

Fachhochschule Köln
Abteilung Gummersbach
Fachbereich Informatik
Studiengang Wirtschaftsinformatik

Diplomarbeit
**Integration einer relationalen Datenbank
mit Lotus Notes
über einen konfigurierbaren
Datenpumpprozess**

Eingereicht von
Achim Stein
Matrikelnummer 102 27 158

Prüfer:
Prof. Dr. Heide Faeskorn – Woyke
Hr. Oliver Trabert

Schwellenbach, den 26. August 1999

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS	2
KAPITEL 1 VORWORT	6
KAPITEL 2 WARUM ENTWICKELT CONET EIN EIGENES WERKZEUG?	7
KAPITEL 3 MITBEWERBERPRODUKTE	9
3.1 NOTESPUMP	10
3.2 DECS	11
3.2.1 DER NAVIGATOR DES DECS-ADMINISTRATOR	11
3.3 PUMPGATE 2	12
3.3.1 SO ENTSTAND PUMPGATE	14
3.4 PERCUSSION NOTRIX	14
KAPITEL 4 LOTUS NOTES	16
KAPITEL 5 DIE GESTALTUNGSELEMENTE VON LOTUS NOTES	19
5.1 PRIMÄRE GESTALTUNGSELEMENTE	19
5.1.1 MASKEN	19
5.1.2 FELDER	20
5.1.3 TEILMASKEN	21
5.1.4 GEMEINSAME FELDER	22
5.1.5 ANSICHTEN	23
5.1.6 NAVIGATOREN	24
5.1.7 AKTIONEN	26

KAPITEL 6 MÖGLICHKEITEN DER PROGRAMMIERUNG IN LOTUS NOTES **27**

6.1	LOTUS SCRIPT	27
6.2	DIE MAKROSPRACHE	28
6.3	EIN VERGLEICH VON MAKROSPRACHE UND LOTUS SCRIPT	29
6.4	DIE C-API	33
6.5	WAS UNTERSCHIEDET EINE LOTUS-NOTES- DATENBANK VON EINER RELATIONALEN DATENBANK ?	35

KAPITEL 7 ODBC **37**

7.1	WAS IST ODBC?	37
7.2	WIESO WURDE ODBC NÖTIG?	38
7.3	EXKURS: ARCHITEKTUREN DER DATENHALTUNG	39
7.3.1	HERKÖMMLICHE RELATIONALE DBMS	39
7.3.2	DATEI-MANAGEMENT-SYSTEME UND ISAMS	40
7.3.3	DESKTOP-DATENBANKEN	42
7.4	DIE ODBC-ARCHITEKTUR	42
7.5	HISTORISCHES ÜBER ODBC	43
7.6	ADAPTIVE PROGRAMMIERUNG UND KONFORMITÄTSSTUFEN	45
7.6.1	DIE API-KONFORMITÄT	46
7.6.2	DIE SQL-KONFORMITÄT	46
7.7	SCHLUBBEMERKUNG ZUM ODBC-KAPITEL	48

KAPITEL 8 DIE PROBLEMSTELLUNG **50**

8.1	DIE PROBLEMSTELLUNG BEIM TEXTIMPORT	50
	DER EXCEL-TEXTIMPORT ALS BEISPIEL FÜR EINE GELUNGENE BENUTZERSCHNITTSTELLE	52
8.2	DIE PROBLEMSTELLUNG BEI ODBC	53
8.3	DIE PROBLEMSTELLUNG BEI LOTUS NOTES	54

KAPITEL 9 VORBEREITENDE MAßNAHMEN **55**

9.1	ODBC - MAßNAHMEN IM VORFELD	55
9.2	DAS EINBINDEN DER BENÖTIGTEN SCRIPT-ERWEITERUNGEN	57
9.2.1	LSX UND DLL	57
9.3	KURZVORSTELLUNG LS:DO	60
9.3.2	NACHWORT ZUM THEMA LS:DO	63

KAPITEL 10 HILFSTOOLS **65**

10.1	MSQUERY	66
10.2	ORACLES 32-BIT ODBC TEST	67

KAPITEL 11 DER CONET-STYLEGUIDE **68**

11.1	KOMMENTARE IM QUELLTEXT	68
11.2	NAMENSKONVENTIONEN IN SCRIPT	69
11.3	DIE HAUPTGESTALTUNGSELEMENTE	71
11.3.1	ANSICHTEN	71
11.3.2	MASKEN	71
11.3.3	FELDNAMEN	72
11.3.4	AGENTEN	72
11.4	SCHRIFTARTEN	72
11.5	ACTIONBUTTONS	73
11.5.1	IN DER KOPFZEILE DER MASKEN	73
11.5.2	IN DER KOPFZEILE DER ANSICHTEN	73

KAPITEL 12 DAS PROGRAMM **74**

12.1	ALLGEMEINE SPEZIFIKATIONEN DES PROGRAMMES	74
12.2	DIE DATENBANKDOKUMENTE	75
12.2.1	DAS DOKUMENT „ÜBER DATENBANK“	75
12.2.2	DAS DOKUMENT „ARBEITEN MIT DER DATENBANK“	76
12.3	DIE SCRIPT-BIBLIOTHEK	77
12.4	DATENBANK-SCRIPT	82
12.5	DIE AGENTEN	82
12.6	DIE MASKE „VERBINDUNG ZU ASCII“	84
12.7	DIE MASKE „VERBINDUNG ZU NOTES“	86
12.8	DIE MASKE „VERBINDUNG ZU ODBC“	88
12.9	DIE VERSTECKTEN MASKEN	92
12.9.1	DIE MASKE “(FRMSEPARATOR)“	93
12.10	DIE TEILMASKE „HISTORIE“	93
12.11	DIE ANSICHT „VERBINDUNGEN“	94
12.12	DER NAVIGATOR	95

KAPITEL 13 EIN PROBELAUF DER DATENPUMPE **97**

KAPITEL 14 TROUBLESHOOTING UND ERKANNTES FUBANGELN **100**

14.1	ODBC: PROBLEME BEIM AUSFÜHREN EINER ABFRAGE VERURSACHT DURCH SONDERZEICHEN	100
14.2	ODBC: BENUTZERRECHTE AM BEISPIEL ACCESS	101
14.3	LIMITS VON LOTUS NOTES	101

KAPITEL 15 AUSBLICK AUF ZUKÜNFTIGE LEISTUNGSERWEITERUNGEN **102**

15.1	ODBC-ABFRAGE ÜBER MEHRERE TABELLEN	102
-------------	---	------------

15.2	ODBC-ABFRAGE MITTELS EINER VOM BENUTZER DEFINIERTEN SELECT-ANWEISUNG	102
15.3	DATENTYPEN ÜBERNEHMEN	103
15.4	UPDATE-FUNKTIONALITÄT	103
15.5	ZEITSTEUERUNG	103
 <u>ANHANG A: LITERATURLISTE</u>		 105
 <u>ANHANG B: ABKÜRZUNGSVERZEICHNIS</u>		 106
 <u>ANHANG C: GLOSSAR</u>		 107
 <u>ANHANG D: ABBILDUNGSVERZEICHNIS</u>		 109
 <u>ANHANG E: TABELLENVERZEICHNIS</u>		 110

Kapitel 1

VORWORT

Am Beginn meiner Diplomarbeit standen Gespräche mit Thomas Radigewski und Oliver Trabert von der CONET CONSULTING AG in Hennef. In der ersten Definition sollte ein Programm entwickelt werden, das auf grafischer Steuerung basierend ASCII-Dateien, später dann auch Inhalte aus anderen Datenbanksystemen übernehmen sollte.

In kürzester Zeit wuchsen dann die Anforderungen, da viele Mitarbeiter ihre Ideen und Anregungen einbrachten.

Das Ergebnis ist nun ein „Stück Software“ das aus insgesamt drei Teilen besteht.

Es wurde ein Import von ASCII-Dateien, sowie aus ODBC-Quellen und aus Lotus Notes selbst realisiert. Der am weitesten entwickelte Teil meiner Arbeit ist die Maske „Verbindung zu ODBC“. Im Laufe der Zeit werden sich die anderen Masken so weit wie möglich diesem Stil des Benutzerinterfaces nähern.

Den CONET-Style-Guide für die Notes-Entwicklung habe ich bei der Programmierung als bindend betrachtet. Meiner Meinung nach ist es wichtig für die Programmierer eines Unternehmens, daß sie die gleiche Sprache sprechen und auch untereinander Code-Teile tauschen können. Auszüge des Styleguides sind im Kapitel „Der CONET-Styleguide“ aufbereitet.

Abschließen möchte ich das Vorwort mit einem Dank an das CONET-Team Industrieprojekte. Sie waren immer da, wenn ich Fragen hatte und haben mich mit Rückfragen und Anregungen stets neu motiviert.

Kapitel 2

WARUM ENTWICKELT CONET EIN EIGENES WERKZEUG?

Das Problem der „Datenintegration und Datenextraktion“ wird es immer geben, zur Zeit sucht CONET erfahrene Mitarbeiter auf diesem Gebiet.

Als Unternehmensberatung oder wie man zu neudeutsch sagt: „Consultant“, muß man schnell und kompetent auf die Fragen der Kunden reagieren.

Nimmt man als Beispiel die Maschinenfabrik Kampf in Wiehl. Dort gab es zu meiner Zeit das Problem der Integration vorhandener Datenbestände aus einer Datenbank unter VMS in Lotus Notes. Wenn man von einem proprietären System wie diesem, Daten zu einem neuen System transferieren möchte, merkt man oft, daß Schnittstellen selbst programmiert werden müssen.

Für diese Aufgabe werden Fachkräfte gebraucht, die so etwas schon kennen und die Daten ohne ungewollte Veränderungen bereitstellen können.

Die Folgenden Gründe haben CONET CONSULTING AG zu der Eigenentwicklung einer „Datenpumpe“ bewogen.

- Mitbewerberprodukte sind teuer
(Percussion Notrix kostet z.B. 7000 £).
- Das Know-How der Mitarbeiter wächst.
Berater die sich nicht ständig weiterbilden, verlieren die Fähigkeit flexibel auf neue Probleme reagieren zu können. Die Lotus-Script-Erweiterungen, die für diese

Diplomarbeit benutzt werden, sind z.B. nicht das „tägliche Brot“ in der Notes-Programmierung. CONET versucht durch ständige Fort- und Weiterbildungsmaßnahmen einen hohen Wissensstand bei allen Mitarbeitern zu halten.

- Importscrippts sollen nicht immer wieder neu geschrieben werden.
Die bisherige Vorgehensweise war, daß sich die Programmierer jedesmal ihre persönliche Quelltextsammlung ansahen und den Import neu programmierten. Das bedeutete jedesmal, daß man eine gewisse Zeit verlor, da der Programmcode meist umgeschrieben werden muß.

Eine geeignete Codezusammenfassung wie meine „Datenpumpe“ kann zu den gewünschten Ergebnissen führen, oder zumindest eine reife Quelltextsammlung darstellen.

- Unser eigenes Tool darf auch der Kunde bekommen (lizenzrechtlich gesehen).
Denkbar ist, daß die CONET CONSULTING AG nicht nur die Datenübernahme als Projekt verkauft, sondern dem Kunden auch die Möglichkeit gibt, in eigener Regie zu handeln.

Kapitel 3

MITBEWERBERPRODUKTE

Da sich das Design von Lotus Notes sehr von den traditionellen Datenbanksystemen unterscheidet, fokussieren sich einige Anstrengungen der Notes-Entwicklergemeinschaft auf die sogenannte Datenbank-Integration.

In dieser Sparte können verschiedene Entwicklungen betrachtet werden:

- Der Zugriff auf RDBMS über die Notes-eigenen Makrofunktionen @DBCColumn, @DBLookup und @DBCommand.
- Der Zugriff auf RDBMS über die Script-Klassen LS:DO auf die ich später eingehen möchte.
- Der umgekehrte Weg - Zugriff auf Lotus Notes über den Treiber NotesSQL.
- Der Server-to-Server Datentransfer.

Letztendlich sind aber nur die Produkte der letzten Kategorie so komplex in ihrer Thematik, daß sie hier mit den nun folgenden Produktvorstellungen näher beschrieben werden.

3.1 NotesPump

NotesPump wurde von der Lotus Development Corp. entwickelt und gehört zur Klasse der serverbasierten Werkzeuge. Es kann sowohl kommandogesteuert als auch zeitgesteuert Daten aus den angegebenen Quellen „pumpen“.

Die Steuerungslogik wurde als Notes-Anwendung implementiert. In dieser Anwendung können Dokumente für „Konfiguration“, „Verbindungen“ und „Aktivitäten“ erstellt werden.

- Das Konfigurationsdokument definiert die Server-Umgebung.
- Die Verbindungsdokumente spezifizieren die genaue Lage von Quelle und Ziel.
- Die Aktivitäten-Dokumente sind die Arbeitspferde. In Ihnen wird ein weites Feld von Zeitsteuerungs- und Datentransferoptionen verwaltet.

Der serverbasierte Teil der Anwendung ist eine 32-bit Applikation. Sie ist multitaskingfähig und erlaubt mehrfachen Datentransfer auszuführen. Über den üblichen Transfer nach Notes hinaus gibt es auch den Transfer Oracle/Sybase, Sybase/DB2 usw.

Man sieht an dieser Stelle, daß es kein auf Lotus Notes spezialisiertes Werkzeug ist.

Programmiertechnisch gesehen ist NotesPump sehr komplex. Es patcht den Notes-Server und für den Notes-Client bringt NotesPump seine eigenen Klassen mit, um die Entwicklungsumgebung zu erweitern.

Preis: Eine NotesPump V2.5 Einzellizenz kostet zwischen 5000 und 6000 US-Dollar.

Softwarevoraussetzungen für den NotesPump-Server: OS/2 Warp 3.0 / 4.0, oder Windows NT 3.51 / 4.0, HP-UX 10.01, oder Sun Solaris 2.5.1

CPU-Leistung: Für OS/2 Warp oder Windows NT Intel-based IBM-compatible; 486 oder besser, ein Pentium wird empfohlen.

Ram-Speicher: Als Minimum 64 MB, jedoch werden über 64 MB empfohlen.

Lotus Notes Version: Lotus Notes Release 4.11 oder höher (Client und Server)

Festplattenplatz: NotesPump Server: 20MB; NotesPump Administrator Datenbank: 10MB

3.2 DECS

Der Domino Enterprise Connection Server ist dem schon beschriebenen NotesPump sehr ähnlich, so daß ich mit der Beschreibung der Benutzerschnittstelle fortfahre.

Der DECS Administrator (decsadm.nsf) ist das Kontrollwerkzeug welches der Benutzer sieht. Beim Öffnen der Datenbank erscheint auf der linken Seite ein Navigator. Mit diesem Navigator kann man zwischen der Ansicht „Aktivitäten“ und „Verbindungen“ wechseln und einige Aktionsknöpfe drücken die ich Ihnen gleich vorstellen werde.

3.2.1 Der Navigator des DECS-Administrator



Hier wählt man zwischen den Ansichten.

Auf Knopfdruck wird ein neues Verbindungsdokument erstellt.

Auf Knopfdruck wird ein neues Aktivitätendokument erstellt.

„Start“ startet eine Aktivität und „Stop“ stoppt sie. „Log“ listet die Startzeit und andere Statusmeldungen. Auf Knopfdruck wird der Benutzerassistent zugeschaltet.

Bei Knopfdruck „Intro“ wird das Dokument „Benutzen dieser Datenbank“ angezeigt, „Doc“ zeigt eine externe Dokumentation und „Exit“ schließt die Datenbank.

Abbildung 1: Der Navigator des DECS-Administrator

3.3 Pumpgate 2

Entwickelt wurde Pumpgate 2 von der vierköpfigen, deutschen Unternehmensberatung Innovation Gate aus Ratingen. Innovation Gate ist ein Softwarehaus und Beratungsunternehmen. Der Schwerpunkt ihrer Arbeit liegt in der Internet-Integration in bestehende DV-Landschaften auf Basis von Lotus Domino.

Auf Ihrer Webseite (www.innovationgate.de) werben Sie für Ihre vier Hauptprodukte Pumpgate 2, File Machine (Verzeichnis-,Datenabgleich über FTP), Contactgate (Kontaktverwaltung) und Webgate (Webautorensystem).

PumpGate 2 ist eine komplett in Lotus Script implementierte Pump-Engine für Lotus Notes, die die Möglichkeit eines zeitgesteuerten Imports bzw. Updates bietet. Möglich ist die Replikation von Notes-Datenbanken mit allen ODBC-fähigen relationalen Datenbanken, sowie der Direktzugriff auf Oracle, DB2, Informix, Sybase und Microsoft SQL Server.

Als Datenquelle und -Ziel können in beliebigen Kombinationen Notes-Datenbanken, relationale Datenbanken, Textdateien und Dateiverzeichnisse verwendet werden.

Bei den Textdatenquellen läßt sich jedoch nicht die Option „Feste Feldlängen“ bestimmen.

Ebenfalls ist der Begriff Datenquelle ein wenig irreführend, da das Programm die sog. Datenquellen auch als Ziel benutzt.

Laut Innovation Gate erfolgt der Zugriff auf relationale Datenbanken "native" mit Hilfe spezieller "Access Libraries", die für ORACLE, DB2, Informix, Sybase und Microsoft SQL Server verfügbar sind. Auf alle anderen Datenbanken kann über ODBC zugegriffen werden.

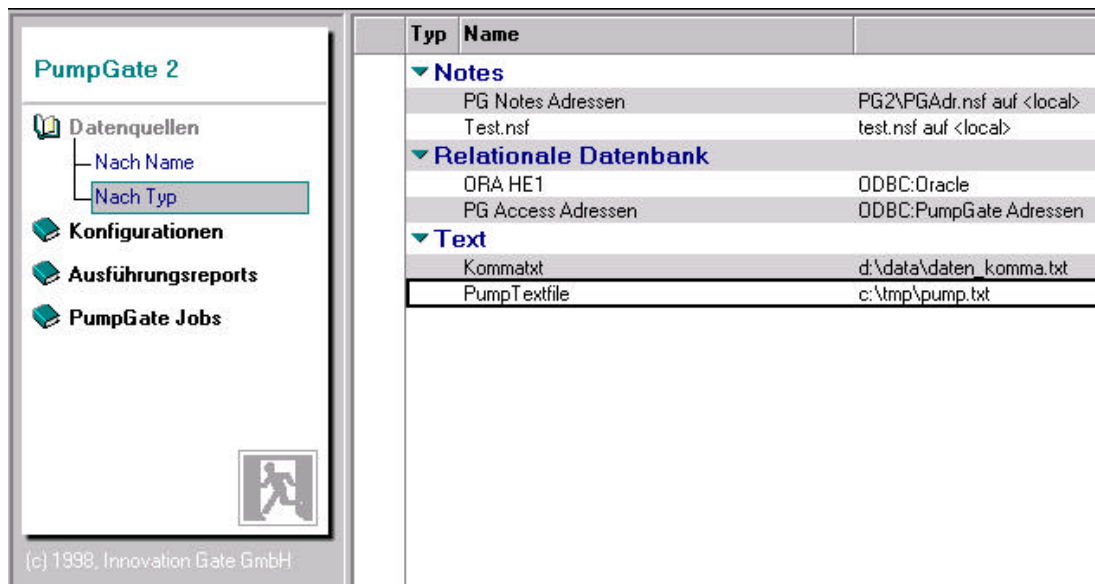


Abbildung 2: Pumpgate - Ansicht der Datenquellen

PumpGate besteht aus einer Notes-Datenbank, die sowohl alle Konfigurationsdokumente als auch die eigentliche Pump-Engine enthält. Es verwendet benutzerspezifische Ausführungsreports zum Steuern der individuellen Datenübertragung. Dadurch ist es möglich, mit einer zentral verwalteten Replik von PumpGate, einer großen Anzahl von Benutzern die Möglichkeit zur Verfügung zu stellen, Daten in ihre lokale, relationale Datenbank zu übertragen, ohne daß der Benutzer selbst die Datenpumpe konfigurieren muß.

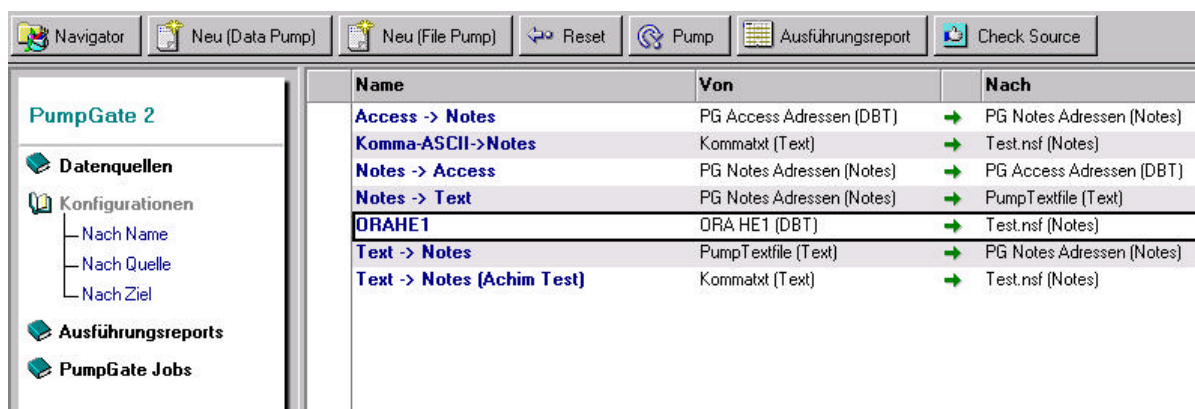


Abbildung 3: Pumpgate - Ansicht der Datenquellen II

3.3.1 So entstand PumpGate

Auszug aus der Webseite <http://www.innovationgate.de>:

Im Rahmen eines Projektes, daß Innovation Gate für einen großen Automobilimporteure durchführte, war die folgende Aufgabe zu lösen:

Eine auf Interbase basierte multimediale Anwendung sollte regional verteilt bei etwa 1000 Autohändlern mit technischen Daten von Fahrzeugmodellen aus einer Notes-Datenbank "versorgt" werden.

3.3.1.1 Die Problematik

Während die Daten per Notes-Replikation zu den Händlern gelangen könnten, müßte dort (in 1000 Lokationen) eine Datenpumpe installiert, konfiguriert und administriert werden, die die Daten von der lokalen Notes-Replik in die Interbasedatenbank überträgt.

Die auf dem Markt vorhandenen Datenpumpen für Lotus Notes hatten in diesem Zusammenhang den Nachteil, daß sie entweder einen separaten Rechner oder die Installation eines Notes-Servers benötigten und damit das Projektbudget schnell gesprengt hätten. Darüber hinaus gab es bei den vorhandenen Produkten keine Möglichkeit der "zentralen" Konfiguration und Administration.

3.4 Percussion Notrix

Das amerikanische Unternehmen Percussion Software wurde 1994 gegründet und hat neben seiner Hauptstelle im Stoneham/Massachusetts noch ein Büro in London. Die Stärke des Unternehmens ist es, dem Notes-Benutzer innovative Werkzeuge an die Hand zu geben, um insbesondere die Notes/Domino Plattform durch e-business-Lösungen zu erweitern.

Zur Zeit wirbt das Unternehmen mit folgender Software:

- Percussion Notrix als Daten-Integrations-Werkzeug.
- Percussion PowerFlow ist ein Workflow-Designer für Lotus Notes mit einem grafischen Benutzerinterface.
- Percussion ServerAdmin Plus ist ein Notes/Domino-Administrations-Werkzeug mit dem der Administrators die Möglichkeit hat, für eine konsistente „Sicherheitspolitik“ über alle seine Notes/Domino-Plattformen zu sorgen.

