Fallstudie

Anwendungsintegration mit MBS Navision 4.0

vorgelegt im Rahmen der Diplomarbeit

Integrationslösungen im ERP-Umfeld -

State of the Art und Entwicklung einer Fallstudie am Beispiel von MBS Navision

von René Allissat, 16. Januar 2006

Inhalt

Abbildungsverzeichnis

Bearbeitungshinweise

Die vorliegende Fallstudie zeigt verschiedene Möglichkeiten der Anwendungsintegration in Microsoft Navision. Auf technischer Ebene lernen Sie unterschiedliche Varianten des Datenimports und –exports innerhalb des ERP-Programms kennen und erarbeiten eine Methodik, um externe Anwendungssysteme an Navision anzubinden. Thematisch erfahren Sie dabei die Vor- und Nachteile verschiedener Integrationsmethoden und können daraus bedeutsame Faktoren für die Integration von Anwendungen im inner- als auch im überbetrieblichen Umfeld herausstellen. Nach Durchführung der Studie sind Sie in der Lage, die gewonnenen Erkenntnisse auf die Integration von Anwendungssystemen im Allgemeinen zu übertragen.

Die Fallstudie ist so konzipiert, dass die einzelnen Kapitel inhaltlich und technisch aufeinander aufbauen: Die ersten Übungen bilden das Fundament für nachfolgende, komplexere Aufgaben. Dies gilt ebenso für den Schwierigkeitsgrad der Studie; anfänglich detailliert beschriebene Grundlagen, z.B. Erstellung von neuen Elementen und das Zurechtfinden in den Menüs, werden in höheren Kapiteln vorausgesetzt. Zweck und Hintergründe einzelner Arbeitsschritte werden erläutert, vor allem wenn es sich um bislang noch nicht erwähnte Programmfunktionen oder Elemente des Integrationsschwerpunkts handelt. Damit eignet sich die Studie für das Selbststudium, insbesondere wenn Sie bereits über Kenntnisse im Umgang mit ERP-Systemen und der Programmierung verfügen. Sofern Sie weitere Hintergrundinformationen benötigen, finden Sie im Anhang der Studie einige Literaturhinweise.

Innerhalb des Dokuments werden einheitliche typographische Konventionen verwendet, um die Durchführung der Studie zu erleichtern: Dateinamen und Programmcode sind in der Schriftart `Courier` abgebildet, Codezeilen werden zudem eingerückt und fett dargestellt. Menüs, Menüpunkte und Schaltflächen innerhalb der beschriebenen Programme sind *kursiv* gekennzeichnet.

Szenario

Zu Beginn des Jahres wurde der mittelständische Büromöbelgroßhändler GeneriCom GmbH von der Cronus AG übernommen. Die Außendienstmitarbeiter der GeneriCom nutzen zur mobilen Kontakt- und Artikelinformation sowie zur Vorerfassung von Aufträgen die Access-Applikation „WinCRM 95“. Das recht einfache Programm wird seit etwa 10 Jahren erfolgreich eingesetzt und ist den Mitarbeitern entsprechend vertraut. Mittelfristig soll es daher weiterbenutzt werden. Langfristig strebt die Cronus AG an, mobile proprietäre Systeme durch eine Portallösung zu ersetzen, allerdings erst, wenn ein mobiler Internetzugang für alle Mitarbeiter verfügbar und wirtschaftlich ist.

Bisher setzte die GeneriCom zur Abwicklung der wichtigsten betrieblichen Prozesse eine Individualsoftware ein. Der Abgleich von Kunden- und Artikeldaten mit WinCRM

erfolgte über Textdateien. In WinCRM erfasste Aufträge wurden vom Außendienst ausgedruckt und an den vertrieblichen Innendienst weitergeleitet. Nach abschließender Prüfung übernahm dieser die Aufträge in die Unternehmenssoftware.

Nach dem Aufkauf durch die Cronus AG wurde bei den Mitarbeitern der ehemaligen GeneriCom Microsoft Navision als neues ERP-System eingeführt, so dass diese auf den zentralen Datenbestand von Cronus zugreifen können.

Während die Mitarbeiter des Customer Service (der ehemalige Innendienst der GeneriCom) von den neuen Funktionen des Systems und der engen Zusammenarbeit mit der Cronus-Zentrale profitieren, gilt es nun, auch den Außendienst an das neue System anzubinden.

Aufgabenstellung

Ihre Aufgabe ist es, den Datenaustausch zwischen Navision und WinCRM zu ermöglichen.

Die Umsetzung der Lösung soll dabei von Navision aus erfolgen, damit der Datenabgleich sowohl durch den Außendienst selbst als auch ggf. durch Innendienstmitarbeiter über das Netzwerk erfolgen kann.

Da aktuelle Kontaktinformationen für den Außendienst geschäftskritisch sind, werden in einem ersten Schritt zunächst die Kundendaten aus Navision in die Access-Datenbank übernommen. Ebenso soll es möglich sein, neue und geänderte Adressinformationen zurück nach Navision zu kopieren.

Zweitens soll der Export von Artikeldaten aus Navision nach WinCRM möglich sein, so dass der Außendienst im Kundendialog Zugriff auf tagesaktuelle Preise, Lieferbestände und Artikelinformationen hat.

Ebenso benutzt der Außendienst WinCRM, um Kundenaufträge zu erfassen. Die Verknüpfung mit dem bisherigen ERP-System war jedoch unzureichend. Ideal wäre es, den Prozess medienbruchfrei zu gestalten und die vom Außendienst erfassten Aufträge direkt an Navision weiterzuleiten.

1. Vorbereitung

1.1. Einrichtung von Navision

Innerhalb der Fallstudie arbeiten Sie mit einer lokalen Installation von Navision, bei der alle zum Betrieb der Software notwendigen Dateien auf Ihrem Client-Rechner installiert sind, einschließlich der verwendeten Datenbank des Beispielmandanten „CRONUS AG“. Obwohl dieser Betriebsmodus für ein Client/Server-basiertes ERP-System unüblich scheint, entspricht es jedoch der Unternehmenspraxis, Änderungen an einem bestehenden ERP-System zunächst innerhalb einer eigenständigen Testumgebung zu entwickeln. Es wäre eindeutig zu riskant, Entwicklung und Test neuer Funktionen direkt an der realen Unternehmensdatenbank durchzuführen. Durch eine Lokalinstallation hingegen ist die gefahrlose Neuerstellung von Programmteilen und Objekten in Navision möglich, ebenso können Sie neue Funktionen oder Befehle nach dem „Trial and Error“-Prinzip ausprobieren, ohne einen gemeinsamen Datenbestand zu gefährden.

Um die Testdatenbank für die Fallstudie auf Ihrem Rechner einzurichten, kopieren Sie zunächst die Datei `database.fdb` von der zweiten Navision CD oder dem Netzwerk auf Ihre lokale Festplatte. Diese Datei enthält die kompletten Daten des Mandanten „CRONUS AG“.

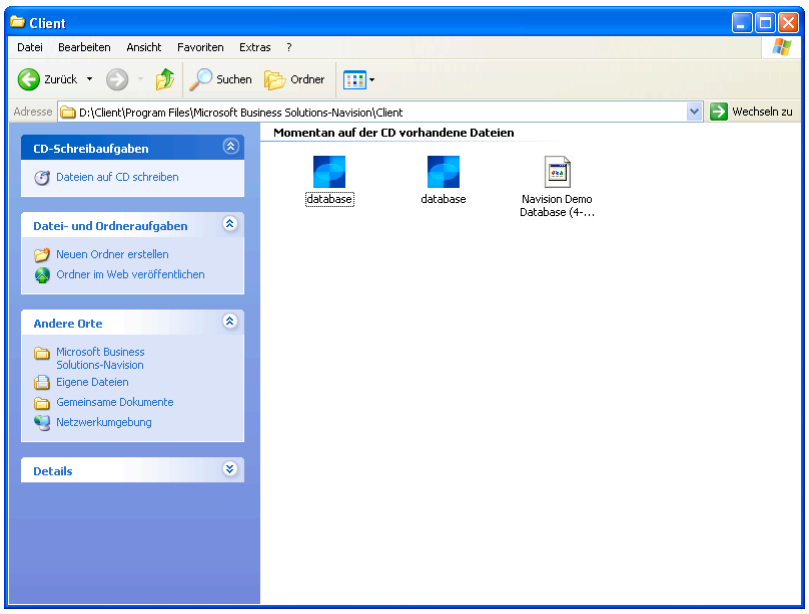

Abbildung 1: Demo-Mandant "CRONUS AG" auf der 2. Navision CD

Nachdem der Kopiervorgang abgeschlossen wurde, starten Sie Navision. Wählen Sie den Menüpunkt *Datei → Datenbank → Öffnen*. Geben Sie im nun erscheinenden Fenster als Datenbankname die von Ihnen kopierte Datei an und bestätigen Sie mit *OK*. Innerhalb der geladenen Datenbank können Sie anschließend über *Datei → Mandant → Öffnen* den Mandanten „CRONUS AG“ auswählen und öffnen. Bestätigen

Sie einen eventuell auftauchenden Hinweis zur Rücksetzung des Arbeitsdatums mit *OK*.

Der Funktionsumfang von Navision wird über Lizenzdateien definiert. Diese bestimmen unter anderem die von einem Unternehmen erworbenen und damit nutzbaren Programm-Module, die maximale Datenbankgröße und ebenso, in welchem Umfang das System editiert werden kann. Für die Benutzung von C/SIDE, der Entwicklungsumgebung von Navision, ist erwartungsgemäß eine umfassende Lizenz erforderlich. Eine solche Entwicklerlizenz ermöglicht es nicht nur, neue Objekte zu erstellen, sondern auch nahezu alle bereits vorhandenen Elemente von Navision zu editieren.

Bevor es losgehen kann, prüfen Sie daher mit dem Menüpunkt *Extras* → *Lizenzinformation*, ob Sie über ausreichende Rechte für die Nutzung der Programmierumgebung verfügen. Sofern an dieser Stelle nicht bereits die Lizenz der PFH Göttingen erscheint, können Sie die bestehende Lizenz mittels *Importieren…* ersetzen. Wählen Sie dazu im erscheinenden Dateidialog die Lizenz der PFH – die Dateiendung von Lizenzen ist `*.flf` – und öffnen diese. Bestätigen Sie den darauf erscheinenden Hinweis mit *Akzeptieren*.

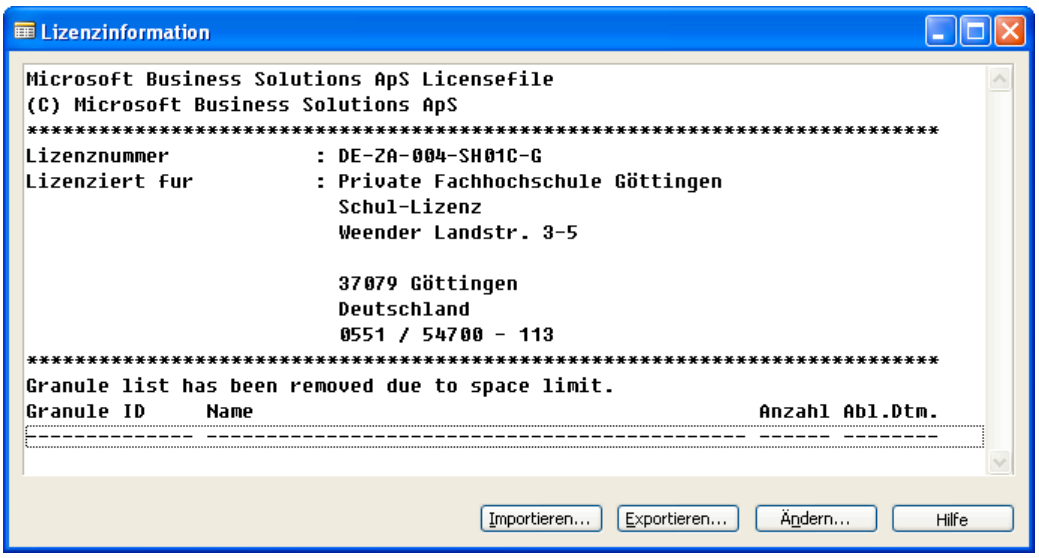

Abbildung 2: Lizenzinformationen

Nun verfügen Sie über eine lokale Datenbank und nahezu unbegrenzte Nutzerrechte und können die Aufgaben der Fallstudie umsetzen.

1.2. Installation von WinCRM

Installieren Sie die vom Außendienst verwendete Access-Datenbank „WinCRM 95“. Dazu ist es ausreichend, den Inhalt des Archivs `CRM95.zip` in das Laufwerk `C:\` zu entpacken.

Auf Ihrer Festplatte finden Sie jetzt das Verzeichnis `C:\CRM95`. Dieses enthält die Datenbank `CRM95.mdb` sowie den Ordner `Daten`. Letzterer beinhaltet Text- und XML-Dateien, die beim Start und beim Beenden der Datenbank automatisch generiert werden und sich damit zum Datenaustausch mit anderen Programmen eignen.

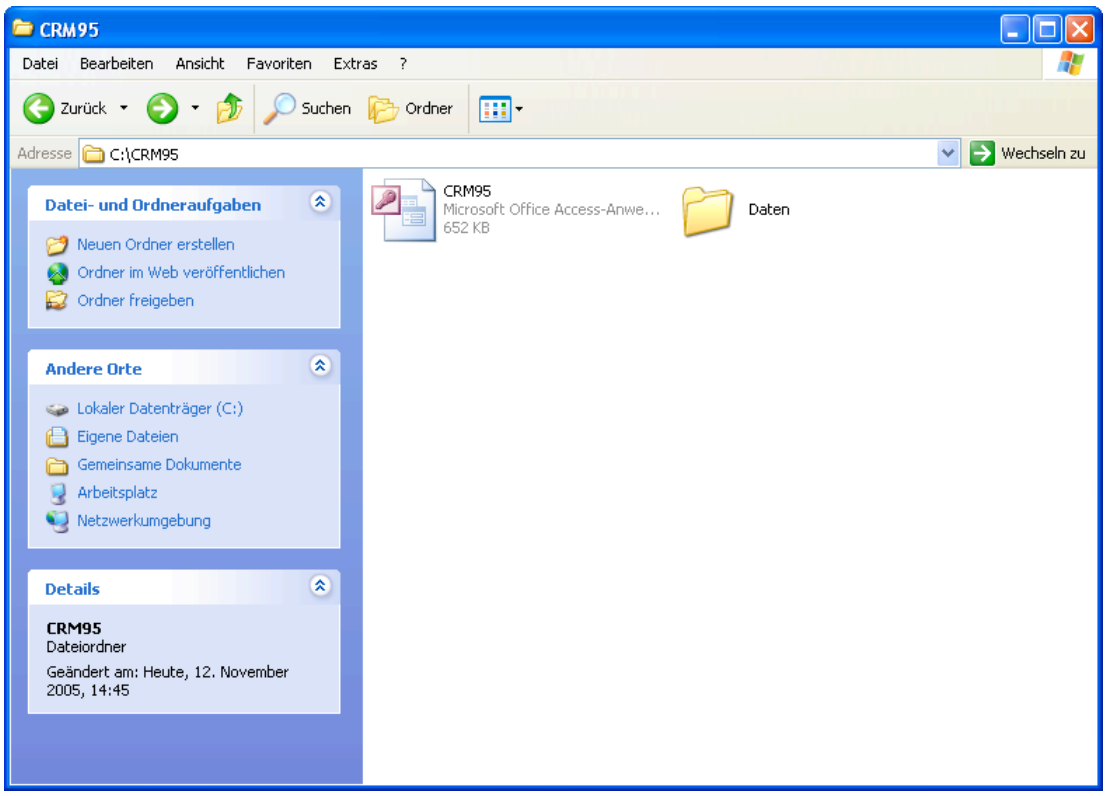

Abbildung 3: Lokalinstallation von WinCRM

2. Synchronisation der Kundendaten

Aktuelle Kundendaten sind eine wesentliche Arbeitsgrundlage für den Außendienst. Es muss sichergestellt werden, dass die Außendienstmitarbeiter über stets aktuelle Kundenstammdaten aus Navision verfügen. Ebenso akquiriert der Außendienst neue Kunden und erfährt oft als erster von Änderungen bei Stammkunden. Er muss daher ebenso in der Lage sein, neue und geänderte Datensätze nach Navision zu übertragen. In einem ersten Schritt entwickeln Sie daher eine Lösung, die den bidirektionalen Austausch der Kundendatensätze zwischen WinCRM und Navision ermöglicht.

2.1. Export aus Navision

Öffnen Sie die Access-Datenbank `CRM95.mdb`. Im nun erscheinenden Hauptmenü wählen Sie *Kundendaten bearbeiten*. Schauen Sie sich den Beispiel-Eintrag im Formular „Kunden“ an und geben Sie eine weitere Adresse ein. Durch einen Klick auf *Datenbankfenster anzeigen* im Hauptmenü von WinCRM gelangen Sie zur Standardansicht von Access. Öffnen Sie hier die Tabelle „Kunden“ und sehen Sie sich ebenfalls die Feldbezeichnungen an.

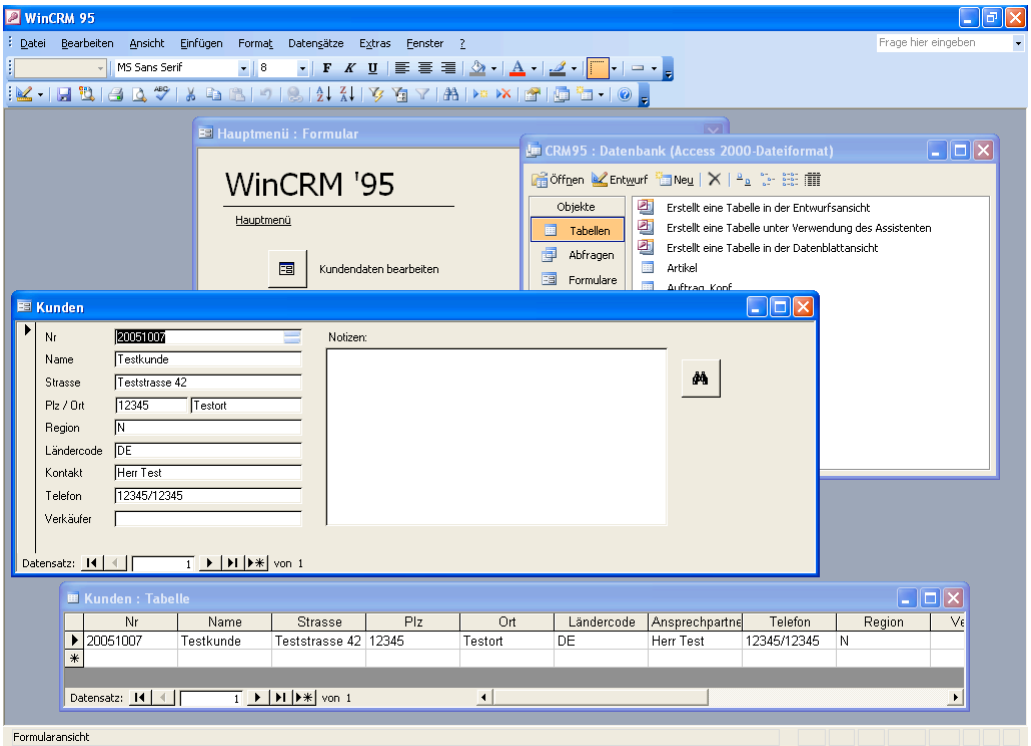

Abbildung 4: WinCRM

Beenden Sie anschließend WinCRM, indem Sie im Hauptmenü-Formular *Beenden und exportieren* klicken. Die Schaltfläche ist mit recht einfachem Programmcode hinterlegt, der die Adressen in der Textdatei `C:\CRM95\Daten\Kunden.txt` speichert und

anschließend Access beendet. Öffnen Sie die Datei mit einem Texteditor und vergleichen Sie deren Aufbau mit dem der Tabelle.

Starten Sie nun Navision und rufen Sie über das Menü *Extras → Object Designer* den Object Designer auf. Dieser ist das gewissermaßen das Hauptmenü der in Navision integrierten Entwicklungsumgebung C/SIDE. Hier sind alle Elemente und Funktionen von Navision in Form von sieben verschiedenen Objekttypen aufgeführt; diese lassen sich auf der linken Seite des Fensters selektieren.

