt ist nur im Masken-Modus anwählbar.

Falls sich an der Cursorposition schon ein Feld befindet, läßt sich kein Feld erzeugen.

Bitte beachten Sie, daß Enter auch Shortcut für Aktives Feld Ändern ist. Enter hat die Funktion Feld Hinzufügen, wenn kein Feld aktiv ist, ansonsten wird Aktives Feld Ändern aufgerufen.
Querverweise: Aktives Feld Ändern, Benanntes Feld Ändern, Relation Ändern, Feld Entfernen

10.4.42 Feld/Aktives Feld Ändern... (Field/Edit active Field)

Shortcut: Enter

Öffnet den Feld-Requester (siehe Abschnitt 10.6.11) für das aktive Feld. Der Feld-Requester kann benutzt werden, um mehrere Attribute des Feldes zu ändern. Wenn bestimmte Änderungen Datenverlust bedeuten würden (z.B. das Ändern von Max. Zeichen eines String-Feldes auf eine niedrige Zahl), werden Sie vor dem Problem gewarnt. Feldtypen können nicht hiermit geändert werden. Dafür müssen Sie Feld konvertieren (siehe Abschnitt 10.4.50) benutzen.

Da diese Funktion das ausgewählte Feld ändert, können Sie verborgene Felder nicht ändern. Benanntes Feld Ändern (siehe Abschnitt 10.4.43) dient diesem Zweck.

Bitte beachten Sie, daß Enter auch Shortcut für Feld Hinzufügen ist. hat die Funktion Feld Hinzufügen, wenn kein Feld aktiv ist, ansonsten wird Aktives Feld Ändern aufgerufen.

Querverweise: Feld Hinzufügen, Benanntes Feld Ändern, Relation Ändern, Feld Entfernen

10.4.43 Feld/Benanntes Feld Ändern... (Field/Edit named Field)

Shortcut: Shift Enter

Öffnet einen Requester mit einer Liste aller Felder. Wenn Sie eins auswählen, öffnet sich der Feld-Requester (siehe Abschnitt 10.6.11) für das ausgewählte Feld. Der Feld-Requester kann zum Ändern mehrerer Attribute des Feldes benutzt werden. Wenn bestimmte Änderungen Datenverlust bedeuten würden (z.B. das Ändern von Max. Zeichen eines String-Feldes auf eine niedrige Zahl), werden Sie vor dem Problem gewarnt. Feldtypen können nicht hiermit geändert werden. Dafür müssen Sie Feld konvertieren (siehe Abschnitt 10.4.50) benutzen.

Dieser Menüpunkt kann nur im Masken-Modus angewählt werden.

Querverweise: Aktives Feld Ändern, Feld Hinzufügen, Relationen Ändern

10.4.44 Feld/Feld Duplizieren (Field/Duplicate Field)

Kein Shortcut

Macht eine exakte Kopie des aktiven Feldes. Es wird so nah wie möglich am Original abgelegt. Die ID wird copy_of_*FeldID* sein.

10.4.45 Feld/Feld Entfernen (Field/Remove Field)

Shortcut: Del

Entfernt das aktive Feld aus der Maske und somit auch alle damit verbundenen Daten. Falls sich noch irgendwelche Relationen auf dieses Feld verlassen, werden diese nicht funktionsfähig sein. *Achtung:* Relationen oder ARexx-Scripts werden sich erst beim nächsten Zugriff über diesen Zustand beschweren!

Dieser Menüpunkt ist nur im Masken-Modus anwählbar.
Querverweise: Feld Ändern, Relation Ändern, Feld Ändern

10.4.46 Feld/Relation Ändern... (Field/Edit Relation)

Shortcut: *A* &

Über diesen Punkt können sie die Relationen (siehe Abschnitt 5.5) für das aktive Feld im Relations-Requester (siehe Abschnitt 10.6.14) einstellen.

Relation Ändern ist nur anwählbar, wenn das Projekt im Masken-Modus ist.
Querverweise: Relationen entfernen

10.4.47 Feld/Relation entfernen (Field/Remove Relation)

Shortcut: *A* 0

Dieser Menüpunkt löscht alle Relations-Informationen des gerade aktiven Feldes. Danach werden die Daten in diesem Feld wieder in die normale Datei geschrieben werden.

Dieser Menüpunkt kann nur im Masken-Modus angewählt werden.

10.4.48 Feld/Gruppe Bilden (Field/Create Group)

Shortcut: *A* J

Erzeugt eine Gruppe (siehe Abschnitt 5.2) der aktiven Felder. Wenn Sie eine Gruppe mit anderen Feldern oder Gruppen ausgewählt haben, werden diese in einer großen Gruppe zusammengefaßt.

Mit Feld/Gruppe Auflösen können Sie die Felder wieder unabhängig machen.

Querverweise: Gruppe Auflösen, Abschitt Gruppen

10.4.49 Feld/Gruppe Auflösen (Field/Resolve Group)

Shortuct: *A* /

Löst die aktive Gruppe auf. Alle Felder dieser Gruppe werden unabhängig. Gruppen, die in diese Gruppe gruppiert wurden, werden auch aufgelöst.
Querverweise: Gruppe Bilden, Abschnitt Gruppen

10.4.50 Feld/Feld Konvertieren... (Field/Convert Field)

Shortcut: *A* "

Öffnent den Umwandlungs-Requester (siehe Abschnitt 10.6.13) für das aktive Feld. Mit Feld Konvertieren kann man den Typ eines Feldes auf einfache Weise ändern.

Dieser Menüpunkt kann nur im Masken-Modus angewählt werden.
Querverweise: Feld Hinzufügen, Feld Ändern

10.4.51 Liste/Spalte verbergen (List/Hide column)

Shortcut: *A* [

Über diesen Menüpunkt kann man eine Spalte aus dem Listenfenster entfernen. Es wird die aktuelle Spalte, die über den Listenkopf ausgewählt werden kann, entfernt. Die Spalten rechts davon rücken danach auf. Die Spalte kann über Spalte sichtbar machen (siehe Abschnitt 10.4.52) wieder in die Liste eingefügt werden.

Dieser Menüpunkt kann sowohl im Record- als auch im Masken-Modus aufgerufen werden, wenn das Listen-Fenster offen ist.

10.4.52 Liste/Spalte sichtbar machen... (List/Show column)

Shortcut: *A*]

Dieser Menüpunkt ermöglicht es, die Spalten, die mit Spalte verbergen aus der Liste entfernt wurden, wieder in die Liste einzufügen. Fiasco versucht,

die in einem Requester ausgewählte Spalte so nahe wie möglich an der alten Position wieder einzufügen.

Dieser Menüpunkt kann sowohl im Record- als auch im Masken-Modus aufgerufen werden, wenn das Listen-Fenster offen ist.

10.4.53 Liste/Alle Spalten sichtbar (List/Show all columns)

kein Shortcut

Fügt alle Spalten, die über Spalte verbergen aus der Liste entfernt wurden, in einem Schritt wieder ein.

Dieser Menüpunkt kann sowohl im Record- als auch im Masken-Modus aufgerufen werden, wenn das Listen-Fenster offen ist.

10.4.54 Liste/Liste neu berechnen (List/Recalc List)

Shortcut: *A* %

Dieser Menüpunkt berechnet alle Positionen und Dimensionen der Spalten in der Liste neu. Verborgene Spalten werden jedoch nicht wieder angezeigt.

Dieser Menüpunkt kann mit dem Menüpunkt Inhalt aufräumen (Clean up) der Workbench verglichen werden.

Dieser Menüpunkt kann nur bei offenem Listen-Fenster angewählt werden.

10.4.55 Vergleichen/Suchen... (Compare/Find)

Shortcut: *A* F

Öffnet den Such-Requester (siehe Abschnitt 10.6.17), in dem man Suchkriterien bestimmen kann.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.
Querverweise: Suchrequester, Weitersuchen, Rückwärts suchen

10.4.56 Vergleichen/Weitersuchen (Compare/Find next)

Shortcut: *A* >

Aktiviert den *nächsten* Eintrag, der mit den Suchkriterien übereinstimmt,

die mit dem Such-Requester (siehe Abschnitt 10.6.17) spezifiziert worden sind. Falls kein Eintrag mehr gefunden wird, werden Sie darüber informiert werden.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.

Querverweise: Such-Requester, Suchen, Rückwärts suchen

10.4.57 Suchen/Rückwärts Suchen (Compare/Find previous)

Shortcut: *A* <

Aktiviert den *vorherigen* Eintrag, der mit den Suchkriterien übereinstimmt, die mit dem Search-Requester (siehe Abschnitt 10.6.17) spezifiziert worden sind. Falls kein Eintrag mehr gefunden wird, werden Sie darüber informiert werden.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.

Querverweise: Such-Requester, Suchen, Weitersuchen

10.4.58 Vergleichen/Filter... (Compare/Filter)

Shortcut: *A* ∼

Öffnet den Filter-Requester (siehe Abschnitt 10.6.18), mit dem man einen Filter erstellen kann.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.

10.4.59 Vergleichen/Ersetzen... (Compare/Replace)

Shortcut: *A* R

Öffnet den Ersetzen-Requester (siehe Abschnitt 10.6.19), mit dem Parameter für das Ersetzen von Daten angegeben werden können.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.

10.4.60 Vergleichen/Zählen... (Compare/Count)

Shortcut: *A* #

Öffnet den Count-Requester (siehe Abschnitt 10.6.20), mit dem Sie die Anzahl des Vorliegendes Musters in dem Projekt zählen können.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.
Querverweise: Suchen

10.4.61 Vergleichen/Sortieren... (Compare/Sort)

Shortcut: *A* =

Öffnet den Sortier-Requester (siehe Abschnitt 10.6.22), der zum Erstellen eines sortierten Index der aktiven Datenbank benutzt werden kann.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.

10.4.62 Vergleichen/Markieren... (Compare/Mark)

Shortcut: *A* K

Öffnet den Markieren-Requester (siehe Abschnitt 10.6.21), der zum Markieren von Records benutzt werden kann, die auf ein Suchmuster passen.

Existierende Markierungen werden überschrieben; Markierte Records werden demarkiert, wenn sie nicht zu einem Muster passen.

Dieser Menüpunkt ist nur anwählbar, wenn sich das aktuelle Projekt im Record-Modus befindet und mindestens einen Record enthält.

10.4.63 Kontrolle/Record-Modus (Control/Record Mode)

Shortcut: Ctrl F1

Hiermit können Sie das aktuelle Projekt in den Record-Modus (siehe Abschnitt 3.6.1) versetzen, in dem sie Records und deren Inhalt verändern können. Wenn dieser Modus aktiv ist, wird ein Haken vor den Menüpunkt gesetzt. Das Gegenstück zu diesem Punkt ist Kontrolle/Masken-Modus (siehe Abschnitt 10.4.64).
Querverweise: Record-Modus, Masken-Modus

10.4.64 Kontrolle/Masken-Modus (Control/Mask Mode)

Shortcut: Ctrl F2

Hiermit können Sie das aktuelle Projekt in den Masken-Modus (siehe Abschnitt 3.6.2) versetzen, in dem sie die Maske verändern können. Wenn dieser Modus aktiv ist, wird ein Haken vor den Menüpunkt gesetzt. Das Gegenstück zu diesem Punkt ist .
Querverweise: Masken-Modus, Record-Modus

10.4.65 Kontrolle/Service-Fenster (Control/Service Window)

Shortcut: Ctrl F3

Mit Hilfe dieses Menüpunkts können sie Fiascos Service-Fenster (siehe Abschnitt 10.3) öffnen und schließen. Das Service-Fenster erleichtert die wichtigsten Record- und Maskenoperationen und zeigt Statusmeldungen an. Das Fenster kann auch über das Close-Gadget geschlossen werden.

Das Service-Fenster ist global für alle Projekte zuständig.

10.4.66 Kontrolle/Listen-Fenster (Control/List Window)

Shortcut: Ctrl F4

Mit diesem Menüpunkt können sie das Listen-Fenster (siehe Abschnitt 3.5) öffnen und schließen. Das Fenster kann auch über das Close-Gadget geschlossen werden.

Jedes Projekt kann seine eigene Liste haben.

10.4.67 Kontrolle/ARexx-Debug (Control/ARexx-Debug)

Kein Shortcut

Hier kann man den speziellen Debug-Modus für die ARexx-Schnittstelle (siehe Abschnitt 12) von Fiasco aktivieren. Wenn ARexx-Kommandos von Fiasco Fehler erzeugen, werden zusätzlich Requester angezeigt, die das Problem näher bezeichen.

10.4.68 Einstellungen/Datenbanken... (Settings/Databases)

Kein Shortcut

Öffnet den Datenbank-Einstellungen-Requester (siehe Abschnitt 10.6.23). Sie können diesen Requester zum Ändern bestimmter Optionen benutzen, die mit Fiasco-Datenbanken zusammenhängen.

10.4.69 Einstellungen/Benutzeroberfläche... (Settings/User Interface)

Kein Shortcut

Dieser Menüpunkt öffnet den Benutzeroberflächen-Einstellungs-Requester (siehe Abschnitt 10.6.24), der bestimmte Elemente von Fiascos GUI kontrolliert, wie z.B. das Service-Fenster, Feld-Aktivierung, etc.

10.4.70 Einstellungen/Benutzermenü... (Settings/User Menu)

Kein Shortcut

Öffnet den Benutzermenü-Requester (siehe Abschnitt 10.6.25), in dem man das Usermenü verändern kann.

10.4.71 Einstellungen/Anzeige... (Settings/Display)

Kein Shortcut

Öffnet den Anzeige-Requester (siehe Abschnitt 10.6.26), der zum Einstellen der Fiasco-Anzeige benutzt werden kann. Sie können hier angeben, ob Fiasco auf einem eigenen Screen oder einem öffentlichen Screen laufen soll. Weiterhin ist hier die Auswahlmöglichkeit für Schriften.

10.4.72 Einstellungen/Externe Programme und Pfade... (Settings/External)

Kein Shortcut

Öffnet den Externe Programme und Pfade-Requester (siehe Abschnitt 10.6.27), in dem Sie die Programme angeben können, die Fiasco aufruft. Außerdem können Sie noch Pfade zu verschiendenen Zwecken angeben.

10.4.73 Einstellungen/Einstellungen speichern (Settings/Save Settings)

Speichert die Einstellungen in den Dateien "env:fiasco.prefs" und "envarc:fiasco.prefs". Sie sind somit auch nach einem Reboot aktiv.

10.4.74 Einstellungen/Einstellungen speichern als... (Settings/Save Settings as)

Öffnet einen ASL-Requester, in dem man eine Datei auswählen kann, in der die Einstellungen gespeichert werden sollen. Wenn Sie die Datei in "env:" speichern, werden die Einstellungen einen Neustart des Systems nicht überstehen. Wenn Sie die Einstellungen in "envarc:" speichern, werden diese erst nach einem Neustart aktiv, da Fiasco nur in "env:" nach seinen Einstellungen sucht.

10.4.75 Einstellungen/Einstellungen laden... (Settings/Load Settings)

Öffnet einen ASL-Requester, in dem man eine Datei auswählen kann, aus der die Einstellungen geladen werden sollen. Sie werden danach benutzt. Um sie dauerhaft benutzen zu können, sollte man sie mit Einstellungen speichern in "envarc:" und "env:" sichern.

10.5 Das Druckfenster

Das Druckfenster kann mit Projekt/Drucken geöffnet werden. Von hier aus können Sie die Druckfunktion von Fiasco kontrollieren. Mehr über die Druckfunktion finden Sie in Kapiel 7.

Das Druckfenster enthält diese Menüs:

10.5.1 Projekt/Leeren (Project/Erase)

Shortcut: *A* Z

Entfernt alle Elemente aus dem Druckfenster. Nach der Benutzung dieses Menüpunktes ist das Druckfenster völlig leer.

10.5.2 Projekt/Öffnen... (Project/Open)

Shortuct: *A* O

Öffnet einen Datei-Requester und liest die Druckmaske aus der ausgewählten Datei. Die alten Daten werden überschrieben.

10.5.3 Projekt/Von Maske (Project/Get from Mask)

Shortcut: *A* M

Dieser Menüpunkt versucht, das Layout der Projekt-Maske im Druckfenster nachzuahmen. Die alte Druckmaske wird überschrieben.

10.5.4 Projekt/Von Liste (Project/Get from List)

Shortcut: *A* L

Dieser Menüpunkt versucht, das Layout der Liste im Druckfenster nachzuahmen. Die alte Druckmaske wird überschrieben.

10.5.5 Projekt/Speichern (Project/Save)

Shortcut: *A* S

Mit Speichern können Sie die aktuelle Druckmaske in eine Datei auf Disk schreiben. Falls Sie noch keine andere mit Öffnen oder Speichern als ausgewählt haben, hat die Datei den Namen *Projekt_Name*.fpr. Der Dateiname wird in der Titelzeile des Druckfensters angezeigt.

10.5.6 Projekt/Speichern als... (Project/Save as)

Shortcut: *A* A

Wählen Sie diesen Menüpunkt an, wenn Sie die Druckmaske in einer anderen Datei als der ausgewählten speichern möchten.

Der Name der aktiven Datei wird in der Titelzeile des Druckfensters angezeigt.

10.5.7 Projekt/Drucken (Project/Print)

Shortcut: *A* P

Dieser Menüpunkt erstellt den Ausdruck des Projekts mit Hilfe der aktiven Druckmaske. Die exakte Funktion dieses Menüpunktes hängt von den Einstellungen im Druckoptionen-Requester (siehe Abschnitt 10.6.28) ab.

10.5.8 Projekt/Optionen... (Project/Options)

Shortcut: *A* T

Dieser Menüpunkt öffnet den Druckoptionen-Requester (siehe Abschnitt 10.6.28), in dem Sie einige Einstellungen für die Druckmaske vornehmen können.

10.5.9 Projekt/Verlassen (Project/Exit)

Shortcut: *A* Q

Dieser Menüpunkt schließt das Druckfenster. Die aktive Druckmaske wird aus dem Speicher entfernt.

10.5.10 Element/Element Typ (Element/Element Type)

Benutzen Sie diese Untermenü, um den aktiven Element-Typ auszuwählen. Dieser Typ wird von nachfolgenden Element/Hinzufügen-Aufrufen benutzt.

Sie haben die Wahl zwischen diesen Element-Typen:

- Feld (Field) (Ctrl F)
- Text (Text) (Ctrl T)
- Seitenumbruch (Formfeed) (Ctrl O)

Mehr Informationen dazu befinden sich im Kapitel Drucken.

10.5.11 Element/Hinzufügen... (Element/Add)

Shortcut: Return

Erzeugt ein neues Element an der Cursorposition. Der Typ des erzeugten Elements wird der unter Element/Element Typ (siehe Abschnitt 10.5.10) ausgewählte Typ sein. Wenn der Typ einen Requester unterstützt, wird der Element-Requester (siehe Abschnitt 10.6.29) erscheinen.

Mehr Informationen dazu befinden sich im Kapitel Drucken.

Bitte beachten Sie, daß dieser Menüpunkt das selbe Tastaturkürzel wie Element/Ändern (siehe Abschnitt 10.5.12) hat. Dieses Kürzel wird Hinzufügen wenn noch kein Element aktiv ist und Ändern wenn bereits ein Element aktiv ist.

10.5.12 Element/Ändern...

Shortcut: Return

Öffnet den Element-Requester (siehe Abschnitt 10.6.29) für das aktive Element. Elemente können mit der Maus oder den Cursor-Tasten aktiviert werden.

Bitte beachten Sie, daß dieser Menüpunkt das selbe Tastaturkürzel wie Element/Hinzufügen (siehe Abschnitt 10.5.11) hat. Dieses Kürzel wird Hinzufügen wenn noch kein Element aktiv ist und Ändern wenn bereits ein Element aktiv ist.

10.5.13 Element/Duplizieren (Element/Duplicate)

Kein Shortcut

Macht eine exakte Kopie des aktiven Elements.

10.5.14 Element/Entfernen (Element/Remove)

Shortcut: Del

Entfernt das aktive Element. Dieses Element kann nur mit einer gespeicherten Version der Druckmaske wiedergewonnen werden.

10.5.15 Kontrolle/Kopfteil Ändern (Control/Edit Head)

Shortcut: *A* K

Wählt den Kopfteil der Druckmaske für Veränderungen aus. Der Kopfteil wird vor allen anderen Daten gedruckt. Er darf keine Feld-Elemente

beinhalten. Dieser Menüpunkt schließt die Menüpunkte Hauptteil Ändern und Fußteil Ändern aus.

Das Kapitel Drucken enthält mehr Informationen.

10.5.16 Kontrolle/Hauptteil Ändern (Control/Edit Body)

Shortcut: *A* H

Wählt den Hauptteil der Druckmaske für Veränderungen aus. Der Hauptteil wird für jeden Record gedruckt. Er kann Verweise auf Felder in der Form von Feld-Elementen enthalten. Diese Verweise werden während des Druckens durch die Feldinhalte ersetzt. Dieser Menüpunkt schließt die Menüpunkte Kopfteil Ändern und Fußteil Ändern aus.

Das Kapitel Drucken enthält mehr Informationen.

10.5.17 Kontrolle/Fußteil Ändern (Control/Edit Foot)

Shortcut: *A* F

Wählt den Fußteil der Druckmaske für Veränderungen aus. Der Fußteil wird nach allen anderen Daten gedruckt. Er darf keine Feld-Elemente beinhalten. Dieser Menüpunkt schließt die Menüpunkte Hauptteil Ändern und Kopfteil Ändern aus.

Das Kapitel Drucken enthält mehr Informationen.

10.6 Alle Requester

Requester (manchmal auch Dialogfenster genannt) werden von Fiasco benutzt, um Informationen zu bekommen, die für bestimmte Operationen notwendig sind. Normalerweise werden Requester nach dem Aufrufen eines Menüpunktes von Fiasco erzeugt. Sogenannte EasyRequester, mit denen Fiasco eine einfache Auswahl verlangt, werden hier nicht erklärt, da diese meistens leicht verständlich sind und meistens in den Bereichen der jeweiligen Funktion beschrieben sind.

Die meisten Requester können auch mit der Tastatur kontrolliert werden. Die Shortcuts, die durch einen unterstrichenen Buchstaben gekennzeichnet werden, sind normalerweise einzelne Zeichen ohne Zusatztasten (wie Ctrl, etc.)

Die Gadgets am unteren Rand eines Requesters sind normalerweise für das Bestätigen oder sonstiges Fortschreiten in einer Funktion gedacht. Nor-

malerweise ist das linke Gadget eine positive Antwort (Ok), während das rechte eine negative Antwort darstellt (Cancel). Das positive Gadget, das zusätzlich hervorgehoben ist, hat Enter als Shortcut. Das negative Gadget hat Escape als Shortcut.

10.6.1 Importieren-Requester

Der Importieren-Requester ist die GUI-Schnittstelle zur Import-Funktion (siehe Abschnitt 8) von Fiasco. Durch Import ist Fiasco in der Lage, Daten von anderen Datenbank-Programmen zu lesen. Meistens kann dies nicht direkt geschehen. Die fremde Datenbank muß dann die Datem "exportieren". Sie können verschiedene Parameter für das Importieren angeben, wodurch Sie in der Lage sein sollten, fast alle Import-/Export-Formate in Fiasco einlesen zu können.

Die Fiasco-Distribution enthält mehrere fertige Import-Formate, die mit dem Laden-Schalter am unteren Rand des Import-Requesters eingelesen werden können.

Die Werte, die in die Gadgets des Import-Requesters eingegeben werden können, sind im Import/Export-Kapitel dieses Dokuments beschrieben.

Datei: Geben Sie hier die Datei an, die die zu importierenden Daten enthält. Sie können den Auswahl-Schalter an der rechten Seite benutzen, um die Datei mit einem ASL-Requester auszuwählen.

Anzeigen: Klicken Sie hier, wenn Sie den Inhalt der Datei anzeigen wollen. Fiasco wird asynchron More oder MultiView, falls dies verfügbar ist, starten.

Records/Start: Geben Sie hier die Anfangs-Zeichen für Records ein. Standard-Wert: Leer.

Records/Ende: Geben Sie hier die End-Zeichen für Records ein. Standard-Wert: Leer.

Records/Tenner: Geben Sie hier die Zeichen zwischen zwei Records ein. Standard-Wert: \n.

Felder/Start: Geben Sie hier die Zeichen an, mit denen Felder beginnen. Standard-Wert: ".

Felder/Ende: Geben Sie hier die Zeichen an, mit denen Felder enden. Standard-Wert: ".

Felder/Trenner: Geben Sie hier die Zeichen zwischen zwei Feldern an. Standard-Wert: \t.

Verschiedenes/Zeilen überspringen: Geben Sie hier die Start-Zeichen für Kommentare an. Standard-Wert: Leer.

Verschiedenes/Am Start überspringen: Geben Sie hier die Anzahl von Zeilen an, die am Anfang übersprungen werden sollen. Standard-Wert: 0.

Verschiedenes/Maximale Felder: Geben Sie hier die maximale Anzahl von Feldern eines Records an. Dieser Wert kann auch benutzt werden, wenn es keine Zeichen gibt, die das Ende eines Records markieren. Standard-Wert: 100.

Optionen/Erster Record enthält IDs: Aktivieren Sie dieses Gadget, wenn der erste Record der Datei die IDs der Felder enthält. Fiasco wird dann diese IDs statt generierter IDs benutzen.

Optionen/Neue Felder anhängen: Aktivieren Sie diese Option, wenn Sie wollen, daß Fiasco neue Felder für die Daten erzeugt und nicht die bereits bestehenden benutzt. Wenn Sie ein völlig leeres Projekt haben, sollten Sie diese Option aktivieren. Wenn diese Option nicht aktiv ist, wird Fiasco zuerst versuchen, die Daten nach ihren IDs zu assoziieren. Wenn nicht genug Felder zur Verfügung stehen, wird Fiasco dennoch neue Felder erzeugen.

Optionen/Altes Projekt überschreiben: Entfernt die alten Daten im aktuellen Projekt-Fenster. Wenn Sie diese Option nicht aktivieren, werden Ihre Daten irgendwie an das existierende Projekt angehängt.

Ok: Startet den Import-Vorgang. Bitte beachten Sie, daß der Speicher aufgrund von falschen Struktur-Parameters oder zu großen Dateien knapp werden kann. Programme, die Probleme mit wenig Speicher haben, sollten während dieses Vorganges nicht laufen.

Sichern: Speichert die aktuellen Einstellungen in einer ausgewählten Datei.

Laden: Liest Einstellungen aus einer ausgewählten Datei und aktiviert diese im Requester.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.2 Exportieren-Requester

Die Export-Funktion gibt die Möglichkeit, Daten, die mit Fiasco erstellt wurden, anderen Datenbank-Programmen zugänglich zu machen. Normalerweise können fremde Datenbanken die Daten von Fiasco nicht lesen, da

sie das Format der Dateien nicht kennen. Der Import/Export-Abschnitt dieses Dokumentes enthält mehr Informationen über diesen Mechanismus.

Datei: Geben Sie hier den Namen der Datei an, in die die Daten geschrieben werden sollen. Falls eine Datei mit diesem Namen bereits bestehen sollte, wird sie überschrieben.

Records/Start: Geben Sie hier die Start-Zeichen für Records an. Standard-Wert: Leer.

Records/Ende: Geben Sie hier die End-Zeichen für Records an. Standard-Wert: Leer.

Records/Trenner: Geben Sie hier die Zeichen zwischen zwei Record an. Standard-Wert: `\n`.

Felder/Start: Geben Sie hier die Zeichen an, mit denen Felder beginnen sollen. Standard-Wert: ".

Felder/Ende: Geben Sie hier die Zeichen an, mit denen Felder enden sollen. Standard-Wert: ".

