RAP – RATIONENPLAN
RE-ENGINEERING EINES DSS-SYSTEMS
(TECHNISCHE DOKUMENTATION)

Tony Hürlimann

Working Paper

Dezember 2000

INSTITUT D'INFORMATIQUE, UNIVERSITE DE FRIBOURG

INSTITUT FÜR INFORMATIK DER UNIVERSITÄT FREIBURG

Institute of Informatics, University of Fribourg

site Regina Mundi, rue de Faucigny 2, CH-1700 Fribourg / Switzerland

tony.huerlimann@unifr.ch

phone: ++41 26 300 2845 fax: ++41 26 300 9726

This research is supported by the Swiss National Science Foundation and financed by the project no. 12-55989.98.

RAP – Rationenplan
Re-engineering eines DSS-Systems
(Technische Dokumentation)

Tony Hürlimann, PD. Dr. lic. rer. pol.

Keywords: mathematische Modellierung, Software (Re)engeneering.

Zusammenfassung:
Dieser Artikel gibt einen Bericht über das Re-Engineering eines konkreten Projektes: der Konsumlenkung für die schweizerische Lebensmittelrationierung.

Make it as simple as possible, but not simpler.

A. Einstein

1 Einleitung

Es soll hier das Re-engineering eines konkreten Software Projektes als Erfahrungsbericht beschrieben werden. Gleichzeitig soll dieses Dokument auch eine minimale Dokumentation der Veränderungen und des neuen Systems beschreiben. Das Re-engineering wurde in mehreren Schritten vorgenommen: (1) Uebersetzen des LPL-Source Code (Version 3.5) in die neue LPL-Version 4.40; (2) Portierung der Access2 Implementation in die Access 97 Version. Da der zweite Schritt misslang, zudem die frühere Implementation eine so komplexe Struktur hatte, wurde beschlossen, (3) die gesamte Applikation von Grund auf neu zu re-implementieren.

Das Paper besteht aus einem Text, welche die Implementation kurz und in groben Zügen beschreibt, und aus zwei Anhängen, welche den Source-Code der Implementation enthält. Diese Dokumentation sollte für einen erfahrenen Delphi-Programmierer genügend sein, um die Implementation in groben Zügen zu verstehen und wenn nötig zu ändern. Drei weitere Anhänge geben zusätzliche Informationen über die Schwierigkeit bei der Portierung und Re-implementation.

Ausgangspunkt war eine Implementation, bei welcher eine Systemintegration in der Datenbank Access Version 2 (Microsoft) vorlag, die von [Horner A. 1994] geschrieben und des weiteren von Marco Moresino und Reto Jud à jour gehalten wurde. Diese Version war eine Weiterentwicklung der reinen Konsolenapplikation von Paul Langenegger 1991 [Langenegger 1991]. Neben andern Aufgaben wurde von Access die Daten als Textdateien so aufbereitet, dass diese vom RAP-LPL Modell gelesen werden konnten. LPL stellt nach einer Optimierung durch seinen Report-Generator die Resutate einer Optimierung als Textdateien zur Verfügung, die dann durch Access wiederum relativ kompliziert eingelesen werden mussten. Verschiedene Zwischenschritte waren notwendig, um eine reibungslose Kommunikation zwischen der Datenbank und LPL (damals Version 3.5) zu gewährleisten. Die Notwendigkeit resultierte einerseits daraus, dass LPL3.5 keine Möglichkeit hatte, Daten aus Datenbank-Tabellen direkt einzulesen (die READ-Instruktion gab es noch nicht), also mussten die Daten vor jeder Optimierung als LPL-Kode extern generiert werden. Dies leistet das Programm *RAP-SS.EXE*. Verschiedene andere (externe) Programme waren nötig, um den Output von LPL noch aufzubereiten. Auf Grund dieser und anderer Schwierigkeiten wurde die Datenbank auch mit einer Vielzahl von zusätzlichen Zwischentabellen überladen und eine ganze Reihe von Macros und Funktionen waren nötig, um diese Tabellen konsistent zu verwalten.

2 Uebersicht

Die Applikation wurde jetzt von Grund neu konzipiert und aus einem Guss implementiert. Die verschiedenen Komponenten aus der Vorversion wurden auf drei Module reduziert: (1) Die Applikation RAP.EXE, welche das Benutzerinterface

implementiert, wurde mit DELPHI Version 5 neu geschrieben, (2) das LPL Module, welches in die neuen Version 4.40 umgeschrieben wurde, und (3) zwei Access Datenbanken, welche die Daten aller Szenarien bereitstellt.
Die README.TXT Datei gibt Auskunft, welche Dateien die Applikation enthält und wie sie installiert und gestartet werden muss.

```
------------- Datei: README.TXT ------------
Anmerkungen zu DSS-RAP
======================

Dateien:
-------
README.TXT     Diese Datei
rap.exe        Das Rap-Programm
rap.mdb        Die aktuelle Working-Datenbank
scen.mdb       Die Szenario Datenbank
rap.lpl        Das Modell in LPL
raprepo.lpl    Das LPL Report-File
lplcfg.lpl     LPL-Konfigurationsdatei
lplmsg.txt     Fehlermeldungen des LPL
XA.EXE         XA-Solver
XA.INI         XA INI-Datei
rap.doc        Technische Dokumentation (in Word98)
Optional:
lplw.exe       Das LPL Programm
lplw.cfg1      LPLW-Startdatei

Installation:
------------
Die 11 ersten Dateien in ein neues Verzeichnis kopieren. Fertig!
Standardmässig ist der XA-Solver eingestellt. Soll der Cplex-
Solver eingestellt werden, so müssen in der Datei LPLCFG.LPL
drei Zeilen verändert werden:
  OPTION path := ' <hier Pfad zum Solver eingeben> ';
  OPTION solver := xa;  abändern zu  --OPTION solver := xa;
  --OPTION solver := cpl6;  abändern zu  OPTION solver := cpl6;

Voraussetzung:
-------------
1) Es wird vorausgesetzt, dass Access97 installiert ist.
2) Unter Windows 95 muss zuerst DCOM von Microsoft installiert werden.
   Unter Windows 98 kann es sein, dass auch noch DCOM98 zuerst
   installiert werden muss:
   Diese können gefunden werden bei:
     www.microsoft.com/com/resources/download.asp
   Dort: DCOM95 for Windows 95, version 1.3 herunterladen und ausführen
3) Unter Windows 98 muss auch noch die MDAC installiert sein:
   Diese findet sich an:
     www.microsoft.com/data
   Microsoft Data Access Components 2.1.2.4202.3 (GA) herunterladen
   Die Datei MDAC_TYP.exe ausführen.


Ausführen:
---------
Doppelklick auf rap.exe


VERSION
=======
Programm: DSS-RAP (NARA) , Version : 4.0 , Revision: 2 Dezember 2000
Author  : TH (Tony Hürlimann) ; tony.huerlimann@unifr.ch
Ansprechpartner: pius.haettenswiler@unifr.ch
```

Listing 1: Datei README.TXT

Die Daten

Die Daten sind in zwei Access 97 Datenbanken abgespeichert: die Betriebsdatenbank RAP.MDB, welche das aktuelle Szenario enthält, und die SCEN.MDB, welche alle Szenarien bereitstellt. Aus der SCEN.MDB kann ein Szenario ausgewählt werden und in die RAP.MDB kopiert werden. Umgekehrt kann das aktuelle Szenario in der RAP.MDB Datenbank als ein (neues) Szenario in die SCEN.MDB überführt werden.
Die Struktur der beiden Datenbanken sind ähnlich. Die RAP.MDB besteht aus 13 Tabellen mit den Relationen, wie sie im Figure 2 gezeigt werden.

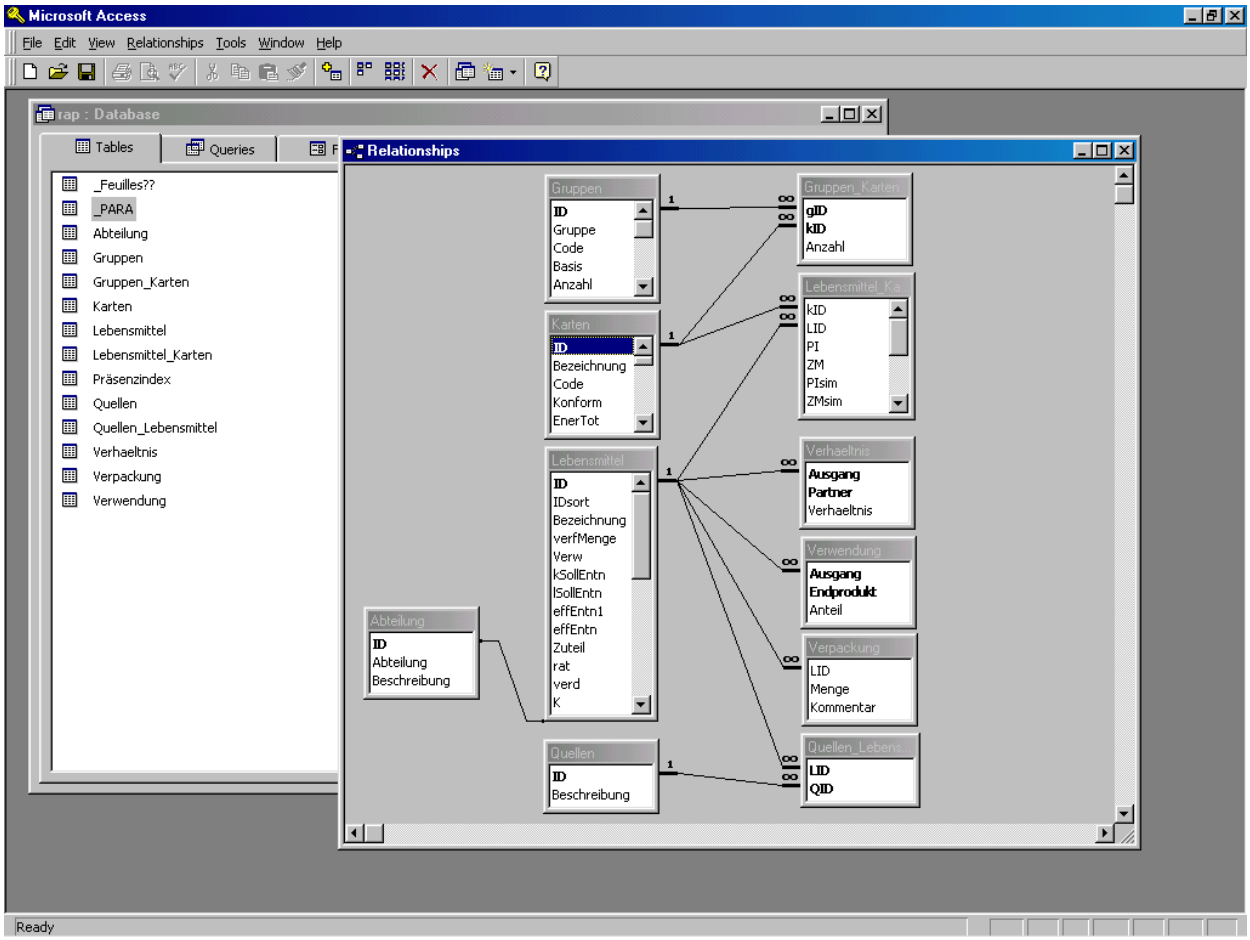

Figure 2: die RAP.MDB Datenbank

Jede Tabelle in SCEN.MDB besitzt noch zwei weitere Felder, welche das Szenario und die Periode bezeichnen. Zudem gibt es in SCEN.MDB noch zwei weitere Tabellen (*_Tabellen* und *_Scenarios*), welche die Liste aller Tabellen, die in beiden Datenbanken gleich sind, sowie alle Szenarios auflistet.
Alle Tabellen und deren Felder sind mit einem Kommentar versehen, welche die Sematik der Objekte beschreibt. (Tabellen-Designer von Access verwenden).
Ausserdem werden noch 9 Queries von RAP.EXE verwendet. Diese haben die Namen *a1* bis *a9*. Beispielsweise besteht der Query *a1* aus folgendem SQL-statement:

```
SELECT L.ID AS c, V.PI, V.ZM, V.PIdef, V.ZMdef, V.ZMopt
FROM Lebensmittel AS L, Lebensmittel_Karten AS V
WHERE (((L.ID)=[V].[LID]) AND ((V.kID)=30));
```

Die Relationen in der RAP.MDB Datenbank sind so aufgebaut, dass automatisch *cascaded deletes* durchgeführt werden bei der Lösung eines Eintrages in der Grundtabelle.

Das LPL Module

Der gesamte LPL-Source Code besteht aus zwei Dateien: RAP.LPL und RARREPO.LPL. Die Datei RAL.LPL enthält die gesamte Modell-Struktur, und die RAPRAPO.LPL Datei enthält alle Rapport Tabellen, die in der Datei RAP.NOM abgelegt werden, welche von LPL generiert wird.

```
READ{p} FROM 'rap,SELECT * FROM Lebensmittel ORDER BY IDsort' :
    p  = 'ID',     NU = 'ID',
    pN = 'Bezeichnung',
    Verw = 'Verw',
    SEK = 'kSollEntn',
    SEL = 'lSollEntn',
    MENGE = 'verfMenge',
    rat = 'Rat',
    GT[p,1] = 'Kal',
    GT[p,2] = 'Ei',
    GT[p,3] = 'Fett',
    GT[p,4] = 'KH',
    GT[p,5] = 'ET',
    GT[p,6] = 'FS',
    MG[p,1] = 'K',
    MG[p,2] = 'Mg',
    MG[p,3] = 'Ca',
    MG[p,4] = 'Fe';

  READ{p,p1} FROM ',Verwendung' :
    p  = 'Ausgang',
    p1 = 'Endprodukt',
    Ausgang = ('Ausgang','Endprodukt'),
    BEDARF  = 'Anteil',
    Gruppe[p1,p] = ('Endprodukt','Ausgang'),
    UM[p1,p] = 'Anteil';

  READ{bg} FROM ',SELECT * FROM Gruppen ORDER BY ID' :
    bg = 'ID',
    bN = 'Code',
    BASIS = 'Basis',
    BEZUEGER = 'Anzahl',
    sBEZUEGER = 'Anzahl';

  READ{k} FROM ',SELECT * FROM Karten ORDER BY ID' :
    k = 'ID',
    kN = 'Code',
    RTYP = 'sTot',
    ENERGIEGEHALT = 'EnerTot',
    VORGABE['EI','KA',k] = 'EnerEi',
    VORGABE['FE','KA',k] = 'EnerFett',
    VORGABE['FS','FE',k] = 'EnerFettS',
    TYP['EI','KA',k] = 'sEi',
    TYP['FE','KA',k] = 'sFett',
    TYP['FS','FE',k] = 'sFettS';

  READ{p,p1} FROM ',Verhaeltnis':
    p = 'Ausgang', p1 = 'Partner',
    VERHAELTNIS = 'Verhaeltnis';
```

```
READ{p,k} FROM ',Lebensmittel_Karten' :
  p = 'lID',
  k  = 'kID',
  ZM = 'ZM',
  PI = 'PI';

READ{bg,k} FROM ',Gruppen_Karten' :
  bg = 'gID', k = 'kID',
  ANZAHL = 'Anzahl';
```

Listing 3: Direktes Lesen von LPL aus RAP.MDB

Die Kommunikation zwischen der Datenbank RAP.MDB und LPL geschieht direkt über READ und WRITE-Statements.
In Listing 3 sind alle (LPL) READ-Statements aufgelistet, die bestimmte Felder der RAB.MDB direkt lesen (Auszug aus RAP.LPL).
Listing 4 zeigt alle WRITE-Statements, welche die Resultate direkt in die RAB.MDB Datenbank zurückschreiben.

```
WRITE{p} TO 'rap,Lebensmittel' :
  'ID' = p,
  'effEntn' = 10000*VLprod[p],
  'Zuteil' = 1000*(SUM{k,bg} ANZAHL*(BEZUEGER+sBEZUEGER)*Kxprod/30)
             / (SUM{bk} BEZUEGER);
WRITE{k,p} TO ',Lebensmittel_Karten' :
  'kID'= k,
  'lID' = p,
  'ZMopt' = 1000*Kxprod[k,p];

WRITE{k} TO ',Karten' :
  'ID'     = k,
  'Ener'   = KxGy[k,1]*1000/30,
  'Ei'     = KxGy[k,2]*1000/30,
  'TierEi' = KxGy[k,5]*1000/30,
  'Fett'   = KxGy[k,3]*1000/30,
  'FettS'  = KxGy[k,6]*1000/30,
  'Kh'     = KxGy[k,4]*1000/30,
  'ratEner'   = SUM{kbR} GT[p,1]*Kxprod/30,
  'ratEi'     = SUM{kbR} GT[p,2]*Kxprod/30,
  'ratTierEi' = SUM{kbR} GT[p,5]*Kxprod/30,
  'ratFett'   = SUM{kbR} GT[p,3]*Kxprod/30,
  'ratFettS'  = SUM{kbR} GT[p,6]*Kxprod/30,
  'ratKh'     = SUM{kbR} GT[p,4]*Kxprod/30,
  'K'         = SUM{q}MG[q,1]*Kxprod/30,
  'Mg'        = SUM{q}MG[q,2]*Kxprod/30,
  'Ca'        = SUM{q}MG[q,3]*Kxprod/30,
  'Fe'        = SUM{q}MG[q,4]*Kxprod/30;
```

Listing 4: Direktes Schreiben von LPL in die RAP.MDB

In den beiden Listings (Listing 3 und 4) wird die *gesamte* Kommunikation zwischen LPL und RAP.MDB transparent ausgeführt.

LPL generiert auch eine RAP.NOM Datei (WRITE-Statements in RAPRAPO.LPL), welche alle Tabellen für das interne Schema bereitstellt. Die Applikation RAP.EXE liest direkt aus dieser Datei. RAP.NOM besteht aus einer Liste von Tabellen, die in

RAP.EXE (internes Schema) einzeln ansprechbar sind. Die Datei RAP.NOM muss dazu folgendes Format haben:

```
>### Header text
Body
```

Wobei > das Grösser-Zeichen bedeutet, ### ist eine dreistellige ganze Zahl, *Header Text* ist der Text auf der ersten Zeile und *Body* ist ein beliebiger Text bestehend aus mehreren Zeilen. Die erste Zeile (`>### Header text`) wird in der ListBox des internen Schemas in RAP.EXE direkt angezeigt. Ein Klick auf einen Eintrag liest direkt aus der RAP.NOM Datei den entsprechenden `Body` und zeigt ihn an. Keine Zwischendateien werden für diese Operation verwendet.