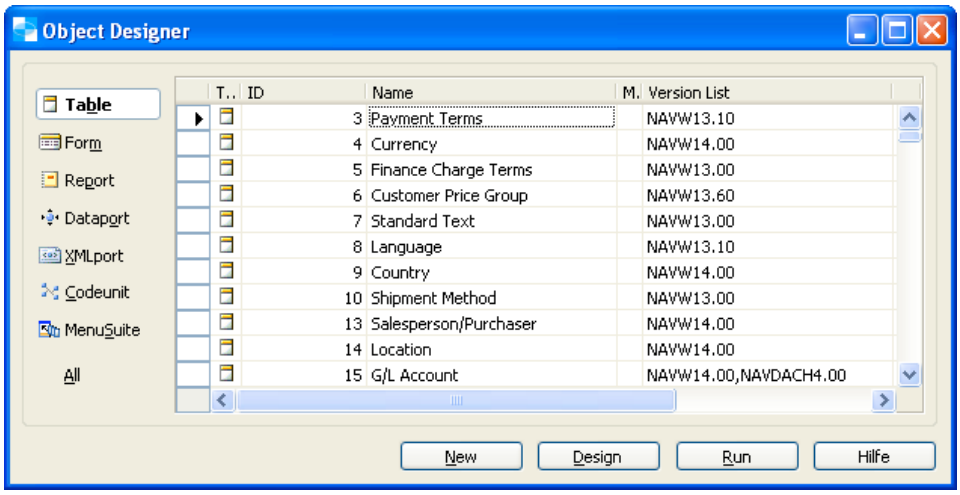

Abbildung 5: Object Designer

Die einzelnen Objekte in der Listenansicht sind nach eindeutigen Kennungsnummern sortiert, die genau definierten Konventionen folgen. Dies geschieht einerseits, um den Zugriff auf Funktionen abhängig von der erworbenen Lizenz zu erlauben oder zu sperren, andererseits, um unterschiedliche Programmbestandteile voneinander abzugrenzen. Objekte der grundlegenden Hauptanwendungen haben gewöhnlich eine niedrige Nummer, während optionale Zusatzmodule in hohen Ziffernbereichen angesiedelt sind. Die Nummernkonventionen sind in folgenden Gruppen geordnet.

Standardapplikation	1 – 9.999 Base Application Design Area
Länderanpassungen	10.000 – 49.999 Country Design Area
Kundenanpassungen	50.000 – 99.999 Customer Design Area
Zusatzmodule	100.000 – 999.999.999 NDP Design Area

Abbildung 6: Nummernkonventionen in Navision

Wenn Sie neue Elemente innerhalb der Programmierungsumgebung anlegen, können Sie diesen eine beliebige noch nicht vorhandene Zahl im Bereich der „Kundenanpassungen“ zuweisen.

Der Im- und Export von Daten aus Textdateien kann über Dataports realisiert werden. Legen Sie daher einen neuen Dataport an. Klicken Sie in der linken Leiste auf *Dataport* und anschließend auf *New*; ein zweispaltiges Fenster, der Dataport Designer, erscheint.

In einem ersten Schritt soll ein Dataport erstellt werden, der alle Adressen aus Navision exportiert. Ziel ist eine Textdatei, die den gleichen Aufbau wie die eben betrachtete `Kunden.txt` hat.

Rufen Sie innerhalb des Dataport Designers die Eigenschaften des Dataports auf, entweder durch *Ansicht → Properties* oder das entsprechende Symbol  in der Menüleiste. Hier können Sie einige allgemeine Einstellungen des Dataports vornehmen. Bereits an dieser Stelle ist es zweckmäßig, einen aussagekräftigen Namen und eine ID gemäß der Navision-Nummernkonventionen zu vergeben.

Setzen Sie daher die Eigenschaft ID z.B. auf *99998* und Name auf *KundenExport*. Stellen Sie weiterhin Import auf *No* – dieser Dataport soll lediglich den Export von Daten ermöglichen. Geben Sie zudem den FileName der Datei mit komplettem Pfad an, z.B. `C:\Kunden2.txt`. Die betreffende Datei muss nicht vorhanden sein, Navision erstellt diese beim Ausführen des Dataports automatisch.

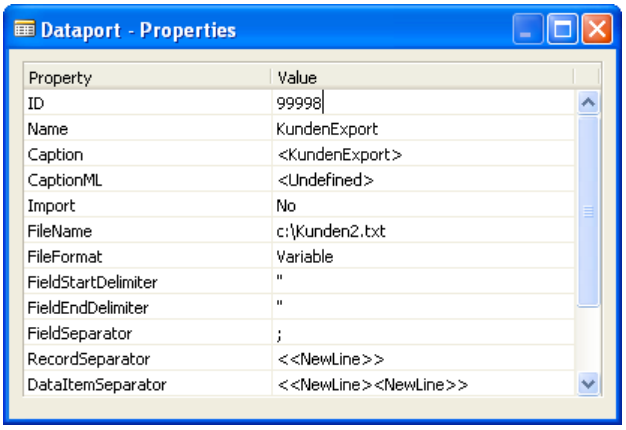

Abbildung 7: Eigenschaften des Dataports

Die nachfolgenden Eigenschaften betreffen den grundlegenden Aufbau der Textdatei. Die Datei `Kunden.txt` enthält durch Semikola voneinander getrennte Werte variabler Länge, die jeweils in Anführungszeichen eingeschlossen sind. Diese Information können Sie gemäß obiger Abbildung auf die entsprechenden Eigenschaften

übertragen: Setzen Sie das FileFormat auf *Variable*. FieldStartDelimiter und FieldEndDelimiter sind jene Zeichen, die Start und Ende eines einzelnen Datenfeldes markieren, in diesem Fall Anführungszeichen. Der FieldSeparator bezeichnet das Symbol, welches einzelne Felder innerhalb der Textdatei voneinander trennt, dies ist hier das Semikolon. Eine neue Zeile innerhalb der Textdatei markiert einen neuen Datensatz, der voreingestellte RecordSeparator <<*NewLine*>> kann beibehalten werden. (Ein in << >> eingeschlossener Wert bedeutet, dass dies die unveränderte Standardeinstellung von Navision ist)

Ferner gibt die Eigenschaft UseReqForm an, ob beim Ausführen des Dataports ein Dialogfeld geöffnet wird, in dem der Benutzer Filtereinstellungen vornehmen und einen Dateinamen angeben kann. Da Sie den Dataport später in ein Programm einbinden wollen, setzen Sie diese Eigenschaft auf *No*. Kehren Sie zum Dataport Designer zurück, das Eigenschaftenfenster kann geöffnet bleiben.

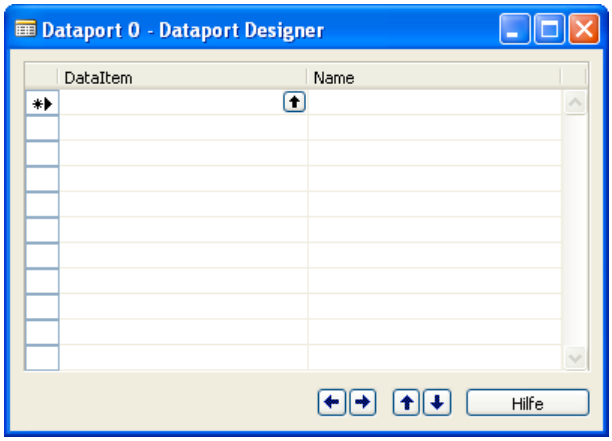

Abbildung 8: Dataport Designer

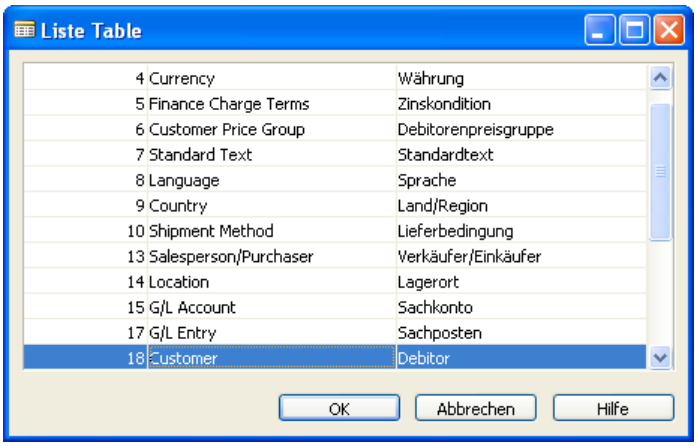

Abbildung 9: Tabellenauswahl

Im Feld DataItem des Dataport Designers können Sie die Tabellen von Navision angeben, deren Daten im- oder exportiert werden sollen. Klicken Sie dazu auf ein freies Feld innerhalb von DataItem und anschließend auf den erscheinenden Pfeil. In

dem folgenden Dialogfeld wählen Sie Tabelle 18 (Customer) aus – hier sind alle Kundendaten gespeichert.

Öffnen Sie jetzt den Menüpunkt *Ansicht → Dataport Fields*. Im jetzt geöffneten Field Designer können Sie festlegen, welche Datenfelder der Tabelle in die Textdatei exportiert werden sollen. Die Reihenfolge der Felder innerhalb des Designers bestimmt hierbei auch die Reihenfolge in der Datei. Sie können Felder hinzufügen, indem Sie ein leeres Feld der Spalte SourceExpr anklicken; nach einem weiteren Klick auf den nun erscheinenden Pfeil öffnet sich ein Dialog mit allen verfügbaren Feldern der Tabelle „Customer“. Da die Elemente der Textdatei eine variable Länge besitzen und durch Semikola getrennt werden, können Sie die Felder StartPos und Width für Textdateien mit fester Datensatzlänge ignorieren.

Nun können Sie die Struktur der `Kunden.txt` bzw. der Tabelle „Kunden“ aus Access in Navision nachbilden. Wählen Sie dazu die korrespondierenden Datenfelder der weitaus umfangreicheren Navision-Tabelle wie folgt aus:

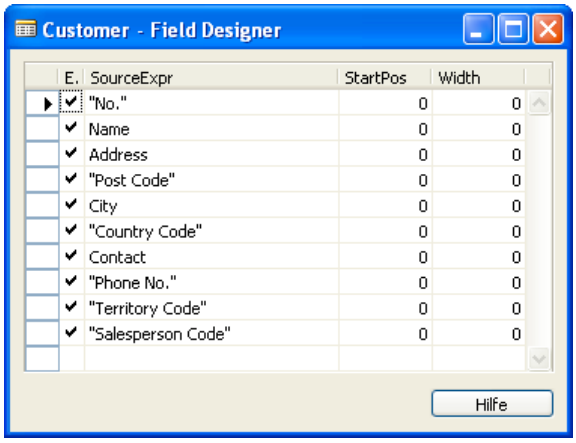

Abbildung 10: Field Designer

Schließen Sie anschließend Field Designer und Dataport Designer. Bestätigen Sie die Nachfrage, ob der Dataport kompiliert und gespeichert werden soll, mit *Yes*. Ihr Dataport ist jetzt einsatzbereit.

Selektieren Sie Ihren Dataport im Object Designer und klicken Sie auf *Run*. Alle Datensätze der Tabelle Debitoren werden nun in die von Ihnen angegebene Datei geschrieben. Dieser Vorgang wird durch ein Statusfenster angezeigt. Ist der Export abgeschlossen, so schließt sich dieses Fenster automatisch.

Rufen Sie anschließend die neu erstelle Datei auf und vergleichen Sie deren Aufbau mit der ursprünglichen `Kunden.txt`. Stimmen Syntax und Anzahl der Felder überein?

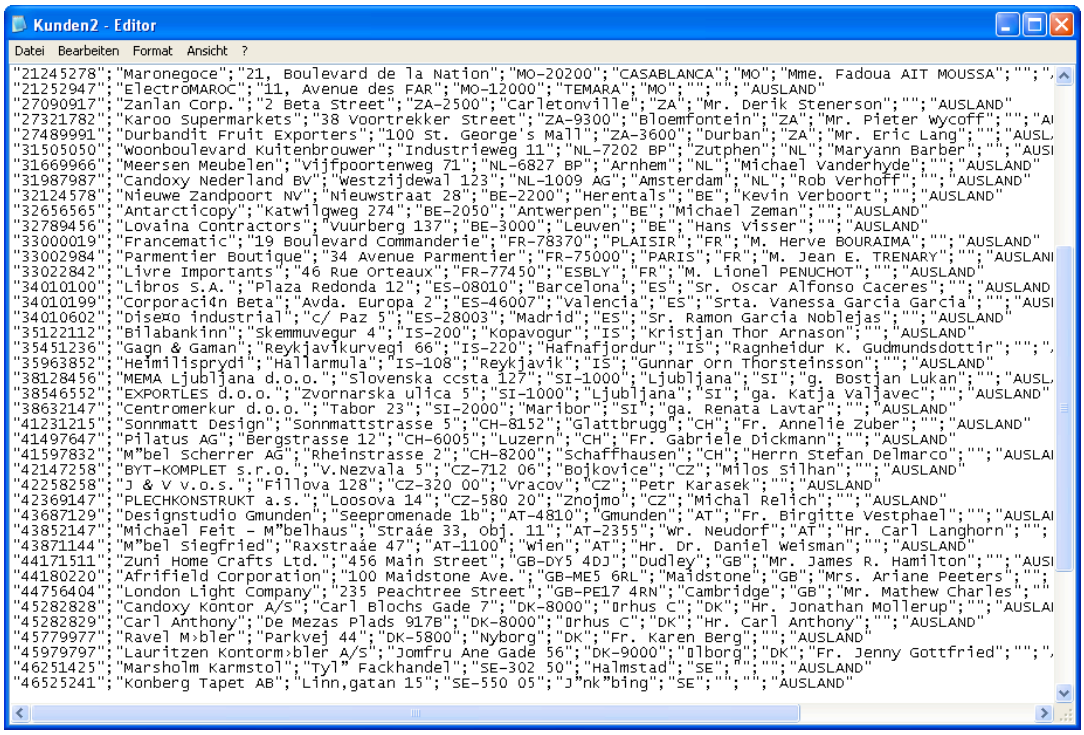

Abbildung 11: Exportierte Kundendaten

Gleichen sich die Textdateien in ihrer Struktur, so haben Sie erfolgreich eine Möglichkeit entwickelt, die Kundendaten nach Access zu exportieren. Bevor Sie jedoch die zahlreichen Adressen nach Access übernehmen, sollten sie zunächst den Adressimport anhand weniger Beispieldaten umsetzen.

2.2. Import aus Access

Die Datei `C:\CRM95\Daten\Kunden.txt` enthält jene Kunden, die Sie eingangs in Access angesehen bzw. selbst angelegt haben, die aber noch nicht in Navision vorhanden sind. Um diese von Navision aus zu importieren, begeben Sie sich wieder in den Object Designer und erstellen Sie einen weiteren Dataport. Rufen Sie dessen Eigenschaften auf und vergeben Sie erneut ID und Name, z.B. *99999* und *KundenImport*. Setzen Sie die Import-Eigenschaft auf *Yes*; dieser Datenport soll nur zum Import von Daten genutzt werden. Geben Sie als Dateinamen die Datei `C:\CRM95\Daten\Kunden.txt` an. Setzen Sie Dateiformat, Start- und Endmarker sowie Trennzeichen wie im vorherigen Beispiel; ebenso ist UseReqForm erneut zu verneinen.

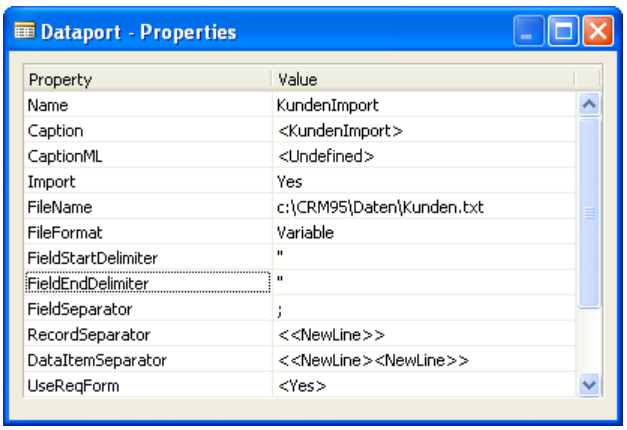

Abbildung 12: Dataport Eigenschaften

Schließen Sie die Eigenschaften und geben Sie im Dataport Designer erneut die Tabelle Customer an. Rufen Sie deren Dataport Fields auf und bilden Sie die Felder genau wie im vorherigen Beispiel ab. Schließen Sie nun den Field Designer.

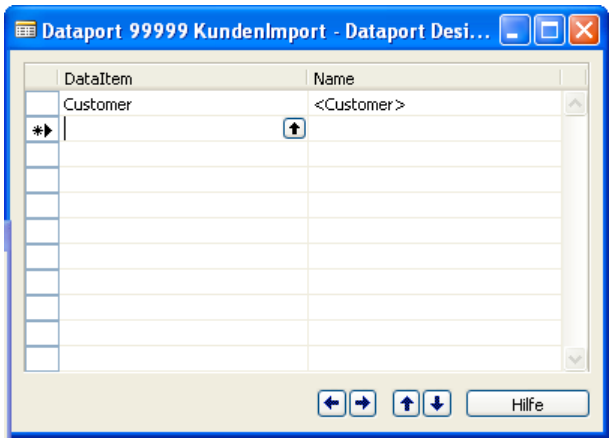

Abbildung 13: Dataport Designer

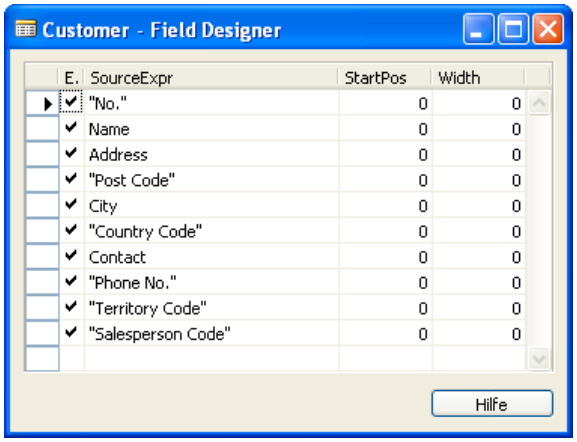

Abbildung 14: Field Designer

Kehren Sie zum Dataport Designer zurück. Markieren Sie die erste Zeile, welche die der Tabelle Customer enthält und sehen Sie sich abschließend deren Eigenschaften an. Für den Datenimport besonders relevant sind die untersten drei Optionen.

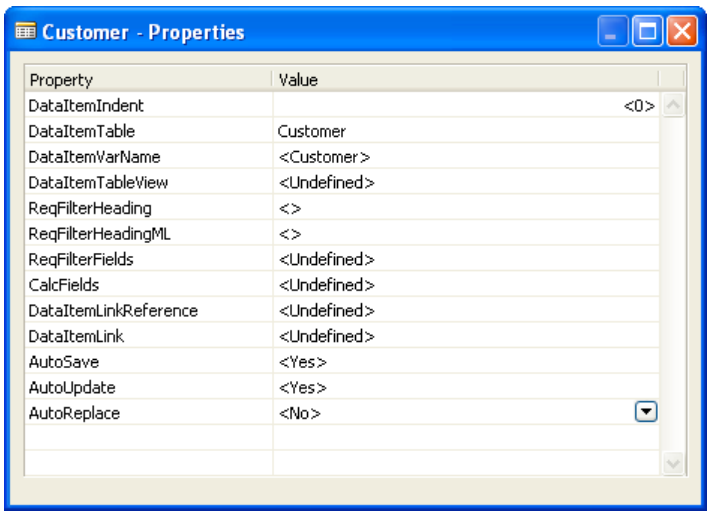

Abbildung 15: Eigenschaften des Felds "Customer"

„AutoSave“ legt fest, ob Navision neu importierte Datensätze automatisch in die entsprechende Tabelle übernimmt oder (vorerst) ignoriert.

Werden Datensätze importiert, deren Primärschlüssel bereits in der korrespondierenden Navision-Tabelle vorhanden sind, definiert „AutoUpdate“, ob diese durch die importierten Daten ergänzt werden sollen – selbst dann, wenn nur einzelne Datenfelder importiert werden.

„AutoReplace“ legt fest, ob Datensätze mit dem gleichen Primärschlüssel durch importierte Datensätze vollständig ersetzt werden sollen.

Im vorliegenden Fall können Sie alle Eigenschaften auf ihren voreingestellten Werten belassen. Damit können Datensätze hinzugefügt werden, die in Access neu angelegt wurden. Auch Änderungen in bereits bestehenden Datensätzen werden von Navision übernommen. Sollen durch den Außendienst neu eingegebene Adressen nicht importiert werden, so könnten Sie „AutoSave“ auf *No* setzen. Das verneinen von „AutoUpdate“ würde dazu führen, dass in Access geänderte Adressen ignoriert werden.

Selektieren Sie den Dataport Designer, so dass dieser das aktuelle Fenster ist. Wählen Sie nun den Menüpunkt *Datei → Run* aus. Entgegen der gleichnamigen Funktion des Object Designers startet dieser einen Testlauf des Dataports, ohne jedoch wirklich Datensätze (Import) oder Dateien (Export) zu verändern. Erkennt Navision einen Fehler, wird eine entsprechende Fehlermeldung angezeigt. Wenn alles in Ordnung ist, verrichtet der Dataport kommentarlos seine Arbeit.

Sie verfügen jetzt auch über einen Dataport für den Import von Kundendaten. Speichern Sie Ihre Arbeit über *Datei → Save*; bestätigen Sie die Sicherheitsabfrage mit *OK*.

Standardmäßig werden durch einen Dataport Daten importiert, ohne den Inhalt einzelner Datenfelder auf Gültigkeit zu überprüfen. Dies könnte zu fehlerhaften und inkonsistenten Datensätzen in Navision führen. Um dies zu vermeiden, rufen Sie noch einmal den Field Designer des gerade bearbeiteten Dataports auf. Sehen Sie sich das Eigenschaften-Fenster für die einzelnen Datenfelder an. Interessant ist die Eigenschaft CallFieldValidate, die per Voreinstellung auf den Wert *No* gesetzt ist. Die Eigenschaft bestimmt, ob für ein Feld hinterlegter Programmcode aufgerufen werden soll, wenn das Feld geändert wird. Typischerweise werden auf diese Art neu eingefügte Daten überprüft, mit anderen Tabellen verglichen oder mit referenzierten Tabellen abgestimmt.

