Mit Visual FoxPro und ODBC auf die Oracle Anwendungen zugreifen, um Bankkontonummern zu prüfen

von Dr. Volker Thormählen

1. Zusammenfassung

Am Beispiel der Prüfziffernberechnung für Bankkontonummern wird berichtet, wie mit Visual FoxPro (im Folgenden VFP genannt) und standardisierter Softwareschnittstelle auf die Datentabellen der Oracle Anwendungen zugegriffen werden kann, wobei diese auf einem entfernten Zentralrechner vorhanden sind.

- Zuerst werden die dafür erforderlichen Softwareprodukte auf der Client- und Server-Seite erläutert.
- Anschließend wird dargestellt, wo Bankleitzahlen und Bankkontonummern im Datenmodell von Oracle-Kreditoren zu finden sind.
- Der Aufbau und die Funktionsweise eines tabellengesteuerten VFP-Programms zur Durchführung der Prüfziffernberechnung wird vorgestellt.
- Abschließend wird kurz gestreift, wie sich die zukünftige allgemeine Verwendung von IBAN (Abk. f. **I**nternational **B**anking **A**ccount **N**umber) im inländischen Zahlungsverkehr auf die Prüfziffernberechnung auswirken wird.

2. Softwarebausteine

VFP ist ein eigenständiges relationales Datenbanksystem von Microsoft, das sich auch dazu eignet, als Front-End-System mit fremden Datenbanksystemen mittels standardisierter Softwareschnittstelle (ODBC) zu kommunizieren, vorausgesetzt, das verwendete Netzwerkprotokoll ist gleich. ODBC ist die engl. Abkürzung für **O**pen **D**ata**B**ase **C**onnectivity.

Mit VFP unter dem Betriebssystems Windows und dem *Oracle8 ODBC* Treiber kann auf Oracle Datenbanken schreibend und lesend zugegriffen werden, sofern auch der von Microsoft unterstütze Treibermanager für die offene Datenbankschnittstelle implementiert ist. Um mit dem *Oracle8 ODBC* Treiber eine zentrale oder entfernte Datenquelle zu nutzen, müssen folgende Voraussetzungen erfüllt sein:

- Auf dem Client (Desktop, Laptop, etc.) muss neben dem Betriebssystem Windows auch *Oracle Net8 Client* installiert sein. Außerdem muss die Datei TNSNAMES.ORA auf dem Client gespeichert sein. Sie enthält Deklarationen für den Hauptrechner und den Namen der zentralen Datenbank.
- Der Hauptrechner, der die zentrale Datenbank beherbergt, muss mit *Oracle Server 7.3.4.0.0* oder höher ausgestattet sein.
- Eine Netzwerkverbindung zwischen Client und Server muss vorhanden sein.
- Eine VFP-Anwendung, die zentrale Daten benötigt, muss entwickelt worden sein.

3. Schichtenmodell

Abb. 1 veranschaulicht die Schichten der standardisierten Datenbankschnittstelle.

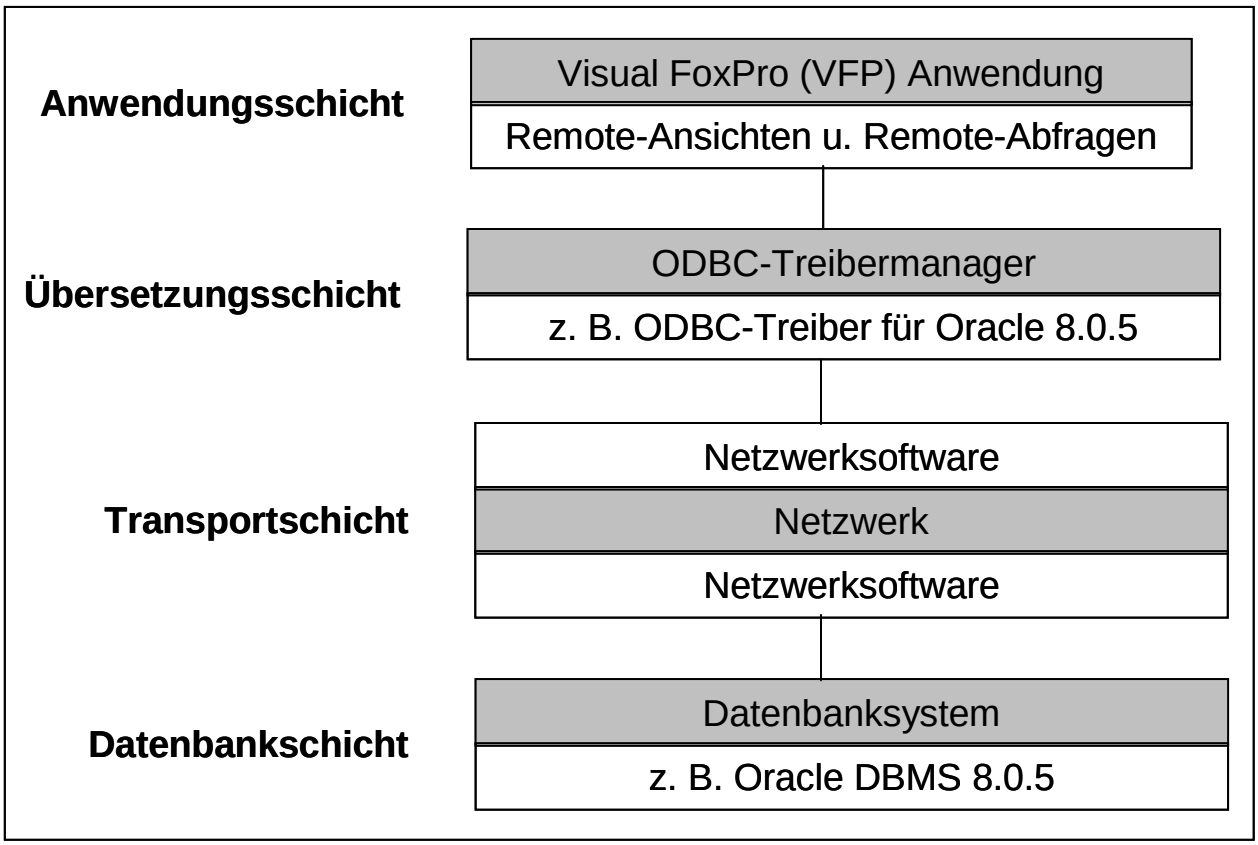

Abb. 1: Verbindung zwischen Client und Server über die offene Datenbankschnittstelle
Quelle: in Anlehnung an [1], Seite 512

Diese Schichten werden nun näher beschrieben.

3.1 Anwendungsschicht

VFP 7.0 wird seit September 2001 angeboten. Wer noch mit VFP 6.0 arbeitet, welches seit 1998 auf dem Markt ist, sollte das aktuelle Service Pack 5 installieren. Es ist von der Microsoft-Download-Seite erhältlich.

Eine Datenbank, die auf dem zentralen Server vorhanden ist, wird auch als ODBC-Datenbank bezeichnet, wenn sie über die offene Datenbankschnittstelle angesprochen werden kann. Ein Vorteil von VFP in Verbindung mit ODBC ist, dass damit Anwendungen entwickelt werden können, die weitgehend unabhängig von der Datenbank auf dem zentralen Server sind.