Felder/Trenner: Geben Sie hier die Zeichen zwischen zwei Feldern an. Standard-Wert: `\t`.

Optionen/Erster Record enthält IDs: Aktivieren Sie dieses Gadget, wenn Sie wollen, daß Fiasco die Feld-IDs in den ersten Record schreibt.

Optionen/Nur markierte Records: Aktivieren Sie dieses Gadget, wenn Sie wollen, daß Fiasco nur Records schreibt, die markiert sind.

Listviews/Trenner für Einträge: Geben Sie hier die Zeichen zwischen zwei Einträgen eines Listview-Feldes. Standard-Wert: | .

Ok: Klicken Sie hier, um den Export-Vorgang zu starten.

Sichern: Speichert die Struktur-Parameter in einer ausgewählten Datei.

Laden: Läd die Struktur-Parameter aus einer ausgewählten Datei.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.3 Projekt anzeigen-Requester

Dieser Requester wird zum Anzeigen eines Fiasco-Projektes benutzt, das mit Projekt/Verbergen (siehe Abschnitt 10.4.10) verboren wurde. Diese Requester kann mit Projekt/Sichtbar machen (siehe Abschnitt 10.4.11) oder durch Doppelklicken auf das Fiasco-Icon auf der Workbench geöffnet werden.

Projekt: Dies ist eine Liste aller verborgenen Fiasco-Projekte. Um ein Projekt wieder sichbar zu machen, wählen Sie es aus und klicken Sie Ok oder klicken Sie zweimal auf den Namen.

Ok: Schließt den Requester und öffnet das ausgewählte Projekt.

Cancel: Schließt den Requester ohne weitere Operation.

10.6.4 Projekt Optionen-Requester

Der Optionen-Requester dient dazu, einige Einstellungen speziell für das aktuelle Projekt zu machen. Er ist über den Menüpunkt Datenbank/Optionen erreichbar.

Autor: Hier kann sich der Ersteller des Projektes verewigen. Der Name wird sich dann am Anfang der Projekt-Datei finden. Ansonsten wird der Inhalt ig- noriert.

Bemerkungen: Noch ein Gadget zur freien Benutzung. Hier kann man irgendwelche Notitzen, z.B. einen Versions-String (mit $VER: am Anfang) eintragen. Dieser String wird noch vor dem Autor abgelegt. Und wiederum wird der Inhalt sonst ignoriert.

Maskendehnung X / Y: Hier kann man einen Wert angeben, der zur Breite bzw. der Höhe des Cursors hinzuaddiert wird. Dies bewirkt eine Dehnung der Make in X-Richtung bzw. Y-Richtung. Mehr dazu in Abschnitt 10.1.1.

Fenster/Liste beim Start öffnen: Wenn dieses Gadget aktiv ist, öffnet Fiasco das Listen-Fenster automatisch wenn das Projekt geladen wird.

Fenster/Listen-Position fest: Aktivieren Sie dieses Gadget, damit Fiasco sich die Position der Liste beim Speichern auf dem Bildschirm merkt. Fiasco öffnet das Listen-Fenster danach wieder an dieser Position.

Fenster/Masken-Position fest: Aktivieren Sie dieses Gadget, damit Fiasco sich die Position des Masken-Fensters beim Speichern auf dem Bildschirm merkt. Fiasco öffnet das Masken-Fenster danach wieder an dieser Position.

Fenster/Schließ-Gadget: Wenn dieses Gadget aktiv ist, bekommt das Masken-Fenster dieser Datenbank ein Close-Gadget, mit dem die Datenbank geschlossen werden kann.

Fenster/Tiefen-Gadget: Wenn dieses Gadget aktiv ist, bekommt das Masken-Fenster dieser Datenbank ein Depth-Gadget, mit dem das Fenster in der Tiefe auf dem Bildschirm angeordnet werden kann.

Fenster/Ziehbar: Wenn dieses Gadget aktiv ist, kann das Masken-Fenster dieser Datenbank auf dem Bildschirm verschoben werden.

Fenster/Größenveränderbar: Wenn dieses Gadget aktiv ist, wird das Masken-Fenster dieser Datenbank ein Size-Gadget zum Verändern der Fenstergröße haben und Proportional-Gadgets zum Verschieben des Fensterinhalts haben. Sonst werden die Fenster-Ränder dünn.

Records/Maximale Lesezeit: Stellen Sie hier die maxmale Zeit für Disk-Zugriff ein, wenn Records von Disk gelesen werden sollen. Die Zeit wird in Microsekunden bemessen. 65000 ist ein guter Wert, der relativ viele Records ohne störende Wartezeiten während Record-Wechseln liest. Höhere Werte führen zu längeren Pausen während Record-Wechseln und mehr gelesenen Records. Niedrigere Werte machen Record-Wechsel schneller und führen zu weniger gelesenen Records. Fiasco wird jedoch immer einen Record lesen (wenn einer vorhanden ist).

Records/Maximale Speicherauslastung: Stellen Sie hier die maximale RAM-Menge ein, die von dieser Datenbank für Records benutzt werden soll. Der Wert wird in Kilobytes angegeben. Dies ist nur ein Standard-Wert, da Fiasco nicht immer diesen Wert überprüft. Niedrigere Werte führen zu mehr Disk-Zugriffen, da mehr Records nochmal gelesen werden müssen. Der Vorgabe-Wert hierfür ist 200. Sie können die wirkliche RAM-Auslastung im Statistik-Requester (siehe Abschnitt 10.6.5) ablesen.

ARexx-Skripte/Datenbank-Start: Ein hier angegebenes ARexx-Script wird direkt nach dem Öffnen dieser Datenbank automatisch ausgeführt. *Beachte:* Die Benutzer-Oberflächen-Einstellungen (siehe Abschnitt 10.6.24) beinhalten eine Kontrolle, um diese Einstellung zu unterdrücken.

ARexx-Skripte/Datenbank-Ende: Ein hier angegebenes ARexx-Script wird ausgeführt, bevor diese Datenbank geschlossen wird. Es kann daher Aufgaben wie Aufräumen, etc. wahrnehmen. *Beachte:* Die Benutzer-Oberflächen-Einstellungen beinhalten eine Kontrolle, um diese Einstellung zu unterdrücken.

CD-ROM-Modus: Wenn aktiviert, ist diese Fiasco-Datenbank im CD-ROM-Modus. Dieser Modus unterstützt die Benutzung von Fiasco auf CD-ROMs (oder anderen nur lesbaren Medien). Es hat zwei Effekte: Alle Masken-Felder sind nur lesbar und alle Indizes, die von den Filter- und Sortieren-Funktionen erzeugt wurden, werden nach T: geschrieben. Die Funktion von Speichern und Speichern Als bleibt unverändert, um es zu ermöglichen, eine Datenbank im CD-ROM-Modus beim Vorbereiten einer CD-ROM-Distribution zu speichern.

10.6.5 Statistik-Requester

Abgesehen vom Ablesen von Informationen können Sie nicht viel im Statistik-Requester tun. Er dient nur zur Anzeige von bestimmten statistischen Werten der aktiven Fiasco-Datenbank.

Datenbank: Der Name der Datenbank wird hier angezeigt.

Index: Der Name des aktiven Index der Datenbank wird hier angezeigt. Wenn kein Index aktiv ist, wird hier – angezeigt. Dies ist der Fall, wenn eine Datenbank neu erstellt wurde oder die Datenbank noch im Fiasco 1.x-Format ist.

Records auf Disk: Dies ist die Anzahl der Records in der Datenbank-Datei.

Records in Index: Dies ist die Anzahl der Records im aktiven Index.

Hinzugefügte Records im RAM: Dies ist die Anzahl der Records, die seit dem letzten Speichern zur Datenbank hinzugefügt wurden.

Von Disk geladene Records im RAM: Dies ist die Anzahl der Records, die Fiasco von Disk gelesen hat und im Speicher hält. Dieser Wert hängt von der Einstellung von Maximale Speicherauslatung im Projekt-Options-Requester (siehe Abschnitt 10.6.4) ab. Weiterhin hängt er von der Anzahl ab, die Fiasco bisher lesen konnte.

Alle Records im RAM: Dies ist die Anzahl aller Records im RAM.

Geänderte Records im RAM: Dies ist die Anzahl von Records im RAM, die seit dem letzten Speichern geändert wurden. Dies schließt sowohl Records ein, die von Disk geladen wurden, als auch Records, die der Datenbank hinzugefügt wurden (diese Records sind immer 'geändert').

Gelöschte Records im RAM: Auch wenn es paradox klingt, ist dies die Anzahl von von Disk geladenen Records im RAM, die seit dem letzten Speichern der Datenbank gelöscht wurden. Fiasco hält von Disk geladene Records, die gelöscht wurden im RAM, um den Index während des nächsten Speicherns richtig verändern zu können. Hinzugefügte Records, die später gelöscht werden, werden von Fiasco nicht im RAM gehalten.

Freies RAM: Dies ist das insgesamt verfügbare RAM in Bytes.

RAM-Größe eines Records: Dies ist die Menge RAM, die von einem Record benötigt wird. Dies ist nur ein ungefährer Wert, da Fiasco Pools zur Speicher-Organisation benutzt.

RAM-Größe aller Records: Dies ist einfach RAM-Größe eines Records multipliziert mit Alle Records im RAM.

Unnötige Records Entfernen: Versucht, so viele Records wie möglich aus dem RAM zu entfernen. Records, die unter Hinzugefügte Records, Geänderte Records oder Gelöschte Records aufgeführt sind, können nicht entfernt werden. Um auch diese entfernen zu können, müssen Sie zuvor die Datenbank speichern.

Ok: Schließt den Requester.

10.6.6 Indizes Requester

Der Indizes-Requester kann zum Aktivieren, Neuerstellen, Ändern oder Löschen von Indizes (siehe Abschnitt 3.2) benutzt werden. Er kann über den Menüpunkt Datenbank/Indizes geöffnet werden.

Index-Dateien: Wählen Sie hier den Index aus, der von der nächsten Operationen benutzt werden soll. Zu beginn ist der aktive Index hier selektiert. Diese Liste ist leer, wenn die Datenbank bisher noch nie gespeichert wurde oder noch im Fiasco 1.x-Format ist. Wenn Sie auf einen Eintrag doppelklicken, wird die Ändern Funktion aufgerufen.

Neu: Öffnet den Index neu/ändern Requester (siehe Abschnitt 10.6.7), der einen neuen Index erstellt, wenn er mit Ok bestätigt wird.

Ändern: Öffnet den Index neu/ändern Requester für den im Listview ausgewählten Index.

Lösch.: Löscht den ausgewählten Index. Wenn dieser Index der letzte ist, werden Sie ihn nicht löschen können.

Ok: Schließt den Requester und führt schließlich alle Änderungen aus, die in dem Requester vorgenommen wurden. Dazu zählt das Erstellen neuer Indizes, Ändern oder Löschen von Indizes. Der ausgewählte Index wird aktiviert.

Abbrechen: Schließt den Requester und ignoriert alle Änderungen. Die Indizes werden unberührt bleiben.

10.6.7 Index neu/ändern Requester

Dieser Requester wird zum Eingeben von Optionen für Indizes (siehe Abschnitt 3.2) benutzt. Er kann auf zwei Wegen geöffnet werden: Einmal über Neu im Indizes-Requester. Wenn Sie den Requester auf diesem Weg geöffnet haben, wird er zusätzliche Kontrollen für die Erzeugung des Index enthalten. Die andere Möglichkeit, ist einen Index im Indizes-Requester zu Ändern.

Name: Dieses Gadget ist nur im Index neu-Requester vorhanden. Es nimmt den Namen des zu erstellenden Index auf.

Index-Quelle: Dieses Gadget ist nur im Index neu-Requester vorhanden. Wählen Sie hier den Index aus, der als Vorbild für den neuen Index benutzt werden soll. Der neue Index wird nur die Records enthalten, die auch der ausgewählte Index enthält. Wenn Sie nicht Index Sortierung benutzen, wird die Reihenfolge der Records übernommen. Wenn Sie weder Index Sortierung, noch Index Filter benutzen, wird eine Kopie des ausgewählten Index erstellt.

Der besondere Eintrag ≪Kein Index≫ steht für einen "virtuellen" Index, der alle Records der Datenbank enthält.

Index Filter: Wenn Sie auf den Schalter klicken, wird der Filter-Requester (siehe Abschnitt 10.6.18) geöffnet, der zum Einstellen von Kriterien benutzt werden kann, mit denen Fiasco entscheidet, ob der Records im neuen Index vorhanden sein wird, oder nicht. Dies wird benutzt, wenn ein neuer Index erstellt wird und wenn ein neuer Record zu der Datenbank hinzugefügt wird, während ein anderer Index aktiv ist und Neue Records automatisch hinzufügen aktiv ist. In anderen Fällen werden die Filter-Informationen nicht benutzt. Wenn

ein Index also hinzugefügt wird, wenn dieser Index aktiv ist, wird er immer dem Index hinzugefügt.

Index Sortierung: Wenn Sie den Schalter anklicken, öffnet sich der Sortier-Requester (siehe Abschnitt 10.6.22). Hier können Sie angeben, nach welchen Feldern der Index sortiert werden soll. Aktivieren Sie die Checkbox, wenn Sie wollen, daß Fiasco den Index auch sortiert hält. Fiasco wird dann immer dafür sorgen, daß nach dem Hinzufügen oder Ändern von Records der Index korrekt sortiert ist.

Neue Records automatisch hinzufügen: Aktivieren Sie diese Checkbox, wenn Sie wollen, daß Fiasco alle Records, die der Datenbank hinzugefügt werden auch automatisch diesem Index hinzufügt, nachdem der Index Filter und die Index Sortierung auf diese Records angewandt wurde. Normalerweise werden nur die Records einem Index hinzugefügt, die der Datenbank hinzugefügt wurden, während der Index aktiv war.

Ok: Akzeptiert die Einstellungen dieses Requesters. Die Einstellungen werden jedoch erst aktiv, wenn Sie auch den Indizes-Requester mit Ok bestätigt haben.

Abbrechen: Schließt den Requester und ignoriert alle neuen Einstellungen.

10.6.8 Funktionen-Requester

Im Funktionen-Requester können Sie benutzerdefinierbare Funktionen für Formeln (siehe Abschnitt 11.1.5) angeben.

Der Requester kann mit dem Menüpunkt Datenbank/Funktionen geöffnet werden.

Wichtig: Rekursive Programmierung ist mit diesen Funktionen nicht möglich.

Funktionen: Diese Liste zeigt alle Funktionen an, die bisher in der aktiven Datenbank definiert wurden. Wenn Sie eine auswählen, können Sie sie mit den anderen Gadgets verändern.

Name: Hier müssen Sie den Kopf der Funktion angeben. Der Kopf setzt sich aus dem Funktions-Namen und den Argumenten eingeschlossen in Klammern zusammen. Wenn Die Funktion keine Argumente hat, müssen Sie ein leeres Klammerpaar angeben. Beispiele: min(a, b) oder recordsum().

Definition: Nimmt die Definition der Funktion auf. Dies ist eine normale Formel. Die Argumente, die im Funktions-Kopf angegeben wurden können von hier aus wie normale Felder benutzt werden. Beispiel für den Funktions-Kopf min(a, b): a <b ? a : b.

Neu: Erzeugt einen neuen Funktionseintrag und aktiviert diesen.

Löschen: Löscht die gerade ausgewälte Funktion.

Ok: Überprüft alle Funktions-Köpfe und -Definitionen nach Fehlern. Falls keine Fehler auftreten, werden die neuen benutzerdefinierten Funktionen aktiviert. Wenn es zu diesem Zeitpunkt bereits Formeln in der Datenbank gibt, werden Sie gefragt, ob Sie diese Formeln neu übersetzen wollen.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.9 Konstanten-Requester

In diesem Requester können Sie benutzerdefinierte Konstanten für Formeln (siehe Abschnitt 11.1.3) angeben.

Der Requester kann mit dem Menüpunkt Datenbank/Konstanten geöffnet werden.

Konstanten: Diese Liste zeigt alle Konstanten an, die bisher in der aktiven Datenbank definiert wurden. Wenn Sie eine auswählen, können Sie sie mit den anderen Gadgets verändern.

Name: Hier müssen Sie den Namen einer Konstanten angeben. Für Konstanten-Namen gelten dieselben Einschränkungen die auch für Feld-IDs gelten.

Definition: Geben Sie hier den Wert für die Konstante an. Der Wert kann numerisch oder ein String sein, der in Anführungszeichen eingeschlossen ist.

Neu: Erzeugt einen neuen Konstanteneintrag und aktiviert diesen.

Löschen: Löscht die gerade ausgewälte Konstante.

Ok: Überprüft alle Konstanten-Namen und -Definitionen nach Fehlern. Falls keine Fehler auftreten, werden die neuen benutzerdefinierten Konstanten aktiviert. Wenn es zu diesem Zeitpunkt bereits Formeln in der Datenbank gibt, werden Sie gefragt, ob Sie diese Formeln neu übersetzen wollen.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.10 Gehe zu-Requester

Der Gehe zu-Requester ist einer der übersichtlichsten Requester in Fiasco überhaupt. Er ist über den Menüpunkt Record/Gehe zu erreichbar und bietet die Möglichkeit, über die Record-Nummer diesen Record zu aktivieren.

Bitte beachten Sie, die Record-Nummern mit verschiedenen Indizes (siehe Abschnitt 3.2) oder nach dem Hinzufügen oder Löschen von Records variieren können.

Record-Nummer: Nimmt die Nummer des Records auf.

Ok: Bestätigt den Requester und springt zum Record mit der Nummer.

Abbrechen: Ist mir gerade entfallen... %-)

10.6.11 Feld-Requester

Der Feld-Requester erlaubt das Manipulieren der Attribute jedes Feldes. Da jeder Feldtyp andere Attribute hat, unterscheidet sich auch der Feld-Requester von Feldtyp zu Feldtyp. Die Gadgets, die im Feld-Requester dargestellt werden entsprechen genau den Feld-Attributen, die bei den Felddokumentationen definiert sind.

Man wird mit dem Feldrequester konfrontiert, wenn man Felder/Feld hinzufügen, Felder/Aktives Feld ändern oder Felder/Benanntes Feld ändern anwählt oder auf einem Feld doppelklickt.

Wenn man den Requester mit Ok bestätigt, werden alle Werte auf Gültigkeit überprüft. Falls ein Wert nicht benutzt werden kann, wird man mit einem Requester darauf aufmerksam gemacht.

Hier eine kleine Zusammenfassung aller Bedingungen: (Vorausgesetzt, daß diese Attribute existieren)

- Eine eindeutige ID muß angegeben sein.
- Max. Zeichen muß > 0 sein.
- Breite muß > 2 sein.

Wenn Dimensionswerte aufgrund weiterer Felder, die im Weg sind, nicht benutzt werden können, erscheint ein anderer Requester, der einem die Möglichkeiten Schieben, Quetschen und Abbrechen anbietet. Abbrechen tut gar nichts, außer zum Feld-Requester zurückzukehren. Quetschen verkleinert das Feld soweit, daß es paßt. Schieben verschiebt das Feld so weit, daß es genug Platz hat. Schieben kann nicht immer angewandt werden.

Falls man ein bereits bestehendes Feld verändert, daß die Daten in String-Form speichert und Max. Zeichen unterstützt (also String, Datatypes und Extern) und Max. Zeichen so weit verkleinert hat, daß bestehende Einträge nicht mehr hinein passen würden, werden Sie gefragt, ob sie diese wirklich auf die neue Länge verkleinern möchten.

10.6.12 Popup-Gadget-Requester

Dieser Requester stellt die Optionen für das Popup-Gadget-Feld-Attribut ein. Der Requester kann nur vom Feld-Requester mit dem Schalter Vordefiniere Werte geöffnet werden.

Popup-Gadget: Hier können Sie ein Popup-Gadget für das Feld aktivieren und den Arbeits-Modus auswählen.

Vordefinierte Werte/Wert: Diese List ist nur aktiv wenn Sie Vordefinierten Wert auswählen unter Popup-Gadget ausgewählt haben. Dann können Sie hier Werte angeben, die später mit dem Popup-Schalter ausgewählt werden können.

Vordefinierte Werte/Neu: Erzeugt einen neuen Eintrag im obigen Listview.

Vordefinierte Werte/Löschen: Löscht den aktiven Eintrag im obigen Listview.

ARexx-Script: Dieses Gadget ist nur aktiv, wenn Sie ARexx-Script ausführen ausgewählt haben. Dann können Sie hier ein ARexx-Script angeben, das ausgeführt wird, wenn das Popup-Gadget ausgewählt wurde.

Ok: Bestätigt die Einstellungen:]

Abbrechen: Schließt den Requester ohne weitere Aktionen.

10.6.13 Feld umwandeln-Requester

Die Umwandlungs-Requester kann zum Ändern des Types eines Felds benutzt werden, ohne ein ARexx-Script dafür erstellen zu müssen.

Feld ID: Dieses Text-Gadget zeigt die ID des Feldes an, dessen Typ geändert werden soll. Bitte überprüfen Sie hier, ob Sie Feld umwandeln für das richtige Feld aufgerufen haben.

Alter Typ: Zeigt den aktuellen Typ des Feldes an.

Neuer Typ: Wählen Sie hier den neuen Typ des Feldes aus. Bitte beachten Sie, daß bei bestimmten Änderungen Daten verloren gehen können. Lesen Sie die Feld-Dokumentationen für mehr Informationen über dieses Thema.

Alternatives Format: Aktivieren Sie diese Option, um ein Ausgabe-Format zu erhalten, das von dem normalen Format abweicht. Der Abschnitt "Feldtypen" enthält mehr Informationen darüber, ob dieses Gadget Auswirkungen hat und welche.

Ok: Startet den Umwandlungs-Vorgang und schließt danach den Requester. Wenn das Feld noch durch irgendein anderes Fiasco-System benutzt werden sollte, wird ein Requester Ihnen die jeweiligen Systeme anzeigen und Ihnen die Wahl geben, ob Sie die Operation fortsetzen und diese Nutzer entfernen wollen oder die Operation abbrechen wollen.

Abbrechen: Schließt den Requester.

10.6.14 Relations-Requester

Dies ist die Schaltzentrale für die Relationen (siehe Abschnitt 5.5) in Fiasco.

Dieser Requester ist über Felder/Relation Ändern erreichbar.

Typ: Stellen Sie hier den Typ der Relation ein. Abschnitt 5.5.1 enthält Beschreibungen der einzelnen Typen. Die verfügbaren Typen hängen vom Typ des ausgewählten Feldes ab. Der ausgewählte Typ beeinflußt auch die Verfügbarkeit der anderen Felder als Schlüssel- und eigentliche Daten-Felder.

Schlüssel lokal: In diesem Listview muß man den Schlüssel in dem aktuellen Projekt auswählen.

eigentl. Feld lokal: Hier wird die ID des Feldes, dessen Relationen gerade bearbeitet werden, angezeigt.

Schlüssel entfernt: Hier muß man das Feld in dem Projekt, das unter Relations-Datei angegeben wurde, auswählen, das den eindeutigen Schlüssel beinhaltet. Da die Typen der beiden Schlüssel übereinstimmen müssen, werden hier nur die Felder angezeigt, die zu dem Schlüssel unter Schlüssel hier passen.

eigentl. Feld entfernt: Hier muß man das Feld in dem Projekt, das unter Relations-Datei angegeben wurde, auswählen, das den Partner zu eigentl. Feld hier darstellt. Aus diesem Feld werden dann die Daten, die in eigentl. Feld hier eingetragen werden, bezogen. Auch hier werden nur die Felder angezeigt, die zu eigentl. Feld hier passen (Typ und Max. Zeichen müssen gleich sein).

Relations-Datei: Hier muß man die Projekt-Datei angeben, in der die passenden Informationen stehen. Die Datei kann relativ zum Verzeichnis des aktuellen Projektes angegeben werden.

Ok: Bestätigt den Requester und versucht die Relationen zu laden. Falls dabei ein Fehler auftritt, wird der Requester mit einer entsprechenden Mitteilung wieder aktiviert, ansonsten wird das Haupt-Fenster aktiviert.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.15 Formel-Requester

Dieser Requester dient als Hilfsmittel zum Erstellen von Formeln für Fiasco. Er kann vom Feld-Requester oder dem Such-Requester aus geöffnet werden. Nachdem Sie den Requester geöffnet haben, können Sie eine Formel erstellen. Wenn Sie ihn mit Ok verlassen, wird die Formel in den aufrufenden Requester kopiert.

Operatoren: Dieses Listview zeigt alle Operatoren an, die von Fiasco unterstützt werden. Das Anklicken eines Operators wird ihn an der aktuellen Cursor-Position im Formula-String-Gadget einfügen.

Funktionen: Dieses Listview zeigt alle eingebauten und benutzerdefinierten Funktionen ein. Wie auch bei allen anderen Listviews, wird der jeweilige Eintrag in deas Formel-String-Gadget eingefügt, wenn er angeklickt wird.

Felder: Zeigt alle Felder der jeweiligen Datenbank an. Das Anklicken eines Felder wird es ein die Formel einfügen.

Konstanten: Zeigt alle Konstanten der Datenbank an. Das Anklicken einer Konstante wird sie auch in die Formel einfügen.

Formel-String-Gadget: Zeigt die Formel an, die gerade bearbeitet wird. Sie können sie von Hand verändern oder mit den Listviews darüber.

Ok: Überprüft die Formel nach Fehlern. Wenn keine gefunden wurden, wird die Formel in den aufrufenden Requester kopiert und der Requester geschlossen.

Cancel: Schließt den Requester ohne weitere Aktion.

10.6.16 Spalte anzeigen-Requester

Mit diesem Requester, der über Liste/Spalte anzeigen erreichbar ist, kann man Listen-Spalten, die vorher mit Liste/Spalte verbergen verborgen wurden, wieder sichtbar machen. Die Spalten werden dann ihrem alten Platz möglichst nahe in die Liste eingefügt.

Feld: Hier werden alle verborgenen Spalten angezeigt, aus denen man eine auswählen sollte.

Ok: Bestätigt den Requester und berechnet die Liste neu, so daß die Spalte wieder zu sehen ist.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.17 Such-Requester

Der Such-Requester ist die wichtigste Schnittstelle zu Fiascos Such-Funktion (siehe Abschnitt 6). Der Requester kann mit dem Menüpunkt Vergl./Suchen geöffnet werden. Es gibt jedoch viele andere Funktionen, die denselben oder einen leicht abweichenden Requester benutzen.

Modus: Wählen Sie hier den Modus aus, den Sie zum Suchen benutzen wollen. Jeder Modus hat eine eigene GUI, die im mittleren Teil des Requesters angezeigt wird. Mit Feldern ist der klassische Modus, in dem für Felder Suchmuster angegeben werden müssen. Mit Formel erlaubt Ihnen, eine Formel zu benutzen, um eine Suche durchzuführen.

Name: Hier können Sie einen Namen angeben, unter dem die aktuellen Such-Kriterien intern gespeichert werden. Mit dem Auswahl-Schalter an der rechten Seite des Gadgets können Sie später diese Kriterien wieder auswählen.

Modus Mit Feldern:

Felder: Diese Liste zeigt alle Felder der aktiven Datenbank an. Das Anklicken eines Feldes wird es in das Suchen nach-Listview legen. Dort können Sie ein Muster für das Feld angeben.

Suchen nach: Hier sind alle Felder aufgelistet, nach denen gesucht wird. Wenn Sie ein Feld hier akitivieren, können Sie ein Suchmuster und andere Parameter für dieses Feld in der Gadget-Gruppe darunter angeben. Um ein Feld in diese Liste einzufügen, müssen Sie es im Felder-Listview anklicken. Um es wieder zu entfernen, aktivieren Sie es und klicken auf Löschen.