Zur Optimierung wird ein Solver verwendet, welcher in der Datei LPCFG.LPL ausgewählt werden muss, wie in README.TXT beschrieben.
Anhang A enthält das gesamte Listing der beiden LPL-Dateien (RAP.LPL und RAPRAPO.LPL).

Das RAP Module

Die Applikation RAP.EXE wurde mit Delphi 5 kompiliert. Der gesamte Programm-Source-Code besteht im wesentlichen aus einer Datei (Datei MAIN.PAS) zusammen mit der Datei MAIN.DFM, welche die Form-Masken auf dem Bildschirm definiert. Dieser Code besteht aus nicht mehr als 1'400 Zeilen Pascal-Code, der etwa 75 Funktionen enthält, wobei die meisten direkt *Message-Handler* sind! Der Code ist für einen Delphi-Programmierer selbsterklärend und so strukturiert, dass man sich schnell zurechtfinden sollte.
Um den Code zu verstehen, muss man im wesentlichen nur verstehen wie die Applikation über die ADO-Komponenten mit den beiden Datenbanken RAP.MDB und SCEN.MDB kommuniziert. Die Verbindung zu RAP.MDB wird durch folgende Komponenten aufrechterhalten:

```
ADOConnection1: TADOConnection;
ADOTable1: TADOTable;      //Lebensmittel
ADOTable2: TADOTable;      //Abteilung
ADOTable4: TADOTable;      //Gruppen
ADOTable5: TADOTable;      //Gruppen
ADOTable6: TADOTable;      //Karten
ADOTable3: TADOTable;      //Quellen
ADOTable7: TADOTable;      //Temporary
ADOTable10: TADOTable;     //Gruppen
ADOTable11: TADOTable;     //Lebensmittel temporary
ADOQuery1: TADOQuery;      //Nachbar-Query
ADOQuery2: TADOQuery;      //Partner-Query
ADOQuery3: TADOQuery;      //Vorgänger-Query
ADOQuery4: TADOQuery;      //Verpackungs-Query
ADOQuery5: TADOQuery;      //Karten-Query
ADOQuery7: TADOQuery;      //Karten-Produkt-Query
```

```
ADOConnection1: TADOConnection;
   Property Connection String =
     Provider=Microsoft.Jet.OLEDB.3.51;Persist Security Info=False;User ID=Admin;Data
     Source=rap.mdb;Mode=Share Deny None;Extended
     Properties=";COUNTRY=0;CP=1252;LANGID=0x0409";Locale Identifier=1033;Jet
     OLEDB:System database="";Jet OLEDB:Registry Path="";Jet OLEDB:Database
     Password="";Jet OLEDB:Global Partial Bulk Ops=2
Property LoginPrompt = false
```

ADOTable1: TADOTable; //Lebensmittel
Steuert die ganze Lebensmittel-Maske (Tab1). Und einige externe Rapport-Tabellen. Keine Besonderheiten.

ADOTable2: TADOTable; //Abteilung
Steuert nur die Anzeige der Abteilungen (Tab1). Und einige externe Rapport-Tabellen. Keine Besonderheiten.

ADOTable4: TADOTable; //Gruppen
Steuert die Anzeige der Konsumenten (Tab3), und zwar der Bevölkerungsgruppen (oben); sowie einige externe Rapport-Tabellen.
Property Filter = Basis = True

ADOTable5: TADOTable; //Gruppen
Steuert die Anzeige der Konsumenten (Tab3), und zwar der sozio-ökonomischen Gruppen (unten); sowie einige externe Rapport-Tabellen.
Property Filter = Basis = False

ADOTable6: TADOTable; //Karten
Steuert die Berechnung der Kartendaten; sowie einige externe Rapport-Tabellen. Keine Besonderheiten.

ADOTable3: TADOTable; //Quellen
Steuert nur einige externe Rapport-Tabellen. Keine Besonderheiten.

ADOTable7: TADOTable; //Temporary
Wird als temporäre Tabelle verwendet, welche nicht zu einer bestimmten Tabelle in RAP.MDB aufrechterhalten wird. Sie wird für verschiedene Zwecke verwendet, besonders aber im Zusammenhang eines Refresh. Keine Besonderheiten.

ADOTable10: TADOTable; //Gruppen
Steuert nur einige externe Rapport-Tabellen und einer Berechnungen. Keine Besonderheiten.

ADOTable11: TADOTable; //Lebensmittel temporary
Ist eine alternative Verbindung zur Tabelle Lebensmittel, die unabhängig von ADOTable1 verwendet werden kann. Sie wird auch meist im Zusammenhang eines Refresh verwendet. Keine Besonderheiten.

ADOQuery1: TADOQuery; //Nachfolger-Query
Dieser Query ist verantwortlich für die Anzeige der Nachfolger eines Produkts (Tab1).
Property CurserLocation = clUseServer
Property CurserType = ctKeySet
Property SQL =

```
SELECT V.Ausgang, V.Endprodukt, L.Bezeichnung+", "+L.Verw, V.Anteil
FROM Verwendung V INNER JOIN Lebensmittel L ON (V.Endprodukt = L.ID)
WHERE (V.Ausgang = :prod)
ORDER BY L.IDsort;
```

ADOQuery2: TADOQuery; //Partner-Query
Dieser Query ist verantwortlich für die Anzeige der Partner eines Produkts (Tab1).
Property CurserLocation = clUseServer
Property CurserType = ctKeySet
Property SQL =

```
SELECT V.Ausgang, V.Partner, L.Bezeichnung + " ", V.Verhaeltnis
```

```
        FROM Verhaeltnis AS V INNER JOIN Lebensmittel L ON (V.Partner=L.ID)
        WHERE (V.Ausgang = :prod);

ADOQuery3: TADOQuery;      //Vorgänger-Query
      Dieser Query ist verantwortlich für die Anzeige der Vorgänger
      eines Produkts (Tab1).
   Property CurserLocation = clUseServer
   Property CurserType = ctKeySet
   Property SQL =
        SELECT V.Ausgang, V.Endprodukt, L.Bezeichnung+", "+L.Verw, V.Anteil
        FROM Verwendung V INNER JOIN lebensmittel L ON (V.Ausgang = L.ID)
        WHERE (V.Endprodukt = :prod)
        ORDER BY L.IDsort;

ADOQuery4: TADOQuery;      //Verpackungs-Query
      Dieser Query ist verantwortlich für die Anzeige der
      Verpackungen eines Produkts (Tab1).
   Property CurserLocation = clUseServer
   Property CurserType = ctKeySet
   Property SQL =
        SELECT LID, Menge, Kommentar
        FROM Verpackung AS V INNER JOIN Lebensmittel AS L ON L.ID=V.LID
        WHERE (V.LID= :prod);

ADOQuery5: TADOQuery;      //Karten-Query
      Dieser Query ist verantwortlich für die Anzeige der Karten
      (Tab2) (oben)
   Property CurserLocation = clUseServer
   Property CurserType = ctKeySet
   Property SQL =
        SELECT ID, Bezeichnung, Code, Konform, EnerTot, sTot, EnerEi,
          sEi, EnerFett, sFett, EnerFettS, sFettS, Ener,
          INT(4100*Ei/Ener)/10, INT(9300*Fett/Ener)/10, INT(9300*FettS/Ener)/10
        FROM Karten
        ORDER BY ID;

ADOQuery7: TADOQuery;      //Karten-Produkt-Query
      Dieser Query ist verantwortlich für die Anzeige der PI- und ZM-
      Werte zweier Karten, produktmässig aufgelistet (Tab2, unten).
   Property CurserLocation = clUseServer
   Property CurserType = ctKeySet
   Property SQL =
        SELECT L.ID+0,L.Bezeichnung+"",a1.PI,a1.ZM,a2.PI,a2.ZM,a2.PIdef,a2.ZMdef,a2.ZMopt,
            IIF(a2.ZM=0,0,INT(100*a2.ZMopt/a2.ZM)), L.Zuteil+0,INT(L.Zuteil*L.Kal/1000),
            L.Verpackung1+""
        FROM a1,a2,Lebensmittel AS L
        WHERE L.ID=a1.c AND L.ID=a2.c
        ORDER BY L.IDsort;
```

Der Grund, wieso die *CursorLocation* der Queries jeweils auf *clUseServer* eingestellt ist, besteht darin, dass eine *Post-Message* im wesentlich von Access übernommen wird; sonst ist der Query jeweils *nicht-updateable*. Dadurch werden alle Queries *updateable* (mit Ausnahme von *Query7*, bei dem die Post-Message abgefangen wird und durch andere Update-Methoden ersetzt ist).
Die ersten vier Queries sind parameterisiert, jeweils mit dem Parameter *:para*. Dieser muss vor jeder Ausführund des jeweiligen Queries zugewiesen werden über das Konstrukt

```
ADOQuery1.Parameters[0].Value:= <Wert>;
```

Der letzte Query (*Query7*) wird jeweils vor jeder Ausführung neu konstruiert. Es ist der einzige Query, welcher die in Access abgelegten Queries *a1* bis *a9* verwendet.

Eine zweite Verbindung mit der SCEN.MDB wird über folgende ADO-Komponenten aufrechterhalten:

```
    ADOConnection2: TADOConnection;  //Verbindung zu SCEN.MDB
    ADOTemp: TADOTable;              //Temorary Table
```

```
ADOConnection2: TADOConnection;
   Property Connection String =
     Provider=Microsoft.Jet.OLEDB.3.51;Persist Security Info=False;User ID=Admin;Data
     Source=scen.mdb;Mode=Share Deny None;Extended
     Properties=";COUNTRY=0;CP=1252;LANGID=0x0409";Locale Identifier=1033;Jet
     OLEDB:System database="";Jet OLEDB:Registry Path="";Jet OLEDB:Database
     Password="";Jet OLEDB:Global Partial Bulk Ops=2
   Property LoginPrompt = false
```

```
ADOTemp: TADOTable;     //Temporary Table
```

Wird für verschiedene Zwecke verwendet, meist im Zusammenhang mit dem Lesen der Szenario-Namen und andern Haupteinträgen Keine Besonderheiten.

Jede Tabelle und Query enthält verschiedene *Message Handler*, welche dafür verantwortlich sind, was vor oder nach einem *Insert*, *Delete*, oder *Post* getan werden soll. Die Namen der Handler sind leicht zu identifizieren. Zum Beispiel wird vor einen *Post* des *Query7* der Handler

```
procedure ADOQuery7BeforePost(DataSet: TDataSet);
```

ausgeführt.
Der LPL-Code wurde direkt in die Applikation eingebunden und ist mit LPL-Compiler Code über die Funktionen in der Datei LPLRAP.PAS verbunden. Dieser Code exportiert nur eine einzige Funktion (`Optimize`), die in der Applikation verwendet wird. Die Kommunikation zwischen dem LPL-Compiler und RAP.EXE ist im folgenden aufgelistet:

```
unit LPLrap;
interface
uses env,lpl3,Windows, SysUtils;
type TProc=procedure;
     str8=string[8];
var
  ErrorOnExit:boolean;
  msg:Tmsg;


procedure LPLinit(Fn:pchar;maxt,maxa,maxr:integer; cb,sb:TProc);
function  LPLcomp(opt:str8):integer;
procedure LPLexit;
function  LPLwhere:integer;
procedure SetError(n:integer);
procedure SetWorkingDir(dir:string);

procedure Optimize;

implementation
uses main;

var err:integer;

procedure LPLinit(Fn:pchar;maxt,maxa,maxr:integer; cb,sb:TProc);
begin lpl3.LPLinit(Fn,maxt,maxa,maxr,cb,sb); end;
function  LPLcomp(opt:str8):integer; begin LPLcomp:=LPLcompile(opt); end;
procedure LPLexit; begin LPLcleanup; end;
```

```
function LPLwhere:integer; begin LPLwhere:=lpl3.LPLwhere; end;
procedure SetError(n:integer); begin LPLsetError(n); end;
procedure SetWorkingDir(dir:string); begin SetTmpDir(dir); end;


procedure LPLcallBack;
begin
  {case LPLwhere of
    0: MainForm.Label2.Caption:='No model';
    1: MainForm.Label2.Caption:='LPL kernel initialized';
    2: MainForm.Label2.Caption:='Model parsed';
    3: MainForm.Label2.Caption:='Model ran';
    4: MainForm.Label2.Caption:='Model ran/instance created';
    6: MainForm.Label2.Caption:='parsing...';
    7: MainForm.Label2.Caption:='running...';
    8: MainForm.Label2.Caption:='constraint generating...';
    9: MainForm.Label2.Caption:='solving...';
  end;}
  MainForm.Update;

  if PeekMessage(msg,0,0,0,pm_remove) then begin
    DisPatchMessage(msg);
  end;
  if ErrorOnExit then begin {SetError(599);} ErrorOnExit:=false; end;
end;

procedure MyWriteToLog(var s:pChar);  stdcall;
begin
  MainForm.Memo1.Lines.Add(s);
end;

procedure Optimize;
begin
  MainForm.Label45.Caption:='';
  ErrorOnExit:=false;
  MainForm.Button1.Caption:='STOP';
  //SetWorkingDir(WORKDIR);
  LPLinit('rap.lpl',50000,50000,50000,LPLcallBack,nil);
  err:=LPLcomp('w');
  if err=0 then begin
    MainForm.Label45.Caption:='Die Optimierung war erfolgreich!'
  end else begin
    MainForm.Label45.Caption:='Die Optimierung war nicht erfolgreich!'
  end;
  LPLexit;
  MainForm.Button1.Caption:='Optimieren';
  ErrorOnExit:=false;
end;

begin
  WriteToLog:=MyWriteToLog;
end.
```

Alle andern Funktionen in MAIN.PAS sollten selbstsprechend sein. Der gesamte Pascal Code der MAIN.PAS Datei ist im *Anhang B* aufgelistet.

Die gesamte Datenbank-Kommunikation zwischen RAP.EXE und den beiden Datenbanken, welche über die ADO-Komponenten läuft, besitzt eine Besonderheit: Da Access 97 offensichtlich nicht genügend schnell gewisse Transaktionen durchführen kann, sind an verschiedenen Stellen Warteschlaufen (in Form von *Message-Boxen*) eingeblendet. Dies ist vor allem kritisch, wenn ein neues Szenario aus der Datenbanl SCEN.MDB in die Arbeitsdatei RAP.MDB übertragen wird. Falls nicht genügend lange gewartet wird, kann nachträglich über das Menü

Bearbeiten/Refresh ein Teil der Transaktion nochmals ausgeführt werden. (Diese Bemerkungen betreffen die Kommunikation zwischen LPL und den Datenbanken nicht, da dort die Kommunikation über *CreateOleObject* bewerkstelligt ist, und davon unabhängig funktioniert.)