Setzen Sie für die Felder Country Code, Territory Code und Salesperson Code die Eigenschaft CallFieldValidate auf *Yes*. Deren OnValidate-Funktion wird fortan bei jeder Änderung des Feldes aufgerufen und prüft, ob der eingegebene Länder-, Regions- bzw. Verkäufercode tatsächlich existiert. Falls nicht, wird der Import abgebrochen und eine Warnmeldung ausgegeben, die auf den fehlenden Referenzwert aufmerksam macht.

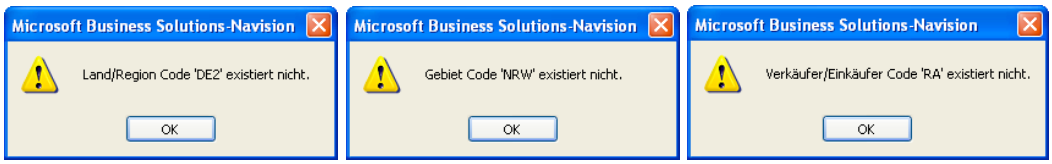

Abbildung 16: Mögliche Fehlermeldungen der Validierung

Schließen Sie Field Designer und Dataport Designer und bestätigen Sie die Rückfrage, ob die Änderungen in dem Dataport gespeichert werden sollen.

2.3. Test der Dataports

Starten Sie den gerade erstellten Dataport über die Schalfläche *Run* des Object Designers. Findet Navision die Daten in `Kunden.txt` so vor, wie durch den Dataport definiert, werden die neuen Kunden anstandslos übernommen. Falls ein Datensatz den gerade festgelegten Validierungsregeln widerspricht, wird dies angezeigt. Starten Sie in diesem Fall WinCRM und ändern Sie Ihre Adressdaten entsprechend.

Sie können den erfolgreichen Import überprüfen, indem Sie die Debitorenkarte aufrufen. Selektieren Sie dazu im linken Menü den Bereich „Finanzmanagement“ und anschließend in der oberen Baumstruktur *Debitoren → Debitoren*. Mit dem Button *Debitor → Übersicht* (F5) öffnen Sie eine Listenansicht, in der Sie schnell die neuen

Datensätze auffinden können. Rufen Sie einen der Datensätze in der Debitorenkarte auf und überzeugen Sie sich davon, dass die Felder durch den Dataport richtig zugeordnet wurden.

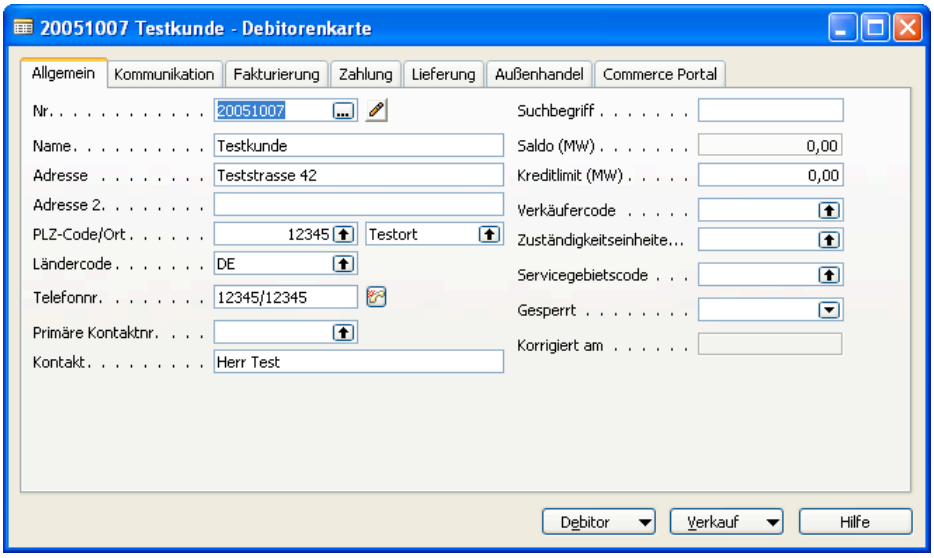

Abbildung 17: Debitorenkarte

Wenn der Import einwandfrei funktioniert, können Sie nun auch den zu Beginn entwickelten Dataport zum Export testen. Öffnen Sie diesen im ObjectDesigner mittels *Design*. Klicken Sie auf ein freies Feld innerhalb des Dataport Designers und rufen Sie die Eigenschaften auf. Setzen Sie die Eigenschaft Filename auf `C:\CRM95\Daten\Kunden.txt`. Schließen Sie den Dataport und klicken Sie im Object Designer auf *Run*. Quittieren Sie die Meldung, dass die alte Datei überschrieben wird, mit *Ja*.

Starten Sie nun die Access-Datenbank WinCRM. Begeben Sie sich über das Hauptmenü in das Formular Kundendaten. Zur besseren Übersicht können Sie sich ebenso Tabelle Kunden ansehen. Hat alles geklappt, so ist die Tabelle mit zahlreichen Datensätzen aus Navision gefüllt. Sie haben nun eine bidirektionale Integrationslösung ohne Programmierung entwickelt.

2.4. Einbindung in die Benutzeroberfläche

Bislang sind die Dataports nur über den Object Designer zugänglich. Damit jeder Anwender die neuen Funktionen nutzen kann, erstellen Sie ein neues Formular und binden dieses in das Navision-Menü ein.

Rufen Sie den Object Designer auf und klicken in diesem auf *Form*. Legen Sie über *New* ein neues Formular an. Das jetzt erscheinende Dialogfeld ermöglicht, dem

Formular eine Tabelle zuzuweisen oder einen Assistenten zur Erstellung zu benutzen. Beides ist hier nicht erforderlich, übergehen Sie die Einstellungen mit *OK*.

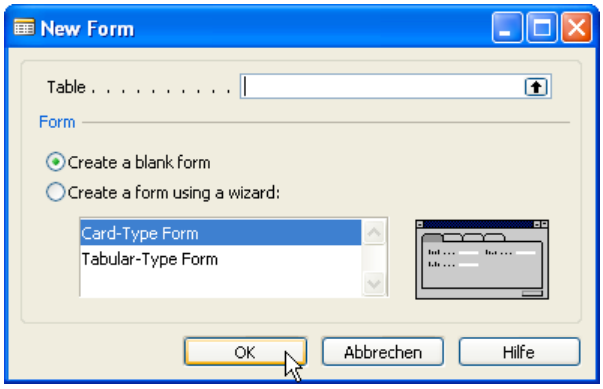

Abbildung 18: Neues Formular

Der Formular Designer unterscheidet sich nicht wesentlich von anderen grafischen Entwicklungswerkzeugen. Über die Toolbox (Symbolleiste: ) können Sie Steuerelemente in das Formular einfügen. Auch hier können Sie über *Eigenschaften* das Aussehen und Verhalten der selektierten Objekte bestimmen. Die Eigenschaften zum Benennen der Elemente sind „Name“ bzw. „Caption“.

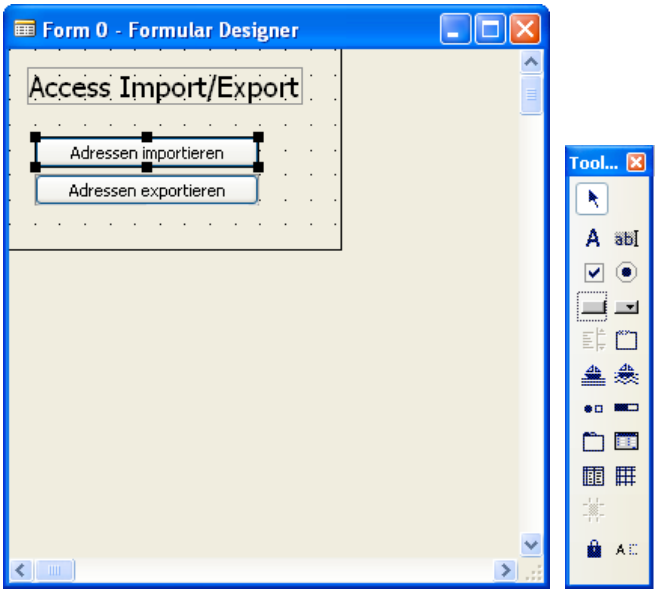

Abbildung 19: Formular Designer und Toolbox

Erstellen Sie ein Formular mit zwei Schaltflächen, die den Import bzw. Export der Kontaktdaten ermöglichen sollen. Sie können sich dabei an obigem Beispiel orientieren. Vergeben Sie abschließend sinnvolle Werte für die Eigenschaften „Name“ und „Caption“.

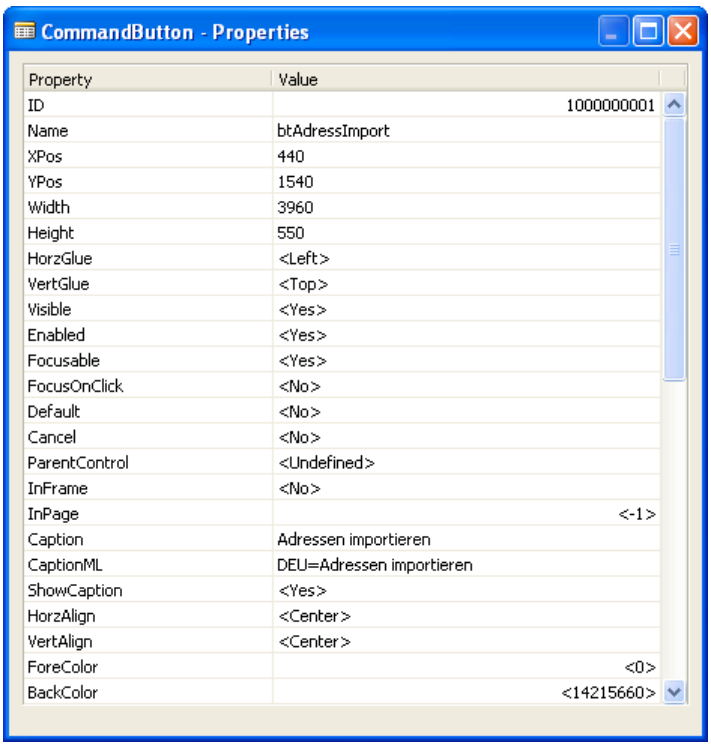

Abbildung 20: CommandButton Eigenschaften

Rufen Sie nun *Ansicht → C/AL Code* auf, um in den Editor der Navision-eigenen Programmiersprache C/AL zu gelangen. Ähnlich wie auch beim Eigenschaften-Fenster werden hier Optionen des gerade selektierten Elements angezeigt.

Wählen Sie eine der Schaltflächen aus. Der C/AL-Editor zeigt nun alle für diese Schaltfläche sinnvollen Prozeduren an. Um den Dataport auf Knopfdruck zu starten, schreiben Sie folgenden Code in die OnPush-Prozedur:

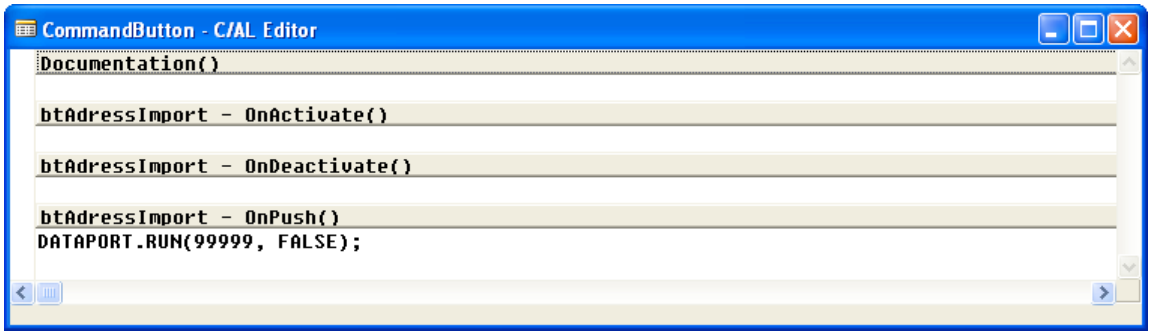

Abbildung 21: Programmcode des CommandButton

Das zweite Argument, hier auf „FALSE“ gesetzt, gibt an, ob das Abfragefenster des Dataports angezeigt werden soll, in dem der Nutzer Datenfilter und Dateiname angeben kann. Beachten Sie, dass in C/AL (in Anlehnung an die verwandte Programmiersprache PASCAL) jede Anweisung mit einem Semikolon abgeschlossen werden muss. Weisen Sie zudem Ihrem Dataport die richtige Nummer zu. Ihren

Programmcode können Sie jederzeit durch Druck auf F11 prüfen (*Extras → Compile*): Entdeckt der Interpreter einen Fehler, so zeigt Navision diesen an.

Verfahren Sie genauso mit der anderen Schaltfläche. Schließen und speichern Sie das neue Formular und starten Sie es über den Object Designer (*Run*). Prüfen Sie, ob die Schaltflächen funktionieren.

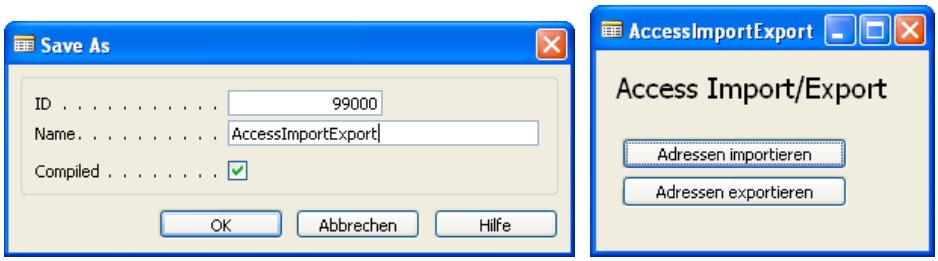

Abbildung 22: Sichern und Testen des neuen Formulars

Damit das Formular für Mitarbeiter des Vertriebs einfach erreichbar ist, fügen Sie es in die Menüstruktur am linken Bildschirmrand ein. Wählen Sie im Object Designer *Menu Suite* und klicken Sie auf *New*. Ein Dialogfeld fragt die gewünschte Designebene ab, wählen Sie hier *Company*, um die Änderung unternehmensweit verfügbar zu machen.

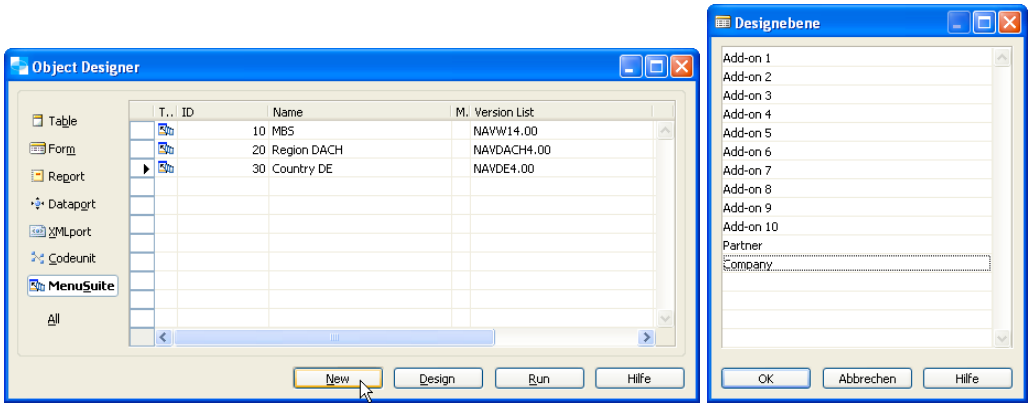

Abbildung 23: MenuSuite

Jetzt können Sie das Menü editieren. Wählen Sie in der unteren Bildschirmhälfte *Verkauf & Marketing*. Rechtsklicken Sie anschließend auf eine freie Stelle in der darüberliegenden Baumstruktur. Mit *Option erstellen…* können Sie ein neues Element einfügen. Wählen Sie im folgenden Dialogfeld Ihr neues Formular aus und vergeben Sie einen passenden Namen für den Menüpunkt. Bestätigen Sie mit *OK*.

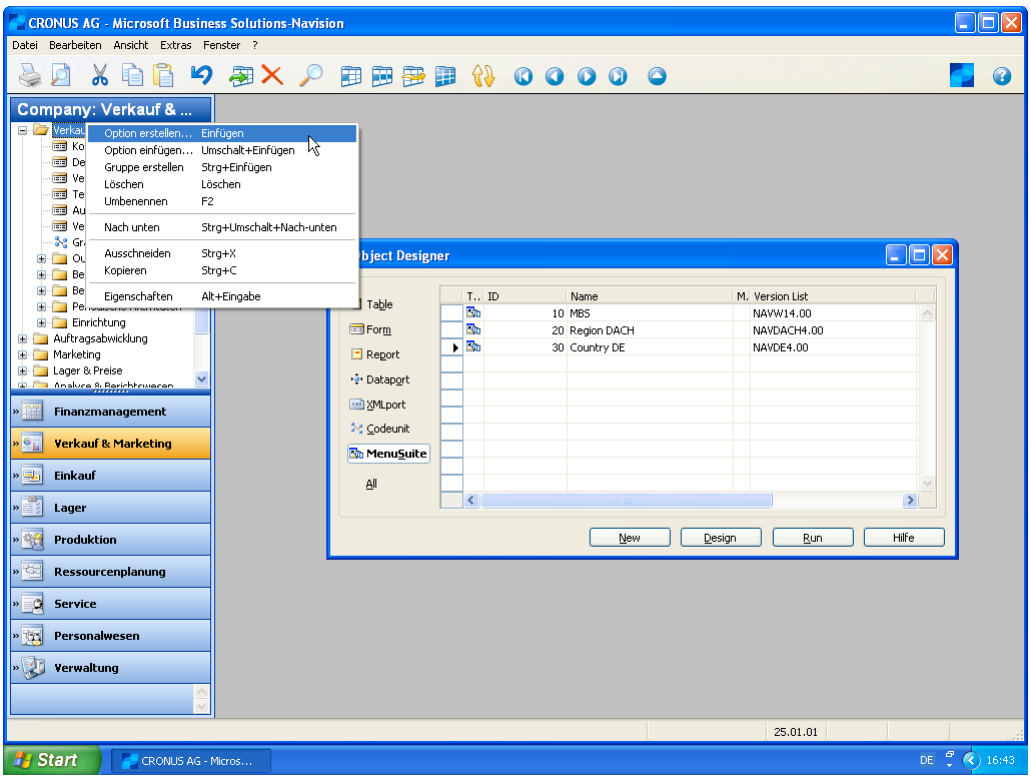

Abbildung 24: Menüs bearbeiten

Schließen Sie den Navigationsbereich-Designer, indem Sie auf die Überschrift der Seitenleiste rechtsklicken und den Menüpunkt 

auswählen. Bestätigen Sie im jetzt erscheinenden Fenster Ihre Änderungen mit *Ja*.

Ihre Integrationslösung ist nun komfortabel für Nutzer aus dem Bereich „Verkauf und Marketing“ zugänglich.

2.5. Filtern importierter Daten

Dataports sind ein einfach anzuwendendes und mächtiges Werkzeug, um automatisiert Daten zwischen Navision und Drittsystemen zu übertragen. Gegenüber moderneren Varianten, die in den nachfolgenden Kapiteln erörtert werden, haben sie vor allem in der Kommunikation mit älteren Anwendungen („Legacy Systems“) ihre Daseinsberechtigung, da hier Textdateien nicht selten der „kleinste gemeinsame Nenner“ des Datenaustauschs sind. Gerade deshalb ist es jedoch leicht vorstellbar, dass bereits ein fehlendes Trennzeichen oder die falsche Zuordnung einzelner Spalten negative Auswirkungen auf den Datenbestand haben. Ein falsch erstellter oder schlecht programmierter Dataport kann zu inkonsistenten Daten und in der Folge zu Fehlfunktionen in Navision führen. Es obliegt also dem Entwickler einer Integrationslösung, dafür Sorge zu tragen, dass nur sinnvolle Daten in die richtigen Tabellen und Felder übertragen werden; zudem muss die Datenkonsistenz gewahrt bleiben, indem Relationen zwischen Tabellen beachtet werden. Weiter oben haben Sie bereits dafür gesorgt, dass bestimmte Datenfelder auf deren Gültigkeit überprüft

werden. Als weitere Sicherheitsmaßnahme soll zusätzlich die Anzahl der importierten Datensätze eingegrenzt werden.

Die ehemalige GeneriCom operiert ausschließlich im nördlichen Raum, die Außendienstmitarbeiter bearbeiten ausschließlich Adressen der Region „N“. Um das Risiko fehlerhaft übernommener Daten einzuschränken, sollen daher nur Datensätze mit diesem Regionscode importiert werden. Ferner soll angezeigt werden, wie viele Datensätze importiert wurden.