Muster: Diese String-Gadget nimmt ein Suchmuster (siehe Abschnitt 6.1.1) für das gerade im Suchen nach-Listview aktive Feld auf. Wenn Sie auf den Auswahl-Button an der rechten Seite des Gadgets klicken, öffnet sich eine Liste mit einigen Werten, die als Muster benutzt werden können. Dies können vordefinierte Werte sein, die Labels eines Cycle-Feldes, etc.

Löschen: Löscht das aktive Feld aus den Suchen nach-Listview.

Unscharfe Suche: Hier können Sie die unscharfe Suche (siehe Abschnitt 6.1.4) anschalten und die Toleranz einstellen. 0 bewirkt, daß die Funktion nur exakt übereinstimmende Einträge akzeptiert, 100 entspricht fast allen Einträgen.

Modus Mit Formel:

Formel: Dieses Gadget nimmt die Formel für die Suche auf. Ein Record wird als Übereinstimmend mit der Formel angesehen, wenn die Formel einen *wahren* Wert zurückgibt, also einen Wert ungleich 0. Der Auswahl-Schalter an der rechten Seite des Gadgets öffnet den Formel-Ändern-Requester (siehe Abschnitt 10.6.15) der beim Erstellen einer Formel hilfreich ist.

Bestätigungs-Gadgets:

Nächstes: Bestätigt den Requester und sucht nach dem nächsten Eintrag, der mit den oben angegebenen Bestimmungen übereinstimmt.

Erstes: Bestätigt den Requester und sucht nach dem ersten Eintrag.

Voriges: Bestätigt den Requester und sucht rückwärts nach dem nächsten Eintrag.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.18 Filter-Requester

Filter (siehe Abschnitt 6.7) bieten die Möglichkeit, eine Übersicht über eine bestimmte Gruppe von Records zu schaffen. Ein Filter erzeugt den Eindruck, als ob ein Projekt nur aus Records bestände, die mit einem bestimmten Suchkriterium übereinstimmen. Fiasco benutzt Indizes (siehe Abschnitt 3.2), um einen Filter zu erstellen. Wie der Sortier-Requester wird der Filter-Requester in zwei Zusammenhängen benutzt.

Wenn Sie den Filter-Requester mit Vergl./Filter öffnen, wird die Filter-Funktion einen gefilterten Index erstellen und aktivieren. Um den Filter auszustellen, müssen Sie den alten Index im Indizes-Requester (siehe Abschnitt 10.6.6) aktivieren. Der erzeugte Index wird auf dem aktiven Index basieren. Deshalb wird er nur Records enthalten, die der aktive Index enthält und mit den Filter-Kriterien übereinstimmen.

Der zweite Weg, den Filter-Requester zu öffnen, ist die Benutzung von Index Filter im Index neu/ändern-Requester (siehe Abschnitt 10.6.7)- Die Filter-Kriterien, die hier eingestellt werden, werden nur zum erstellen des Indexes und wenn Neue Records automatisch hinzufügen aktiv ist und ein Record hinzugefügt wird während ein anderer Index aktiv ist, benutzt.

Filter werden nicht während des normalen Programmablaufs ausgewertet, sondern immer nur aufeinmal erstellt. Dies bedeutet, daß Records, die während ein Filter aktiv ist hinzugefügt werden, auch angezeigt werden, wenn das Suchkriterium nicht übereinstimmt. Das selbe gilt für veränderte Records.

Da Filter Indizes benutzen, die auf einmal erstellt werden, werden Records, die zu einer Datenbank hinzugefügt werden während ein Filter aktiv ist, unabhängig vom Inhalt in den Index aufgenommen. Dasselbe gilt für Änderungen in Records.

Da der Filter-Requester fast vollständig mit dem Such-Requester (siehe Abschnitt 10.6.17) identisch ist, werden hier nur die veränderten oder neuen Gadgets erklärt.

Index: Der Requester enthält dieses Gadget nur, wenn Sie ihn mit Vergl./Filter öffnen. Geben Sie hier den Namen ein, den der neue Index tragen soll. Wenn der Index bereits existiert, wird der Index überschrieben, nachdem Sie eine Warnung bestätigt haben.

Bestätigungs-Gadgets:

Ok: bestätigt die Einstellungen. Wenn der Filter-Requester mit Vergl./Filter geöffnet wurde, wird der Index sofort erstellt.

Abbrechen: schließt den Requester ohne jede weitere Aktion.

10.6.19 Ersetzen-Requester

Die meisten Teile des Ersetzen-Requesters sind mit denen des Such-Requesters (siehe Abschnitt 10.6.17) identisch. Daher werden nur die abweichenden Gadgets in diesem Abschnitt erklärt.

Der Ersetzen-Requester kann mit dem Menüpunkt Vergl./Ersetzen geöffnet werden.

Ersetzen/Felder: Wenn Sie hier ein Feld anklicken, wird es in das Ersetzen Listview eingefügt. Dies ist sehr ähnlich zu der Funktionsweise der Felder und Suchen nach Listviews darüber.

Ersetzen/Ersetzen: Dieses Listview enthält alle Felder, deren Inhalt bei passenden Records ersetzt werden soll. Um ein Feld hier einzufügen, klicken Sie es im Felder Listview an. Um es wieder zu entfernen, wählen Sie es im Ersetzen Listview aus und klicken auf Löschen. Wenn ein Feld in diesem Listview aktiv ist, können Sie die Ersatz-Einstellungen darunter verändern.

Ersetzen/Modus: Benutzen Sie dieses Cycle-Gadget, um entweder Ersetzen durch Wert auszuwählen, um einen festen Wert in das unter Ersetzen ausgewählte Feld einzufügen, oder mit Ersetzen durch Ergebis von Formel das Ergebis vorher mit einer Formel zu berechnen.

Ersetzen/Ersatz: Hier müssen Sie entweder einen festen Wert als Ersatz oder eine Formel angeben. Dies hängt von der Einstellung des Modus Gadgets ab.

Ersetzen/Löschen: Entfernt das ausgewählte Feld im Ersetzen Listview.

Ersetzen/Bestätigen: Wenn Sie wünschen, daß Sie jedesmal gefragt werden, ob ein Eintrag wirklich ersetzt werden soll, sollten sie dieses Gadget aktivieren.

Bestätigungs-Gadgets:

Alle: Startet die Suche und durchsucht alle Records im aktiven Index. Wenn Sie Ersetzen/Bestätigen aktiviert haben, können Sie bei jedem gefundenen Record individuell auswählen, ob Sie den Wert ersetzen wollen, oder nicht. Sie können in diesem Fall sogar die gesamte Suche abbrechen.

Nächstes: Ersetzt nur den Wert im nächsten passenden Record und aktiviert diesen.

Abbrechen: Schließt den Requester ohne jede weitere Aktion.

10.6.20 Zählen-Requester

Dieser Requester erlaubt es Ihnen, die Records zu zählen, die einem bestimmten Suchkriterium entsprechen. Mehr zum Zählen im Kapitel Suchen.

Der Requester ist über den Menüpunkt Vergl./Zählen (siehe Abschnitt 10.4.60) erreichbar.

Da der Zählen-Requester fast vollständig mit dem Such-Requester (siehe Abschnitt 10.6.17) identisch ist, werden hier nur die veränderten oder neuen Gadgets erklärt.

Bestätigungs-Gadgets:

Ok: Bestätigt den Requester und zählt, wie oft das Suchmuster gefunden wird. Das Ergebnis wird zum Schluß angezeigt.

Abbrechen: Schließt den Requester ohne jede weitere Aktion.

10.6.21 Markieren-Requester

Die Markieren-Funktion von Fiasco gibt Ihnen die Möglichkeit, sich einen besonderen Record zu merken. Der Markieren-Requester hat den Sinn, alle Records zu markieren, die auf ein angegebenes Muster passen. Dieser Requester ist stark verwandt mit dem Such-Requester.

Die Markieren-Requester kann mit Vergl./Markieren geöffnet werden.

Da der Markieren-Requester fast vollständig mit dem Such-Requester (siehe Abschnitt 10.6.17) identisch ist, werden hier nur die veränderten oder neuen Gadgets erklärt.

Bestätigungs-Gadgets:

Ok: bestätigt den Requester und markiert die Records. Die alten Markierungen gehen dabei verloren!

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.22 Sortieren-Requester

Der Sotier-Requester kann zum Angeben von Sortier-Kriteria für eine Datenbank benutzt werden. Fiasco benutzt Indizes (siehe Abschnitt 3.2), um die Reihenfolge von Records zu kontrollieren.

Der Sortier-Requester kann auf zwei Wegen geöffnet werden. Einmal können Sie Vergl./Sortieren benutzen. Dieser Menüpunkt erlaubt es Ihnen, eine Datenbank schnell zu sortieren. Die Funktion erzeugt einen neuen Index und aktiviert ihn. Der Name des Index kann in dem Index-Gadget

angegeben werden. Der Index wird auf dem aktiven Index basieren, das bedeutet, daß er nur die Records enthalten wird, die im aktiven Index enthalten sind.

Der andere Zusammenhang, in dem der Sortier-Requester benutzt wird, ist der Index-Requester (siehe Abschnitt 10.6.7). Der Sortier-Requester wird geöffnet, wenn Sie auf das Index Sortierung-Gadget im Index-Requester klicken. Hier können Sie dann die Sortier-Kriteria für das automatische Sortieren eines betimmten Index angeben.

Index: Dieses Gadget existiert nur, wenn der Requester mit Vergl./Sortieren geöffnet wurde. Dies wird der Name für den neu erzeugten Index sein. Wenn der Name bereits existiert, wird der alte Index überschrieben. Fiasco wird Sie jedoch vor dem Überschreiben warnen.

Felder: Diese Liste zeigt alle Felder des aktiven Projektes an. Klicken Sie auf ein Feld, wird es in die Liste Sortieren nach übernommen.

Sortieren nach: Diese Liste zeigt die Felder an, nach denen das Projekt sortiert wird. Das oberste Feld hat die höchste Priorität beim Sortieren. Die meisten Einträge werden nach diesem Feld sortiert; wenn gleiche Einträge auftauchen, werden Sie nach den folgenden Feldern sortiert. Mit Löschen können Felder aus dieser Liste entfernt werden. Die Pfeile dienen zum Bewegen der Felder in der Liste.

Absteigend: Aktivieren Sie diese Option, damit Fiasco die Daten "rückwärts" sortiert, also von großen zu kleinen Werten (z.B. Z, Y, X, ..., C, B, A)

Locale-Einstellungen benutzen: Wenn Sie dieses Gadget aktivieren, wird Fiasco die aktuellen Locale-Einstellungen benutzen, um alphabetische Daten zu sortieren. Dies garantiert die richtige Sortierung von nationalen Zeichen, wie den deutschen Umlauten ä, ö, ü, etc. Beachten Sie jedoch, daß diese Regeln von Land zu Land anders sind. Deshalb kann eine Datenbank, die unter einem Locale-Land sortiert wurde, unter einem anderen Locale-Land falsch sortiert werden.

Ok: bestätigt die Einstellungen. Wenn der Sortier-Requester mit Vergl./Sortieren geöffnet wurde, wird der Index sofort gebaut.

Abbrechen: schließt den Requester ohne irgendetwas anderes noch zu tun.

10.6.23 Datenbank Einstellungen-Requester

Hier können Sie bestimmte Datenbank-Funktionen von Fiasco kontrollieren. Dieser Requester kann mit Einstellungen/Datenbanken geöffnet werden. Er ersetzt einige der Einstellungs-Menüpunkte von Fiasco 1.2.

Relationen/Relationen Schreiben: Wenn diese Option aktiv ist, wird Fiasco Relationen wieder zurück in die "dort" Datenbanken schreiben. Sonst gehen die Änderungen, die in diesen Feldern gemacht wurden, verloren. Diese Option sollte nur aktiv sein, wenn Relationen Aktualisieren auch aktiv ist, oder wenn Sie vor dem Speichern Projekt/Relationen neuladen (siehe Abschnitt 10.4.20) aufrufen. Sonst riskieren Sie, daß Daten im "dort" Projekt von ungültigen Daten einiger Felder im "hier" Projekt überschrieben werden.

Relationen/Relationen Aktualisieren: Hier kann man steuern, ob bei Eingabe eines neuen Schlüsselwortes in ein Relations-Feld die Daten sofort erneuert werden sollen. Dazu sind Diskettenzugriffe nötig, die für Menschen, die keine Festplatte haben, leicht nervig werden können. Falls man diese Option deaktiviert, sollte man auch Relationen Schreiben deaktivieren, damit keine falschen Daten in ein anderes Projekt geschrieben werden. Um die Änderungen so zu betrachten, kann man die Relationen in einem Abwasch über Projekt/Relationen neuladen (siehe Abschnitt 10.4.20) aktualisieren.

Verschiedenes/'*' als Synonym für '#?' benutzen: Aktivieren Sie diese Option, wenn Sie die Unterstützung des Sterns als ein Suchmuster aktivieren wollen. Der * hat dann dieselbe Bedeutung wie #?.

Ok: Schließt den Requester und akzeptiert die Änderungen.

Abbrechen: Schließt den Requester und verwirft die Änderungen.

10.6.24 Benutzeroberflächen Einstellungen-Requester

Dieser Requeste erlaubt es Ihnen, bestimmte Teile von Fiascos GUI zu kontrollieren. Er kann mit Einstellungen/Benutzeroberfläche geöffnet werden. Er ersetzt einige der Einstellungs-Menüpunkte von Fiasco 1.x.

Service-Fenster/Beim Start öffnen: Hier kann man kontrollieren, ob beim Programmstart das Service-Fenster (siehe Abschnitt 10.3) automatisch geöffnet werden soll.

Service-Fenster/Feste Position: Hier kann man kontrollieren, ob beim Öffnen des Service-Fensters eine freie Stelle gesucht werden soll, oder

ob feste Koordinaten benutzt werden sollen. Falls diese Option aktivi ist, speichert Fiasco beim Schreiben der Einstellungen die aktuelle Position des Fensters und aktiviert diese beim Neustart wieder.

ARexx-Skripte/Programm-Start: Ein hier angegebenes ARexx-Skript wird automatisch ausgeführt, nachdem Fiasco gestartet wurde.

ARexx-Skripte/Programm-Ende: Ein hier angegebenes ARexx-Skript wird automatisch ausgeführt, bevor Fiasco beendet wird.

ARexx-Skripte/Datenbank-Spezifische Start-/End-Skripte: Hier können Sie kontrollieren, ob Start- und End-Skripte für Datenbanken ausgeführt werden sollen oder nicht. Solche Skripte können lokal für jede Datenbank in den Projekt-Optionen (siehe Abschnitt 10.6.4) angegeben werden. Automatisch ausführen führt diese Skripte immer aus. Vor dem Ausführen fragen öffnet vorher einen Requester, in dem Ihnen die Wahl gelassen wird, ob das Skript ausgeführt werden soll, oder nicht. Nie Ausführen führt die Skripte nicht aus.

Verschiedenes/Icons erzeugen: Hier wird festgelegt, ob beim Speichern von Fiasco-Projekten Icons erzeugt werden, oder nicht.

Verschiedenes/Sprache: Aktivieren Sie diese Option, wenn Sie wollen, daß Fiasco bestimmte Mitteilungen "spricht".

Verschiedenes/Bei leerem Start nach Datenbank fragen: Wenn aktiv wird Fiasco einen Datei-Requester öffnen, wenn es gestartet wird und keine Datenbanken zum laden angegeben wurden (entweder durch Projekt-Icons oder in der Shell).

Verschiedenes/Sicherheitsabfragen: Hier kann man einstellen, ob vor dem Löschen von Feldern oder Records noch eine Bestätigung benötigt wird, um Datenverlust zu vermeiden.

Verschiedenes/'Enter' springt durch Felder: Aktivieren Sie diese Option, damit Fiasco das nächste Feld aktiviert, wenn eins mit Enter verlassen wurde. Dies ist die Fiasco 1.x-Art. Seit Fiasco 2.0 können Sie die Tab-Taste benutzen, um durch die Felder durchzuspringen.

Verschiedenes/Erstes Feld automatisch aktivieren: Ist diese Option aktiv, wird Fiasco automatisch das erste Feld aktivieren, wenn ein neuer Record aktiviert wird.

Ok: Schließt den Requester und akzeptiert die Änderungen.

Cancel: Schließt den Requester und verwirft die Änderungen.

10.6.25 Benutzermenü-Requester

Fiasco bietet die Möglichkeit, eigene Menüpunkte zu definieren und mit speziellen Funktionen zu belegen. Dabei sind Name und Kommando, das der Menüpunkt ausführen soll, frei wählbar. Die in diesem Requester definierten Menüpunkte sind neben dem "normalen" Weg mit der Maus über die F-Tasten erreichbar. F1 bis F10 entsprechen den ersten zehn Menüpunkten. Shift und F1 bis F10 entsprechen dann den Menüpunkten 11 bis 20. Falls irgendjemand mehr als 20 Menüpunkte definieren sollte, muß sich dann damit abfinden, die verbleibenden Menüpunkte mit der Maus aufrufen zu müssen.

Weiterhin lassen sich nicht mehr als 63 Menüpunkte definieren. Tut man es doch, wird ein Requester erscheinen, der darauf hinweist, daß die Anzahl der Menüpunkte auf 63 zurückgeschraubt wurde.

Die Menüpunkte lassen sich über Einstellungen/Einstellungen Speichern (siehe Abschnitt 10.4.73) dauerhaft sichern.

Items: Hier wird eine Liste aller bereits bestehenden Menü-Items dargestellt. Mit Neu kann man weitere Items hinzufügen, mit Lösch. das aktive Item entfernen und mit den </>-Buttons kann man die Position des aktuellen Items in der Liste verändern.

Typ: Hier kann man einstellen, ob das aktuelle Item ein Programm oder ein ARexx-Script ausführen soll.

Kommando: Hier kann man das Programm bzw. ARexx-Script eingeben, das ausgeführt werden soll.

10.6.26 Anzeige-Requester

Diese Requester kontrolliert die Anzeige-Elemente von Fiasco. Sie können hier einen eigenen Screen für Fiasco öffnen und die Schriften für diesen Screen und für die Maske auswählen.

Bildschirm/Bildschirmtyp: Wählen Sie hier aus, ob Sie einen Public Screen oder einen eigenen Screen benutzen möchten.

Bildschirm/PubScreen Name: Geben Sie hier den Namen des Public Screens an, auf dem Fiasco seine Fenster öffnen soll. Fiasco benutzt diese Angabe nur, wenn Bildschirmtyp auf Public Screen steht. Wenn Sie dieses Gadget leer lassen, wird Fiasco den Standard Public Screen benutzen (normalerweise die Workbench).

Bildschirm/Bildschirm-Modus: Hier können Sie den Anzeige-Modus für den eigenen Screen auswählen. Wenn Sie auf das Popup-Gadget klicken, öffnet sich ein ASL Screenmode-Requester. Dies setzt voraus, daß eine asl.library mit Version 38 oder höher installiert ist.

Bildschirm/Bildschirm Zeichensatz : Diese Gadget kontrolliert, ob Sie eine eigene Schrift oder den Font des Workbench-Screens für einen eigenen Screen benutzen möchten. Die Schrift der Workbench wird durch den Font-Voreinsteller festgelegt.

Bildschirm/Eigener Zeichensatz: Falls Sie einen eigenen Font für den eigenen Bildschirm benutzen möchten, können Sie diesen hier auswählen.

Masken Zeichensatz/Masken Zeichensatz: Dieses Gadget kontrolliert, ob Sie einen eigenen Font oder den System-Default-Font für die Maske benutzen möchten. Der System-Default-Font wird durch den Font-Voreinsteller kontrolliert.

Masken Zeichensatz/Eigener Zeichensatz: Hier können Sie eine eigene Schrift für die Maske auswählen. Die Schrift darf nicht proportional sein, d.h. alle Buchstaben müssen die gleiche Breite haben.

Grafiken/Proportional-Gagdets New-Look: Kontrolliert, ob die Proportional-Gadgets in der Maske, im Standard-Aussehen oder in einem neuen Look gezeichnet werden. Statt der schwarzen Balken, werden weiße Balken mit einem schwarzen Rand rechts und unten benutzt.

Ok: Bestätigt den Requester und baut die Anzeige von Fiasco neu auf.

Abbrechen: Schließt den Requester ohne weitere Aktion.

10.6.27 Externe Programme und Pfade Requester

Dieser Requester kann zum Einstellen der externen Programme, die von Fiasco gestartet werden können, und der externen Pfade benutzt werden.

Programme/Editor-Programm: Geben Sie hier Ihr bevorzugtes Editor-Programm an. Sie können ein `%s` angeben, das durch den Dateinamen der zu verändernden Datei ersetzt wird. Das Programm sollte synchron starten. Bei Programmen wie GED, die normalerweise asynchron starten, sollten Sie die jeweiligen Argumente angeben, damit das Programm nicht asynchron started. Bei GED wäre dies das Argument `sticky`.

Programme/Editor-Programm/Stack: Geben Sie hier den Stapelspeicher für das Editor-Programm an. Der Wert sollte normalerweise mindestens 4000 Bytes sein.

Programme/Text-Anzeige-Programm: Nimmt Ihr Text-Anzeige-Programm auf. Dies kann z.B. More, Most oder MultiView sein.

Programme/Text-Anzeige-Programm/Stack: Geben Sie hier den Stapelspeicher für das Editor-Programm an. Der Wert sollte normalerweise mindestens 4000 Bytes sein.

Pfade/Fiasco-Pfad für Icons: Der hier angegebene Pfad wird als Standardprogramm in den Fiasco-Datenbank-Icons gespeichert. So wird beim Öffnen einer Fiasco-Datenbank über ihr Icon dieses Programm gestartet. Beispiel: Work:Fiasco/Fiasco.

Pfade/Standard-Datenbank Verzeichnis: Geben Sie hier das Standard-Verzeichnis für Datenbanken an. Der Datei-Requester wird nach Datenbanken relativ zu diesem Verzeichnis suchen.

Ok: Schließt den Requester und akzeptiert die Änderungen die gemacht wurden.

Abbrechen: Schließte den Requester und ignoriert die Veränderungen.

10.6.28 Druckoptionen-Requester

Der Druckoptionen-Requester kann mit Projekt/Optionen im Druck-Fenster geöffnet werden. Sie können ein paar Optionen für das Drucken hier kontrollieren. Der Abschnitt Drucken enthält mehr Informationen über Drucken.

Die hier gemachten Einstellungen können mit den Menüpunkten Projekt/Speichern und Projekt/Speichern als des Druckfensters gespeichert werden.

Drucke nach: Die Druck-Funktion von Fiasco schreibt die Daten in diese Datei. Wenn Sie normal drucken wollen, sollten Sie hier PRT: für den Drucker angeben.

Mit ARexx drucken: Wenn dieses Gadget aktiv ist, ruft Fiasco nach dem Drucken das Script PROGDIR:ARexx/ARexxPrint.rexx auf. Fiasco wird das Script mit dem Dateinamen aus Drucke nach als Argument aufrufen. In einer Standard-Fiasco-Installation ruft dieses Script TeX auf, um die Datei in eine DVI-Datei zu übersetzen und auszudrucken. Sie können jedoch das Script so ändern, daß es etwas völlig anderes

macht. Wenn Sie Mit ARexx drucken benutzen dürfen Sie in Drucke nach nicht PRT: angeben. Eine temporäre Datei, z.B. T:FiascoPrint, ist das Beste.

Nur markierte Records: Wenn Sie dieses Gadget aktivieren druckt Fiasco nur markierte Records aus.

10.6.29 Druckelement-Requester

Sie können die Eigenschaften eines Druck-Elements im Druckfenster mit diesem Requester beeinflussen. Der Requester erscheint wenn Sie ein Element mit Element/Hinzufügen hinzufügen oder mit Element/Ändern ändern. Der Aufbau des Requesters hängt vom unter Element/Typ ausgewählten Element-Typ ab.

Abschnitt 7 enthält mehr Informationen über Elemente.

Feld-Elemente:

Feld: Wählen Sie hier das Feld aus, dessen Daten an der Position des Druck-Elements eingefügt werden.

Breite: Die Breite des Druck-Elements in der Druck-Maske.

Höhe: Die Höhe des Druck-Elements in der Druck-Maske. Nur verfügbar bei Var String oder Listview Feldern.

Abschneiden: Wenn dieser Schalter aktiv ist, werden die Daten des Feldes abgeschnitten, wenn sie größer als die verfügbare Größe des Elementes werden. Wenn der Schalter nicht aktiv ist, wird in diesem Fall das Element vergrößert und andere Daten nach rechts bzw. – wenn das Element mehr vertikalen Raum braucht – nach unten verschoben.

Stil/Fett: Wenn aktiv, werden die Feld-Daten fett gedruckt.

Stil/Kursiv: Wenn aktiv, werden die Feld-Daten kursiv gedruckt.

Stil/Unterstrichen: Wenn aktiv, werden die Feld-Daten unterstrichen gedruckt. Beachten Sie: Der Effekt von Fett, Kursiv und Unterstrichen wird durch Escape-Sequenzen erzeugt. Er ist nicht mit Drucken mit TeX kompatibel.

Stil/Ausrichtung: Hier können Sie auswählen, ob der Inhalt des Elements nach links oder rechts ausgerichtet wird, oder im Element zentriert wird. Standard: Links. Wenn Abschneiden nicht aktiv ist, kann Fiasco auf Links-Ausrichtung ausweichen, wenn der Feld-Inhalt zu groß wird.

Kapitel 11

Formeln

Seit Release 2.1 unterstützt Fiasco Formeln. Formeln können benutzt werden, um den Inhalt von Feldern automatisch zu berechnen oder um komplexe Suchen durchzuführen. Formeln für Felder können im Feld-Requester (siehe Abschnitt 10.6.11) angegeben werden, Formeln für Suchen im Such-Requester (siehe Abschnitt 10.6.17). Von beiden Requestern haben Sie Zugriff auf den Formel Requester (siehe Abschnitt 10.6.15), der das Erstellen und Ändern von Formeln sehr einfach macht.

Eine Formel kann so aussehen:

floor(preis * mwst / 100) + preis

Die Teile aus denen die Formel aufgebaut ist werden in den folgenden Abschnitten beschrieben.

11.1 Elemente von Fiasco-Formeln

11.1.1 Konstante Werte

Konstante Werte werden direkt in der Formel definiert. In der obigen Formel ist die 100 eine Definition eines konstanten numerischen Werts. Sie können auch Strings als konstante Werte definierten, indem Sie ihn mit Anführungszeichen umschließen. Beispiel: "Test-String".

In Fiasco-Formeln sind Strings und Zahlen austauschbar. Wenn eine Zahl benötigt wird, ein String aber nur zur Verfügung steht, wird der String zu einer Zahl konvertiert. Wenn ein String benötigt wird und nur eine Zahl zur Verfügung steht, funktioniert das ganze umgekehrt.

Die Regeln für konstante Werte treffen auch für die im Folgenden beschriebenen Elemente von Formeln zu, die Werte "zurückgeben". Bildlich gesprochen, wird ein Element einer Formel, das einen Wert während der

Berechnung zurückgegeben hat, durch diesen Wert ersetzt, bevor weitergerechnet wird.

So gesehen, ist das endgültige Ergebnis eines Formel der Wert, der endgültig zurückgegeben wurde.

Boolsche Werte

Die Boolschen Werte (Wahrheits-Werte, die bei logischen Operationen benutzt werden) "Wahr" und "Falsch" werden in Fiasco-Formeln durch Zahlen repräsentiert. Zahlen ungleich Null repräsentieren einen wahren Wert, Null selbst repräsentiert einen falschen Wert.