Es gibt zwei Möglichkeiten, um mittels VFP auf ODBC-Datenbanken zuzugreifen:

- Der *Assistent für Remote-Ansichten* gestattet die Einbindung zentraler Datentabellen. Die Einbindung fremder Datentabellen in Form von entfernten Ansichten bietet den Vorteil, dass mit diesen wie gewohnt gearbeitet werden kann. VFP versucht, diese Ansichten/Abfragen zu optimieren, damit sie möglichst schnell Ergebnisse liefern.
- Sogenannte Pass-Throuh-Abfragen ermöglichen die direkte Verbindung zu fremden Datentabellen durch das Absetzen von SQL-Anweisungen in VFP. SQL ist die Abk. für **S**tructured **Q**uery **L**anguage. Pass-Throuh-Abfragen sind also SQL-spezifische Abfragen. Diese werden im Gegensatz zu normalen Abfragen nicht optimiert. Auch die syntaktische Richtigkeit der SQL-Anweisungen wird nicht geprüft. Diese Abfrageart ist vorzuziehen, wenn die auszuführenden SQL-Anweisungen nicht von VFP interpretiert werden können oder wenn es sich um kurze Abfragen handelt.

Abb. 2 wird gezeigt, wie eine Pass-Throuh-Abfrage in Form von SQL-Anweisungen in ein VFP-Programm eingebaut werden kann.

```
SET TALK OFF
CLOSE ALL
CLEAR ALL
CLEAR
OPEN DATABASE c:\fox\fox_prog\daten1.dbc EXCLUSIVE
= DBSETPROP("my_connect","CONNECTION","DispLogin",3)
= DBSETPROP("my_connect","CONNECTION","DispWarnings",.f.)
STORE SQLCONNECT("oap","usr","passw") TO gnConnHandle
IF gnConnHandle <= 0
   = MESSAGEBOX('Kann Verb. nicht herstellen',16,'SQL-Verb.Fehler')
ELSE
   = MESSAGEBOX('Verb. hergestellt',48,'SQL-Verb.Meldung')
   = SQLEXEC(gnConnHandle,;
   'SELECT DISTINCT BA.BANK_ACCOUNT_NUM,BB.BANK_NUM '+;
   'FROM AP_BANK_ACCOUNTS_ALL BA,AP_BANK_BRANCHES BB '+;
   'WHERE BA.BANK_BRANCH_ID = BB.BANK_BRANCH_ID '+;
   'AND LENGTH(RTRIM(BB.BANK_NUM)) = 8 '+;
   'AND LENGTH(RTRIM(BA.BANK_ACCOUNT_NUM)) > 1 '+;
   'AND LENGTH(RTRIM(BA.BANK_ACCOUNT_NUM)) < 11',;
   'my_cursor')
   BROWSE
   = SQLDISCONNECT(gnConnHandle)
ENDIF
SET TALK ON
RETURN
```

Abb.2: Einbau einer SQL-Abfrage in ein VFP-Programm

3.2 Übersetzungsschicht

Bei ODBC handelt es sich um eine zweistufige Softwareschnittstelle, die aus einem Treibermanager und mehreren Datenbanktreibern besteht. Der Manager trägt den Dateinamen ODBC32.dll und wird im Verzeichnis WINDOWS\SYSTEM eingerichtet. Der Manager kann beliebig viele Treiber für unterschiedliche Datenbanksysteme bzw. SQL-Dialekte verwalten.

Jeder ODBC-Treiber ist für den Zugriff auf eine bestimmte Datenbank geeignet. Deshalb werden sie für das jeweilige Datenbanksystem in der Regel vom Hersteller mitgeliefert. Außerdem können ODBC-Treiber von verschiedenen Fremdherstellern eingesetzt werden, beispielsweise von DataDirect (vgl. dazu [4]). Wenn der passende ODBC-Treiber noch nicht

auf dem Client installiert ist, kann dies nachgeholt werden. Dazu ist ein Setup-Programm des jeweiligen Herstellers erforderlich (z. B. Oracle Universal Installer).

Um Daten aus einer ODBC-Datenbank abrufen zu können, muss noch eine Datenquelle definiert werden. Für jedes Datenbanksystem können beliebig viele Datenquellen definiert werden. Dazu dient in der *Systemsteuerung* des Clients das Symbol *ODBC Data Sources (32 Bit)* . Durch Klicken auf das Symbol wird der ODBC-Verwalter aufgerufen. Nach dessen Start werden standardmäßig die ODBC-Datenquellen angezeigt. Durch Klicken auf die Schaltfläche *Hinzufügen* (engl. *Add*) wird ein zuvor bereits installierter ODBC-Treiber ausgewählt, der Name der Datenquelle eingegeben und die sonst noch erforderlichen Angaben definiert. Anschließend kann über das Netzwerk auf die Informationen in der ODBC-Datenbank mit den beschriebenen Abfragearten zugegriffen werden.

ODBC übernimmt die Rolle eines Übersetzers. Einerseits werden die Datenquellen im Netzwerk lokalisiert und andererseits werden SQL-Dialekte und Zeichensätze konvertiert.

3.3 Transportschicht

In der Transportschicht werden Daten zwischen Client und Server ausgetauscht, die über ein Netzwerk miteinander verbunden sind. Dazu ist ein Kommunikationsprotokoll notwendig, beispielsweise TCP/IP (Abk. für **T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol). Die TCP/IP Protokollfamilie hat sich als de facto Marktstandard im Bereich der Kommunikation über das Intra- und Internet etabliert.

3.4 Datenbankschicht

Die meisten Standarddatenbanken, wie z.B. Oracle, unterstützen ODBC. Die wesentlichen Aufgaben von ODBC beinhalten den Verbindungsaufbau, die Verwaltung der Datenquellen und die Analyse von Abfragen. ODBC-Treiber verarbeiten ODBC-Funktionsaufrufe, senden SQL-Abfragen an bestimmte Datenquellen und geben die Ergebnisse an die aufrufende Anwendung zurück.

4. Datenmodell

Bankleitzahlen und Bankkontonummern werden auf der Kreditoren-Seite im Wesentlichen zur Zahlungsregulierung mittels EFT (Abk. für **E**lectronic **F**unds **T**ransfer) herangezogen. Auf der Debitoren-Seite werden diese Informationen hauptsächlich für das Abbuchungsverfahren benötigt. Abb. 3 beinhaltet einen Ausschnitt aus dem Datenmodell für Oracle-Kreditoren. Er zeigt, wo die Eingabe-Informationen für die Prüfziffernberechnung bei Bankkontonummern zu finden sind.