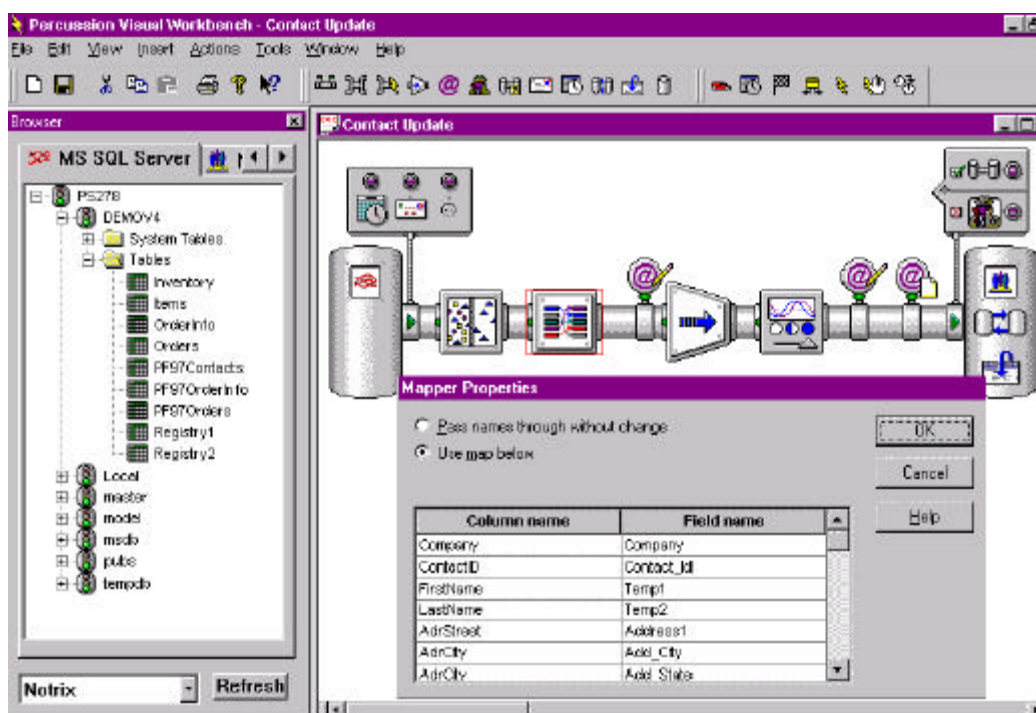


Abbildung 4: PercussionMapper

Das Produkt Percussion Notrix zeichnet sich durch seine innovative, grafische Oberfläche aus. Der Navigator zeigt die grafische Darstellung einer Pumpe. Klickt man in den linken Bereich, kann man die Datenquelle angeben. Der rechte Bereich definiert das Notes-Ziel. In der Mitte sind auf dem Weg von der Quelle zum Ziel verschiedene Transformationen möglich. An dieser Stelle muß auch die Feld-/Spaltenzuordnung (engl. Mapping) vorgenommen werden.

Kapitel 4

LOTUS NOTES

Lotus Notes ist ein verteiltes, auf Dokumenten basierendes Anwendungssystem für Datenbanken. Es handelt sich um ein Produkt zur Kommunikation und Zusammenarbeit in der Gruppe in dem Benutzer auf Informationen zugreifen, sie bearbeiten, organisieren und publizieren können.

Lotus-Notes-Anwendungen bieten Benutzergruppen die Möglichkeit asynchron miteinander zu arbeiten. Der Datenabgleich kann über Internet oder über die gegebene Infrastruktur wie Telefon/ISDN oder WAN/LAN realisiert werden.

Hervorragend ist auch die Möglichkeit, das einzelne Benutzer über eine Insellösung Informationen erstellen, bearbeiten und organisieren können und nur gelegentlich an ein Netzwerk angeschlossen sein müssen um die Datenbestände bidirektional abzugleichen.

In diesen beiden Fällen ist der Einsatz von Lotus Notes klug, das mit seinem Repliziermechanismus in der Lage ist, die Daten konsistent zu halten und zu aktualisieren.

Im Büro von heute finden sich oft ineffiziente Arbeitsabläufe und Doppelarbeit. Zeitaufwendige Suche nach Dokumenten, lange Durchlaufzeiten, Mehrfachablage, mangelnde Transparenz der Systeme und Abläufe sowie Medienbrüche und Inkompatibilitäten der Teilsysteme erschweren ein integriertes, kundenorientiertes und reibungsloses Arbeiten erheblich.

Hinzu kommt, daß das Dokumentenaufkommen in den letzten Jahren drastisch gewachsen ist und noch weiter anwachsen wird. Auch die Verwaltung neuer, zusätzlicher Dokumententypen, die in der nahen Zukunft eine immer größere Rolle spielen werden, wie

beispielsweise Animations- oder Videosequenzen, stellt die Dokumentenverwaltung vor neue Herausforderungen.

Durch den Einsatz eines computergestützten Dokumenten- und Workflow-Management-Systems können erhebliche Optimierungspotentiale ausgeschöpft werden.

Groupware-Systeme unterstützen die schnelle Abstimmung zwischen verschiedenen Personen an räumlich getrennten Orten und die zeitnahe Information der Mitarbeiter. Sie bieten eine Kommunikations-/Kooperationsinfrastruktur für die Bewältigung dieser Aufgaben. Joint Planning, Joint Editing, Bulletin-Boards, E-Mail, Application Sharing und Videoconferencing stellen als Teilfunktionalitäten den Weg in die Zukunft dar.

Dokumenten-Management-Systeme unterstützen im technischen Büro sowohl die Ablage von Konstruktionszeichnungen als auch die Verwaltung von Handbüchern in Form elektronischer Bücher, bei Banken die Speicherung von Belegen, bei Zeitschriftenverlagen und Presseagenturen die Archivierung und Wiederauffindung von vielfältigem Text- und Bildmaterial, sowie bei Versicherungen und in öffentlichen Verwaltungen den Fluß von Akten.

Bei typischen Notes-Anwendungen handelt es sich in der Regel um eine, oder um eine Kombination der folgenden Anwendungstypen:

- Workflow-Management-Systeme

unterstützen den organisatorischen Ablauf von Verwaltungsprozessen.

Dadurch ist es möglich, den Sachbearbeiter bei aufwendiger Routinearbeit zu entlasten und den Ablauf solcher Prozesse effizienter zu gestalten. Hierzu ist es erforderlich, die bestehenden Geschäftsprozesse mit entsprechenden Methoden zu analysieren, geeignete Prozesse auszuwählen und zu modellieren.

Notes kann in diesem Umfeld Systemunterstützung für Gruppenentscheidungen und Besprechungen bieten, sowie die zahlreichen workflowgeeigneten Anwendungen abbilden.

Workflow-Anwendungen bilden einen wichtigen Bestandteil zahlreicher Notes-Anwendungen. Man kann sie allen Hauptanwendungstypen von Notes hinzufügen um ständig wiederkehrende Prozesse zu automatisieren. Es lassen sich besonders gut

Routineaufgaben abbilden, wie das Senden von Mahnungen oder die Anforderung von Genehmigungen.

- Broadcast

Verwaltung von Gruppendokumenten, damit alle Mitarbeiter eines Unternehmens die neuesten Informationen teilen.

- Bibliothek

Dienstprogramme für Arbeitsgruppen und Entwicklungstools, dynamische mit den aktuellen Problemen des Unternehmens wachsende Hilfsdatenbanken, die bei Bedarf aktualisiert werden.

- Protokoll

Controlling-Tools die Projekte, Prozesse und Aufgaben überwachen und über den Stand dieser Tätigkeiten berichten, an der normalerweise mehrere Benutzer beteiligt sind.

- Diskussion

Weg zu einer gemeinsamen Entscheidungsfindung. Ansporn zum Suchen eines Konsens und anschließend gemeinsame Nutzung der Konferenzergebnisse.

Kapitel 5

DIE GESTALTUNGSELEMENTE VON LOTUS NOTES

5.1 Primäre Gestaltungselemente

5.1.1 Masken

Die Maske ist ein wesentliches Gestaltungselement in einer Notes-Datenbank. Bei einer Maske handelt es sich um eine Dokumentschablone, die eine Struktur für die Dateneingabe vorgibt, sowie die Verarbeitung der Daten steuert. Nach der Eingabe der Daten in die Maske werden die Informationen als Dokument in der Datenbank gespeichert. Masken bestimmen das Format und das Layout der Anzeige von Dokumenten.

Jede Maske kann Felder, statischen Text, Hotspots, Aktionsleisten, Teilmasken, Layoutbereiche, Abschnitte, Grafiken, Tabellen und Schaltflächen enthalten. Diese Elemente bestimmen das Erscheinungsbild der Maske.

In Notes gibt es drei Arten von Masken. In der obersten Hierarchiestufe steht die Maske vom Typ „Dokument“. Eine Maske dieses Typs kann ganz unabhängig von anderen Masken eingesetzt werden. Anders verhält es sich bei den Maskentypen „Antwort“, oder „Antwort auf Antwort“. Sie kann man als Hierarchiestufe zwei und drei sehen. Beide

Maskentypen brauchen Bezugsdokumente, deren eindeutige Kennung in einem Referenzfeld gespeichert werden kann.

Der Maskentyp „Antwort“ bezieht sich dabei immer auf ein Dokument.

Der Maskentyp „Antwort aus Antwort“ bezieht sich auf ein Dokument oder auf ein Antwortdokument.

Beide letztgenannten Typen können Werte aus dem Bezugsdokument erben.

Ein wichtiges Designmittel ist die „Verbergen-Wenn-Option“ (engl.: Hide-When). Sie ist Teil eines jeden Maskenelementes und man kann damit die Anzeige dieser Elemente steuern. An dieser Stelle sind der Phantasie des Programmierers keine Grenzen gesetzt. Er kann frei entscheiden, ob er das Element nur für eine Benutzergruppe (z.B.: Administratoren) anzeigt, oder bei Erfüllung einer Bedingung (z.B.: Wenn Feldwert > 10.000). Aus meiner Erfahrung sprechend empfehle ich: „Sorgsam die Bedingung überdenken; und wo Klammern gesetzt werden können, diese auch setzen.“ Es können sonst seltsame Effekte entstehen!

Da die Auswertung jeder Formel eine gewisse Rechenzeit fordert, sollte mit der „Verbergen-Wenn-Option“ sparsam umgegangen werden

Der darstellbare Fenstertitel ist eine Eigenschaft der Maske. Voreingestellt ist der Maskenname, aber man kann verschiedene Zustände des aktuellen Dokuments abfragen und den Fenstertitel nach eigenem Ermessen ändern.

5.1.2 Felder

Ein Feld ist ein Bereich in einer Maske, der eine einzelne Informationseinheit enthält. Jedes Maskenfeld bestimmt den Typ der Daten, die nach dem Speichern als Notes-Dokument zusammengefaßt werden.

Jedes Feld hat die Attribute „Name“, „Datentyp“, „Berechnet“ oder „Bearbeitbar“ sowie einige Anzeigooptionen. Berechnete Felder haben darüber hinaus Formeln oder Script hinterlegt, die ihren Wert bestimmen.

Beim Erstellen eines Feldes muß man nach folgendem Schema verfahren:

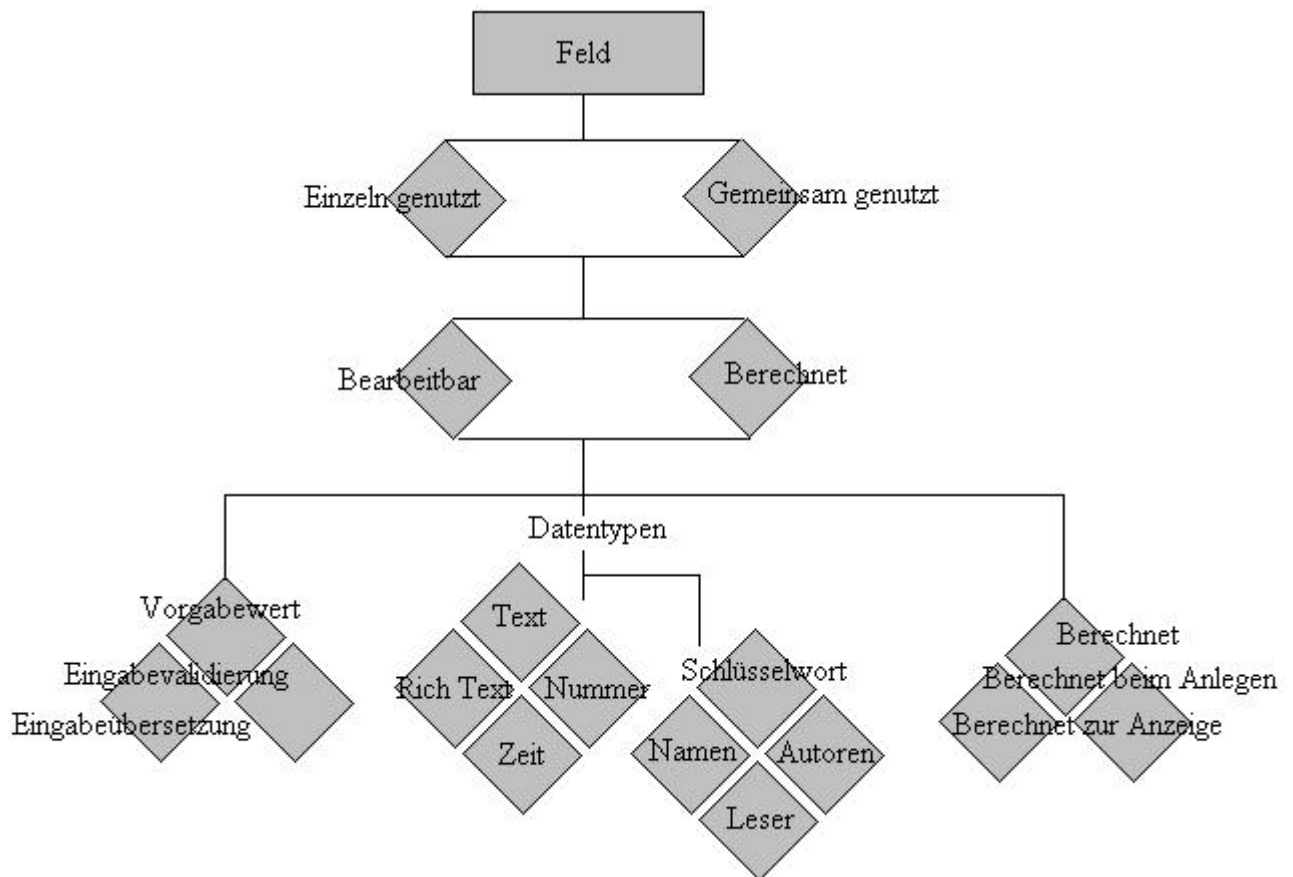


Abbildung 5: Felder erstellen

5.1.3 Teilmasken

Wenn mehrere Masken dieselben Elemente enthalten, sollte man geschickterweise schon in der Planung der Anwendung daran denken, diese Elemente auszukoppeln um sie in einer Teilmaske zu vereinen.

Sollen später Änderungen an Layout oder an den Feldattributen gemacht werden, erspart man sich viel Arbeit. Eine Einschränkung muß man erwähnen:

„Teilmasken können nur von Masken der gleichen Datenbank gemeinsam benutzt werden“.

In meiner Arbeit habe die Maske „Historie“ als Teilmaske definiert, da ich ihre Elemente in drei weiteren Masken benutze.



Abbildung 6: Gestaltung der Teilmaske Historie

5.1.4 Gemeinsame Felder

Ganz ähnlich wie bei den Teilmasken verhält es sich mit den gemeinsamen Feldern. Soll ein gemeinsamer Informationspool für verschiedene Masken einer Notes-Datenbank gelten, kann man ein gemeinsames Feld verwenden.

Ein sinnvolles Beispiel wäre ein Auswahlfeld, in dem die Programmierung die Auswahlmöglichkeiten dynamisch generiert:

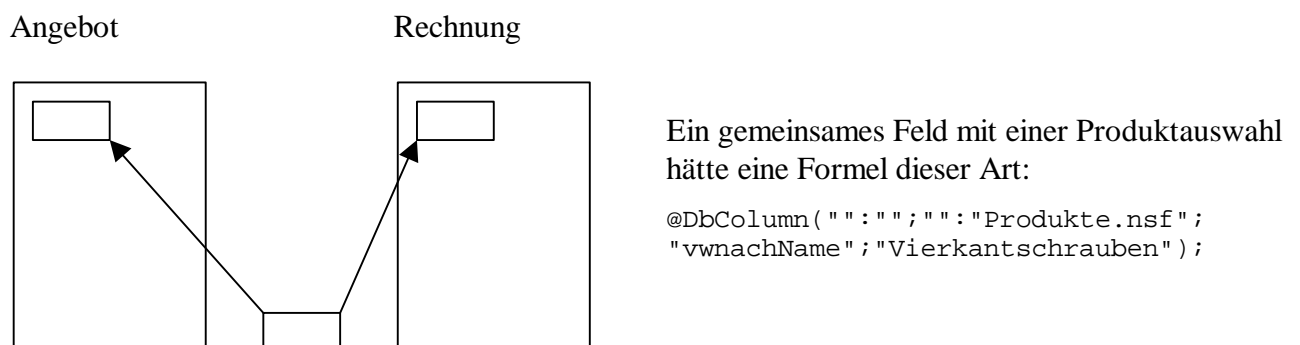


Abbildung 7: Gemeinsame Felder

Das Unternehmen stellt verschiedene Produkte her, die an zentraler Stelle gepflegt werden. Bei Fehleingabe des Benutzers geht die Feldlogik des gemeinsamen Feldes

einher und fragt die zentrale Datenquelle „Produkte.nsf“ ab und generiert eine Produktauswahl für den Benutzer.

5.1.5 Ansichten

Ansichten helfen dem Benutzer, die gewünschten Dokumente problemlos zu finden. Darüber hinaus liefern sie eine Zusammenfassung über den Inhalt einer Datenbank und bieten die Möglichkeit darauf zuzugreifen. Ansichten bestehen aus einer oder mehreren Spalten, die jeweils den Inhalt eines Feldes oder die Ergebnisse einer Formel anzeigen.

In meinen Arbeiten habe ich hauptsächlich die Sortierung folgender Typen benutzt:

Diese Spaltenformel sortiert nach Erstellmaske:

```
@If(Form="frmASCII";"Verbindung zu Text";
      Form="frmODBC";"Verbindung zu ODBC";
      Form="frmNotes";"Verbindung zu Notes";
      Form="frmSelect";"Eigene Abfrage";
      "" )
```

Typ	Verbindung
Verbindung zu Notes	Wartungsdatenbank Pläne
Verbindung zu ODBC	OracleTest2 Reparatur Wartung Wartungsplan cols von ORA
Verbindung zu Text	PersDat-Flatfile AS400

Abbildung 8: Ausgabe der Ansichtsformel

Dies ist eine typische Spaltenformel, die Inhalte von Felder manipuliert und anzeigt:

```
@If (telnummer!="";
      (" "+@Trim(vorwahl)+" ) "+@Trim(telnummer);
      "-")
```

	Name	Telefonnummer	Faxnummer	Telefonnr (Arbeit)
nach Name	Bielak, Steffen	(02359) 903980	-	(02351) 965500
	Bosbach-Kleinen, Jürgen	-	-	(02262) 81 375
	CONET	(02242) 9390	(02242) 939 393	-
nach Ort	Czolbe, Wolfgang	-	-	(02262) 81 409
	Dehn, Christian	(02359) 4082	-	-
	Dimitiou, Kosta	-	-	(02261) 8196 405
nach Firma	Duman, Adnan	(02233) 977 150	-	-
	Faeskorn-Woyke, Heide	(02191) 963685	-	(02261) 8196 293
	FHS Gummersbach	(02261) 81960	-	-
Ende	Gallus, Marc	(02359) 6916	-	-

Abbildung 9: Ausgabe der Ansichtsformel II

5.1.6 Navigatoren

Man kann Navigatoren als grafische Hilfsmittel auffassen, mit deren Hilfe das Bewegen in der Datenbank erleichtert wird. Sie bieten einen visuellen Index für den Inhalt einer Datenbank und führen den Benutzer durch die Anwendung.

Es gibt zwei Arten von Objekten:

- Objekte die mit der Notes-Zeichenhilfe erstellt wurden.
- Objekte die importiert wurden (Grafiken).

Die Objekte können durch anklicken folgende Aktionen ausführen:

- Eine Ansicht öffnen
- Einen Ordner öffnen
- Eine Verknüpfung öffnen
- Einen anderen Navigator ausführen
- Eine Formel ausführen

...und Script-Code ausführen

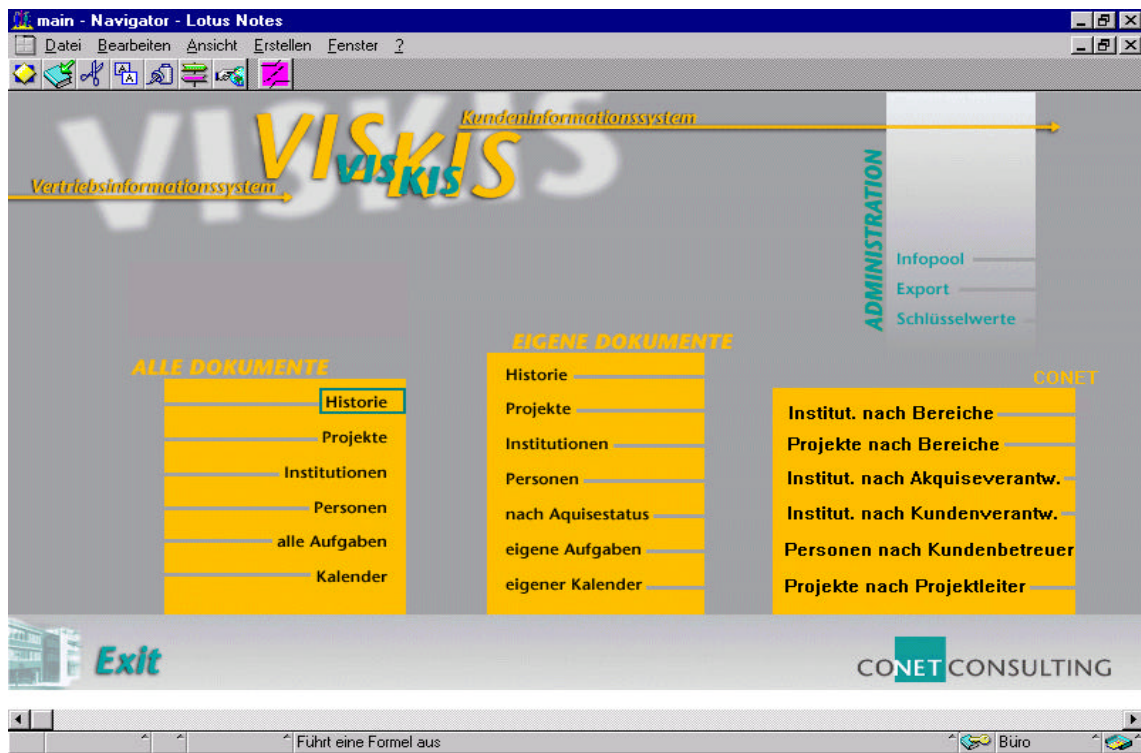


Abbildung 10: Screenshot von VIS KIS

5.1.7 Aktionen

Aktionen werden in Masken und Ansichten oben aus dem Bildschirm als Schaltflächen auf der Aktionsleiste angezeigt. Mit Aktionen werden Datenbanktasks ausgeführt, die die Durchführung einfacher Aufgaben automatisiert und Benutzerfehlern vorbeugt.