11.1.2 Felder

Dies sind Verweise auf Felder in der Datenbank. Diese Verweise werden während der Berechnung durch den Inhalt des Feldes ersetzt. Wenn der Inhalt eines Feldes geändert wird, das in der Formel eines anderen Feldes referenziert wird, wird diese Formel automatisch neu berechnet. Felder in Fiasco-Formeln werden wie Variablen in normalen, mathematischen Formeln benutzt. Ein Feld im Beispiel oben ist preis.

Die Inhalte der Felder mit den Typen

- String
- Datum
- Zeit
- Extern
- Datatypes
- Var String

werden als Strings zurückgegeben. Die Inhalte der übrigen Feldtypen werden als Zahlen zurückgegeben. Cycle-Felder geben die Zahl des ausgewählten Labels zurück und Boolean-Felder geben eine 1 für ein ausgewähltes Feld und eine 0 für ein nicht ausgewähltes Feld zurück.

Felder müssen mit ihren IDs in der Formel angegeben werden. Um dort als Felder erkannt werden zu können, dürfen die Feld-IDs nicht mit Zahlen beginnen und dürfen keine Leerzeichen beinhalten.

Die Elemente von Listview-Feldern können mit eckigen Klammern nach der Feld-ID angesteuert werden. Diese Klammern müssen die Nummer des zu lesenden Elements beinhalten. Die Nummer hat die Basis 0, das heißt,

daß der erste Eintrag die Index-Nummer 0, der zweite 1, etc. hat. NamenListe[5] repräsentiert beispielsweise den sechsten Eintrag in dem Listview-Feld mit der ID NamenListe.

11.1.3 Konstanten

Konstanten sehen in Fiasco-Formeln genau so wie Felder aus. Der Unterschied liegt darin, daß Konstanten global definierte Werte haben, die für die gesamte Datenbank gleich sind. Der Konstanten-Requester (siehe Abschnitt 10.6.9) kann zum Definierten neuer Konstanten benutzt werden. mwst könnte eine Konstante in dem Beispiel sein.

11.1.4 Operatoren

Operatoren sind die Symbole, die die wichtigsten mathematischen oder logischen Operationen darstellen. Ein Operator benötigt normalerweise zwei Operanden die auf beiden Seiten des Operators angegeben werden müssen. Manche Operatoren benötigen nur ein oder auch drei Operanden. Operatoren im obigen Beispiel sind *, / und +.

Weiterhin haben Operatoren in Fiasco-Formeln Auswertungs-Prioritäten. Operationen mit Operatoren von höherer Priorität werden vor Operationen mit Operatoren von niedrigerer Priorität ausgewertet. Diese Prioritäten ermöglichen beispielsweise die Einhaltung der mathematischen Regel, daß Punktrechung vor Strichrechung geht.

Die Tabelle enthält die Operatoren, die zur Zeit von Fiasco unterstützt werden.

Pri	Op.	Beschr.	Syntax	Ergebnis
10	!	Logisches nicht	! boolean	boolean
9	^	Potenzieren	Zahl ^ Exponent	Zahl
8	*	Multiplikation	4 * 4	Zahl
	/	Division	10 / 2	Zahl
7	+	Addition	2 + 3	Zahl
	-	Substraktion	5 - 1	Zahl
6	<	Kleiner als	Wert1 <Wert2	boolean
	>	Größer als	Wert1 >Wert2	boolean
	<=	Kleiner od. gleich	Wert1 <= Wert2	boolean
	>=	Größer od. gleich	Wert1 >= Wert2	boolean
5	==	Gleich	Wert1 == Wert2	boolean
	!=	Ungleich	Wert1 != Wert2	boolean
4	&&	Logisches Und	bool && bool	boolean
	||	Logisches Oder	bool || bool	boolean
3	? :	Bedingung (siehe unten)	bool ? Ausdr. : Ausdr.	Erg. e. Ausdr.
2	numentries()	Anzahl d. Eintr. in Lv.	numentries(fieldid)	Ganze Zahl
	active()	Akt. Eintr. e. Listv.	active(fieldid)	Ganze Zahl
	sum()	Summe e. Listv. (s. unten)	sum(fieldid[,start][,end])	Zahl
	current()	Aktueller LV-Eintrag (s. unten)	current()	Ganze Zahl

Bedingungs-Operator

Der Bedingungs-Operator ist eine kleinere Version des if-else-Konstrukts in normalen Programmiersprachen. Wenn Sie sich mit der Programmiersprache "C" auskennen, sollten Sie bereits mit diesem Operator vertraut sein.

Der Operand vor dem Fragezeichen sollte einen boolschen Wert ergeben. Wenn der Wert wahr ist (also ungleich null) wird der Operator durch das Ergebnis des Ausdrucks nach dem ? und vor dem : ersetzt. Wenn der boolsche Wert falsch ist (also null) wird der Ausdruck nach dem : benutzt.

Eine Formel, die die größere zweier Zahlen zurückgibt könnte so aussehen:

a >b ? a : b

Natürlich können Sie auch mehrere Bedingungs-Operatoren in einer Formel benutzen. Eine Formel, die das Vorzeichen einer Zahl bestimmt, könnte so aussehen:

n >0 ? "Positiv" : n <0 ? "Negativ" : "Null"

Wenn n größer als 0 ist, wird der String Positiv zurückgegeben. Wenn dies nicht wahr ist, wird überprüft, ob n kleiner 0 ist. Wenn dies wahr ist, wird der String Negativ zurückgegeben. Wenn dies auch nicht wahr ist, muß die Zahl Null sein.

Listview-Summen-Operator

Der Operator sum() kann zum Berechnen der Summe der Einträge eines Listview-Feldes benutzt werden. Unabhängig vom Typ des Feldes wird Fiasco versuchen, die Daten als Zahlen aufzusummieren. Der einfache Aufruf von sum() benötigt nur die Feld-ID des zu summierenden Feldes, beispielsweise also sum(Preise). Sie können jedoch auch in einem zweiten Argument den Eintrag angeben, bei dem erst mit dem Summieren begonnen werden soll und in einem dritten Argument den Eintrag, an dem die Berechnung beendet werden soll. Die Nummern der Einträge müssen – wie auch beim Zugreifen von einzelnen Listview-Einträgen mit eckigen Klammern – auf Basis Null angeben werden. Daher, ist der erste Eintrag Eintrag 0, usw. Beispielsweise addiert die Formel sum(Preise, 2, 4) den dritten, vierten und fünften Eintrag des Listviews Preise.

Aktueller-Eintrag-Operator

Der Operator current() kann nur innerhalb von Formeln benutzt werden, die zum Berechnen von Werten von Listview-Feldern benutzt werden. Der Operator gibt die Nummer des Eintrages zurück, der gerade von dieser

Formel berechnet wird. Diese Nummer ist auf Basis Null. Daher wird current() 0 zurückgeben, wenn gerade der erste Eintrag eines Listviews gerade berechnet wird.

Auf diese Weise können Sie einen Wert über mehrere gruppierte Listview-Felder berechnen. Ein Beispiel ist eine Rechnung, in der der Einzelpreis und die Anzahl der bestellten Artikel angegeben sind. Der Gesamtpreis für einen bestimmten Artikel berechnet sich dann so:

einzelpreis[current()] * anzahl[current()]

11.1.5 Funktionen

Funktionen werden von dem Funktions-Namen und den Argumenten für die Funktion, eingeschlossen in runden Klammern, definiert. Wenn eine Funktion keine Argumente braucht, muß trotzdem ein leeres Klammerpaar angegeben werden. Eine Funktion kann einige Argumente annehmen, etwas damit berechnen und einen neuen Wert zurückgeben. Dieser Wert wird für die weiteren Berechnungen benutzt. Im obigen Beispiel ist floor() eine Funktion. Die Funktion hat nur ein Argument, das das Ergebnis der Berechnung preis * mwst / 100 ist. Eine Liste aller eingebauten Funktionen von Fiasco befindet sich in Abschnitt 11.2.

Neben den eingebauten Funktionen von Fiasco kann der Benutzer auch eigene Funktionen definieren, die auf anderen Formeln aufbauen. So können Sie eine Funktion mit dem Namen

min(a, b)

definieren, die folgendes tut:

a ¡ b ? a : b

Diese Funktion bestimmt also, welche Zahl die kleinere ist und gibt sie zurück. Um diese Funktion in allen Formeln der jeweiligen Datenbank benutzen zu können, muß sie im Funktions-Requester (siehe Abschnitt 10.6.8) definiert werden.

Da Aufrufe von Benutzer-definierten Funktionen intern vor Aufrufe von eingebauten Funktionen verarbeitet werden, können Sie eine eingebaute Funktion durch eine Benutzer-definierte mit dem selben Namen ersetzen.

Wenn Sie andere Namen für die eingebauten Funktionen vorziehen würden, können Sie auch einfach einen Aufruf mit einer Benutzer-definierte Funktion umlenken. Wenn Sie eine Funktion mit dem Namen

laenge(a)

definieren, dies zu tun:

strlen(a)

können Sie ab sofort eine Funktion mit dem Namen laenge(a) benutzen, die dieselbe Funktion wie strlen() hat.

Wichtig: Die aktuelle Implementation von Funktionen erlaubt nicht die Benutzung rekursiver Programmierung. Versuchen Sie es besser nicht!

11.2 Funktions-Referenz

Fiasco hat zur Zeit diese eingebauten Funktionen:

11.2.1 abs()

Name: abs() - Betrag eines Werts

Synopsis: abs(wert)

Funktion: Rechnet den Betrag eines Werts aus, entfernt also das Vorzeichen.

Argumente: wert - Zahl

Ergebnis: Eine positive Zahl

11.2.2 activerecord()

Name: activerecord() - Die Nummer des aktiven Records **(8)**

Synopsis: activerecord()

Funktion: Gibt die Nummer des aktiven Records zurück. Beachten Sie jedoch, daß der aktive Record nicht unbedingt der Record des Felds sein muß, dessen Wert von der Formel berechnet wird.

Argumente: Keine

Ergebis: Eine positive Zahl

11.2.3 asin()

Name: asin() - Arcussinus-Funktion

Synopsis: asin(wert)

Funktion: Berechnet den Arcussinus eines Werts und gibt den Winkel als Radiant zurück (Vielfaches von pi).

Argumente: wert - zwischen 1 und -1

Ergebnis: Ein Winkel zwischen -pi/2 und pi/2

11.2.4 acos()

Name: acos() - Arcuscosinus-Funktion

Synopsis: acos(wert)

Funktion: Berechnet den Arcuscosinus eines Werts und gibt den Winkel als Radiant zurück (Vielfaches von pi).

Argumente: wert - zwischen 1 und -1

Ergebnis: Ein Winkel zwischen -pi/2 und pi/2

11.2.5 atan()

Name: atan() - Arcustangens-Funktion

Synopsis: atan(wert)

Funktion: Berechnet den Arcustangens eines Werts und gibt den Winkel als Radiant zurück (Vielfaches von pi).

Argumente: wert

Ergebnis: Ein Winkel zwischen -pi und pi

11.2.6 ceil()

Name: ceil() - Runden

Synopsis: ceil(wert)

Funktion: Gibt die kleinste ganze Zahl zurück, die nicht kleiner als die angegebene Zahl ist.

Argumente: wert - Eine reelle Zahl

Ergebnis: Eine ganze Zahl

11.2.7 cos()

Name: cos() - Cosinus-Funktion

Synopsis: cos(winkel)

Funktion: Berechnet den Cosinus eines als Radiant angegeben Winkels (Vielfaches von pi).

Argumente: winkel - Winkel als Radiant.

Ergebnis: Der Cosinus des Winkels. Zwischen -1 und 1.

11.2.8 currentdate()

Name: currentdate() - Lese das aktuelle Datum

Synopsis: currendate()

Funktion: Gibt das aktuelle Datum als String im Format der Locale-Einstellungen zurück.

Argumente: Keine

Ergebnis: Ein Datums-String

11.2.9 currenttime()

Name: currenttime() - Lese die aktuelle Zeit

Synopsis: currentime()

Funktion: Gibt die aktuelle Zeit als String im Format der Locale-Einstellungen zurück.

Argumente: Keine

Ergebnis: Ein Zeit-String

11.2.10 datediff()

Name: datediff() - Berechne die Zeit zwischen zwei Daten

Synopsis: datediff(datum1, datum2)

Funktion: Berechnet die Zeitspanne in Tagen zwischen den zwei Datumsangaben. Wenn eine Datumsangabe ein nicht spezifiziertes Element enthält (als ?? angezeigt), wird das Element des anderen Datums benutzt. Es wird grundsätzlich die Zeit berechnet, die datum2 nach datum1 liegt. Wenn datum2 vor datum1 liegt, wird ein negatives Ergebnis zurückgegeben.

Argumente: datum1, datum2 - Zwei Datums-Strings

Ergebnis: Anzahl an Tagen zwischen den beiden Datumsangaben.

11.2.11 day()

Name: day() - Hole den Tag eines Datums

Synopsis: day(datumsstring)

Funktion: Zerlegt einen Datums-String im aktuellen Locale-Format und gibt den Tag des Datums zurück.

Argumente: datumsstring - Ein Datums-String

Ergebnis: Der Tag des Datums-Strings

11.2.12 floor()

Name: floor() - Runden

Synopsis: floor(wert)

Funktion: Gibt die größte ganze Zahl zurück, die nicht größer als oder gleich der angegebenen Zahl ist.

Argumente: Wert - Reelle Zahl

Ergebnis: Ganze Zahl

11.2.13 formatdate()

Name: formatdate() - Erzeuge einen Datums-String

Synopsis: formatdate(jahr, monat, tag)

Funktion: Formatiert das angegebene Datum nach den Angaben der aktuellen Locale-Einstellungen. Somit wird ein Fiasco-Datums-String erzeugt. Alle anderen Datums-Funktionen von Fiasco benutzen dieses Format.

Argumente: jahr, monat, tag - Die Elemente des Datums

Ergebnis: Ein formatierter Datums-String.

11.2.14 formattime()

Name: formattime() - Erzeuge einen Zeit-String

Synopsis: formattime(stunde, minute, sekunde)

Funktion: Formatiert die angegebene Zeit nach den Angaben der aktuellen Locale-Einstellungen. Somit wird ein Fiasco-Zeit-String erzeugt. Alle anderen Zeit-Funktionen von Fiasco benutzen dieses Format.

Argumente: stunde, minute, sekunde - Die Elemente der Zeit

Ergebnis: Ein formatierter Zeit-String.

11.2.15 hour()

Name: hour() - Hole die Stunde einer Zeit

Synopsis: hour(zeitstring)

Funktion: Zerlegt einen Zeit-String im Format der Locale-Einstellungen und gibt die Stunde davon zurück.

Argumente: zeitstring - Ein Zeit-String

Results: Die Stunde im 24-Stunden-Format.

11.2.16 left()

Name: left() - Hole den linken Teil eines Strings

Synopsis: left(string, laenge)

Funktion: Gibt den linken Teil eines Strings mit der angegebenen Länge zurück. Wenn die Länge größer als der String selbst ist, wird der ganze String zurückgegeben.

Argumente: string - Zu verarbeitender String
length - Länge des Teils, der zurückgegeben wird.

Ergebnis: Ein Teilstring

11.2.17 lg()

Name: lg() - Basis 10 Logarithmus

Synopsis: lg(wert)

Funktion: Gibt den Logarithmus auf der Basis 10 zurück.

Argumente: wert - Eine positive reelle Zahl.

Ergebnis: Eine reelle Zahl

11.2.18 ln()

Name: ln() - Logarithmus Naturalis

Synopsis: ln(wert)

Funktion: Gibt den Logarithmus der Basis e zurück.

Argumente: wert - Eine positive reelle Zahl.

Ergebnis: Eine reelle Zahl.

11.2.19 minute()

Name: minute() - Minute einer Zeit

Synopsis: minute(zeitstring)

Funktion: Zerlegt einen Zeit-String im Format der Locale-Einstellungen und gibt die Minute davon zurück.

Argumente: zeitstring - Ein Zeit-String

Results: Eine ganze Zahl

11.2.20 month()

Name: month() - Monat eines Datums

Synopsis: month(datumsstring)

Funktion: Zerlegt einen Datums-String im aktuellen Locale-Format und gibt den Monat des Datums zurück.

Argumente: datumsstring - Ein Datums-String

Ergebnis: Der Monat des Datums-Strings

11.2.21 numrecords()

Name: numrecords() - Anzahl von Records **(8)**

Synopsis: numrecords()

Funktion: Gibt die Anzahl von Records im aktiven Index der Datenbank zurück.

Argumente: Keine.

Result: Die Anzahl der Records. >= 0

11.2.22 printf()

Name: printf() - Erzeuge einen formatierten String

Synopsis: printf(formatstring, ...)

Funktion: Formatiert die Daten die als Argumente angegeben wurden nach den Regeln, die im formatstring angegeben wurden. Der Formatstring ist ein String, der Kontroll-Sequenzen enthalten darf, die durch die formatierten Argumente ersetzt werden. Alle Kontroll-Sequenzen beginnen mit einem Prozent-Zeichen. Eine Kontroll-Sequenz hat das Format (Felder in Klammern sind optional):
%[flags][width][.precision]type
Die Argumente werden mittels der Reihenfolge den Kontroll-Sequenzen zugeordnet.

Flags: Kann eins der Folgenden sein:
- Das Ergebnis wird in width linksbündig ausgegeben
+ Bewirkt, das ein Plus-Zeichen vor einer positiven Zahl ausgegeben wird
0 Die Breite des Feldes wird mit Nullen gefüllt wenn es eine Zahl ist

Width: ist die minimale Anzahl von Zeichen, die von diesem Element benutzt werden sollen.

Precision: Eine ganze Zahl. Die Präzision der Ausgabe, abhängig vom Typ der Kontroll-Sequenz:
d,u,o,x,X - Minimale Anzahl von Ziffern
e,f - Ziffern nach dem Dezimalpunkt
g - Anzahl der signifikanten Ziffern
s - Maximale Länge des zu kopiereden Strings

Type: Kann eins der folgenden sein:
c - Ein Zeichen des angegebenen ASCII-Wertes wird erzeugt.
d - Eine vorzeichenbehaftete ganze Zahl wird erzeugt.
u - Eine vorzeichenlose ganze Zahl wird erzeugt.
o - Eine oktale Zahl wird erzeugt.
x - Eine hexadezimale Zahl wird erzeugt. Die Zeichen a-f werden klein geschrieben.
X - Eine hexadezimale Zahl wird erzeugt. Die Zeichen a-f werden groß geschrieben.
e - Eine gebrochene Zahl im Format d.dde-ddd wird erzeugt.
f - Eine gebrochene Zahl im Format dd.dd wird erzeugt.
g - Eine gebrochene Zahl wird erzeugt. Das Format wird abhängig von der Höhe der Zahl gewählt.
s - Ein String wird eingefügt.

Argumente: Der Formatstring und die weiteren Argumente.

Ergebnis: Ein formatierter String.

11.2.23 rand()

Name: rand() - Zufallszahlengenerator

Synopsis: rand()

Functions: Erzeugt eine Pseudo-Zufallszahl.

Argumente: keine

Ergebnis: Eine gebrochene Zahl 0.0 <= r < 1.0.

11.2.24 right()

Name: right() - Hole rechten Teil eines Strings

Synopsis: right(string, laenge)

Funktion: Gibt den rechten Teil eines Strings mit der angegebenen Länge zurück. Wenn die Länge größer als die wirkliche Länge eines Strings ist, wird der ganze String zurückgegeben.

Argumente: string - Zu verarbeitender String
length - Länge des zurückzugebenden Teils.

Ergebnis: Ein Teilstring

11.2.25 round()

Name: round() - Runden

Synopsis: round(wert)

Funktion: Gibt eine mathematische Rundung von wert zurück.

Argumente: wert - Reelle Zahl

Ergebnis: Ganze Zahl

11.2.26 second()

Name: second() - Sekunde einer Zeit

Synopsis: second(zeitstring)

Funktion: Zerlegt einen Zeit-String im Format der Locale-Einstellungen und gibt die Sekunde davon zurück.

Argumente: zeitstring - Ein Zeit-String

Results: Eine ganze Zahl

11.2.27 sign()

Name: sign() - Vorzeichen-Test

Synopsis: sign(wert)

Funktion: Ermittelt das Vorzeichen einer Zahl

Argumente: wert - Eine Zahl

Ergebnis: -1 wenn wert negativ ist, 1 wenn positiv, 0 wenn Null.

11.2.28 sin()

Name: sin() - Sinus-Funktion

Synopsis: sin(winkel)

Funktion: Berechnet den Sinus eines als Radiant angegeben Winkels (Vielfaches von pi).

Argumente: winkel - Radiant-Winkel.

Ergebnis: Der Sinus des Winkels. Zwischen -1 und 1

11.2.29 sqrt()

Name: sqrt() - Quadratwurzel

Synopsis: sqrt(wert)

Funktion: Gibt die Quadratwurzel des angegebenen Zahl zurück.

Argumente: wert - Muß >= 0 sein.

Ergebnis: Eine positive Zahl

11.2.30 strcat()

Name: strcat() - Verkette Strings

Synopsis: strcat(string1, string2, ...)

Funktion: Verkettet zwei oder mehr Strings zu einem String, der zurückgegeben ist.

Argumente: string1 ... stringn - Die zu verkettenden Strings. Mindestens zwei, kein Maximum.

Ergebnis: Der verkettete String.

11.2.31 strcmp()

Name: strcmp() - Vergleiche Strings

Synopsis: strcmp(string1, string2)

Funktion: Überprüft, ob die angegebenen Strings gleich oder ungleich sind. Beim Vergleich wird die Groß-/Kleinschreibung beachtet, d.h. auch diese muß gleich sein, damit die beiden Strings als gleich angesehen werden. Vergleiche unabhängig von der Groß-/Kleinschreibung können mit stricmp (siehe Abschnitt 11.2.32) gemacht werden.

Argumente: string1, string2 - Zu vergleichende Strings

Ergebnis: 0 wenn beide Strings gleich sind, -1 wenn string1 < string2 und +1 wenn string1 > string2

11.2.32 stricmp()

Name: stricmp() - Vergleiche zwei Strings

Synopsis: stricmp(string1, string2)

Funktion: Überprüft die Strings auf Gleichheit. Beim Vergleich wird nicht auf Groß-/Kleinschreibung geachtet. strcmp (siehe Abschnitt 11.2.32) tut dies hingegen.

Argumente: string1, string2 - Zu vergleichende Strings

Ergebnis: 0 wenn beide Strings gleich sind, -1 wenn string1 $<$ string2 und $+1$ wenn string1 $>$ string2

11.2.33 strlen()

Name: strlen() - Länge eines Strings.

Synopsis: streln(string)

Funktion: Gibt die Länge eines Strings in Zeichen zurück.

Argumente: string - Zu messender String.

Ergebnis: Länge des Strings in Zeichen.

11.2.34 strmid()

Name: strmid() - Kopiere einen Teil eines Strings

Synopsis: strmid(string, start, laenge)

Funktion: Gibt den angegebenen Abschnitt im angegebenen String zurück.

Argumente: string - Zu verarbeitender String
start - Start des Teilstrings, zählend ab 0.
length - Länge des Teilstrings.

Ergebnis: Teilstring des angegebenen Strings

11.2.35 strrev()

Name: strrev() - String umkehren

Synopsis: strrev(string)

Funktion: Gibt einen String zurück, dessen Zeichen die umgekehrte Reihenfolge als das Argument haben. strrev(strrev(string)) hat effektiv keine Wirkung.

Argumente: string - Umzukehrender String

Ergebnis: Umgekehrter String

11.2.36 strstr()

Name: strstr() - Suche nach einem String

Synopsis: strstr(string, substring)

Funktion: Sucht nach substring in string und gibt die Position zurück, an der der String gefunden wurde.

Argumente: string - Zu durchsuchender String
substring - String nach dem gesucht wird.

Ergebnis: Position an der substring gefunden wurde, 0 entspricht dem ersten Zeichen. Gibt -1 zurück, wenn der String nicht gefunden wurde.

11.2.37 tan()

Name: tan() - Tangens-Funktion

Synopsis: tan(winkel)

Funktion: Berechnet den Tangens eines als Radiant angegeben Winkels (Vielfaches von pi).

Argumente: winkel - Radiant-Winkel.

Ergebnis: Der Tanges des Winkels.

11.2.38 tolower()

Name: tolower() - Wandlung in Kleinschreibung

Synopsis: tolower(string)

Funktion: Wandelt alle Buchstaben im angegebenen String in Kleinbuchstaben.

Argumente: Zu wandelnder String

Ergebnis: String nur mit Kleinbuchstaben

11.2.39 toupper()

Name: toupper() - Wandlung in Großschreibung

Synopsis: toupper(string)

Funktion: Wandelt alle Buchstaben im angegebenen String in Großbuchstaben.

Argumente: Zu wandelnder String

Ergebnis: String nur mit Großbuchstaben

11.2.40 version()

Name: version() - Fiasco-Version

Synopsis: version()

Funktion: Gibt Fiascos interne Version zurück. Für Fiasco 2.2 ist dies 8.

Argumente: Keine

Ergebnis: Fiascos interne Version als ganze Zahl

11.2.41 year()

Name: year() - Jahr eines Datums

Synopsis: year(datumsstring)

Funktion: Zerlegt einen Datums-String im aktuellen Locale-Format und gibt das Jahr des Datums zurück.

Argumente: datumsstring - Ein Datums-String

Ergebnis: Das Jahr des Datums-String.

Kapitel 12

Der ARexx-Port

ARexx ist eine Macro-Sprache mit der Fähigkeit, Programme untereinander kommunizieren zu lassen. ARexx wurde von William S. Hawes entwickelt und befindet sich im Lieferumfang von Amiga OS 2.0 oder höher.

Der ARexx-Port von Fiasco kann entweder aus einem Script unabhängig vom Programm angesprochen werden, oder Fiasco ruft selber Scripts auf. Dies geschieht beispielsweise, wenn bei einem Button ein ARexx-Script angegeben wurde und der Button angeklickt wurde. Dieses Script hat dann die Möglichkeit irgendwas zu tun.

Für Fiasco 2.1 wurde Fiascos ARexx-Port komplett überarbeitet, um leistungsfähiger zu sein und Style-Guide-Konform zu sein. ARexx-Scripts, die für Fiasco 1.x oder 2.0x geschrieben wurden, werden dennoch weiter arbeiten. "Alte" Scripts können an dem Kommando Address FIASCO am Anfang des Scripts und an den Kommandos mit dem Präfix F_ erkannt werden. Die alten Kommandos sind in dieser Version der Anleitung nicht mehr enthalten. Um mehr über die Kommandos zu erfahren, ziehen Sie die 2.0x-Versionen der Anleitung zu rate. Bitte beachten Sie auch, daß Sie keine alten und neuen Scripts mischen können. Fiasco-ARexx-Scripts müssen entweder vollständig dem 1.x oder dem 2.1-Standard gehorchen.

Scripts, die im neuen Standard geschrieben wurden, steuern den Port mit dem Namen FIASCO nicht an. Der neue ARexx-Port weist jeder Datenbank einen eigenen Namen zu. Die Namen dieser Ports sind FIASCO.n, wobei n eine Zahl ist. Wenn das Script direkt von Fiasco gestartet wurde (z.B. durch Anklicken eines Buttons wird das Script automatisch auf den ARexx-Port der entsprechenden Datenbank gesetzt. Deshalb ist kein zusätzliches Address-Kommando nötig. Mehr über das Zugreifen auf Fiascos ARexx-Port im Abschnitt 12.2.

Fast alle Operationen, die über die GUI von Fiasco ausgeführt wer-

den können, können auch mit ARexx-Scripts erledigt werden. Außerdem kann Fiasco mit ARexx fast beliebig an Funktionen erweitert werden. Viele ARexx Kommandos tun genau das selbe wie die Menüpunkte. Das heißt, daß sie unter Umständen einen Requester öffnen. Es ist aber meistens möglich, dies zu umgehen. Weiterhin gibt es Befehle, die immer einen Requester öffnen.