Öffnen Sie dazu erneut Ihren Import-Dataport im Designmodus. Wählen Sie zunächst den Menüpunkt *Ansicht → C/AL Globals*. In diesem Fenster können Sie Variablen für die Programmierung festlegen. Übernehmen Sie folgende Variable:

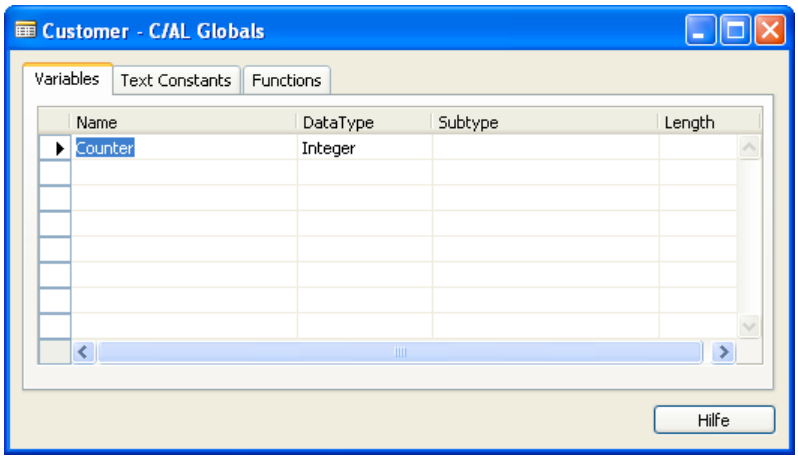

Abbildung 25: C/AL Globals

Öffnen Sie nun über das Menü *Ansicht → C/AL Code*. Um einen einfachen Zähler zu implementieren, übernehmen Sie folgenden Programmcode:

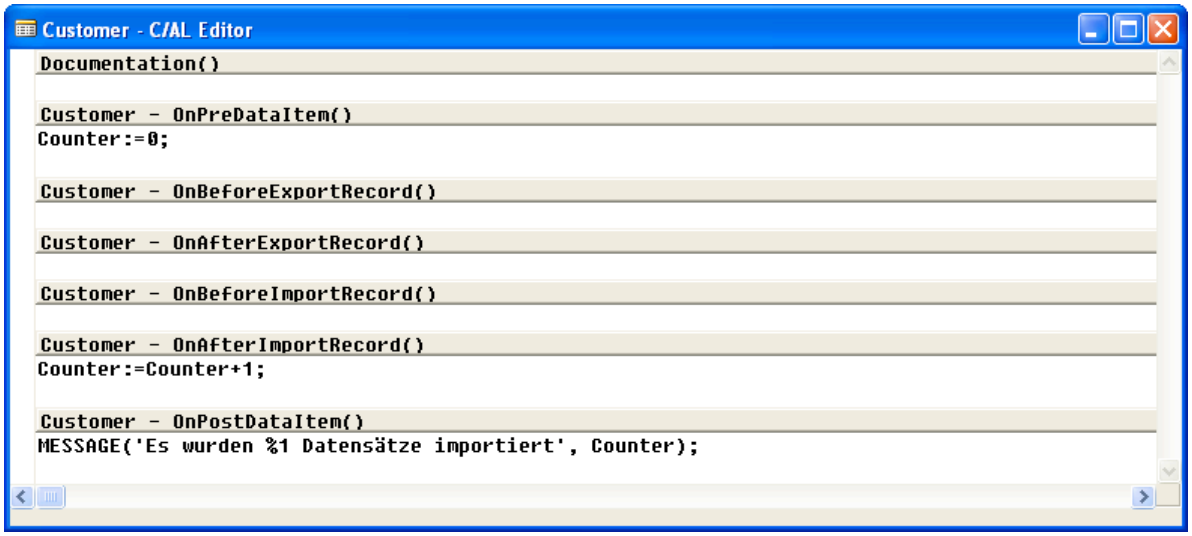

Abbildung 26: Programmcode für einen Zähler

Beachten Sie erneut die Semikola am Ende jeder Anweisung. Nutzen Sie abschließend die Möglichkeit, über die Funktion *Extras → Compile* bzw. durch die Taste F11 den Programmcode auf Fehler überprüfen zu lassen.

Die drei Zeilen sind selbsterklärend: Vor dem Start des Dataports (OnPreDataItem) wird der Zähler auf 0 gesetzt und nach jedem importierten Datensatz um 1 erhöht. Abschließend wird das Ergebnis in einer Message-Box ausgegeben.

Schließen Sie den Dataport und speichern Sie Ihre Änderungen. Prüfen Sie die neu programmierte Funktion, indem Sie den Dataport im Object Designer über den Button *Run* starten.

Kehren Sie anschließend in den Programmcode zurück und ergänzen Sie den Code gemäß nachfolgender Abbildung, um die Datenfilterung zu implementieren.

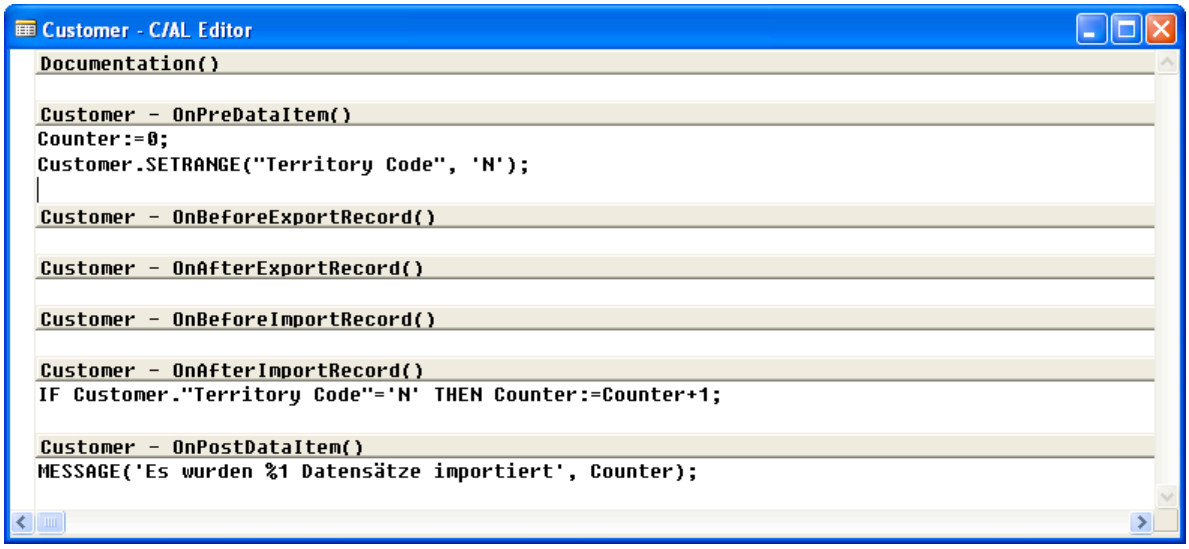

Abbildung 27: Programmcode zum Filtern der Datensätze nach Regionscode

Die Zeile

```
Customer.SETRANGE("Territory Code", 'N');
```

beinhaltet den eigentlichen Filter: Es werden nur Datensätze der Tabelle Customer bearbeitet, deren „Territory Code“ den Wert „N“ hat. Der Zähler wird dahingehend modifiziert, dass er nur dann zählt, wenn ein Datensatz dem obigen Kriterium entspricht:

```
IF Customer."Territory Code"='N' THEN
      Counter:=Counter+1;
```

Beachten Sie, dass entgegen vieler anderer Programmiersprachen innerhalb von C/AL Zeichenfolgen in einfache Hochkommata eingeschlossen werden. Anführungszeichen hingegen sind für Variablen reserviert, diese dürfen in Navision auch Leer- und Sonderzeichen enthalten.

Schließen Sie den Dataport und starten Sie ihn erneut. Die Anzahl der übernommenen Datensätze sollte nun weitaus geringer sein.

3. Export der Artikeldaten

Im vorhergehenden Kapitel der Fallstudie haben Sie Dataports als einfache Möglichkeit kennengelernt, Daten zwischen Programmen auszutauschen. Ebenso wurden allerdings die Schwächen einer solchen Lösung offenbar, die auf dem Austausch von kommagetrennten Textdateien basiert: Beide Programme müssen sich auf eine eindeutige und in beiden Systemen gleiche Syntax einigen, ohne dass diese im übertragenen Dokument transparent wird. Ein Fehler innerhalb des ausgetauschten Dokuments oder eine falsche Zuordnung von Datenfeldern durch das Zielsystem kann ausreichen, um die Konsistenz der Daten im ERP-System zu gefährden. Dieses Risiko ist um so größer, wenn mit einer Transaktion eine große Menge Daten an das ERP-System übertragen wird.

Um die Problematik der gemeinsamen Syntax zu reduzieren, bietet sich die Nutzung der Auszeichnungssprache XML an. Navision unterstützt diese seit der Version 4.0 durch XMLports. Ebenso wie Dataports ermöglichen diese den Datenim- und -export. Grundlegender Unterschied ist die Verwendung von Struktur und Auszeichnungselementen der XML. Durch die einfache, aber strenge Syntax können Fehlzuordnungen von Datenfeldern verhindert werden, der hierarchische Aufbau von XML ermöglicht darüber hinaus die Übertragung komplexerer Dokumente.

Access 2003 ermöglicht ebenfalls die Nutzung von XML-Dokumenten. Es liegt also nahe, die Extensible Markup Language als Austauschformat zwischen den beiden Programmen zu verwenden.

3.1. Erstellen eines XMLports

Zunächst ist es zweckmäßig, herauszufinden, in welchem Format WinCRM die Artikeldaten erwartet. Starten Sie dazu WinCRM. Rufen Sie den Menüpunkt *Artikelinformationen anzeigen* auf und sehen Sie sich ebenso im Datenbankfenster die Tabelle *Artikel* an. Die Tabelle sollte einen Beispieldatensatz enthalten.

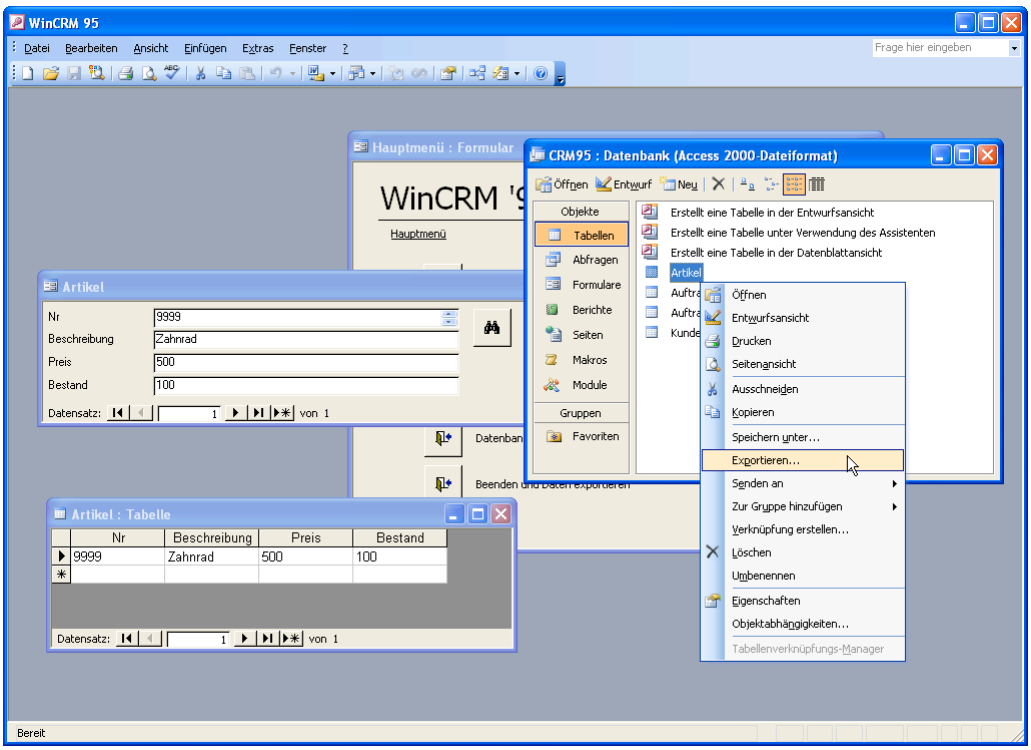

Abbildung 28: Export der Artikel-Tabelle in WinCRM

Schließen Sie das Tabellenfenster. Klicken Sie im Datenbankfenster mit der rechten Maustaste auf die Tabelle und wählen Sie *Exportieren*…. Setzen Sie im nachfolgenden Dateifenster den Dateityp auf XML und vergeben Sie einen Dateinamen, klicken Sie anschließend auf *Exportieren*. Ein weiteres Fenster öffnet sich und erfragt, welche Informationen exportiert werden sollen. Selektieren Sie hier *Daten (XML)*, damit nur das eigentliche XML-Dokument generiert wird.

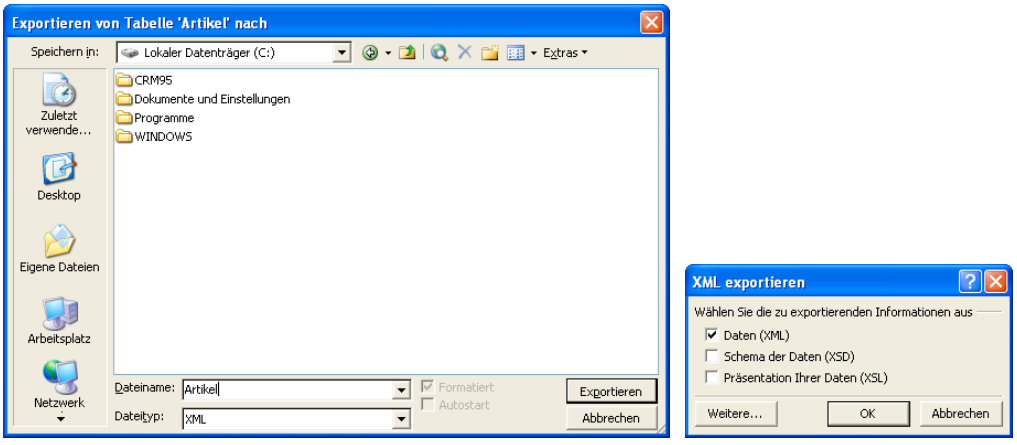

Abbildung 29: Sichern der Tabelle als XML-Dokument

Nun können Sie sich die XML-Datei in einem Browser oder Texteditor Ihrer Wahl ansehen. Die meisten Browser (z.B. Internet Explorer oder Firefox) interpretieren XML und zeigen den Inhalt in einer übersichtlichen Baumstruktur an.

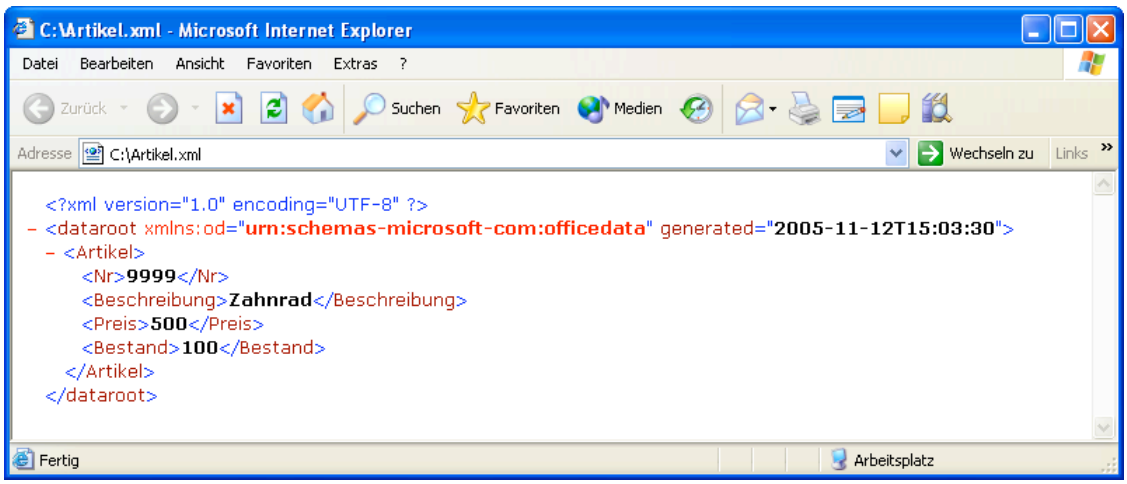

Abbildung 30: Exportierte Artikeldaten im XML-Format

Um die Artikeldaten aus Navision nach Access zu exportieren gilt es nun, Navision dazu zu bewegen, ein XML-Dokument zu generieren, welches dem gerade erstellten in Struktur und Syntax gleicht.

Starten Sie Navision und öffnen Sie den Object Designer. Klicken Sie auf der linken Seite des Fensters auf „XMLport“ und anschließend auf „New“.

Im nun erscheinenden XMLport Designer können Sie die Struktur und die Syntax einer XML-Datei modellieren. In die Spalte TagName können Sie alle im XML-Dokument vorhandenen Tags (und, sofern vorhanden, auch Attribute) eintragen. Nehmen Sie die Artikel-Datei als Vorlage und schreiben Sie zunächst alle dort eingetragenen Tags in jeweils eine Zeile des XMLports. Beachten Sie, dass Sie jeden (Start-)Tag nur einmal eintragen müssen; ebenso ist es wichtig, dass die Reihenfolge der Tags im XMLport der Reihenfolge in der Datei entspricht.

Anstelle des von Microsoft Office generierten „dataroot“ können Sie dem ersten Tag auch eine etwas aussagekräftigere Bezeichnung geben, z.B. „Artikeldaten“.

Nach diesem Schritt sollte Ihr XMLport so aussehen.

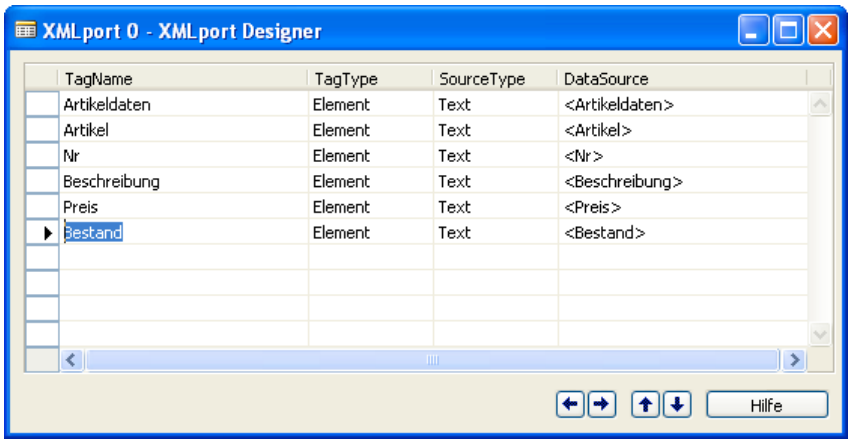

Abbildung 31: XMLport anlegen

Als Nächstes müssen Sie die Hierarchie der Elemente in der XML-Datei abbilden. Haben Sie diese mit dem Internet Explorer geöffnet, so ist dieser Schritt recht einfach, da der IE ineinander verschachtelte Tags eingerückt darstellt. Der XMLport Designer arbeitet nach dem gleichen Prinzip: Tags innerhalb eines übergeordneten Tags werden eingerückt. Dies können Sie mit den Pfeilen rechts unten im Designer bewerkstelligen, alternativ können Sie auch die Eigenschaften jedes Feldes aufrufen und dort die Eigenschaft „Indentation“ auf den gewünschten Wert setzen. Durch diese Darstellung können Sie logische und inhaltliche Zuordnungen der Daten vornehmen.

In der Artikel-Datei gibt es drei Ebenen. Dataroot bzw. Artikeldaten ist die höchste Ebene, da deren Tag alle weiteren Tags einschließt. Diese Ebene entspricht damit der gesamten Artikel-Tabelle. Rücken Sie daher alle Tags unterhalb „Artikeldaten“ um eine Stufe ein. Dazu wählen Sie eine Zeile aus und klicken jeweils einmal auf den Pfeil nach rechts.

Die dritte Ebene wird durch das Artikel-Tag eingeschlossen, sie enthält die Tags Nr, Beschreibung, Preis und Bestand. „Artikel“ ist somit ein Datensatz der Access-Tabelle, die untergeordneten Tags entsprechen den einzelnen Feldern. Rücken Sie alle Tags unterhalb „Artikel“ um einen weiteren Schritt ein.

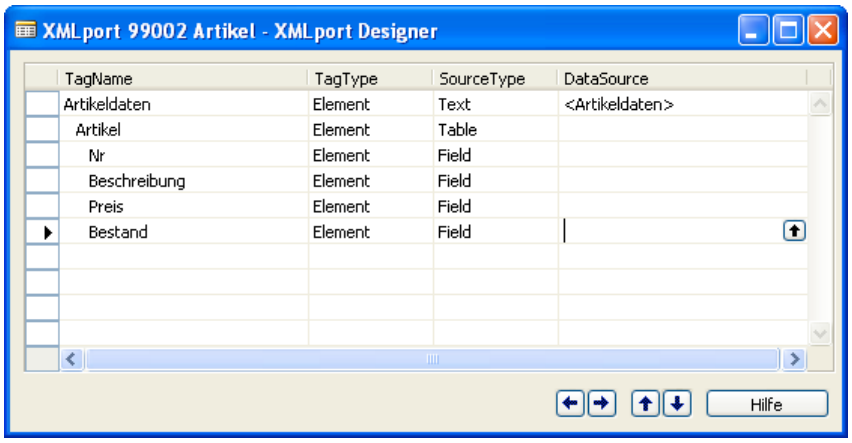

Abbildung 32: Hierarchie der XML-Elemente festlegen

Nun haben Sie die Struktur des XML-Dokuments in Navision abgebildet und können sich den weiteren Spalten des XMLport Designers widmen. TagType unterscheidet „Element“ und „Attribut“. Ein Element ist ein Datum, welches innerhalb von XML-Tags eingeschlossen ist (z.B. `<Artikelname>Zahnrad</Artikelname>`). Attribute dagegen sind beschreibende Eigenschaften innerhalb von XML-Tags (z.B. `<Artikel updated=“01.08.2005“>`). In der Artikel-Datei gibt es keine Attribute, sie können also den voreingestellten Typ *Element* in allen Zeilen belassen.