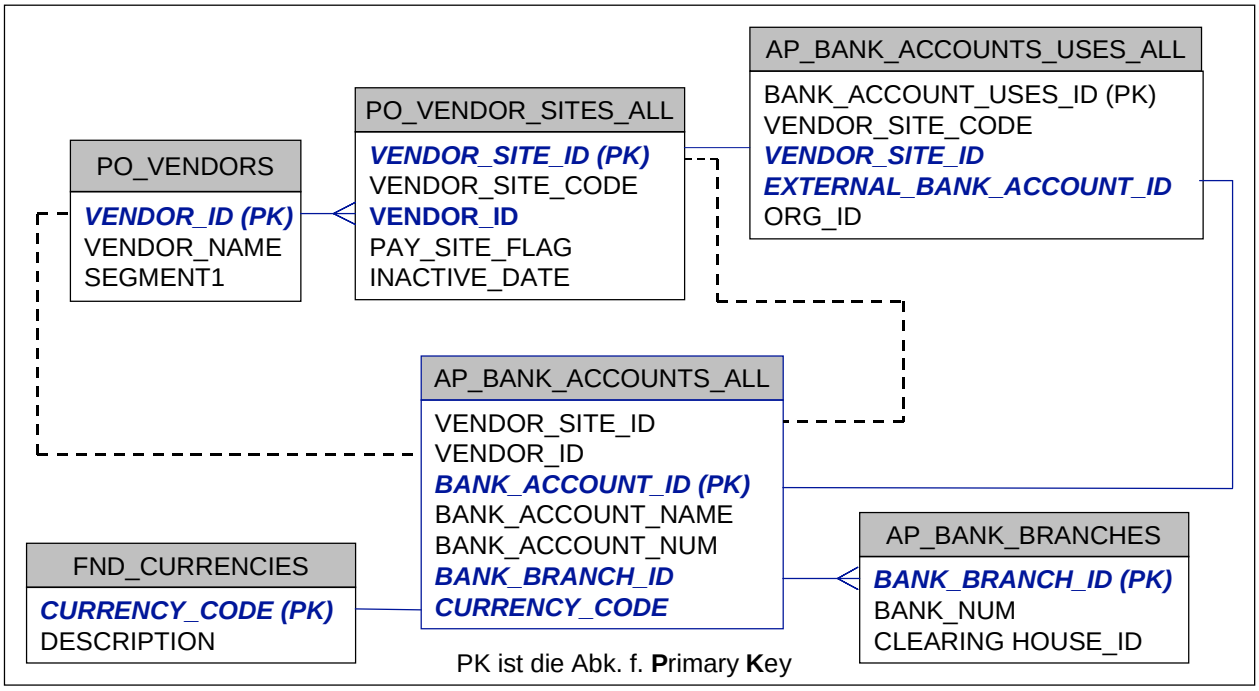

Abb. 3: Ausschnitt aus dem Datenmodell für Oracle-Kreditoren, Release 11.0.3

Die grau hinterlegten Boxen enthalten die Tabellennamen. Darunter stehen jeweils die wichtigsten Spaltennamen. Die Verbindungslinien verdeutlichen, wie die Tabellen miteinander verbunden sind.

5. Extraktionsskript

SQL (**S**tructured **Q**uery **L**anguage) ist eine Abfragesprache für relationale Datenbanken, die weit verbreitet ist und von allen namhaften Datenbankherstellern unterstützt wird. Ausgehend von dem in Abb. 3 dargestellten Datenmodell kann ein SQL-Skript entwickelt werden, mit dem die Inputdaten für die Prüfziffernberechnung bei Bankkontonummern gewonnen werden können.

Das SQL-Skript in Abb. 4 extrahiert 7 Datenfelder unter Verwendung von 5 Tabellen und unter Berücksichtigung von 10 Bedingungen.

Die Spaltennamen hinter dem **SELECT**-Befehl kommen auch in Abb. 3 vor. Mit der Zeichenkettenfunktion **SUBSTR** wird die gewünschte Zahl der Datenstellen der auf der linken Seite selektierten Datenfelder definiert. Auch die Tabellennamen hinter dem reservierten Wort **FROM** korrespondieren mit denen in Abb. 3. Die ersten vier Bedingungen hinter der **WHERE**-Klausel können in Abb. 3 als Verbindungslinien zwischen den benutzten Tabellen identifiziert werden. Die restlichen 6 Bedingungen dienen zur Einschränkung der Ergebnismenge.

```
SELECT
    SUBSTR(BA.BANK_ACCOUNT_NAME,1,30) AS BK_KTO_BEZ,
    SUBSTR(BA.BANK_ACCOUNT_NUM,1,10)  AS BK_KTO_NR,
    SUBSTR(BA.CURRENCY_CODE,1,3)      AS WAEHRUNG,
    SUBSTR(BB.BANK_NUM,1,8)           AS BLZ,
    SUBSTR(PV.VENDOR_NAME,1,30)       AS LIEF_NAME,
    SUBSTR(PV.SEGMENT1,1,10)          AS LIEF_NR,
    SUBSTR(VS.VENDOR_SITE_CODE,1,20)  AS LIEF_ORT
FROM
    AP_BANK_ACCOUNT_USES_ALL BAU,
    AP_BANK_ACCOUNTS_ALL     BA,
    AP_BANK_BRANCHES         BB,
    PO_VENDORS               PV,
    PO_VENDOR_SITES_ALL      VS
WHERE
        BAU.EXTERNAL_BANK_ACCOUNT_ID = BA.BANK_ACCOUNT_ID
    AND VS.VENDOR_ID                 = PV.VENDOR_ID
    AND VS.VENDOR_SITE_ID            = BAU.VENDOR_SITE_ID
    AND BA.BANK_BRANCH_ID            = BB.BANK_BRANCH_ID
    AND LENGTH(RTRIM(BB.BANK_NUM))   = 8
    AND LENGTH(RTRIM(BA.BANK_ACCOUNT_NUM)) > 1
    AND LENGTH(RTRIM(BA.BANK_ACCOUNT_NUM)) < 11
    AND VS.PAY_SITE_FLAG IS NOT NULL
    AND VS.INACTIVE_DATE IS NULL
    AND BAU.ORG_ID = 599;
```

Abb. 4: SQL-Skript zur Extraktion der Eingabeinformationen zur Prüfziffernberechnung

Das SQL-Skript in Abb. 4 muss an die jeweiligen Verhältnisse angepasst werden, nicht zuletzt wegen der ORG_ID, die hier als Konstante (599) vorgegeben wurde.

6. Programmierte Kontrollen

Bei der manuellen Eingabe von deutschen Bankleitzahlen und Bankkontonummern können Fehler vorkommen. Diese könnten durch programmierte Kontrollen entdeckt werden. Aber das Oracle-Kreditoren-Modul umfasst dafür keine brauchbaren Lokalisierungen. Tab. 1 enthält alle Kontrollarten, die in der Standardversion von Oracle-Kreditoren, Release 11, zur Anwendung kommen.

Default (minimum)			
Required Fields	Accepted Data Type	Maximum Length	Padded with Leading Zeros?
Bank Account Name	alphanumeric	25	no *)
Bank Account Number	alphanumeric	30	no *)
Currency Code	value set	15	
Quelle: [3], Seite 2-139 bis 2-140.			Hinweis: *) „no“ ergänzt vom Verfasser

Tab. 1: Programmierte Kontrollen für Bankinformationen im Standard von Oracle-Kreditoren

Im Abschnitt „*Bank Account Validation by Country*“ (siehe [3], S. 2–139 bis 2-141) wird die Prüfziffernberechnung lediglich für 3 Länder (Frankreich, Portugal, Spanien) genannt.