12.1 Stil-Konventionen

Dieser Abschnitt beschreibt einige stilistische Vereinbarungen für Fiasco-ARexx-Scripts.

ARexx erlaubt, einen eigenen Suffix für Programm-Spezifische Scripts zu wählen. Für Fiasco-2.1-ARexx-Scripts sollte der Suffix .frx statt .rexx benutzt werden.

Fiasco ARexx-Scripts sollten immer Fiascos GUI mit dem Kommando LockGUI verschließen, um Einflüsse durch den Benutzer zu verhindern, die den Ablauf stören könnten. Deshalb sollte das ARexx-Script auch ausgiebige Fehler-Überprüfungen enthalten, um dem Fall vorzubeugen, daß das Script durch einen Fehler unterbrochen wird und der Benutzer aus Fiascos GUI ausgeschlossen ist. Das Script dummy.frx der Fiasco-Distribution enthält bereits sämliche nötigen Überprüfungen und kann als Basis für neue ARexx-Scripts benutzt werden.

12.2 Zugreifen auf den Port

Seit Fiasco 2.1, werden ARexx-Scripts, die von Fiasco gestartet werden, automatisch auf den ARexx-Port der Datenbank gesetzt, von der das Script gestartet wurde. Deshalb müssen Sie sich nicht darum kümmern, wie der Name des ARexx-Ports ist, auf den Sie zugreifen müssen.

Um in einem ARexx-Script zwischenzeitlich auf einen anderen Port zuzugreifen und später wieder auf den Fiasco, gibt es zwei Möglichkeiten:

Wenn Sie das Script mit Address *Port* auf einen anderen ARexx-Port zuweisen, können Sie später das Kommando Address ohne Argumente benutzen, um zum vorherigen aktiven Port zurückzugehen. Wenn das Script von Fiasco gestartet wurde, ist dies der Port von der Fiasco Datenbank. Zum Beispiel:

```
/* Ein Script, das von Fiasco gestartet wurde */

Address MULTIVIEW.1
```

```
/* Etwas mit MultiView machen */

Address

/* Nun kann man wieder mit Fiasco arbeiten */
```

Eine andere Möglichkeit ist, den Namen des aktiven ARexx-Ports in einer Variablen zwischenzuspeichern und später Address mit dieser Variablen zu benutzen, um zum ursprünglichen Port zurückzukehren. Hier müssen Sie dann Address Value *Variable* benutzen, um ARexx zu sagen, daß Sie einen Variablen-Namen angegeben haben und nicht direkt einen Port-Namen. Die Vorteile sind, daß Sie mehrere Ports ansteuern können, bevor Sie zum ursprünglichen Port zurückkehren. Zum Beispiel:

```
/* Ein Script, das von Fiasco gestartet wurde */

fiasco_port = address()

Address MULTIVIEW.1

/* Etwas mit MultiView machen */

Address Value fiasco_port

/* Nun kann man wieder mit Fiasco arbeiten */
```

Das Script dummy.frx der Fiasco-Distribution initialisiert bereits eine Variable mit dem Namen fiasco_port auf den entsprechenden ARexx-Port.

Den Port des aktiven Projektes herausfinden

Wenn Sie ein Fiasco-2.1-ARexx-Script von ausserhalb Fiasco starten wollen, z.B. über doppelklicken eines Projekt-Icons mit rx als Default-Tool, muß das Script nach dem Port der aktiven Fiasco-Datenbank suchen.

Dies kann so geschehen:

```
/* Eine Liste aller verfügbaren Ports holen */

ports = show("Ports")

/* Nach einem Fiasco-Port suchen */

do i = 1 to words(ports)
```

```
    if abbrev(word(ports, i), "FIASCO.") then
    do
        /* Ein Port von Fiasco wurde gefunden.
         * Nun bei Fiasco nach dem Port-Namen
         * der aktiven Datenbank fragen.
         */

        Address Value word(ports, i)

        GetAttr Project Name Active ARexx

        Address Value Result

        break
    end
end
```

Diese Prozedur ist auch Teil des Scripts dummy.frx.

12.3 Argumente für Kommandos

Fiasco interpretiert die Daten nach dem Kommando als Argument für das Kommando. Dabei benutzt Fiasco die Funktion ReadArgs() der dos.library. Somit gelten dieselben Argument-Konverionen für Fiasco wie die von CLI-Kommandos. Argumente durch Leerzeichen begrenzt. Wenn einzelne Parameter Leerzeichen enthalten sollen, reicht es *nicht*, diese einfach in Anführungszeichen einzuschließen, da ARexx alle Anführungszeichen schluckt. Um dies zu vermeiden, sollten Sie die Anführungszeichen in andere Anführungszeichen einschließen. (z.B. Open ' "Test Datei" ') Sie müssen die einfachen Anführungzeichen in der äußeren Position angeben, da Fiasco (bzw. ReadArgs()) nur doppelte Anführungszeichen verarbeiten kann. Beachten Sie, daß sie keine Variablen in den Anführungszeichen benutzen können. Schließen Sie dann die äußeren Anführungszeichen, sagen dann den Variablennamen und öffnen dann wieder die Anführungszeichen, um eine Variable in ein Argument einzufügen. Diese Prozeduren sind für Argumente, die den /F-Modifier haben, nicht nötig.

12.4 Rückgabewerte von Kommandos

Wenn ein Kommando einen Wert zurückliefert steht dieser normalerweise in der Variablen Result. Um Result zu benutzen, muß man die Zeile Options Results am Anfang eines ARexx-Scripts einfügen. Das Script dummy.frx hat dies bereits. Ein Wert, der in Result zurückgegeben wird, kann auch auf eine andere Variable umgelenkt werden. Um dies zu tun, müssen Sie das Argument Var gefolgt vom Namen der Variablen bei den Argumenten des Kommandos angeben. *Beachen Sie:* Wenn Sie mehrere Kommandos mit dem gleichen Var-Argument nacheinander aufrufen und Sie das Argument nicht in Anführungszeichen angeben, wird ARexx beim zweiten Kommando den Variablen-Namen durch das Ergebnis des ersten Kommandos ersetzen.

Kommandos, die komplexere Ergebnisse zurückgeben, benutzen Stem-Variablen. Wenn Sie solche Kommandos benutzen, müssen Sie einen Namen für die zu setzende Stem-Variable angeben. Wenn das Kommando eine Liste mit einer unbekannten Anzahl von Einträgen zurückgibt, werden die Werte der Einträge in *stem.*1 . . . *stem.n* zurückgegeben. *stem.*count enthält dann die Anzahl der Einträge.

Ein Fiasco-Kommando gibt normalerweise in der Variable RC 0 zurück, wenn alles glatt gelaufen ist. Falls ein Kommando nicht die Gegebenheiten, die es benötigt, vorfindet, wird RC auf 5 gesetzt, was soviel wie "Warnung" bedeutet. Fehler wie falsche Parameter werden mit RC = 10 belohnt, fatalere Fehler kriegen sogar RC = 20.

Seit Fiasco 2.0 wird eine besondere ARexx-Variable mit dem Namen FIASCO.LASTERROR unterstützt, die bei einem Fehler eines Fiasco-ARexx-Kommandos gesetzt wird.

Die Fehler-Codes die von Fiasco zurückgegeben werden können, sind in Anhang B dokumentiert.

Das ARexx-Kommando Fault (siehe Abschnitt 12.6.23) kann zum Umwandeln einer Fehlernummer in lesbaren Text benutzt werden.

12.5 Fehlersuche in ARexx-Scripts

Fiasco hat zusätzliche Eigenschaften, um die Fehlersuche in ARexx-Scripts so einfach wie möglich zu gestalten.

Wann immer ein Fiasco-Kommando nicht funktioniert, gibt Fiasco in der Variable rc eine Zahl ungleich 0 zurück. Weiterhin wird die spezielle Variable Fiasco.LastError auf einen Fiasco-Fehler-Code gesetzt. Wenn Sie Fehler-Signale benutzen, kann das ARexx-Script den Fehler abfangen und einen komfortablen Requester mit dem Fehler anzeigen. Ein Beispiel für ein solches Script ist dummy.frx aus der Fiasco-Distribution.

Einen Schritt weiter geht die ARexx-Debug-Funktion von Fiasco. Sie könnnen diese Funktion aktivieren, indem Sie den Menü-Punkt Kontrolle/ARexx-Debug aktivieren. Wann immer nun ein Fiasco-ARexx-Kommando nicht funktioniert, wird Fiasco die Ausführung des Scripts anhalten und einen Requester anzeigen, der den Fehler erklärt, die richtige Kommando-Schablone anzeige und Ihnen die Möglichkeit gibt, die AmigaGuide-Hilfe für das ARexx-Kommando anzuzeigen. Dies geschieht mit einem Klick auf Hilfe. Die anderen Schalter Weiter und Fehler Ignorieren setzen das Script fort, wobei der letztere Schalter rc auf 0 setzt. Dadurch nimmt das Script an, daß das Kommando erfolgreich ausgeführt wurde.

12.6 ARexx-Kommandos

12.6.1 ActivateDBWindow

Name: ActivateDBWindow - Aktiviere ein Datenbank-Fenster **(8)**

Synopsis: ActivateDBWindow Project/K
RC = *Erfolg*

Funktion: Aktiviert das Masken-Fenster der angegebenen Datenbank.

Der Eingabe-Focus des Benutzers wird auf dieses Fenster umgelenkt. Dies hat jedoch keine Auswirkungen auf den Ablauf des aufrufenden ARexx-Skripts.

Wenn die Datenbank verborgen ist, wird nichts geschehen.

Argumente: Project - Name der zu aktivierenden Datenbank

Ergebnisse: RC - Erfolg

12.6.2 ActivateField

Name: ActivateField - Aktiviere ein Feld in der Maske.

Synopsis: ActivateField FieldID/A
RC = *Erfolg*

Funktion: Aktiviert das Feld mit der angegebenen ID in der Maske. Nur Felder, die als String/Longint-Gadgets erscheinen, können aktiviert werden. Wenn das Projekt bzw. das Fenster nicht aktiv ist, kann das Feld nicht aktiviert werden.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: FieldID - ID des zu aktivierenden Feldes.

Ergebnisse: RC - 0 wenn das Feld aktiviert wurde.

12.6.3 ActiveIndex

Name: ActiveIndex – Kontrolle über den aktiven Index

Synopsis: ActiveIndex Index/K,Next/S,Prev/S,NoHistory/S,Force/S,Var/K
RC = *Erfolg*
Result = *Aktiver Index*

Funktion: Dieses Kommando kontrolliert den aktiven Index. Sie können einen Index über seinen Namen aktivieren oder mit den Argumenten Next und Prev durch die Index-History (siehe Abschnitt 5.3.1) gehen. Der Name des neu aktivierten Index wird zurückgegeben. Wenn sie weder den Index Namen, noch Next oder Prev angeben, wird nur der Name des aktiven Index zurückgegeben.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Index - Aktiviere angegebenen Index.
Next - Aktiviere den nächsten Index in der History.
Prev - Aktiviere den vorigen Index in der History.
NoHistory - Diese Veränderung soll nicht in der Index-History aufgezeichnet werden.
Force - Alle Warnungen ignorieren.

Ergebnisse: RC - Erfolg. Result - Name des nun aktiven Index.

12.6.4 ActiveRecord

Name: ActiveRecord – Kontrolle über den aktiven Record

Synopsis: ActiveRecord Record/K/N,First/S,Last/S,Next/S,Prev/S,Var/K
RC = *Success*
Result = *Neuer aktiver Record*

Funktion: Mit ActiveRecord können Sie den aktiven Record in der angesteuerten Datenbank kontrollieren. Sie können einen Record über die Record-Nummer aktivieren oder mit First, Last den ersten oder letzten Record aktivieren oder mit Next, Prev den Record vor bzw. nach dem derzeit aktiven Record aktivieren. Die Nummer des neuen aktiven Records wird zurückgegeben. Wenn Sie keins der ersten

fünf Argumente angeben, wird nur die Nummer des aktiven Records zurückgegeben und die Position nicht verändert.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Aktiviere den angegebenen Record.
First - Aktiviere den ersten Record.
Last - Aktiviere den letzten Record.
Next - Aktiviere den Record nach dem derzeit aktiven.
Prev - Aktiviere den Record vor dem derzeit aktiven.

Ergebnisse: RC - Erfolg
Result - Nummer des aktiven Records.

12.6.5 AddLVFieldEntry

Name: AddLVFieldEntry - Einem Listview einen Eintrag zufügen

Synopsis: AddLVFieldEntry FieldID/A,Record/K/N,ListEntry/K/N
RC = *Erfolg*

Funktion: Fügt einen neuen Eintrag nach den Einträgen des Listview-Felds ein. Er wird den Startwert aus dem Feld-Requester enthalten. Um den Inhalt zu ändern, benutzen Sie SetField.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: FieldID - ID eines Listview-Feldes
Record - Nummer des zu ändernden Records. Standard: Aktiver Record.
ListEntry - Nummer des Eintrages, nach dem der Eintrag eingefügt wird. Beginnend bei 1. 0 fügt einen Eintrag am Anfang der Liste ein.

Ergebnisse: RC - Erfolg

12.6.6 AddRecord

Name: AddRecord - Erzeuge einen neuen Record

Synopsis: AddRecord Record/K/N,Inactive/S,Var/K
RC = *Erfolg*
Result = *Record-Position*

Funktion: Erzeugt einen neuen Record und fügt diesen in die angesteuerte Datenbank ein. Der Record enthält die Startwerte.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Record, nach dem der neue Record eingefügt wird. Beachten Sie bitte, daß Fiasco eine andere Position für den Record wählen kann. Dies ist z.B. mit automatischem Sortieren der Fall. Wenn nicht angegeben, wird der Record nach dem aktiven eingefügt.
Inactive - Der Record wird nicht aktiviert.

Ergebnisse: RC - Erfolg
Result - Die tatsächliche Position des neuen Records.

12.6.7 CalculateFormula

Name: CalculateFormula - Berechnet eine Formel.

Synopsis: CalculateFormula Formula/A,Record/K/N,Var/K
RC = *Erfolg*
Result = *Ergebnis der Berechnung*

Funktion: Berechnet eine Formel und gibt das Ergebis zurück. Wenn Felder in der Formel benutzt werden, werden die Feld-Inhalte des aktiven oder des angegebenen Records in der angesteuerten Datenbank benutzt.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Formula - Zu berechnende Formel.
Record - Record, der bei der Berechnung benutzt werden soll.

Ergebnisse: RC - Erfolg.
Result - Ergebnis der Berechnung.

12.6.8 Clear

Name: Clear – Leere eine Datenbank

Synopsis: Clear Force/S
RC = *Erfolg*

Funktion: Löscht alle Daten im aktuellen Projekt, so daß es sich hinterher in einem Zustand wie gerade über Projekt/Neu neu geöffnete Projekte befindet. Wenn Sie nicht Force angeben, entspricht dieses Kommando dem Menüpunkt Projekt/Leeren, so ist es also auch möglich, daß ein Requester erzeugt wird, in dem gefragt wird, ob das Projekt vorher noch gespeichert werden soll. Um dies zu umgehen, sollte Force angeben werden.

Argumente: Force - Alle Warnungen unterdrücken.

Ergebnisse: RC - Erfolg.

12.6.9 CloneRecord

Name: CloneRecord – Klone einen Record

Synopsis: CloneRecord Record/K/N,To/K/N
RC = *Erfolg*

Funktion: Erzeugt eine exakte Kopie des aktiven oder angegebenen Records und fügt diese an der angegebenen Stelle in den aktiven Index ein. Dies kann auch Auswirkungen auf andere Indizes haben.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Nummer des zu klonenden Records. Standard: Aktiver Record.
To - Nummer des Records, nach dem der neue Record eingefügt wird. Standard: Aktiver Record.

Ergebnisse: RC - Erfolg

12.6.10 Close

Name: Close – Schließe eine Datenbank

Synopsis: Close Force/S
RC = *Erfolg*

Funktion: Schließt das angesteuerte Projekt. Wenn Sie nicht Force angeben, entspricht dieses Kommando dem Menüpunkt Projekt/Beenden, so ist es also auch möglich, daß ein Requester erzeugt wird, in dem gefragt wird, ob das Projekt vorher noch gespeichert werden soll. Um dies zu umgehen, sollte Force angeben.

Argumente: Force - Alle Warnungen unterdrücken.

Ergebnisse: RC - Erfolg.

12.6.11 CloseListWindow

Name: CloseListWindow – Schließe das Listen-Fenster

Synopsis: CloseListWindow

Funktion: Schließt das Listen-Fenster. Falls das Fenster nicht offen sein sollte, passiert gar nichts. Diese Kommando entspricht dem Deaktivieren des Menüpunktes Kontrolle/Listen-Fenster.

Argumente: Keine.

Ergebnisse: Keine.

12.6.12 CloseServiceWindow

Name: CloseServiceWindow - Schließe das Service-Fenster

Synopsis: CloseServiceWindow

Funktion: Schließt das Service-Fenster. Falls das Fenster nicht offen sein sollte, passiert gar nichts. Diese Kommando entspricht dem Deaktivieren des Menüpunktes Kontrolle/Service-Fenster.

Argumente: Keine.

Ergebnisse: Keine.

12.6.13 ConvertField

Name: CovertField - Ändere den Typ eines Feldes

Synopsis: ConvertField ID/A,NewType/A,Listview/S,AltFormat/S
RC = *Erfolg*

Funktion: Ändert den Typ des angegebenen Feldes. Sie können keine Text-, Leisten- oder Button-Felder konvertieren. Kann nur im Masken-Modus aufgerufen werden.

Argumente: ID - ID des Feldes.
NewType - Neue Typ des Feldes (z.B. String).
Listview - Bestimmt, ob das Feld ein Listview-Feld wird.
AltFormat - Aktiviert alternatives Format.

Ergebnisse: RC - Erfolg

12.6.14 CopyRecord

Name: CopyRecord – Kopiere einen Record ins Clipboard

Synopsis: CopyRecord Record/K/N,Unit/K/N
RC = *Erfolg*

Funktion: Kopiert den aktiven oder angegebenen Record ins Clipboard.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Nummer des zu kopierenden Records. Standard: Aktiver Record.
Unit - Nummer des Clipboard-Unit. Standard: 0

Ergebnisse: RC - Erfolg

12.6.15 CountRecords

Name: CountRecords - Ermittele die Anzahl von Records.

Synopsis: CountRecords Var/K
RC = *Erfolg*
Result = *Record-Anzahl*

Funktion: Gibt die Anzahl der Records im aktiven Index der angesteuerten Datenbank zurück.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente:

Ergebnisse: RC - Erfolg
Result - Anzahl der Records im aktiven Index.

12.6.16 CreateField

Name: CreateField - Erzeuge ein Feld

Synopsis: CreateField Type/A,Listview/S,ID/A
RC = *Erfolg*

Funktion: Erzeugt ein neues Feld in der angesteuerten Datenbank. Die Feld-Attribute werden Standard-Werte enthalten. Das Feld wird sowohl in der Liste als auch in der Maske verborgen sein. Um die Attribute zu verändern und um das Feld sichtbar zu machen, müssen Sie das Kommando SetAttr (siehe Abschnitt 12.6.59) benutzen.

Kann nur im Masken-Modus aufgerufen werden.

Argumente: Type - Typ des Feldes.
Listview - Angeben wenn Feld ein Listview-Feld werden soll.
ID - ID des neuen Feldes. Das Kommando wird fehlschlagen, falls die ID bereits existiert.

Ergebnisse: RC - Erfolg.

12.6.17 CutRecord

Name: CutRecord – Schneide einen Record aus

Synopsis: CutRecord Record/K/N,Unit/K/N,Force/S
RC = *Erfolg*

Funktion: Kopiert den aktiven oder angegebenen Record ins Clipboard und löscht diesen aus dem aktiven Index.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Nummer des auszuschneidenden Records. Standard: Aktiver Record.
Unit - Nummer der Clipboard-Unit. Standard: 0
Force - Alle Warnungen unterdrücken.

Ergebnisse: RC - Erfolg

12.6.18 DeleteAllRecords

Name: DeleteAllRecords – Entferne alle Records

Synopsis: DeleteAllRecords Force/S
RC = *Erfolg*

Funktion: Entfernt alle Records aus dem aktiven Index. Die Maske bleibt unangetastet. Wenn Sie nicht Force angeben, entspricht dieses Kommando dem Menüpunkt Record/Alle Löschen, so ist es also auch

möglich, daß ein Requester erzeugt wird, in dem gefragt wird, ob das Projekt vorher noch gespeichert werden soll. Um dies zu umgehen, sollte Force angeben.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Force - Alle Warnungen unterdrücken

Ergebnisse: RC - Erfolg

12.6.19 DeleteConstant

Name: DeleteConstant - Lösche eine Konstante **(7)**

Synopsis: DeleteConstant Name/A
RC = *Erfolg*

Funktion: Löscht die angegebene Konstante in der angesteuerten Datenbank. Wenn eine Konstante mit dem angegebenen Namen nicht existieren sollte, wird ein Fehler zurückgegeben.

Konstanten werden im Allgemeinen in Fiasco-Formeln benutzt, können aber auch für andere Zwecke in Anspruch genommen werden. Der Benutzer kann die Konstanten einer Datenbank mit dem Konstanten-Requester (siehe Abschnitt 10.6.9) betrachten und ändern.

Argumente: Name - Name der zu löschenden Konstante.

Ergebnisse: RC - Erfolg.

12.6.20 DeleteLVFieldEntry

Name: DeleteLVFieldEntry - Lösche einen Listview-Eintrag

Synopsis: DeleteLVFieldEntry FieldID/A,Record/K/N,ListEntry/A/N
RC = *Erfolg*

Funktion: Löscht den angegebenen Eintrag in einem Listview-Feld.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: FieldID - ID eines Listview-Feldes.
ListEntry - Nummer des zu löschenden Eintrags.
Record - Nummer des Records, in dem der Eintrag gelöscht wird. Wenn Sie dieses Argument auslassen, wird der aktive Record benutzt.

Ergebnisse: RC - Erfolg

12.6.21 DeleteRecord

Name: DeleteRecord – Lösche einen Record

Synopsis: DeleteRecord Record/K/N,Force/S
RC = *Erfolg*

Funktion: Entfernt den aktiven oder angegebenen Record aus dem aktiven Index. Wenn der aktive Record gelöscht wurde, wird ein anderer aktiviert. Wenn Sie den Force-Parameter nicht angeben, macht dieses Kommando exakt das selbe wie Record/Löschen. Das bedeutet, daß ein Requester erzeugt werden kann, der Sie vor dem Entfernen warnt. Um dies zu unterdrücken, geben Sie Force an.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Zu löschender Record. Standard: Aktiver Record.
Force - Keine Warnungen.

Ergebnisse: RC - Erfolg

12.6.22 Export

Name: Export – Exportiere eine Datenbank

Synopsis: Export File/A,RecStart=RS/K,RecEnd=RE/K,RecSep=RP/K,
FieldStart=FS/K,FieldEnd=FE/K,FieldSep=FP,FirstRecIDs/K,
MarkedOnly/S
RC = *Erfolg*

Funktion: Ruft die Export-Funktion von Fiasco auf. Der Import/Export-Abschnitt enthält mehr Informationen über das Exportieren. Wenn Sie ein Argument nicht angeben, wird es leer sein.

Argumente: File - Zu schreibende Datei
RecStart,RecEnd,RecSep,FieldStart,FieldEnd,FieldSep -
Struktur-Parameter
LVEntrySep - Struktur-Parameter für Listview-Felder
FirstRecIDs - Erster Record wird Feld-IDs enthalten
MarkedOnly - Exportiert nur markierte Records

Ergebnisse: RC - Erfolg

12.6.23 Fault

Name: Fault – Einen Fehlercode in Text umwandeln

Synopsis: Fault ErrorCode/N,Var/K
RC = *Erfolg*
Result = *Fehler-Text*

Funktion: Wandelt einen Fiasco-Fehlercode aus FIASCO.LASTERROR in ein menschlich lesbares Format um. Er wird an die aktive Sprache angepaßt. Da FIASCO.LASTERROR auch Amiga DOS Fehlercodes zurückgeben kann, kommt diese Funktion auch damit zurecht.

Argumente: ErrorCode - Fiasco-Fehlercode

Ergebnisse: RC - Erfolg.
Result - Lokalisierter Fehler-Text.

12.6.24 Filter

Name: Filter - Erzeuge einen gefilterten Index

Synopsis: Filter SearchInfo/K/A,Index/K
RC = *Erfolg*

Funktion: Erzeugt einen Filter, der sich nach den Kriterien, die im Such-Info (SearchInfo) angegben wurden, richtet. Der Filter wird aktiviert, nachdem er erzeugt wurde.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: SearchInfo - Name des zu benutzenden Search Infos.
Index - Name des zu erzeugenden Index. Standard: ARexxFilter.fidx.

Ergebnisse: RC - Erfolg.

12.6.25 Find

Name: Find - Suche nach Daten

Synopsis: Find SearchInfo/K/A,Record/K/N,Reverse/S,All=Stem/K,Var/K
RC = *Erfolg*
Result = *Gefundener Record*

Funktion:

Sucht nach einem Muster, das im SearchInfo angegeben ist. Dieses kann mit den Kommandos NewSearchInfo (siehe Abschnitt 12.6.40) und SetSearchField (siehe Abschnitt 12.6.63) erzeugt werden.

Dieses Kommando hat zwei Modi: Sie können einmal nach einem passenden Record suchen. In diesem Modus können Sie den Record angeben *nach* dem die Suche beginnen wird. Die Nummer des nächsten übereinstimmenden Records (bzw. des vorigen, wenn Reverse benutzt wurde) wird dann in Result zurückgegeben. Um nach der nächsten Übereinstimmung zu suchen, müssen Sie das Kommando danach mit der Nummer des jetzt gefundenen Records als Record-Argument aufrufen. Das Beispiel zeigt die Benutzung dieses Modus.

Der zweite Modus erlaubt Ihnen, die ganze Datenbank mit einem Kommando-Aufruf zu durchsuchen. Wenn Sie dem Namen einer Stem-Variablen nach dem Schlüsselwort All angeben, wird diese Kommando die ganze Datenbank durchsuchen und die Record-Nummern der Übereinstimmungen in dem Stem-Elementen *stem.*1, . . . *stem.n* ablegen. Die Anzahl der Übereinstimmungen wird in *stem.*count abgelegt. *Wichtig:* Benutzen Sie diesen Modus nicht, wenn eine sehr große Anzahl von Übereinstimmungen vorhersehbar ist. In diesem Fall ist der erste Modus schneller und RAM-Effizienter.

Mehr über das Suchen mit ARexx befindet sich in Abschnitt 6.4.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: SearchInfo - Name des zu benutzenden SearchInfos.
Record - Nummer des Records *nach* die Suche beginnt. Schließt All aus.
Reverse - Durchsuche die Datenbank rückwärts.
All - Durchsuche die ganze Datenbank auf einmal und speichere das Ergebis in der angegebenen Stem-Variable. Schließt Record aus.

Ergebnisse: RC - Null wenn eine Übereinstimmung gefunden wurde.
Result - Nummer des gefundenen Records.