Anhang A: LPL-Source Dateien

Datei RAP.LPL

Diese Datei enthält die gesamte Struktur des LPL Modells.

```
MODEL Rationenplanung;
  SET
    Gehalte ALIAS G,G1 := / KA EI FE KH ET FS /;
    Mikro   ALIAS M := / Kalium Magnesium Calcium Eisen /;
    Alle_Produkte ALIAS p,p1 "Produktcode" STRING pN "Produktname";
    Bezuegergruppen ALIAS bg STRING bN "Bezuegercode";
    Karten ALIAS k,k1 STRING kN "Kartencode";
  PARAMETER
    NUMMER{p} ALIAS NU;
  SET
    Ausgang{p,p1 | NU<300} "Ausgangsprodukte";
    Gruppe{p,p1 | NU>=300} ALIAS g;
  PARAMETER
    FAKTOR{G,G1} := / EI KA 4.1 , FE KA 9.3 , FS FE 1 /;
    MENGE{p} "Verfuegbare Mengen (t)";
    SEK{p} "kurzfristige Soll-Entnahmen";
    SEL{p} "langfristige Soll-Entnahmen";
    RATIONIERT{p} ALIAS rat;
    BEDARF{p,p1 | NU<300};
    UMRECHNUNGSFAKTOR{p,p1 | NU>=300} ALIAS UM;
    BASIS{bg};
    BEZUEGER{bg | BASIS};
    sBEZUEGER{bg | ~BASIS};
    ENERGIEGEHALT{k};
    VERHAELTNIS{p,p1};
    ANZAHL{bg,k} "Karten pro Bezugskategorien";
    GEHALTE_TAB{p,G} ALIAS GT "Gehalte pro Produkt";
    MIKRO_GEHALTE{p,M} ALIAS MG "Mikro Gehalte pro Produkt";
    VORGABE{G,G1,k};
    ZIELMENGE{p,k} ALIAS ZM;
    PI{p,k} "Praesenzindizes";
  STRING
    Verw{p} "Verwendung";
    RTYP{k};
    TYP{G,G1,k};

  READ{p} FROM 'rap,SELECT * FROM Lebensmittel ORDER BY IDsort' :
    p  = 'ID',
    NU = 'ID',
    pN = 'Bezeichnung',
    Verw = 'Verw',
    SEK = 'kSollEntn',
    SEL = 'lSollEntn',
    MENGE = 'verfMenge',
    rat = 'Rat',
    GT[p,1] = 'Kal',
    GT[p,2] = 'Ei',
    GT[p,3] = 'Fett',
    GT[p,4] = 'KH',
    GT[p,5] = 'ET',
    GT[p,6] = 'FS',
    MG[p,1] = 'K',
    MG[p,2] = 'Mg',
    MG[p,3] = 'Ca',
    MG[p,4] = 'Fe';

  READ{p,p1} FROM ',Verwendung' :
    p  = 'Ausgang',
```

```
  p1 = 'Endprodukt',
  Ausgang = ('Ausgang','Endprodukt'),
  BEDARF  = 'Anteil',
  Gruppe[p1,p] = ('Endprodukt','Ausgang'),
  UM[p1,p] = 'Anteil';

READ{bg} FROM ',SELECT * FROM Gruppen ORDER BY ID' :
  bg = 'ID',
  bN = 'Code',
  BASIS = 'Basis',
  BEZUEGER = 'Anzahl',
  sBEZUEGER = 'Anzahl';

READ{k} FROM ',SELECT * FROM Karten ORDER BY ID' :
  k = 'ID',
  kN = 'Code',
  RTYP = 'sTot',
  ENERGIEGEHALT = 'EnerTot',
  VORGABE['EI','KA',k] = 'EnerEi',
  VORGABE['FE','KA',k] = 'EnerFett',
  VORGABE['FS','FE',k] = 'EnerFettS',
  TYP['EI','KA',k] = 'sEi',
  TYP['FE','KA',k] = 'sFett',
  TYP['FS','FE',k] = 'sFettS';

READ{p,p1} FROM ',Verhaeltnis':
  p = 'Ausgang', p1 = 'Partner',
  VERHAELTNIS = 'Verhaeltnis';

READ{p,k} FROM ',Lebensmittel_Karten' :
  p = 'lID',
  k  = 'kID',
  ZM = 'ZM',
  PI = 'PI';

READ{bg,k} FROM ',Gruppen_Karten' :
  bg = 'gID', k = 'kID',
  ANZAHL = 'Anzahl';

SET
  Verfueg{p} ALIAS v "Verfuegbare Produkte" := SEL>0;
  Produkt{p} ALIAS q "Produkte die zur Verteilung gelangen" := Verw='E' or Verw='M';
  Kartenbelegung{p} ALIAS kb,kb1 := Verw='E' or Verw='M' or Verw='G';
  Bezugskategorie{bg} ALIAS bk := BASIS;
  Zuschlag{k} ALIAS z "Zuschlaege fuer Untergruppen" := 1;

  Lagerbare{p} := (MENGE>SEK) AND Verfueg ;
  g1{p} "Alkoholika" := NU<100 AND Lagerbare ;
  g2{p} "Mehlspeisen 2ter Kategorie & Kolonialwaren (Importprodukte)"
    := NU>310 AND NU<500 AND Lagerbare ;
  g3{p} "Eiweissreiche Nahrungsmittel"
    := ~g1 AND ~g2 AND Lagerbare AND 4.1*GT[p,'EI']>0.2*GT[p,'KA'] ;
  g4{p} "Fettreiche Nahrungsmittel"
    := ~g1 AND ~g2 AND ~g3 AND Lagerbare AND 9.2*GT[p,'FE']>4.1*GT[p,'KH'] ;
  g5{p} "Mehlspeisen erste Klasse, kohlenhydratreiche Produkte"
    := ~g1 AND ~g2 AND ~g3 AND ~g4 AND Lagerbare ;

  Lager{p, p1} := p<>p1 AND
    (  g1[p] AND g1[p1] AND p=ARGMAX{k=g1} MENGE[k]*GT[k,'KA']
    OR g2[p] AND g2[p1] AND p=ARGMAX{k=g2} MENGE[k]*GT[k,'KA']
    OR g3[p] AND g3[p1] AND p=ARGMAX{k=g3} MENGE[k]*GT[k,'KA']
    OR g4[p] AND g4[p1] AND p=ARGMAX{k=g4} MENGE[k]*GT[k,'KA']
    OR g5[p] AND g5[p1] AND p=ARGMAX{k=g5} MENGE[k]*GT[k,'KA'] );

  mG{p} "Hauptgruppen" := EXIST{p1} Gruppe[p,p1];
  Gr{p} "NebenGruppen" := EXIST{p1} Gruppe[p1,p];
  mL{p} "Hauptlager"   := EXIST{p1} Lager[p,p1];
  La{p} "Lagerprodukte" := mL OR EXIST{p1} Lager[p1,p];

PARAMETER
  KARTENZAHL{bg,k} ALIAS KZ "Anzahl Karten pro Bezuegergruppe"
    := IF(bk,ANZAHL,ANZAHL+ANZAHL[1,k]);
  ZUSATZ{k} := SUM{bg|~bk} sBEZUEGER*(KARTENZAHL-KARTENZAHL[1,k])/1000;

  KK{k} := ZUSATZ*1000+SUM{bk} BEZUEGER*ANZAHL;
  MEMBER{p,p1 |EXIST{p2=Alle_Produkte}(Lager[p2,p] AND Lager[p2,p1]) AND p1>p} :=1;
```

```
    GS1{mL} := SUM{p1| Lager[mL,p1]} 1;
    GS{p} := MAX(GS1[p],GS1[ARGMAX{p1} Lager[p1,p]]);
    KALKOR{p,p1 |Lager OR MEMBER} "Energieverhaeltnis" :=
      IF( EXIST{k}(PI[p ,k]<>100 AND PI[p ,k]<>9)
      AND EXIST{k}(PI[p1,k]<>100 AND PI[p1,k]<>9)
      AND SEK[p1]>SUM{k|PI[p1,k]=9 OR PI[p1,k]=100}
IF(PI[p1,k]=9,0.8*ZM[p1,k],ZM[p1,k])/1000000*KK
      AND SEK[p]> SUM{k|PI[p,k]=9 OR PI[p,k]=100}
IF(PI[p,k]=9,0.8*ZM[p,k],ZM[p,k])/1000000*KK ,
      (MAX(GT[p,1],GT[p1,1]))/(MIN(GT[p,1],GT[p1,1])) / GS[p1]
       * ( (1-SEK[p]/MENGE[p])*(1-SEK[p1]/MENGE[p1]) )^2 , 0);

    ZK1{k,k1,kb} "Glaettung zwischen den Karten" :=
      k<k1 AND ~(EXIST{l=Karten} ZK1[k,l,kb] OR EXIST{l=Karten} ZK1[l,k1,kb])
      AND PI[kb,k]>0 AND PI[kb,k]<=10 AND PI[kb,k1]>0 AND PI[kb,k1]<=10;
    f{kb} := MAX{k,k1} IF(ZK1,k1);
    h{kb} := MIN{k,k1} IF(ZK1,k,10);
    ZK{k,k1,kb} := ZK1 OR (k=f AND k1=h);

    AK1{k,kb,kb1} "Glaettung auf den Karten" :=
      kb<kb1 AND ~(EXIST{l=Kartenbelegung} AK1[k,kb,l] OR EXIST{l=Kartenbelegung}
AK1[k,l,kb1])
      AND PI[kb,k]>0 AND PI[kb,k]<=10 AND PI[kb1,k]>0 AND PI[kb1,k]<=10;
    f1{k} := MAX{kb,kb1} IF(AK1,kb1);
    h1{k} := MIN{kb,kb1} IF(AK1,kb,100);
    AK{k,kb,kb1} := AK1 OR (kb=f1 AND kb1=h1);

  VARIABLE
    VLprod{p} "Verteilte Menge des Produkts in einem Monat (in 10000 t)";
    RLprod{p} "Menge des Produkts an Lager am Ende des Monats (in 10000 t)";
    VxLprod{bk,p} "Menge des Produkts welches an Bezugskategorie verteilt wird (in
kg)";
    VKxLprod{z,p} "Menge des Produkts welches über Zusatzkarten verteilt wird (in
10000 t)";
    Kxprod{k,p} "Menge des Produkts auf der Karte (in kg)";
    KxGy{k,G} "Menge des Gehalts auf der Karte (in 1000 kcal oder kg)";
    FL;
    EL;
    IFL;
    IFGr;
    IFGy;
    IFLprodN{v};         --in 10000 t
    IFKxGrupN{k,mG};     --in kg
    IFKxGrupP{k,mG};     --in kg

    IFKxGyP{k,G};                    --in 1000 kcal
    IFKxGyN{k,G};                    --in 1000 kcal

  CONSTRAINT
    -- 1.1 Lagerbilanz der verfügbaren Produkte
    RLprodT{v} "Lagerbilanz der verfügbaren Produkte" :
      RLprod + VLprod - IFLprodN = MENGE/10000;
    RLTZ: IFL = SUM{v} IFLprodN;
    -- 1.2 Berechnung der zu verteilenden Menge eines Produktes in einem Monat
    RAprodD{p|EXIST{p1} Ausgang} "Menge der benötigten Ausgangsprodukte in einem
Monat" :
      VLprod = SUM{p1|Ausgang} VLprod[p1]*BEDARF/100;
    RMprodD{mG} "Berechnung der Menge einer Produktgruppe":
      VLprod = SUM{p1|g[mG,p1]} VLprod[p1]/(UM[mG,p1]/100);
    RVLprodD{q} "Berechnung der Menge eines Produktes aus dem Basisbezug pro
Bezügerkategorie und der Verteilung über Zusatzkarten":
      VLprod = SUM{bk} BEZUEGER*VxLprod/10000000 + SUM{z} VKxLprod;
    -- 1.3 Basiszuteilung der Produkte pro Bezügerkategorie
    RxMgrupD{mG,bk} "Basiszuteilung der Produktgruppen":
      VxLprod = SUM{p1|g[mG,p1]} VxLprod/(UM[mG,p1]/100);
    RKxLprod{bk,q} "Berechnung der Basiszuteilung einer Bezügerkategorie aufgrund der
Anzahl Karten pro Bezügerkategorie" :
      VxLprod = SUM{k} Kxprod*ANZAHL;
    -- 1.4 Menge der Produkte auf den Zusatzkarten für sozio-professionelle Gruppen
    RKxMgrupZ{mG,z} "Menge der Produktgruppen auf den Zusatzkarten":
      VKxLprod = SUM{p1|g[mG,p1]} VKxLprod/(UM[mG,p1]/100);
    RKKxLprod{z,q} "Verteilung der Produkte auf sozio-professionelle Gruppen" :
      VKxLprod = Kxprod*ZUSATZ/10000;
    -- 1.5 Verteilung der Produkte auf die Rationenkarten
    RKxMgrup{k,mG} "Menge der Produktgruppen auf den Rationenkarten":
```

```
      Kxprod = SUM{p1|g[mG,p1]} Kxprod/(UM[mG,p1]/100) + IFKxGrupP[k,mG] -
IFKxGrupN[k,mG];
    RKxMPxZ: IFGr = SUM{k,mG} (IFKxGrupP + IFKxGrupN);
    RKxGyD{k,G} "Energie- und Nährstoffgehalt pro Karte ": KxGy = SUM{q}
(Kxprod/1000)*GT;
    -- Nährstoffgehalte pro Karte
    RKxGxGyGE{G,G1,k |TYP='>'}: KxGy[k,G]*FAKTOR + IFKxGyP[k,G] >=
KxGy[k,G1]*VORGABE[G,G1,k]/100;
    RKxGxGyLE{G,G1,k |TYP='<'}: KxGy[k,G]*FAKTOR - IFKxGyN[k,G] <=
KxGy[k,G1]*VORGABE[G,G1,k]/100;
    RKxGxGyEQ{G,G1,k |TYP='='}: KxGy[k,G]*FAKTOR + IFKxGyP[k,G] - IFKxGyN[k,G] =
KxGy[k,G1]*VORGABE[G,G1,k]/100;
    --Energiegehalte pro Karte
    RKxKAGE{k |RTYP[k]='>'}: KxGy[k,'KA'] + IFKxGyP[k,'KA'] >= ENERGIEGEHALT[k]*
30/1000;
    RKxKALE{k |RTYP[k]='<'}: KxGy[k,'KA'] - IFKxGyN[k,'KA'] <= ENERGIEGEHALT[k]*
30/1000;
    RKxKAEQ{k |RTYP[k]='='}: KxGy[k,'KA'] + IFKxGyP[k,'KA'] - IFKxGyN[k,'KA']  =
ENERGIEGEHALT[k]*30/1000;
    RKXGYZ "Gehaltsschlupf": IFGy = SUM{k,G} (IFKxGyP + IFKxGyN);

    SFrisch "Frischprodukte an Lager" : FL = SUM{v |(MENGE AND (MENGE = SEK))} RLprod;
    SEnergie "Energie an Lager": EL=SUM{v |(MENGE AND (MENGE<>SEK))}
RLprod*GT[v,'KA'];

  -- 2. Glättungen
  VARIABLE
    SL1L2P{p,p1 |KALKOR};
    SL1L2N{p,p1 |KALKOR};
    XL1L2P{p,p1 |KALKOR};
    XL1L2N{p,p1 |KALKOR};
    ZL1L2P{p,p1 |KALKOR};
    ZL1L2N{p,p1 |KALKOR};
    K1K2ProdP{k,k1,kb |ZK};
    K1K2ProdN{k,k1,kb |ZK};
    PxP1P2{k,kb,kb1 |AK>0};
    NxP1P2{k,kb,kb1 |AK>0};
    KxProdP09{k,p |PI>0 AND PI<10};
    KxProdN09{k,p |PI>0 AND PI<10};
    IFSLprodP{mL};
    IFSLprodN{mL};
    IFSMprodP{mL};
    IFSMprodN{mL};
    IFSHprodP{mL};
    IFSHprodN{mL};
    ES;
    SL;
    SLG;
    AZK;
    AAK;
    A09;
    UPK;

  CONSTRAINT
    -- 2.1 Glättung der Lagerentnahmen
    RSL1L2{p,p1 |KALKOR}:
     (VLprod*10000/(SEK+0.001)) +SL1L2P +XL1L2P +ZL1L2P =
     (VLprod[p1] * 10000/(SEK[p1]+0.001)) + SL1L2N +XL1L2N +ZL1L2N;
    RSSP{p,p1 |KALKOR}: SL1L2P <= 0.1;
    RSSN{p,p1 |KALKOR}: SL1L2N <= 0.1;
    RSXP{p,p1 |KALKOR}: XL1L2P <= 0.2;
    RSXN{p,p1 |KALKOR}: XL1L2N <= 0.2;
    RSLG: SL = SUM{p,p1 |KALKOR} (SL1L2P + SL1L2N + 3*(XL1L2P+XL1L2N) + 10*(ZL1L2P +
ZL1L2N))*KALKOR;
    -- Substitutionen
    RLSlg{mL}: (VLprod*GT[mL,1] + SUM{p1|Lager[mL,p1]} VLprod[p1]*GT[p1,1]) /
      (SEK[mL]/10000*GT[mL,1] + SUM{p1|Lager[mL,p1]}(SEK[p1]/10000*GT[p1,1]))
     + IFSLprodN - IFSLprodP + IFSMprodN - IFSMprodP + IFSHprodN - IFSHprodP = ES;
    RLSLlgN{mL}: IFSLprodN <= 0.1;
    RLSLlgP{mL}: IFSLprodP <= 0.1;
    RLSMlgN{mL}: IFSMprodN <= 0.1;
    RLSMlgP{mL}: IFSMprodP <= 0.1;
    RSLGT: SLG = SUM{mL} (IFSLprodN + IFSLprodP + 3*(IFSMprodN + IFSMprodP) +
10*(IFSHprodN + IFSHprodP));
    RxLag{p| La AND SEK<SUM{k|PI=9 OR PI=100} ZM[p,k]/1000000*KK}:
      SUM{k| PI<>9 AND PI<>100} Kxprod = 0;
```

```
    -- 2.2 Glättung der Verteilung auf die Rationenkarten
     -- Ausgleich zwischen den Karten
    RK1K2Prod{k,k1,kb |ZK}: Kxprod[k1,kb] * (1000/(ZM[kb,k1]+0.0001)) + K1K2ProdN
      = Kxprod*(1000/(ZM+0.0001)) + K1K2ProdP;
    RSZK: AZK = SUM{k,k1,kb | ZK>0}
      ((((PI[kb,k]+PI[kb,k1]))/(ABS(PI[kb,k]-PI[kb,k1])+4)) * K1K2ProdP
     + (((PI[kb,k]+PI[kb,k1]))/(ABS(PI[kb,k]-PI[kb,k1])+4)) * K1K2ProdN);
    -- Ausgleich auf den Karten
    KxP1P2{k,kb,kb1 |AK>0}:
      Kxprod*(1000/(ZM+0.0001)) + PxP1P2 = Kxprod[k,kb1]*(1000/(ZM[kb1,k]+0.0001)) +
NxP1P2;
    RSAK: AAK = SUM{k,kb,kb1 |AK}
      ((((PI[kb,k]+PI[kb1,k]))/(ABS(PI[kb,k]-PI[kb1,k])+4)) *  PxP1P2
     + (((PI[kb,k]+PI[kb1,k]))/(ABS(PI[kb,k]-PI[kb1,k])+4)) * NxP1P2);
    -- Annäherung der optimalen Mengen an die gewünschten Mengen
    RKxProd09{k,kb |PI>0 AND PI<10}: Kxprod*(1000/(ZM+0.0001)) + KxProdP09 - KxProdN09
=1;
    RS09: A09 = SUM{k,kb |PI>0 AND PI<10} (PI*KxProdP09 + PI*KxProdN09);
    RKxProd9P{k,kb |PI=9}: KxProdP09 <= 0.2;
    RKxProd9N{k,kb |PI=9}: KxProdN09 <= 0.2;
    RKxProd100{k,kb |PI=100}: Kxprod = ZM/1000;
    RKxProd10{k,kb |PI=10}: Kxprod <= ZM/1000;
    RS_1: UPK = SUM{k,kb |PI=-1} Kxprod;

  -- 3. Verhältnisse
  VARIABLE
    VP1P2P{p,p1 |VERHAELTNIS};
    VP1P2N{p,p1 |VERHAELTNIS};
    IFV;

  CONSTRAINT
    RVP1P2{p,p1 |VERHAELTNIS}: VLprod[p] - VLprod[p1]*VERHAELTNIS/100 + VP1P2P -
VP1P2N =0;
    RVP1P2Z: IFV = SUM{p,p1|VERHAELTNIS} (VP1P2P + VP1P2N);

  -- 5. Zielfunktion
  CONSTRAINT
    zfsmooth FREEZE: 100*EL - 999999999*FL - 5000*SL - 250000*SLG - 1000*UPK - 200*AZK
- 20*AAK - 0.01*A09 - 9999*IFV - 99999999999*IFL - 99999999*IFGy - 99999999999*IFGr;

(*$I 'raprepo.lpl' *)


BEGIN  -- Hauptmodell
  MAXIMIZE obj: zfsmooth;
  Report;
  WRITE TO 'xxx.txt' : zfsmooth; WRITE: 9681006.7772; WRITE : 9687654.1090;
END
```