Standardaktionen sind im CONET-Styleguide gut beschrieben, man kann jedoch jegliche Script- oder Makroprogrammierung hinter einen Aktionsknopf legen.



Abbildung 11: Menu

Die Abbildung zeigt eine Bestelldatenbank, in der die Mitarbeiter unserer Abteilung das Bestellen des Mittagessens koordinieren.

Die Aktionen von links nach rechts:

- Schließen die Datenbank
- Verzweigen in den Programmfuß für eine „Neue Bestellung“
- Die Administratoren können „Neue Teilnehmer“ zulassen
- Alle Besteller werden per Mail benachrichtigt
- Information über aktuelle Angebote o.ä.

Kapitel 6

MÖGLICHKEITEN DER PROGRAMMIERUNG IN LOTUS NOTES

6.1 Lotus Script

Lotus Script ist eine Programmiersprache, die dem Visual Basic der Firma Microsoft sehr ähnlich ist. Lotus Script ist eine Interpretersprache. Es gibt auch keine Runtime-Version, so daß Lotus Script nur innerhalb von Lotusprodukten lauffähig ist.

Es werden Merkmale der objektorientierten Programmierung unterstützt, d.h. man kann eigene Klassen definieren und es wird mit Objekten, Klassen und Ereignissen gearbeitet.

Lotus Script kann man überall dort einsetzen, wo beim Entwickeln das Programmierfenster die Möglichkeit dazu anzeigt. Der Einsatzbereich überschneidet sich fast gänzlich mit dem der Makrosprache. Aber dazu mehr im Kapitel „Vergleich von Makrosprache und Lotus Script“.

Die Entwicklungsumgebung ist sehr ausgefeilt und bietet einen syntax-sensitiven Scripteditor.

In Lotus Script gibt es elf vordefinierte Datentypen:

Integer:	-32.768 bis 32.767	
Long:	- 2.147.483.648 bis 2.147.483.647	
Single:	-3,402823E+38 bis 3,402823E+38	
Double:	-1,797693148623158E+308 bis 1,797693148623158E+308	
Currency:	-922.337.203.685.477,5807 bis 922.337.203.685.477,5806	
String:	fix, variabel	
Array		
Liste		
Variant:	(universeller Typ)	
Benutzerdefinierte Datentypen:		In C heißt so etwas „Struktur“(engl. struct).
Klassen:		In etwa mit den Klassen von C++
	vergleichbar.	

Des weiteren sind Schleifenkonstrukte und Verzweigungen möglich.

6.2 Die Makrosprache

Sie besteht überwiegend aus Formeln, die Bestandteile von Feldern sind und beim Erstellen, Bearbeiten oder Speichern aktiv werden.

Vorweg gesagt gibt es in der Makrosprache keine Schleifen! Es gibt jedoch die Möglichkeit, zu verzweigen und Blöcke auszuführen.

Mit einem guten Verständnis der Formelsprache kann man schnell und effektiv Anwendungen entwickeln, man sollte sich jedoch immer der benutzten Datentypen sicher sein. Bei fehlerhaftem Umgang mit Datentypen treten die Auswirkungen erst beim Ausführen der Formeln zu Tage.

Gebräuchliche @Funktionen sind:

@If (Bedingung;Aktion;Sonst Aktion);

@If (Bedingung1;Aktion1; Bedingung2;Aktion2; Bedingung3;Aktion3;Sonst Aktion);

@Do (Anweisungsblock);

Zeichenkettenoperationen:

@Left(Textstring;Position);

@Left(Textstring;Zeichen);

@Trim(Textstring);

@Uppercase(Textstring); ...

Zeit-/Datumsfunktionen:

@Now; **@Today**; ...

Stoppen der Bearbeitung:

@Return;

Umwandlung

@Text; **@TextToTime**; **@TextToNumber**;

Fehlerbehandlung:

@Success; **@Failure**(„Fehlertext“);

Feldwerte setzen:

@SetField(„Feldname“ ; Wert)

Mit **@Prompt**(); kann man Dialogboxen der verschiedensten Typen erzeugen.

6.3 Ein Vergleich von Makrosprache und Lotus Script

Ein Entscheidungsträger bei dieser Wahl ist der ganz persönliche Geschmack des jeweiligen Entwicklers. Aber davon abgesehen gibt es in der Notes-Programmierung auch zwingende Faktoren.

Sehen Sie sich zum Beispiel die folgende Gegenüberstellung an. Beide Codes dienen dazu eine recht einfache Listenmanipulationen durchzuführen.

Makrosprache:

```
Eintrag:= fldNotesFelderAnzeige ;
@if(Eintrag="" ;@Return(0) ; "");
FIELD fldNotesFelder:=
  @Replace(fldNotesFelder;Eintrag; "");
FIELD fldSelNotesFelder:=
  fldSelNotesFelder:Eintrag;
@Command([ViewRefreshFields])
```

Script:

```
Sub Click(Source As Button)
  Dim ws As New notesuiworkspace
  Dim uidoc As Notesuidocument
  Set uidoc =ws.currentdocument
  Dim sSelNotesFelder As String,sNotesFeld() As String,
  sAktNotesFeld As String, sTemp As String
  Dim i As Integer, iAnzeigePos As
  IntegerselNotesFelder=uidoc.fieldgettext("fldSelNotesFelder")
  i=0
  While(sSelNotesFelder<>"")
```

```

        If (Instr(sSelNotesFelder, ";")) Then
            Redim Preserve sNotesFeld(i) As String

            sNotesFeld(i) = Left(sSelNotesFelder, Instr(sSelNotesFelder, ";") - 1)

            sSelNotesFelder = Right(sSelNotesFelder, Len(sSelNotesFelder) -
            Len(sNotesFeld(i)) - 2)
            i = i + 1
        Else
            Redim Preserve sNotesFeld(i) As String
            sNotesFeld(i) = sSelNotesFelder
            sSelNotesFelder = ""
            i = i + 1
        End If
    End If
Wend

sAktNotesFeld = uidoc.fieldgettext("fldSelNotesFelderAnzeige")
For i = Lbound(sNotesFeld) To Ubound(sNotesFeld)
    If (sAktNotesFeld = sNotesFeld(i)) Then
        If (i = 0) Then
            Exit Sub
        End If
        iAnzeigePos = i - 1
        sTemp = sNotesFeld(i - 1)
        sNotesFeld(i - 1) = sNotesFeld(i)
        sNotesFeld(i) = sTemp
    End If
Next
Call uidoc.fieldclear("fldSelNotesFelder")
For k = Lbound(sNotesFeld) To Ubound(sNotesFeld)
    Call
uidoc.FieldAppendText("fldSelNotesFelder", sNotesFeld(k) + ";")
Next
Call
uidoc.FieldSetText("fldSelNotesFelderAnzeige", sNotesFeld(iAnzeig
epos))
Call uidoc.refresh
End Sub

```

Was unverkennbar ist: „Der Codieraufwand ist in diesem Falle beim Script-Programmieren höher.“

Was nicht jeder weiß: „Der Script-Code ist effektiv langsamer!!!“

Script-Code wird zwar nicht wirklich jedesmal wieder interpretiert, er wird in sogenannten Token abgelegt. Token sind eine Art Metacode. Trotz und alledem sind die Makrobefehle prinzipiell schneller.

Der Grund ist: „Die @Befehle der Makrosprache sind in C geschrieben und kompiliert“.

Doch in der Makrosprache ist vieles einfach nicht möglich. Abgesehen davon, daß man kein Schleifenkonstrukt zur Verfügung hat, fehlen viele Möglichkeiten der Manipulation.

Ein Beispiel hierzu ist die Zugriffskontrollliste (engl. ACL). Mit der Makrosprache kann man nur Abfragen auf die ACL starten. Mit Lotus Script kann man auf die Personen, Gruppen und Server in der ACL zugreifen, ihnen Rechte und Rollen geben und zuweisen, diese auch entziehen und letztendlich kann man die Personen, Gruppen und Server auch löschen.

Es sei der Fall gegeben, daß ich dem aktuellen Benutzer folgende drei Rollen zugeteilt habe. So wird mit dem Makrobefehl @userroles folgende Ausgabe erzeugt:

```
[DokuAutor]; [DokuGenehmiger]; [DokuManager]
```

In der Makrosprache ist @userroles die einzige Möglichkeit um auf die ACL zuzugreifen. Er zeigt die zugewiesenen Rollen des derzeit angemeldeten Benutzers an.

Lotus Script bietet mit den Klassen "NotesACL" und "NotesACLEntry" weit mehr als nur eine Möglichkeit zur Anzeige der Rollen. Hier jedoch zuerst einmal ein Beispiel für die Anzeige der Rollen eines jeden Benutzers.

```
Sub Click(Source As Button)
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim acl As NotesACL
    Dim entry As NotesACLEntry
    Dim sAusgabe As String
    Set db = session.CurrentDatabase
    Set acl = db.ACL
    Set entry = acl.GetFirstEntry
    Do
        Forall xy In entry.roles
```

```

        sAusgabe=sAusgabe & entry.Name &Chr(9) &Chr(9) & xy
        & Chr(10)
    End Forall
    Set entry= acl.GetNextEntry(entry)
    Loop Until (entry Is Nothing)
    MsgBox sAusgabe
End Sub

```

Ausgabe:

Manuela Manager	[DokuGenehmiger]
Manuela Manager	[DokuManager]
Stefan Schreiber	[DokuAutor]
Karlo Könner	[DokuGenehmiger]

Jetzt folgt ein Beispiel zum Erstellen von Einträgen in der ACL. Dies erreicht man mit der Methode "CreateACLEntry" der NotesACL-Klasse.

```
Set notesACLEntry = notesACL.CreateACLEntry(sName, iLevel)
```

Mit dem String "sName" wird der Name definiert und mit iLevel legt man die Befugnisse fest. In der "LSCONST.LSS" sind folgende Konstanten definiert, die den Quelltext lesbarer machen:

ACLLEVEL_NOACCESS	No access
ACLLEVEL_DEPOSITOR	Depositor access
ACLLEVEL_READER	Reader access
ACLLEVEL_AUTHOR	Author access
ACLLEVEL_EDITOR	Editor access
ACLLEVEL_DESIGNER	Designer access
ACLLEVEL_MANAGER	Manager access

Folgendes Beispiel erstellt einen Benutzer "Susie Schlau" mit Editor-Zugriff und weist die Rolle "DokuAutor" zu.

```
Sub Click(Source As Button)
```



```
Dim session As New NotesSession
Dim db As NotesDatabase
Dim acl As NotesACL
Dim entry As NotesACLEntry
Set db = session.CurrentDatabase
Set acl = db.ACL
Set entry = acl.CreateACLEntry("Susie
Schlau",ACLLEVEL_EDITOR)
entry.EnableRole("DokuAutor")
acl.save
End Sub
```

Ausgabe:

Manuela Manager	[DokuGenehmiger]
Manuela Manager	[DokuManager]
Stefan Schreiber	[DokuAutor]
Karlo Könner	[DokuGenehmiger]
Susie Schlau	[DokuAutor]

...aber auch dieses Werkzeug hat seine Grenzen.

6.4 Die C-API

Mit Lotus Script kann man keine Design-Elemente ermitteln, geschweige denn, manipulieren. Wo man mit Lotus Script nicht mehr weiterkommt, da kann einem die C-API helfen.

Es gibt zum Einen die direkte Möglichkeit von Lotus Script heraus Aufrufe zu starten. Folgende Abbildung zeigt einen Statusbalken, den man z.B. in eine Installationsroutine einbinden kann.

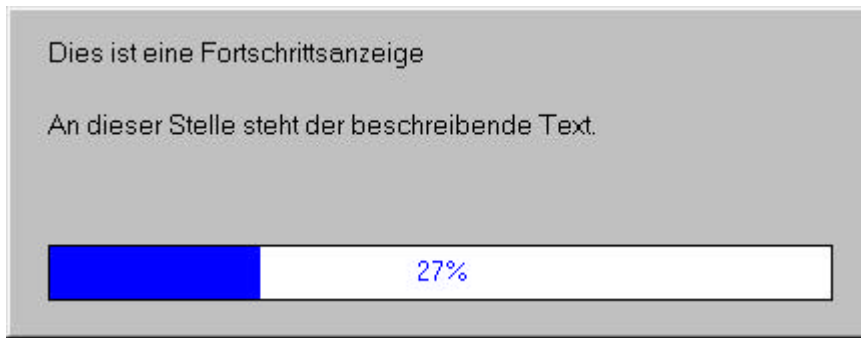


Abbildung 12: Fortschrittsanzeige

Diese Anzeige kann man durch folgenden Quelltext erreichen:

```

Declarations
Declare Sub NEMProgressSetText Lib "nnotesws.dll" ( Byval hwnd As Long,
Byval pcszLine1 As String, Byval pcszLine2 As String )
Declare Sub NEMProgressEnd Lib "nnotesws.dll" ( Byval hwnd As Long )
Declare Function NEMProgressBegin Lib "nnotesws.dll" ( Byval wFlags As
Integer ) As Long
Declare Sub NEMProgressSetBarPos Lib "nnotesws.dll" ( Byval hwnd As
Long, Byval dwPos As Long)
Declare Sub NEMProgressSetBarRange Lib "nnotesws.dll" ( Byval hwnd As
Long, Byval dwMax As Long )
Sub Click(Source As Button)
    Const GesamtlängeStatusanzeige = 100
    'Zweizeilige Darstellung : Flag auf 1 setzen...
    hwnd = NEMProgressBegin(1)
    NEMProgressSetBarRange hwnd, GesamtlängeStatusanzeige
    'Die Anzeigeposition auffrischen
    For i =1 To GesamtlängeStatusanzeige
        For k=1 To 10000
            'Verzögerungsschleife
        Next
        NEMProgressSetBarPos hwnd, i
        NEMProgressSetText hwnd,"Dies ist eine Fortschritt-
sanzeige","An dieser Stelle steht der beschreibende Text."
        i=i+1
    Next
    NEMProgressEnd hwnd
End Sub

```

Nun wird die zweite Möglichkeit der API-Programmierung anhand eines aktuellen Beispiels beschrieben.

„Meine Aufgabenstellung ist es, Daten aus einer ASCII-Datei einzulesen, die Bitmapnamen, Position, Höhe, Breite, Objekttext, Aktion und ähnliche Informationen enthält.

Mit diesen Informationen soll automatisch in einer gegebenen Notes-Datenbank eine Anzahl von Navigatoren generiert werden. Die Hintergrundbilder sind einzufügen und klickbare Objekte und Ihre gespeicherten Aktionen sind zu erstellen.“ In der Hilfe zu Lotus Script steht zu der Klasse Navigator nur geschrieben: „Die Klasse Navigator hat weder Eigenschaften noch Methoden“.

Man braucht einen C-Compiler (mit MS Visual C++ 5.0 funktioniert es) und eine zu der Notes-Version passende C-API (die Version 4.6 ist frei verfügbar).

Weil der Umfang des Quelltextes zu groß ist und er den Rahmen dieser Arbeit sprengt, lege ich ihn auf CD bei (makenav.c).

6.5 Was unterscheidet eine Lotus-Notes-Datenbank von einer relationalen Datenbank ?

Grundsätzlich können sich Lotus Notes und RDBMS gut ergänzen, da sie aufgrund ihrer Konzipierung unterschiedliche Anwendungstypen unterstützen. Durch die Möglichkeit des Datenaustauschs können sehr effektive Anwendungen implementiert werden.

Um entscheiden zu können, wann Notes für eine (Teil-)Anwendung sinnvoll ist, sollten die Unterschiede wie folgt bekannt sein:

RDBMS	Lotus Notes
Basiert auf einem	Basiert auf einem

Transaktionsverarbeitungsmodell	Dokumentverarbeitungsmodell
Arbeitet mit strukturierten Daten. (Jedoch gibt es mittlerweile den Datentyp BLOB (Binary large objects))	Arbeitet mit halbstrukturierten Datenelementen (Realisiert über RichText-Felder)
Bietet Echtzeitzugriff auf Daten	Verwendet regelmäßige Replizierung zur Aktualisierung
Datensuche über direkte Abfrage (SQL oder Bericht)	Datensuche durch Ansichten, die die Daten mittels Sortierkriterien organisiert darstellen, oder durch Volltextsuche (wenn gewünscht Indexgestützt)
Zentrale Datenhaltung	Dezentrale Datenhaltung

Tabelle 1: Vergleich von RDBMS und Lotus Notes

Kapitel 7

ODBC

7.1 Was ist ODBC?

ODBC steht für Open Database Connectivity und ist eine definierte standardisierte Schnittstelle zu den vielen proprietären Datenbanksystemen.

ODBC ist eine plattformübergreifende Lösung. Die Programmierung wird durch vorgefertigte C-Klassen, durch Datenbankobjekte unter Visual C++ u.v.m. erleichtert.

Durch ODBC wird es möglich Anwendungen zu schreiben, die nicht auf ein ganz spezielles Datenbanksystem zugeschnitten sind.

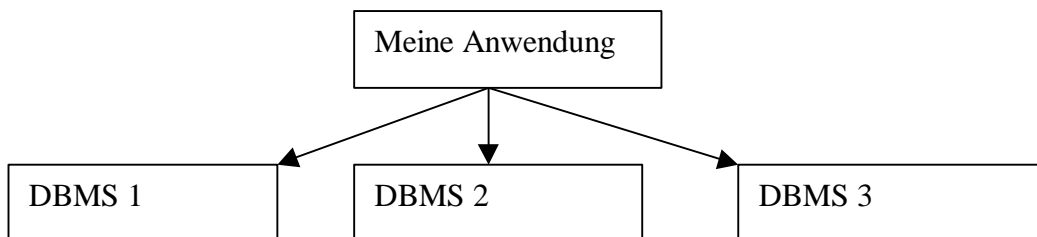


Abbildung 13: Die Notwendigkeit der Datenbank-Connectivity aus der Sicht des Anwendungsentwicklers [Kyle Geiger 1996, Kapitel 1, Seite 27]

Um die DBMS-Unabhängigkeit zu erlangen, schafft die ODBC-Architektur eine logische Abgrenzung von Anwendung und DBMS.

7.2 Wieso wurde ODBC nötig?

An dieser Stelle steht das Schlagwort „Database Connectivity“ im Raum.

Die Datenbankhersteller dachten nicht gerne daran, den Kunden die Migration auf andere Systeme durch Standardisierung ihrer Schnittstellen einfacher zu machen. Sie wollten eine feste Kundenbindung an ihr Produkt. Aber es wurde auf die Datenbankhersteller seitens der Kunden und Anwendungsentwickler Druck ausgeübt. Das waren vor allem Kunden, die entweder bereits mehrere DBMS installiert hatten, oder die sich nicht an die Datenbank, die Programmierschnittstelle oder die Entwicklungswerkzeuge eines einzelnen Herstellers binden wollten.

Insbesondere die amerikanische Regierung verlangte Standards und da man große Aufträge in Gefahr sah, reagierten die Datenbankhersteller sehr schnell.

So sah es aus vor ODBC aus:

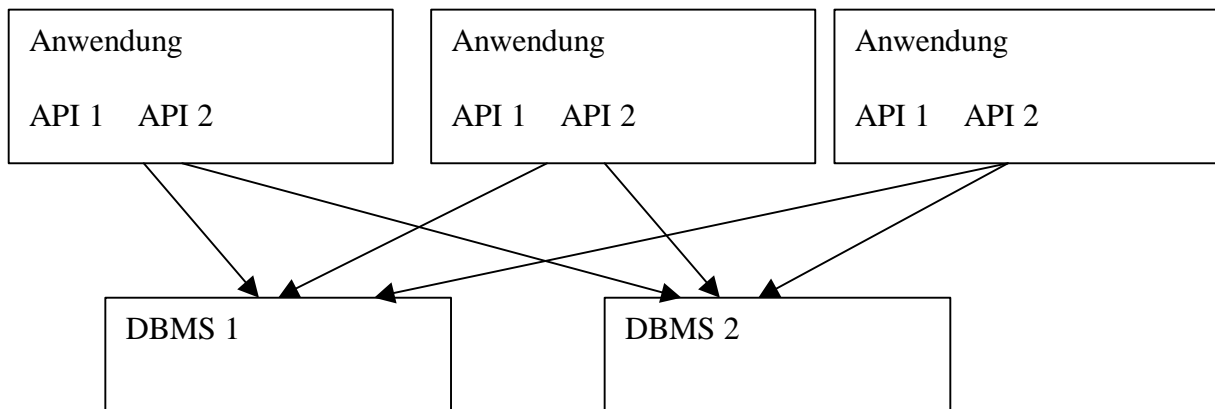


Abbildung 14: Die Notwendigkeit der Datenbank-Connectivity aus der Sicht des Benutzers [Kyle Geiger 1996, Kapitel 1, Seite 26]

7.3 Exkurs: Architekturen der Datenhaltung

Um die Thematik besser zu beleuchten, werde ich in einem kleinen Exkurs auf die ehemals vorhandenen Architekturen eingehen, die es zu verbinden galt.

Hier sollen drei Modelle beschrieben werden. Unter den herkömmlichen RDBMS verstehe ich Oracle, DB2, MS SQL Server, Informix, Ingres usw.

Als zweites Modell beschreibe ich Datei-Management-Systeme und die Erweiterung derer namens „ISAM“.

Das dritte und letzte Modell bieten die Desktop-Datenbanken wie: MS Access, dBase, FoxPro, Paradox usw.

7.3.1 Herkömmliche relationale DBMS

Die Architektur besteht im Prinzip aus den folgenden sechs Komponenten.

- Benutzeroberfläche (GUI)
- Anwendung
- Datenbank-API
- Netzwerk-/Kommunikationssoftware
- DBMS
- Datenspeicher

Die Anwendung ist verantwortlich für die Interaktion mit dem Benutzer und die Bereitstellung der Benutzeroberfläche.

Eine Datenbank-API stellt Methoden zur Verfügung um mit dem RDBMS zu interagieren. Es müssen SQL-Anweisungen an das RDBMS übergeben, sowie Ergebnisse an die Anwendung zurückgegeben werden. ODBC ist so eine API, wie auch OCI von Oracle oder DB-Library von Sybase.

Die Netzwerk-/Kommunikationssoftware baut Verbindungen zu anderen Rechnern auf, und beendet sie. Sie liest oder schreibt Informationsströme über diese Verbindungen.

Das DBMS umfaßt einen SQL-Parser, einen Optimierer, ein Ausführungsmodul und viele weitere Dienste zur Datenverwaltung.

7.3.2 Datei-Management-Systeme und ISAMs

Datei-Management-System, auch Flatfiles, flache Dateien genannt, sind ein äußerst primitives Mittel der Datenhaltung. Man verwendet Funktionen der jeweils benutzten Programmiersprache zum:

- Öffnen und Schließen der Dateien
- Positionieren des Cursors
- Lesen und Schreiben

Die semantische Information über den Aufbau der Zeilen, die in der Datei vorhanden ist, muß in der Anwendung codiert sein.

Ein C-Beispiel ist das Struct, welches ich z.Zt. in einem Projekt benutze:

```
typedef struct{
    char bildname[20];
    char position[7];
    char hoehe[4];
    char breite[4];
    char eoln[2];
}itemdefinition;

itemdefinition einitem;
```


Erzeugt man ein Array mit diesem Typ und durchläuft man eine ASCII-Datei zeilenweise, kann man in den Objekten seine Information ablegen.