Die im Standard vorgesehenen Kontrollen sind aus der Sicht eines deutschen Anwenders nicht straff genug. Um die Datenqualität (und damit Sicherheit der Fehlererkennung) zu erhöhen, muss der Anwender selbst strengere Kontrollmaßnahmen ergreifen.

Bei deutschen Bankleitzahlen können zum Beispiel folgende Kontrollen vorgenommen werden:

- Zeichenprüfung (nur Ziffern sind erlaubt)
- Formatkontrolle (genau 8 Ziffern sind erforderlich)
- Existenzkontrolle (Bankleitzahl ist in entsprechender Datentabelle vorhanden)

Bei deutschen Bankkontonummern sind folgende Kontrollen möglich:

- Zeichenprüfung (nur Ziffern erlaubt)
- Formatkontrolle (höchstens 10 Stellen erlaubt)
- Prüfziffernkontrolle (Prüfziffernberechnung geht auf)

Die bekanntesten Prüfziffernverfahren, die auch bei Bankkontonummern angewandt werden, sind die sogenannten Modulo-n-Verfahren. Dabei werden die einzelnen Stellen einer nummerischen Nummer jeweils mit einem unterschiedlichen Faktor gewichtet und die Summe der Produkte durch einen bestimmten Wert dividiert. Dieser Divisor (meistens 7, 9, 10 oder 11) entspricht n. Wenn beispielsweise n = 11 ist, spricht man vom Modulo-11-Verfahren, oder kurz vom 11er-Verfahren.

Bei dem in der Praxis häufig verwendeten 11er-Verfahren wird die Prüfziffer wie folgt errechnet:

- Den Stellen der Kontonummer werden von rechts nach links, mit 2 beginnend, in aufsteigender Folge die Multiplikatoren 2, 3, 4, 5, 6, 7 fest zugeordnet. Ab der siebten Stelle wiederholen sich die Faktoren.
- Jede Ziffer der Kontonummer wird mit dem zugehörigen Multiplikator multipliziert.
- Die Ergebnisse der einzelnen Multiplikationen werden addiert.
- Die auf diese Weise gebildete Produktsumme wird durch 11 dividiert.
- Ergibt sich kein Divisionsrest, ist die Prüfziffer gleich Null.
- Ist der Divisionsrest gleich 1, wird die Kontonummer nicht vergeben; dadurch fallen rund 9 Prozent (1/11) aller möglichen Kontonummern aus.
- Ist der Divisionsrest größer als 1, wird dieser von 11 subtrahiert; die Differenz ist dann die Prüfziffer.

Die Anwendung des Modulo-11-Verfahrens auf die 10stellige Bankkontonummer “0013027692“ wird in Abb. 5 in Form einer benutzerdefinierten VFP-Funktion mit dazugehöriger Testroutine demonstriert. Die Testroutine soll lediglich zeigen, wie die Funktion *modulo_11* von einem übergeordneten Modul aufgerufen wird. Der Ablauf der Testroutine wird nicht über programmexterne Datentabellen gesteuert, um die Verständlichkeit nicht unnötig zu erschweren. Im Allgemeinen ist die Position der Prüfziffer in einer Bankkontonummer nicht fix, sondern variabel. Zur Vereinfachung wird im Testbeispiel unterstellt, dass die Einerstelle der Bankkontonummer die Prüfziffer (hier 2) enthält.

```
SET TALK OFF
CLEAR
Kontonr = "0013027692"
PZ      = Modulo_11(kontonr)
IF PZ   = RIGHT(kontonr,1)
   ? "Kontonummer richtig"
ELSE
   ? "Kontonummer falsch"
ENDIF
SET TALK ON
RETURN
*-----------------------------------------------------------------------------
FUNCTION Modulo_11
PARAMETERS Kontonummer
PRIVATE Divisionsrest, Gewichtung, Modulus, Multiplikand, Multiplikator,;
        Pruefziffer, Produktsumme, Stelle
IF LEN(Kontonummer) <> 10
   Pruefziffer = "Err"
ELSE
   Modulus      = 11
   Gewichtung   = "432765432"
   Produktsumme = 0
   Stelle       = 1
   DO WHILE Stelle <= 9
      Multiplikand  = VAL(SUBSTR(Kontonummer,Stelle,1))
      Multiplikator = VAL(SUBSTR(Gewichtung, Stelle,1))
      Produktsumme  = Produktsumme + (Multiplikand * Multiplikator)
      Stelle        = Stelle + 1
   ENDDO
   Divisionsrest = MOD(Produktsumme, Modulus)
   DO CASE
      CASE Divisionsrest = 0
         Pruefziffer = "0"
      CASE Divisionsrest = 1
         Pruefziffer = "Err"
   OTHERWISE
         Pruefziffer = STR(Modulus - Divisionsrest,1,1)
   ENDCASE
ENDIF
RETURN (Pruefziffer)
```

Abb. 5: VFP-Funktion mit Testroutine zur Anwendung des Modulo-11-Verfahrens

Die Bundesbank führt eine Übersicht der im Kreditgewerbe angewandten Verfahren zur Prüfziffernberechnung mit den von ihr dafür vergebenen Kennziffern. Diese Kennziffern sind in Feld 15 des Bankleitzahlen-Datensatz enthalten. Die Zahl der Prüfziffernverfahren für deutsche Bankkontonummern beträgt inzwischen mehr als 100, siehe [6], Rubrik Zahlungsverkehr. Es stellt sich daher die Frage, ob wirklich jedes der vielen Verfahren individuell programmiert werden muss oder ob hierarchisch verbundene, funktionsspezifische Unterprogramme eingesetzt werden können, um diese Kontrollaufgaben insgesamt wirksamer zu bewältigen. Im folgenden Abschnitt wird gezeigt, dass tatsächlich eine allgemein gültige Subroutine für alle Verfahren der Prüfziffernberechnung bei Bankkontonummern nach den Grundsätzen der tabellengesteuerten und modularen Programmgestaltung entwickelt werden kann.