Die Datei RAPRAPO.LPL

Die RAPRAPO.LPL Datei enthält alle WRITE-Statements. Und zwar diejenigen, welche die RAP.NOM Datei generiert, als auch diejenigen, welche direkt in die RAP.MDB zurückschreiben (am Ende der Datei).

```
(* included model into RAP.LPL *)
MODEL Report "Generiert alle Resultate-Tabellen";

---------------------------------
-- 1. Konsistenzprüfung
---------------------------------
WRITE  -- Kontrolle
">###  Ueberbeanspruchung der Lager

Achtung !
---------

Von folgenden Produkten wurde mehr verteilt als vorhanden ist:

     Code                                  in t
     $$$ $$$$$$$$$$$$$$$$$$$$         #########

>###  Produktgruppenbilanz
```

```
Achtung !
---------

Summe der Gruppenprodukte !!


     Karte  Code                        zu hoch in g   zu tief in g
     $$$$$  $$$ $$$$$$$$$$$$$$$$$$$      #########      #########

>###  Verhaeltnisverletzungen

Achtung !
---------

Verletzung der gewünschten Verhältnisse !!

     Code                      Code                  gefordert
$$$  $$$ $$$$$$$$$$$$$$$$$   : $$$$    =  ###.##     ( ###.##)

>###  Gehaltsverletzungen

Achtung !
---------

Ernährungsphysiologische Anforderungen nicht erfüllt !!

                                        in kcal oder g
     Karte  Gehalt                    zu hoch        zu tief
     $$$$$  $$$$$$                  #########      #########

>###  Zuteilung > Sollentnahme

Achtung !
---------

Bei folgenden Lagerprodukten wird infolge PI=9 oder PI=100 über mehr als
die kurzfristig erwünschte Sollentnahme verfügt:
                               Zuteilung    kurzfristige
                                            Sollentnahme
                                 in t           in t
$$$$  $$$$$$$$$$$$$$$$$$$    ########        #######
\n" :

 IF(SUM{v}IFLprodN>0,901,0),
 ROW{v|IFLprodN} (v,pN,IFLprodN*10000),
 IF(SUM{k,mG} (IFKxGrupP+IFKxGrupN)>0,902,0),
 ROW{k,mG|IFKxGrupP OR IFKxGrupN} (kN,mG,pN, IFKxGrupN*1000,IFKxGrupP*1000),
 IF(IFV>0,903,0),
 ROW{p,p1|VP1P2P OR VP1P2N}
   ('bei',p,pN,p1,IF(VLprod[p1]>0,VLprod[p]/VLprod[p1],999.9),VERHAELTNIS/100),
 IF(SUM{k,G}(IFKxGyP+IFKxGyN)>0,904,0),
 ROW{k,G|IFKxGyP OR IFKxGyN}(kN,G,IFKxGyN*1000/30,IFKxGyP*1000/30),
 IF(SUM{p| La AND SEK < SUM{k|PI=9 OR PI=100} ZM/1000000*KK}1>0,905,0),
 ROW{p| La AND SUM{k|PI=9 OR PI=100} ZM/1000000*KK>=SEK}
  (p, pN, SUM{k|PI=9 OR PI=100} IF(PI=9,0.8*ZM,ZM)/1000000*KK, SEK);


  PARAMETER
    ESEK  := SUM{La} (SEK/10000*GT[La,1]);
    EOPT  := SUM{La} VLprod*GT[La,1];
    EFSEK := SUM{v| MENGE AND MENGE=SEK} SEK/10000*GT[v,1];
    EFOPT := SUM{v| MENGE AND MENGE=SEK} VLprod*GT[v,1];
    EnT   := SUM{k}(KxGy[k,1]/10000)*KK;


WRITE  --Kennziffern
">906  Ausgleich Entnahme/Zuteilung

Kennziffern zur Angebots-Nachfrage Ausgleichung

Energieentnahme

Produkt             gemäss Soll-Entnahme       geplant        %-Auswirkung
                      kurzfristig in        in        in %  auf Entnahmesatz
                        10 Mio kcal     10 Mio kcal            bei -1000 t
```

```
$$$ $$$$$$$$$$$$$$$$$$  #########       #########    ###.##     ###.####
_____________________________________________________________________________
Total Lagerprodukte     #########       #########    ###.## (= Entnahmesatz)
Total Frischprodukte    #########       #########    ###.##

=> Resultierendes Energieniveau bei 100% Sollentnahme: #####

Karte     Anzahl      kcal/Tag    10 Mio kcal   in %      Auswirkung auf
                    IST     SOLL    /Monat             Durchschnittsration
                                                       bei +100 kcal/Karte
$$$$   #########  #####    #####  ########     ###.##        ######
_____________________________________________________________________________
Total Ausgabe                     ########

==> Durchschnittsration: #####  (bei ##.## Mio Bezügern)
\n" :

 ROW{La} (La, pN, SEK/10000*GT[La,1], VLprod*GT[La,1],
   100*VLprod*GT[La,1]/(SEK/10000*GT[La,1]), 10*GT[La,1]/ESEK),
 ESEK,EOPT,100*EOPT/ESEK,
 EFSEK,EFOPT,IF(EFSEK,100*EFOPT/EFSEK,0),
 10*(ESEK+EFOPT)/(SUM{bk} BEZUEGER/1000000)/30,
 ROW{k} (kN, KK, KxGy[k,1]*1000/30,
   ENERGIEGEHALT, (KxGy[k,1]/10000)*KK,
   100*(KxGy[k,1]/10000)*KK/EnT,
   KK*100/(SUM{bk} BEZUEGER)),
 EnT,
 (SUM{k}(KxGy[k,1]/1000)*KK) / (SUM{bk} BEZUEGER/1000000)/30,
 SUM{bk} BEZUEGER/1000000;


  PARAMETER MV{kb} := SUM{k} (Kxprod/1000*KK);


WRITE --Energie
">907  Total Energie/Produkt

Verteilung der Energie pro Produkt

I--------------------------I------------I-------------I
I Int                      I Verteilung I Verteilung  I
I Bez  Lebensmittel        I in Tonnen  I in Mio kcal I
I--------------------------I------------I-------------I
I $$$ $$$$$$$$$$$$$$$$$$  I #########  I #########   I
I--------------------------I------------I-------------I
 nicht rationiert
I--------------------------I------------I-------------I
I $$$ $$$$$$$$$$$$$$$$$$  I #########  I #########   I
I--------------------------I------------I-------------I
\n" :

 ROW{kb |RATIONIERT}  (kb,pN,MV,GT[kb,1]*MV/1000),
 ROW{kb |~RATIONIERT} (kb,pN,MV,GT[kb,1]*MV/1000);


---------------------------------
-- 2. R-Tab.1
---------------------------------
WRITE --rtab1
">101  Frischprodukte (R-TAB.1)

Unverteilte Reste von vollständig zu verteilenden Produkten
R-TAB.1

  I--------------------------I------------I--------------------I
  I Int                      I    Zur     I      Unverteilt    I
  I Bez  Lebensmittel        I Verteilung I                    I
  I                          I in Tonnen  I  in Tonnen I in %  I
  I--------------------------I------------I------------I-------I
  I $$$ $$$$$$$$$$$$$$$$$$$$ I #########  I #########  I###.#% I
  I--------------------------I------------I------------I-------I
\n" :

 ROW{v|MENGE AND MENGE=SEK}
   (v,pN,SEK,(RLprod-IFLprodN)*10000,(RLprod-IFLprodN)*1000000/SEK);
```

```
--------------------------------
-- 3. R-Tab.2
--------------------------------
WRITE --rtab2
">201  Lagerbewirtschaftung (R-TAB.2)

Bewirtschaftung der lagerbaren Produkte:
verfuegbare Lebensmittel / Soll-Lagerentnahme / optimale Lagerentnahme

R-TAB.2

 I------------------------I--------------I---------I-----------------I----------I-----
---------I--------------I
 IInt.                       I Lagerbestand I Lager-  I Sollentnahmen    I Lager-   I
Lagerbestand I Lagerent-    I
 IBez. Produkt             I am Anfang    I erstr.  I kurz-1   lang-2 I entnahme I am
Ende      I nahme bzgl   I
 I                           I        I      I          I     fristig     I optimal  I
I      I Soll in %    I
 I                                  ITonnen  I  Mte.  I  Mte.      I  Tonnen  I  Tonnen  I
ITonnen I Mte. I kurzf Ilangf I
 I------------------------I-------I------I---------I--------I--------I----------I-----
--I------I-------I------I
 I  $$$  $$$$$$$$$$$$$$$$$$  I#######I###.#  I  ###.#     I#######  I#######  I########
I#######I###.# I ###.# I ###.#I
 I------------------------I-------I------I---------I--------I--------I----------I-----
--I------I-------I------I
 1) Sollvorgaben aus der Sicht der Lagerbewirtschaftung gemäss:
                             - erwarteten Lagerzugängen
                             - gewünschter Lagererstreckung
                             - Zwang zur Auslagerung
                             - Wunsch zur generellen Erhöhung in den Rationen
 2) Mittlere, vorgesehene Ergänzung der Frischproduktion aus der Sicht
    Ernährungsplanung
\n" :

 ROW{La}(La, pN, MENGE,
 IF(MENGE/SEL>=1000,999.9,MENGE/SEL),
 MENGE/SEK, SEK, SEL, VLprod*10000,
 MENGE-VLprod*10000,
 IF((MENGE-VLprod*10000)/SEL>=1000,999.9,(MENGE-VLprod*10000)/SEL),
 IF(100*VLprod*10000/SEK>=1000,999.9,100*VLprod*10000/SEK),
 IF(100*VLprod*10000/SEL>=1000,999.9,100*VLprod*10000/SEL));

--------------------------------
-- 4. Lagerbuchhaltung
--------------------------------
WRITE --Lagerbuchhaltung
">202  Lagerbuchhaltung

Lagerbuchhaltung (Ueberblick)

I------------------------I--------------I-----------I--------------I
I Int.                   I Lagerbestand I Lager-    I Lagerbestand I
I Bez. Produkt           I Anfang Monat I entnahme  I  Ende Monat  I
I                        I  in Tonnen   I in Tonnen I  in Tonnen   I
I------------------------I--------------I-----------I--------------I
I $$$ $$$$$$$$$$$$$$$$$$ I     #######  I  #######  I     #######  I
I------------------------I--------------I-----------I--------------I

\n" :

 ROW{La} (La,pN,MENGE,VLprod*10000,MENGE-VLprod*10000);


---------------------------------------
-- 5. Einteilung der Lagerprodukte in Lagergruppen
---------------------------------------
WRITE --Lagergruppen
">203  Einteilung in Lagergruppen

Einteilung der Lagerprodukte in Gruppen

  Code Name                      $$$$$$$

Alkoholische Getränke (Code < 100)
```

```
   $$$ $$$$$$$$$$$$$$$$$$  #######

Import-Produkte (310 < Code < 500)
   $$$ $$$$$$$$$$$$$$$$$$  #######

Eiweissreiche Produkte (4.1*Eiweissgehalt > 0.2*Energiegehalt)
   $$$ $$$$$$$$$$$$$$$$$$  #######

Fettreiche Produkte (9.2*Fettgehalt > 4.1*Kohlehydratgehalt)
   $$$ $$$$$$$$$$$$$$$$$$  #######

Kohleydratträger (restliche Lagerprodukte)
   $$$ $$$$$$$$$$$$$$$$$$  #######

\n" :

 COL{G} G,
 ROW{g1} (g1,pN,COL{G} GT),
 ROW{g2} (g2,pN,COL{G} GT),
 ROW{g3} (g3,pN,COL{G} GT),
 ROW{g4} (g4,pN,COL{G} GT),
 ROW{g5} (g5,pN,COL{G} GT);

--------------------------------
-- 6. R-Tab.3 : Zuteilungsmengen nach Karten
--------------------------------
OPTION formatmask ALIAS rtab3a
">###  Kartenzuteilung in $$$$$$ (R-TAB.3 A)

Zuteilungsmengen nach $$$$$$$$$

I----------------------------I-$$$$$$$$$$
IInt. Lebensmittel           I $$$$$$$  $
IBez.                        I $$$$$$$  $
I----------------------------I-$$$$$$$$$$
I $$$ $$$$$$$$$$$$$$$$$$$$$  I #######  $
I----------------------------I-$$$$$$$$$$
Nicht rationierte, zur Restdeckung verfügbare Ware
I----------------------------I-$$$$$$$$$$
I $$$ $$$$$$$$$$$$$$$$$$$$$  I #######  $
I----------------------------I-$$$$$$$$$$


>###  Kartenzuteilung in $$$$$$ (R-TAB.3 A)

Zusammensetzung der Sammelprodukte für die Kalkulation
der Nährstoffversorgung

Int. Lebensmittel             $$$$$$$$
Bez.                          $$$$$$$$
$$$ $$$$$$$$$$$$$$$$$$$$      ########
  $$$ $$$$$$$$$$$$$$$$$$$$    ########
\n" ;


-- r-tab3a : Zuteilungsmengen nach Karten (in Gramm) (Karten 1-5)
WRITE "@rtab3a" :

 301, 'Gramm', 'Karten',
 COL{k|k<6} '---------I',
 COL{k|k<6} (kN,'I'),
 COL{k|k<6} ('Gramm','I'),
 COL{k|k<6} '---------I',
 ROW{kb|RATIONIERT} (kb, pN, COL{k|k<6} (Kxprod*1000,'I')),
 COL{k|k<6} '---------I',
 COL{k|k<6} '---------I',
 ROW{kb|~RATIONIERT} (kb, pN, COL{k|k<6} (Kxprod*1000, 'I')),
 COL{k|k<6} '---------I',

 302, 'Gramm',
 COL{k|k<6} kN,
 COL{k|k<6} 'Gramm   ',
 ROW{mG} (mG,pN, COL{k|k<6} Kxprod*1000,
   ROW{p|Gruppe[mG,p]} (p,pN, COL{k|k<6} Kxprod*1000));
```

```
-- r-tab3a : Zuteilungsmengen nach Karten (in Gramm) (Karten >5)
WRITE "@rtab3a" :
 IF(SUM{k|k>5}1>0,303,0), 'Gramm', 'Karten',
 COL{k|k>5} '---------I',
 COL{k|k>5} (kN,'I'),
 COL{k|k>5} ('Gramm','I'),
 COL{k|k>5} '---------I',
 ROW{kb|RATIONIERT} (kb, pN, COL{k|k>5} (Kxprod*1000,'I')),
 COL{k|k>5} '---------I',
 COL{k|k>5} '---------I',
 ROW{kb|~RATIONIERT} (kb, pN, COL{k|k>5} (Kxprod*1000,'I')),
 COL{k|k>5} '---------I',

 IF(SUM{k|k>5}1>0,304,0), 'Gramm',
 COL{k|k>5} kN,
 COL{k|k>5} 'Gramm   ',
 ROW{mG} (mG,pN, COL{k|k>5} Kxprod*1000,
   ROW{p|Gruppe[mG,p]} (p,pN, COL{k|k>5} Kxprod*1000));


-- r-tab3a : Zuteilungsmengen nach Karten (in Tonnen) (Karten 1-5)
WRITE "@rtab3a" :
 305, 'Tonnen', 'Karten',
 COL{k|k<6} '---------I',
 COL{k|k<6} (kN,'I'),
 COL{k|k<6} ('Tonnen','I'),
 COL{k|k<6} '---------I',
 ROW{kb|RATIONIERT} (kb, pN, COL{k|k<6} (Kxprod/1000*KK,'I')),
 COL{k|k<6} '---------I',
 COL{k|k<6} '---------I',
 ROW{kb|~RATIONIERT} (kb, pN, COL{k|k<6} (Kxprod/1000*KK,'I')),
 COL{k|k<6} '---------I',

 306, 'Tonnen',
 COL{k|k<6} kN,
 COL{k|k<6} 'Tonnen',
 ROW{mG} (mG,pN,COL{k|k<6} Kxprod/1000*KK,
   ROW{p|Gruppe[mG,p]} (p,pN,COL{k|k<6} Kxprod/1000*KK));


-- r-tab3a : Zuteilungsmengen nach Karten (in Tonnen) (Karten >5)
WRITE "@rtab3a" :
 IF(SUM{k|k>5}1>0,307,0), 'Tonnen', 'Karten',
 COL{k|k>5} '---------I',
 COL{k|k>5} (kN,'I'),
 COL{k|k>5} ('Tonnen','I'),
 COL{k|k>5} '---------I',
 ROW{kb|RATIONIERT} (kb, pN,COL{k|k>5} (Kxprod/1000*KK,'I')),
 COL{k|k>5} '---------I',
 COL{k|k>5} '---------I',
 ROW{kb|~RATIONIERT} (kb, pN,COL{k|k>5} (Kxprod/1000*KK,'I')),
 COL{k|k>5} '---------I',

 IF(SUM{k|k>5}1>0,308,0), 'Tonnen',
 COL{k|k>5} kN,
 COL{k|k>5} 'Tonnen',
 ROW{mG} (mG,pN,COL{k|k>5} Kxprod/1000*KK,
   ROW{p|Gruppe[mG,p]} (p,pN,COL{k|k>5} Kxprod/1000*KK));


--------------------------------
-- 7. R-Tab.3a : Zuteilungsmengen nach Gruppen
--------------------------------
OPTION formatmask ALIAS rtab3g
">###  Gruppenzuteilung in Gramm

Zuteilungsmengen nach $$$$$$$$$

I---------------------------I-$$$$$$$$$$
I                           I $$$$$$$  $
I---------------------------I-$$$$$$$$$$
  NAEHRSTOFF- U. ENERGIE ZUFUHR (G/TAG, KCAL/TAG)
I---------------------------I-$$$$$$$$$$
I            ENERGIE        I #####    $
I            PROTEIN        I ###/###  $
I            FETT           I ###/###  $
```

```
I            KOHLEHYDRAT     I #####    $
I----------------------------I-$$$$$$$$$$
 Code Lebensmittel             LEBENSMITTEL - MONATSRATION IN GRAMM   (TOTAL)
I----------------------------I-$$$$$$$$$$
I $$$ $$$$$$$$$$$$$$$$$$$$   I #######  $
I----------------------------I-$$$$$$$$$$
Nicht rationierte, zur Restdeckung verfügbare Ware
I----------------------------I-$$$$$$$$$$
I $$$ $$$$$$$$$$$$$$$$$$$$   I #######  $
I----------------------------I-$$$$$$$$$$


>###  Gruppenzuteilung in Gramm

Zusammensetzung der Sammelprodukte für die Kalkulation
der Nährstoffversorgung

Int. Lebensmittel             $$$$$$$$
Bez.                          $$$$$$$$
$$$ $$$$$$$$$$$$$$$$$$$$      ########
  $$$ $$$$$$$$$$$$$$$$$$$$    ########

";

-- Gruppen 1-5
WRITE "@rtab3g" :
 311, 'Gruppen',
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} (bN,'I'),
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,1]*1000/30,'I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,2]*1000/30,SUM{k} KZ*KxGy[k,5]*1000/30,'I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,3]*1000/30,SUM{k} KZ*KxGy[k,6]*1000/30,'I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,4]*1000/30,'I'),
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} '---------I',
 ROW{kb|~Gr AND RATIONIERT}
 (kb, pN, COL{bg|bg<6} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} '---------I',
 ROW{kb|~Gr AND ~RATIONIERT}
  (kb, pN, COL{bg|bg<6} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg<6} '---------I',

 312,
 COL{bg|bg<6} bN,
 COL{bg|bg<6} 'Gramm   ',
 ROW{mG} (mG,pN,COL{bg|bg<6} SUM{k} KZ*Kxprod*1000,
   ROW{p|Gruppe[mG,p]} (p,pN,COL{bg|bg<6} SUM{k} KZ*Kxprod*1000));

-- Gruppen 6-10
WRITE "@rtab3g" :
 IF(SUM{bg|bg>5 AND bg<11}1>0,313,0), 'Gruppen',
 COL{bg|bg>5 AND bg<11} '---------I',
 COL{bg|bg>5 AND bg<11} (bN,'I'),
 COL{bg|bg>5 AND bg<11} '---------I',
 COL{bg|bg>5 AND bg<11} '---------I',
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,1]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,2]*1000/30,SUM{k} KZ*KxGy[k,5]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,3]*1000/30,SUM{k} KZ*KxGy[k,6]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,4]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} '---------I',
 COL{bg|bg>5 AND bg<11} '---------I',
 ROW{kb|~Gr AND RATIONIERT}
   (kb, pN,COL{bg|bg>5 AND bg<11} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg>5 AND bg<11} '---------I',
 COL{bg|bg>5 AND bg<11} '---------I',
 ROW{kb|~Gr AND ~RATIONIERT}
   (kb,pN, COL{bg|bg>5 AND bg<11} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg>5 AND bg<11} '---------I',

 IF(SUM{bg|bg>5 AND bg<11} 1>0,314,0),
 COL{bg|bg>5 AND bg<11} bN,
 COL{bg|bg>5 AND bg<11} 'Gramm   ',
 ROW{mG} (mG,pN,COL{bg|bg>5 AND bg<11} SUM{k} KZ*Kxprod*1000,
```

```
  ROW{p|Gruppe[mG,p]} (p,pN,COL{bg|bg>5 AND bg<11} SUM{k} KZ*Kxprod*1000));


-- Gruppen 11-15
WRITE "@rtab3g" :
 IF(SUM{bg|bg>10 AND bg<16} 1>0,315,0), 'Gruppen',
 COL{bg|bg>10 AND bg<16} '---------I',
 COL{bg|bg>10 AND bg<16} (bN,'I'),
 COL{bg|bg>10 AND bg<16} '---------I',
 COL{bg|bg>10 AND bg<16} '---------I',
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ*KxGy[k,1]*1000/30,'I'),
 COL{bg|bg>10       AND       bg<16}       (SUM{k}       KZ*KxGy[k,2]*1000/30,SUM{k}