Mit steigender Komplexität kann man in der Realität viele Nachteile erkennen.

- Wenn Sie das Datenformat ändern, müssen Sie die Anwendung neu codieren
- Haben Sie mehrere Anwendungen, müssen Sie mehrere Anwendungen neu codieren
- Einen Datensatz zu suchen, erfordert das zeilenweise Durchlaufen der Datensätze
- Relationen lassen sich nur mit hohem Codieraufwand darstellen
- Wenn mehrere Anwendungen gleichzeitig zugreifen, ist die Integrität der Daten in Gefahr

ISAM (Index Sequential Access Methods, indexsequentielle Zugriffsmethoden)

Die ISAMs stellen einen Fortschritt dar. Es gibt Sperrprimitiven auf Dateibasis oder auf Datensatzbasis. Mit Hilfe von Sperrern können die Anwendungen eine ganz einfache Form der Transaktionsverarbeitung realisieren. Dabei müssen alle Anwendungen die gleiche Sperrsemantik benutzen. Indize ermöglichen eine schnelle Sortierung und Suche. Mit ihnen kann man Datensätze oder eine Menge von Datensätzen schneller finden. Die folgende Tabelle zeigt Funktionsaufrufe der X/OPEN ISAM-Programmierschnittstelle [Kyle Geiger 1996, Kapitel 2, Seite 72].

Funktionsname	Funktionalität
Isopen	Öffnen einer ISAM-Datei
Isclose	Schließt eine ISAM-Datei
Isread	Liest einen Datensatz
Iswrite	Schreibt einen Datensatz
Isdelcurr	Löscht den aktuellen Datensatz
Isaddindex	Fügt einer ISAM-Datei einen Index hinzu
Isdelindex	Entfernt einen Index aus einer ISAM-Datei

Isstart	Selektiert einen Index
Islock	Sperrt eine ISAM-Datei
Isrelease	Hebt die Sperre für einen Datensatz auf
Isunlock	Hebt die Sperre für einen ISAM-Datei auf

Tabelle 2: Funktionsaufrufe der X/OPEN ISAM-Programmierschnittstelle

Man sieht, daß die ISAM-Funktionen zeilen- bzw. datensatzorientiert sind. Da SQL mengenorientiert arbeitet, versteht man nun auch besser, daß es nicht einfach ist, Operationen auf das jeweils andere System abzubilden.

Als Beispiele für ISAM-basierte Systeme seien VSAM von IBM, DEC's RMS-Dateien und Guardian von Tandem genannt.

7.3.3 Desktop-Datenbanken

Desktop DBMS-Systeme sind ähnlich konzipiert wie die schon beschriebenen herkömmlichen RDBMS. Ihnen fehlen lediglich mehr oder weniger die komplexen Werkzeuge um die Arbeit mit mehreren gleichzeitigen Benutzern, Transaktionen und andere Funktionen effektiv zu realisieren. Der Grund für die Verbreitung von Access, dBase oder Paradox liegt darin, daß sie nicht so speicherintensiv und performanceträchtig erscheinen wie die „Großen“. Ein anderes Argument ist auch die einfache Installation und Wartung.

7.4 Die ODBC-Architektur

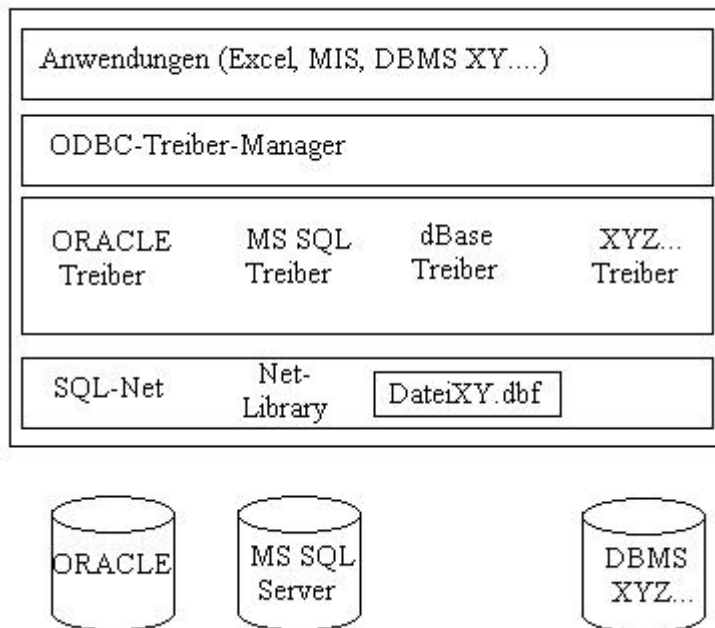


Abbildung 15: ODBC Architektur

7.5 Historisches über ODBC

In den '80er Jahren war es noch so, daß die meisten Firmen Mainframe-Rechner und Datenbanksysteme benutzten. Mit der Zeit nahm aber mehr und mehr die Anzahl der PC in den Büros zu.

Damit begann auch die Phase, in der diese PC's zur Datenanalyse eingesetzt wurden. Im geringeren Umfang wurde auch Datenmanipulation durch PC's betrieben und zur selben Zeit nutzten viele kleinere Firmen Desktop-Datenbankprodukte.

Um jedoch diese unterschiedlichen Quellen anzapfen zu können, mußte man sehr großen Aufwand treiben.

ODBC begann als Projekt, dessen Ziel es war, das Problem des Datenzugriffs auf unterschiedliche Datenbanken vom PC aus zu lösen.

Das Interesse des Anwenders war klar. Der Anwender wollte eine einfach zu bedienende und obendrein noch performante Schnittstelle.

Die Belange der Anwendungsentwickler lagen so, daß sie durch eine Standardschnittstelle viel Arbeit einsparen wollten.

1988: gab es seitens Microsoft erste Spezifikationen für die Datenbank-Connectivity Technologie. Zur selben Zeit hatte die Firma Lotus ein Produkt namens „Blueprint“ fertig. Dieses gestattete Lotus-Software den Zugriff auf verschiedene Datenquellen und war in die Lotus-Produkte integriert.

Bei Microsoft und Lotus kamen Gespräche in Gang, die aber zu nichts führten. Man begründete dies mit unterschiedlichen technischen Ansätzen.

1989: kristallisierten sich zwei Ansätze der Database-Connectivity heraus. Interessanterweise hießen sowohl der Microsoft-Ansatz, als auch der Tandem-(und der Rest der DB-Firmen) Ansatz „Open SQL“. Technisch gesehen waren die beiden jedoch anders gebaut und inkompatibel. Erwähnenswert ist für dieses Jahr noch die absolute Dominanz der IBM auf dem Gebiet der Datenbankforschung.

Im Sommer dieses Jahres wurde aus Open SQL u.a. unter Mitwirkung von Lotus, Microsoft und Sybase SQLC. Es zeichnete sich ab, das die Entwicklung auf einen SQL-basierten Standard zugehen würde

Trotzdem gab es von Lotus-Blueprint eine neue Revision namens Datalens.

1990: wurde SQL zur offenen Spezifikation die von jedem beeinflußt werden konnte, der an den Konferenzen der SQL-Access-Group (SAG) teilnahm. (Vorher nur DEC, Microsoft, Lotus und Sybase).

1992: nannte man die überarbeitete SAG-Spezifikation dann ODBC.

1993: Obwohl drei verschiedene Standardisierungsorganisationen (SAG, ANSI und ISO) an der Spezifikation beteiligt waren, wurde Ende 1993 ODBC zur Konformitätsstufe 1 entwickelt.

1995: ODBC 2.5 Erweiterung durch die Implementierung der System-DSN.

1996: ODBC 3.0 OLE-Unterstützung, bessere Installationsroutinen, sowie viele Verbesserungen an den Funktionen kennzeichnen die Version 3.

7.6 Adaptive Programmierung und Konformitätsstufen

Da die DBMS der unterschiedlichen Hersteller nicht ein und dasselbe leisten konnten, mußte man sich auf eine Kernfunktionalität einigen.

Der Begriff der adaptiven Programmierung sagt in diesem Zusammenhang aus, daß der Anwendungsprogrammierer Funktionen aufrufen kann, die die Fähigkeiten des DBMS abfragen.

Er kann anhand dieser Resultate bestimmen, welchen Funktionsumfang der spezifische DBMS-Treiber bietet und daraufhin dynamisch die Funktionen ansteuern, die dem gegenwärtigem Leistungsumfang entsprechen.

Ganz klar war, daß bei der Entwicklung von ODBC erst einmal der kleinste gemeinsame Nenner gefunden werden mußte. In diesem Zusammenhang wurden dann auch jeweils drei Konformitätsstufen für die API-Konformität und für die SQL-Konformität definiert.

Man kann die SQL-Konformitätsstufe über die Funktion „SQLGetInfo“ ermitteln. Die API-Konformität gibt die Funktion „SQLGetFunctions“ zurück.

Ein Treiber muß mindestens alle SQL-Anweisungen unterstützen, die für seine Stufe vereinbart sind.

7.6.1 Die API-Konformität

In der Stufe „Kern“ gewährleistet der Treiber die Verwaltung einfacher Verbindungen.

Er kann SQL-Anweisungen in einem Aufruf vorbereiten, ausführen und an das DBMS senden.

Das Laden der zurückgelieferten Daten in Variablen wird durch eine einfache Cursorverwaltung unterstützt.

Transaktionen können rückgängig gemacht oder festgeschrieben werden.

Es existiert eine Standardmethode für die Fehlerbehandlung.

Die „Stufe 1“ erweitert die Funktionalität zur Verbindungsverwaltung.

Große Datenwerte, wie sie in BLOBs vorkommen, können etappenweise geladen werden.

Funktionen ermöglichen es, die Fähigkeiten des Treibers abzufragen.

Per „SQLGetInfo“ kann ermittelt werden, wie die Abbildung der Datentypen von Quelle und Ziel aussieht.

In der „Stufe 2“ sind „Scrollbare Cursor“ implementiert, es gibt eine erweiterte Verwaltung für Verbindungen und es gibt Funktionen mit denen man Benutzerrechte und Integritätsinformationen (Schlüssel) abfragen kann.

7.6.2 Die SQL-Konformität

„**Minimales SQL**“ leistet die absolute Grundfunktionalität. Es unterstützt folgende Anweisungen und Datentypen:

- CREATE TABLE ,
- DROP TABLE ,
- SELECT , INSERT ,

-
- UPDATE ,
 - DELETE .
 - CHAR ,
 - VARCHAR ,
 - LONG VARCHAR .

Das „**Kern-SQL**“ erweitert den Funktionsumfang um folgende Anweisungen und Datentypen:

- ALTER TABLE ,
- CREATE/DROP INDEX ,
- CREATE/DROP VIEW ,
- GRANT ,
- REVOKE .
- DECIMAL ,
- NUMERIC ,
- SMALLINT ,
- INTEGER ,
- REAL ,
- FLOAT ,
- DOUBLE PRECISION .

Für erweiterte SELECT-Anweisung beherrscht „Kern-SQL“ auch Unterabfragen und Mengenfunktionen.

Beim „**Erweiterten SQL**“ wird die Darstellung von Zeit-/Datumstypen möglich. Es beherrscht Outer-Joins sowie skalare Funktionen und die Ausführung gespeicherter Prozeduren.

7.7 **Schlußbemerkung zum ODBC-** **Kapitel**

Es ist bemerkenswert, wie sich der Gebrauch dieser API durchgesetzt hat. Meiner Meinung nach hat die Marktmacht von Microsoft einen wesentlichen Teil dazu beigetragen.

Häufig eingesetzt wird ODBC in den Bereichen Data Warehouse, im Berichtswesen und MIS, bei der Integration von Datenbeständen, bzw. kompletten DBMS und bei der Umstellung auf neue DBMS.

Zu guter Letzt möchte ich noch eine nette Analogie beschreiben, die Kyle Geiger, der Autor von „INSIDE ODBC“ dazu benutzte den Erfolg von ODBC zu erklären. Kyle sah Parallelen zu der Ära in der es noch für jede Anwendung eigene Bildschirm-, Grafikkarten- und Druckertreiber gab.

Zu dieser Zeit gab es zu einer neuen Grafikkarte eine Diskette, auf der fünfzehn verschiedene Verzeichnisse waren. In diesen Verzeichnissen waren Treiber für Autocad, Word für DOS, DOS, Windows und für sehr viele Programme von deren Existenz heute kaum noch jemand weiß.

Das war so, weil manche Softwarehersteller glaubten, bessere Performance oder höhere Funktionalität als die Standardtreiber realisieren zu können. Aber in der Zeit, in der die Programmierer dieser Hersteller neue Treiber schrieben, gab es neue Hardware mit neuer Funktionalität und dazu auch neue Standardtreiber.

Die Softwarehersteller liefen in der Konsequenz ständig der Entwicklung hinterher und steckten „Man-Power“ in einen Geschäftszweig, der für die Akzeptanz der Anwendung unwesentlich war.

Das Ergebnis dieser Entwicklung ist, daß man heute nur noch selten Anwendungen findet, die ihre eigenen Treiber mitbringen und daß auf den Treiber-Disketten der Hardwarehersteller nun erheblich weniger Verzeichnisse sind.

Diese kleine Geschichte sollte nun die Parallelen zur Entwicklung im Datenbank-API-Bereich deutlich gemacht haben.

Kapitel 8

DIE PROBLEMSTELLUNG

8.1 Die Problemstellung beim Textimport

Es sollen zwei verschiedene Formate von Text gelesen werden.

Das eine Format ist „Separierter Text“. Dies heißt, daß zwischen den Datensätzen ein eindeutiges Trennzeichen steht.

Die richtige Wahl des Separators ist wichtig, denn wenn ein Datensatz den Separator auch als Inhalt enthält, kann das Programm diesen nicht richtig einlesen.

Diese Wahl fällt jedoch in den Bereich der Aufgaben des Benutzers.

Bislang werden folgende Separatoren unterstützt: ; # , \$ % Tabulator und Leerzeichen.

Ein Beispiel für separierten Text:

```
Berthold#Junghans#Fichtenweg 15#24939#Flensburg
D,#Elsner#Luisenstr. 18#28717#Bremen
Irmgard#Pfänder#Flughafenstrasse 81#01099#Dresden
Thorsten#Vaupel#Hansaring 12#99096#Erfurt
Nina#Giebert#Kaiserstrasse 121#99096#Erfurt
Anja#Bachenberg#Theodor-Heuss-Allee 2#15234#Frankfurt/ Oder
```

Beispiel für einen schlecht gewählten Separator:

Berthold/Junghans/Fichtenweg 15/24939/Flensburg
 D./Elsner/Luisenstr. 18/28717/Bremen
 Irmgard/Pfänder/Flughafenstrasse 81/01099/Dresden
 Thorsten/Vaupel/Hansaring 12/99096/Erfurt
 Nina/Giebert/Kaiserstrasse 121/99096/Erfurt
 Anja/Bachenberg/Theodor-Heuss-Allee 2/15234/Frankfurt/ Oder

Das zweite Format ist Text mit festen Feldlängen.

Eine Zeile besteht zum Beispiel immer aus den Namen mit der Länge von 20 Zeichen und dem Vornamen mit der Länge von 25 Zeichen.

Hier ein besonders übles Beispiel, in dem man die festen Feldlängen erst mal ermitteln mußte. Dies ist eine von 2000 Zeilen:

SAP0101998M00200G10520DM	146,3	3554,8	3235,9			
793,7	8919,3	37,2	508,8	113,5	659,4	
8259,9	155,2	166,6	321,8	4993,7	479,5	
5473,1	449,4	2465,1	147,3	0,0	2193,9	
124,1	-367,7	20,9	140,0	139,4	156,0	
0,0	22,9	376,5	855,5	0,0	-363,7	-6,6
22,8	0,0	79,9	-472,9	0,0	0,0	0,0
0,0	0,0	0,0	0,0	0,0	0,0	116,9
0,4	-589,4	0,0	-589,4	855,5	-363,7	0,0
1554,1	1554,1	7387,9	430,8	4960,2	8319,3	
1093,4	14803,6	2406,7	1052,7	2910,2	6369,4	
17376,1	5754,0	1344,4	24474,3	0,0	24474,3	
0,0	16155,1	17376,1	40,5	46,0	13,9	
66,3	76,2	29,9	-11,2	-4,4	-25,1	-14,6
-5,7	-23,1	-18,2	-7,1	-28,8	-18,2	-7,1
-43,7	-11,2	-4,4	-25,1			

Bei beiden Formaten stellt sich manchmal die Frage, ob der Text wirklich mit der ersten Zeile beginnt, oder ob die erste/ersten Zeile/n zur Beschreibung der Datensätze dienen.

Vorname#Nachname#Strasse#PLZ#ORT

Berthold#Junghans#Fichtenweg 15#24939#Flensburg

D,#Elsner#Luisenstr. 18#28717#Bremen

Irmgard#Pfänder#Flughafenstrasse 81#01099#Dresden

Thorsten#Vaupel#Hansaring 12#99096#Erfurt

Nina#Giebert#Kaiserstrasse 121#99096#Erfurt

Anja#Bachenberg#Theodor-Heuss-Allee 2#15234#Frankfurt/ Oder

8.1.1 Der Excel-Textimport als Beispiel für eine gelungene Benutzerschnittstelle

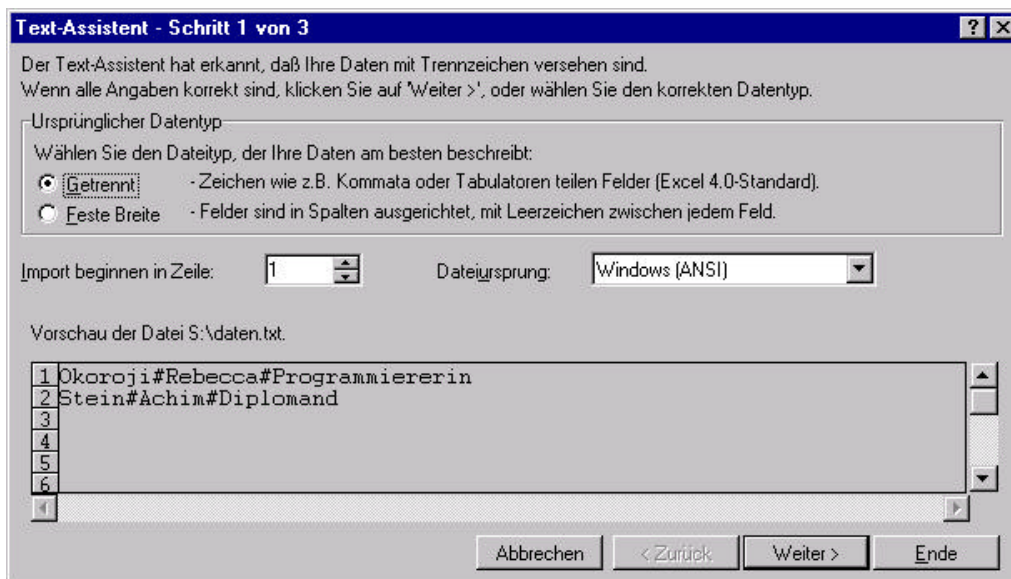


Abbildung 16: Eingabe Trennzeichen oder feste Feldlänge

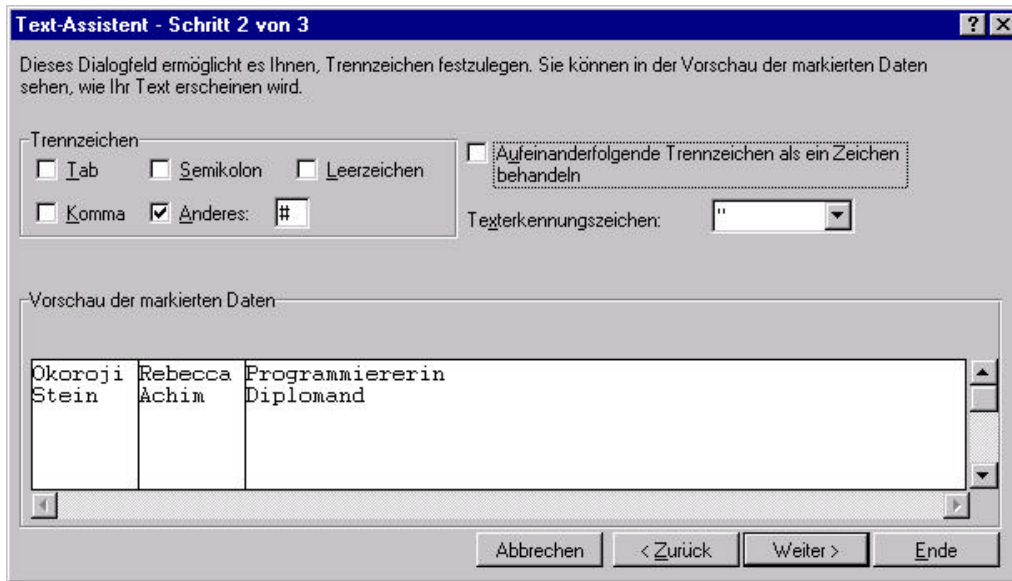


Abbildung 17: Festlegung des Trennzeichens

8.2 Die Problemstellung bei ODBC

Da ODBC die Daten aus anderen Systemen strukturiert darbietet, war es die Aufgabe des Tools sich der API zu bedienen und dem Benutzer ein leicht zu verstehendes Interface an die Hand zu geben.

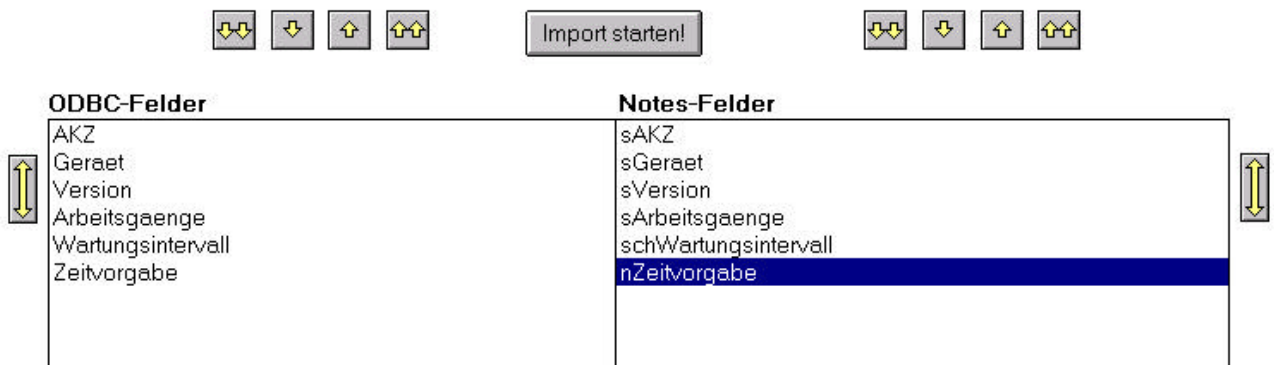


Abbildung 18: Userinterface

8.3 Die Problemstellung bei Lotus Notes

Sie werden denken, daß ein Import von Notes-Daten nach Notes irgendwie widersinnig klingt.

Oft ergeben sich aber Aufgabenstellungen, in denen man Feldinhalte einer bestimmten Notes-Datenbank irgendwo anders braucht. Ich finde es unnötig, jedesmal einen Agenten dafür zu schreiben.

Auf dieser Basis habe ich den Programmteil „Von Notes nach Notes“ implementiert.