7. Tabellengesteuerte, modulare Prüfziffernberechnung

Das VFP-Unterprogramm zur Prüfziffernberechnung bei Bankkontonummern benötigt im Wesentlichen 3 Eingabeinformationen:

- Kennziffer des Verfahrens = {00, 01, 02, ..., 99, A0, ...}, 2stellig, alphanummerisch
- Bankleitzahl, 8stellig, nummerisch)
- Bankkontonummer, maximal 10stellig, nummerisch

Zusätzlichen werden 3 Datentabellen mit folgenden Informationen benötigt:

- Parameter zur Beschreibung der einzelnen Verfahren (z. B. Position der Prüfziffer in der jeweiligen Bankkontonummer)
- Gewichtungsschemata für die Stellen der Bankkontonummern
- Transformationswerte, die für bestimmte Verfahren notwendig sind

Das VFP-Unterprogramm gibt 3 Ausgabeninformationen an das aufrufende Treiberprogramm zurück:

- Fehlerschalter (0, wenn die überprüfte Bankkontonummer fehlerfrei ist, sonst 1)
- Fehlermeldung (leer, wenn Fehlerkode = 0, sonst gefüllt)
- Errechnete Prüfziffer

Die Prüfziffernberechnung umfasst im Normalfall folgende 3 Verarbeitungsschritte:

a) Bankkontonummer in eine 10 bzw. 12stellige Zeichenkette umwandeln und ggf. weiter aufbereiten. Dazu zählt u. a. das Auffüllen mit führenden Nullen. Die aufbereitete Bankkontonummer bildet die Grundlage für die folgenden zwei Schritte.

b) In Abhängigkeit von der Kennziffer des Prüfziffernverfahrens sind folgende Einzelschritte durchzuführen:

   - Die Position der Prüfziffer in der Kontonummer bestimmen und isolieren
   - Beginn und Ende der Kontonummer feststellen
   - Das benötigte Gewichts- bzw. Transformationsschema auswählen
   - Die Art der (Quer-) Summenbildung ermitteln und durchführen
   - Den ganzzahligen Divisor x (mit $x \in \{7, 9, 10, 11\}$) bestimmen und damit den Rest der ganzzahligen Division sowie die Prüfziffer berechnen

c) Die isolierte Prüfziffer mit der berechneten Prüfziffer vergleichen und bei Übereinstimmung den Fehlerschalter gleich 0, sonst gleich 1 setzen.

Für die Ablaufsteuerung des VFP-Unterprogramms sind 4 weitere Datentabellen unter VFP notwendig, auf deren ausführliche Beschreibung hier verzichtet wird.

Die Hierarchie des VFP-Unterprogramms zur Prüfziffernberechnung bei Bankkontonummern veranschaulicht Abb. 6.

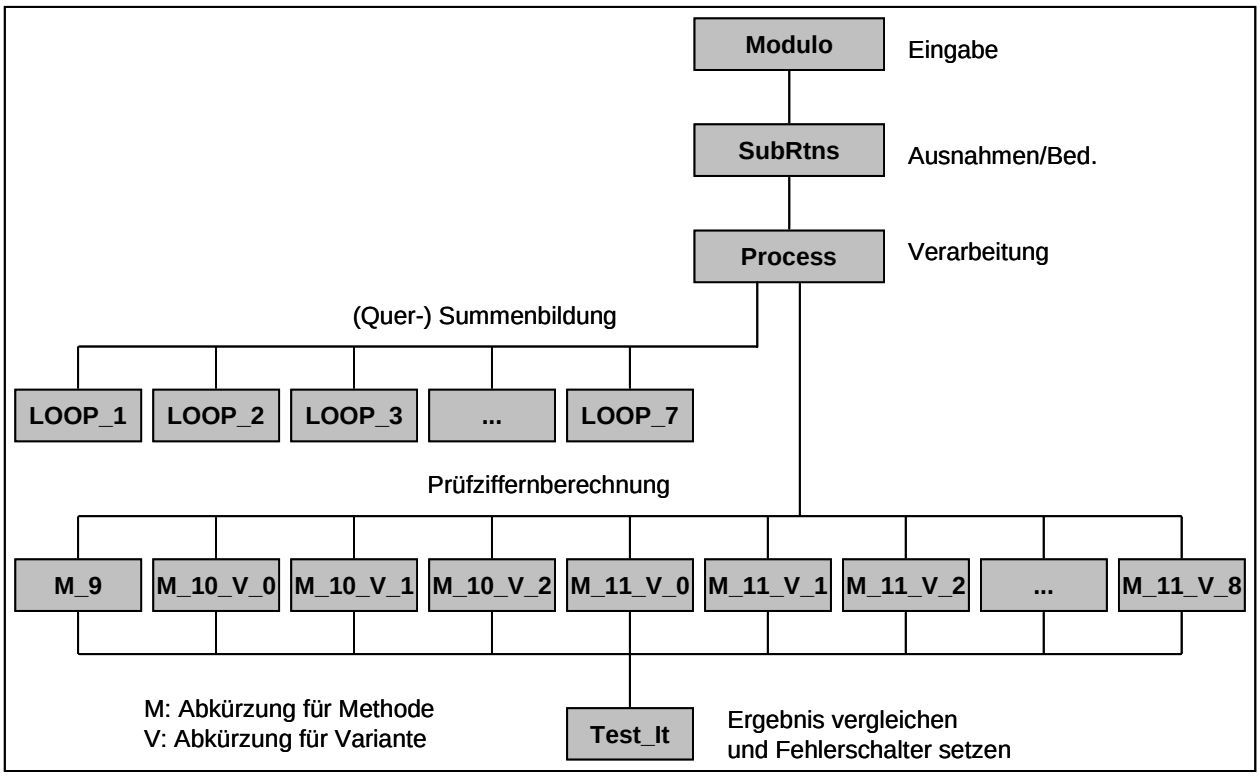

Abb. 6: Struktur des VFP-Unterprogramms zur Prüfziffernberechnung

Aus Abb. 6 ist unmittelbar ersichtlich, dass die Subroutinen zu einer 5-stufigen Hierarchie gehören. Das Zusammenspiel zwischen Hauptprogramm und Unterprogrammen sowie zwischen Eingabe- und Ausgabeinformationen geht aus folgendem Datenflussdiagramm hervor (siehe Abb. 7).