Beispiel:

```
/* Such-Beispiel.frx */

/* Erzeuge ein neues SearchInfo */

NewSearchInfo Name "ARexxSI"

/* Das SearchInfo soll nach einem Record suchen,
 * in dem das Feld mit der ID Test nicht leer ist.
 */
```

```
SetSearchField SearchInfo "ARexxSI" FieldID "Test" Pattern "?#?"

rc = 0
startrecord = 0

/* Setze die Suche fort bis keine
 * Übereinstimmung mehr gefunden wird
 */

do while rc = 0

   Find SearchInfo "ARexxSI" Record startrecord Var "startrecord"

   if rc = 0 then
   do
       /* Die Nummer der Übereinstimmung wird in
        * startrecord abgelegt. Nun kann man was
        * damit machen.
        */

       say startrecord
   end
end

/* Alle Records durchsucht */
```

12.6.26 FlushRecords

Name: FlushRecords - Räume RAM auf **(7)**

Synopsis: FlushRecords

Funktion: Versucht so viel Speicher wie möglich freizugeben. Dies geschieht, indem Records aus dem RAM entfernt werden, die von Disk wieder geladen werden können.

Argumente: Keine.

Ergebnisse: Keine.

12.6.27 GetAttr

Name: GetAttr – Hole ein Fiasco-Attribut

Synopsis: GetAttr Object/A,Name/K,Attribute,Stem/K,Var/K
RC = *Erfolg*
Result = *Attribut-Wert*

Funktion: Liest das angegebene Attribut und gibt es zurück.

Argumente: Object - Zu untersuchender Objekt-Typ.
Name - Bei manchen Objekt-Typen muß ein Name angegeben werden.
Attribute - Name des zurückzugebenden Attributs.
Stem - Alle Attribute eines Objekts werden in einer Stem-Variable zurückgegeben.

Ergebnisse: RC - Erfolg.
Result - Wert des angeforderten Attributs.

Objekte: Application - Daten allgemein zu Fiasco.
Projects - Gibt alle offenen Datenbanken in einer Stem-Variable zurück. Keine weiteren Attribute.
Project - Daten zu einer Fiasco-Datenbank. Der Name kann angegeben werden. Wenn Sie Active hier angeben, wird das aktive Projekt (im Kontrast zu dem *angesteuerten*) benutzt. Standard: Angesteuerte Datenbank.
Window - Daten zu einem Datenbank-Fenster. Name wie bei Project.
Fields - Gibt alle Felder der angesteuerten Datenbank in einer Stem-Variable zurück. Keine weiteren Attribute.
Field - Daten zu einem Feld in der angesteuerten Datenbank. Name (ID) benötigt.

Attribute für Application: Version - Interne Version von Fiasco im Format *ver.rev*.
ReleaseVersion - Release-Version von Fiasco.
Screen - Name des Public Screen auf dem Fiasco läuft.
RegUser - Name des registrierten Benutzers.

Attribute für Projekt: ARexx - Name des ARexx-Ports.
FileName - Kompletter Dateiname der Datenbank.
Path - Pfad der Datenbank.
File - Dateiname der Datenbank.
Changes - Ungleich null wenn Datenbank verändert wurde.

Attribute für Window: Left - Linke Kante des Fensters.
Top - Obere Kante des Fensters.
Width - Breite des Fensters.
Height - Höhe des Fensters.

Title - Titel des Fensters.
Screen - Public Screen des Fensters.

Attribute für Field: Left - Linke Kante des Feldes.
Top - Obere Kante des Feldes.
Width - Breite des Feldes.
Height - Höhe des Feldes.
ARexxScript - ARexx-Script des Feldes.
PickerARexx - ARexx-Script für Auswahl-Schalter.
Formula - Formel für Feld.
Type - Typ des Felds.
ListLeft - Linke Kante im Listen-Fenster.
ListWidth - Breite Kante im Listen-Fenster.
Shortcut - Tastatur-Abkürzung des Feldes.
InitContType - Typ des Startwerts. Einer von LAST, KEY oder OWN.
InitContValue - Wert des eigenen Startwerts.
MaxChars - Max. Zeichen des Feldes. Nicht von allen Feldtypen unterstützt.
MinValue - Minimum Wert des Feldes. Nicht von allen Feldtypen unterstützt.
MaxValue - Maximum Wert des Feldes. Nicht von allen Feldtypen unterstützt.
Format - Format des Feldes. Nicht von allen Feldtypen unterstützt.
Labels - Labels des Feldes. Wird in einer Stem-Variable zurückgegeben. Nicht von allen Feldtypen unterstützt.
Precision - Präzision des Feldes. Nicht von allen Feldtypen unterstützt.
Command - Kommando für Feld. Nicht von allen Feldtypen unterstützt.
Stack - Stack für Feld. Nicht von allen Feldtypen unterstützt.
Virtual - Status der Virtual-Option. 1 oder 0.
ListHidden - Ist das Feld in der Liste verborgen? 1 oder 0.
Hidden - Ist das Feld in der Mask verborgen? 1 oder 0.
ReadOnly - Ist das Feld nur lesbar? 1 oder 0.
Listview - Ist das Feld ein Listview? 1 oder 0.

12.6.28 GetConstant

Name: GetConstant - Lese eine Konstante **(7)**

Synopsis: GetConstant Name/A,Var/K
RC = *Erfolg*
Result = *Wert der Konstante*

Funktion: Liest den Wert einer Konstante in der angesteuerten Fiasco-Datenbank. Wenn die Konstante nicht existiert, wird ein Fehler zurückgegeben.

Konstanten werden im Allgemeinen in Fiasco-Formeln benutzt, können aber auch für andere Zwecke in Anspruch genommen werden. Der Benutzer kann die Konstanten einer Datenbank mit dem Konstanten-Requester (siehe Abschnitt 10.6.9) betrachten und ändern.

Argumente: Name - Name der zu lesenden Konstante.

Ergebnisse: RC - Erfolg.
Result - Der Wert der angegebenen Konstante.

12.6.29 GetField

Name: GetField – Lesen des Feldinhalts

Synopsis: GetField FieldID,Record/K/N,ListEntry/K/N,ListEntryCount/S, ExtFormat/S,Stem/K,Var/K RC = *Erfolg*
Result = *Feld-Inhalt*

Funktion: Liest den Inhalt des Feldes mit der angegebenen ID im aktuellen bzw. im angegebenen Record aus und liefert ihn in Result zurück.

Wenn Sie eine Stem-Variable angeben, werden die Inhalte aller Felder in die ARexx-Stem-Variable geschrieben. Die Stem-Elemente werden die Feld-IDs sein. Sie können nach diesem Aufruf die Werte einiger Variablen ändern und mit SetField (siehe Abschnitt 12.6.61) Stem die Werte zurück in die Datenbank schreiben. Bitte beachten Sie, daß das Stem-Argument die Ausführungszeit stark erhöht. Sie sollten es daher nur benutzen, wenn Sie mehrere Felder auf einmal ändern müssen.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: FieldID - ID des Feldes
Record - Nummer des Records
ListEntry - Nur wenn Feld ein Listview-Feld ist. Gibt die Nummer des Eintrags im Listview an, dessen Inhalt zurückgegeben werden soll. Beginnend bei 1.
ListEntryCount - Nur wenn Feld ein Listview-Feld ist. Gibt die Anzahl von Einträgen im Listview zurück.
ExtFormat - Wenn angegeben, wird der Feldinhalt in einem erweiterten Format zurückgegeben, das beispielsweise für Ausgabe besser geeignet ist. Dieses erweiterte Format kann Locale-Einstellungen, etc.

benutzen. Sie sollten keine Annahmen über das resultierende Format machen. Siehe auch die Tabelle unter Result. (Fiasco 2.2)
Stem - Gibt alle Feldinhalte in der angegebenen Stem-Variable zurück.

Ergebnisse: RC - Erfolg

result - enthält den aktuellen Inhalt des Feldes, falls rc = 0 ist.
Das Format des Inhaltes (In Klammern: Bei ExtFormat):
String - der String selbst
Integer - die Zahl selbst
Float - die Fließkommazahl
Slider - der Wert des Sliders
Cycle - die Nummer des aktiven Eintrages (der aktive Eintrag selbst)
Date - das Datum im Format *TT.MM.[JJ]JJ* (im Locale-Format)
Time - die Zeit im Format *HH:MM:SS* (im Locale-Format)
Extern - der String selbst
Datatyp.- der String selbst
Var String- der String selbst. Zeilenumbrüche werden in den String *n umgewandelt.

Wenn das Feld ein Listview-Feld ist, müssen Sie das ListEntry Argument benutzen, um den Eintrag auszuwählen, der ausgelesen werden soll. Das Format wird dann das Format des eigentlichen Feldtypen sein.

Bemerkungen: Vor Release 2.2 behauptete dieses Dokument, daß die Daten aus Datums- und Zeit-Feldern im aktuellen Locale-Format zurückgegeben würden. Es ist und war jedoch der Fall, daß - es sei denn es wird ExtFormat benutzt - diese Werte unabhängig von den Locale-Einstellungen formatiert werden.

12.6.30 GetRecordMark

Name: GetRecordMark - Überprüfe die Markierung eines Records

Synopsis: GetRecordMark Record/K/N,Var/K
RC = *Erfolg*
Result = *Markiert oder nicht*

Funktion: Gibt den aktuellen Zustand der Markierung eines Records in der angesteuerten Fiasco-Datenbank zurück.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Nummer des Records. Standard: Aktiver.

Ergebnisse: RC - Erfolg
Result - 1 wenn Record markiert ist, sonst 0.

12.6.31 HideProject

Name: HideProject - Verberge ein Fiasco-Projekt

Synopsis: HideProject Project/K
RC = *Erfolg*

Funktion: Schließt alle Fenster des angesteuerten oder angegebenen Projektes.

Argumente: Project - Name des zu vergergenden Projekts kann der Dateiname oder der ganze Pfad sein.

Ergebnisse: RC - Erfolg

12.6.32 Import

Name: Import – Importiere eine Datenbank

Synopsis: Import File/A,RecStart=RS/K,RecEnd=RE/K,RecSep=RP/K,
FieldStart=FS/K,FieldEnd=FE/K,FieldSep=FP,
SkipLines/K,StartLine/N/K,FirstRecIDs/K,AppendFields/S
RC = *Erfolg*

Funktion: Ruft die Import-Funktion von Fiasco auf. Die angegebene Datei wird in das aktuelle Projekt mit den angegebenen Parametern importiert werden. Für mehr Informationen über Import und Export siehe Abschnitt 8. Sie können auch die Escape-Sequenzen von Fiasco benutzen. Wenn Sie einen Parameter nicht angeben, wird dieser leer sein.

Argumente: File - Name der Datei
RecStart,RecEnd,RecSep,FieldStart,FieldEnd,FieldSep - Struktur-Zeichen
SkipLines - Kommentar-Beginn
StartLine - Länge des Start-Kommentars
FirstRecIDs - Erster Record enthält IDs
AppendFields - Neue Felder anhängen

Ergebnisse: Result - Erfolg

Notes: Die Option Altes Projekt überschreiben des Import-Requesters ist nicht direkt unterstützt. Sie müssen dies mit Clear (siehe Abschnitt 12.6.8) emulieren.

12.6.33 LoadDTFieldObject

Name: LoadDTFieldObject - Lade ein Datatypes-Feld

Synopsis: LoadDTFieldObject FieldID/A
RC = *Erfolg*

Funktion: Läd den Inhalt eines Datatypes-Feldes, das "verzögert" ist.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: FieldID - ID des Datatypes-Feldes

Ergebnisse: RC - Erfolg

12.6.34 LockGUI

Name: LockGUI - Lock Fiasco's GUI

Synopsis: LockGUI

Funktion: Macht die GUI von Fiasco unverfügbar für den Benutzer. Der Maus-Zeiger wird als eine "Warteuhr" erscheinen. Nachdem die GUI verschlossen wurde, kann das ARexx-Script laufen, ohne daß die Gefahr besteht, daß das Script vom Benutzer beinflußt wird. Bevor das Script endet, muß UnlockGUI aufgerufen werden, um die Kontrolle zum Benutzer zurückzugeben. LockGUI und UnlockGUI können verschachtelt werden.

Argumente: Keine.

Ergebnisse: Keine.

12.6.35 MarkMatch

Name: MarkMatch - Mark matching records

Synopsis: MarkMatch SearchInfo/K/A
RC = *Success*

Funktion: Markiert die Records die auf den angegebenen SearchInfo passen. Die alten Markierungen werden überschrieben.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: SearchInfo - Name des zu benutzenden SearchInfos.

Ergebnisse: RC - Erfolg

12.6.36 MarkRecord

Name: MarkRecord – Kontrolliere die Markierung von Records

Synopsis: MarkRecord Record/K/N,All/S,Set/S,UnSet/S,Toggle/S
RC = *Erfolg*

Funktion: MarkRecord kann benutzt werden, um die Markierung eines (aktiven oder angegebenen) Records oder allen (all) Records in der angesteuerten Datenbank zu ändern. Sie können die Markierungen setzen (mit set), löschen (mit unset) oder umschalten (mit toggle).

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Nummer des zu ändernden Records. Standard: Aktiver.
All - Ändere alle Records. Schließt Record aus.
Set - Setze die Markierung(en).
UnSet - Lösche die Markierung(en).
Toggle - Umschalten: Markiert wird Nicht-Markiert und umgekehrt.

Ergebnisse: RC - Erfolg

12.6.37 MenuControl

Name: MenuControl – Schalte Pull-Down-Menüs ann oder aus (8)

Synopsis: MenuControl On/S,Off/S
RC = *Erfolg*

Function: Dieses Kommando schaltet die Pull-Down-Menüs von Fiasco an oder aus. Dies geschieht *lokal für eine Datenbank*.

Arguments: On - Menüs verfügbar
Off - Menüs nicht verfügbar

Results: RC - Erfolg.

12.6.38 MoveRecord

Name: MoveRecord – Record-Position ändern

Synopsis: MoveRecord Record/K/N,To/A/N
RC = *Erfolg*

Funktion: Verschiebt den aktiven oder angegebenen Record im aktiven Index an die neu angegebene Position. Dieses Kommand wird nicht funktionieren, wenn der aktive Index automatisches Sortieren macht. Außerdem hat dieses Kommando keine Auswirkungen auf andere Indizes.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Nummer des zu verschiebenden Records. Wenn nicht angegeben, wird der aktive Record benutzt.
To - Nummer des Records, nach dem der Record eingefügt wird. Sie müssen die Nummer des Records angeben, die aktuell ist bevor der Record verschoben wird.

Ergebnisse: RC - Erfolg

12.6.39 New

Name: New – Erzeuge ein neues Projekt

Synopsis: New Project/K,Iconified/S,Var/K
RC = *Success*
Result = *Neuer ARexx-Port*

Funktion: Erzeugt ein neues Datenbank-Projekt. Wenn Iconified nicht angegeben wird, wird ein neues Fenster geöffnet und aktiviert. Das Projekt ist dann völlig leer.

Argumente: Project - Neuer Dateiname des Projektes. Standard: Unbenannt.fdb.
Iconified - Die Fenster werden nicht geöffnet.

Ergebnisse: rc - Erfolg.
Result - Der Name des neuen ARexx-Ports. Kann mit Address benutzt werden.

12.6.40 NewSearchInfo

Name: NewSearchInfo - Erstelle ein neues SearchInfo

Synopsis: NewSearchInfo Name/K,Fields/S,Formula/K,Var/K
RC = *Erfolg*
Result = *SearchInfo Name*

Funktion: Erstellt ein neues SearchInfo. Sie können einen eigenen Namen für das neue SearchInfo mit dem Argument Name angeben. Wenn ein SearchInfo mit diesem Namen bereits existieren sollte, wird dieses gelöscht. Wenn Sie das Name-Argument nicht angeben, wird diese Funktion einen eindeutigen Namen erzeugen und diesen in Result zurückgeben.

Wenn Sie mit diesem SearchInfo über Felder suchen wollen, müssen Sie mit dem Kommando SetSearchField die Felder und entsprechenden Muster angeben. Wenn Sie mit einer Formel suchen wollen, müssen Sie diese bei diesem Aufruf mit dem Formula-Argument angeben.

Argumente: Name - Name des zu erstellenden SearchInfos. Wenn nicht angegeben, wird ein eindeutiger Name erzeugt.
Field - Die Suche soll über Felder erfolgen.
Formula - Wenn Sie mit einer Formel suchen wollen, geben Sie die Formel nach diesem Schlüsselwort an.

Ergebnisse: RC - Erfolg
Result - Name des neu erzeugten SearchInfos.

12.6.41 Open

Name: Open - Eine Fiasco-Datenbank öffnen

Synopsis: Open File/A,New/S,Iconified/S,Force/S,Var/K
RC = *Success*
Result = *ARexx Port Name*

Funktion: Versucht, eine Fiasco-Datenbank zu laden. Wenn Sie nur das File-Argument angeben, werden die Daten in das angesteurte Projekt-Fenster geladen. Wenn sich dort noch geänderte Daten befinden, wird vorher noch ein Warnungs-Requester geöffnet. Um dies zu unterdrücken, müssen Sie das Force-Argument angeben. Wenn Sie New angeben, wird Fiasco die Daten in ein neu erstelltes Projekt laden. Wenn Sie Iconified angeben, wird Fiasco die Fenster dieses Projektes nicht öffnen.

Argumente: Name - Dateiname der zu öffnenden Datenbank.
New - Lade in ein neues Projekt.
Iconified - Öffne nicht das Projekt-Fenster. Nur in Kombination mit New.

Ergebnisse: *RC* - Erfolg.
Result - Name des ARexx-Ports des Projektes, in das die Datenbank geladen wurde. Nützlich im Zusammenhang mit New.

12.6.42 OpenListWindow

Name: OpenListWindow – Öffne das Listen-Fenster

Synopsis: OpenListWindow

Funktion: Öffnet das Listen-Fenster des angesteuerten Projektes, in dem der Benutzer einen Überblick über die Records des Projektes bekommt. Falls das Listen-Fenster bereits offen sein sollte, passiert gar nichts. Dieses Kommando entspricht dem Aktivieren des Menüpunktes Kontrolle/Listen-Fenster (siehe Abschnitt 10.4.66).

Argumente: Keine.

Ergebnisse: Keine.

12.6.43 OpenServiceWindow

Name: OpenServiceWindow - Öffne das Service-Fenster

Synopsis: OpenServiceWindow

Funktion: Öffnet das Service-Fenster (siehe Abschnitt 10.3), falls es noch nicht offen ist.

Argumente: Keine.

Ergebnisse: Keine.

12.6.44 PasteRecord

Name: PasteRecord – Füge einen Record aus dem Clipboard ein

Synopsis: PasteRecord Record/K/N,Unit/K/N,Inactive/S
RC = *Erfolg*

Funktion: Fügt einen Record aus dem Clipboad in die angesteuerte Datenbank ein. Der Record kann mit CopyRecord (siehe Abschnitt 12.6.14) oder CutRecord (siehe Abschnitt 12.6.17) ins Clipboard geschrieben worden sein.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Record - Nummer des Records, nach dem der neue Record eingefügt wird. Standard: Der aktive Record.
Unit - Nummer der Clipboard-Unit. Standard: 0.
Inactive - Der neue Record wird nicht aktiviert.

Ergebnisse: RC - Erfolg

12.6.45 Progress

Name: Progress – Zeige eine Aktivitäts-Leiste

Synopsis: Progress Done/A/N,Max/A/N
RC = *Erfolg*

Funktion: Zeigt eine Aktivitäts-Leits im Sevice-Fenster an. Sie sollten das Status-Gadget mit ResetStatus (siehe Abschnitt 12.6.54) zurücksetzen, wenn die Operation beendet ist.

Argumente: Done – Die Anzahl von Daten-Einheiten, die bereits durchlaufen wurden.
Max – Die Anzahl aller Daten-Einheiten.

Ergebnisse: RC - Erfolg.

12.6.46 Quit

Name: Quit – Beende Fiasco

Synopsis: Quit

Funktion: Schließt alle Datenbanken und verläßt Fiasco. Keine Warnungen werden erzeugt. Nach diesem Kommando werden Sie auf keine ARexx-Ports von Fiasco mehr zugreifen können.

Argumente: Keine.

Ergebnisse: Keine.

12.6.47 ReadSettings

Name: ReadSettings – Lade Fiasco-Einstellungen **(8)**

Synopsis: ReadSettings File/A
RC = *Erfolg*

Function: Liest Fiasco-Einstellungen aus der angegebenen Datei und aktiviert diese.

Arguments: File - Zu lesende Datei.

Results: RC - Erfolg.

12.6.48 RecompileFormulas

Name: RecompileFormulas - Alle Formeln aktualisieren **(7)**

Synopsis: RecompileFormulas
RC = *Erfolg*

Funktion: Rekompiliert (d.h. aktualisiert) alle Formeln in der angesteuerten Datenbank. Dies ist insbesonders nach einem Aufruf von SetConstant nützlich.

Argumente: Keine.

Ergebnisse: RC - Erfolg.

12.6.49 RequestChoice

Name: RequestChoice - Eine Auswahl anfordern

Synopsis: RequestChoice Body/A,Gadgets/A,Title/K,Var/K
RC = *Erfolg*
Result = *Auswahl*

Funktion: Erzeugt einen Intuition Easy-Requester mit den angegebenen Parametern. Arbeitet sehr ähnlich zum CLI-Kommando RequestChoice. Die Unterschiede: Leicht andere Parameter, erzeugt den Requester auf Fiascos Screen.

Argumente: Body - Haupt-Text des Requesters.
Gadgets - Gadgets am unteren Rand des Requesters. Jedes Gadget muß durch einen | getrennt werden.
Title - Titel des Requesters.

Ergebnisse: RC - Erfolg.
Result - Nummer des ausgewählten Gadgets, 0 für das rechte.

12.6.50 RequestField

Name: RequestField - Frage eine Feld-ID ab.

Synopsis: RequestField Default/K,Text/A,Var/K
RC = *Erfolg*
Result = *Ausgewähltes Feld*

Funktion: Öffnet einen Requester mit einer Liste aller Felder des aktiven Projektes. Der Requester kann eine zusätzliche Nachricht anzeigen, die mit dem Text Argument übergeben wird. Der Benutzer kann ein Feld auswählen und auf Ok klicken oder den Requester Abbrechen.

Argumente: Default - Dieses Feld wird ausgewählt sein, wenn sich der Requester öffnet.
Text - Der anzuzeigende Text.

Ergebnisse: RC - Erfolg.
Result - ID des augewählten Feldes.

12.6.51 RequestFile

Name: RequestFile – Eine Datei abfragen

Synopsis: RequestFile
File,Pattern/K,Title/K,SaveMode/S,DrawersOnly/S,NoIcons/S,
ProjectRelative/S,Var/K
RC = *Erfolg*
Result = *Ausgewählte Datei*

Funktion: Erzeugt einen ASL-Filerequester. Arbeitet sehr ähnlich zu dem CLI-Kommando RequestFile. Die Unterschiede: Leicht andere Parameter, erzeugt den Requester auf Fiascos Screen.

Argumente: File - Startwert für Datei (incl. Pfad)
Pattern - Startwert für Pattern
Title - Titel für Requester
SaveMode - Aktiviert Savemode: Schwarzer Hintergrund, keine Auswahl via Doppelklick
DrawersOnly - Zeigt nur Verzeichnisse an
NoIcons - Filtert Icons aus

ProjectRelative - Die ausgewählte Datei wird relativ zur angesteuerten Datenbank sein.

Ergebnisse: RC - Erfolg.
Result - Ausgewählte Datei.

12.6.52 RequestNumber

Name: RequestNumber – Frage eine Zahl ab

Synopsis: RequestNumber Default/K/N,Title/K,Text/K/A,Var/K
RC = *Erfolg*
Result = *Angegebene Zahl*

Funktion: Fragt den Benutzer nach einer ganzen Zahl. Der Benutzer kann den Requester abbrechen. Sie können mit dem Text-Argument weitere Informationen bereitstellen.

Argumente: DefaultValue - Wert des Gadgets beim Start. Ist 0 wenn nicht angegeben.
Title - Optional. Fenstertitel des Requesters.
Text - Zusätzlicher Text, der im Requester angezeigt wird. Kann Newlines (*n) enthalten. Bitte beachten Sie, daß dieses Argument unbedingt angegeben muß und das Schlüsselwort vorgangestellt werden muß. Dies ist für Kompatibelität mit zukünftigen Versionen von Fiasco, die dieses Argument nicht benötigen.

Ergebnisse: RC - Erfolg.
Result - Angegebene Zahl.

12.6.53 RequestString

Name: RequestString – Frage einen String ab

Synopsis: RequestString Default/K,Title/K,Text/K/A,Var/K
RC = *Erfolg*
Result = *Ausgewählter String*

Funktion: Fragt den Benutzer nach einem String. Der Benutzer kann den Requester abbrechen. Sie können mit dem Text-Argument weitere Informationen bereitstellen.

Argumente: DefaultValue - Wert des Gadgets beim Start. Ist leer wenn nicht angegeben.
Title - Optional. Fenstertitel des Requesters.

Text - Zusätzlicher Text, der im Requester angezeigt wird. Kann Newlines (*n) enthalten. Bitte beachten Sie, daß dieses Argument unbedingt angegeben muß und das Schlüsselwort vorgangestellt werden muß. Dies ist für Kompatibelität mit zukünftigen Versionen von Fiasco, die dieses Argument nicht benötigen.

Ergebnisse: RC - Erfolg.
Result - Angegebener String.

12.6.54 ResetStatus

Name: ResetStatus – Status-Gadget zurücksetzen

Synopsis: ResetStatus

Funktion: Setzt das Status-Gadget des Service-Fensters auf den normalen Inhalt. Dies ist RecordNummer / AlleRecords im Record-Modus oder X / Y im Masken-Modus. Sie sollten dieses Kommando benutzen, um die Status-Informationen, die mit SetStatus (siehe Abschnitt 12.6.64) eingestellt wurden, zurückzusetzen.

Argumente: Keine.

Ergebnisse: Keine.

12.6.55 RevealProject

Name: RevealProject – Zeige ein verborgenes Projekt an

Synopsis: RevealProject Project/K
RC = *Erfolg*

Funktion: Öffnet die Fenster eines verborgenen Fiasco-Projekts. Dieses Projekt wird das aktive werden. Wenn ein Projekt angegeben ist, wird dieses angezeigt, ansonsten wird das angesteuerte benutzt.

Argumente: Project - Dateiname oder voller Pfad des Projektes.

Ergebnisse: RC - Erfolg

12.6.56 Save

Name: Save – Datenbank Speichern

Synopsis: Save
RC = *Erfolg*

Funktion: Sichert die angegebene Datenbank unter dem alten Namen auf Disk.

Argumente: Keine.

Ergebnisse: RC - Erfolg.

12.6.57 SaveAs

Name: SaveAs – Datenbank Speichern.

Synopsis: SaveAs Name/K
RC = *Erfolg*

Funktion: Speicher die angesteuerte Datenbank unter einem angegebenen Namen auf Disk. Wenn Sie den neuen Namen nach dem Name-Schlüsselwort angeben, wird dieser Name benutzt. Wenn Sie Name nicht angeben, wird der Benutzer mit einem Dateirequester aufgefordet, einen neuen Namen anzugeben.

Argumente: Name - Neuer Dateiname. Wenn nicht angegeben, wird ein Datei-Requester geöffnet.

Ergebnisse: RC - Erfolg.

12.6.58 SaveSettings

Name: SaveSettings – Speichere Fiasco-Einstellungen (8)

Synopsis: SaveSettings File/A
RC = *Erfolg*

Function: Speichere die aktiven Fiasco-Einstellungen in die angegebene Datei.

Arguments: File - Datei, in die die Einstellungen gespeichert werden.

Results: RC - Erfolg

12.6.59 SetAttr

Name: SetAttr – Setze ein Fiasco-Attribut

Synopsis: SetAttr Object/A,Name/K,Attribute,Value/A
RC = *Erfolg*

Funktion: Setzt das angegebene Attribut.

Argumente: Object - Objekt-Typ der gesetzt werden soll.
Name - Bei manchen Objekt-Typen muß ein Name angegeben werden.
Attribute - Name des zu setzenden Attributs.
Value - Neuer Wert für das Attribut.