Die beiden nachfolgenden Spalten im Designer bestimmen, wie Navision den Inhalt der Tags interpretiert. Da Sie sich bereits mit der Struktur der Access-Tabelle und des

exportierten XML-Dokuments auseinandergesetzt haben, sind die Werte des Felds *SourceType* nahezu selbsterklärend. Der Wert „Table“ bedeutet, dass ein Tag einem Datensatz in einer Tabelle von Navision entspricht, bzw. einen solchen initialisiert. Ein solches Tag schließt in der Regel weitere Tags ein, die Datenfelder für die jeweilige Tabelle enthalten und damit als „Field“ bezeichnet werden können. Kann oder soll der Inhalt eines Tags nicht direkt der Datenbank zugewiesen werden, so kann es als „Text“ deklariert und somit in einer Variable zwischengespeichert werden.

Das Element „Artikeldaten“ enthält keine sinnvollen Daten, dessen SourceType können Sie daher den Wert *Text* zuweisen. Das Tag „Artikel“ hingegen enthält jeweils genau einen Datensatz für eine Artikeltabelle, der SourceType ist entsprechend *Table*. Die von dem Tag „Artikel“ eingeschlossenen Tags entsprechen Feldern der Tabelle, ihr Wert ist also stets *Field*.

Der Wert der letzten Spalte DataSource ist vom gewählten Feldtyp abhängig. Ist der SourceType ein Text, so können Sie hier einen Variablennamen angeben, unter dem der Inhalt des Tags zwischengespeichert wird. Voreingestellter Name ist der Name des Tags, so dass sie für die erste Zeile „Artikeldaten“ nichts ändern müssen. Verweist der SourceType auf eine Tabelle, so können Sie durch einen Klick auf DataSource eine Liste mit allen verfügbaren Tabellen abrufen, um dem Tag eine Tabelle zuweisen zu können. Im Falle von „Artikel“ ist dies Tabelle 27 (Item). Haben Sie diese zugeordnet, so können Sie für die untergeordneten Tags die entsprechenden Datenfelder der Tabelle wählen. Ein Klick auf DataSource öffnet dann eine Liste mit allen Datenfeldern innerhalb der Tabelle. Wählen Sie für Nr, Beschreibung, Preis und Bestand die korrespondierenden Felder in der Item-Tabelle von Navision aus. Orientieren Sie sich dabei an untenstehender Abbildung, die den fertigen XMLport zeigt.

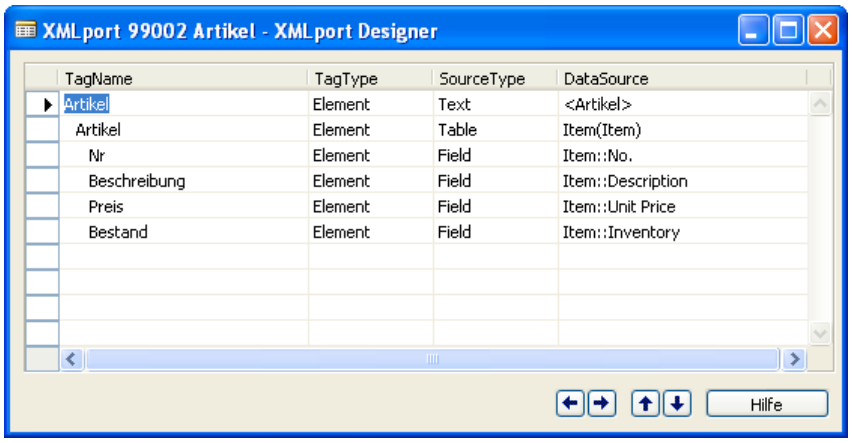

Abbildung 33: Fertiggestellter XMLport

Schließen Sie den XMLport und vergeben Sie eine ID und einen aussagekräftigen Namen.

3.2. Aufruf des XMLports über C/AL

Anders als Dataports werden XMLports nicht direkt gestartet, sondern über Programmcode aufgerufen. Um den XMLport zu testen, legen Sie ein neues Codeunit im Object Designer an.

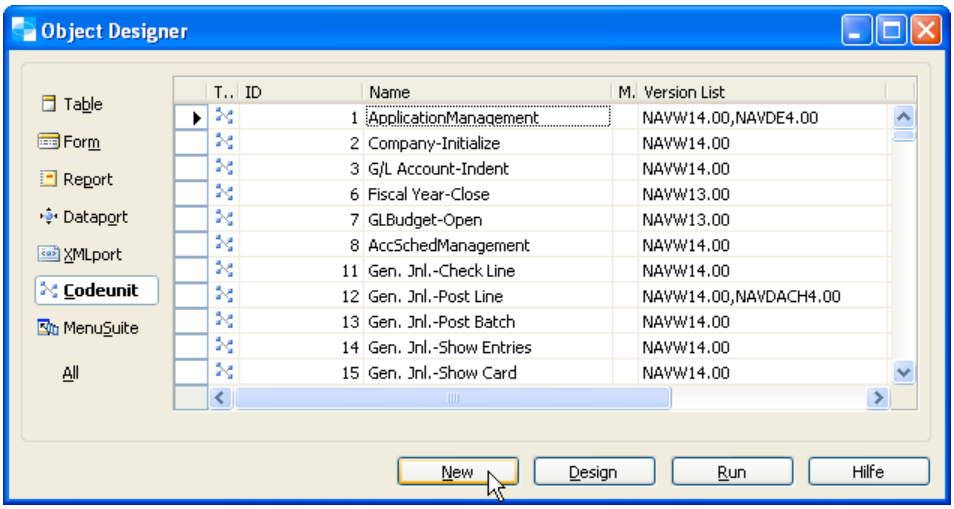

Abbildung 34: Codeunits im Object Designer

Für den Export soll eine neue Datei erstellt werden; diese wird im Anschluss mit Daten aus dem XMLport gefüllt. Legen Sie dazu zwei Variablen an. Im Gegensatz zu den meisten anderen Programmiersprachen werden in C/AL Variablen nicht direkt im Quellcode deklariert, sondern in einer separaten Ansicht. Wählen Sie den Menüpunkt *Ansicht → C/AL Globals* und tragen Sie die beiden untenstehenden Variablen in das Register „Variables“ ein.

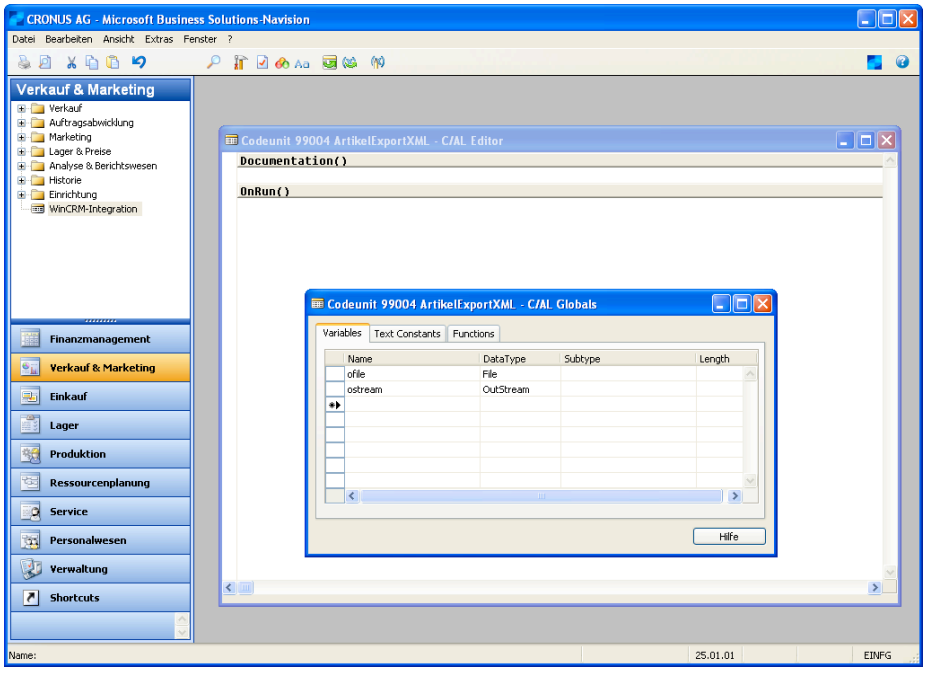

Abbildung 35: Variablen im neuen Codeunit

Schließen Sie das Fenster. Jetzt können Sie die Variablen verwenden. Übernehmen Sie nun den unten abgebildeten Programmcode.

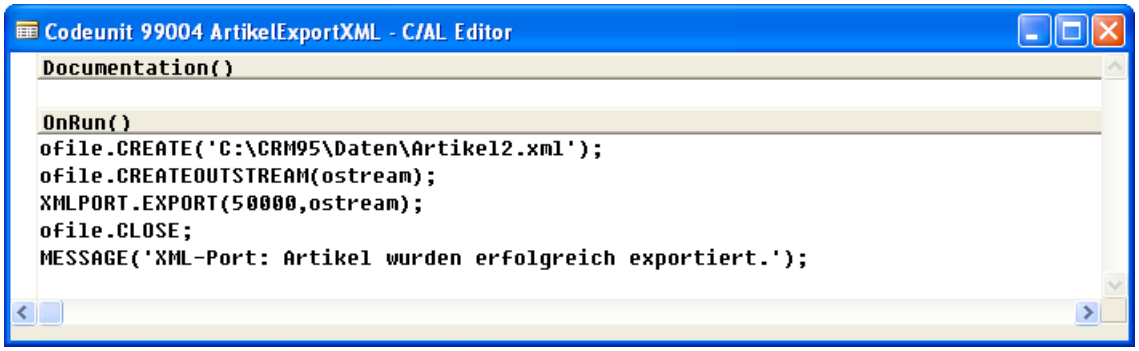

Abbildung 36: Programmcode zum Aufruf des Artikel-Export-XMLports

Das Programm erstellt eine neue Datei (eine eventuell bestehende Datei wird gelöscht!), streamt den vom XMLport generierten Inhalt in die Datei und schließt diese. Abschließend weist eine Messagebox den User darauf hin, dass der Export erfolgreich war.

Beachten Sie wiederum Semikola am Ende jeder Anweisung und machen Sie von der Möglichkeit Gebrauch, durch Kompilierung (Taste F11) den Code auf Fehler zu überprüfen.

Schließen Sie das Codeunit und vergeben Sie ID und Dateinamen. Anschließend können Sie Ihr Programm über den Button *Run* im Object Designer starten.

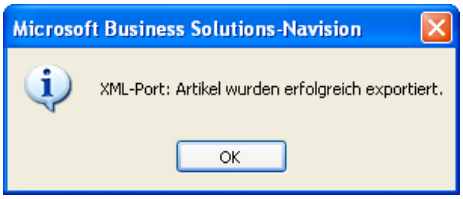

Abbildung 37: Messagebox - erfolgreicher Export

3.3. Test des XMLports

War der Export erfolgreich, so können Sie sich die neu erstellte Datei im Browser ansehen. Vergleichen Sie Syntax und Struktur mit der eingangs erstellten Vorlage.

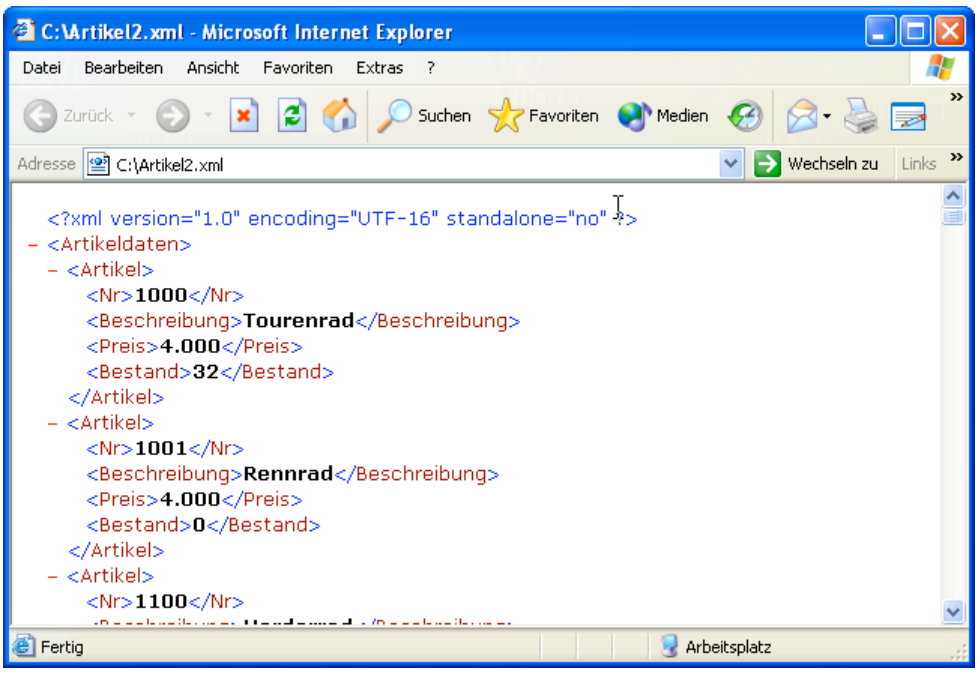

Abbildung 38: Aus Navision exportierte Artikel im XML-Format

Wenn Sie sich überzeugt haben, dass beide Dateien den gleichen Aufbau besitzen, können Sie den Import nach Access testen. Benennen Sie die gerade erstellte `Artikel2.xml` in `Artikel.xml` um und kopieren Sie diese nach `C:\CRM95\Daten\`. WinCRM enthält Programmcode, der beim Start des Programms nach der Datei `C:\CRM95\Daten\Artikel.xml` sucht. Wenn diese vorhanden ist, wird sie importiert und ersetzt die bisherige Tabelle „Artikel“. Starten Sie nun WinCRM und begutachten Sie das Ergebnis Ihrer Arbeit, indem Sie das Formular *Artikel bearbeiten* bzw. die Tabelle „Artikel“ öffnen.

3.4. Einbindung in die Benutzeroberfläche

Um den Benutzern den Zugang zu dieser Funktion ebenfalls so einfach wie möglich zu machen, fügen Sie diese zu dem im ersten Teil der Studie erstellten Formular hinzu.

Zuvor sollten Sie allerdings noch das oben erstellte Codeunit anpassen. Kehren Sie zur Code-Ansicht des XMLports zurück und nehmen Sie folgende Änderungen vor: Der Name der zu exportierenden Datei soll nun auf `C:\CRM95\Daten\Artikel.xml` verweisen. Ferner soll die Messagebox nicht mehr angezeigt werden. Löschen Sie die entsprechende Programmzeile oder kommentieren Sie diese durch zwei vorangestellte Schrägstriche aus.

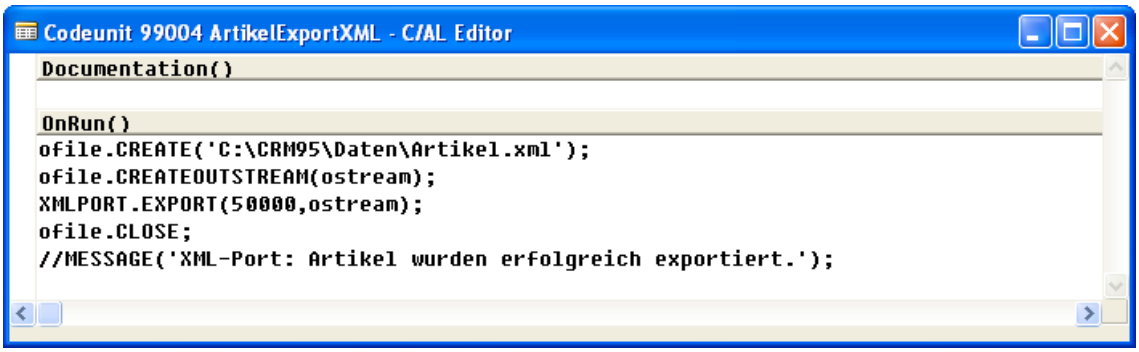

Abbildung 39: Überarbeiteter Aufruf des Artikel-Export-XMLports

Rufen Sie anschließend Ihr Import/Export-Formular über den Object Designer auf und fügen Sie eine weitere Schaltfläche ein. Um Tipp- und Klickarbeit zu sparen, können Sie auch mit Copy & Paste arbeiten. Vergeben Sie erneut sinnvolle Werte für die Eigenschaften „Name“ und „Caption“ der neuen Schaltfläche. Markieren Sie diese anschließend und rufen Sie den C/AL-Editor auf (*Ansicht → C/AL Editor* oder Taste F9). Haben Sie Copy & Paste benutzt, wurde der Programmcode des kopierten Buttons übernommen; diesen können Sie löschen. Der Code für den neuen Button soll das zuvor erstellte Codeunit aufrufen. Er lautet

```
CODEUNIT.RUN(ID ihres Codeunits);
```

Beenden Sie anschließend den Designmodus, indem Sie Ihr Formular schließen.

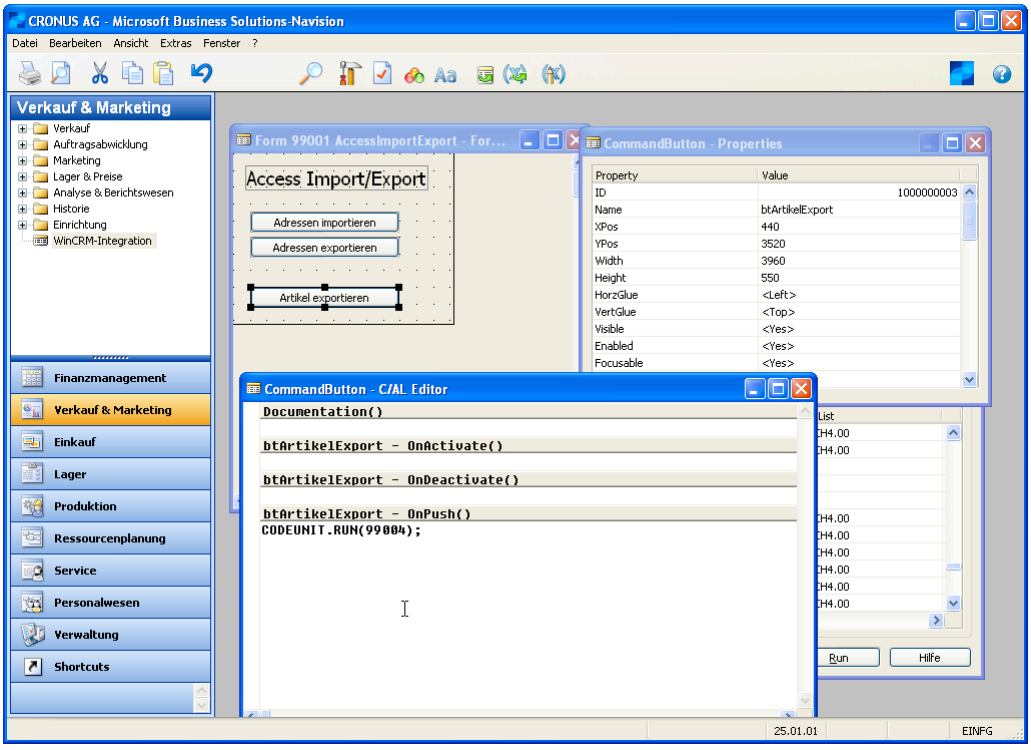

Abbildung 40: Einbindung in das Import/Export-Formular

Um sich abschließend von der Funktion des Artikelexports zu überzeugen, rufen Sie in Navision die Artikelkarte auf. Klicken Sie dazu im Seitenmenü auf *Verkauf & Marketing*

und anschließend in der oberen Baumstruktur des Menüs auf *Lager & Preise → Artikel*. Ändern Sie einige Artikeldaten oder fügen Sie einen neuen Artikel hinzu. Rufen Sie danach Ihr Formular über *WinCRM-Integration* auf und klicken Sie den neuen Button. Starten Sie anschließend Access und prüfen Sie, ob Ihre Änderungen importiert wurden.