KZ*KxGy[k,5]*1000/30,'I'),
 COL{bg|bg>10       AND       bg<16}       (SUM{k}       KZ*KxGy[k,3]*1000/30,SUM{k}
KZ*KxGy[k,6]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ*KxGy[k,4]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} '---------I',
 COL{bg|bg>10 AND bg<16} '---------I',
 ROW{kb|~Gr AND RATIONIERT}
  (kb,pN,COL{bg|bg>10 AND bg<16} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg>10 AND bg<16} '---------I',
 COL{bg|bg>10 AND bg<16} '---------I',
 ROW{kb|~Gr AND ~RATIONIERT}
  (kb,pN, COL{bg|bg>10 AND bg<16} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg>10 AND bg<16} '---------I',

 IF(SUM{bg|bg>10 AND bg<16}1>0,316,0),
 COL{bg|bg>10 AND bg<16} bN,
 COL{bg|bg>10 AND bg<16}'Gramm   ',
 ROW{mG} (mG,pN,COL{bg|bg>10 AND bg<16} SUM{k} KZ*Kxprod*1000,
  ROW{p|Gruppe[mG,p]} (mG,pN,COL{bg|bg>10 AND bg<16} SUM{k} KZ*Kxprod*1000));


---------------------------------------
-- 8. R-Tab.3b
---------------------------------------
SET kbR{p} := kb AND RATIONIERT AND ~(p IN mG);

OPTION formatmask ALIAS rtab3b
">###  Gehalte $$$$$$$$$

Physiologische Werte der $$$$$$$$ (Pro Tag)
$$$$$$$$$$

I----------------------------I-$$$$$$$$$$
I                            I $$$$$$   $
I----------------------------I-$$$$$$$$$$
I   Energie (kcal)           I ######   $
I      Davon rationiert      I ######   $
I                            I $$$$$$   $
I   Eiweiss (Gramm)          I ######   $
I      Davon rationiert      I ######   $
I                            I $$$$$$   $
I   Tier. Eiweiss (Gramm)    I ######   $
I      Davon rationiert      I ######   $
I                            I $$$$$$   $
I   Fett (Gramm)             I ######   $
I      Davon rationiert      I ######   $
I                            I $$$$$$   $
I   Sichtb. Fett (Gramm)     I ######   $
I      Davon rationiert      I ######   $
I                            I $$$$$$   $
I   Kohlenhydrat (Gramm)     I ######   $
I      Davon rationiert      I ######   $
I                            I $$$$$$   $
I----------------------------I-$$$$$$$$$$

         1000 kcal = 4.19 MJ

";

-- Physiologische Werte der Karten (Karten 1-5)
WRITE "@rtab3b" :
 351, '(Karten)', 'Karten', 'R-TAB.3B',
 COL{k|k<6} '---------I',
 COL{k|k<6} (kN,'I'),
```

```
 COL{k|k<6} '---------I',
 COL{k|k<6} (KxGy[k,1]*1000/30,'I'),
 COL{k|k<6} (SUM{kbR} GT[p,1]*Kxprod/30,'I'),
 COL{k|k<6} ('        ','I'),
 COL{k|k<6} (KxGy[k,2]*1000/30,'I'),
 COL{k|k<6} (SUM{kbR} GT[p,2]*Kxprod/30,'I'),
 COL{k|k<6} ('        ','I'),
 COL{k|k<6} (KxGy[k,5]*1000/30,'I'),
 COL{k|k<6} (SUM{kbR} GT[p,5]*Kxprod/30,'I'),
 COL{k|k<6} ('        ','I'),
 COL{k|k<6} (KxGy[k,3]*1000/30,'I'),
 COL{k|k<6} (SUM{kbR} GT[p,3]*Kxprod/30,'I'),
 COL{k|k<6} ('        ','I'),
 COL{k|k<6} (KxGy[k,6]*1000/30,'I'),
 COL{k|k<6} (SUM{kbR} GT[p,6]*Kxprod/30,'I'),
 COL{k|k<6} ('        ','I'),
 COL{k|k<6} (KxGy[k,4]*1000/30,'I'),
 COL{k|k<6} (SUM{kbR} GT[p,4]*Kxprod/30,'I'),
 COL{k|k<6} ('        ','I'),
 COL{k|k<6} '---------I';


-- Physiologische Werte der Karten (Karten >5)
WRITE "@rtab3b" :
 IF(SUM{k|k>5} 1>0,352,0), '(Karten)','Karten', 'R-TAB.3Bf',
 COL{k|k>5} '---------I',
 COL{k|k>5} (kN,'I'),
 COL{k|k>5} '---------I',
 COL{k|k>5} (KxGy[k,1]*1000/30,'I'),
 COL{k|k>5} (SUM{kbR} GT[p,1]*Kxprod/30,'I'),
 COL{k|k>5} ('        ','I'),
 COL{k|k>5} (KxGy[k,2]*1000/30,'I'),
 COL{k|k>5} (SUM{kbR} GT[p,2]*Kxprod/30,'I'),
 COL{k|k>5} ('        ','I'),
 COL{k|k>5} (KxGy[k,5]*1000/30,'I'),
 COL{k|k>5} (SUM{kbR} GT[p,5]*Kxprod/30,'I'),
 COL{k|k>5} ('        ','I'),
 COL{k|k>5} (KxGy[k,3]*1000/30,'I'),
 COL{k|k>5} (SUM{kbR} GT[p,3]*Kxprod/30,'I'),
 COL{k|k>5} ('        ','I'),
 COL{k|k>5} (KxGy[k,6]*1000/30,'I'),
 COL{k|k>5} (SUM{kbR} GT[p,6]*Kxprod/30,'I'),
 COL{k|k>5} ('        ','I'),
 COL{k|k>5} (KxGy[k,4]*1000/30,'I'),
 COL{k|k>5} (SUM{kbR} GT[p,4]*Kxprod/30,'I'),
 COL{k|k>5} ('        ','I'),
 COL{k|k>5} '---------I';


-- Physiologische Werte für die Gruppen (Gruppen 1-5)
WRITE "@rtab3b" :
 371, '(Gruppen)', 'Gruppen', ' ',
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} (bN,'I'),
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,1]*1000/30,'I'),
 COL{bg|bg<6} (SUM{kbR} GT[p,1]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg<6} ('        ','I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,2]*1000/30,'I'),
 COL{bg|bg<6} (SUM{kbR} GT[p,2]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg<6} ('        ','I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,5]*1000/30,'I'),
 COL{bg|bg<6} (SUM{kbR} GT[p,5]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg<6} ('        ','I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,3]*1000/30,'I'),
 COL{bg|bg<6} (SUM{kbR} GT[p,3]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg<6} ('        ','I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,6]*1000/30,'I'),
 COL{bg|bg<6} (SUM{kbR} GT[p,6]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg<6} ('        ','I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,4]*1000/30,'I'),
 COL{bg|bg<6} (SUM{kbR} GT[p,4]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg<6} ('        ','I'),
 COL{bg|bg<6} '---------I';
```

```
-- Physiologische Werte für die Gruppen (Gruppen 6-10)
WRITE "@rtab3b" :
 IF(SUM{bg|bg>5 AND bg<11} 1>0,372,0), '(Gruppen)', 'Gruppen', 'f ',
 COL{bg|bg>5 AND bg<11} '---------I',
 COL{bg|bg>5 AND bg<11} (bN,'I') ,
 COL{bg|bg>5 AND bg<11} '---------I',
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,1]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{kbR} GT[p,1]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>5 AND bg<11} ('        ','I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,2]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{kbR} GT[p,2]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>5 AND bg<11} ('        ','I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,5]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{kbR} GT[p,5]*(SUM{k} KZ*Kxprod)/30,'I'),

 COL{bg|bg>5 AND bg<11} ('        ','I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,3]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{kbR} GT[p,3]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>5 AND bg<11} ('        ','I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,6]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{kbR} GT[p,6]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>5 AND bg<11} ('        ','I'),
 COL{bg|bg>5 AND bg<11} (SUM{k} KZ*KxGy[k,4]*1000/30,'I'),
 COL{bg|bg>5 AND bg<11} (SUM{kbR} GT[p,4]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>5 AND bg<11} ('        ','I'),
 COL{bg|bg>5 AND bg<11} '---------I';


-- Physiologische Werte für die Gruppen (Gruppen 11-15)
WRITE "@rtab3b" :
 IF(SUM{bg|bg>10 AND bg<16} 1>0,373,0), '(Gruppen)', 'Gruppen', 'f ',
 COL{bg|bg>10 AND bg<16} '---------I',
 COL{bg|bg>10 AND bg<16} (bN,'I') ,
 COL{bg|bg>10 AND bg<16} '---------I',
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ*KxGy[k,1]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} (SUM{kbR} GT[p,1]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>10 AND bg<16} ('        ','I'),
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ*KxGy[k,2]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} (SUM{kbR} GT[p,2]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>10 AND bg<16} ('        ','I'),
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ[bg,k]*KxGy[k,5]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} (SUM{kbR} GT[p,5]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>10 AND bg<16} ('        ','I'),
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ*KxGy[k,3]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} (SUM{kbR} GT[p,3]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>10 AND bg<16} ('        ','I'),
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ*KxGy[k,6]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} (SUM{kbR} GT[p,6]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>10 AND bg<16} ('        ','I'),
 COL{bg|bg>10 AND bg<16} (SUM{k} KZ*KxGy[k,4]*1000/30,'I'),
 COL{bg|bg>10 AND bg<16} (SUM{kbR} GT[p,4]*(SUM{k} KZ*Kxprod)/30,'I'),
 COL{bg|bg>10 AND bg<16} ('        ','I'),
 COL{bg|bg>10 AND bg<16} '---------I';


-------------------------------------
-- 9. R-Tab.3c
-------------------------------------
OPTION formatmask ALIAS rtab3c
">###  Mikronaehrstoffe
$$$$$$$$$$  Versorgung mit Mikronaehrstoffen der Karten (mg/Tag)

I----------------------------I-$$$$$$$$$$
I                            I $$$$$$   $
I----------------------------I-$$$$$$$$$$
I   $$$$$$$$$$               I ######   $
I                            I $$$$$$   $
I----------------------------I-$$$$$$$$$$

\n";

-- Karten 1-5
WRITE "@rtab3c" :
 381, 'R-TAB.3C',
 COL{k|k<6} '---------I',
 COL{k|k<6} (kN,'I'),
```

```
 COL{k|k<6} '---------I',
 ROW{M} (M,COL{k|k<6} (SUM{q} MG*Kxprod/30,'I'),COL{k|k<6} ('        ','I')),
 COL{k|k<6} '---------I';

-- Karten >5
WRITE "@rtab3c" :
 IF(SUM{k|k>5} 1>0,382,0), 'R-TAB.3Cf',
 COL{k|k>5} '---------I',
 COL{k|k>5} (kN,'I'),
 COL{k|k>5} '---------I',
 ROW{M} (M,COL{k|k>5} (SUM{q} MG*Kxprod/30,'I'),COL{k|k>5} ('        ','I')),
 COL{k|k>5} '---------I';


--------------------------------
-- 10. Rationenkarten
---------------------------------------
PARAMETER EPS := 0.00001;
WRITE   --Karte
">###  Rationenkarten $$$$

I---------------------------I-----------------------------------I
I                           I  KCAL  Eiweiss    Fett  Kohlehydr.I
I  $$$$$$$$$$$$$$$$$$$$$    I                                   I
I                           I         in Gramm pro Tag          I
I                           I                                   I
I                           I-----------------------------------I
I                           I                                   I
I                           I  ####  ### (###)  ### (###)   ### I
I                           I                                   I
I---------------------------I-----------------------------------I
I  Int                       Zielmenge   geplant       geplant  I
I  Bez  Lebensmittel                       in %        in Gramm I
I-------------------------------------------------I-------------I
I  $$$ $$$$$$$$$$$$$$$$$$   #####        ###.##  I     #####   I
I                                                I             I
I  Nicht rationierte, zur Restdeckung verfügbare Ware          I
I                                                I             I
I  $$$ $$$$$$$$$$$$$$$$$$   #####        ###.##  I     #####   I
I-------------------------------------------------I-------------I
  Anmerkung: 1000 KCAL = 4.19 MJ
Zusammensetzung der Sammelprodukte für die Kalkulation
der Nährstoffversorgung

                                          Gramm     %
$$$ $$$$$$$$$$$$$$$$$                    #####    ###.#%
  $$$$$$ $$$ $$$$$$$$$$$$$$$$$$          #####    ###.#%

\n" :

 ROW{k}
 (400+k,kN,kN,
 KxGy[k,'KA')*1000/30, KxGy[k,'EI')*1000/30,
 KxGy[k,'ET')*1000/30, KxGy[k,'FE')*1000/30,
 KxGy[k,'FS')*1000/30, KxGy[k,'KH')*1000/30,
 ROW{kb|Kxprod AND ~Gr AND RATIONIERT}
  (kb,pN,ZM,IF(ZM,100*Kxprod*1000/ZM,999.99),Kxprod*1000),
 ROW{kb|Kxprod>EPS AND ~Gr AND ~RATIONIERT}
  (kb,pN,ZM,IF(ZM,100*Kxprod*1000/ZM,999.99),Kxprod*1000),
 ROW{mG|Kxprod>EPS} (mG,pN,Kxprod*1000,100,
  ROW{p|Gruppe[mG,p] AND Kxprod} ('DAVON',p,pN,Kxprod*1000,Kxprod*100/Kxprod)));


------------------------------
-- 11. Bezuegerkarten
------------------------------
SET Spalten := /1:5/;

WRITE --Bezuegerkarte
">###  Bezuegerrationen $$$$

I-----------------------------------I-----------------------------------I
I Erhaelt ueber:                    I  KCAL  Eiweiss    Fett  Kohlehydr.I
I                                   I         in Gramm pro Tag          I
I-----------------------------------I-----------------------------------I
I ####  $$$$$$$$$$$$$$$             I  ####  ### (###)  ### (###)   ### I
```

```
I----------------------------------I-----------------------------------I
I                           Total  I  ####  ### (###)  ### (###)   ### I
I                                  I-----------------------------------I
I Int                                                          geplant  I
I Bez  Lebensmittel        Monatsration in Gramm               in Gramm I
I----------------------------------------------------------I--------------I
I                          $$$$$ $$$$$  I    Total     I
I----------------------------------------------------------I--------------I
I $$$ $$$$$$$$$$$$$$$$$   ##### ##### I     #####       I
I                                                          I              I
I Nicht rationierte, zur Restdeckung verfügbare Ware                      I
I                                                          I              I
I $$$ $$$$$$$$$$$$$$$$$   ##### ##### I     #####       I
I----------------------------------------------------------I--------------I
  Anmerkung: 1000 KCAL = 4.19 MJ

Zusammensetzung der Sammelprodukte für die Kalkulation
der Nährstoffversorgung

                                                 Gramm     %
$$$ $$$$$$$$$$$$$$$$$                           #####    ###.#%
   $$$$$$ $$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  #####    ###.#%

\n" :

 ROW{bg} (500+bg,bN,
  ROW{k|KZ} (KZ,kN,
   KxGy[k,'KA']*1000/30*KZ,
   KxGy[k,'EI']*1000/30*KZ,
   KxGy[k,'ET']*1000/30*KZ,
   KxGy[k,'FE']*1000/30*KZ,
   KxGy[k,'FS']*1000/30*KZ,
   KxGy[k,'KH']*1000/30*KZ),
   SUM{k|KZ} KxGy[k,'KA']*1000/30*KZ,
   SUM{k|KZ} KxGy[k,'EI']*1000/30*KZ,
   SUM{k|KZ} KxGy[k,'ET']*1000/30*KZ,
   SUM{k|KZ} KxGy[k,'FE']*1000/30*KZ,
   SUM{k|KZ} KxGy[k,'FS']*1000/30*KZ,
   SUM{k|KZ} KxGy[k,'KH']*1000/30*KZ,
   COL{k|KZ} kN, COL{s=Spalten|s<6-(SUM{k|KZ}1)} ' ',
   ROW{kb|EXIST{k} Kxprod>EPS AND ~Gr AND RATIONIERT}
     (kb,pN,
     COL{k|KZ} (KZ*Kxprod*1000),
     COL{s=Spalten|s<6-(SUM{k|KZ} 1)} ' ',
     SUM{k} KZ*Kxprod*1000),
     ROW{kb|EXIST{k} Kxprod>EPS AND ~Gr AND ~RATIONIERT}
     (kb,pN,
     COL{k|KZ} (KZ*Kxprod*1000),
     COL{s=Spalten|s<6-(SUM{k|KZ} 1)} ' ',
     SUM{k} KZ*Kxprod*1000),
     ROW{mG|SUM{k} KZ*Kxprod>0}
      (mG,pN,SUM{k} KZ*Kxprod*1000,100,
      ROW{p|Gruppe[mG,p] AND SUM{k} KZ*Kxprod>0}
      ('DAVON',p,pN,SUM{k} KZ*Kxprod*1000,
      (SUM{k} KZ*Kxprod*100) / (SUM{k} KZ*Kxprod))));


---------------------------------
-- 12. R-Tab.7
---------------------------------
WRITE --rtab7
">###  Zuteilung : $$$ $$$$$$$$$$$$$$$$$$$$$

                   IST    IST/SOLL1   SOLL1   SOLL2
                   in t       %       in t    in t
Lagerentnahme   #######    ###.##   ####### #######
-----------------------------------------------------
Zuteilung auf    in Gramm  PI  Zielmenge       %
$$$$$$$$$$          #####  ###    ######    ###.##%

Anmerkung:
  IST   = geplante Lagerentnahme
  SOLL1 = kurzfristige Sollvorgabe gemäss D-TAB.1
  SOLL2 = langfristige mittlere, geplante (EP) Sollvorgabe
          gemäss D-TAB.1
\n" :
```

```
 ROW{p} (700+p,p,pN,VLprod*10000,IF(SEK=0,999.99,VLprod*1000000/SEK),SEK,SEL,
  ROW{k} (kN,Kxprod*1000,PI,ZM,IF(ZM>0.1,Kxprod*1000/ZM*100,0)));


WRITE --Produktbedarf
">###  Bedarf aller Produkte (R-TAB.4)

I------------------------I-----------I
I Int.                   I Bedarf    I
I Bez. Produkt           I           I
I                        I in Tonnen I
I------------------------I-----------I
I $$$ $$$$$$$$$$$$$$$$$$ I  #######  I
I------------------------I-----------I

\n" :

 801,ROW{p}(p,pN,VLprod*10000);


----------------
WRITE{p} TO 'rap,Lebensmittel' :
  'ID' = p,
  'effEntn' = 10000*VLprod[p],
  'Zuteil' = 1000*(SUM{k,bg} ANZAHL*(BEZUEGER+sBEZUEGER)*Kxprod/30)
             / (SUM{bk} BEZUEGER);
WRITE{k,p} TO ',Lebensmittel_Karten' :
  'kID'= k,
  'lID' = p,
  'ZMopt' = 1000*Kxprod[k,p];

WRITE{k} TO ',Karten' :
  'ID' = k,
  'Ener'   = KxGy[k,1]*1000/30,
  'Ei'     = KxGy[k,2]*1000/30,
  'TierEi' = KxGy[k,5]*1000/30,
  'Fett'   = KxGy[k,3]*1000/30,
  'FettS'  = KxGy[k,6]*1000/30,
  'Kh'     = KxGy[k,4]*1000/30,
  'ratEner'   = SUM{kbR} GT[p,1]*Kxprod/30,
  'ratEi'     = SUM{kbR} GT[p,2]*Kxprod/30,
  'ratTierEi' = SUM{kbR} GT[p,5]*Kxprod/30,
  'ratFett'   = SUM{kbR} GT[p,3]*Kxprod/30,
  'ratFettS'  = SUM{kbR} GT[p,6]*Kxprod/30,
  'ratKh'     = SUM{kbR} GT[p,4]*Kxprod/30,
  'K'   = SUM{q}MG[q,1]*Kxprod/30,
  'Mg'  = SUM{q}MG[q,2]*Kxprod/30,
  'Ca'  = SUM{q}MG[q,3]*Kxprod/30,
  'Fe'  = SUM{q}MG[q,4]*Kxprod/30;

END -- Modell Report
```