Kapitel 9

VORBEREITENDE MAßNAHMEN

9.1 ODBC - Maßnahmen im Vorfeld

Die generellen Voraussetzungen möchte ich am Beispiel von Windows32-Systemen vorstellen.

Sie brauchen für jede ODBC-Quelle einen geeigneten Treiber. Eine ganze Menge von Treibern installieren sich schon automatisch bei der Installation von MS Office.

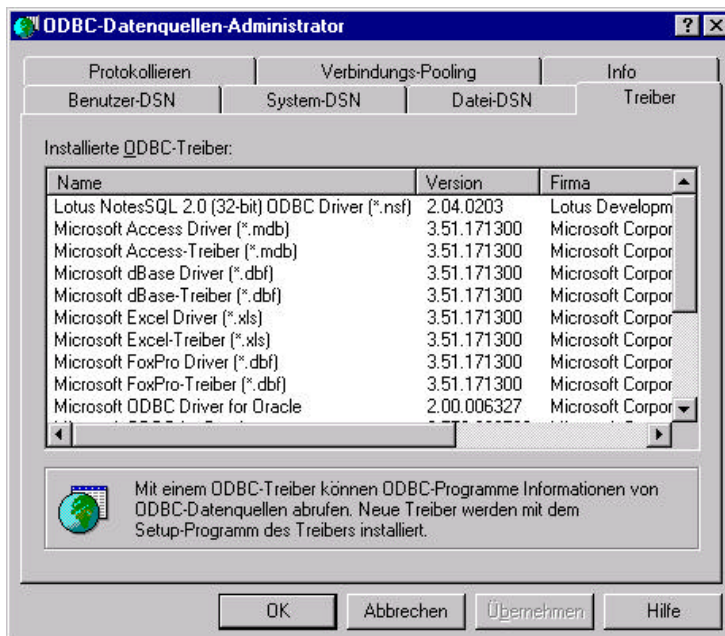


Abbildung 19: ODBC Treiber

Falls der von Ihnen gewünschte Treiber nicht dabei ist, rufen Sie den Support für Ihr Datenbank-System an. Um zu der oben gezeigten Box zu gelangen, klicken Sie „Arbeitsplatz/Systemsteuerung/ODBC“.

Hier möchte ich nun zwei Wege beschreiben, um dem System eine Datenquelle bekannt zu machen. Im ersten Fall entscheiden Sie sich für eine Benutzer-Datenquelle. Wenn Sie sich von Ihrem Windows32-System abmelden, kann kein anderer Benutzer der sich danach anmeldet diese Quelle ansprechen.

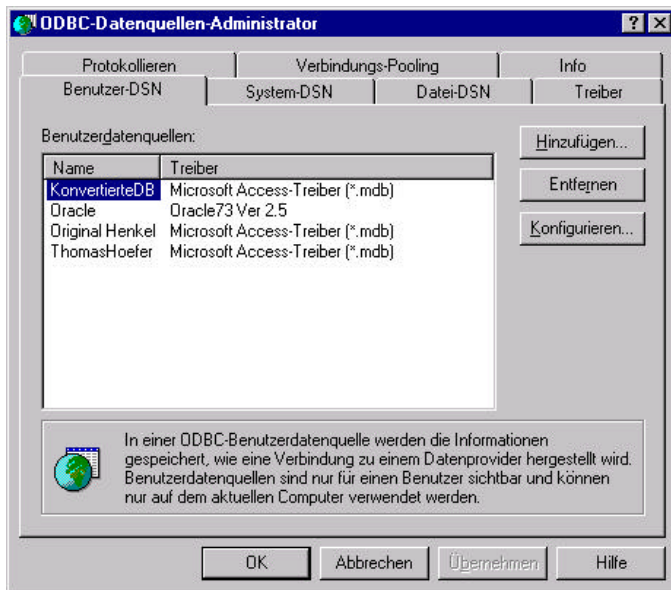


Abbildung 20: ODBC Datenquellen BenutzerDSN

Die zweite Möglichkeit ist die System-Datenquelle. Sie bietet allen Benutzern des Rechners die Möglichkeit die Datenquelle zu nutzen.

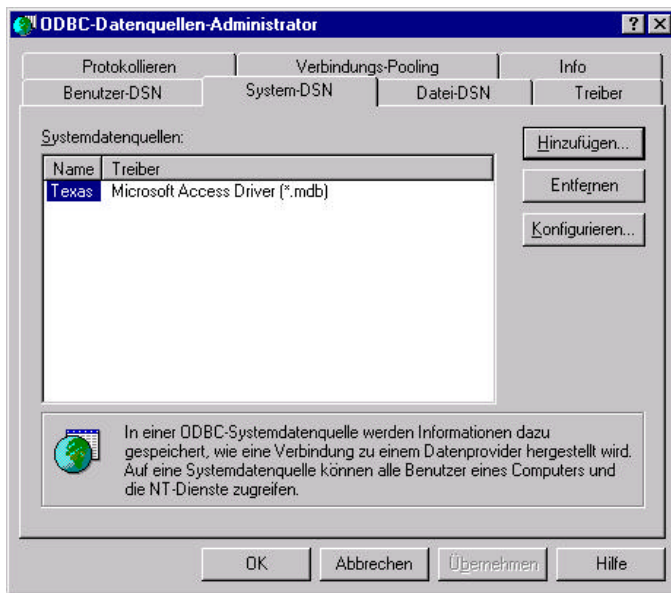


Abbildung 21: ODBC Datenquellen SystemDSN

9.2 Das Einbinden der benötigten Script-Erweiterungen

9.2.1 LSX und DLL

Für die Arbeit mit ODBC müssen in einem Notes-Programm Erweiterungen eingebunden werden, deren Zweck ich auf den folgenden Seiten erläutern möchte.

Eine Lotus Script Extension (des weiteren LSX genannt) ist im Prinzip eine DLL (Dynamic Link Library), die bestimmte Voraussetzungen erfüllt.

9.2.1.1 Was sind wiederum DLLs?

DLLs sind kompilierte Modulbibliotheken (Prozeduren und Funktionen) mit offengelegter Schnittstelle, die sich zur Laufzeit in andere Programme mit DLL-Schnittstelle einbinden lassen. In der Applikation befindet sich lediglich ein Verweis auf die entsprechende

DLL bzw. die exportierte Funktion/Prozedur.

Dynamisches Linken bietet bei den Microsoft Windows Betriebssystemen einem Prozeß die Möglichkeit, eine Funktion aufzurufen und auszuführen, die nicht Bestandteil eines ausführbaren Codes (EXE- Datei) ist.

Hierbei befindet sich der auszuführende Code der Funktion in einer dynamisch gelinkten Bibliothek. Eine DLL beinhaltet eine oder mehrere Funktionen, die übersetzt, gelinkt und separat von den sie aufrufenden Prozessen gespeichert werden.

Beispielsweise ist das Win32 API als eine Menge von DLLs implementiert, so daß jeder Prozeß, der das Win32API verwendet, dynamisches Linken benutzt.

9.2.1.2 Die Verwendung von DLLs:

Wie vorher schon erwähnt wurde, beinhaltet eine DLL eine oder mehrere Funktionen. Um diese zu verwenden, tätigt man einen Aufruf der entsprechenden Funktion. Man unterscheidet dabei zwei Verfahren, um Funktionen einer DLL aufzurufen:

9.2.1.2.1 Load-time dynamic linking

Dies geschieht, wenn eine Applikation einen expliziten Aufruf einer DLL-Funktion macht. Diese Art von Linken setzt voraus, daß das ausführbare Modul der Anwendung unter der Verwendung der DLL-Import-Bibliotheken gelinkt wird, die die Informationen enthalten, wo sich die benötigten DLL-Funktionen befinden, wenn die Anwendung gestartet wird. Dies stellt

praktisch ein Verfahren dar, das nur pseudomäßig dynamisches Linken propagiert. Trotzdem wird es in den Microsoft Manuals als dynamic linking zur Ladezeit vorgestellt.

9.2.1.2.2 Run-time dynamic linking

Dieses Verfahren wird dann angewandt, wenn die Anwendung die Funktionen LoadLibrary und GetProcAddress verwendet, um die Startadresse der benötigten DLL- Funktion(en) zu erhalten.

Mit dieser Art zu Linken braucht man nicht mehr mit Import-Bibliotheken linken, da die Adresse der benötigten DLL- Funktion(en), wie der Name bereits sagt, erst zur Laufzeit geholt wird (werden).

...nun zurück zu Lotus Notes.

Eine LSX kann in Lotus Script mit der Anweisung USELSX in eine Script-Routine eingebunden werden. Alle öffentlichen Klassen der LSX stehen anschließend der Script-Routine zur Verfügung.

Der Anweisung USELSX muß als Parameter der Name der DLL in Hochkommata mitgegeben werden.

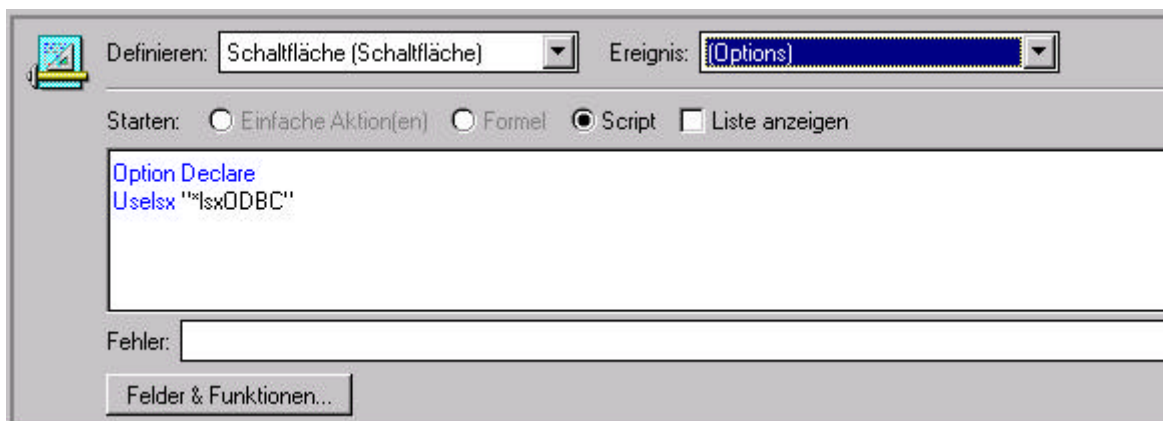


Abbildung 22: uselsx

Eine gewisse Plattformunabhängigkeit kann man erreichen, wenn die DLL's für die verschiedenen Zielsysteme realisiert sind. Dann unterscheidet sich ihr Name z.B. bei OS/2 und Windows32 nur dadurch, daß das erste Zeichen ein „I“ bei OS/2 bzw. ein „N“ bei

Windows32 ist. Ersetzt man nun das erste Zeichen mit einem „*“ sucht Lotus Script in der Registry nach der Information, wie der Name intern zu verwenden ist.

9.3 Kurzvorstellung LS:DO

Mein Tool benutzt eine ganz bestimmte LSX. Sie heißt LS:DO (Lotus Script Data Objects). Diese Lotus-Script-Erweiterung ermöglicht es, per Lotus Script auf ODBC-Datenquellen zuzugreifen.

Wie in früheren Kapiteln beschrieben, müssen die Voraussetzungen für Datenzugriffe über ODBC erfüllt sein. Danach kann man mit Hilfe der LS:DO auf ODBC-Datenquellen zugreifen.

Die LS:DO besteht aus drei Klassen.

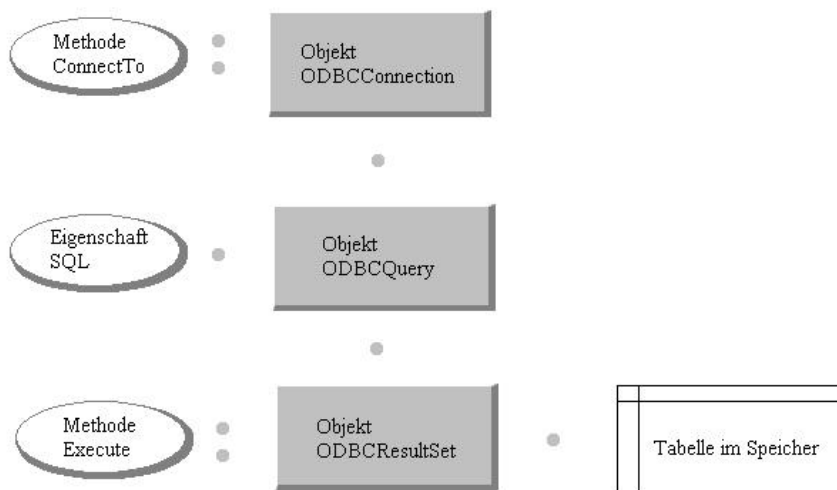


Abbildung 23: Beziehungen zwischen LS:DO-Klassen

In Instanzen des Objektes ODBCConnection werden Verbindungen verwaltet. Die Instanzen des Objektes ODBCQuery nehmen einen SQL-Abfrage-String auf und speichern die Verbindung auf die sich die Abfrage beziehen soll.

Das Arbeitspferd ist die Klasse ODBCResultSet. Ihm wird ein Query-Objekt zugewiesen, wodurch die Verbindung und die Abfrage definiert werden. Durch die Methode „Execute“ erstellt das Objekt eine Tabelle im Speicher mit den Werten die sich durch die Abfrage definieren.

Die Klasse ODBCResultSet stellt mannigfaltige Methoden zur Verfügung um Daten zu erstellen und zu manipulieren.

9.3.1.1.1 ODBC CONNECTION

Eigenschaften

AUTOCOMMIT
 COMMITONDISCONNECT
 DATASOURCENAME
 GETLSDOMASTERREVISION
 SILENTMODE

Methoden

COMMITTRANSACTIONS
 CONNECTTO
 DISCONNECT
 GETERROR
 GETERRORMESSAGE
 GETEXTENDEDERRORMESSAGE
 ISCONNECTED
 ISSUPPORTED
 LISTDATASOURCES
 LISTFIELDS
 LISTPROCEDURES
 LISTTABLES
 NEW
 ROLLBACKTRANSACTIONS
 SQLEXECDIRECT

Im Quelltext des Tools werden Sie auf die Methoden „New“, „ConnectTo“, „Disconnect“, „ListDataSources“, „ListTables“ und „ListFields“ stoßen. Durch Aufruf der Methode „New“ erhält man eine neue Instanz.

Die Methode „ConnectTo“ ordnet dieser Instanz eine bestimmte Datenquelle zu. „Disconnect“ beendet Verbindungen und gibt damit auch zum Cachen benutzter Speicher frei.

„ListDataSources“ listet alle zur Zeit möglichen Verbindungen auf. „ListTables“ listet alle Tabellen der derzeitigen Verbindung und „ListFields“ listet alle Spalten der aktuellen Tabelle auf.

9.3.1.1.1.2 ODBCQUERY

Eigenschaften

CONNECTION
 QUERYEXECUTETIMEOUT
 SQL

Methoden

GETERROR
 GETERRORMESSAGE
 GETEXTENDEDERRORMESSAGE
 NEW

Mit „New“ erhält man eine neue Instanz. Dieser kann man ein ODBCConnection-Objekt als Eigenschaft zuweisen. Ebenfalls als Eigenschaft wird das SQL-Statement zugewiesen.

9.3.1.1.1.3 ODBCRESULTSET

Eigenschaften

CACHELIMIT
 CONNECTION
 CURRENTROW
 FETCHBATCHSIZE
 MAXROWS
 QUERY
 READONLY

Methoden

ADDROW
 CLOSE
 DELETEROW
 EXECUTE
 FIELDEXPECTEDDATATYPE
 FIELDID
 FIELDINFO
 FIELDNAME
 FIELDNATIVEDATATYPE
 FIELDSIZE
 FIRSTROW
 GETERROR
 GETERRORMESSAGE

HASROWCHANGED	NUMPARAMETERS
ISBEGINOFDATA	NUMROWS
ISENDOFDATA	PREVROW
ISRESULTSETAVAILABLE	SETPARAMETER
ISVALUEALTERED	SETVALUE
ISVALUENULL	UPDATEROW
LASTROW	GETEXTENDEDERRORMESSAGE
LOCATEROW	GETPARAMETER
NEW	GETPARAMETERNAME
NEXTROW	GETROWSTATUS
NUMCOLUMNS	GETVALUE

Mit der Zuweisung der Instanz eines ODBC-Query-Objektes zu einem ODBCResultSet-Objekt werden der aktuellen Instanz die Eigenschaften „Connection“ und „Query“ gesetzt.

Die Methode „Execute“ führt die in der Eigenschaft „Query“ gesetzte SQL-Anweisung aus.

Im Speicher wird eine Tabelle angelegt. Die Anzahl der gelieferten Datensätze kann man mit der Methode „Numrows“ abfragen. Mit den Methoden „NextRow“ und „PrevRow“ kann man den Cursor auf einen Datensatz positionieren. Die Methode „GetValue“ gibt den Wert eines Feldes zurück.

Meinerseits ist geplant, die Methoden „FieldNativeDatatype“ und „FieldExpectedDatatype“ zum Anzeigen von Vorgabewerten für die Typumwandlung zu nutzen. Bislang ist das jedoch an der schlechten Dokumentation dieser Methoden gescheitert.

Auf keinen Fall sollte man den Aufruf der Methode „Close“ dieses Objektes vergessen, da daß puffern der Tabellen im Speicher sehr „ressourcenfressend“ sein kann.

9.3.2 Nachwort zum Thema LS:DO

Ab der Notes-Version 4.5 wird die Script-Erweiterung automatisch installiert. Jedoch erst ab der Version 4.54 wird eine fehlerfreie Erweiterung geboten.

Das Tool ist in der Lage die Versionsnummer abzufragen und warnt den Benutzer falls es zu Problemen durch zu alte Notes-Versionen kommen kann.

Ein bekanntes Problem ist, daß nach dem Abbrechen einer Verbindung keine neue Verbindung aufgebaut werden kann.

Durch Installation einer aktuellen Notes-Version oder dem Nachinstallieren der LS:DO die auf den Internet-Seiten der Lotus Development Corporation zum Herunterladen bereit stehen, kann man diese Probleme lösen.

Kapitel 10

HILFSTOOLS

In manchen Fällen bringen laufende Systeme recht seltsame und undurchsichtige Symptome hervor. Bis man den Fehler eingekreist hat, vergeht oft zuviel Zeit.

Folgende Abfragewerkzeuge eignen sich dazu, schon im Vorfeld festzustellen, ob eine Verbindung aufgebaut wird.

Des weiteren kann man auch sehen, ob man genügend Zugriffsrechte auf die gewünschten Ressourcen hat.

Anwendungen dieses Typus werden Ad-hoc-Abfragewerkzeuge genannt. Sie ermöglichen es dem Benutzer schrittweise Tabellen auszuwählen und dann die gewünschten Spalten.

Unter Berücksichtigung der vom Benutzer angegebenen Zusatzkriterien werden dann die benötigten Zeilen angezeigt.

Beispiele für diese Art Werkzeuge sind : Oracles 32-bit ODBC Test, Microsoft MSQuery, Intersolv Q+E, Andyne CQL, Fairfield Software Clear Access, und Brio DataPrism.

10.1 MSQuery

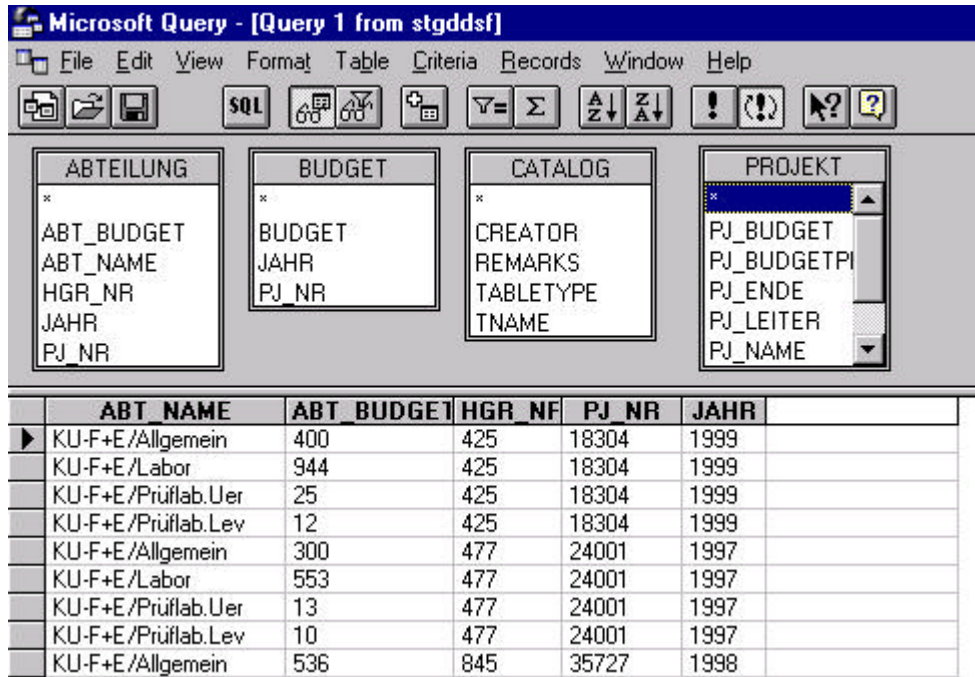


Abbildung 24: MSQuery

10.2 Oracles 32-bit ODBC TEST



The screenshot shows a window titled "Oracle ODBC 32Bit Test - [QUERY1]". The menu bar includes "File", "Edit", "View", "Window", "RowSet", and "Help". The toolbar contains icons for file operations and a help icon. Below the toolbar are buttons for "Connect..", "Disconnect", "All Tables", "User Tables", "Execute...", and "Next Row Set". The main text area contains the SQL query: "select * from Projekt". Below the text area is a table with the following data:

Row #	PJ_NR	PJ_NAME	PJ_LEITER
1	18304	Produktbegleitende F+E Desmopan	P. Müller
2	24001	Produktbegleitende F+E Dralon	T. Schneider
3	35727	Verfahren für Polyurethan	R. Radke
4	48301	Freiraumforschung Desmopan	S.Schleuter
5	59004	Produkt- und Verfahrensentwicklung	L. Waldmann

Abbildung 25: Oracle 32bit ODBCtest

Kapitel 11

DER CONET-STYLEGUIDE

Der CONET-Styleguide ist eine Leitlinie für die Entwicklung von Lotus Notes-Anwendungen bei CONET. Ich habe dabei folgende Themen zusammengefaßt:

- Kommentieren von Quelltexten
- Benennung von Variablen
- Standardverfahren für Script und Formelsprache
- Checklisten für das Layout der Anwendung

Der Styleguide möchte mit seinen Richtlinien und Regeln dazu beitragen, daß Quelltexte lesbarer und leichter verstehbar werden. Lesbare Quelltexte ermöglichen auch den sprach- und projektübergreifenden Know-how-Austausch. Der Styleguide soll für den Entwickler keine zusätzliche Belastung bei der Entwicklung seiner Programme darstellen. Die Einhaltung der Richtlinien und Regeln stellt jedoch sicher, daß ein Entwickler auch fremden Quelltext lesen und verstehen kann. Gerade für die Entwicklung im Team ist dies von entscheidender Bedeutung.

11.1 Kommentare im Quelltext

Die Programmdokumentation dient dem besseren Verstehen. Zur Programmdokumentation gehören auch Kommentare im Quelltext.