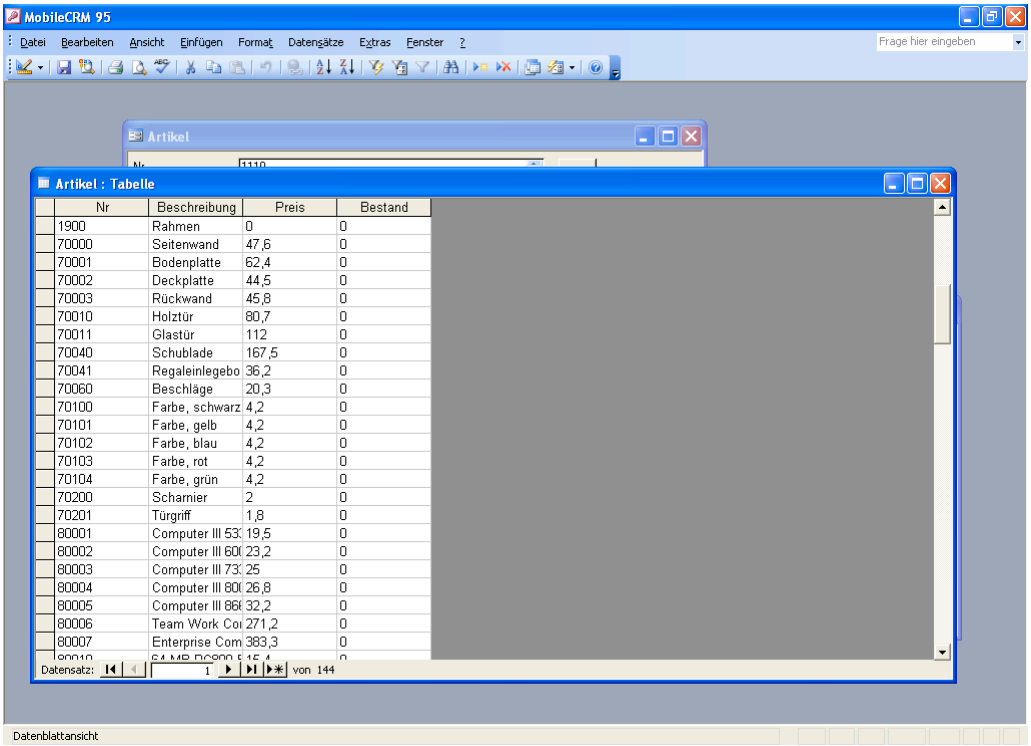

Abbildung 41: Erfolgreich exportierte Artikeldaten

Sie haben eine Integrationslösung auf Basis von XML entwickelt und sich nebenbei die Grundlagen für die Verwendung von XMLports in Navision angeeignet.

4. Import von Aufträgen

In der vorhergehenden Aufgabe haben Sie festgestellt, dass die Verwendung von Standards im Allgemeinen und XML im Besonderen zahlreiche Vorteile bei der Integration von Anwendungssystemen bietet. Zum einen liefert XML die Bezeichnung der übertragenen Daten mit und erleichtert damit sowohl Anwendern als auch Programmen die Interpretation des Inhalts. Neben der hohen Transparenz der Inhalte bildet XML darüber hinaus die Hierarchie von übertragenen Daten ab. Für den Datenaustausch zwischen (unterschiedlichen) Anwendungssystemen bedeutet dies, neben einfachen Datensätzen auch Geschäftsdokumente von hoher Komplexität übertragen zu können, die sich zudem flexibel auswerten lassen.

WinCRM ermöglicht dem Außendienst die Erfassung von Aufträgen. Eine Integration mit dem bisherigen ERP-System wurde jedoch nicht realisiert. Die optimale Lösung wäre es, den Inhalt neu erfasster Aufträge aus WinCRM direkt nach Navision zu übernehmen, so dass diese dort geprüft und gebucht werden können. Da WinCRM den Export neuer Aufträge in ein XML-Dokument ermöglicht, sind Sie optimistisch, diese Lösung mit Ihren bisher erlangten Kenntnissen umsetzen zu können.

Im dritten Kapitel fanden Sie ideale Bedingungen für den Einsatz von XMLports vor: Auf Basis einer Vorlage haben Sie einen XMLport für den Export von Artikeldaten erstellt. In diesem Teil der Fallstudie entwickeln Sie einen XMLport für den Import von Daten. Nun tauchen einige Herausforderungen auf, die sie auch unter „realen“ Bedingungen antreffen würden:

- Sie erarbeiten einen XMLport für ein komplexes Geschäftsdokument, dessen Inhalt sich auf mehrere Tabellen in Navision bezieht, die in Relation zueinander stehen
- Das XML-Dokument, welches die Aufträge beinhaltet, enthält überflüssige Datensätze, die von Navision nicht bearbeitet werden
- umgekehrt fehlen im Quelldokument Datenfelder, die von Navision jedoch zwingend benötigt werden
- Im Quelldokument liegen einige Datenfelder in einer anderen Reihenfolge vor, als Navision dies erwartet

Das Vorgehen in diesem Kapitel kann daher als Vorlage für die Entwicklung eines komplexeren XMLports dienen.

4.1. Export aus Access

Schauen Sie sich zunächst an, wie die Auftragserfassung in WinCRM realisiert wurde. Öffnen Sie die Access-Datei und rufen Sie das Formular zur Erfassung von Aufträgen auf.

Da WinCRM nun bereits über die aktuellen Kunden- und Artikelstammdaten verfügt, können sie einen Beispielauftrag eingeben. Übernehmen Sie bitte den folgenden Auftrag der Blütenhaus GmbH – eine Auftragsnummer wird automatisch vergeben.

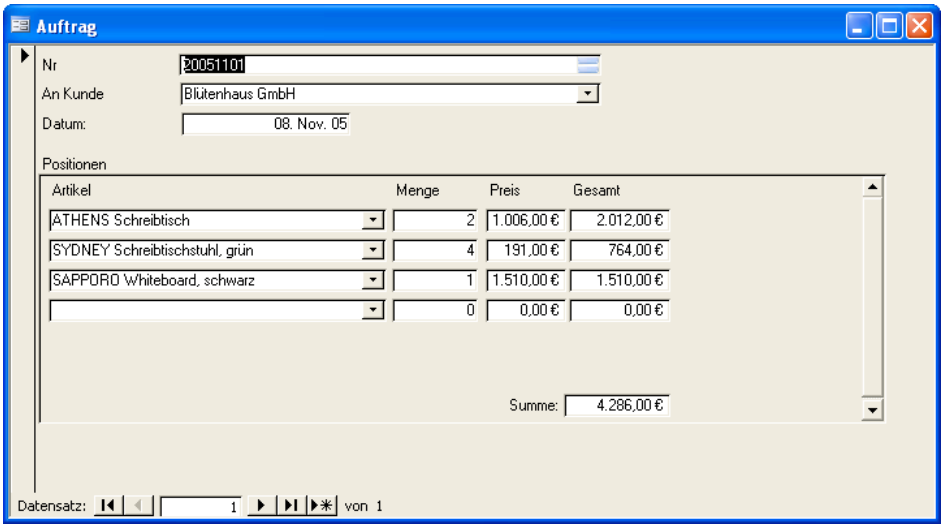

Abbildung 42: Auftragserfassung in WinCRM

Schließen Sie das Formular, begeben Sie sich in das Datenbankfenster und sehen Sie sich die korrespondierenden Tabellen an. Sie sehen, dass im Hauptteil des Formulars die Daten des Artikelkopfs erfasst und in der Tabelle Auftrag_Kopf gespeichert werden: Auftragsnummer, Kunde und Datum. Das Unterformular des Auftrags erfasst die einzelnen Artikel, diese werden in der Tabelle Auftrag_Zeile festgehalten. Das Feld DokumentNr in der Tabelle Auftrag_Zeile verweist dabei auf die Auftragsnummer in Auftrag_Kopf.

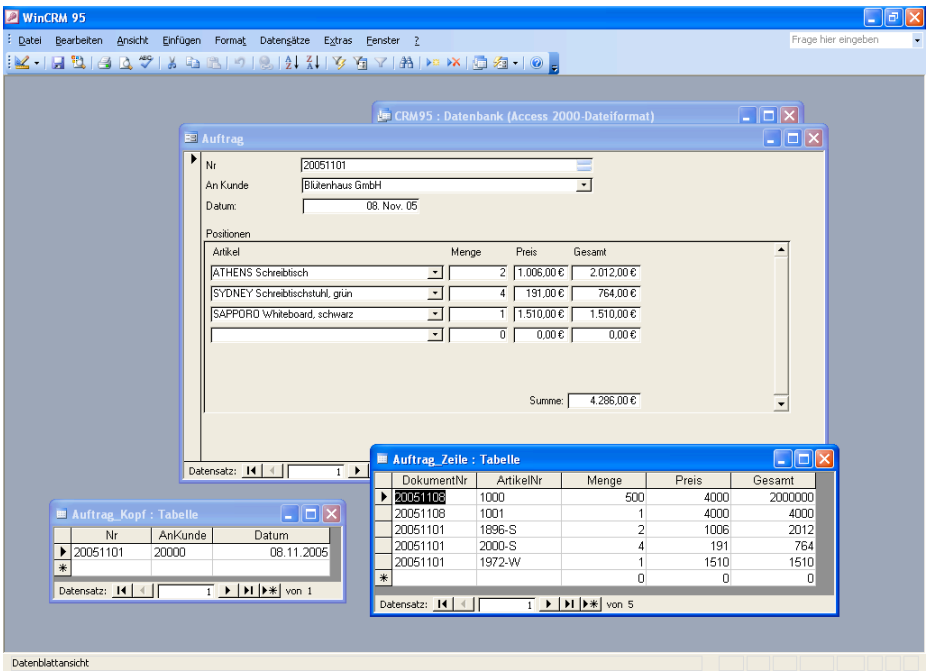

Abbildung 43: Tabellen zur Auftragserfassung in WinCRM

Beenden Sie WinCRM über das Hauptmenü mittels „Beenden und Daten exportieren“. Mit diesem Button wird ein XML-Dokument generiert, welches alle in WinCRM erfassten Aufträge enthält. Sehen Sie sich die Datei `C:\CRM95\Daten\Aufträge.xml` in einem Browser an; obige Datenbankrelation spiegelt sich hier wieder. Wie im vorhergehenden Kapitel können Sie die strukturierte Ansicht des XML-Dokuments als Vorlage für einen neuen XMLport verwenden.

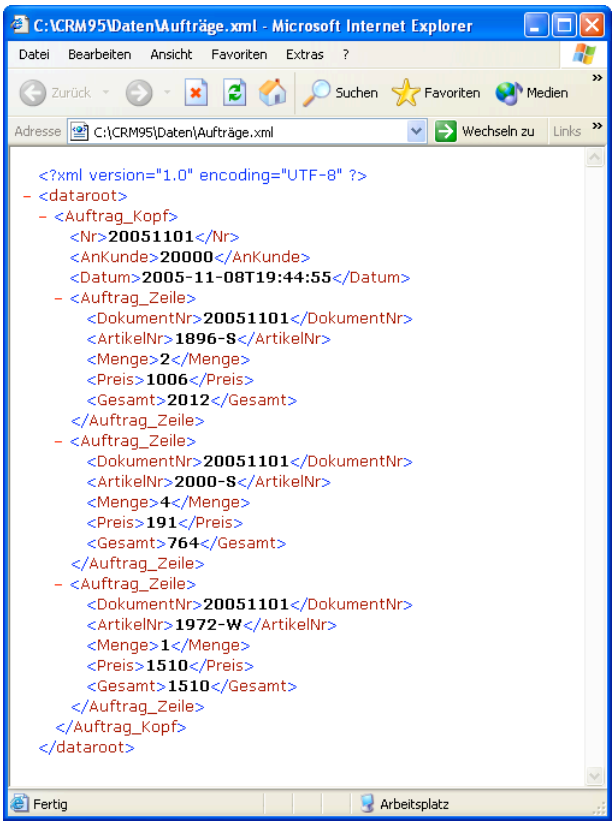

Abbildung 44: Ein WinCRM-Auftrag als XML-Dokument

4.2. Erstellen des XMLports

4.2.1. Abbildung der XML-Datei

Sofern noch nicht geschehen, starten Sie Navision. Öffnen Sie den Object Designer und legen Sie einen neuen XMLport an. Übernehmen Sie nun alle Tags der XML-Datei in jeweils ein „TagName“-Feld des XMLport Designers. Nehmen Sie anschließend die Einrückungen vor.

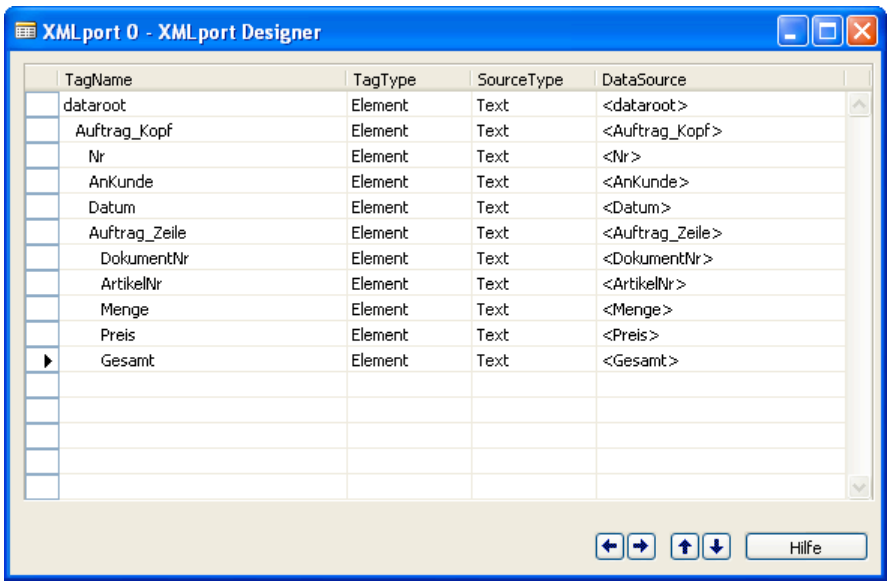

Abbildung 45: XMLport zum Auftragsimport - Felder und Hierarchie

Rufen Sie die allgemeinen Eigenschaften des XMLports auf. Setzen Sie vorsichtshalber „Direction“ auf *Import*, damit der Port nicht dazu verwendet werden kann, alle Aufträge des Unternehmens zu exportieren.

Wichtig für das Funktionieren des XMLports ist die Eigenschaft DefaultFieldsValidation. Wie die CallFieldValidate-Eigenschaft der Dataports entscheidet diese, ob für einzelne Tabellenfelder hinterlegter Programmcode aufgerufen wird, wenn der XMLport Daten in die Felder schreibt. „Yes“ bedeutet, dass der „OnValidate“-Trigger des jeweiligen Felds aufgerufen wird, bei „No“ wird der Trigger übergangen. Wählen Sie letztere Option, um zu vermeiden, dass es zu Fehlern wegen unvollständiger Daten kommt. Trigger für wichtige Felder werden Sie später einzeln oder innerhalb von Programmcode ausführen.

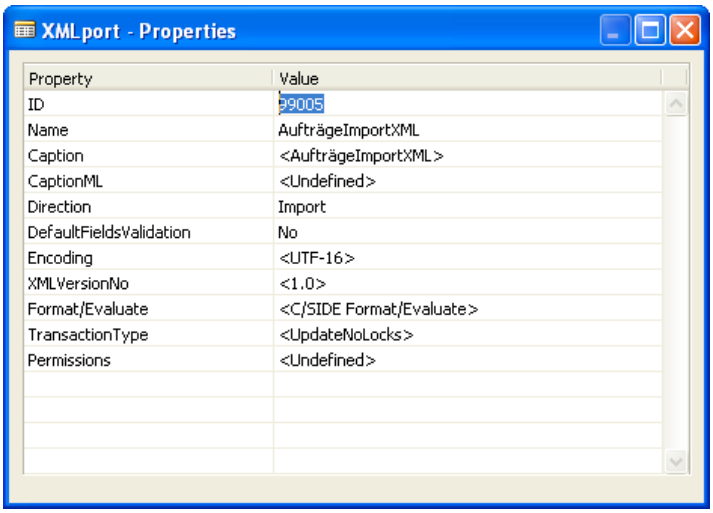

Abbildung 46: Eigenschaften des XMLport

Speichern und schließen Sie den XMLport vorerst. Vergeben Sie wie gehabt ID und Dateinamen. Um herauszufinden, in welche Tabellen der Inhalt der XML-Datei

gespeichert werden muss, schauen Sie sich an, wie Aufträge in Navision erfasst werden. Das entsprechende Formular finden Sie in der seitlichen Menüleiste unter *Verkauf & Marketing → Auftragsabwicklung → Aufträge*. Obgleich die Auftragserfassung in Navision um ein Vielfaches komplexer ist, gleicht sich der grundlegende Aufbau mit Verkaufskopf und einzelnen Zeilen. Wählen Sie *Extras → Designer* (Strg + F2), um das aktuelle Formular zu editieren. Dies entspricht der Funktion *Design* im Object Designer.

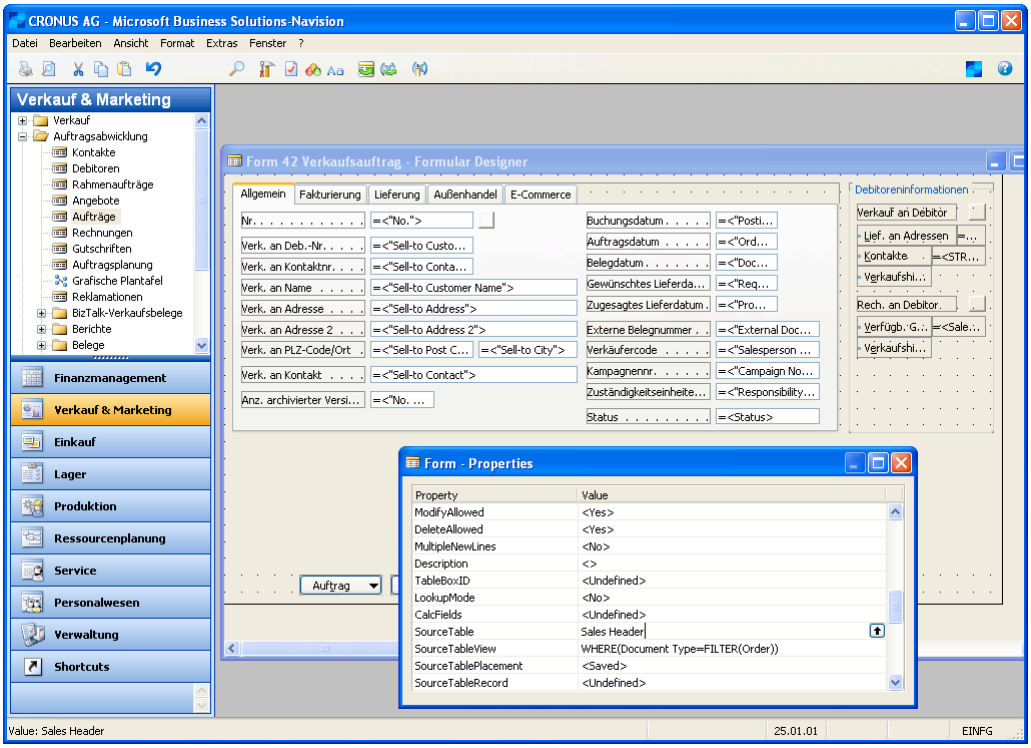

Abbildung 47: Auftragserfassung in Navision

Über die Eigenschaften des Formulars (SourceTable) erfahren Sie, dass die Daten aus der Tabelle „Sales Header“ stammen. Überprüfen Sie dies, indem Sie im Object Designer *Tables* anwählen und die Tabelle suchen. Geben Sie dazu *Sales Header* ein; nach Eingabe des ersten Buchstabens erscheint ein Suchfenster.

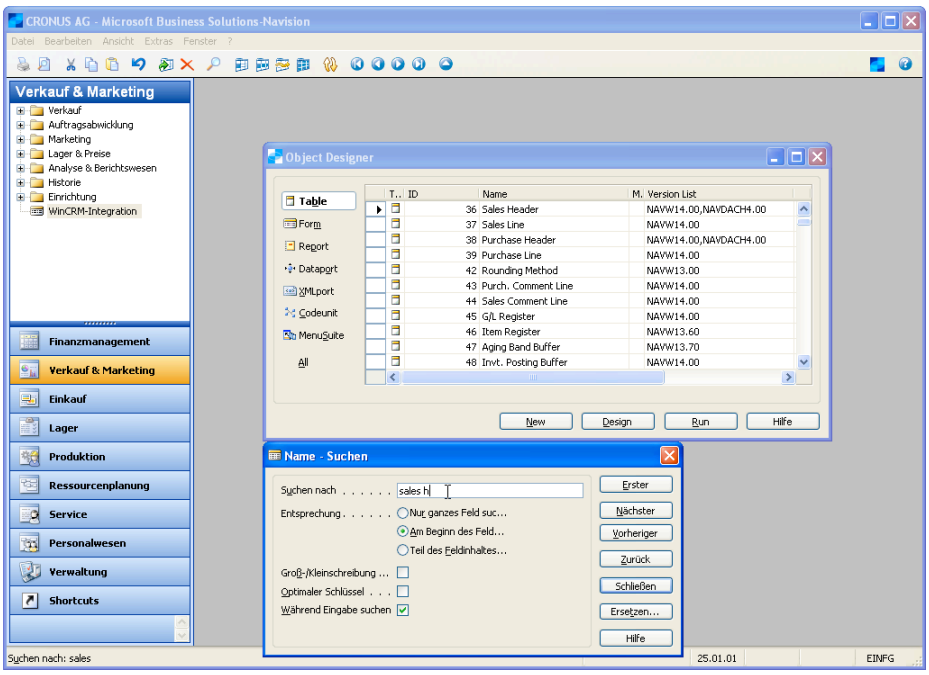

Abbildung 48: Auffinden der Tabellen zur Auftragsspeicherung

Sie gelangen zu Tabelle 36 „Sales Header“. Die dazugehörigen Auftragszeilen finden sich in der unmittelbar darauffolgenden Tabelle 37 „Sales Line“. Sehen Sie sich Aufbau und Inhalt beider Tabellen an, sowohl in der Entwurfs- (*Design*) als auch in der Tabellenansicht (*Run*). Auf diese Weise können Sie abschätzen, mit welchen Feldern sie die Elemente des XMLports verknüpfen können. Beachten Sie außerdem, dass dank der später ausgeführten Validierungsfunktionen die wenigen Datenfelder aus WinCRM ausreichen, um innerhalb der komplexen Tabellen in Navision einen vollständigen Auftrag anzulegen.