Ergebnisse: RC - Erfolg.

Objekte: Field - Daten für ein Fiasco-Feld in der angesteuerten Datenbank. Name (ID) muß angegeben werden. Kann nur im Masken-Modus benutzt werden.

Attribute für Field: Left - Linke Kante des Feldes.
Top - Obere Kante des Feldes.
Width - Breite des Feldes.
Height - Höhe des Feldes.
ARexxScript - ARexx-Script des Feldes.
PickerARexx - ARexx-Script für Auswahl-Schalter.
Formula - Formel für Feld.
ListWidth - Breite Kante im Listen-Fenster.
Shortcut - Tastatur-Abkürzung des Feldes.
InitContType - Typ des Startwerts. Einer von LAST, KEY oder OWN.
InitContValue - Wert des eigenen Startwerts.
MaxChars - Max. Zeichen des Feldes. Nicht von allen Feldtypen unterstützt.
MinValue - Minimum Wert des Feldes. Nicht von allen Feldtypen unterstützt.
MaxValue - Maximum Wert des Feldes. Nicht von allen Feldtypen unterstützt.
Format - Format des Feldes. Nicht von allen Feldtypen unterstützt.
Precision - Präzision des Feldes. Nicht von allen Feldtypen unterstützt.
Command - Kommando für Feld. Nicht von allen Feldtypen unterstützt.
Stack - Stack für Feld. Nicht von allen Feldtypen unterstützt.
Virtual - Status der Virtual-Option. 1 oder 0.
ListHidden - Ist das Feld in der Liste verborgen? 1 oder 0.
Hidden - Ist das Feld in der Maske verborgen? 1 oder 0.
ReadOnly - Ist das Feld nur lesbar? 1 oder 0.

12.6.60 SetConstant

Name: SetConstant - Setze eine Konstante **(7)**

Synopsis: SetConstant Name/A,Value/A/F
RC = *Erfolg*

Funktion: Setzt eine Konstante auf den angegebenen Wert in der angesteuerten Datenbank. Wenn eine Konstante mit gleichem Namen schon existieren sollte, wird diese überschrieben, ansonsten wird eine neue Konstante erzeugt.

Konstanten werden im Allgemeinen in Fiasco-Formeln benutzt, können aber auch für andere Zwecke in Anspruch genommen werden. Der Benutzer kann die Konstanten einer Datenbank mit dem Konstanten-Requester (siehe Abschnitt 10.6.9) betrachten und ändern.

Wenn die Formeln der Datenbank die neue bzw. geänderte Konstante benutzen sollen, müssen Sie RecompileFormulas (siehe Abschnitt 12.6.48) nach diesem Kommando aufrufen.

Argumente: Name - Name der zu setzenden Konstante.
Value - Wert für die Konstante.

Ergebnisse: RC - Erfolg.

12.6.61 SetField

Name: SetField - Feldinhalt ändern

Synopsis: SetField FieldID,Record/K/N,ListEntry/K/N,CreateListEntries/S,ExtFormat/S,Stem/K,Cont/F
RC = *Erfolg*

Funktion: Setzt den Inhalt des angegebenen Feldes im aktiven oder angegebenen Record auf den angegebenen Inhalt.

Wenn Stem benutzt wird, wird Fiasco die Werte der ARexx-Variablen mit den Namen *StemName.FeldID* in die jeweiligen Felder kopieren. Wenn eine Variable nicht gesetzt ist, wird das Feld nicht verändert. Nützlich mit GetField Stem (siehe Abschnitt 12.6.29).

Einträge in Listview-Feldern werden normalerweise von diesem Kommando weder erzeugt, noch entfernt. Wenn Sie SetField Stem MyStem mit MyStem auf vier Einträge initialisiert auf ein Listview-Feld mit drei Einträgen anwenden, werden nur die ersten drei kopiert und der

vierte Eintrag wird ignoriert. Wenn MyStem mit weniger Einträgen als im Listview initialisiert ist, werden die letzten Einträge im Listview unverändert bleiben. In Gegensatz dazu sorgt der CreateListEntries-Schalter dafür, daß beim Aufruf dieses Kommandos, alle alten Einträge entfernt werden und vollständig durch die in der Stem-Variablen angegebenen Einträge ersetzt werden.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: FieldID - ID eines einzelnen zu setzenden Feldes
Record - Nummer des Records
ListEntry - Nur bei einem Listview-Feld: Nummer des zu setzenden Eintrags. Zählt von 1 an.
CreateListEntries - Erklärung siehe oben. Nur in Kombination mit Stem anwendbar. Benötigt Fiasco 2.2. ExtFormat - Wenn angegeben, benutzt SetField das erweiterte Format, das von GetField ExtFormat (siehe Abschnitt 12.6.29) zurückgegeben wird. Neu in Fiasco 2.2. Cont - Neuer Inhalt für das Feld. Dieses Argument nimmt die gesamte restliche Kommandozeile ein, inclusive Leerzeichen. Die Interpretation dieses Arguments hängt von Feldtypen ab (In Klammern: ExtFormat):
String - wird direkt übernommen
Integer - Zahlen werden direkt übernommen, anderes ergibt 0
Float - entspricht Integer
Boolean - 1 oder TRUE = gesetzt, 0 oder FALSE = nicht gesetzt
Slider - Zahl wird übernommen, zu große bzw. zu kleine werden angepaßt.
Cycle - Zahl oder Name des Eintrags wird übernommen
Date - Datum im Format TT.MM.[JJ]JJ (in aktuellen Locale-Format)
Time - Zeit im Format HH:MM:SS (im aktuellen Locale-Format)
Extern - wird direkt übernommen
Datat. - wird direkt übernommen
Var String - wird direkt übernommen. Der String *n wird in einen Zeilenumbruch umgewandelt. Dies funktioniert jedoch nur, wenn das Argument in Anführungszeichen angegeben wird.

Wenn das Feld ein Listview-Feld ist, benutzen Sie ListEntry, um den Inhal eines bestimmten Eintrags zu setzen. Benutzen Sie dann das Format des unterliegenden Feldtyps. Um Einträge einem Listview hinzuzufügen, benutzen Sie entweder den CreateListEntries-Schalter oder AddLVFieldEntry (siehe Abschnitt 12.6.5).

Ergebnisse: RC - Erfolg

Bemerkungen: Vor Release 2.2 behauptete dieses Dokument, daß die Daten aus Datums- und Zeit-Feldern im aktuellen Locale-Format

benötigt würden. Es ist und war jedoch der Fall, daß - es sei denn es wird ExtFormat benutzt - diese Werte unabhängig von den Locale-Einstellungen formatiert werden.

12.6.62 SetMode

Name: SetMode - Wähle den Bearbeitungs-Modus

Synopsis: SetMode Mask=MaskMode/S,Record=RecordMode/S
RC = *Erfolg*

Funktion: Aktiviert den angegebenen Modus für das angesteuerte Projekt.

Argumente: MaskMode - Aktiviere Masken-Modus.
RecordMode - Aktiviere Record-Modus.

Ergebnisse: RC - Erfolg.

12.6.63 SetSearchField

Name: SetSearchField - Füge ein Feld einem SearchInfo hinzu

Synopsis: SetSearchField
SearchInfo/K/A,FieldID/K/A,Blur/K/N,Pattern/K/A/F
RC = *Erfolg*

Funktion: Fügt ein Feld und das Muster zu einem SearchInfo im Feld-Modus hinzu. Das angegebene Feld *muß* auf das Muster passen, damit das SearchInfo auf einen Record paßt.

Argumente: SearchInfo - Name des zu ändernden SearchInfos.
FieldID - ID des zu benutzenden Feldes.
Blur - Faktor für unscharfe Suche. Wenn nicht angegeben, wird unscharfe Suche ausgeschaltet.
Pattern - Suchmuster für das Feld.

Ergebnisse: RC - Erfolg

12.6.64 SetStatus

Name: SetStatus - Zeige einen Status-String

Synopsis: SetStatus Text/A
RC = *Erfolg*

Funktion: Zeigt den übergebenen String im Status-Gadget des Service-Fensters (siehe Abschnitt 10.3) an. Wenn das Service-Fenster nicht offen ist, passiert nichts.

Argumente: Text - Der anzuzeigende String.

Ergebnisse: RC - Erfolg.

12.6.65 Sort

Name: Sort – Sortiere eine Datenbank

Synopsis: Sort Fields/A/M,Descending/S,Index/K
RC = *Erfolg*

Funktion: Sortiert die Records des aktiven Projektes nach alphabetischer oder entsprechender Reihenfolge der Inhalte der angegebenen Felder.

Dieses Kommando kann nur im Record-Modus aufgerufen werden.

Argumente: Fields - Die Felder nach dennen die Datenbank sortiert wird. Das erste Feld hat die höchste Prioriät.
Descending - Sortiere rückwärts.
Index - Name des zu erstellenden Index. Standard: ARexxSort.fidx

Ergebnisse:

12.6.66 UnlockGUI

Name: UnlockGUI - Entriegele Fiascos GUI

Synopsis: UnlockGUI
RC = *Erfolg*

Funktion: Schließt die GUI von Fiasco auf, die vorher durch LockGUI (siehe Abschnitt 12.6.34) verschlossen wurde. Der Benutzer hat dann wieder Zugriff auf Fiasco. LockGUI und UnlockGUI können verschachtelt werden.

Argumente: Keine.

Ergebnisse: RC - Ungleich Null wenn GUI nicht verschlossen war.

Kapitel 13

Beispiel-Projekte

Im Verzeichnis Databases liegen mehrere Fiasco-Projekte, die z.T. auch so für eigene Zwecke weiterverwendet werden können. Hier folgen die Beschreibungen dieser Projekte:

13.1 Organizer

Das Organizer-Verzeichnis enthält zwei Datenbanken: Eine Adress-Buch-Datenbank und eine Termine-Datenbank, die als Erinnerungs-Programm benutzt werden kann.

Das Adress-Buch ist eine erweiterte Version der Adress-Datenbank der vorigen Fiasco-Release-Versionen. Es enthält die klassischen Felder wie Name, Adresse, Telefon-Nummer, etc.

Die Felder für Telefon, Fax und Postleitzahl sind String-Felder, da hier auch Sonderzeichen wie "/" oder eine "0" am Anfang beachtet werden müssen.

Weiterhin gehören zu den Feldern Telefon, Fax und eMail Schalter, die ARexx-Scripts aufrufen. Das Script für das Telefon-Feld ruft das Script dial.frx auf, das versucht, die Telefon-Nummer mit einem Modem anzuwählen. Wenn die Verbindung aufgebaut wird, öffnet sich ein Requester mit dem Gadget Hang up. Dann können Sie den Telefonhörer abheben und Hang up anklicken, um das Gespräch am Telefon zu empfangen. Mehr Details über dial.frx stehen als Kommentare direkt im Script.

Der Fax-Schalter kann zum Senden eines Faxes benutzt werden. Das Script fax.frx, das von dem Schalter aufgerufen wird, unterstützt z. Zt. nur das Programm STFax. Wenn Sie ein anderes Fax-Programm mit ARexx-Port habem, sollte das Einfügen von Unterstützung für dieses Programm

nicht schwer sein.

Der Mail-Schalter ruft über ARexx ein eMail-Programm auf, um eine Mail an die Adresse zu schicken. Das Script mail.frx unterstützt z. Zt. nur den Mailer YAM von Marcel Beck.

Die andere Datenbank im Verzeichnis kann zum Verwalten von Terminen benutzt werden. Geben Sie einfach Ihre Termine ein. Sie können entweder ein festes Datum oder nur einen Wochentag für einen Termin angeben. Ob Sie das Datum oder den Wochentag benutzen wollen, wird von den Boolean-Feldern an der rechten Seite der jeweilgen Felder bestimmt.

Der Schalter Check Appointments durchsucht die Datenbank nach aktuellen Terminen. Mit dem Ann.-Feld können Sie die Zeit in Tagen vor dem Termin angeben, ab der Sie auf den Termin hingewiesen werden.

13.2 Stammbaum

Der Stammbaum besteht aus den Projekten "persons.fdb" und "families.fdb". In "persons.fdb" müssen alle Personen, die im Stammbaum vorkommen sollen, aufgeführt werden. Hier können auch Geschlecht, Beruf, Geburtsdatum, usw. eingetragen werden.

Diese Daten werden dann von "families.fdb" über Relationen benutzt, um Namen von Ehepartnern, Kindern zu holen. Weiterhin sind hier noch Datums- und String-Felder für Heirat und Scheidung. Durch die intensive Nutzung von Relationen gibt es in diesem Projekt nur noch 7 "echte" Felder, die auf Disk abgelegt werden. Die restlichen 6 werden aus "persons.fdb" geladen.

13.3 PD-Disks

Diese Datenbank speichert die Beschreibungen von Programmen auf PD-Serien, wie den Fish-Disks. Die Datenbank enthält Felder für den Seriennamen, die Disketten-Nummer, Programmnamen, Programmversion, Beschreibung und den Autor.

Der Schalter Scan disk kann zum automatischen Importieren von Contents-Dateien einer Serie in die Datenbank benutzt werden. Das ReadFish.rexx-Script unterstützt z.Zt. die Formate der Fish-Disks, SaarAG-Disks und der PD-Disks des deutschen Amiga-Magazins. Es unterstützt auch die Contents-Dateien, die zu einer großen Datei zusammengesetzt wurden.

13.4 Videos

In der Video-Datenbank können Homevideos komfortabel verwaltet werden. Die Datenbank besteht aus zwei Projekten: “movies.fdb” und “tapes.fdb”. In “movies” werden Daten zu einzelnen Filmen gespeichert, u.a. Genre, Regie, Darsteller und Jahr. Über das Feld “Kasette” wird jeder Film einer Kasette zugeordnet, die in “tapes” zu finden ist. Hier sind Marke und Länge der Kasette definiert. Über ein ARexx-Script wird dann die restliche freie Zeit jeder Kasette berechnet.

13.5 Bilder-Datenbank

Für Fiasco 2.0 wurde eine komplett neue Bilder-Datenbank entwickelt. Da sie Datatypes.Felder benutzt, benötigen Sie Amiga OS 3.0, sie zu benutzen. Die Maske enthät acht Datatypes-Felder, die zum Anzeigen von Miniatur-Bildern benutzt werden. Die Bilder-Datenbank wird durch ARexx-Scripts kontrolliert.

Wenn die Datenbank eine neue Datei einfügt, was mit Add Dir oder Add File erfolgen kann, erzeugt das Programm CreateThumbnail eine kleinere Version des Bildes und legt es im IFF-Format im TN-Verzeichnis ab. CreateThumbnail benutzt auch die datatypes.library. Deshalb können Sie alle Bild-Formate benutzen, für die Sie auch Datatypes-Treiber installiert haben. CreateThumbnail kann jedoch keine HAM oder EHB verarbeiten.

Mit Add File können Sie ein Bild mit einem Datei-Requester auswählen, das automatisch in die Datenbank eingefügt wrid.

Add Dir dient zum Einfügen der Bilder eines ganzen Verzeichnisses. Nachdem Sie das Verzeichnis zum Durchsuchen angegeben haben, öffnet sich ein anderer Requester, der Sie fragt, ob Sie nur das ausgewähltes Verzeichnis, alle Unterverzeichnisse oder auch die Lha-Archive darin scannen wollen. Die Bilder-Datenbank verarbeitet Lha-Archive transparent, Sie brauchen keine Archiv-Dateisystem zu installieren. Mit einigen Änderungen an den ARexx-Scripte können Sie auch andere Archiv-Programme wie LZX benutzen.

Unter jedem Datatypes-Feld sind drei Buttons, mit denen Sie das Feld kontrollieren können. Dis startet MultiView, um das Bild anzuzeigen. Natürlich können Sie auch ein anderes Programm einstellen, indem Sie das ARexx-Script display.rexx ändern. Inf zeigt zusätzliche Informationen über das Bild an. Dazu gehören Größe und Tiefe. Von hier können Sie auch das Bild mit VT anzeigen[1] oder das Bild an eine andere Stelle kopieren.

[1] Sie müssen VT natürlich im Suchpfad haben

Der Search Button oben rechts kann zum Suchen nach einem Dateinamen benutzt werden.

13.6 Multimedia-Datenbank

Dies ist eine Datenbank, die die Multimedia-Eigenschaften von Fiasco für eine Art von Lexikon nutzt. Die Daten, die als Beispiel in dieser Datenbank einbegriffen sind, sind jedoch etwas spärlich für ein Lexikon. Auf jeden Fall zeigt diese Datenbank aber was möglich ist und ist für Ihre eigenen Daten offen.

Jeder Record repräsentiert einen Eintrag für einen Begriff. Dieser Begriff kann in dem Feld Term oben links in der Maske angegeben werden. Unter diesem Feld ist ein Var String-Feld das für einen längeren Text genutzt werden kann, der diesen Punkt beschreibt.

Kein Lexikon ohne Querverweise. Das Listview-Feld unten links kann mehrere andere Begriffe dieser Datenbank enthalten. Um irgendeinen Verweis anzuspringen, müssen Sie diesen im Listview aktivieren und auf den Goto Cross Reference Schalter klicken. Dieser startet ein ARexx-Script das Fiascos Such-Funktion benutzt, um nach dem angegebenen Begriff zu suchen. Das Cross References Listview benutzt das Nur Auswählen Feld-Attribut. Wenn Sie die Einträge verändern wollen, müssen Sie dieses Attribut deaktivieren.

Der rechte Teil der Maske dient für die “Multimedia”. Für jeden Record können Sie Documents angeben. Wenn Sie einen Eintrag in diesem Listview auswählen, wird dieser automatisch in das Datatypes-Feld mit der ID Document_Display kopiert. Die Dokumente brauchen nicht nur Bilder zu sein, Sie können auch Klang-Dateien oder alle anderen Dateien, für die Datatypes verfügbar sind angeben.

Der Mechanismus hinter diesem Multimedia-Teil ist etwas komplizierter als die anderen Teile der Mask, aber auch einfach zu verstehen. Das Documents Listview zeigt nur frei verteilbare Namen für die Dokumente an. Dann gibt es ein anderes Listview, das normalerweise verborgen ist. Seine ID ist Doc_Files. Dort müssen Sie die Dateinamen für alle Dokumente angeben, die im Documents Listview angegeben sind. Um das ausgewählte Dokument in dem Datatypes-Feld anzuzeigen, wird eine Formel benutzt. Diese ist im Datatypes-Feld angeben, das auch das Virtuell-Attribut gesetzt hat, da der Inhalt dieses Feldes nur für die Laufzeit wichtig ist. Die Formel lautet so:

```
active(documents) != -1 ? doc_files[active(documents)] : ""
```

Das bedeutet, daß das Datatypes-Feld schaut, ob ein Eintrag im Documents Listview ausgewählt ist. Wenn einer aktiv ist, wird der Dateiname von der gleichen Position im doc_files Listview in das Datatypes-Feld kopiert, das dann die Datei anzeigt. Wenn kein Eintrag aktiv ist, wird ein leerer String kopiert, was bewirkt, daß keine Datei angezeigt wird.

Um diese Daten zu verändern, müssen Sie das Nur Auswählen Attribut das Documents Listview ausschalten und sie müssen das Doc_files Listview hervorholen.

13.7 Mailing List Archiv

Wenn Sie eine Mailing-Liste betreiben, kann diese Datenbank für sie nützlich sein. Sie hilft Ihnen, alle Nachrichten aus der Liste zu archivieren und – was der wichtigste Punkt ist – sie erzeugt automatisch eine HTML-Version des Archivs, die sofort auf Web-Seiten benutzt werden kann. Somit können die Besucher Ihrer Web-Seite den Inhalt der Mailing-Liste komfortabel betrachten.

Um die Datenbank zu erzeugen müssen Sie die Mails aus Ihrem Mail-Programm exportieren. Dabei müssen Sie beachten, daß die Mails die kompletten Header beinhalten. Ansonsten kann das Projekt die Mails nicht importieren. Mit MicroDot II können Sie die Mails inclusive Header so exportieren: Öffnen Sie das Anzeige-Fenster für die jeweilige Mail, aktivieren (whole message) im Listview oben rechts und klicken auf Save.

Um diese Mail in das Mailing-Listen-Archiv zu importieren, klicken Sie auf den Button Import Mail im Masken-Fenster. Sie werden aufgefordert, die zu importierende Datei anzugeben. Wenn Sie mehrere Mail-Dateien in ein Verzeichnis exportiert haben, können Sie den Scan Directory Buttom benutzen. Wenn das Mailing-List-Archiv die Mails importiert, erzeugt es für jede Mail einen Record. Die Header-Daten für jede Mail werden auf der rechten Seite der Maske angezeigt, während der Mail-Text an der unteren Seite angezeigt wird.

Das Fiasco Mailing-List-Archiv erlaubt es Ihnen, das Layout der erzeugten HTML-Seiten nach eigenen Wünschen anzupassen. Überschriften und Fußzeilen aller erzeugten Seiten können in den Dateien tpl1.html und tpl2.html definiert werden. Diese Dateien werden am Beginn bzw. am Ende aller Seiten eingefügt. Die Fiasco-Distribution enthält Beispiel-Versionen dieser Dateien. Der Name der Mailing-Liste, Dokument-Farben und einige andere Einstellungen können direkt in der createhtml.frx Datei angegeben werden, das auch im Projekt-Verzeichnis liegt.

Um die HTML-Seiten zu erzeugen, klicken Sie auf den Createl HTML Button. Es ist empfehlenswert, ein eigenes Verzeichnis für die Ausgabe des

Archivs anzulegen. Nach dem Exportieren kann die Hauptseite mit *Verzeichnis*/index.html geladen werden. Somit können Sie die Seiten mit jedem WWW-Browser lokal auf Ihrem Computer betrachten. Mit einem FTP-Programm können Sie die Seiten dann auf Ihre Internet-Site übertragen.

Ein reelles Beispiel können Sie sich auf der Fiasco-Support-Site anschauen:
http://www.amigaworld.com/support/fiasco

Anhang

Anhang A

Rechtliches

A.1 Nutzungsbedingungen

Das Programm "Fiasco" und zugehörige Daten werden "wie sie sind" zur Verfügung gestellt. Der Autor übernimmt keinerlei Gewährleistung für Fehlerfreiheit, Genauigkeit, Verläßlichkeit oder Freiheit von sonstigen Mängeln der oben genannten Programme und Daten, weder ausdrücklich, noch stillschweigend. In keinem Falle kann ich für irgendwelche Schäden oder Datenverluste, die von diesem Programm verursacht wurden, verantwortlich gemacht werden.

Wenn Sie wichtige Daten mit Ihrem Computer verwalten, sollten Sie in jedem Fall Sicherheitskopien dieser Daten erstellen!

A.2 Copyright

Fiasco ist *nicht* Public Domain. Ich behalte mir jegliche Urheberrechte vor. Fiasco Copyright © 1995-1998 Nils Bandener.

Fiasco darf frei verbreitet werden, solange folgende Bedingungen erfüllt sind:

- Das Programm-Paket muß komplett sein. Seit Release 2.0 ist das Programm-Paket aufgrund der Größe in mehrere Teile aufgeteilt. Das Archiv Fiasco_main.lha enthält das Hauptprogramm, Libraries, Locale-Kataloge und Beispiel-Datenbanken. Die Archive Fiasco_doc_eng.lha, Fiasco_doc_deu.lha und Fiasco_doc_ita.lha enthalten die Fiasco-Anleitung in den entsprechenden Sprachen im AmigaGuide- und TeX-DVI-Format. Der Abschitt Datei-Liste enthält

eine Auflistung aller Archive der Fiasco 2.2 Distribution. Wegen der seperaten Archive für die Dokumentation enthält das Haupt-Archiv keine Anleitung. Daher **müssen** Sie mindestens eines der Sprachen-Archive mit verbreiten. Dennoch wird immernoch empfohlen, alle Archive zu verbreiten, insbesondere wenn Sie Fiasco auf einer CD-ROM verbreiten. Das Verbreiten von Fiasco in unarchivierter Form ist solange erlaubt, wie Sie die obigen Bedingungen einhalten.

- Fiasco darf ohne *schriftliche* Genehmigung des Autors nicht für kommerzielle Zwecke verkauft werden. Dies schließt den Vertrieb von Fiasco zu überhöhten Preisen ein. Es dürfen lediglich Kosten für Medien und Kopieren erhoben werden. Der Vertrieb auf CD-ROMs ist gestattet, solange der Preis nicht DM 30 oder USD 20 übersteigt. Der Vertrieb auf Cover-Disks oder Cover-CDs ist gestattet, solange der Preis nicht DM 12 oder USD 10 im Fall von Disketten oder DM 16 oder USD 12 im Fall von CDs übersteigt.

Hiermit gebe ich die besondere Erlaubnis, Fiasco auf den "Meeting Pearls" CD-ROMs und auf den "Aminet" CD-ROMs zu verbreiten.

Es gibt eine spezielle Disketten-basierende Distribution von Fiasco 2.2. Sie besteht aus zwei Disketten mit archiviertem Inhalt und einem Installer-Script, das angepasst wurde, um das Entpacken und Installieren automatisch vorzunehmen. Diese Distribution ist nicht im Aminet verfügbar. Wenn Sie darin interessiert sind, schicken Sie mir einen Brief und die Versandkosten, die im Shareware-Abschnitt aufgeführt sind. Sie bekommen diese Distribution auch, wenn Sie eine registrierte Fiasco-Version auf DD-Disketten bestellen.

Falls Sie Fiasco in Ihre PD-Sammlung, Coverdisk oder ähnliches aufgenommen haben und am Schluß ein Exemplar übrig bleibt, dürfen Sie gerne mir dieses schicken.

Das Keyfile, das Sie nach der Registrierung für Fiasco bekommen, ist nicht frei verteilbar. Es darf nur auf Ihrem eigenen Computer-System benutzt werden. Manipulationen am Keyfile sind auch untersagt.

A.3 Weitere Copyrights

textfield.gadget 3.1 ist Copyright © 1995 Mark Thomas. Die komplette textfield.gadget Distribution, inclusive Programmier-Unterlagen, kann im Aminet oder auf der Aminet Set 2C CD-ROM unter dev/gui/textfield.lha gefunden werden.

gtlayout.library ist von Olaf Barthel. Die Library darf frei verteilt und

frei benutzt werden. Neue Versionen der gtlayout.library erscheinen regelmäßig mit 'term'.

WBPath ist Copyright © 1994 Ralph Babel. Die WBPath Programmierer-Unterlagen werden mit Fiasco im Development/WBPath Verzeichnis verteilt.

LX wurde von Jonathan Forbes geschrieben und ist Copyright © 1993 Xenomiga Technology. LX wird vom Installer-Script der Fiasco-Disketten-Distribution benutzt. Die komplette LX-Distribution kann im Aminet oder auf der Aminet Set 2A CD-ROM unter util/arc/lx103.lha gefunden werden.

Installer und das Installer-Projekt-Icon sind Copyright © 1995-1996 ESCOM AG. Alle Rechte vorbehalten. Reproduziert und verbreitet unter Lizenz von ESCOM AG.

Installer Software wird "wie sie ist" abgeben und unterliegt Änderungen. Es werden keine Garantien gemacht. Sämtliche Benutzung geschieht auf Ihr eigenes Risiko. Es wird keine Verantwortung angenommen.

A.4 Shareware

Seit Release 2.0 ist Fiasco Shareware.. Das bedeutet, daß Sie Fiasco für eine Zeitspanne von 30 Tagen testen dürfen. Wenn Sie nach dieser Zeitspanne Fiasco weiter benutzen wollen, müssen sie sich für Fiasco registrieren lassen. Sie werden ein "Keyfile" bekommen, das Ihnen erlaubt, mehr als 15 Records zu speichern.