Die meisten Programme enthalten Kommentare, die unverständlich, mehrdeutig und schlichtweg falsch sind. Ein schlechter Kommentare kann schlimmer sein als gar kein Kommentar!

Wohl formulierte und gut ausgewählte Kommentare sind ein wesentlicher Bestandteil eines guten Programms. Gute Kommentare zu schreiben kann ebenso schwer sein wie gutes Programmieren. Es ist eine Kunst, die es ebenfalls verdient, kultiviert zu werden.

Jedes Modul sollte mit einem Textblock beginnen, der die nachfolgenden Entitäten und Codeteile dokumentiert.

Was kommentiert werden sollte:

- ein Kommentar zu jeder Klasse
- ein Kommentar zu jeder nicht-trivialen Funktion, der ihre Aufgabe, den verwendeten Algorithmus (falls nicht offensichtlich), und gegebenenfalls einige Annahmen über ihre Umgebung macht
- ein Kommentar zu jeder globalen Variable
- ein Kommentar zu Sonderbehandlungen die von der Regel abweichen
- ein Kommentar zu Code-Abschnitten, die schwer verständlich und/oder nicht portabel sind

11.2 Namenskonventionen in Script

Die sinnvolle Benennung von Variablen ist nicht immer leicht. Sie ermöglicht jedoch das schnelle und überschaubare Lesen des Programmtextes. Weiterhin machen „sprechende“ Variablennamen häufig zusätzliche Kommentare überflüssig.

- Jeder Variable sollte der Prefix ihres Typs vorangestellt werden.
- Variable beginnen grundsätzlich mit einem Kleinbuchstaben.

- Bei zusammengesetzten Worten beginnt jedes weitere Wort mit einem Großbuchstaben.

Die Deklaration der Script-Klassen Variablen erfolgt stets global in den jeweiligen Masken, Agenten und Script-Bibliotheken. Die Definition der Variablen erfolgt einmalig, soweit sie benötigt werden, im QueryOpen Event der Masken bzw. im Abschnitt 'Definitionen in Agenten und Script-Bibliotheken.

Um die Konsistenz der Variablen zu gewährleisten, wird in der Optionsdefinition die Funktion „Options Declare“ eingeschaltet. Die Deklaration der Variablennamen erfolgt nach folgendem Schema:

Klasse	Namenskonvention	Variablentyp	Namenskonvention
Notes ACL Entry	o[name]ACLEntry	Array<typeprefix>	arrby<typeprefix> [name]
Notes Database	o[name]DB	Boolean (über Integer)	b[name]
Notes Date Time	o[name]DateTime	Double	dbl[name]
Notes Database Directory	o[name]DBDir	Integer	i[name]
Notes Document	o[name]Doc	String	s[name]
Notes Document Collection	o[name]DocCol	File handle (über Integer)	fh[name]
Notes Item	o[name]Item	Object	o[name]
Notes Log	o[name]Log	Structure	struct[name]
Notes News Letter	o[name]Newsletter	Systemkonstante	k[name]
Notes Rich Text Item	o[name]RTItem	Variant	v[name]
Notes Session	o[name]Session	Long	lng[name]
Notes UI Document	o[name]UIDoc	Datum	dat[name]
Notes UI Workspace	o[name]UIWs		

Notes View	o[name]View	
Notes ODBC Connection	o[name]ODBCCon	
Notes ODBC Query	o[name]ODBCQuery	
Notes ODBC Result Set	o[name]ODBCRes	

Tabelle 3: Namenskonventionen

[name] ist eine freie optionale Erweiterung für die Verwendung mehrerer Instanzen einer Klasse.

Die Objekte Notessession, UiWorkspace, UiDocument und das aktuelle Notesdocument können in einer Maske global definiert werden und brauchen dann nicht immer wieder neu in jeder Funktion/Event definiert zu werden.

11.3 Die Hauptgestaltungselemente

11.3.1 Ansichten

Für Ansichtennamen sind sprechende Namen zu wählen, die über die Druckfunktion als Überschriften genutzt werden können. Für jede Ansicht muß ein Alias definiert werden. In Script und Formeln darf nur über diesen Alias auf die Ansicht zugegriffen werden.

Namenskonvention für Alias: vw<Name>.

11.3.2 Masken

Es muß immer der Alias verwendet werden. Bei Teilmasken ist als Namenskonvention der Prefix sf zu verwenden. Aliasnamen werden sprechend wie Variablennamen vergeben.

Namenskonvention für Alias: frm<Name>

11.3.3 Feldnamen

Namenskonvention für Feldnamen: fld<Name>.

11.3.4 Agenten

Namenskonventionen für Agents werden nicht festgelegt.

Namen werden sprechend wie Variablennamen vergeben.

11.4 Schriftarten

In Ansichten wird die Schriftart Arial 10 verwendet. Kategorien werden in Arial 12 blau Fett dargestellt. Für die Farben der Ansichten gilt folgende Abbildung. Für die Farbe von „anderen Zeilen“ ist 10% Schwarz zu verwenden.

11.5 Actionbuttons

11.5.1 In der Kopfzeile der Masken



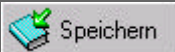

Action				
Befehl	@PostedCommand ([FileCloseWindow])	@PostedCommand ([EditDocument])	@PostedCommand ([FileSave])	@PostedCommand ([Compose]; <Maskenname>)
Verbergen	nein	Vorschau und Öffnen zum Bearbeiten	Vorschau und Öffnen zum Lesen	Nur für untergeordnete Dokumente

Tabelle 4: Actionbuttons in Masken

11.5.2 In der Kopfzeile der Ansichten




Action			
Befehl	@PostedCommand ([FileCloseWindow])	@PostedCommand ([EditDocument])	@PostedCommand ([Compose]; Maskenname>)
Verbergen	nein	Vorschau und Öffnen zum Bearbeiten	Nur für untergeordnete Dokumente

Tabelle 5: Actionbuttons in Ansichten

Kapitel 12

DAS PROGRAMM

12.1 Allgemeine Spezifikationen des Programmes

Die physikalische Datei heißt DATENPUMPE.NSF. Sie hat eine Größe von unter 800 Kilobyte und paßt damit auf eine 3.5“-Diskette.

Das Programm ist für die Notes-Revision 4.6 geschrieben worden. In der auslieferungsfertigen Version ist es plattformabhängig von WIN32, da in einigen Funktionen die Win-API aufgerufen wird. Der Zeitaufwand für eine mögliche Umstellung liegt jedoch nur im Stundenbereich.

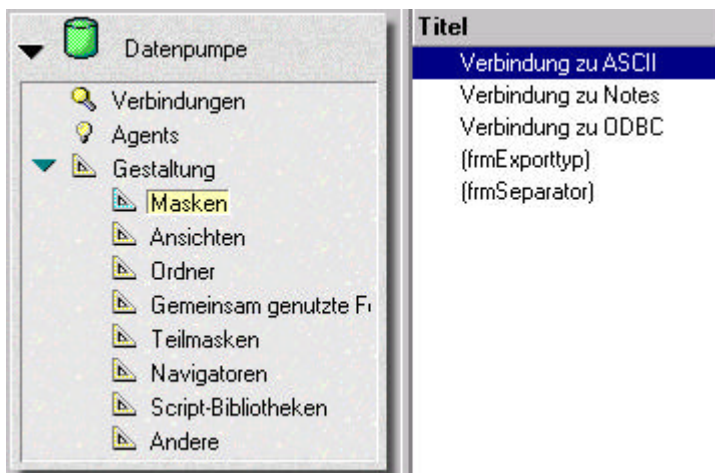


Abbildung 26: Gestaltung Masken

Abbildung 26 bietet einen Überblick über die verwendeten Gestaltungselemente. Das Programm besteht aus fünf Masken. Zwei Masken sind für den Benutzer versteckt. Man erkennt Sie an

den Klammern um ihren Namen. Die versteckten Masken kann der Benutzer nicht direkt anwählen um Dokumente zu erstellen. Ich beschreibe ihren Zweck in einem der folgenden Kapitel.

In der Datenbank gibt es sonst noch die Ansicht „Verbindungen“, die Teilmaske Historie, eine Script-Bibliothek, ein Navigator und unter dem Menüpunkt „Andere“ habe ich das Datenbank-Icon und die beiden Datenbank-Dokumente hinterlegt.

Gemeinsam benutzte Felder oder Ordner brauchte ich als Gestaltungselemente nicht.

12.2 Die Datenbankdokumente

Generiert man eine neue Notes-Datenbank, so muß man sich darum Gedanken machen, daß die Benutzer die Anwendung auch leicht verstehen.

Zum besseren Verständnis des Benutzers kann der Entwickler die Datenbank-Dokumente mit Informationen für den Benutzer füllen oder parallel eine Hilfe-Datenbank entwickeln.

Hier soll nur auf den ersteren Fall eingegangen werden. Das Dokument „Über Datenbank“, kann in den Eigenschaften der Datenbank als erstes aufzurufendes Dokument eingestellt werden.

Das Konzept ist, daß der Benutzer die wichtigsten organisatorischen Informationen zur Datenbank bekommt.

12.2.1 Das Dokument „Über Datenbank“

Name	Datenpumpe
Beschreibung	Als Quelle werden Textdateien, Notes-Dokumente oder ODBC-Quellen bearbeitet. Als Ziel dient eine von Ihnen gefertigte oder schon

	bestehende Notes-Maske mit ihren Feldern.
DB-Manager	Achim Stein
Pflege	Achim Stein
Verfügbar auf	
Eigentümer	CONET CONSULTING GmbH
Erstellt am:	07.01.1999
haltbar bis:	Unbegrenzt

Das nun folgende Dokument hat eher die Funktion einer Hilfe. Die Benutzer können sich noch weitergehend über das Programm informieren, falls Schwierigkeiten oder Fragen auftreten.

12.2.2 Das Dokument „Arbeiten mit der Datenbank“

12.2.2.1 Text-Import

Geben Sie den Verbindungsnamen an, wählen Sie die ASCII-Datei, sowie die Notes-Zieldatenbank.

Bei *Text mit fester Feldlänge* können Sie sich durch Knopfdruck eine Liste mit Vorschlägen für Ihre Feldlängen generieren lassen. Nutzen Sie dieses Feature mit dem Wissen, daß Sie meist manuell nachsteuern müssen.

Bei *separiertem Text* müssen Sie den Separator angeben. Geben Sie an, wieviele Zeilen Ihres Textes zur Beschreibung der Daten dienen.

Mit den Knöpfen + und - können Sie die Datensätze durchscrollen, verlieren aber immer die Feldlängeninformationen. Nachdem alle Einstellungen getroffen sind, stellen Sie bitte die gewählten Textspalten ihren vordefinierten Notes-Feldern gegenüber.

Dabei helfen die Knöpfe zur Spaltenauswahl. Doppelter Pfeil bedeutet "alle Spalten", einfacher Pfeil bedeutet "Übernahme der selektierten Spalten". Die Pfeile neben den Boxen sind zum Positionieren einzelner Spalten oder Felder. Das drücken der Pfeile rückt den aktuellen Wert höher oder tiefer.

Abschließend drücken Sie den Import-Knopf. Am Ende des Imports erscheint eine Statusbox

12.2.2.2 Import aus ODBC-Quellen

Es ist zu beachten, daß in Ihrer Quelle keine Tabellennamen mit Sonderzeichen oder Leerzeichen existieren dürfen!

Ist dies der Fall, machen Sie sich eine Kopie Ihrer Quelle und benennen Sie alle unbrauchbaren Titel um.

Geben Sie einen Verbindungsnamen an und bestimmen Sie die ODBC-Quelle, sowie das Notes-Ziel.

Optional können Sie Benutzernamen und Paßwort eingeben. Nun stellen Sie, wie schon im Abschnitt "Text" beschrieben die ODBC-Felder den Notes-Feldern gegenüber.

Achten Sie bitte darauf, daß Berichte und Tabellen nicht die selben Namen tragen. Das Programm kann derzeit nicht zwischen diesen Typen unterscheiden. Ihnen werden in der Folge Spalten doppelt angezeigt und das Programm kann den Import nicht leisten!

Access-Datenquellen sind bis Dato ein Problemfall, da sie bei fehlenden Benutzerrechten regelmäßig ohne weitere Meldung abstürzen.

12.3 Die Script-Bibliothek

Ab der Lotus Notes-Version 4.5 gibt die Möglichkeit, oft benutzte Funktionen in Script-Bibliotheken auszulagern.

Diese Sammlung von Lotus Script-Routinen steht dann datenbankweit zur Verfügung und kann durch „Kopieren und Einfügen“ in jede beliebige Datenbank gebracht werden.

Wenn man Script-Bibliotheken benutzen möchte, muß man sie jedoch deklarieren. Dazu wählt man den Menüpunkt „Options“ im Programmierfenster an und gibt dort USE „Name“ ein.

„Name“ steht für den Namen Ihrer Bibliothek.

Folgende Routinen habe ich für mein Tool geschrieben. Die Intention war, daß ich z.B. den Programmiereteil „ASCII-Text bis zur Zeile n vorspulen“ insgesamt an fünf verschiedenen Stellen brauchte.

Kommentare zu meinen Programmteilen beginnen, angelehnt an die Programmiersprache C mit der Zeichenfolge „/“.

```
Option Declare
```

```
//Option Declare legt fest, daß die implizite Variablendeklaration nicht erlaubt ist. Jede Variable  
//muß vor Ihrer Verwendung mit der DIM-Anweisung deklariert werden.
```

```
Declare Sub Initialize
```

```
//Es folgen Vorwärtsdeklaration der beiden Funktionen
```

```
Declare Function spulen(fHandle As Integer) As String
```

```
//Die Funktion „Spulen“ erwartet einen Dateihandle und gibt die gewünschte Zeile als  
//Zeichenkette zurück.
```

```
Declare Function ODBCstringvalidierung (sInput As String)As Integer
```

```
//Nimmt eine Zeichenkette entgegen und gibt bei ODBC-konformer Schreibweise eine -1  
//zurück.
```

```
Dim sVerboten() As String
```

```
//Zeichenarray mit nicht ODBC-konformen Zeichen. Aus programmtechnischen Gründen  
als //globales Array definiert.
```

```
Function spulen(fHandle As Integer) As String
```

```
Dim oWS As New NotesUIWorkspace
```

```
//Dieses Objekt beschreibt die aktuelle Arbeitsumgebung des Notes-Clients.
```

```
Dim oUIdoc As NotesUIDocument
```

```
//Das NotesUIDocument beschreibt das aktuell im Notes-Client geöffnete Dokument
```

```
Set oUIdoc=oWS.CurrentDocument
```

```
//Das NotesUIDocument wird instanziiert.
```

```
Dim lBeginnErstInZeile As Long, i As Long
```

```

Dim inline As String, sDatei As String
//Variablendeklarationen
BeginnErstInZeile = CInt(oUdoc.FieldGetText( "fldBeginnZeile" ))
//An dieser Stelle wird das Feld fldBeginnZeile im aktuellen Dokument ausgelesen. Zur
//Gewährleistung der Typsicherheit wird nach Long konvertiert und die Variable initialisiert.
sDatei = oUdoc.FieldGetText( "flddateipfad" )
//Der Name und Pfad der Textdatei wird in die Stringvariable sDatei gelesen.
Open sDatei For Input As fHandle
//Die Datei wird für die zeilenweise Eingabe geöffnet.
i=0
Do
    If (lBeginnErstInZeile =1)Then
        Line Input #fHandle%, inline$
        spulen=iline
        Exit Do
    Else
        If (Eof(fHandle)) Then
            MsgBox "Sie haben das Dateiende erreicht, _
                die Datei hat nur "+CStr(i)+" Zeilen !", 48 ,_
                "Dateiende erreicht!"
            Call oUdoc.Fieldclear( "fldbody")
            Call oUdoc.FieldSetText( "fldBeginnZeile", CStr(i))
            Call oUdoc.FieldAppendText("fldbody", inline )
            Call oUdoc.FieldAppendText("fldbody", Chr$(10) )
            Call oUdoc.FieldAppendText("fldbody", Chr$(10) )
            Exit Function
        End If
        Line Input #fHandle%, inline$
        i=i+1
    End If
    spulen=iline
Loop Until(i=lBeginnErstInZeile )
End Function

Function ODBCstringvalidierung (sInput As String)As Integer
//Nur eine intensive Eingangsprüfung kann gewährleisten, daß der SELECT später
//funktioniert!!

```

```

Dim i As Integer,k As Integer, iIndikator As Integer,
iSortierwechselindikator As Integer, iLaenge As Integer,
iVorgaengerpos As Integer
Dim sVerboten() As String, sVerbotenErtappt() As String,
sVerbotenChar As String, sTempstring As String, sErrorString As String
iIndikator=0

```

//ein String mit allen Zeichen, die die ODBC-Schnittstelle beim Select ablehnt:

```

sVerbotenChar="ÄäÖöÜü!§$%&/()=?*+><|_~:.,';#@+*/^°ßµ\²³{[]}~ "
Redim sVerboten(Len(sVerbotenChar)-1) As String

```

//Der String wird in ein Array zerlegt

```

For i=Lbound(sVerboten) To Ubound(sVerboten)
    sVerboten(i)=Right(Left(sVerbotenChar,i+1),1)
Next

```

//Die Werte des Arrays werden im Inputstring gesucht.

//Ein neues Array mit den gefundenen Zeichen und Ihren Stellen angelegt.

```

k=0
For i=Lbound(sVerboten) To Ubound(sVerboten)
    sTempstring = sInput
    Do While (Instr(sTempstring,sVerboten(i)))
        Redim Preserve sVerbotenErtappt(k) As String
        sVerbotenErtappt(k)=sVerboten(i)+Chr$(9)+_
        Cstr(Instr(sTempstring,sVerboten(i)))
        sTempstring=Right(sTempstring,Len(sTempstring)-
        Instr(sTempstring,sVerboten(i)))
        k=k+1
        iIndikator=1
    Loop
Next

```

//Die Fehler-Ausgabe soll eine Zeile mit den ODBC-unkonformen Zeichen enthalten.

//Die Reihenfolge soll darin stimmen und evt. vorkommende Leerzeichen genannt werden

```

If (iIndikator=1)Then
    //Hier sortiere ich das Array sVerbotenErtappt um eine schöne Ausgabe vorzubereiten
    Do
        iSortierwechselindikator=0
        k=0
        If (Ubound(sVerbotenErtappt)<>0)Then
            For i=Lbound(sVerbotenErtappt) To Ubound(sVerbotenErtappt)-1)

```



```

If (Cint(Right(sVerbotenErtappt(k),Len(sVerbotenErtappt(k))-
Instr(sVerbotenErtappt(k),Chr$(9)))) >
Cint(Right(sVerbotenErtappt(k+1),Len(sVerbotenErtappt(k+1))-
Instr(sVerbotenErtappt(k+1),Chr$(9)))))) Then
    stempstring=sVerbotenErtappt(k)
    sVerbotenErtappt(k)=sVerbotenErtappt(k+1)
    sVerbotenErtappt(k+1)=stempstring
    k=k+1
    iSortierwechselindikator=1
Else
    k=k+1
End If
Next
End If
Loop While (iSortierwechselindikator)

```

//Hier wird der Fehler-Ausgabe-Text generiert

```

iVorgaengerpos=0
For k=Lbound(sVerbotenErtappt) To Ubound(sVerbotenErtappt)

iLaenge=Cint(Right(sVerbotenErtappt(k),Len(sVerbotenErtappt(k))-
Instr(sVerbotenErtappt(k),Chr$(9))))
    For i=1 To (iLaenge-iVorgaengerpos)
        sErrorString=sErrorString+" "
    Next
    sErrorString=sErrorString+Left(sVerbotenErtappt(k),
Len(sVerbotenErtappt(k))-Instr(sVerbotenErtappt(k),Chr$(9)))
    iVorgaengerpos=iLaenge
Next
If (Instr(sInput," "))Then
    sErrorString=sErrorString+Chr(10)+Chr(10)+"Leerzeichen"
End If

```

// Jetzt wird der Fehler-Ausgabe-Text ausgegeben.

```

MessageBox sInput +" enthält verbotene Zeichen:"_
+Chr$(10)+Chr$(10)+sErrorString+Chr$(10)+Chr$(10)+"_
Ändern Sie dies in Ihrer Quelle!" , 0 + 48 , "Warnung"
ODBCstringvalidierung=1024
Else

```

```
'Sonst String OK
ODBCstringvalidierung=-1
End If
End Function
```

12.4 Datenbank-Script

In der Gestaltungsansicht gibt es unter dem letzten Punkt „Andere“ die Möglichkeit beim Eintreten von Datenbank-Ereignisse den vom Entwickler gewünschten Code laufen zu lassen.

Im Ereignis „Postopen“, welches direkt nach dem Öffnen der Datenbank behandelt wird, wird mit folgendem Code die Nummer der Notes-Version abgefragt.

```
@If (@Integer (@TextToNumber (@Version)) < 146;
  @Prompt ([OK]; "Warnung!"; "Installieren Sie Notes-Version 4.6 oder
  höher, bzw. die nötigen ODBC-Klassen!");
  "")
```

Außerdem gibt es die Ereignisse `Postdocumentdelete`, `Queryclose`, `Querydocumentdelete` und `Querydocumentundelete`.

`Postdocumentdelete` kann codiert werden, um nach dem Löschen von Dokumenten Protokollfunktionen aufzurufen.

`Queryclose` wird vor dem Schließen der Datenbank aufgerufen und kann im Hinblick auf gesetzte Umgebungsvariablen für Aufräumarbeiten eingesetzt werden.

Mit `Querydocumentdelete` kann man das Setzen der Löschmarkierung überwachen und `Querydocumentundelete` stellt dazu das Gegenstück dar.

12.5 Die Agenten

Agenten werden in der Makrosprache durch den folgenden Befehl aufgerufen:

```
@Command ( [ToolsRunMacro] ; "Agentenname" )
```

In meiner Anwendung existieren zwei Agenten, auf die ich jedoch nicht tiefer eingehen möchte, da sie zum ASCII-Import gehören.

Allgemein gesagt kann man Agenten ereignisgesteuert, zeitgesteuert, manuell oder durch den Aufruf aus einer Routine triggern.