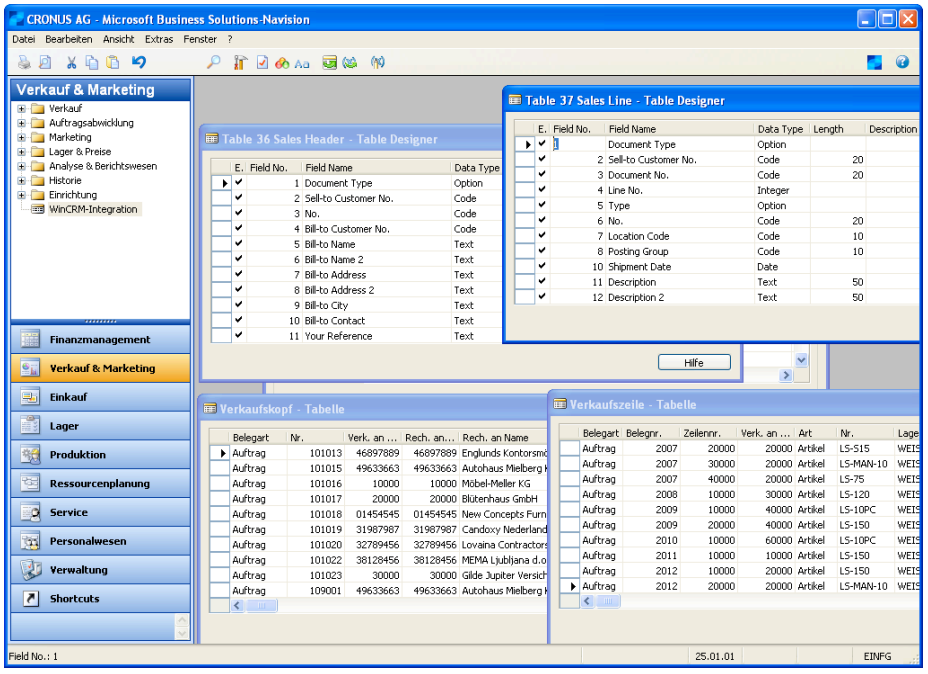

Abbildung 49: Tabellen zur Auftragsspeicherung - Felder und Inhalte

Schließen Sie die Tabellen und kehren Sie zu Ihrem XMLport zurück. Setzen Sie die Eigenschaft SourceType der Tags „Auftrag_Kopf“ und „Auftrag_Zeile“ auf Table und wählen Sie als DataSource die jeweils passende der gerade betrachteten Tabellen aus. Um später Tipparbeit zu sparen, rufen Sie außerdem die Eigenschaften der jeweiligen Tags aus und ersetzen den voreingestellten VariableName durch *SH* bzw. *SL*. Unter diesen Bezeichnungen lassen sich die Tabellen im Programmcode ansprechen.

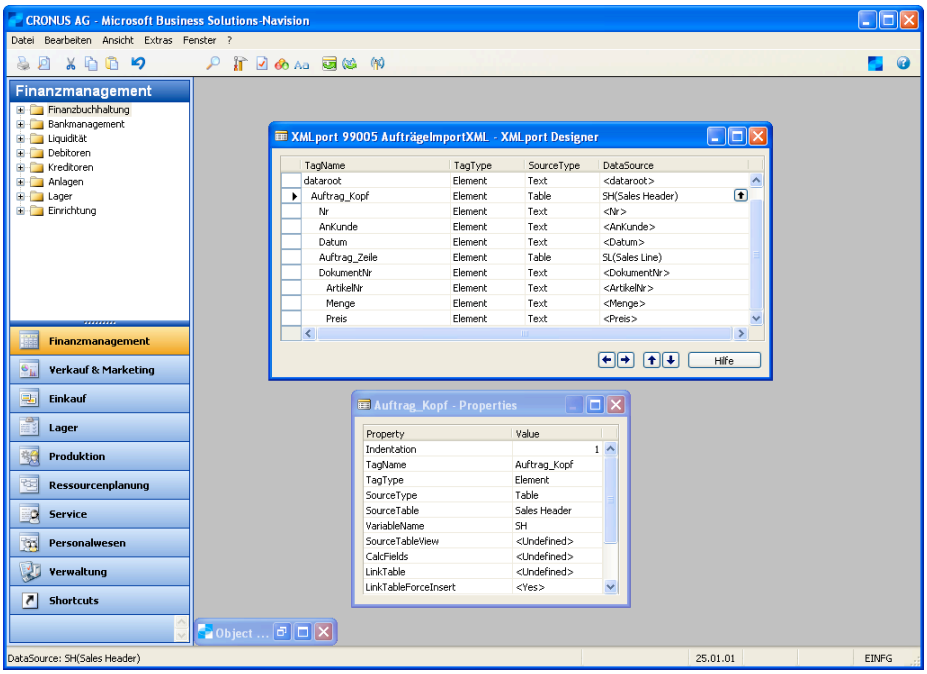

Abbildung 50: Zuweisen der Tabellen im XMLport

Öffnen Sie erneut die Eigenschaften von Auftrag_Zeile. Setzen Sie die Eigenschaft LinkTable auf SH und LinkTableForceInsert auf Yes. Öffnen Sie nun den Dialog in der darunter liegenden Eigenschaft „LinkFields“ und verknüpfen Sie, wie abgebildet, das Feld „Document No.“ (aus Sales Line) mit dem Feld „No.“ (aus Sales Header).

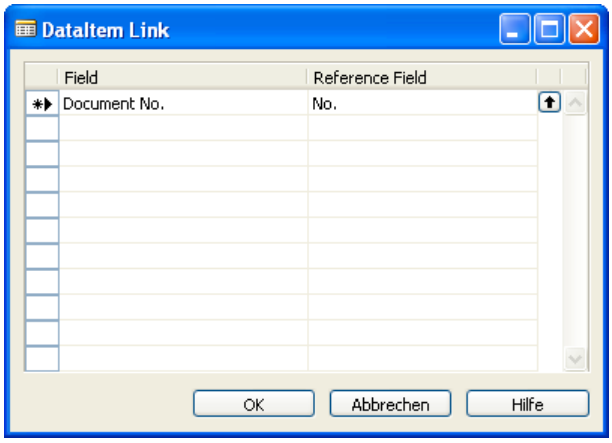

Abbildung 51: Dataltem Link

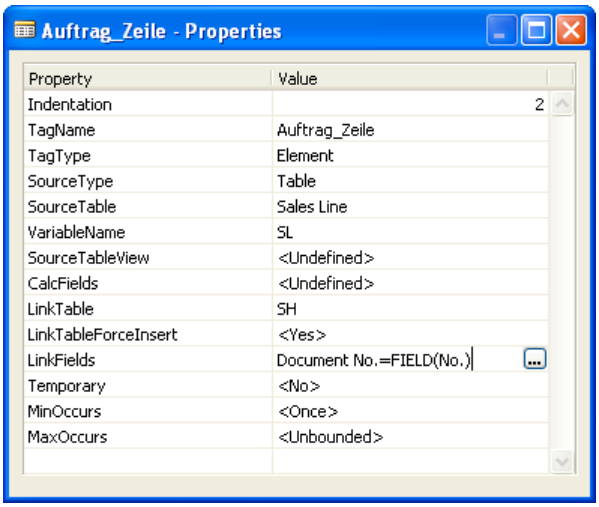

Abbildung 52: Abbilden der Tabellenrelation

Damit haben Sie die Relation zwischen den beiden Tabellen abgebildet und weisen Navision an, zuerst den Auftragskopf zu bearbeiten und anschließend die Auftragszeilen, wobei die Auftragsnummer in letztere übernommen wird.

4.2.2. Mapping der Tabellenfelder

Nachdem Sie die Tabellen zugeordnet haben, können Sie die Datenfelder zuweisen, die sich ohne Probleme übernehmen lassen.

Auftrags- bzw. Dokumentnummern sind Primärschlüssel und werden von Navision automatisch vergeben, sobald ein neuer Auftrag erstellt wird. Die Übernahme der Auftragsnummer aus Access würde langfristig zu Kollisionen führen; also verknüpfen Sie dieses Feld nicht.

Der Tag „AnKunde“ ist unproblematisch: Er entspricht dem „Sell-to Customer No.“-Feld der Tabelle „Sales Header“. Setzen Sie den SourceType auf *Field* und wählen Sie das Feld als DataSource aus. Rufen Sie zudem die Eigenschaften des Feldes auf und setzen Sie FieldValidate auf *Yes*. Wird nun ein Wert in dieses Feld geschrieben, wird, wie oben beschrieben, Programmcode ausgeführt. Dieser sorgt u.a. dafür, dass die Kundennummer in der Tabelle „Customer“ nachgeschlagen wird und die Auftragstabelle um fehlende Kundendaten ergänzt wird.

Das Auftragsdatum speichert Access in einem Format, welches von Navision nicht direkt verarbeitet werden kann und daher mit etwas Programmcode konvertiert werden muss. Überschreiben Sie den voreingestellten Variablennamen in der Spalte *DataSource* mit „TmpDatum“.

Bei den Auftragszeilen können Sie die DokumentNr ebenfalls ignorieren, da sie identisch mit der von Navision vergebenen Auftragsnummer sein muss. Dies haben Sie bereits über die „LinkTables“ Eigenschaft gewährleistet.

ArtikelNr und Menge können Sie mit Feldern der Tabelle Sales Line verknüpfen, die korrespondierenden Felder sind „No.“ und „Quantity“.

Artikelpreis und Gesamtpreis können ignoriert werden, da Navision diese selbst abrufen bzw. berechnen kann.

Nach diesen Einstellungen sollte Ihr XMLport folgendermaßen aussehen:

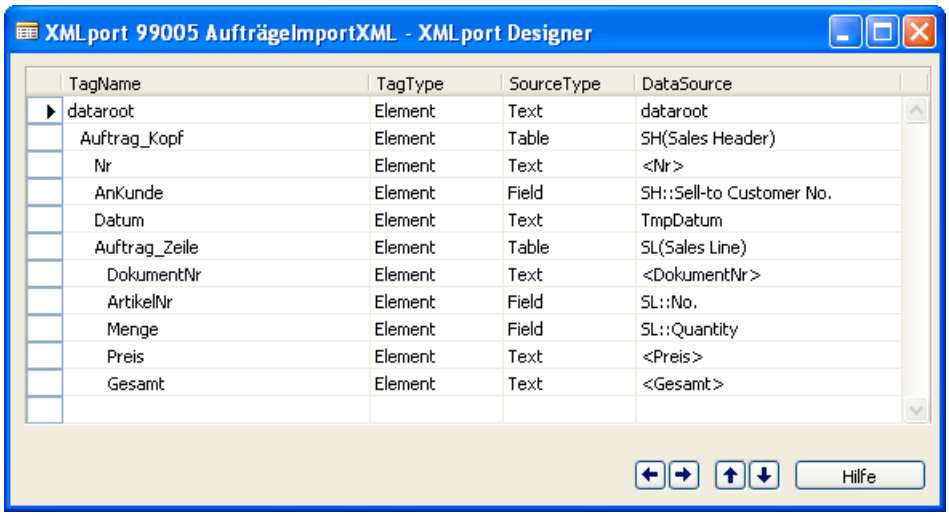

Abbildung 53: SourceType und DataSource im XMLport

4.2.3. Coding weiterer Felder

Jetzt können Sie sich den noch fehlenden Feldern zuwenden. Neben oben genannten Feldern gilt es außerdem, einige Pflichtfelder manuell zu ergänzen. Vergleichen Sie die folgenden Schritte bei Bedarf mit den verschiedenen Perspektiven der Tabellen „Sales Header“ und „Sales Line“.

Öffnen Sie vorab den Dialog C/AL Globals (Menü *Extras*) und tragen Sie die folgenden Variablen ein; diese werden Sie im Programmcode benötigen:

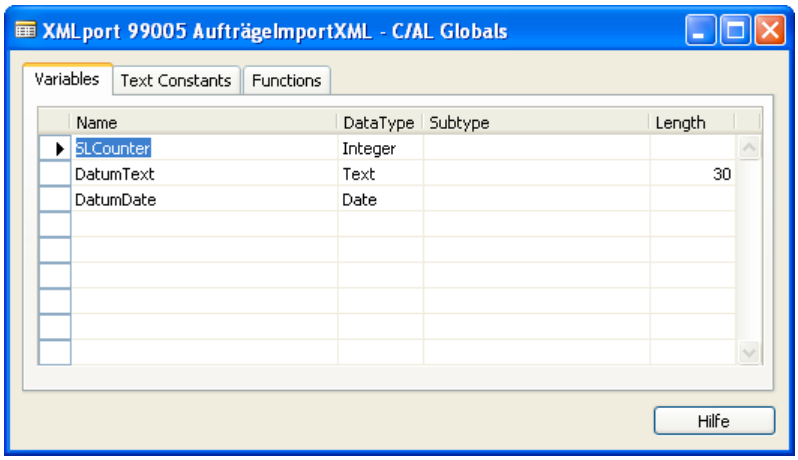

Abbildung 54: C/AL Globals

Öffnen Sie anschließend den C/AL Editor (*Ansicht → C/AL Code* bzw. F9). Die günstigste Gelegenheit, fehlende Daten einzufügen, ist, nachdem ein Datensatz initialisiert wurde. Dieses Ereignis wird durch den OnAfterInitRecord()-Trigger definiert.

Um Felder für den Auftragskopf zu ergänzen, wenden Sie sich zunächst dem Block

```
Auftrag_Kopf - Import::OnAfterInitRecord()
```

zu. Werfen Sie einen Blick auf die Tabelle „Sales Header“. Erstes Feld der Tabelle ist die Angabe einer Belegart. Öffnen Sie die Tabelle im Designmodus.

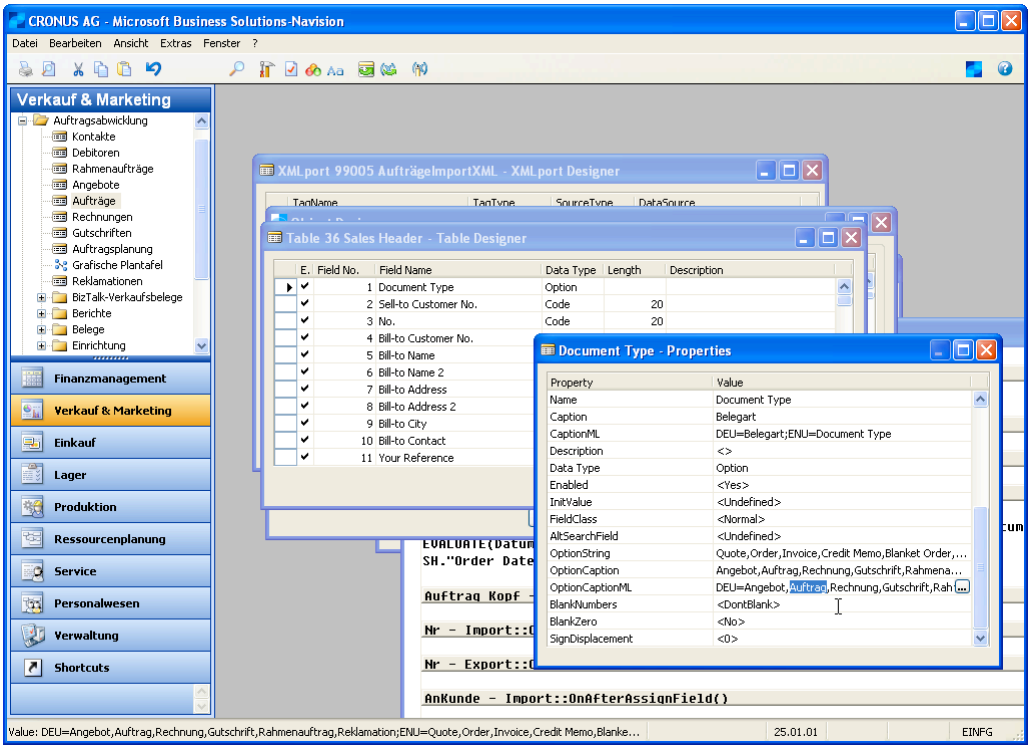

Abbildung 55: Auffinden des "Document Type"

Hier sehen Sie die interne Bezeichnung des Feldes („Document Type“) und dessen Datentyp („Option“). Die Eigenschaften der Tabelle offenbaren, dass „Auftrag“ bzw. „Order“ die zweite Option ist. Ausgehend davon, dass die Zählung der Optionen bei Null anfängt, tragen Sie folgendes in Ihren Programmcode ein:

```
SH."Document Type":=1;
```

Damit weist der XMLport jedem neuen Datensatz die Belegart „Auftrag“ zu.

Wie bereits angedeutet, gestaltet sich die Konvertierung des Datums etwas aufwendiger; übernehmen Sie folgenden Code:

```
DatumText:=COPYSTR(TmpDatum,9,2) + COPYSTR(TmpDatum,6,2) +
      COPYSTR(TmpDatum,1,4);
EVALUATE(DatumDate, DatumText);
SH."Order Date":=DatumDate;
```

Die Variable TmpDatum enthält das Auftragsdatum aus der XML-Datei, z.B. `2005-09-24T17:05:48`. In der ersten Zeile werden mit dem Befehl `COPYSTR` Tag, Monat und Jahr als Teilstrings entnommen und in der Variable `DatumText` zusammengefügt; das Ergebnis ist z.B. `23092005`. In der zweiten Anweisung wird dieser Text in eine Datumsvariable umgewandelt, die dann in der dritten Zeile in das Feld „Order Date“ der Tabelle Sales Header geschrieben wird.

Der gesamte Block sieht nun so aus:

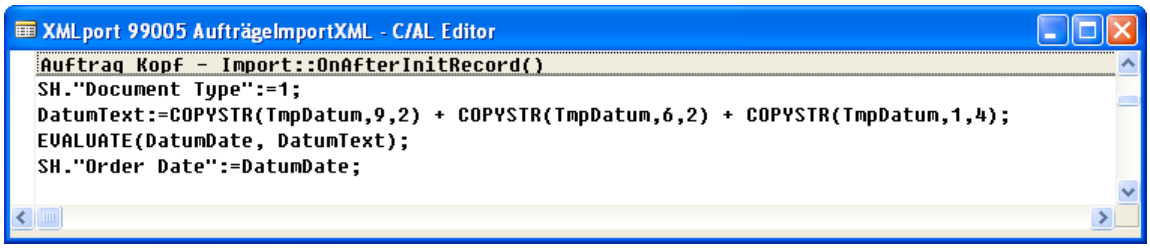

Abbildung 56: Programmcode Auftrag_Kopf - Import::OnAfterInitRecord()

Fahren Sie mit Code-Block für die Auftragszeilen fort:

```
Auftrag_Zeile - Import::OnBeforeInsertRecord()
```

Setzen Sie den Dokumententyp wie oben:

```
SL."Document Type":=1;
```

Die Tabelle Sales Line enthält ebenfalls ein Feld, welches den Zielkunden bezeichnet. Bei den Aufträgen aus Access entspricht stets dem Kunden im Auftragskopf. Setzen Sie daher:

```
SL."Sell-to Customer No.":=SH."Sell-to Customer No.";
```

Bei Betrachtung der Tabelle Sales Line werden Sie festgestellt haben, dass die Zeilen eines Auftrags in Zehntausenderschritten nummeriert sind. Die folgenden Zeilen bilden die in der XML-Datei nicht vorhandene Nummerierung nach, indem für jeden neuen Datensatz die Variable SLCounter um 10.000 erhöht und anschließend dem Feld „Line No.“ zugewiesen wird.

```
SLCounter:=SLCounter+10000;
SL."Line No.":=SLCounter;
```

Setzen Sie nun noch das Feld „Type” auf 2, um jeden einzelnen Posten als Artikel zu deklarieren.

```
SL.Type:=2;
```

Mit dem abschließenden Aufruf von „Validate“ wird der OnValidate-Trigger des Feldes „No.“ aufgerufen; erst jetzt, da nun alle erforderlichen Felder ausgefüllt wurden. Die Validierung zu einem früheren Zeitpunkt (etwa über die entsprechende Eigenschaft) würde zu einer Fehlermeldung führen. Der Trigger fügt der Auftragszeile u.a. Artikelinformationen hinzu und berechnet Preise.

```
SL.VALIDATE(SL."No.");
```

Der gesamte Code für diesen Abschnitt sieht damit wie folgt aus:

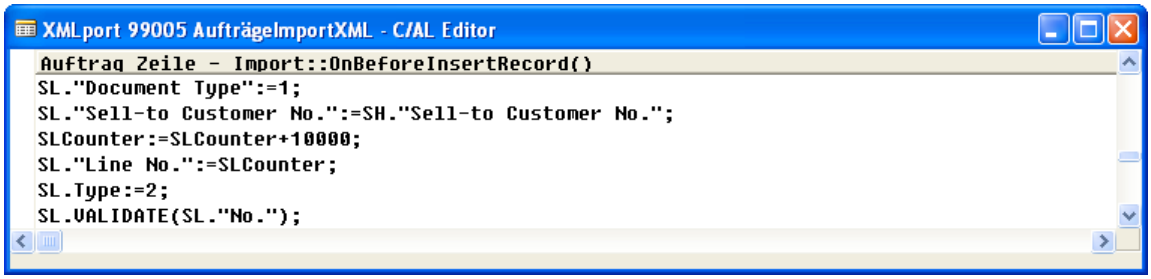

Abbildung 57: Programmcode Auftrag_Zeile - Import::OnBeforeInsertRecord()

Suchen Sie den Abschnitt

```
Auftrag_Kopf - Import::OnAfterInsertRecord()
```

und tragen Sie folgende Zeile ein:

```
SLCounter:=0;
```

Damit wird der Zähler für die Auftragszeilen mit jedem neuen Auftrag auf 0 gesetzt.