Anhang B: MAIN.PAS (rap.exe) Source-Code

```
unit main;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, ExtCtrls, StdCtrls, Menus, Buttons, Mask, DBCtrls, Grids,
  DBGrids, Db, ADODB, About,ShellAPI, env;

const
  MAXSCEN=30; {maximale Anzahl Scenarien}
  MAXKAR=10;  {maximale Anzahl Karten}
  MAXPR=200;  {maximale Anzahl Produkte}

type
  TMainForm = class(TForm)
    MainMenu1: TMainMenu;
```

```
Datei1: TMenuItem;
Beenden1: TMenuItem;
Bearbeiten1: TMenuItem;
Lebensmittel1: TMenuItem;
Karten1: TMenuItem;
Konsumenten1: TMenuItem;
N1: TMenuItem;
Einstellungen1: TMenuItem;
Hilfe1: TMenuItem;
Manual1: TMenuItem;
Info1: TMenuItem;
StatusBar1: TStatusBar;
ADOConnection1: TADOConnection; //Verbindung zu RAP.MDB
ADOTable1: TADOTable;     //Lebensmittel
ADOTable2: TADOTable;     //Abteilung
ADOTable4: TADOTable;     //Gruppen
ADOTable5: TADOTable;     //Gruppen
ADOTable6: TADOTable;     //Karten
ADOTable3: TADOTable;     //Quellen
ADOTable7: TADOTable;     //Temporary
ADOTable10: TADOTable;    //Gruppen
ADOTable11: TADOTable;    //Lebensmittel temporary
ADOQuery1: TADOQuery;     //Nachfolger-Query
ADOQuery2: TADOQuery;     //Partner-Query
ADOQuery3: TADOQuery;     //Vorgänger-Query
ADOQuery4: TADOQuery;     //Verpackungs-Query
ADOQuery5: TADOQuery;     //Karten-Query
ADOQuery7: TADOQuery;     //Karten-Produkt-Query
DataSource1: TDataSource;
DataSource2: TDataSource;
DataSource4: TDataSource;
DataSource5: TDataSource;
DataSource6: TDataSource;
DataSource7: TDataSource;
DataSource8: TDataSource;
DataSource9: TDataSource;
DataSource10: TDataSource;
PageControl1: TPageControl;
TabSheet1: TTabSheet;
TabSheet2: TTabSheet;
TabSheet3: TTabSheet;
TabSheet4: TTabSheet;
TabSheet6: TTabSheet;
TabSheet7: TTabSheet;
Lebensmittel: TPanel;
Label5: TLabel;
DBNavigator1: TDBNavigator;
Panel5: TPanel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
DBCheckBox1: TDBCheckBox;
DBCheckBox2: TDBCheckBox;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
DBEdit4: TDBEdit;
DBEdit5: TDBEdit;
DBEdit6: TDBEdit;
Panel8: TPanel;
Panel11: TPanel;
Label27: TLabel;
Panel1: TPanel;
Panel2: TPanel;
DBGrid1: TDBGrid;
ComboBox1: TComboBox;
Label40: TLabel;
Label1: TLabel;
CheckBox1: TCheckBox;
DBEdit15: TDBEdit;
DBEdit17: TDBEdit;
Panel6: TPanel;
Kalcium: TLabel;
```

```
Label15: TLabel;
Label16: TLabel;
Label17: TLabel;
DBEdit7: TDBEdit;
DBEdit8: TDBEdit;
DBEdit9: TDBEdit;
DBEdit10: TDBEdit;
Panel7: TPanel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
Label21: TLabel;
Label26: TLabel;
DBEdit11: TDBEdit;
DBEdit12: TDBEdit;
DBEdit13: TDBEdit;
DBEdit14: TDBEdit;
DBComboBox3: TDBComboBox;
Panel9: TPanel;
DBGrid8: TDBGrid;
DBNavigator2: TDBNavigator;
Label14: TLabel;
Panel10: TPanel;
DBGrid5: TDBGrid;
Label25: TLabel;
DBEdit25: TDBEdit;
DBNavigator3: TDBNavigator;
Panel16: TPanel;
Label43: TLabel;
DBGrid6: TDBGrid;
DBNavigator4: TDBNavigator;
DataSource11: TDataSource;
Panel3: TPanel;
Label41: TLabel;
Label42: TLabel;
Memo1: TMemo;
Label44: TLabel;
Optimieren: TButton;
Label45: TLabel;
Panel17: TPanel;
Panel18: TPanel;
IntSchema: TListBox;
ExtSchema: TListBox;
Panel19: TPanel;
Ausdrucken: TButton;
Panel20: TPanel;
Label46: TLabel;
Panel21: TPanel;
Label47: TLabel;
StartEditor: TButton;
RichEdit1: TRichEdit;
ADOTable8: TADOTable;
ADOTable9: TADOTable;
Panel22: TPanel;
Label2: TLabel;
Panel12: TPanel;
Label48: TLabel;
DBGrid3: TDBGrid;
StringGrid1: TStringGrid;
Label3: TLabel;
Panel13: TPanel;
Label49: TLabel;
DBGrid4: TDBGrid;
StringGrid2: TStringGrid;
Panel23: TPanel;
Panel24: TPanel;
ScenLoad: TButton;
ScenDelete: TButton;
Panel26: TPanel;
scNa: TLabel;
Label50: TLabel;
LPLModell: TButton;
ADOConnection2: TADOConnection;  //Verbindung zu SCEN.MDB
ADOTemp: TADOTable;              //Temorary Table
Panel28: TPanel;
Label51: TLabel;
```

```
Label55: TLabel;
Label56: TLabel;
Label57: TLabel;
Label58: TLabel;
Panel29: TPanel;
Label59: TLabel;
Label60: TLabel;
Label61: TLabel;
Label62: TLabel;
Label63: TLabel;
DBNavigator6: TDBNavigator;
DBNavigator7: TDBNavigator;
Panel30: TPanel;
DBGrid2: TDBGrid;
Label4: TLabel;
Label6: TLabel;
UpDown1: TUpDown;
UpDown2: TUpDown;
Panel4: TPanel;
Label7: TLabel;
DBNavigator5: TDBNavigator;
ADOTable2ID: TIntegerField;
ADOTable2Abteilung: TStringField;
ADOTable2Beschreibung: TStringField;
ComboBox2: TComboBox;
PopupMenu1: TPopupMenu;
Szenarioladen1: TMenuItem;
Szenariolschen1: TMenuItem;
Button5: TButton;
Button7: TButton;
Label24: TLabel;
ListBox3: TListBox;
Panel15: TPanel;
Button9: TButton;
Label22: TLabel;
Edit1: TEdit;
ScenGenerate: TButton;
ScenExport: TButton;
ScenImport: TButton;
Refresh1: TMenuItem;
Panel14: TPanel;
Label23: TLabel;
DBGrid7: TDBGrid;
Panel25: TPanel;
Label28: TLabel;
Label29: TLabel;
Label30: TLabel;
Label31: TLabel;
Label32: TLabel;
DBNavigator8: TDBNavigator;
Label33: TLabel;
Label34: TLabel;
Label35: TLabel;
procedure FormShow(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Beenden1Click(Sender: TObject);
procedure Info1Click(Sender: TObject);
procedure Refresh1Click(Sender: TObject);
procedure PageControl1Changing(Sender: TObject; var AllowChange: Boolean);

procedure ComboBox1Change(Sender: TObject);              //Produkte
procedure DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
procedure CheckBox1Click(Sender: TObject);
procedure ADOTable1AfterEdit(DataSet: TDataSet);
procedure ADOTable1AfterInsert(DataSet: TDataSet);
procedure ADOTable1BeforePost(DataSet: TDataSet);
procedure ADOTable1AfterPost(DataSet: TDataSet);
procedure ADOTable1BeforeDelete(DataSet: TDataSet);
procedure ADOTable1AfterDelete(DataSet: TDataSet);
procedure ADOQuery1AfterInsert(DataSet: TDataSet);
procedure ADOQuery1BeforePost(DataSet: TDataSet);
procedure ADOQuery1AfterDelete(DataSet: TDataSet);
procedure ADOQuery2AfterInsert(DataSet: TDataSet);
procedure ADOQuery2BeforePost(DataSet: TDataSet);
procedure ADOQuery3AfterInsert(DataSet: TDataSet);
procedure ADOQuery3BeforePost(DataSet: TDataSet);
```

```
    procedure ADOQuery3AfterDelete(DataSet: TDataSet);
    procedure ADOQuery4AfterInsert(DataSet: TDataSet);

(*    procedure ADOTable6AfterInsert(DataSet: TDataSet);       //Karten
    procedure ADOTable6AfterPost(DataSet: TDataSet);
    procedure ADOTable6BeforeDelete(DataSet: TDataSet);
    procedure ADOTable6AfterDelete(DataSet: TDataSet); *)
    procedure ADOQuery5AfterDelete(DataSet: TDataSet);       //Karten
    procedure ADOQuery5AfterInsert(DataSet: TDataSet);
    procedure ADOQuery5AfterPost(DataSet: TDataSet);
    procedure ADOQuery5BeforeDelete(DataSet: TDataSet);
    procedure ADOQuery7BeforePost(DataSet: TDataSet);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure DBGrid1TitleClick(Column: TColumn);
    procedure DBGrid1KeyPress(Sender: TObject; var Key: Char);
    procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);
    procedure UpDown2Click(Sender: TObject; Button: TUDBtnType);
    procedure ADOQuery7BeforeEdit(DataSet: TDataSet);
    procedure DBGrid1Exit(Sender: TObject);
    procedure DBNavigator5Click(Sender: TObject; Button: TNavigateBtn);
    procedure TabSheet2Show(Sender: TObject);

    procedure ADOTable4AfterInsert(DataSet: TDataSet);        //Konsumenten
    procedure ADOTable4AfterPost(DataSet: TDataSet);
    procedure ADOTable4BeforeDelete(DataSet: TDataSet);
    procedure ADOTable4AfterDelete(DataSet: TDataSet);
    procedure ADOTable5AfterInsert(DataSet: TDataSet);
    procedure ADOTable5AfterPost(DataSet: TDataSet);
    procedure ADOTable5BeforeDelete(DataSet: TDataSet);
    procedure ADOTable5AfterDelete(DataSet: TDataSet);
    procedure DBNavigator6Click(Sender: TObject; Button: TNavigateBtn);
    procedure DBNavigator7Click(Sender: TObject; Button: TNavigateBtn);

    procedure SaveScen(Sender: TObject);           // Szenarios
    procedure DeleteScen(Sender: TObject);
    procedure LoadScen(Sender: TObject);
    procedure NewScen(Sender: TObject);
    procedure ScenGenerateClick(Sender: TObject);
    procedure NextPeriod(Sender: TObject);

    procedure OptimierenClick(Sender: TObject);      //optimieren
    procedure LPLModellClick(Sender: TObject);

    procedure IntSchemaClick(Sender: TObject);     // Resultate
    procedure StartEditorClick(Sender: TObject);
    procedure AusdruckenClick(Sender: TObject);
    procedure TabSheet7Show(Sender: TObject);
    procedure ExtSchemaClick(Sender: TObject);
    procedure ScenExportClick(Sender: TObject);
    procedure ScenImportClick(Sender: TObject);
  private
  public
    procedure UpdateStatusBar(s:string);
    procedure OpenRapDB;
    procedure ReOpenRapDB;
    procedure CloseRapDB;
    procedure InitProducts(stay:boolean);
    procedure ExecProdQueries;
    procedure UpdateET_FS;       // in Tabelle Lebensmittel
    procedure UpdateQuellen;     // in Tabelle Lebensmittel
    procedure UpdateRezept;      // in Tabelle Lebensmittel
    procedure UpdateVerpackung;  // in Tabelle Lebensmittel
    procedure ExecKarteQuery;
    procedure InitKartenGruppen;
    procedure GruppenTotale;

    procedure GetScenarioName;
    procedure PutScenarioName;
    procedure InitScenario;
    procedure SaveScenario(ask:boolean);

    procedure ReadNOMHeaders;
  end;

var
  MainForm: TMainForm;
```

```
  pID:array[1..MAXPR] of integer; nPr:integer;  // Alle Produkt-Codes
  kID,kX,kY:array[1..MAXKAR] of integer;        // KartenID, total Karten*Benützer
  kA,kN:array[1..MAXKAR] of string; nK:integer; // KartenCode, -Name
  FirstCode:string;                  // erste Konsumentengruppe (Code: NV)
  scen,per:integer;                             //actual scenario and period
  AllScen,AllPer: array[1..MAXSCEN] of integer; nSc:integer; // all scen.
  AllTables,AllFields: TStrings;   // all tables from _Tabellen in Scen.mdb
  PInserted,Pedit:boolean;         // a new product is about to be inserted
  G1Inserted,G2Inserted:boolean;   // a new group is about to be inserted
  KInserted:boolean;               // a new Karte is about to be inserted
  Q7Edit:boolean;                  // Query7 about to change
  Q1Inserted,Q2Inserted,Q3Inserted:boolean;   // Query1, Query2, Query3 insert
  K1,K2:integer;                   // the two Karten shown in PI,ZM-Grid
  MySCENDB,MyDB:string;            // Datenbanknamen
  Q7sort:string;                   // sort on Query7

implementation
uses comobj, LPLrap, unit1, Unit2, Unit3, Unit4, Unit5, Unit6, Unit7,
     Unit8, Unit9, Unit10;
{$R *.DFM}

function GetListBoxIndex(Sender:TObject):integer; //low-level function
{return index [1...count] of the selected ListBox-Item}
var k,i:integer;
begin
  k:=0; for i:=0 to TListBox(Sender).Items.count-1 do
    if TListBox(Sender).Selected[i] then begin k:=i; break; end;
  Result:=k+1;
end;

procedure TMainForm.UpdateStatusBar(s:string);
begin StatusBar1.Panels[0].Text:=s; Update; end;

procedure TMainForm.OpenRapDB; //initializing and open the DBs
{ --- nothing to do: already open at beginning}
var i:integer;
begin with ADOConnection1 do begin //open rap.mdb
  //if Connected then Close;
  //ConnectionString:=ConnectionString+';Data Source='+MyDB;
  //Open;
  for i:=0 to DataSetCount-1 do begin
    //DataSets[i].active:=true;
    AboutBox.UpProgress('Datenbank Tabellen und Queries öffnen '+IntToStr(i),2);
  end; end;
  with ADOConnection2 do begin //open scen.mdb
    //if Connected then Close;
    //ConnectionString:=ConnectionString+';Data Source='+MySCENDB;
    //Open;
end; end;

procedure TMainForm.CloseRapDB;
begin
  if ADOConnection1.Connected then begin
    ADOTable1.Close; ADOTable2.Close; ADOTable3.Close; ADOTable4.Close;
    ADOTable5.Close; ADOTable6.Close; ADOTable8.Close; ADOTable9.Close;
    ADOTable10.Close;ADOQuery1.Close; ADOQuery2.Close; ADOQuery3.Close;
    ADOQuery4.Close; ADOQuery5.Close; ADOQuery7.Close;
    ADOConnection1.Close;
  end;
end;

procedure TMainForm.ReOpenRapDB;
begin
  CloseRapDB;
  ADOConnection1.Open;
  ADOTable1.Open; ADOTable2.Open; ADOTable3.Open; ADOTable4.Open;
  ADOTable5.Open; ADOTable6.Open; ADOTable8.Open; ADOTable9.Open;
  ADOTable10.Open; ADOQuery1.Open; ADOQuery2.Open; ADOQuery3.Open;
  ADOQuery4.Open; ADOQuery5.Open; ADOQuery7.Open;
end;

procedure TMainForm.InitProducts(stay:boolean);
{generate product-mask, stay:jump to actual product}
var s,s1,s2:string;
begin
  ADOTable1.Refresh;
```

```
  nPr:=0;
  ComboBox1.Items.Clear; ComboBox2.Items.Clear;
  ADOTable1.First;
  s2:=DBEdit1.text; if s2='' then exit;
  while not ADOTable1.eof do begin
    inc(nPr); pID[nPr]:=ADOTable1.FieldByName('IDsort').AsInteger;
    s1:=ADOTable1.FieldByName('ID').AsString+' ';
    if length(s1)=2 then s1:='0'+s1; if length(s1)=3 then s1:='0'+s1;
    s:=ADOTable1.FieldByName('Bezeichnung').AsString;
    ComboBox2.Items.Add(s1+s);
    if CheckBox1.Checked then s:=s1+s;
    ComboBox1.Items.Add(s);
    ADOTable1.Next;
  end;
  ADOTable1.First;
  if stay then ADOTable1.Locate('ID',s2,[]);
  ComboBox1.Text:=DBEdit2.text;
  ExecProdQueries;
end;

procedure TMainForm.ExecProdQueries;
{run product-mask queries (Nachbar etc.)}
var para:integer;
begin
  para:=StrToInt(DBEdit1.text);
  ADOQuery1.Close; ADOQuery1.Parameters[0].Value:=para; ADOQuery1.Open;
  ADOQuery2.Close; ADOQuery2.Parameters[0].Value:=para; ADOQuery2.Open;