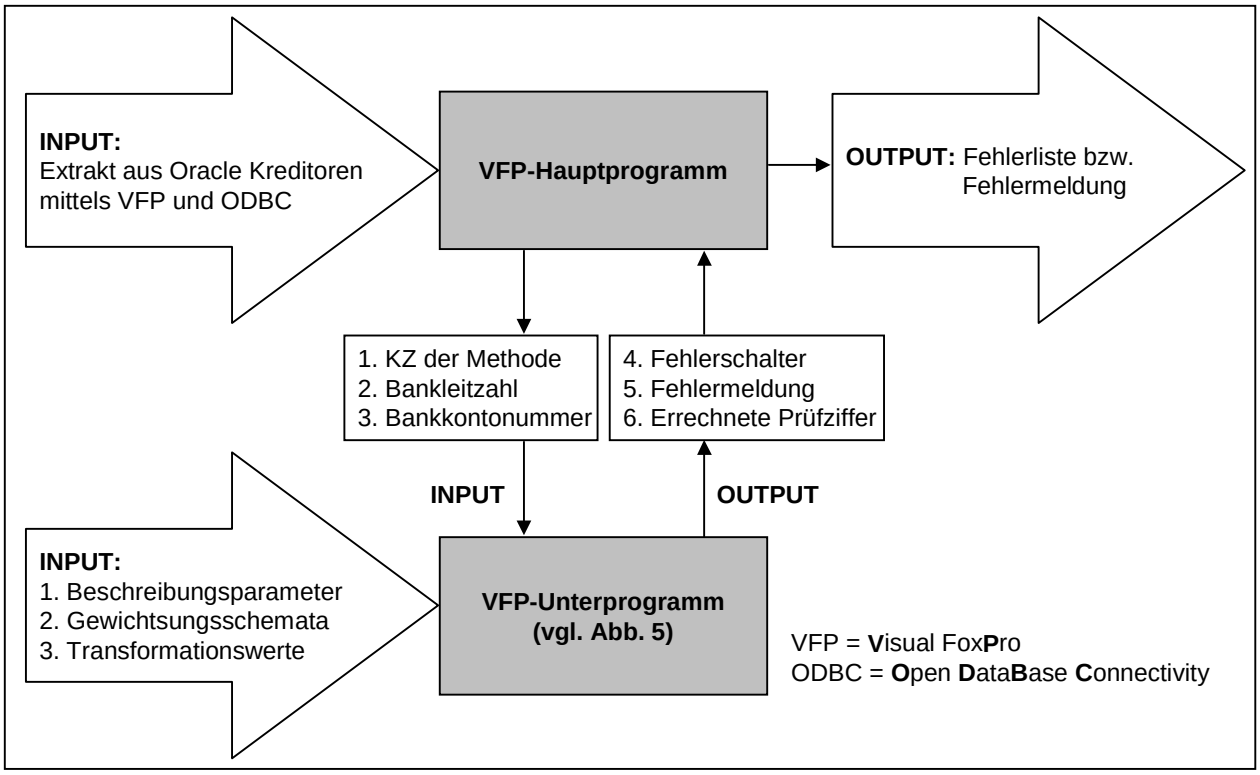

Abb. 7: Datenfluss bei der Prüfziffernberechnung für Bankkontonummern

8. Vorteile

Richtige Bankleitzahlen und Bankkontonummern bieten den Vorteil, dass Doppelarbeit bei der Zahlungsregulierung im Inland weitgehend vermieden werden kann.

Die beschriebene VFP-Lösung bietet den Vorteil, dass auf die entfernte ODBC-Datenbank problemlos zugegriffen werden kann, um Bankverbindungen aus Oracle-Kreditoren zu extrahieren und zu prüfen.

Anzahl und Art der Prüfziffernverfahren können sich mit jeder Bankleitzahlenänderung ändern. Die tabellengesteuerte Ablaufsteuerung der VFP-Programme gestattet eine schnelle und sichere Programmpflege, weil ggf. nur neue Tabelleneinträge notwendig sind. Hinzu kommt als Vorteil, dass die Datentabellen nicht nur den Programmablauf steuern, sondern zugleich die relevanten Prüfziffernverfahren einheitlich und präzise dokumentieren.

9. Ausblick

Neben BIC (Abk. f. **B**ank **I**dentifier **C**ode) für den beleglosen Datenaustausch im Auslandszahlungsverkehr, auch als SWIFT-Nummer (Abk. f. **S**ociety for **W**orldwide **I**nterbank **F**inancial **T**elecommunication) bekannt, wird in Zukunft auch IBAN zum Einsatz kommen. IBAN (**I**nternational **B**anking **A**ccount **N**umber) ist eine vom ***E**uropean **C**ommittee for **B**anking **S**tandards* (ECBS, siehe [7], [8], [9]) entwickelte Norm, mit der Bankkontoadressen international nach einheitlicher Logik aufgebaut, angegeben, erkannt und geprüft werden können. Ein hoher Automatisierungsgrad durch ***S**traight **T**hrough **P**rocessing* (STP) ist das Ziel. Die Norm wird derzeit eingerichtet und kann demnächst genutzt werden, auch für den inländischen Zahlungsverkehr. Die allgemeine Verwendungspflicht wird vom ZKA (dem Zentralen Kreditausschuss, siehe [5]) entschieden, dem die Spitzenverbände der deutschen Kreditwirtschaft angehören.

Die IBAN wird für jedes Bankkonto *individuell* berechnet und setzt sich aus Länderschlüssel, IBAN-Prüfziffern, Bankleitzahl und Bankkontonummer zusammen. Jede deutsche IBAN ist 22 Stellen lang. Tab. 2 beinhaltet den genauen Satzaufbau für deutsche IBAN.

Anzahl Stellen	Position	Format	Inhalt
2	1 - 2	alphabetisch	Länderschlüssel gem. ISO 3166 (**DE** für Deutschland)
2	3 - 4	nummerisch	IBAN-Prüfziffern
8	5 - 12	nummerisch	Bankleitzahl
10	13 - 22	nummerisch	Bankkontonummer (inkl. Prüfziffer, falls vorhanden)
22		alphanummerisch	Gesamtlänge der IBAN-Adresse

Tab. 2: Nummernschema für deutsche IBAN

Die IBAN-Prüfziffern können mit Rechenvorschriften bestimmt werden, die in [8] auf den Seiten 12 bis 14 veröffentlicht worden sind. Die Darstellung dieser Rechenvorschriften als VFP-Quellkode ist im *Anhang* zu finden.

Weiterhin zu unterscheiden ist zwischen den IBAN-Prüfziffern einerseits und der Prüfziffer für deutsche Bankkontonummern andererseits. Diese Unterscheidung gilt sinngemäß auch für andere europäische Länder, die sich an der Einführung von IBAN beteiligen.

Abgesehen von den zu erwartenden Übergangsschwierigkeiten bei der allgemeinen Einführung der IBAN sind keine nennenswerten Änderungen an dem beschriebenen Ablauf zur Prüfziffernberechnung bei Bankkontonummern notwendig. Die in diesem Beitrag beschriebene Lösung wird allerdings überflüssig, wenn Oracle die deutsche Lokalisierung des Kreditoren- und Debitorenmoduls entsprechend ausweiten würde.