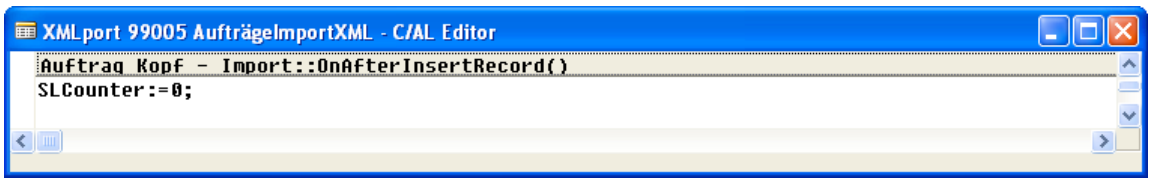

Abbildung 58: Programmcode Auftrag_Kopf - Import::OnAfterInsertRecord()

Speichern und schließen Sie den nun vollständigen XMLport.

4.3. Aufruf des XMLports über C/AL

Erstellen Sie ein neues Codeunit, welches den XMLport aufruft. Orientieren Sie sich dabei an untenstehenden Abbildungen. Beachten Sie, dass der XMLport entgegen dem vorherigen Kapitel diesmal dem Import von Daten dient; entsprechend wird eine Datei geöffnet und deren Inhalt in einen `InStream` geschrieben.

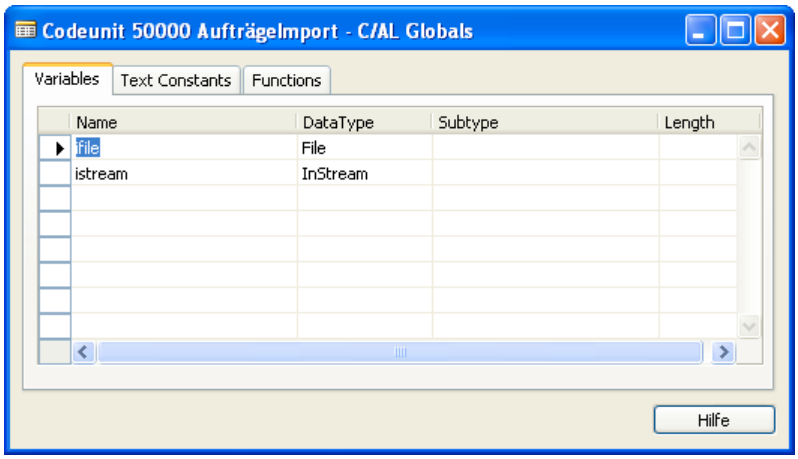

Abbildung 59: C/AL Globals

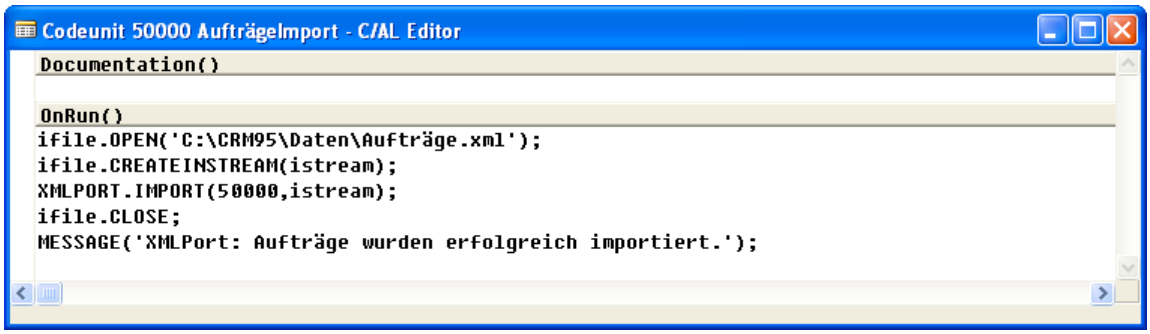

Abbildung 60: Programmcode - Aufruf des XMLports zum Auftragsimport

Schließen Sie das Codeunit und geben Sie ID und Bezeichnung an.

4.4. Test des XMLports

Der XMLport ist nun einsatzbereit. Um zu sehen, welcher Programmcode durch den Import ausgelöst wird, aktivieren über den Menüpunkt *Extras → Debugger → Active (Shift + Strg + F11)* den Debugger.

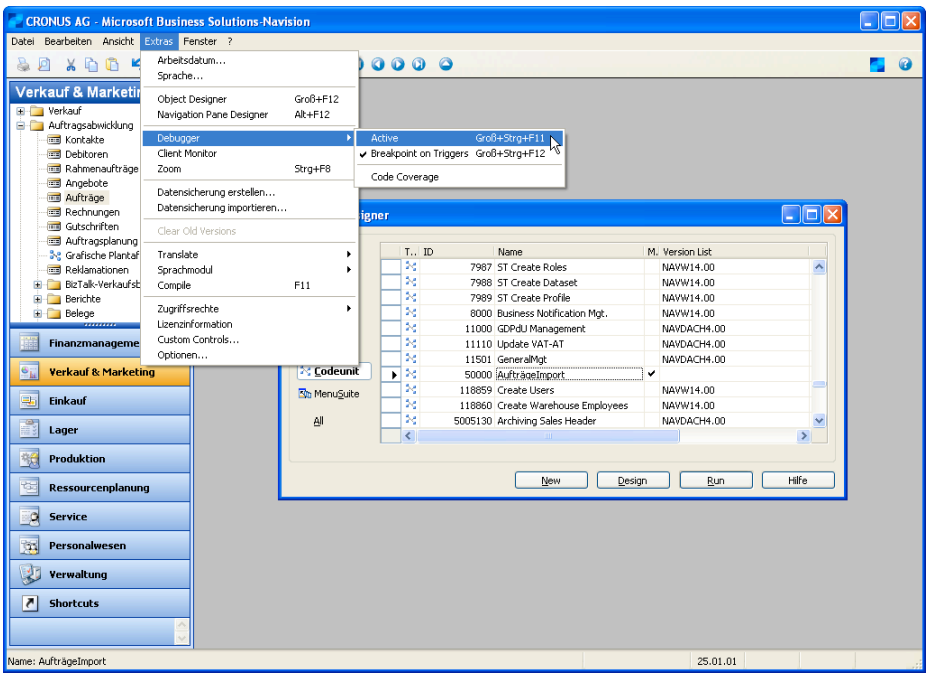

Abbildung 61: Aktivieren des Debuggers

Starten Sie das Codeunit wie gehabt über den Button *Run* des Object Designers. Daraufhin öffnet sich der Debugger mit folgender Ansicht:

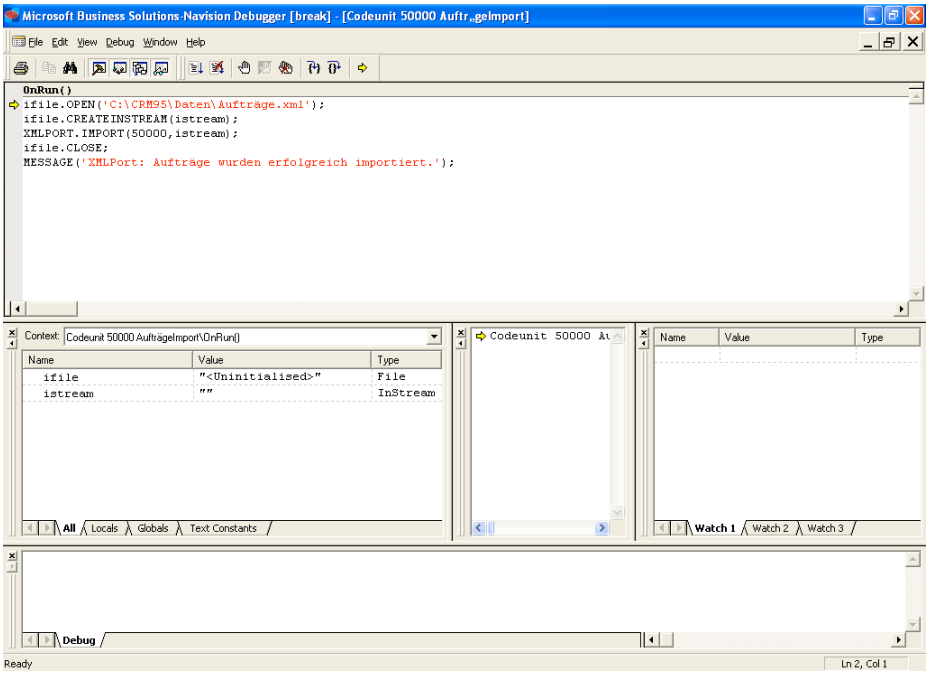

Abbildung 62: Nutzung des Debuggers

Im oberen Drittel des Fensters wird der gerade ausgeführte Programmcode angezeigt, der aktuellen Zeile des Codes ist ein gelber Pfeil vorangestellt. Mit der Taste F8 (Menü: *Debug → Step into*) können Sie die jeweils aktuelle Zeile ausführen, ein Druck auf F5 (Menü: *Debug → Go*) arbeitet einen ganzen Codeblock ab. Bewegen Sie sich auf diese Weise durch Ihren eigenen Programmcode. Zunächst wird das Codeunit

ausgeführt, anschließend die Befehle, die Sie im XMLport hinterlegt haben. Beachten Sie zudem die Auswirkungen der FieldValidate-Eigenschaften: Wird ein zu validierendes Feld mit Daten gefüllt, springt der Programmcode zur entsprechenden OnValidate-Funktion des Feldes innerhalb der Navision-Datenbank. Im Debugger können Sie so beobachten, wie durch das Ereignis `Sell-to Customer No. - OnValidate()` ein neuer Auftrag mit Kundendaten gefüllt wird, die nicht im ursprünglichen XML-Dokument hinterlegt sind.

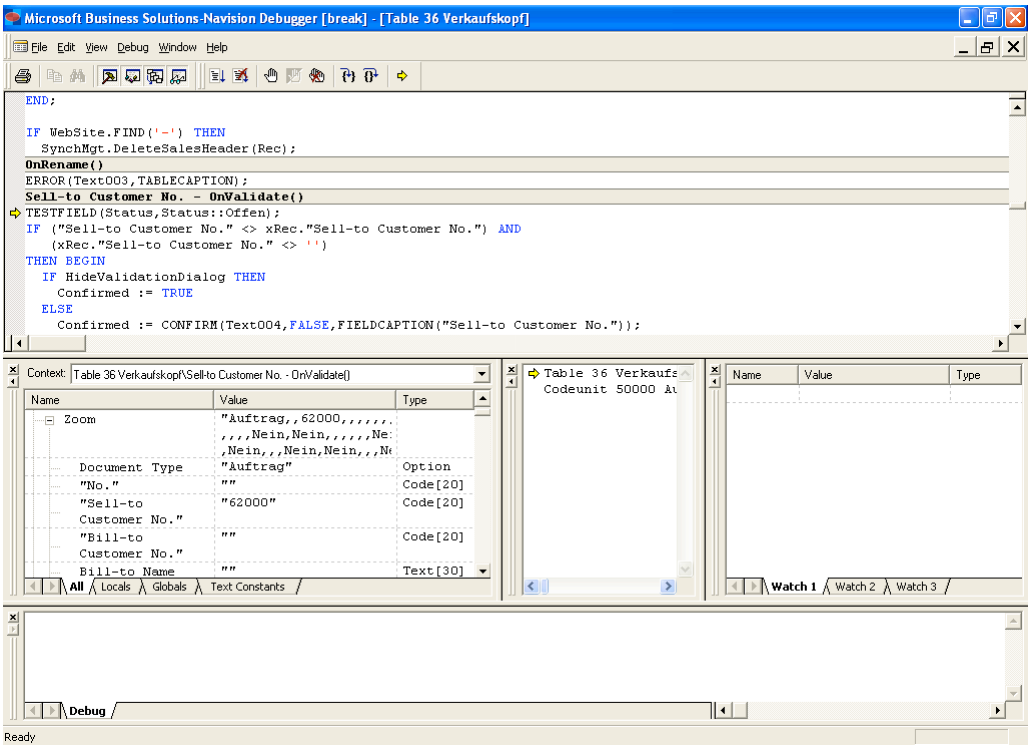

Abbildung 63: Debugger - Programmablauf und zusätzliche Informationen

Im mittleren Teil des Debuggers werden zusätzliche Informationen angezeigt: Im Fenster auf der linken Seite finden Sie alle zur Laufzeit des Programms vorhandenen Variablen, deren Werte und Datentypen. Wie in der obigen Abbildung können Sie z.B. verfolgen, wie der aktuelle Datensatz der Tabelle „Sales Header“ mit Werten aus dem Dataport und anderen Tabellen gefüllt wird. Das Fenster in der Mitte des Bildschirms zeigt alle gerade aktiven Funktionsaufrufe an; hier können Sie die Verzweigungen in andere Objekte und Programmteile von Navision überblicken. Ein Doppelklick auf eine der Funktionen zeigt deren Fortschritt im Programmcode durch einen grünen Pfeil an.

Wenn Sie sich ein Bild von den zahlreichen Programmaufrufen gemacht haben, können Sie den Debugger jederzeit über den Menüpunkt *Debugger → Debugger Active* ausschalten. Schließen Sie anschließend das Debugger-Fenster, um den verbleibenden Programmcode mit normaler Geschwindigkeit fortzusetzen. Wenn alle Aufträge erfolgreich importiert wurden, erscheint die in der letzten Zeile des Codeunits aufgerufene Messagebox.

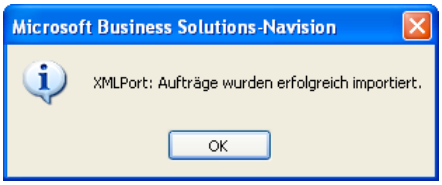

Abbildung 64: Messagebox - erfolgreicher Import neuer Aufträge

Rufen Sie nach erfolgtem Import die Tabellen Sales Header (36) und Sales Line (37) über den Object Designer auf und suchen Sie die neu hinzugefügten Datensätze. Beachten Sie, dass für jeden Auftrag automatisch eine neue Auftragsnummer vergeben wird, auf die zudem in jeder Auftragszeile referenziert wird. Außerdem wurden die wenigen Ausgangsdaten durch die Validierungen um zahlreiche Kunden- und Artikelinformationen ergänzt. Ferner sehen Sie, dass auch der Zähler für die einzelnen Auftragszeilen funktioniert.

Betrachten Sie abschließend das Ergebnis Ihrer Arbeit auch aus der Anwenderperspektive: Selektieren Sie im Seitenmenü *Verkauf & Marketing → Auftragsabwicklung → Aufträge* und rufen Sie den importierten Auftrag auf. Sie sehen, dass alle relevanten Daten des Auftrags vorhanden sind. Vertriebsmitarbeiter können die neuen Aufträge nun prüfen, weiterbearbeiten oder direkt buchen.

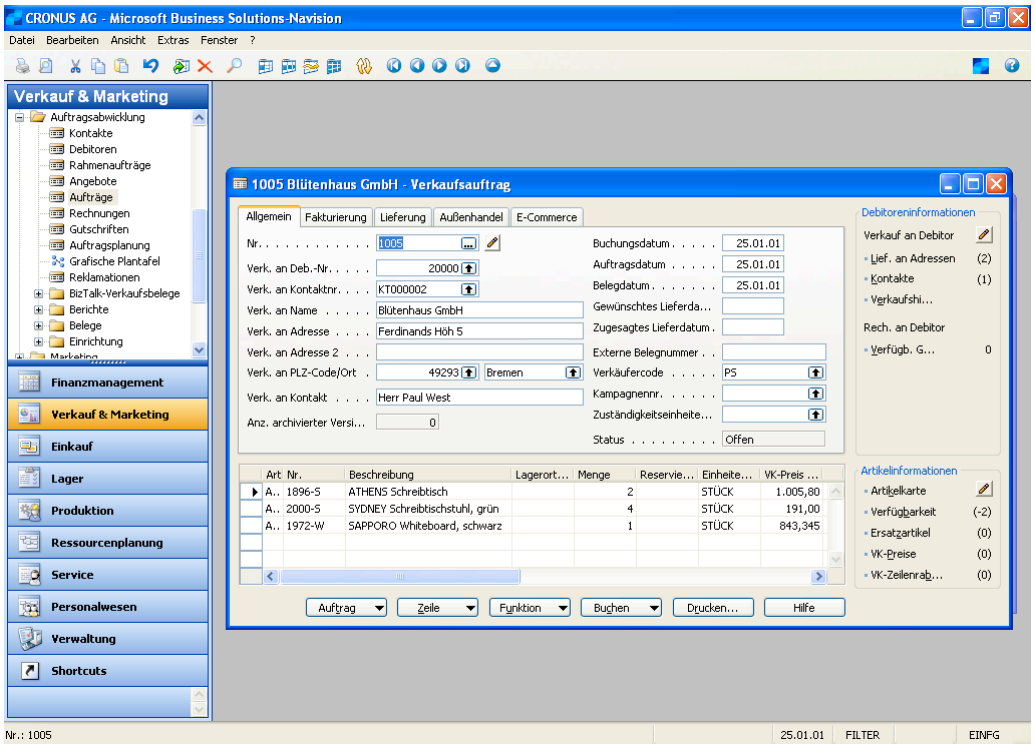

Abbildung 65: Importierte Aufträge in der Navision Auftragsabwicklung

4.5. Einbindung in die Benutzeroberfläche

Abschließend können Sie die neue Funktionalität zu Ihrem Import/Export-Fomular hinzufügen. Rufen Sie dieses über den Object Designer auf. Kopieren Sie die Schaltfläche für den Artikelexport und fügen Sie diese erneut ein. Benennen Sie die neue Schaltfläche um und ändern Sie deren Programmcode, so dass sie das eben erstellte Codeunit ausführt. Beenden Sie anschließend den Designmodus, indem Sie Ihr Formular schließen.

Abbildung 66: Import/Export-Formular

Mit Abschluß dieses Kapitels haben Sie nun eine Integrationslösung entwickelt, welche die Auftragsdaten aus WinCRM medienbruchfrei nach Navision überträgt.

Sie sind nun am Ende der Fallstudie angelangt und haben erfolgreich eine komplette Integrationslösung für die Applikation WinCRM erstellt. Alle neu eingegebenen Daten aus der Access-Anwendung sind jetzt in Navision nutzbar, ebenso wird WinCRM stets mit aktuellen Daten aus Navision versorgt.

Literatur

An dieser Stelle finden Sie Literaturempfehlungen, welche die Arbeit mit der Fallstudie u.a. durch weitere Anleitungen und technisches Hintergrundwissen weiter unterstützen können.

Navision Application Designer's Guide

Diese Einführung in die Anwendungsentwicklung innerhalb von Navision beschreibt ausführlich und leicht verständlich alle grundlegenden Objekte der C/SIDE-Programmierung. Für die Fallstudie relevant sind insbesondere die folgenden Abschnitte:

Part 2 Fundamentals	***S. 37***
Part 4 Forms	***S. 137***
Part 6 Codeunits	***S. 281***
Part 7 Dataports	***S. 409***
Part 8 XMLports	***S. 443***

Sie finden den Application Designer's Guide auf der 1. Navision-Installtions-CD im Verzeichnis *doc*.

C/SIDE Reference Guide

Dieser Teil der Navision-Online-Hilfe (zu erreichen über das Menü *?* → *C/SIDE Reference Guide*) beinhaltet Informationen zu allen Funktionen, Datentypen, Eigenschaften und Triggern von C/SIDE. Ferner werden auch einige grundlegende Funktionen von C/SIDE näher erläutert. Wie üblich können Sie die Hilfe zu gerade selektierten Programmfunktionen oder Eigenschaften durch Drücken von F1 direkt aufrufen.