  ADOQuery3.Close; ADOQuery3.Parameters[0].Value:=para; ADOQuery3.Open;
  ADOQuery4.Close; ADOQuery4.Parameters[0].Value:=para; ADOQuery4.Open;
end;

procedure TMainForm.UpdateET_FS; {update ET & FS in Table Lebensmittel}
var SQL:string;
begin
  UpdateStatusBar('ET & FS modifizieren');
  //ADOConnection1.BeginTrans;
  ADOConnection1.Execute('UPDATE Lebensmittel SET ET=0, FS=0;');
  //ADOConnection1.CommitTrans;
  //ADOConnection1.BeginTrans;
  ADOConnection1.Execute('UPDATE Lebensmittel SET ET=Ei WHERE ET_FS LIKE "%ET%";');
  //ADOConnection1.CommitTrans;
  //ADOConnection1.BeginTrans;
  ADOConnection1.Execute('UPDATE Lebensmittel SET FS=Fett WHERE ET_FS Like "%FS%";');
  //ADOConnection1.CommitTrans;
end;

procedure TMainForm.UpdateQuellen;
var  s:string; id:integer;
begin
  UpdateStatusBar('Quellen modifizieren');
  ADOTable7.TableName:='Quellen_Lebensmittel'; ADOTable7.IndexFieldNames:='LID';
  ADOTable7.Open; ADOTable7.First; ADOTable11.Open;
  s:='';
  while not ADOTable7.eof do begin
    id:=ADOTable7.FieldByName('LID').AsInteger;
    s:=s+ADOTable7.FieldByName('QID').AsString+' / ';
    ADOTable7.next;
    if ADOTable7.eof or (id<>ADOTable7.FieldByName('LID').AsInteger) then begin
      delete(s,length(s)-1,2);
      ADOTable11.First;
      ADOTable11.Locate('ID',id,[]);
      ADOTable11.Edit;
      ADOTable11.FieldByName('Quellen').AsString:=s;
      ADOTable11.Post;
      s:='';
    end;
  end;
  ADOTable7.Close; ADOTable11.Close;
end;

procedure TMainForm.UpdateRezept;
var s:string; id:integer;
begin
  UpdateStatusBar('rezept modifizieren');
  ADOTable7.TableName:='Verwendung'; ADOTable7.IndexFieldNames:='Ausgang';
  ADOTable7.Open; ADOTable7.First; ADOTable11.Open;
```

```
  s:='';
  while not ADOTable7.eof do begin
    id:=ADOTable7.FieldByName('Ausgang').AsInteger;
    s:=s+ADOTable7.FieldByName('Endprodukt').AsString+' '
        +ADOTable7.FieldByName('Anteil').AsString+'% / ';
    ADOTable7.next;
    if ADOTable7.eof or (id<>ADOTable7.FieldByName('Ausgang').AsInteger) then begin
      delete(s,length(s)-1,2);
      ADOTable11.First;
      ADOTable11.Locate('ID',id,[]);
      ADOTable11.Edit;
      ADOTable11.FieldByName('rezept').AsString:=s;
      ADOTable11.Post;
      s:='';
    end;
  end;
  ADOTable7.Close; ADOTable11.Close;
end;

procedure TMainForm.UpdateVerpackung;
var s,s1:string; id:integer;
begin
  UpdateStatusBar('Verpackung modifizieren');
  ADOTable7.TableName:='Verpackung'; ADOTable7.IndexFieldNames:='LID;Menge';
  ADOTable7.Open; ADOTable7.First;  ADOTable11.Open;
  s:=''; s1:='';
  while not ADOTable7.eof do begin
    id:=ADOTable7.Fields[0].AsInteger;
    s1:=s1+ADOTable7.FieldByName('MENGE').AsString+' ';
    s:=s+ADOTable7.FieldByName('MENGE').AsString+' '
        +ADOTable7.FieldByName('Kommentar').AsString+' / ';
    ADOTable7.next;
    if ADOTable7.eof or (id<>ADOTable7.FieldByName('LID').AsInteger) then begin
      delete(s,length(s)-1,2);
      ADOTable11.First;
      ADOTable11.Locate('ID',id,[]);
      ADOTable11.Edit;
      ADOTable11.FieldByName('Verpackung').AsString:=s;
      ADOTable11.FieldByName('Verpackung1').AsString:=s1;
      ADOTable11.Post;
      s:=''; s1:='';
    end;
  end;
  ADOTable7.Close; ADOTable11.Close;
  UpdateStatusBar('Verpackung ok!');
end;

procedure TMainForm.ExecKarteQuery;
{build and run the Karten-query}
var sK1,sK2,SQL:string;
begin
  ADOQuery7.Close;
  Label6.Caption:=kN[K1]; Label7.Caption:=kN[K2];
  sK1:='a'+IntToStr(K1); sK2:='a'+IntToStr(K2);
  DBGrid1.Columns[2].FieldName:=sK1+'.PI';
  DBGrid1.Columns[3].FieldName:=sK1+'.ZM';
  DBGrid1.Columns[5].FieldName:=sK2+'.PI';
  DBGrid1.Columns[6].FieldName:=sK2+'.ZM';
  if K1=K2 then begin                        // that's the way it is !!!
    DBGrid1.Columns[2].FieldName:='PI';
    DBGrid1.Columns[3].FieldName:='ZM';
    DBGrid1.Columns[5].FieldName:='PI';
    DBGrid1.Columns[6].FieldName:='ZM';
  end;
  DBGrid1.Columns[7].FieldName:={sK2+'.}'PIdef'; // the same!!!
  DBGrid1.Columns[8].FieldName:={sK2+'.}'ZMdef';
  DBGrid1.Columns[9].FieldName:={sK2+'.}'ZMopt';
  ADOQuery7.SQL.Clear;
  ADOQuery7.SQL.Add('SELECT L.ID+0,L.Bezeichnung+"",'+sK1+'.PI,'+sK1+'.ZM,'+sK2+'.PI,'
    +sK2+'.ZM,'+sK2+'.PIdef,'+sK2+'.ZMdef,'+sK2+'.ZMopt, '
    +'IIF('+sK2+'.ZM<1,0,INT(100*'+sK2+'.ZMopt/'+sK2+'.ZM)),'
    //+'0,'
    +'L.Zuteil+0,INT(L.Zuteil*L.Kal/1000), L.Verpackung1+"" FROM ');
  if sK1=sK2 then ADOQuery7.SQL.Add(sK1+',Lebensmittel AS L WHERE L.ID='+sK1
                  +'.c ORDER BY '+Q7sort+';')
  else ADOQuery7.SQL.ADD(sK1+','+sK2+',Lebensmittel AS L WHERE L.ID='+sK1
```

```
                  +'.c AND L.ID='+sK2+'.c ORDER BY '+Q7sort+';');
  //ADOQuery7.SQL.SaveToFile('xxx.txt');  //for debugging
  ADOQuery7.Open;
end;

procedure TMainForm.InitKartenGruppen;
var gID,kID1,An,i,k:integer;
begin
  UpdateStatusBar('Karten reinitialisieren');
  nK:=0;
  ADOTable6.Close; ADOTable6.Open; ADOTable6.First;
  while not ADOTable6.eof do begin
    inc(nK);
    kID[nK]:=ADOTable6.FieldByName('ID').AsInteger;
    kA[nK]:=ADOTable6.FieldByName('Code').AsString;
    kN[nK]:=ADOTable6.FieldByName('Bezeichnung').AsString;
    ADOTable6.next;
  end;
  ADOTable6.First;
  for i:=1 to nK do begin
    DBGrid3.Columns[3+i].Title.Caption:=kA[i];
    DBGrid3.Columns[3+i].FieldName:='K'+IntToStr(i);
    DBGrid4.Columns[3+i].Title.Caption:=kA[i];
    DBGrid4.Columns[3+i].FieldName:='K'+IntToStr(i);
  end;
  for i:=nK+1 to MAXKAR-1 do begin
    DBGrid3.Columns[3+i].Title.Caption:=''; DBGrid3.Columns[3+i].FieldName:='';
    DBGrid4.Columns[3+i].Title.Caption:=''; DBGrid4.Columns[3+i].FieldName:='';
  end;
  ADOTable8.Close; ADOTable8.Open; ADOTable8.First;
  while not ADOTable8.eof do begin
    gID:=ADOTable8.FieldByName('gID').AsInteger;
    kID1:=ADOTable8.FieldByName('kID').AsInteger;
    for i:=1 to MAXKAR do if kID1=kID[i] then begin k:=i; break; end;
    An:=ADOTable8.FieldByName('Anzahl').AsInteger;
    ADOTable9.First;
    ADOTable9.Locate('ID',gID,[]);
    ADOTable9.Edit; ADOTable9.FieldByName('K'+IntToStr(k)).AsInteger:=An;
    ADOTable9.Post;
    ADOTable8.next;
  end;
  GruppenTotale;
  ADOTable4.Refresh; ADOTable5.Refresh;
  K1:=1; K2:=2;
  //ExecKarteQuery;
  UpdateStatusBar('Karten initialisiert.');
end;

procedure TMainForm.GruppenTotale;
var i,y:integer;
begin
  for i:=1 to MAXKAR do begin kX[i]:=0; kY[i]:=0; end;
  ADOTable9.Close; ADOTable9.Open; ADOTable9.First;
  FirstCode:=ADOTable9.FieldByName('Code').AsString;
  while not ADOTable9.eof do begin
    y:=ADOTable9.FieldByName('Anzahl').AsInteger;
    if ADOTable9.FieldByName('Basis').AsBoolean then begin
      kX[1]:=kX[1]+y;
      for i:=1 to nK do
        kX[i+1]:=kX[i+1]+ADOTable9.FieldByName('K'+IntToStr(i)).AsInteger*y;
    end else begin kY[1]:=kY[1]+y;
      for i:=1 to nK do
        kY[i+1]:=kY[i+1]+ADOTable9.FieldByName('K'+IntToStr(i)).AsInteger*y;
    end;
    ADOTable9.next;
  end;
  for i:=0 to nK do begin
    kX[i+1]:=kX[i+1] div 1000; StringGrid1.Cells[i,0]:=IntToStr(kX[i+1]);
    kY[i+1]:=kY[i+1] div 1000; StringGrid2.Cells[i,0]:=IntToStr(kY[i+1]);
  end;
  for i:=nK+1 to MAXKAR-1 do begin StringGrid1.Cells[i,0]:='';
    StringGrid2.Cells[i,0]:=''; end;
end;

{---------- on inititialization -------}
procedure TMainForm.FormShow(Sender: TObject);
```

```
var i,j:integer; TheFields:string;
begin
  Q7sort:='L.IDsort';
  MyDB:=DirExe+'rap.mdb'; MySCENDB:=DirExe+'scen.mdb';
  OpenRapDB;
  AboutBox.UpProgress('Szenarios finden',5);
  PInserted:=false; Pedit:=false;
  G1Inserted:=false; G2Inserted:=false; KInserted:=false;
  Q7Edit:=false; Q1Inserted:=false; Q2Inserted:=false; Q3Inserted:=false;
  AllTables:=TStringList.Create;
  ADOTemp.TableName:='_Tabellen'; ADOTemp.Open;
  while not ADOTemp.eof do begin
    AllTables.Add(ADOTemp.FieldByName('Name').AsString);
    ADOTemp.next;
  end;
  ADOTemp.Close;
  AllFields:=TStringList.Create;
  for i:=0 to AllTables.count-1 do begin
    ADOTemp.TableName:=AllTables[i]; ADOTemp.Open;
    TheFields:=ADOTemp.Fields[2].FieldName;
    for j:=3 to ADOTemp.Fields.count-1 do
      TheFields:=TheFields+', '+ADOTemp.Fields[j].FieldName;
    AllFields.Add(TheFields);
    ADOTemp.Close;
  end;
  GetScenarioName;
  AboutBox.UpProgress('Produktdaten initialisieren',10);
  InitProducts(false);
  AboutBox.UpProgress('Kartendaten initialisieren',5);
  InitKartenGruppen;
  InitScenario;
  AboutBox.UpProgress('',100);
end;

procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  AllTables.Free;
  AllFields.Free;
end;

procedure TMainForm.Beenden1Click(Sender: TObject);
begin
  Close;
end;

procedure TMainForm.Info1Click(Sender: TObject);
begin
  AboutBox.Show;
end;

procedure TMainForm.Refresh1Click(Sender: TObject);
begin
  UpdateET_FS;
  UpdateQuellen;
  UpdateRezept;
  UpdateVerpackung;
  ReOpenRapDB;
  InitProducts(true);
  InitKartenGruppen;
  UpdateStatusBar('Refresh durchgeführt.');
end;

procedure TMainForm.PageControl1Changing(Sender: TObject; var AllowChange: Boolean);
begin
  if ((Sender as TPageControl).ActivePage=TabSheet1)
  then AllowChange:=DBEdit1.ReadOnly else AllowChange := True;
  if not AllowChange then exit;
  if PInserted or Pedit then ADOTable1.Post;
  if KInserted then ADOQuery5.Post;
  if G1Inserted then ADOTable4.Post; if G2Inserted then ADOTable5.Post;
  //if Q1Inserted then ADOQuery1.Post;
  //if Q2Inserted then ADOQuery2.Post;
  //if Q3Inserted then ADOQuery3.Post;
end;

// -------------------------------- PRODUKTE --------------------------------
```

```
procedure TMainForm.ComboBox1Change(Sender: TObject);
begin
  if (ComboBox1.Text<>'') and (ComboBox1.Text[1] in ['0'..'9']) then
    ComboBox1.Text:=copy(ComboBox1.Text,5,length(ComboBox1.Text));
  if not ADOTable1.Locate('Bezeichnung',ComboBox1.Text,[]) then InitProducts(false)
  else ExecProdQueries;
  DBEdit3.SetFocus;
end;

procedure TMainForm.DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
begin
  ComboBox1.text:=DBEdit2.text;
  if Button in [nbFirst,nbPrior,nbNext,nbLast] then ExecProdQueries;
  if Button=nbDelete then PInserted:=false;
end;

procedure TMainForm.CheckBox1Click(Sender: TObject);
begin
  InitProducts(true);
end;

procedure TMainForm.ADOTable1AfterEdit(DataSet: TDataSet);
begin
  Pedit:=true;
end;

procedure TMainForm.ADOTable1AfterInsert(DataSet: TDataSet);
begin
  DBEdit1.ReadOnly:=false; DBEdit2.text:='Produkt Name';
  ComboBox1.text:=DBEdit2.text;
  ShowMessage('Aendern Sie den Produktcode und die Produktbezeichnung.'#10#13
   + '  000-099: Alkoholika'#10#13
   + '  100-300: ...'#13#10
   + '  301-999: ....');
  DBEdit1.SetFocus;
  PInserted:=true;
end;

  function GetProduct(ID:integer):boolean; var i:integer;
  begin
    for i:=1 to nPr do if pID[i]=ID then begin GetProduct:=true; exit; end;
    GetProduct:=false;
  end;
procedure TMainForm.ADOTable1BeforePost(DataSet: TDataSet);
var LID:integer;
begin
  if ADOTable1.FieldByName('ID').AsString='' then
      ADOTable1.FieldByName('ID').AsString:='0';
  LID:=ADOTable1.FieldByName('ID').AsInteger; if LID<100 then LID:=LID+1000;
  if PInserted then begin
    ADOTable1.FieldByName('IDsort').AsInteger:=LID;
    if GetProduct(LID) or (StrToInt(DBEdit1.Text)<1) or (StrToInt(DBEdit1.text)>999)
    then begin
      ShowMessage('Dieser Code ist nicht erlaubt. Bitte einen anderen wählen.');
      DBEdit1.SetFocus;
      Abort;
     end;
  end;
end;

procedure TMainForm.ADOTable1AfterPost(DataSet: TDataSet);
var i,LID:integer; SQL:string;
begin
  LID:=ADOTable1.FieldByName('ID').AsInteger;
  if PInserted then begin
    for i:=1 to nK do begin
      SQL:='INSERT INTO Lebensmittel_Karten (KID,LID) VALUES ( '+
            IntToStr(kID[i])+' , '+IntToStr(LID)+')';
      ADOConnection1.Execute(SQL);
    end;
    DBEdit1.ReadOnly:=true;
    PInserted:=false;
  end;
  Pedit:=false;
  ADOTable1.Refresh;
```

```
  InitProducts(true);
end;

procedure TMainForm.ADOTable1BeforeDelete(DataSet: TDataSet);
begin
  {LID:=ADOTable1.FieldByName('ID').AsInteger;}
  if MessageDlg('Wollen Sie das Produkt wirklich löschen?',mtConfirmation,
               [mbYes,mbNo],0)=mrNo then Abort;
  UpdateStatusBar(IntToStr(ADOTable1.FieldByName('ID').AsInteger));
    // Comments to the programmer:
    // Cascaded DELETEs in Lebensmittel_Karten,Quellen_Lebensmittel,Verhaeltnis,
    // Verwendung,Verpackung are automatically done: for example:
    //  SQL:='DELETE FROM Lebensmittel_Karten WHERE LID='+IntToStr(LID);
    //  ADOConnection1.Execute(SQL);
end;

procedure TMainForm.ADOTable1AfterDelete(DataSet: TDataSet);
begin
  //ADOConnection1.CommitTrans;
  ADOTable1.Refresh;
  InitProducts(false);
end;

// ---- Produkte : Nachfolger
procedure TMainForm.ADOQuery1AfterInsert(DataSet: TDataSet);
begin
  Q1Inserted:=true;
  DBGrid5.Fields[2].Text:='100';
  ComboBox2.Text:=IntToStr(Random(998)+1);
  ComboBox2.Visible:=true; ComboBox2.DroppedDown:=true;
end;

procedure TMainForm.ADOQuery1BeforePost(DataSet: TDataSet);
var id,idP,Ant,SQL:string;
begin
  if PInserted then ADOTable1.Post;
  if Q1Inserted and (ComboBox2.Text<>'') then begin
    id:=DBEdit1.text; idP:=copy(ComboBox2.Text,1,3);
    Ant:=DBGrid5.Fields[2].Text;
    SQL:='INSERT INTO Verwendung (Ausgang,Endprodukt,Anteil) VALUES ( '