Preisliste

Registriergebühr für Fiasco 2.2	USD	25,00	DM	30,00
Versandkosten Europa	USD	4.00	DM	5.00
Versandkosten International	USD	6.00	DM	8.00
Versandkosten eMail		frei		frei

Versandkosten sind inklusive zwei DD Disketten oder einer HD Diskette, die das Keyfile und die neueste Release-Version von Fiasco enthalten. Sie können sich auch entscheiden, das Keyfile per eMail zugeschickt zu bekommen. In diesem Fall müssen Sie keine Versandkosten zahlen. Sie bekommen jedoch nur das Keyfile, nicht die neueste Fiasco-Version.

Zahlungsweisen

Es gibt drei Wege, Fiasco zu bezahlen:

- Bargeld: Fügen Sie einfach Ihrem Brief das Geld bei.

- Euro-Cheques: Fügen Sie den Cheque Ihrem Brief bei. Bitte senden Sie keine anderen Cheques!
- Überweisung auf mein Bankkonto:
  Kasseler Sparkasse; BLZ 520 503 53; Konto-Nr. 1100353258
  Bitte geben Sie auf Ihrer Überweisung Namen und Adresse an. Dies ist die einzig mögliche Bezahlungsweise, wenn Sie per eMail bestellen. Die Lieferung wird begonnen, wenn das Geld eingegangen ist.

Die einzigen akzeptierten Währungen sind DM oder US Dollar.

Bitte benutzen Sie das Registrierungs-Formular, das der Fiasco-Distribution beiliegt. Senden sie es ausgefüllt an eine der folgenden Adressen:

Nils Bandener
Dekanatsgasse 4
D-34369 Hofgeismar
Deutschland

Internet: Nils@dinoex.sub.org

Einsender von Gifts für Fiasco 1.x

Wenn Sie mir für Fiasco 1.x bis zum 31.12.1996 ein Geschenk geschickt haben, können Sie ein kostenloses Keyfile für Fiasco 2.2 bekommen. Sie müssen nichts bezahlen, wenn Sie das Keyfile per eMail zugeschickt bekommen wollen. Wenn Sie es über normale Post zugeschickt bekommen wollen, müssen Sie die Versandkosten bezahlen, die oben aufgeführt sind.

A.5 Datei-Liste

Die Fiasco Release 2.2 Distribution besteht aus diesen Dateien:
Archiv Fiasco_main.lha:

```
Fiasco_2.2/ARexx.info
Fiasco_2.2/ARexx/age.frx
Fiasco_2.2/ARexx/age.frx.info
Fiasco_2.2/ARexx/arexxprint.rexx
Fiasco_2.2/ARexx/arexxprint.rexx.info
Fiasco_2.2/ARexx/cap.frx
Fiasco_2.2/ARexx/cap.frx.info
Fiasco_2.2/ARexx/converttolistview.frx
Fiasco_2.2/ARexx/converttolistview.frx.info
Fiasco_2.2/ARexx/dbstructure.frx
Fiasco_2.2/ARexx/dbstructure.frx.info
Fiasco_2.2/ARexx/dummy.frx
```

```
Fiasco_2.2/ARexx/dummy.frx.info
Fiasco_2.2/ARexx/graphprint.frx
Fiasco_2.2/ARexx/graphprint.frx.info
Fiasco_2.2/ARexx/importcolumn.frx
Fiasco_2.2/ARexx/importcolumn.frx.info
Fiasco_2.2/ARexx/print.frx
Fiasco_2.2/ARexx/print.frx.info
Fiasco_2.2/ARexx/requestdt.frx
Fiasco_2.2/ARexx/requestdt.frx.info
Fiasco_2.2/ARexx/unlockgui.frx
Fiasco_2.2/ARexx/unlockgui.frx.info
Fiasco_2.2/Catalogs/Dansk/Fiasco.catalog
Fiasco_2.2/Catalogs/deutsch/fiasco.catalog
Fiasco_2.2/Catalogs/español/fiasco.catalog
Fiasco_2.2/Catalogs/Italiano/fiasco.catalog
Fiasco_2.2/Catalogs/svenska/fiasco.catalog
Fiasco_2.2/Databases.info
Fiasco_2.2/Databases/Addresses2.info
Fiasco_2.2/Databases/Addresses2/Adressen.fdb
Fiasco_2.2/Databases/Addresses2/Adressen.fdb.info
Fiasco_2.2/Databases/Addresses2/Adressen.frec
Fiasco_2.2/Databases/Addresses2/Adressen/Standard.fidx
Fiasco_2.2/Databases/Addresses2/Adressmanager-Konv.rexx
Fiasco_2.2/Databases/Addresses2/Adressmanager-Konv.rexx.info
Fiasco_2.2/Databases/Aminet.info
Fiasco_2.2/Databases/Aminet/Aminet.fdb
Fiasco_2.2/Databases/Aminet/Aminet.fdb.info
Fiasco_2.2/Databases/Aminet/Aminet.frec
Fiasco_2.2/Databases/Aminet/Aminet/Standard.fidx
Fiasco_2.2/Databases/Aminet/copyarc.frx
Fiasco_2.2/Databases/Aminet/extract.frx
Fiasco_2.2/Databases/Aminet/Scancont.frx
Fiasco_2.2/Databases/FamilyTree.info
Fiasco_2.2/Databases/FamilyTree/Families.fdat
Fiasco_2.2/Databases/FamilyTree/Families.fdb
Fiasco_2.2/Databases/FamilyTree/Families.fdb.info
Fiasco_2.2/Databases/FamilyTree/Families.frec
Fiasco_2.2/Databases/FamilyTree/Families/Standard.fidx
Fiasco_2.2/Databases/FamilyTree/Persons.fdb
Fiasco_2.2/Databases/FamilyTree/Persons.fdb.info
Fiasco_2.2/Databases/FamilyTree/Persons.frec
Fiasco_2.2/Databases/FamilyTree/Persons/Standard.fidx
Fiasco_2.2/Databases/GraphDemo.info
Fiasco_2.2/Databases/GraphDemo/Fragments.fdb
Fiasco_2.2/Databases/GraphDemo/Fragments.fdb.info
Fiasco_2.2/Databases/GraphDemo/Fragments.frec
Fiasco_2.2/Databases/GraphDemo/Fragments/Standard.fidx
Fiasco_2.2/Databases/Mailing List Archive.info
Fiasco_2.2/Databases/Mailing List Archive/createhtml.frx
Fiasco_2.2/Databases/Mailing List Archive/CreateHTML.frx.info
Fiasco_2.2/Databases/Mailing List Archive/Example Output.info
Fiasco_2.2/Databases/Mailing List Archive/Example Output/1.html
```

```
Fiasco_2.2/Databases/Mailing List Archive/Example Output/2.html
Fiasco_2.2/Databases/Mailing List Archive/Example Output/index.html
Fiasco_2.2/Databases/Mailing List Archive/importmails.frx
Fiasco_2.2/Databases/Mailing List Archive/MLArchive.fdat
Fiasco_2.2/Databases/Mailing List Archive/MLArchive.fdb
Fiasco_2.2/Databases/Mailing List Archive/MLArchive.fdb.info
Fiasco_2.2/Databases/Mailing List Archive/MLArchive.frec
Fiasco_2.2/Databases/Mailing List Archive/MLArchive/Standard.fidx
Fiasco_2.2/Databases/Mailing List Archive/scandir.frx
Fiasco_2.2/Databases/Mailing List Archive/tpl1.html
Fiasco_2.2/Databases/Mailing List Archive/tpl1.html.info
Fiasco_2.2/Databases/Mailing List Archive/tpl2.html
Fiasco_2.2/Databases/Mailing List Archive/tpl2.html.info
Fiasco_2.2/Databases/Multimedia.info
Fiasco_2.2/Databases/Multimedia/A4000T.iff
Fiasco_2.2/Databases/Multimedia/Amiga.iff
Fiasco_2.2/Databases/Multimedia/gotoxref.frx
Fiasco_2.2/Databases/Multimedia/MMEnc.fdat
Fiasco_2.2/Databases/Multimedia/MMEnc.fdb
Fiasco_2.2/Databases/Multimedia/MMEnc.frec
Fiasco_2.2/Databases/Multimedia/MMEnc/Standard.fidx
Fiasco_2.2/Databases/Organizer.info
Fiasco_2.2/Databases/Organizer/Addresses.fdat
Fiasco_2.2/Databases/Organizer/Addresses.fdb
Fiasco_2.2/Databases/Organizer/Addresses.fdb.info
Fiasco_2.2/Databases/Organizer/Addresses.frec
Fiasco_2.2/Databases/Organizer/Addresses/Standard.fidx
Fiasco_2.2/Databases/Organizer/Appointments.fdat
Fiasco_2.2/Databases/Organizer/Appointments.fdb
Fiasco_2.2/Databases/Organizer/Appointments.fdb.info
Fiasco_2.2/Databases/Organizer/Appointments.frec
Fiasco_2.2/Databases/Organizer/Appointments/Standard.fidx
Fiasco_2.2/Databases/Organizer/checkappointments.frx
Fiasco_2.2/Databases/Organizer/dial.frx
Fiasco_2.2/Databases/Organizer/fax.frx
Fiasco_2.2/Databases/Organizer/Labels.fpr
Fiasco_2.2/Databases/Organizer/Labels.fpr.info
Fiasco_2.2/Databases/Organizer/ListLaTeX.fpr
Fiasco_2.2/Databases/Organizer/ListLaTeX.fpr.info
Fiasco_2.2/Databases/Organizer/mail.frx
Fiasco_2.2/Databases/PD-Disks.info
Fiasco_2.2/Databases/PD-Disks/Disks.fdat
Fiasco_2.2/Databases/PD-Disks/Disks.fdb
Fiasco_2.2/Databases/PD-Disks/Disks.fdb.info
Fiasco_2.2/Databases/PD-Disks/Disks.frec
Fiasco_2.2/Databases/PD-Disks/Disks/Standard.fidx
Fiasco_2.2/Databases/PD-Disks/ReadFish.rexx
Fiasco_2.2/Databases/PD-Disks/ReadFish.rexx.info
Fiasco_2.2/Databases/PictureDatabase.info
Fiasco_2.2/Databases/PictureDatabase/AddPicture.frx
Fiasco_2.2/Databases/PictureDatabase/CreateThumbnail
Fiasco_2.2/Databases/PictureDatabase/DelPicture.frx
```

```
Fiasco_2.2/Databases/PictureDatabase/Display.frx
Fiasco_2.2/Databases/PictureDatabase/PicInfo.frx
Fiasco_2.2/Databases/PictureDatabase/Pictures.fdb
Fiasco_2.2/Databases/PictureDatabase/Pictures.fdb.info
Fiasco_2.2/Databases/PictureDatabase/Pictures.frec
Fiasco_2.2/Databases/PictureDatabase/Pictures/Standard.fidx
Fiasco_2.2/Databases/PictureDatabase/ScanDir.frx
Fiasco_2.2/Databases/PictureDatabase/SearchPicture.frx
Fiasco_2.2/Databases/PictureDatabase/StartProg
Fiasco_2.2/Databases/PictureDatabase/TN/TN_1_0
Fiasco_2.2/Databases/PictureDatabase/TN/TN_1_1
Fiasco_2.2/Databases/Videos.info
Fiasco_2.2/Databases/Videos/Movies.fdat
Fiasco_2.2/Databases/Videos/Movies.fdb
Fiasco_2.2/Databases/Videos/Movies.fdb.info
Fiasco_2.2/Databases/Videos/Movies.frec
Fiasco_2.2/Databases/Videos/Movies/Standard.fidx
Fiasco_2.2/Databases/Videos/Tapes.fdat
Fiasco_2.2/Databases/Videos/Tapes.fdb
Fiasco_2.2/Databases/Videos/Tapes.fdb.info
Fiasco_2.2/Databases/Videos/Tapes.frec
Fiasco_2.2/Databases/Videos/Tapes/Standard.fidx
Fiasco_2.2/Development.info
Fiasco_2.2/Development/Locale.info
Fiasco_2.2/Development/Locale/Fiasco.cd
Fiasco_2.2/Development/Locale/Fiasco.cd.info
Fiasco_2.2/Development/Locale/Fiasco.ct
Fiasco_2.2/Development/Locale/Fiasco.ct.info
Fiasco_2.2/Development/Locale/Locale.readme
Fiasco_2.2/Development/Locale/Locale.readme.info
Fiasco_2.2/Development/Locale/v5_v6_changes.txt
Fiasco_2.2/Development/Locale/v5_v6_changes.txt.info
Fiasco_2.2/Development/Locale/v6_v8_changes.txt
Fiasco_2.2/Development/Locale/v6_v8_changes.txt.info
Fiasco_2.2/Development/WBPath.info
Fiasco_2.2/Development/WBPath/pathtest
Fiasco_2.2/Development/WBPath/pathtest.c
Fiasco_2.2/Development/WBPath/pathtest.c.info
Fiasco_2.2/Development/WBPath/pathtest.info
Fiasco_2.2/Development/WBPath/wbpath.h
Fiasco_2.2/Development/WBPath/wbpath.h.info
Fiasco_2.2/Development/WBPath/wbpath.o
Fiasco_2.2/Development/WBPath/wbpath.o.info
Fiasco_2.2/Documentation.info
Fiasco_2.2/Fiasco
Fiasco_2.2/Fiasco.info
Fiasco_2.2/gadgets/textfield.gadget
Fiasco_2.2/gtlayout.library
Fiasco_2.2/icons/ARexx.info
Fiasco_2.2/icons/ARexxScript.info
Fiasco_2.2/icons/Databases.info
Fiasco_2.2/icons/def_FiascoPrint.info
```

```
Fiasco_2.2/icons/Documentation.info
Fiasco_2.2/icons/Drawer.info
Fiasco_2.2/icons/Fiasco.dvi.info
Fiasco_2.2/icons/Fiasco.guide.info
Fiasco_2.2/icons/Fiasco.info
Fiasco_2.2/icons/FiascoProject.info
Fiasco_2.2/icons/XPort.info
Fiasco_2.2/icons/XPortData.info
Fiasco_2.2/Install.info
Fiasco_2.2/Install/Deutsch.info
Fiasco_2.2/Install/English.info
Fiasco_2.2/Install/Install
Fiasco_2.2/Libs/MC68020.info
Fiasco_2.2/Libs/MC68020/gtlayout.library
Fiasco_2.2/ReadMe.txt
Fiasco_2.2/ReadMe.txt.info
Fiasco_2.2/RegForm.txt
Fiasco_2.2/RegForm.txt.info
Fiasco_2.2/XPort.info
Fiasco_2.2/XPort/mpearls_III_findpeals.fxp
Fiasco_2.2/XPort/mpearls_III_findpeals.fxp.info
Fiasco_2.2/XPort/RFF.fxp
Fiasco_2.2/XPort/RFF.fxp.info
Fiasco_2.2/XPort/StdTwist.fxp
Fiasco_2.2/XPort/StdTwist.fxp.info
```

Archiv Fiasco_doc_eng.lha:

```
Fiasco_2.2/Documentation/English/Fiasco.dvi
Fiasco_2.2/Documentation/English/Fiasco.dvi.info
Fiasco_2.2/Documentation/English/Fiasco.guide
Fiasco_2.2/Documentation/English/Fiasco.guide.info
Fiasco_2.2/Documentation/English.info
```

Archiv Fiasco_doc_deu.lha:

```
Fiasco_2.2/Documentation/Deutsch/Fiasco.dvi
Fiasco_2.2/Documentation/Deutsch/Fiasco.dvi.info
Fiasco_2.2/Documentation/Deutsch/Fiasco.guide
Fiasco_2.2/Documentation/Deutsch/Fiasco.guide.info
Fiasco_2.2/Documentation/Deutsch.info
```

Anhang B

Fehler-Codes

Dieser Abschnitt enthält alle Fehler-Codes die von Fiasco 2.2 erzeugt werden können. In den meisten Fällen sind die Fehler-Codes für den Benutzer nur über Fiascos ARexx-Port und der Variable FIASCO.LASTERROR verfügbar. In manchen Fällen zeigen auch Fiasco-Fehlerrequester den Fehler-Code an.

Das ARexx-Kommando Fault (siehe Abschnitt 12.6.23) kann zum Umwandeln des Codes in einen lokalisierten Fehler-Text benutzt werden.

1000 - Unbekanntes Kommando oder Funktion: Ein ARexx-Kommando oder eine Formel-Funktion ist Fiasco unbekannt. Kann durch Tippfehler oder eine veraltete Fiasco-Version erzeugt werden.

1001 - Ungültige Argumente: Die Argumente für ein ARexx-Kommando oder eine Formel-Funktion sind nicht gültig.

1002 - Unbekannte Feld-ID: Eine für ein ARexx-Kommando oder eine Formel angegebene Feld-ID existiert nicht in der jeweiligen Datenbank.

1003 - Kein Record: Ein Record ist benötigt und nicht verfügbar.

1004 - Unbekanntes Projekt: Das angegebene Projekt ist zur Zeit nicht von Fiasco geladen.

1005 - Falscher Modus: Ein Kommando benötigt daß Fiasco in einem bestimmten Modus ist, der zur Zeit nicht aktiv ist.

1006 - Falscher Feldtyp: Ein ARexx-Kommando oder eine Formel versuchte auf ein Feld zuzugreifen, daß einen solchen Zugriff aufgrund seines Typs nicht erlaubt.

1007 - Suche fehlgeschlagen: Die Suchfunktion hat keine Übereinstimmung gefunden.

1008 - Projekt bereits aktiv: Das Projekt ist bereits aktiv.

1009 - GUI nicht verschlossen: UnlockGUI wurde bei einer bereits offenen GUI aufgerufen.

1010 - Konnte Feld nicht aktivieren: ActivateField konnte ein Feld nicht aktivieren.

1011 - Nicht Listview: Ein ARexx-Kommando oder eine Formel versuchte ein Feld als Listview-Feld anzusprechen, obwohl es kein Listview-Feld ist.

1012 - Index außer Reichweite: Die Index/Eintrags-Nummer eines Listview-Feldes existiert nicht.

1013 - Unbekannter Index: Index-Datei kann nicht gefunden werden.

1014 - Konnte Index nicht aktivieren: Index-Datei kann nicht aktiviert werden.

1015 - Unbekanntes SearchInfo: Ein ARexx-Kommando versuchte, ein SearchInfo zu benutzen, das nicht existiert.

1016 - ARexx-Server läuft nicht: Der RexxMast-Prozess existiert nicht.

1017 - ARexx-Fehler: Allgemeiner ARexx-Fehler.

1018 - Unbekannter Feldtyp: Ein Feldtyp der nicht existiert wurde angegeben. Kann durch einen Tippfehler oder durch Benutzen einer veralteten Fiasco-Version verursacht werden.

1019 - Feld existiert bereits: CreateField versuchte ein Feld mit einer bereits existierenden ID zu erstellen.

1101 - Keine passende Klammer: Eine geöffnete Klammer wurde nicht geschlossen.

1102 - Operand fehlt: In einer Formel wurde ein Operator benutzt, dem ein Operand fehlt.

1103 - Operand ungültig: Ein Operator kann einen Operand nicht benutzen.

1104 - Syntax-Fehler: Allgemeiner Syntax-Fehler.

1105 - Ausdruck erwartet: Ein Ausdruck wird benötigt.

1106 - If ohne else: Der ?-Operator wurde ohne : angegeben.

1107 - Else ohne If: Ein : wurde ohne ? angegeben.

1200 - Mathematik-Fehler: Allgemeiner Mathematik-Fehler.

1201 - Unterlauf: Eine Zahl ist zu klein geworden.

1202 - Überlauf: Eine Zahl ist zu groß geworden.

1203 - Division durch Null: Es wurde eine Division durch Null versucht.

1204 - Keine gültige Zahl: Eine Zahl ist nicht gültig.

1205 - Nicht vergleichbar: Zahlen können nicht verglichen werden.

1206 - Domain: Der Definitionsbereich einer Funktion wurde nicht eingehalten.

1207 - Außer Reichweite: Eine Zahl ist außerhalb von Grenzen.

1301 - Unbekannter Objekt-Typ: Ein Objekt-Typ für GetAttr oder SetAttr ist unbekannt. Kann durch Tippfehler oder eine veraltete Fiasco-Version verursacht werden.

1302 - Objekt-Name fehlt: Ein Objekt benötigt einen Namen, der aber nicht angegeben wurde.

1303 - Unbekannter Objekt-Name: Der Name für ein Objekt ist nicht bekannt. Kann durch Tippfehler verursacht werden.

1304 - Unbekanntes Attribut: Ein Attribut ist Fiasco nicht bekannt. Kann durch Tippfehler oder eine veraltete Fiasco-Version verursacht werden.

Fiasco kann auch Amiga DOS Fehler-Codes erzeugen. Dies sind die häufigsten:

103 - Speicherplatzmangel: Für eine Operation steht nicht genüg RAM zur Verfügung. Versuchen Sie, etwas RAM freizumachen, oder Ihr System zu erweitern.

115 - Ungültige Zahl: Eine Nummer ist benötigt, aber Fiasco bekam etwas anderes.

116 - Gefordertes Argument fehlt: Fiasco benötigt mehr Argumente als derzeit angegeben sind.

117 - Argument nach Schlüsselwort fehlt: Ein Schlüsselwort wurde ohne Argument angegeben.

118 - Zu viele Argumente: Sie haben mehr Argumente als erlaubt angegeben.

119 - Ungerade Anzahl von Anführungszeichen: Einem öffnenden Anführungszeichen fehlt ein schließendes.

Anhang C

Relations-Checkliste

- Erzeuge Schlüssel-Feld “dort”. Eventuell “eindeutiger Schlüssel” aktivieren.
- Erzeuge eigentliches Feld “dort”. Bei String, Extern oder Datatypes “Max. Zeichen” merken.
- Projekt speichern.
- Erzeuge Schlüssel-Feld “hier”. Muß den selben Typ wie “dort” haben.
- Erzeuge eigentliches Feld “hier”. Muß den selben Typ wie “dort” haben. Bei String, Extern oder Datatypes muß “Max. Zeichen” gleich sein.
- Projekt speichern.
- Öffne Relations-Requester für eigentliches Feld “hier”.
- Schlüssel “hier” auswählen.
- Relations-Datei auswählen
- Schlüssel und eigentliches Feld “dort” auswählen. Wenn das gewünschte Feld nicht angzeigt wird, Typ und bei String, Extern oder Datatypes Max. Zeichen überprüfen.
- Ok anwählen

Anhang D

Implementation der Clipboard-Unterstützung

Die Menüpunkte Record/Ausschneiden, Record/Kopieren und Record/Einfügen benutzen das Clipboard, um die Daten temporär zu speichern. Das Clipboard des Amiga OS soll eine Schnittstelle für verschiedene Programme sein, um bestimmte Daten auszutauschen. Um dies sicherzustellen darf das Clipboard nur IFF-Daten enthalten.

Fiasco benutzt Einheit 0 des Clipboards und speichert seine Daten in IFF-FTXT-Dateien in einem besonderen Format. Jedes Feld hat einen eigenen Chunk. In diesem Chunk ist der Feldinhalt im ASCII-Format abgelegt.

Die Reihenfolge hängt mit der internen Feld-Liste von Fiasco zusammen. Fiasco benutzt auch diese Reihenfolge, um herauszufinden, welche Daten zu welchem Feld gehören.

Mit den meisten anderen Programmen können Sie so strukturierte IFF-FTXT Dateien nicht erzeugen. Das Einfügen in andere Programme ist besser unterstützt. Zum Beispiel fügt das Conclip-Programm die Daten korrekt ein, während MultiView nur den ersten Chunk anzeigt.

Anhang E

Bugs

Falls Sie irgendwelche Bugs oder Fehlfunktionen in Fiasco entdecken, wenden Sie sich bitte an mich. Bitte beschreiben Sie den Fehler detailliert, d.h. geben Sie an, wie sich der Fehler ausdrückt, und geben Sie eventuell im Hintergrund laufende Programme und die Computer-Konfiguration an. Diese können Sie gut mit dem Programm ShowConfig im Tools-Verzeichnis auflisten.

Diese Fehler sind z.Zt. bekannt:

- Komischer Bug in der Druckfunktion: Auf manchen System druckt Fiasco nur den ersten Record aus, wenn Position 0;0 von einem Feld belegt wird. Wenn Sie das Feld eine Position nach links verschieben, wird das Drucken normal funktionieren.
- Unter Kick 37.x flackert der Fensterrahmen des List-Fensters in höchst merkwürdiger weise, wenn Menüoperationen ausgeführt wurden, und das Fenster aktiv war.
- ARexx scheint Probleme mit Dateinamen zu haben, die Leerzeichen enthalten. Der Name wird dann immer nur bis zum Leerzeichen interpretiert.
- Scheint manchmal (nicht immer!) etwas Speicher nicht freizugeben.
- Erzeugt mit asl.library 40.6 und Kickstart 40.70 MungWall-Hits, nachdem ein Dateirequester geschlossen wurde. Ich denke, daß dies ein Bug in ASL oder Intuition ist, aber nicht in Fiasco.

Anhang F

Was es noch zu tun gibt

Natürlich ist an Fiasco noch lange nicht alles perfekt. Hier ist eine Liste aller Dinge, die ich noch irgendwann ändern oder hinzufügen will. Wenn Sie auch Ideen haben, wie man Fiasco noch verbessern können, schreiben sie mir!

- "Packen" von Projekten, indem nach unbenutzten Feldern gesucht wird, und benutzte auf die benutzte Länge verkleinert werden.
- Überprüfen, ob ein ähnlicher Record bereits vorhanden ist (automatisch)
- Bessere Unterstützung des Masken-Modus von ARexx
- "Nur einmal eingeben" Feld Attribut
- Option, ein Fiasco-Projekt nur in einer Datei zu speichern (wie es unter Fiasco 1.x war)
- Option, Datenbanken mit Passwörten zu schützen. Vielleicht mit mehreren "User-Levels", also Schreib- oder Schreib- und Lesezugriff.
- Datatypes-Felder, die ihren Inhalt nach eine kurzen Zeitspanne laden, in der der Benutzer nichts getan hat.
- "Cache" für Datatypes-Feld-Daten.
- Fiasco sollte die Nummer des aktiven Records und die Anzahl aller Records (wie im Service-Fenster) in der Titelzeile anzeigen.
- Es sollte eine Kontrolle geben (z.B. von ARexx), ob Felder aktiv sind, oder nicht.

Weiterhin plane ich, Fiasco auf pOS, dem Betriebssystem von proDAD, umzusetzen.

Anhang G

Credits

Ich möchte mich bei diesen Leuten bedanken,
die mir irgendwie beim Erstellen von Fiasco geholfen haben:

Reinhard Katzmann	Betatesten
Giuseppe Sacco	Betatesten
Ulrich Scholz	Betatesten
Carsten Klein	Betatesten
Curtis Stanton	Revidieren der englischen Anleitung
Dirk Hartstein	Betatesten
Gregor B. Rosenauer	Betatesten
Lutz Kalkof	Betatesten und Werbung ;-)
Giuseppe Chillemi	Betatesten
Claudio Mazzuco	Italienischer Catalog
Thomas Schwarz	Betatesten
Michael A. Krehan	Betatesten
Martin Sahlén	Schwedischer Catalog
Per Torp	Dänischer Catalog
Ralf Terber	Betatesten
Javier Romero	Spanischer Catalog
Olaf Barthel	gtlayout.library
Mark Thomas	textfield.gadget
Ralph Babel	WBPath
Jay Miner	Erfinden des Amiga
Commodore	Produzieren des Amiga
ESCOM	Naja, Amiga Developer CD?
VisCORP	Gewollt haben, den Amiga zu kaufen
Gateway 2000	Mal sehen ...
proDAD	pOS

Und natürlich

auch Dank an alle registrierten Benutzer!

Index