10 Anhang: Prüfziffernberechnung für IBAN

Die in Abb. 8 (ab Zeile 7) gezeigten 19 IBAN-Adressen (mit Leerstellen zur externen Darstellung) wurden aus [7] und [8] für Testzwecke entnommen.

```
SET TALK OFF
CLOSE ALL
CLEAR ALL
CLEAR
max_Schleifen = 19                                  && Zahl der Testfälle
DECLARE IBAN_Adr[max_Schleifen]
IBAN_Adr[01]="AT61 1904 3002 3457 3201"          && Austria
IBAN_Adr[02]="BE62 5100 0754 7061"               && Belgium
IBAN_Adr[03]="DK50 0040 0440 1162 43"            && Denmark
IBAN_Adr[04]="FI21 1234 5600 0007 85"            && Finland
IBAN_Adr[05]="FR14 2004 1010 0505 0001 3M02 606" && France
IBAN_Adr[06]="DE75 3805 0000 0108 6053 46"       && Germany
IBAN_Adr[07]="GR16 0110 1250 0000 0001 2300 695" && Greece
IBAN_Adr[08]="IS14 0159 2600 7654 5510 7303 39"  && Iceland
IBAN_Adr[09]="IE29 AIBK 9311 5212 3456 78"       && Ireland
IBAN_Adr[10]="IT21 Q054 2801 6000 0ABC D12Z E34" && Italy
IBAN_Adr[11]="LU13 7490 1976 0571 0110"          && Luxembourg
IBAN_Adr[12]="NL53 ABNA 0540 4271 52"            && Netherlands
IBAN_Adr[13]="NO93 8601 1117 947"                && Norway
IBAN_Adr[14]="PL27 1140 2004 0000 3002 0135 5387"&& Poland
IBAN_Adr[15]="PT50 0002 0123 1234 5678 9015 4"   && Portugal
IBAN_Adr[16]="ES91 2100 0418 4502 0005 1332"     && Spain
IBAN_Adr[17]="SE35 5000 0000 0549 1000 0003"     && Sweden
IBAN_Adr[18]="CH93 0076 2011 6238 5295 7"        && Switzerland
IBAN_Adr[19]="GB29 NWBK 6016 1331 9268 19"       && United Kingdom
Schleifenzaehler = 1
DO WHILE Schleifenzaehler <= max_Schleifen
   IBAN_Extern = IBAN_Adr[Schleifenzaehler]
   Laenge      = LEN(IBAN_Extern)
   IBAN        = LEFT(IBAN_Extern,4)
   Stelle      = 5
   * Leerstellen entfernen und alphabetische Zeichen
   * in 2stellige nummerische Nummern umwandeln
   DO WHILE Stelle <= Laenge
      Zeichen = SUBSTR(IBAN_Extern,Stelle,1)
      DO CASE
         CASE Zeichen $ "0123456789"
            IBAN = IBAN + Zeichen
         CASE Zeichen = SPACE(1)
            IBAN = IBAN + ""
         OTHERWISE
            IBAN = IBAN + STR(ASC(Zeichen) - 55,2,0)
      ENDCASE
      Stelle = Stelle + 1
    ENDDO
    * Pruefziffernberechnung ausführen für IBAN
    Pruefziffer = IBAN_pruefen(IBAN)
    * Wenn Ergebnis = 1, dann IBAN richtig, sonst falsch
   IF Pruefziffer = 1
      ? "IBAN richtig:", IBAN_Adr[Schleifenzaehler]
   ELSE
      ? "IBAN falsch: ", IBAN_Adr[Schleifenzaehler]
   ENDIF
   Schleifenzaehler = Schleifenzaehler + 1
ENDDO
SET TALK ON
RETURN
```

Abb. 8: Treiberroutine zur Durchführung der Prüfziffernberechnung für 19 Testfälle

In Abb. 8 steuert die äußere DO WHILE Schleife die Verarbeitung der 19 IBAN-Testfälle. Die innere DO WHLE Schleife steuert (1) das Entfernen der Zwischenräume und (2) das Umwandeln von Buchstabenstellen in 2stellige nummerische Nummern (vgl. Abb. 8).

Die folgende benutzerdefinierte VFP-Funktion (s. Abb. 9) verlagert ISO-Länderschlüssel und IBAN-Prüfziffern vom Anfang an das Ende der IBAN-Adresse und wandelt den ISO-Länderschlüssel in zwei 2stellige nummerische Nummern um.

```
FUNCTION IBAN_pruefen
PARAMETERS IBAN
PRIVATE IBAN, IBAN_PZ, IBAN_Teil1, IBAN_Teil2, ISO_Land
* Länderschlüssel nach ISO-Norm, 2stellig, alphabetisch
ISO_Land   = SUBSTR(IBAN,1,2)
* Prüfziffer, 2-stellig, nummerisch
IBAN_PZ    = SUBSTR(IBAN,3,2)
* Nummerischen Teil hinter der Prüfziffer als 1. Nummernteil speichern
IBAN_Teil1 = SUBSTR(IBAN,5)
* Alphabetischen Länderschlüssel in 4stellige nummerische Nummer
* umwandeln und hinten als 2. Nummernteil anhängen, gefolgt von der
* nummerischen Prüfziffer als 3. Nummernteil
IBAN_Teil2 = STR(ASC(SUBSTR(ISO_Land,1,1)) - 55,2,0) +;
             STR(ASC(SUBSTR(ISO_Land,2,1)) - 55,2,0) + IBAN_PZ
* Modulo 97 Prüfziffernberechnung ausführen
RETURN Modulo_97(IBAN_Teil1 + IBAN_Teil2)
```

Abb. 9: Benutzerdefinierte VFP-Funktion zur Durchführung der vorbereitenden Schritte für die Prüfziffernberechnung mit Aufruf der benutzerdefinierten VFP-Funktion Modulo_97

In Abb. 8 und Abb. 9 wird die Umwandlung der alphabetischen Stellen der IBAN in 2stellige Zahlen mit folgender Methodik durchgeführt (siehe Abb. 10):

```
STR(ASC('A') – 55, 2, 0) ergibt 10
STR(ASC('B') - 55, 2, 0) ergibt 11
STR(ASC('C') - 55, 2, 0) ergibt 12
STR(ASC('D') - 55, 2, 0) ergibt 13
STR(ASC('E') - 55, 2, 0) ergibt 14
...
STR(ASC('Z') - 55, 2, 0) ergibt 35
```

Abb. 10: Methode zur tabellenfreien Umwandlung von Buchstabenstellen in 2stellige Zahlen

Die nachstehende VFP-Funktion Modulo_97 ist erforderlich, weil MOD(VAL(Ziffernfolge), 97) nicht funktioniert, wenn in VFP eine ganze Zahl mehr als 16 Stellen umfasst. Das Modulo-97-Verfahren wird schrittweise innerhalb einer DO WHILE Schleife ausgeführt. Dabei darf der jeweilige Nummernteil der IBAN höchstens 9 Ziffern lang sein.

```
FUNCTION Modulo_97
PARAMETERS Verbundnummer
PRIVATE Divisionsrest, neun_Stellen, Verbundnummer, Ziffernfolge
Ziffernfolge = Verbundnummer
Divisionsrest = 0
DO WHILE LEN(Ziffernfolge) > 0
   DO CASE
   CASE Divisionsrest > 9
      neun_Stellen = LTRIM(STR(Divisionsrest)) + SUBSTR(Ziffernfolge,1,7)
      Ziffernfolge = SUBSTR(Ziffernfolge,8)
   CASE Divisionsrest > 0
      neun_Stellen = LTRIM(STR(Divisionsrest)) + SUBSTR(Ziffernfolge,1,8)
      Ziffernfolge = SUBSTR(Ziffernfolge,9)
   OTHERWISE
      neun_Stellen = SUBSTR(Ziffernfolge,1,9)
      Ziffernfolge = SUBSTR(Ziffernfolge,10)
   ENDCASE
   Divisionsrest =  MOD(VAL(neun_Stellen),97)
ENDDO
RETURN (Divisionsrest)
```

Abb. 11: Benutzerdefinierte VFP-Funktion zur Berechnung von Modulo 97

Statt der Funktion Modulo_97 (siehe Abb. 11) könnte auch das schriftliche Dividieren durch 97 simuliert werden, wobei IBAN als alphanummerischer Datentyp gespeichert wird. Die in Abb. 11 präsentierte MOD-Funktion ist jedoch kürzer und eleganter.