12.6 Die Maske „Verbindung zu ASCII“

12.6.1 Abbildung der Programmlogik

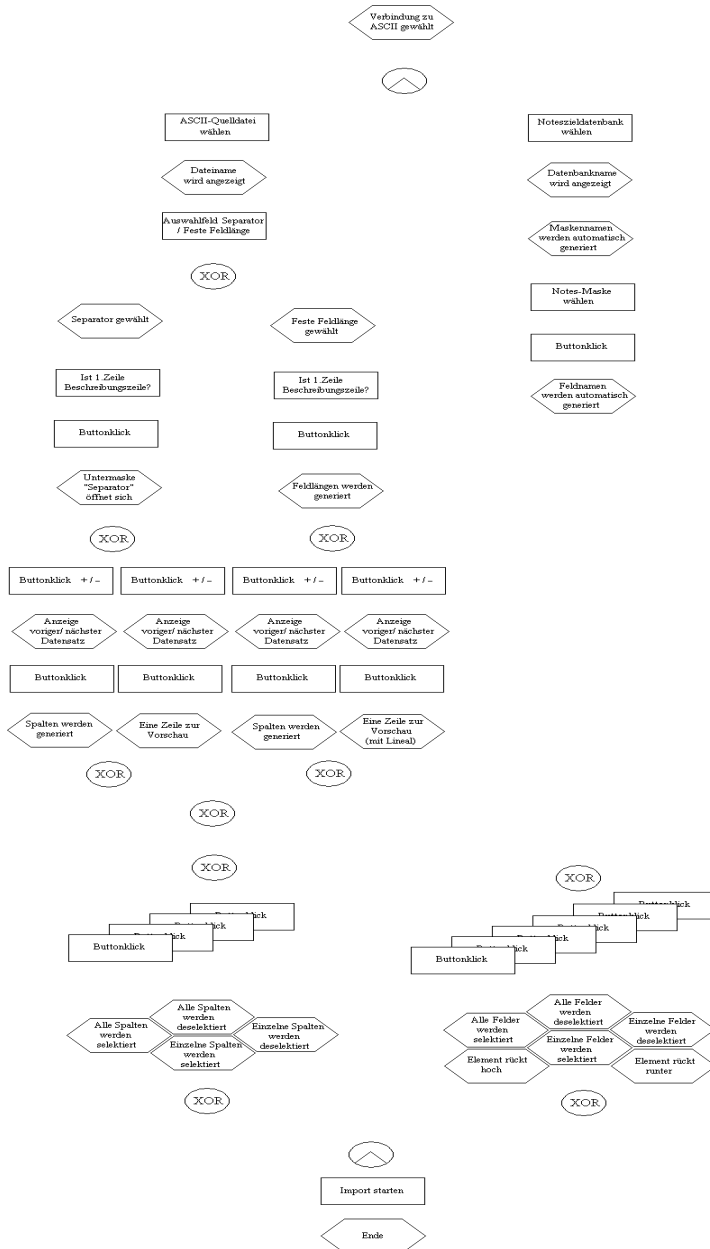


Abbildung 27: Programmablauf ASCII

12.6.2 Benutzerschnittstelle der Maske „Verbindung zu ASCII“

Abbildung 28: Benutzerschnittstelle Import von ASCII



-> Subform <-

▶ Versteckte Felder

12.7 Die Maske „Verbindung zu Notes“

12.7.1 Abbildung der Programmlogik

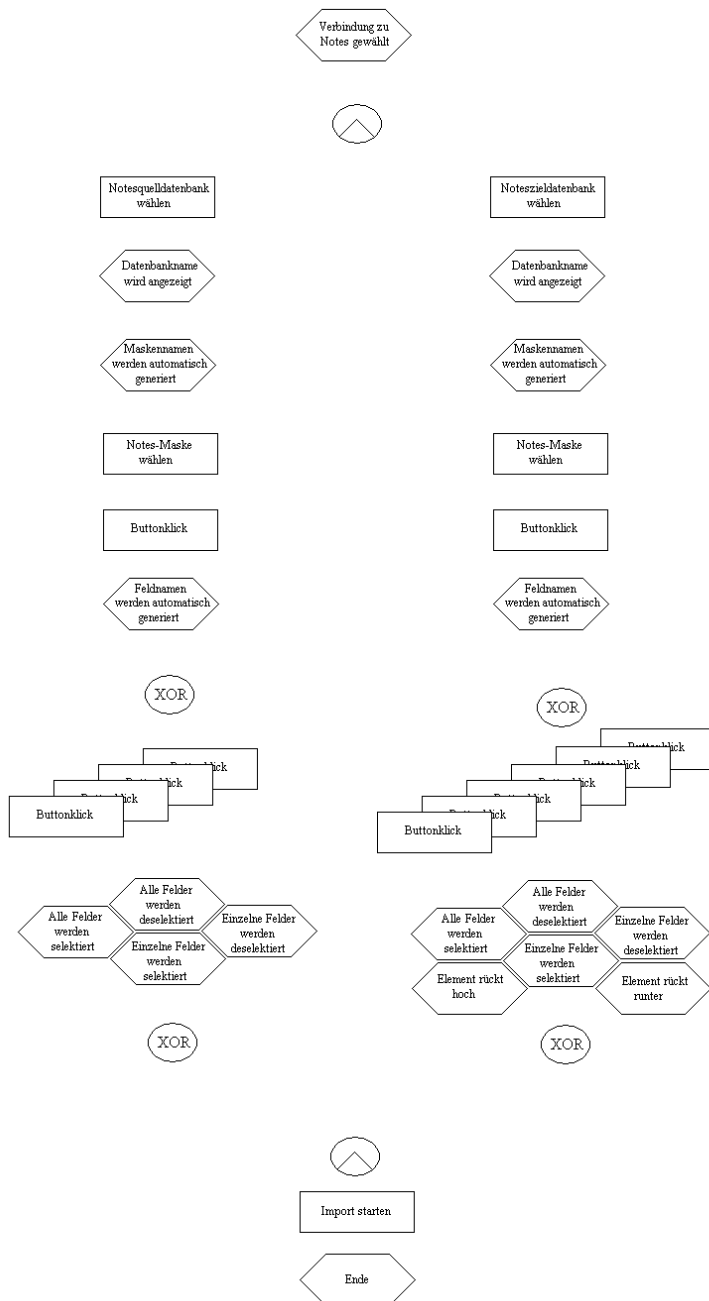


Abbildung 29: Programmablauf Notes

12.7.2 Benutzerschnittstelle der Maske „Verbindung zu Notes“

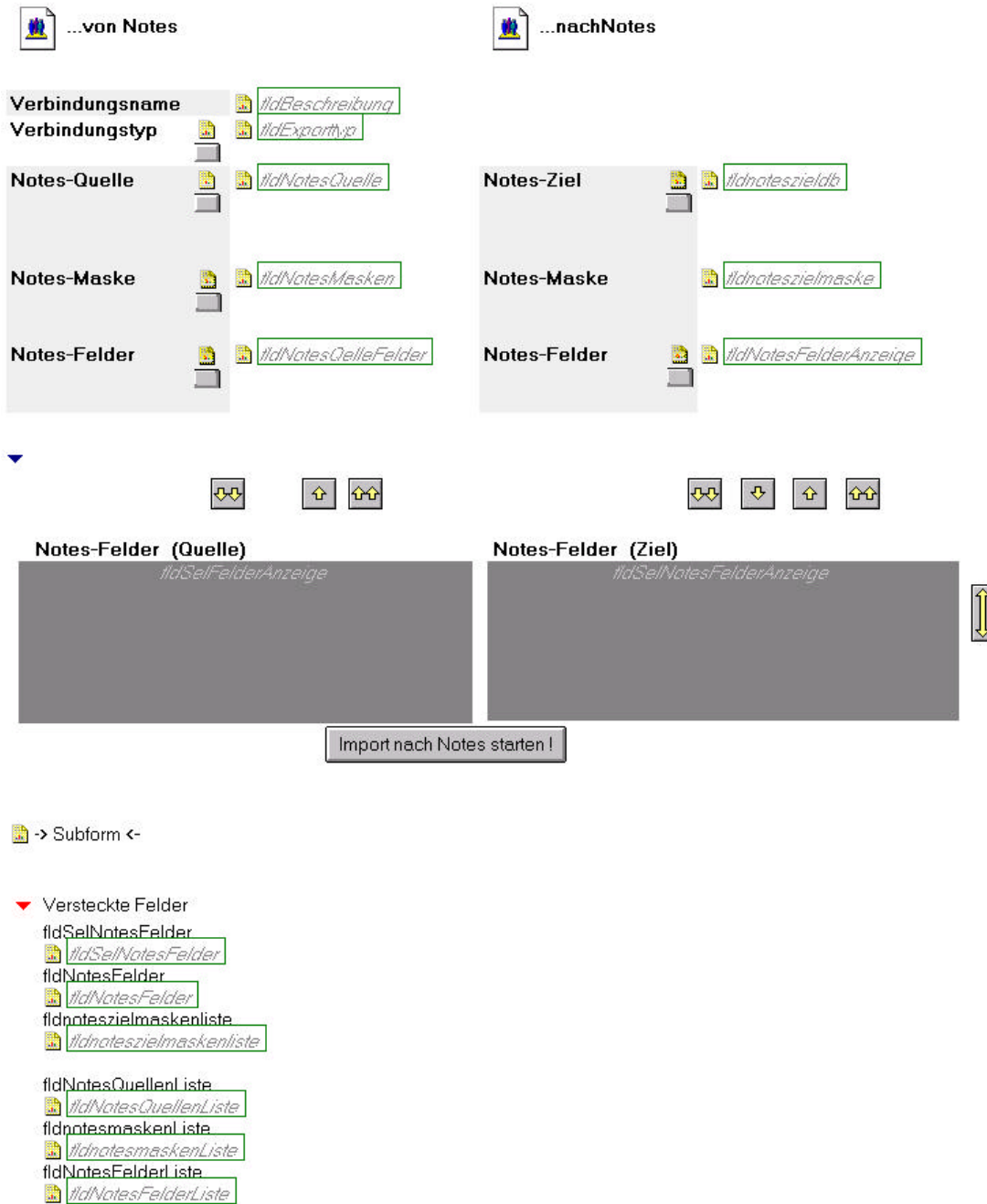


Abbildung 30: Benutzerschnittstelle Import aus Notes

12.8 Die Maske „Verbindung zu ODBC“

12.8.1 Abbildung der Programmlogik

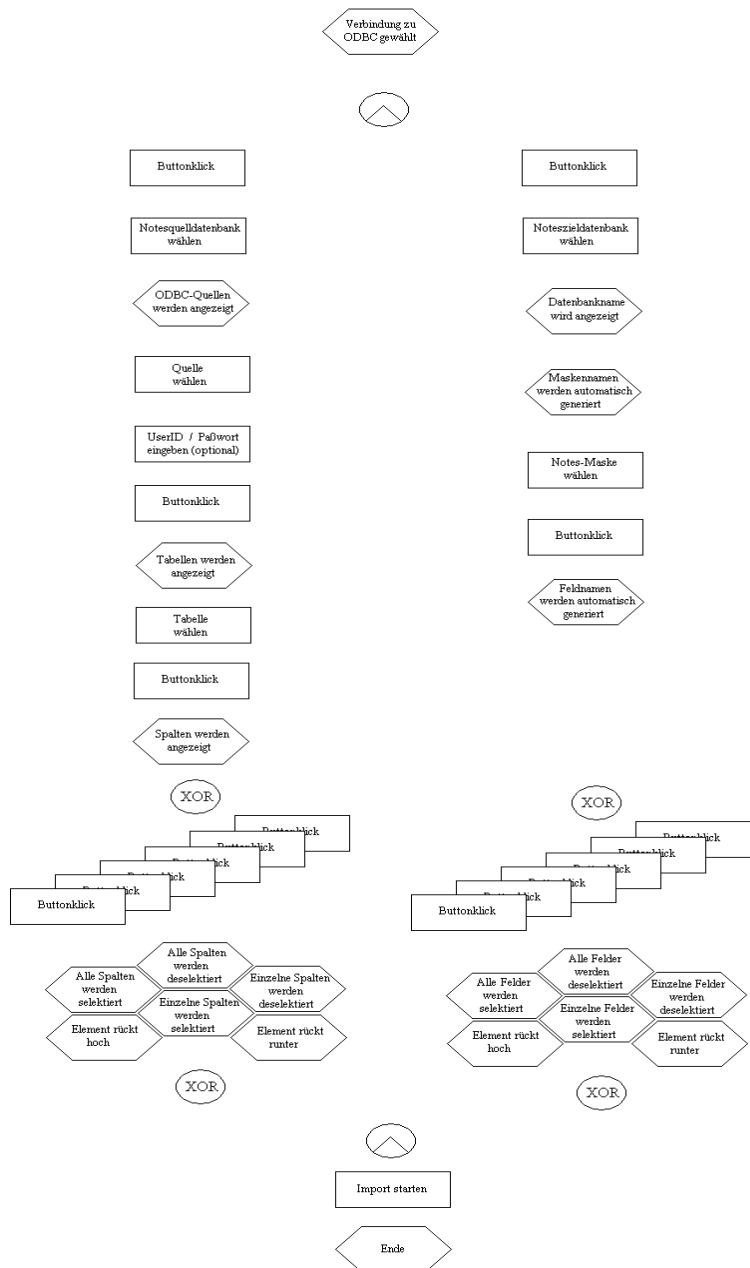


Abbildung 31: Programmablauf ODBC

12.8.2 Benutzerschnittstelle der Maske „Verbindung zu ODBC“

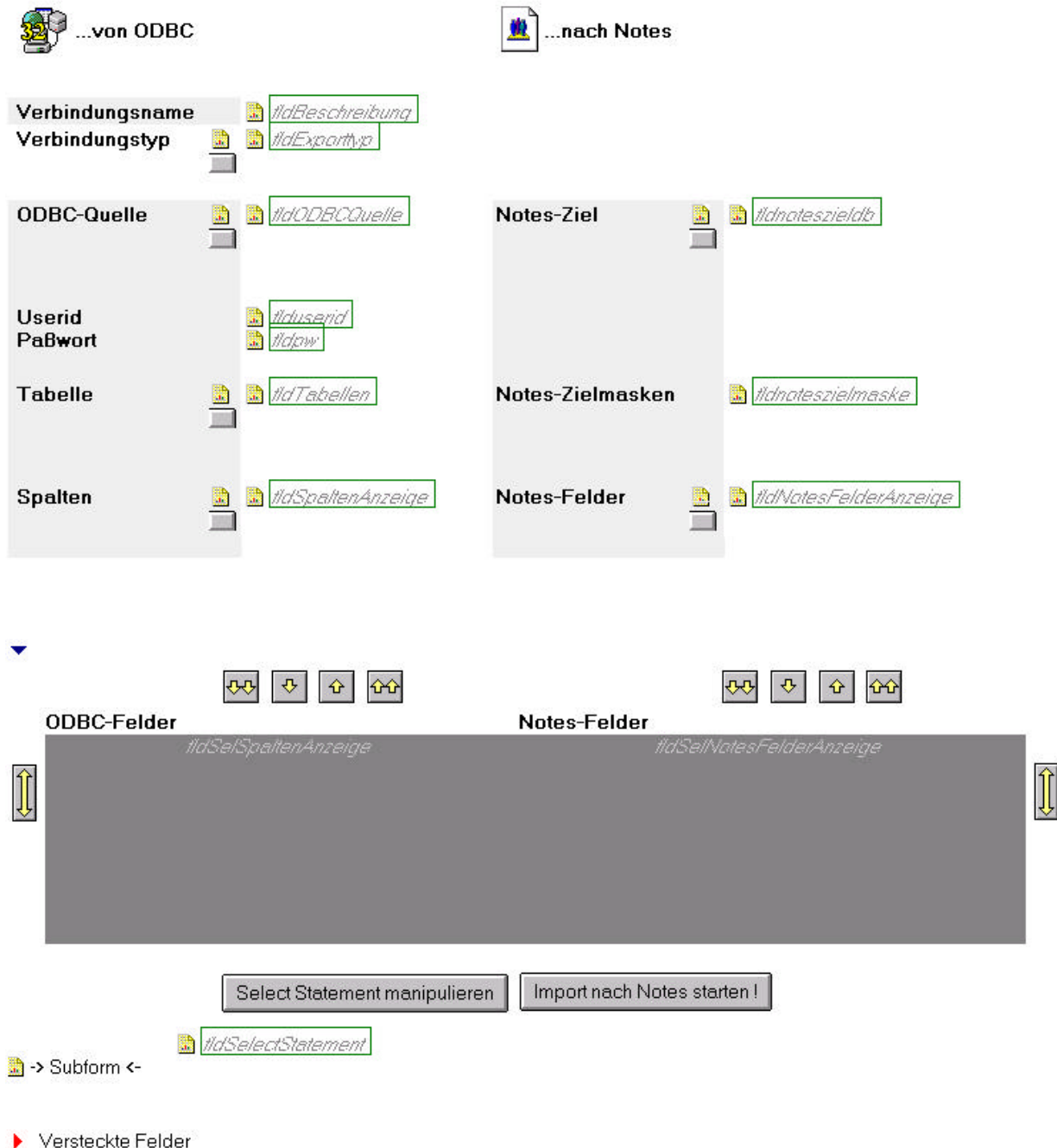


Abbildung 32: Benutzerschnittstelle Import von ODBC

12.8.3 Beispiele für die Programmierung der Maske „Verbindung zu ODBC“

Der Maskenalias, der bei der Programmierung verwendet wird, lautet „frmODBC“

Beim Erstellen eines Dokumentes wird zunächst einmal die Titelformel ausgewertet. Es erscheint in der Titelzeile "Neue Verbindung". Beim Öffnen eines bestehenden Dokumentes erscheint der Text des Feldes „fldBeschreibung“.

```
@If(
  @IsNewDoc; "Neue Verbindung";
  @If((fldBeschreibung)!=" " ; fldBeschreibung ;
    "Ohne Titel"
  )
)
```

Folgende Felder wurden erstellt:

- fldBeschreibung , bearbeitbar, speichert die Verbindungsbeschreibung des Benutzers
- fldODBCQuelle, bearbeitbares Anzeigefeld zu fldODBCQuellenListe
- flduserid, bearbeitbares Textfeld für die optionale Angabe des Benutzernamens
- fldpw, bearbeitbares Textfeld für die optionale Angabe des Benutzerpaßwortes
- fldTabellen, bearbeitbares Anzeigefeld zu fldTabellenListe
- fldSpaltenAnzeige, bearbeitbares Anzeigefeld zu fldSpalten
- fldSelSpaltenAnzeige, bearbeitbares Anzeigefeld zu fldSelSpalten in einer Layoutregion
- fldnoteszieltb, bearbeitbares Anzeigefeld
- fldnoteszielmaske, bearbeitbares Anzeigefeld zu fldnoteszielmaskeliste
- fldNotesFelderAnzeige, bearbeitbares Anzeigefeld zu fldNotesFelder

- fldSelNotesFelderAnzeige, bearbeitbares Anzeigefeld zu fldSelNotesFelder in einer Layoutregion

Versteckte Felder :

- fldSelSpalten, verstecktes, berechnetes Listenfeld für die vom Benutzer ausgewählten Spalten
- fldSpalten, verstecktes, berechnetes Listenfeld für die vom Programm erzeugten Spaltennamen
- fldTabellenListe, verstecktes, berechnetes Listenfeld für die vom Programm erzeugten Tabellennamen
- fldOdbcQuellenListe, verstecktes, berechnetes Listenfeld für die vom Programm erzeugten ODBCQuellennamen
- fldNoteszielmaskenliste, verstecktes, berechnetes Listenfeld für die vom Programm erzeugten Notesmaskennamen
- fldNotesFelder, verstecktes, berechnetes Listenfeld für die vom Programm erzeugten Notesfeldnamen
- fldSelNotesFelder, verstecktes, berechnetes Listenfeld zur Darstellung der Notesfelder als Auswahl in fldSelNotesFelderAnzeige
- fldSelectStatement, experimentelles Feld zur späteren Verwendung
- fldExporttyp, für zukünftige Erweiterung reserviert
- fldEditor, geerbtes Feld aus der Teilmaske „Historie“ alias frmHistory
- fldAenderdatum, geerbtes Feld aus der Teilmaske „Historie“ alias frmHistory
- fldAutor, geerbtes Feld aus der Teilmaske „Historie“ alias frmHistory
- fldErstelldatum, geerbtes Feld aus der Teilmaske „Historie“ alias frmHistory

12.8.3.1 Lotus Script in den Masken-Events:

Das Postopen-Ereignis mußte ich benutzen, da die Dokumente beim Öffnen immer an den unmöglichsten Stellen angesprungen wurden, meist sehr weit unten. Das empfand ich als schlecht und habe durch folgenden Code auf das Feld "fldBeschreibung" zentrieren lassen:

```
Sub Postopen(Source As Notesuidocument)
  If source.EditMode Then
    source.gotofield("fldBeschreibung")
  End If
End Sub

Sub Postmodechange(Source As Notesuidocument)
  If source.EditMode Then
    source.gotofield("fldBeschreibung")
  End If
End Sub
```

12.9 Die versteckten Masken

Die Maske "(frmExporttyp)" ist für die zukünftige Erweiterung des Programms gedacht. Sie bietet in Kombination mit der Makrosprachen-Formel ...

```
@DialogBox("(frmExporttyp)" ; [AutoHorzFit] : [AutoVertFit] ; "Auswahl
Dialog Exporttyp");
```

folgende Auswahlbox:

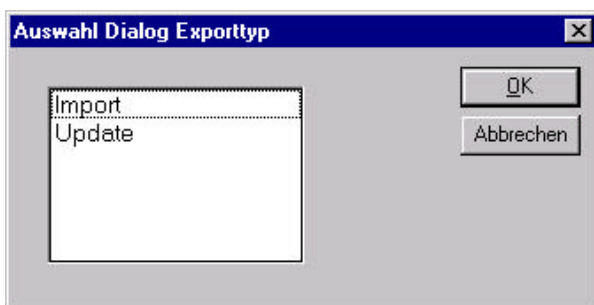


Abbildung 33: Auswahldialog Exporttyp

Als Voraussetzung für diese Art der Dialogprogrammierung müssen Felder mit dem selben Namen in der versteckten Hilfsmaske, sowie in der Hauptmaske vorkommen. Mit der obigen

Notation nimmt die Funktion @Dialogbox aus der Hilfsmaske das Schlüsselwortfeld „fldExporttyp“ und paßt den Inhalt der Box auf die Größe des Feldes in der versteckten Maske an.

12.9.1 Die Maske “(frmSeparator)”

Diese Maske funktioniert nach dem zuvor beschriebenen Grundsatz. In der Abbildung sehen sie die Eigenschaften des gleichnamigen Feldes in der versteckten Hilfsmaske.



Abbildung 34: Eigenschaften des Felds fldSeparator

12.10 Die Teilmaske „Historie“

Die Teilmaske „Historie“ trägt den Maskenalias „frmHistory“. Dokumente die durch die drei Hauptmasken angelegt werden, beinhalten ebenfalls die Felder der Teilmaske.

Ihr Zweck liegt darin, Benutzeraktivitäten zu protokollieren

Felder:

- fldEditor, berechnet, hält den letzten Bearbeiter fest.
- fldAenderdatum, berechnet
- fldAutor, berechnet beim Anlegen, hält den Ersteller des Dokumentes fest.
- fldErstelldatum berechnet beim Anlegen

Die Programmierung dazu geschieht durch Lotus Script in den Masken-Events:

```
Sub Querysave(Source As Notesuidocument, Continue As Variant)
```

```
//Ereignis vor dem Speichern, Continue kann man auf True oder False setzen, das Verhindern  
das //Speichern oder läßt es zu.
```

```
rc=Evaluate(|@Name([CN];@UserName)|)
```

```
//Der Variabel rc (steht für Rückgabewert, returncode) wird Name des Bearbeiters in  
kanonischer //Form zugewiesen.
```

```
Call source.Fieldsettext("fldEditor", Cstr(rc(0)))
```

```
//Das Feld "fldEditor" kriegt jetzt den Wert zugewiesen.
```

```
rc=Evaluate(|@Now|)
```

```
//Der Variablen wird nun der aktuelle Zeit-Wert zugewiesen.
```

```
Call source. Fieldsettext ("fldAenderdatum", Cstr(rc(0)))
```

```
//...und als Letztes in das Feld "fldAenderdatum" der Wert abgelegt.
```

```
End Sub
```

12.11 Die Ansicht „Verbindungen“

Die Aufgabe dieser Ansicht ist es, die Verbindungsdokumente übersichtlich darzustellen.

Die Ansicht ist in den meisten Projekten eine der leichtesten und kürzesten Arbeiten. So auch hier.

In der Abbildung zur Ansicht erkennt man die eine von zwei Spaltenformeln. Nach Erstellmaske wird in fetter, blauer Schriftart die passende Kategorienüberschrift erzeugt.

Das Kategorisieren ist eine Eigenschaft der jeweiligen Spalte.

Die zweite Spaltenformel wertet ein Feld mit der Beschreibung der Verbindung aus und stellt das Ergebnis mit schwarzer Schrift dar.

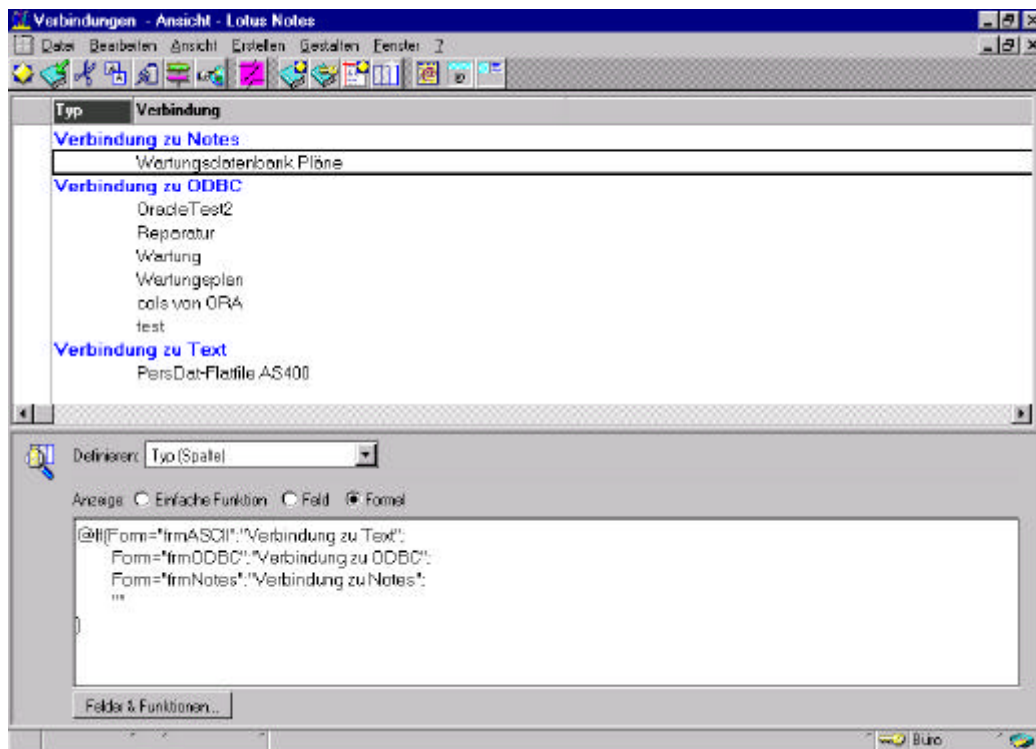


Abbildung 35: Spaltenformel in der Ansicht

Der programmiertechnisch verwendete Aliasname der Ansicht lautet vwconn. Das „vw“ steht dabei für View und „conn“ für das englische „Connection“ für „Verbindung“.

Die Ansicht selektiert alle Dokumente der Datenbank und wird direkt aufgeklappt.

12.12 Der Navigator

Der Navigator besteht aus drei Kacheln hinter denen der Aufruf zum Erstellen eines neuen Dokumentes liegt. Der Aufruf lautet :@Command([Compose]; "Maskenaliasname")

Das ist nicht aufregend. Worauf ich ein wenig stolz bin, ist ein kleiner Trick mit dem ich Lotus Notes überlistet habe.

Lotus Notes kennt bis zur Revision 4.6 noch kein Schaltflächen-Ereignis „OnMouseOver“. Das ist eigentlich schade, da man somit kaum ansprechende Oberflächen gestalten kann.

Mein „Workaround“ bestand darin, eine Kachel mit Schatten zu produzieren und ein passendes Viereck. Das Viereck bekam dann die Farbe „transparent“ und die Eigenschaften konnte ich so setzen, daß beim Berühren des Vierecks die Hintergrundfarbe erscheint.

Diese Kombination ergibt das Erscheinungsbild eines gedrückten Knopfes wenn darüber gefahren wird. Bitte achten Sie auf den Schatten:

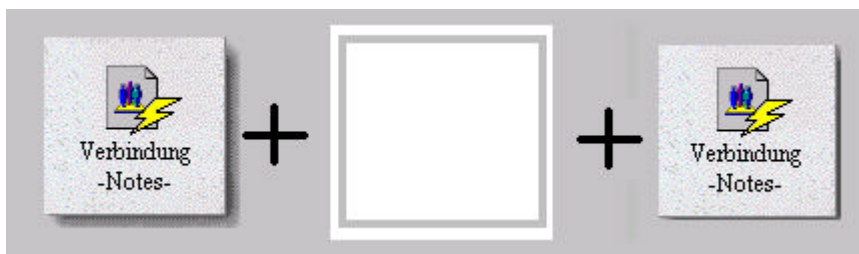


Abbildung 36: Navigator

Die Datenbank mit einem Navigator zu starten ist eine Eigenschaft der Datenbank, die man unter dem Reiter „Starten“ setzen kann.

Um den Navigator schick aussehen zu lassen, kann man dann noch in den Eigenschaften des Navigators die Option „Beim Starten Fenster automatisch anpassen“ ankreuzen. Ebenfalls in den Eigenschaften des Navigators weist man die Startansicht zu. In diesem Fall ist das die Ansicht „Verbindungen“.

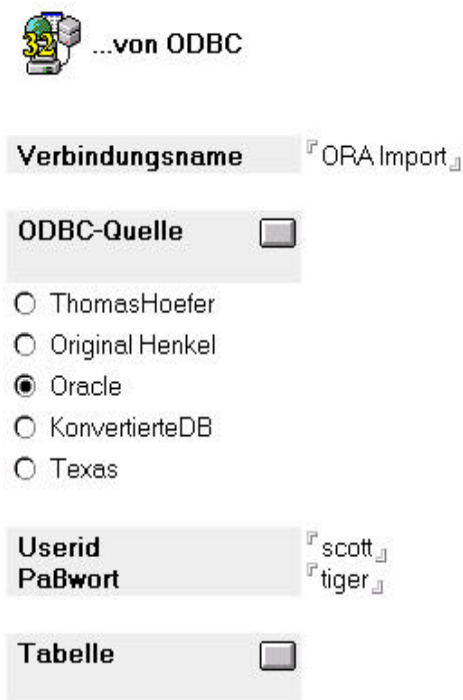
Kapitel 13

EIN PROBELAUF DER DATENPUMPE

Voraussetzungen : Die Datenpumpe muß installiert sein.

Die gewünschte ODBC-Quelle ist eingerichtet.

Die Notes-Zieldatenbank existiert.



Verbindungsname ORA Import

ODBC-Quelle

ThomasHoefler

Original Henkel

Oracle

KonvertierteDB

Texas

Userid scott

Paßwort tiger

Tabelle

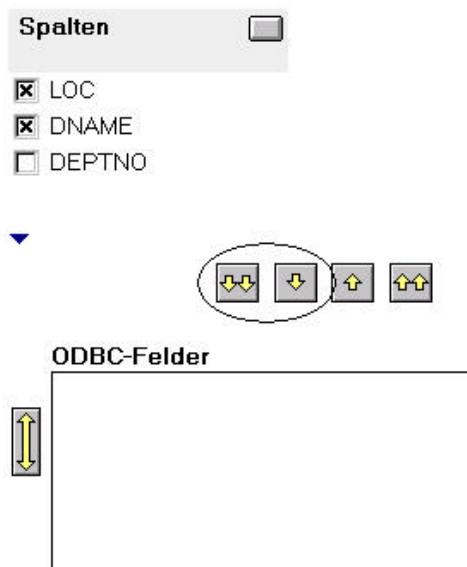
Im ersten Schritt erstellen wir ein neues Verbindungsdokument für den ODBC-Import.

Klicken Sie im Navigator auf die Kachel „Verbindung ODBC“.

Geben Sie den Namen der Verbindung an und klicken Sie auf den Knopf neben dem statischen Text „ODBC-Quelle“. Je nach Konfiguration erscheinen nun im Auswahlfeld mehr oder weniger viele Quellen.

Wählen Sie eine aus und geben Sie, falls erforderlich Benutzernamen und Paßwort ein.

Drücken Sie jetzt auf den Knopf neben dem statischen Text „Tabelle“. Nun erscheinen die Tabellen die Sie mit Ihrer Benutzerkennung einsehen dürfen im Auswahlfeld.



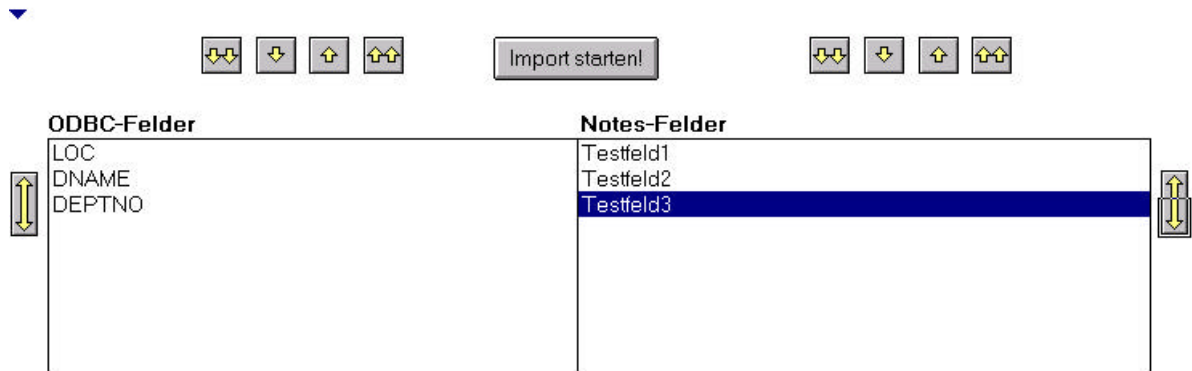
Nach der Auswahl der Tabelle drücken Sie jetzt auf den Knopf neben dem statischen Text „Spalten“. Markieren Sie die Spalten, deren Werte Sie importieren möchten. Mit der Doppel-Pfeiltaste können entweder alle Spalten sofort in die Mapping-Box übernommen werden; mit der einfachen Pfeiltaste gilt das nur für Ihre definierte Auswahl.

Im zweiten Schritt geht es um die Auswahl des Datenzieles. Klicken Sie auf den Knopf neben dem statischen Text „Notes-Ziel“. Es erscheint folgende Dialogbox:



Wählen Sie Ihre Notes-Datenbank. Es erscheint sofort eine Auswahl der Masken dieser Datenbank. Markieren Sie dort Ihre Maske und klicken Sie auf den Knopf neben dem statischen Text „NotesFelder“. Die Auswahl der Felder geschieht auf dem gleichem Wege wie die schon beschriebene Auswahl der Spalten.

Der letzte Schritt ist das „Mapping“. Das ist die Zuordnung der Spalten der Oracletabelle zu den passenden Notes-Feldern. Dazu dienen Ihnen die Navigationsknöpfe am rechten und linken Rand der Mapping-Box.



Sichern Sie Ihre Arbeit und klicken Sie auf „Import starten!“. Der Arbeitsfortschritt des Imports wird Ihnen durch einen Statusbalken und das Endergebnis durch eine Meldungbox angezeigt.

Kapitel 14

TROUBLESHOOTING UND ERKANNTEN FÜßANGELN

14.1 ODBC: Probleme beim Ausführen einer Abfrage verursacht durch Sonderzeichen

Wenn man eine Select-Anweisung ausführen möchte in der die Tabellennamen oder die Spaltennamen eines dieser Zeichen enthält, stürzte das Programm ab:

„ÄäÖöÜü!\$\$%&/()=?*+><|_~:.,';#@+*/^°βμ\²³{[]}~“

Das lag nicht am Programm selbst. Ich führe es auf Microsofts ODBC-Treiber zurück. Als Workaround wies ich den Benutzer durch das Programm auf derartige Zeichen hin. Meine Idee war zunächst, daß der Benutzer die unpäßlichen Namen bearbeitet.

Der Zustand ärgerte mich lange Zeit und so lasse ich es in der letzten Programmversion zu, daß der Benutzer einen alternativen Weg gehen kann. Das Dilemma besteht nur noch im Hinblick auf die Tabellennamen. Alle anderen Namen lasse ich programmintern durch Ziffern ansteuern.

14.2 ODBC: Benutzerrechte am Beispiel Access

Access hat eine seltsame Handhabung im Hinblick auf Benutzerrechte. Bei ungenügenden Zugriffsrechten auf seiten von Access, stürzt jeglicher ODBC-Zugriff ohne weitere Fehlermeldung ab.

14.3 Limits von Lotus Notes

Problematisch sind riesige Feldinhalte unter Notes. Bei der Darstellung von fünf Zeilen Text einer SAP-Auswertung (siehe Seite 37) war das darstellende Textfeld nicht in der Lage, mehr als 64 kB an Information zu fassen. Mein Workaround bestand darin, daß ich nur eine Zeile anzeigen ließ, aber dafür zeilenweises Scrollen durch Navigationsknöpfe zuließ.

Kapitel 15

AUSBLICK AUF ZUKÜNFTIGE LEISTUNGSERWEITERUNGEN

15.1 ODBC-Abfrage über mehrere Tabellen

Wünschenswert ist meiner Meinung nach, daß man verschiedene Tabellen als Quellen angeben kann. Diese Funktionalität erfordert eine höhere Komplexitätsstufe der Quellnamensverwaltung. Ebenfalls muß noch analysiert werden, wie das Verbindungshandling effizient gehandhabt werden kann.

15.2 ODBC-Abfrage mittels einer vom Benutzer definierten Select-Anweisung

Dies ist die naheliegendste Verfeinerung. Die Abfrage des Benutzers übersteuert in diesem Falle den berechneten Abfragestring des Programms. Der Aufwand um die Benutzereingabe zu debuggen steht in keinem Verhältnis zur ursprünglichen Programmplanung und wird von mir nicht implementiert. Auch sehe die Gefahr, daß die Benutzer Abfragen ausführen, die zu

Programmabstürzen führen. Dieses zusätzliche Mittel werde ich deshalb in einer „erweiterten Programmversion“ nur erfahrenen Benutzern an die Hand geben.

15.3 Datentypen übernehmen

Zur Zeit wird jeder Datentyp zu einem String konvertiert. Es gibt eine Möglichkeit Notes-Datentypen in SQL-Datentypen umzuwandeln. Die automatische Konvertierung ist das drängendste Problem, dem ich in naher Zeit nachgehen muß. Ich plane dem Benutzer die möglichen Datentyp-konvertierungen als Vorschlag im einem separaten Fenster darzustellen.

15.4 Update-Funktionalität

Daten sollen in Zukunft nicht nur einfach importiert werden. Für den Algorithmus, der hinter einer Update-Funktionalität steckt, muß ich mir überlegen, ob ich jedes einzelne Feld von den Inhalten her vergleiche, oder in den Quell-/ und Zieldokumenten ein Feld einführe, in dem Bearbeitungs-Flags gesetzt werden. Ebenfalls ist ein Zeitvergleichsalgorithmus sinnvoll, der dafür sorgt, daß nur veränderte Datensätze und Dokumente weiter berücksichtigt werden.

15.5 Zeitsteuerung

Geplant ist auch das künftige Betreiben des Programmes auf einem Server, der in regelmäßigen Abständen Daten übernimmt. Die Realisierung steht aber gewiß hinter der Implementierung der Update-Funktionalität zurück, da sich bei einer ständigen Komplettübernahme die Server-Performance untragbar verschlechtern würde.

Für die Implementierung ist ein völliger Code-Review nötig, da Server-Programme auf keinen Fall UI-Klassen beinhalten dürfen. UI-Klassen sind die bei der Programmierung der Benutzeroberfläche verwendeten Klassen. Sie werden u.a. benötigt um Ergebnisse von Berechnungen direkt anzuzeigen.

Anhang A:

LITERATURLISTE

Enterprise Integration with Domino.Connect (IBM Redbook)

Lotus Notes Release 4.5: A Developer's Handbook (IBM Redbook)

Lotus Solutions for The Enterprise, Volume 1 Lotus Notes: An Enterprise Application Platform (IBM Redbook)

Lotus Solutions for the Enterprise, Volume 5 NotesPump: The Enterprise Data Mover (IBM Redbook)

DECS (Lotus Handbuch)

Developing Lotus Applications with Components (Lotus Handbuch)

Domino Install Guide R 4.6 (Lotus Handbuch)

Enterprise Integration (Lotus Handbuch)

Data Sources Connectivity Guide

Programmers Guide 4.6 Buch: a (Lotus Handbuch)

Erster Teil des Lotus Notes Programmierhandbuch (Script)

Programmers Guide 4.6 Buch: b (Lotus Handbuch)

Zweiter Teil des original Lotus Programmierhandbuchs (Formelsprache)

ARIS-Methodenhandbuch, Buch 5 der Handbücher zum ARIS-Toolset

Inside ODBC von Kyle Geiger, Microsoft Press 1996, ISBN 3-86063-359-7

Anhang B:

ABKÜRZUNGSVERZEICHNIS

API	Application Programming Interface, Anwendungsschnittstelle
BLOB	Binary Large Object
CLI	Call Level Interface
DECS	Domino Enterprise Connection Server
DLL	Dynamic Link Library
EPK	Ereignisgesteuerte Prozeßkette
(G)UI	(Graphical) User Interface
ISAM	Index Sequential Access Methods
LS:DO	Lotus Script Data Objects
LSX	Lotus Script Extensions
ODBC	Open Database Connectivity
(R)DBMS	(Relational) Database Management System, Datenbank-Management-System
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung
SQL	Structured Query Language

Anhang C:

GLOSSAR

API	Eine Menge verwandter Funktionen, die ein Programmierer einsetzt, um dadurch einen bestimmten Service von einer anderen Software zu erhalten.
CLI	Definiert eine Menge von Funktionsaufrufen (siehe API). CLI kommt aus dem SQL-Umfeld und beschreibt eine Schnittstelle, bei der es sich nicht um eingebettetes SQL handelt.
Data Warehouse	Ein Data Warehouse ist eine fachbezogene, integrierte Datensammlung mit zeitlichem Bezug für Analysen zur Entscheidungsunterstützung und liefert dabei reproduzierbare Ergebnisse. Es ist eine vom produktiven System separierte Datenbank und speziell zur Datenanalyse gedacht.
DBMS	Software, die den Zugriff auf strukturierte Daten realisiert.
Embedded SQL, eingebettetes SQL	ist eine alternative Programmierschnittstelle bei der SQL-Anweisungen mit normaler Programmsyntax kombiniert und durch einen Precompiler in Funktionsaufrufe für die DBMS-Laufzeitbibliothek übersetzt werden.
EPK	Nach IDS Professor Scheer, beginnt eine EPK mit einem Startereignis und endet mit einem definierten Ende-Ereignis. Ereignisse sind Auslöser von Funktionen und Ergebnisse von Funktionen. Die Ablauffolge von Funktionen wird in Prozeßketten dargestellt. Von einem Ereignis können mehrere Funktionen

	ausgehen, andererseits kann eine Funktion mehrere Ereignisse als Ergebnis haben. Diese Verzweigungen und Bearbeitungsschleifen werden grafisch durch sog. Verknüpfungsoperatoren dargestellt (AND - ^, OR, XOR).
Nebenläufigkeit	<p>(engl.:concurrency) Heißt soviel wie: „Parallelität“ und ist der Aspekt eines Systems, der besagt, daß mehrere Subjekte des Systems gleichzeitig aktiv sind.</p> <p>Bemerkung: Um aktiv zu sein, benötigt ein Subjekt eine Ausführungseinheit (z.B. eine Person oder einen Prozessor eines IT-Systems). Werden mehr gleichzeitig aktive Subjekte benötigt als Ausführungseinheiten verfügbar sind, kann Nebenläufigkeit durch Time-sharing-Verfahren simuliert werden.</p>
Sperrprimitive	Ein Befehl zum Setzen oder Aufheben einer Sperre für eine Zeile in einer ISAM-Datei oder der ganzen ISAM-Datei. Sperren verhindern entweder das Lesen oder das Schreiben der Zeilen.
Transaktion	<p>Eine Transaktion ist eine abgeschlossene Arbeitseinheit, in der einzelne Bearbeitungsschritte zur Veränderung von Daten einer Datenbank zusammengefaßt sind und die gewährleistet, daß dabei die Datenbank von einem konsistenten Zustand in einen nächsten konsistenten Zustand überführt wird.</p> <p>Wird eine Transaktion unterbrochen (z.B. durch einen Systemausfall), wird ein Roll-Back durchgeführt.</p>
Windows32	Sind alle Betriebssysteme über Windows for Workgroups (Windows 3.11).

Anhang D:

ABBILDUNGSVERZEICHNIS

Abbildung 1: Der Navigator des DECS-Administrator	11
Abbildung 2: Pumpgate - Ansicht der Datenquellen	13
Abbildung 3: Pumpgate - Ansicht der Datenquellen II	13
Abbildung 4: PercussionMapper	15
Abbildung 5: Felder erstellen	21
Abbildung 6: Gestaltung der Teilmaske Historie	22
Abbildung 7: Gemeinsame Felder	22
Abbildung 8: Ausgabe der Ansichtsformel	23
Abbildung 9: Ausgabe der Ansichtsformel II	24
Abbildung 10: Screenshot von VIS KIS	25
Abbildung 11: Menu	26
Abbildung 12: Fortschrittsanzeige	34
Abbildung 13: Die Notwendigkeit der Datenbank-Connectivity aus der Sicht des Anwendungsentwicklers [Kyle Geiger 1996, Kapitel 1, Seite 27]	37
Abbildung 14: Die Notwendigkeit der Datenbank-Connectivity aus der Sicht des Benutzers [Kyle Geiger 1996, Kapitel 1, Seite 26]	38
Abbildung 15: ODBC Architektur	43
Abbildung 16: Eingabe Trennzeichen oder feste Wortlänge	52
Abbildung 17: Festlegung des Trennzeichens	53
Abbildung 18: Userinterface	53
Abbildung 19: ODBC Treiber	55
Abbildung 20: ODBC Datenquellen BenutzerDSN	56
Abbildung 21: ODBC Datenquellen SystemDSN	57
Abbildung 22: uselsx	59
Abbildung 23: Beziehungen zwischen LS:DO-Klassen	60
Abbildung 24: MSQuery	66
Abbildung 25: Oracle 32bit ODBCtest	67
Abbildung 26: Gestaltung Masken	74
Abbildung 27: Programmablauf ASCII	84
Abbildung 28: Benutzerschnittstelle Import von ASCII	85
Abbildung 29: Programmablauf Notes	86
Abbildung 30: Benutzerschnittstelle Import aus Notes	87
Abbildung 31: Programmablauf ODBC	88
Abbildung 32: Benutzerschnittstelle Import von ODBC	89
Abbildung 33: Auswahldialog Exporttyp	92
Abbildung 34: Eigenschaften des Felds fldSeparator	93
Abbildung 35: Spaltenformel in der Ansicht	95
Abbildung 36: Navigator	96

Anhang E:

TABELLENVERZEICHNIS

<i>Tabelle 1: Vergleich von RDBMS und Lotus Notes</i>	36
<i>Tabelle 2: Funktionsaufrufe der X/OPEN ISAM-Programmierschnittstelle</i>	42
<i>Tabelle 3: Namenskonventionen</i>	71
<i>Tabelle 4: Actionbuttons in Masken</i>	73
<i>Tabelle 5: Actionbuttons in Ansichten</i>	73