Abb. 12 beinhaltet die Ergebnisse der 19 Testläufe (vgl. Feldgruppe in Abb. 8). Alle IBAN aus den entsprechenden europäischen Ländern sind richtig.

```
IBAN richtig: AT61 1904 3002 3457 3201
IBAN richtig: BE62 5100 0754 7061
IBAN richtig: DK50 0040 0440 1162 43
IBAN richtig: FI21 1234 5600 0007 85
IBAN richtig: FR14 2004 1010 0505 0001 3M02 606
IBAN richtig: DE75 3805 0000 0108 6053 46
IBAN richtig: GR16 0110 1250 0000 0001 2300 695
IBAN richtig: IS14 0159 2600 7654 5510 7303 39
IBAN richtig: IE29 AIBK 9311 5212 3456 78
IBAN richtig: IT21 Q054 2801 6000 0ABC D12Z E34
IBAN richtig: LU13 7490 1976 0571 0110
IBAN richtig: NL53 ABNA 0540 4271 52
IBAN richtig: NO93 8601 1117 947
IBAN richtig: PL27 1140 2004 0000 3002 0135 5387
IBAN richtig: PT50 0002 0123 1234 5678 9015 4
IBAN richtig: ES91 2100 0418 4502 0005 1332
IBAN richtig: SE35 5000 0000 0549 1000 0003
IBAN richtig: CH93 0076 2011 6238 5295 7
IBAN richtig: GB29 NWBK 6016 1331 9268 19
```

Abb. 12: Ergebnisse der 19 IBAN-Testfälle (vgl. Abb. 8)

Die Eingabe (vgl. Abb. 8) und die Ausgabe (vgl. Abb. 12) der IBAN erfolgt in der offiziellen *Vierergliederung* für die *externe* Darstellung (vgl. Abb. 13).

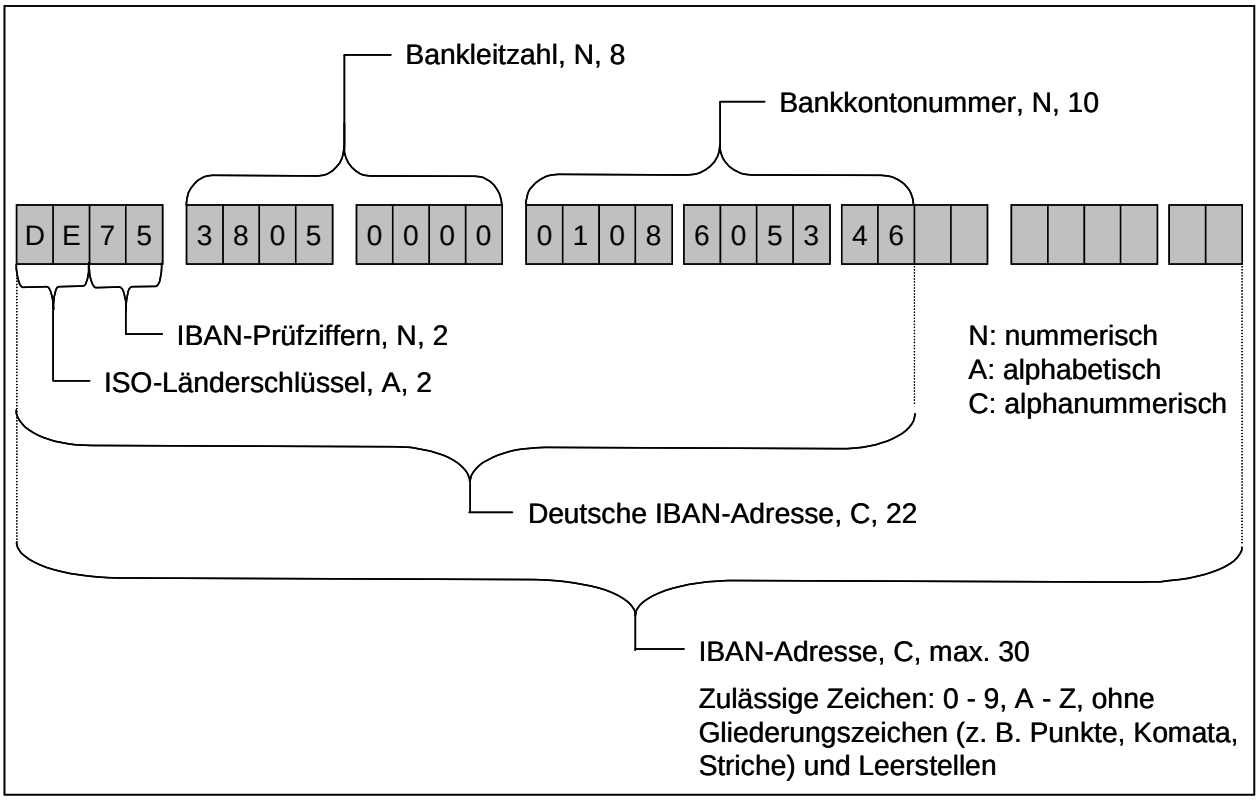

Abb. 13: Vierergliederung der IBAN mit deutschem Beispiel

Für die benutzerfreundliche Erfassung der IBAN in Oracle-Kreditoren und Oracle-Debitoren bietet sich die in Abb. 13 gezeigte Vierergliederung als Maskierung in einem POPUP-Fenster an. Dieses Fenster sollte so gestaltet werden wie die bekannten *FlexFelder* in den Oracle Anwendungen. Dadurch lässt sich eine übersichtliche Aufgliederung der IBAN nach Nummernteilen (das entspricht *Segmenten* in *FlexFeldern*) erreichen.

11. Informationsquellen

[1] Gladis, Ralf*, Das Programmierbuch Visual FoxPro 5.0*, SYBEX-Verlag, Düsseldorf, San Francisco, Paris Soest (NL), 1997, ISBN 3-8155-0263-2

[2] North, Ken, *ODBC Drivers, Servers, and Vendors*, http://ourworld.compuserve.com/homepages/Ken_North/odbcvend.htm, am 23. April 2002

[3] o. V., *Oracle Payables User's Guide, Release 11*, Oracle Corporation, Redwood Shores, CA, March 1998, Part # A58473-01

[4] http://www.datadirect-technologies.com/odbc/eval/odbcdownload.asp

[5] http://www.zka.de/

[6] http://www.bundesbank.de/

[7] http://www.ecbs.org/

[8] o. V., *IBAN: International Bank Account Number*, Hrsg.: ECBS, European Committee for Banking Standards, Brüssel, 2001

[9] o. V., *Register of European Account Numbers*, TR201, Version 2.2.14, Hrsg.: ECBS, European Committee for Banking Standards, Brüssel, März 2002

[10] Schärer, Partrick E., *Der Visual FoxPro 7.0 Anwendungsentwickler, Objektorientierte Datenbankanwendungen mit Visual FoxPro 7.0*, Addison-Wesley Verlag, München, Boston, etc., 2002, ISBN 3-8273-1858-0

12. Kontraktadresse

Dr. Volker Thormählen
Steinhauser Str. 54
40882 Ratingen
Telefon 02102 / 52141
Mobiltelefon 0173 7935232
E-Mail: volker@dr-thormaehlen.de
Internet: http://www.dr-thormaehlen.de