          +id+','+idP+','+Ant+')';
    ADOConnection1.Execute(SQL);
    ADOQuery1.CancelUpdates; ADOQuery1.Cancel;
    ADOQuery1.Close; ADOQuery1.Open;
    {ADOTable1.Edit;} DBEdit25.Text:='A';
    {ADOTable1.Post;}
    ComboBox2.Visible:=false; Q1Inserted:=false;
    Abort;
  end;
  Q1Inserted:=false;
end;

procedure TMainForm.ADOQuery1AfterDelete(DataSet: TDataSet);
begin
  if ADOQuery1.FieldByName('Ausgang').AsString='' then begin
    {ADOTable1.Edit;} DBEdit25.text:='E'; {ADOTable1.Post;} ADOTable1.Refresh;
  end;
end;

// ---- Produkte : Partner
procedure TMainForm.ADOQuery2AfterInsert(DataSet: TDataSet);
begin
  Q2Inserted:=true;
  DBGrid6.Fields[2].Text:='100';
  ComboBox2.Text:=IntToStr(Random(998)+1);
  ComboBox2.Visible:=true; ComboBox2.DroppedDown:=true;
end;

procedure TMainForm.ADOQuery2BeforePost(DataSet: TDataSet);
var id,idP,Ant,SQL:string;
begin
  if PInserted then ADOTable1.Post;
  if Q2Inserted and (ComboBox2.Text<>'')  then begin
    id:=DBEdit1.text; idP:=copy(ComboBox2.Text,1,3);
    Ant:=DBGrid6.Fields[2].Text;
    SQL:='INSERT INTO Verhaeltnis (Ausgang,Partner,Verhaeltnis) VALUES ( '
```

```
          +id+','+idP+','+Ant+')';
    ADOConnection1.Execute(SQL);
    ADOQuery2.CancelUpdates; ADOQuery2.Cancel;
    ADOQuery2.Close; ADOQuery2.Open;
    ComboBox2.Visible:=false; Q2Inserted:=false;
    Abort;
  end;
  Q2Inserted:=false;
end;

// ---- Produkte : Vorgänger
procedure TMainForm.ADOQuery3AfterInsert(DataSet: TDataSet);
begin
  Q3Inserted:=true;
  DBGrid7.Fields[2].Text:='100';
  ComboBox2.Text:=IntToStr(Random(998)+1);
  ComboBox2.Visible:=true; ComboBox2.DroppedDown:=true;
end;

procedure TMainForm.ADOQuery3BeforePost(DataSet: TDataSet);
var id,idP,Ant,SQL:string;
begin
  if PInserted then ADOTable1.Post;
  if Q3Inserted and (ComboBox2.Text<>'') then begin
    id:=DBEdit1.text; idP:=copy(ComboBox2.Text,1,3);
    Ant:=DBGrid7.Fields[2].Text;
    SQL:='INSERT INTO Verwendung (Ausgang,Endprodukt,Anteil) VALUES ( '
         +idP+','+id+','+Ant+')';
    ADOConnection1.Execute(SQL);
    ADOQuery3.CancelUpdates; ADOQuery3.Cancel;
    ADOQuery3.Close; ADOQuery3.Open;
    {ADOTable1.Edit;} DBEdit25.Text:='G';
    {ADOTable1.Post;}
    ADOConnection1.Execute('UPDATE Lebensmittel SET Verw=''M'' WHERE ID='+idP);
    ComboBox2.Visible:=false; Q3Inserted:=false;
    Abort;
  end;
  Q3Inserted:=false;
end;

procedure TMainForm.ADOQuery3AfterDelete(DataSet: TDataSet);
begin
  if ADOQuery3.FieldByName('Ausgang').AsString='' then begin
    {ADOTable1.Edit;} DBEdit25.text:='E'; {ADOTable1.Post;} ADOTable1.Refresh;
  end;
end;

// ---- Produkte : Verpackung
procedure TMainForm.ADOQuery4AfterInsert(DataSet: TDataSet);
begin
  if PInserted then ADOTable1.Post;
  ADOQuery4.Edit;
  ADOQuery4.FieldByName('LID').AsString:=DBEdit1.text;
end;

// --------------------------------- KARTEN -------------------------------

  function RandString(len:integer):string; var i:integer; s:string;
  begin
    SetLength(s,len); for i:=1 to len do s[i]:=chr(Random(26)+65);
    RandString:=s;
  end;
(*
procedure TMainForm.ADOTable6AfterInsert(DataSet: TDataSet);
begin
  if nK>=MAXKAR-1 then begin
    ShowMessage('Nicht mehr als 9 Karten sind erlaubt.'); Abort;
  end;
  KInserted:=true;
  ADOTable6.FieldByName('ID').AsInteger:=30+nK;
  ADOTable6.FieldByName('Code').AsString:=RandString(2);
end;

procedure TMainForm.ADOTable6AfterPost(DataSet: TDataSet);
var ID,xID:integer; SQL:string;
begin
```

```
  if KInserted then begin
    ID:=ADOTable6.FieldByName('ID').AsInteger;
    ADOTable9.First;
    while not ADOTable9.eof do begin
      xID:=ADOTable9.FieldByName('ID').AsInteger;
      SQL:='INSERT INTO Gruppen_Karten (kID,gID) VALUES ('+ IntToStr(ID)+', '
           +IntToStr(xID)+')';
      ADOConnection1.Execute(SQL);
      ADOTable9.next;
    end;
    ADOTable1.First;
    while not ADOTable1.eof do begin
      xID:=ADOTable1.FieldByName('ID').AsInteger;
      SQL:='INSERT INTO Lebensmittel_Karten (kID,LID) VALUES ('+ IntToStr(ID)
           +', '+IntToStr(xID)+')';
      ADOConnection1.Execute(SQL);
      ADOTable1.next;
    end;
  end;
  InitKartenGruppen;
  KInserted:=false;
end;

procedure TMainForm.ADOTable6BeforeDelete(DataSet: TDataSet);
begin
  if ADOTable6.FieldByName('ID').AsInteger<>kID[nK] then begin
    ShowMessage('Nur die letzte Karte kann gelöscht werden!');
    Abort;
  end;
  if MessageDlg('Wollen Sie diese Karte wirklich löschen?',
                mtConfirmation,[mbYes,mbNo],0)=mrNo then
    Abort;
end;

procedure TMainForm.ADOTable6AfterDelete(DataSet: TDataSet);
begin
  InitKartenGruppen;
end;
*)
procedure TMainForm.ADOQuery5AfterInsert(DataSet: TDataSet);
begin
  if nK>=MAXKAR-1 then begin
    ShowMessage('Nicht mehr als 9 Karten sind erlaubt.'); Abort;
  end;
  KInserted:=true;
  ADOQuery5.FieldByName('ID').AsInteger:=30+nK;
  ADOQuery5.FieldByName('Code').AsString:=RandString(2);
end;

procedure TMainForm.ADOQuery5AfterPost(DataSet: TDataSet);
var ID,xID:integer; SQL:string;
begin
  if KInserted then begin
    ID:=ADOQuery5.FieldByName('ID').AsInteger;
    ADOTable9.First;
    while not ADOTable9.eof do begin
      xID:=ADOTable9.FieldByName('ID').AsInteger;
      SQL:='INSERT INTO Gruppen_Karten (kID,gID) VALUES ('+ IntToStr(ID)+', '
           +IntToStr(xID)+')';
      ADOConnection1.Execute(SQL);
      ADOTable9.next;
    end;
    ADOTable1.First;
    while not ADOTable1.eof do begin
      xID:=ADOTable1.FieldByName('ID').AsInteger;
      SQL:='INSERT INTO Lebensmittel_Karten (kID,LID) VALUES ('+ IntToStr(ID)
           +', '+IntToStr(xID)+')';
      ADOConnection1.Execute(SQL);
      ADOTable1.next;
    end;
  end;
  InitKartenGruppen;
  KInserted:=false;
end;

procedure TMainForm.ADOQuery5BeforeDelete(DataSet: TDataSet);
```

```
begin
  if ADOQuery5.FieldByName('ID').AsInteger<>kID[nK] then begin
    ShowMessage('Nur die letzte Karte kann gelöscht werden!');
    Abort;
  end;
  if MessageDlg('Wollen Sie diese Karte wirklich löschen?',
                mtConfirmation,[mbYes,mbNo],0)=mrNo then
    Abort;
end;

procedure TMainForm.ADOQuery5AfterDelete(DataSet: TDataSet);
begin
  InitKartenGruppen;
end;

// ---- Karten : Query7 : ZMs PIs
procedure TMainForm.ADOQuery7BeforePost(DataSet: TDataSet);
var i,x:integer; s,d,LID,k,SQL:string;
begin
  LID:=DBGrid1.Columns[0].Field.Text;  //DBGrid1.Fields[0].Text; is an error, why??
  if LID<>'' then begin
    s:=DBgrid1.SelectedField.Text;
    d:=DBGrid1.Columns[DBGrid1.SelectedIndex].Title.Caption;
    k:='a'+IntToStr(K1); if DBGrid1.SelectedIndex>4 then k:='a'+IntToStr(K2);
    SQL:='UPDATE '+k+' SET '+d+'='+s+' WHERE c='+LID;
    ADOConnection1.Execute(SQL);
  end;
  Q7Edit:=false;
  ADOQuery7.CancelUpdates;
  ADOQuery7.Cancel;
  ADOQuery7.Close; ADOQuery7.Open;
  if LID<>'' then ADOQuery7.Locate('Expr1000',LID,[]);
  Abort;
end;

procedure TMainForm.DBGrid1DblClick(Sender: TObject);
var s,d,k,LID,SQL:string;
begin
  LID:=DBGrid1.Columns[0].Field.Text;
  case DBGrid1.SelectedIndex of 5,7 : d:='PI';  6,8,9: d:='ZM'; else d:=''; end;
  if d='' then exit;
  s:=DBgrid1.SelectedField.Text;
  k:='a'+IntToStr(K1); {a1}
  { UPDATE a1 SET ZM=s WHERE c=LID;  // for example}
  SQL:='UPDATE '+k+' SET '+d+'='+s+' WHERE c='+LID;
  ADOConnection1.Execute(SQL);
  ADOQuery7.Close; ADOQuery7.Open;
  ADOQuery7.Locate('Expr1000',LID,[]);
end;

procedure TMainForm.DBGrid1TitleClick(Column: TColumn);
var i:integer; s,d,k,m,SQL:string;
begin
  i:=DBGrid1.SelectedIndex;
  if i=0 then begin Q7sort:='L.IDsort'; ExecKarteQuery; end;
  if i=11 then begin Q7sort:='-L.Zuteil'; ExecKarteQuery; end;
  if i=12 then begin Q7sort:='-INT(L.Zuteil*L.Kal/1000)'; ExecKarteQuery; end;
  case i of 5,7 : d:='PI';  6,8,9: d:='ZM';  else d:=''; end;
  if d='' then exit;
  if MessageDlg('Wollen Sie die ganze Kolonne links ersetzen?',
                mtConfirmation,[mbYes,mbNo],0)=mrNo then  exit;
  s:=DBGrid1.Columns[i].Title.Caption;
  k:='a'+IntToStr(K1); m:='a'+IntToStr(K2); if i<4 then m:=k; {a1,a2}
  { UPDATE a1,a2 SET a1.ZM=a2.ZMopt WHERE a1.c=a2.c;  // for example}
  if m=k then SQL:='UPDATE '+k+' SET '+d+'='+s
  else SQL:='UPDATE '+k+', '+m+' SET '+k+'.'+d+'='+m+'.'+s+' WHERE '+k+'.c='+m+'.c';
  ADOConnection1.Execute(SQL);
  ADOQuery7.Close; ADOQuery7.Open;
end;

procedure TMainForm.DBGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if (Key=#13) then ADOQuery7.Post;
end;

procedure TMainForm.UpDown1Click(Sender: TObject; Button: TUDBtnType);
```

```
begin
  if Button=btNext then begin inc(K1); if K1>nK then K1:=1; end
  else begin dec(K1); if K1<1 then K1:=nK; end;
  ExecKarteQuery;
end;

procedure TMainForm.UpDown2Click(Sender: TObject; Button: TUDBtnType);
begin
  if Button=btNext then begin inc(K2); if K2>nK then K2:=1; end
  else begin dec(K2); if K2<1 then K2:=nK; end;
  ExecKarteQuery;
end;

procedure TMainForm.ADOQuery7BeforeEdit(DataSet: TDataSet);
begin
  Q7Edit:=true;
end;

procedure TMainForm.DBGrid1Exit(Sender: TObject);
begin
  if Q7Edit then ADOQuery7.Post;
end;

procedure TMainForm.DBNavigator5Click(Sender: TObject; Button: TNavigateBtn);
begin
  if Button=nbDelete then KInserted:=false;
end;

// ------------------------------ KONSUMENTEN --------------------------------
procedure TMainForm.ADOTable4AfterInsert(DataSet: TDataSet);
begin
  G1Inserted:=true;
  ADOTable4.FieldByName('Basis').AsBoolean:=true;
end;

procedure TMainForm.ADOTable4AfterPost(DataSet: TDataSet);
var i,ID,Anz:integer; SQL:string;
begin
  ADOTable4.FieldByName('Zuschlag').AsString :=
    '= '+ADOTable4.FieldByName('Code').AsString;
  if G1Inserted then begin
    ADOTable4.Close; ADOTable4.Open; ADOTable4.Last; // for autoinc field ID
    ID:=ADOTable4.FieldByName('ID').AsInteger;
    for i:=1 to nK do begin
      SQL:='INSERT INTO Gruppen_Karten (gID,kID) VALUES ('+ IntToStr(ID)+', '
           +IntToStr(kID[i])+')';
      ADOConnection1.Execute(SQL);
    end;
  end
  else begin
    ID:=ADOTable4.FieldByName('ID').AsInteger;
    for i:=1 to nK do begin
      Anz:=ADOTable4.FieldByName('K'+IntToStr(i)).AsInteger;
      SQL:='UPDATE Gruppen_Karten SET Anzahl = '+IntToStr(Anz)+
           ' WHERE gID='+IntToStr(ID)+' AND kID='+IntToStr(kID[i]);
      ADOConnection1.Execute(SQL);
    end;
  end;
  G1Inserted:=false;
  GruppenTotale;
end;

procedure TMainForm.ADOTable4BeforeDelete(DataSet: TDataSet);
begin
  if MessageDlg('Wollen Sie die Gruppe wirklich löschen?',
                mtConfirmation,[mbYes,mbNo],0)=mrNo then
    Abort;
end;

procedure TMainForm.ADOTable4AfterDelete(DataSet: TDataSet);
begin
  G1Inserted:=false;
  GruppenTotale;
end;

procedure TMainForm.ADOTable5AfterInsert(DataSet: TDataSet);
```

```
begin
  G2Inserted:=true;
  ADOTable5.FieldByName('Basis').AsBoolean:=false;
end;

procedure TMainForm.ADOTable5AfterPost(DataSet: TDataSet);
var i,ID,Anz:integer; SQL:string;
begin
  ADOTable5.FieldByName('Zuschlag').AsString :=
    '= '+FirstCode+' '+ADOTable5.FieldByName('Code').AsString;
  if G2Inserted then begin
    ADOTable5.Close; ADOTable5.Open; ADOTable5.Last; // for autoinc field ID
    ID:=ADOTable5.FieldByName('ID').AsInteger;
    for i:=1 to nK do begin
      SQL:='INSERT INTO Gruppen_Karten (gID,kID) VALUES ('+ IntToStr(ID)
           +', '+IntToStr(kID[i])+')';
      ADOConnection1.Execute(SQL);
    end;
  end
  else begin
    ID:=ADOTable5.FieldByName('ID').AsInteger;
    for i:=1 to nK do begin
      Anz:=ADOTable5.FieldByName('K'+IntToStr(i)).AsInteger;
      SQL:='UPDATE Gruppen_Karten SET Anzahl = '+IntToStr(Anz)+
           ' WHERE gID='+IntToStr(ID)+' AND kID='+IntToStr(kID[i]);
      ADOConnection1.Execute(SQL);
    end;
  end;
  G2Inserted:=false;
  GruppenTotale;
end;

procedure TMainForm.ADOTable5BeforeDelete(DataSet: TDataSet);
begin
  if MessageDlg('Wollen Sie die Gruppe wirklich löschen?',
                mtConfirmation,[mbYes,mbNo],0)=mrNo then
    Abort;
end;

procedure TMainForm.ADOTable5AfterDelete(DataSet: TDataSet);
begin
  G2Inserted:=false;
  GruppenTotale;
end;

procedure TMainForm.DBNavigator6Click(Sender: TObject; Button: TNavigateBtn);
begin
  if Button=nbDelete then G1Inserted:=false;
end;

procedure TMainForm.DBNavigator7Click(Sender: TObject; Button: TNavigateBtn);
begin
  if Button=nbDelete then G2Inserted:=false;
end;

procedure TMainForm.TabSheet2Show(Sender: TObject);
begin
  ExecKarteQuery;
end;

// ------------------------------- SZENARIOS --------------------------------
procedure TMainForm.GetScenarioName;
begin
  ADOTable7.TableName:='_PARA'; ADOTable7.IndexFieldNames:='';
  ADOTable7.Open;
  scen:=ADOTable7.FieldByName('scen').AsInteger;
  per:=ADOTable7.FieldByName('per').AsInteger;
  scNa.Caption:=ADOTable7.FieldByName('scenName').AsString;
  ADOTable7.Close;
end;

procedure TMainForm.PutScenarioName;
begin
  ADOTemp.TableName:='_PARA'; ADOTemp.Open; ADOTemp.Edit;
  ADOTemp.FieldByName('scen').AsInteger:=scen;
  ADOTemp.FieldByName('per').AsInteger:=per;
```

```
  ADOTemp.FieldByName('scenName').AsString:=scNa.Caption;
  ADOTemp.Post; ADOTemp.Close;
  ADOTable7.TableName:='_PARA'; ADOTable7.IndexFieldNames:='';
  ADOTable7.Open; ADOTable7.Edit;
  ADOTable7.FieldByName('scen').AsInteger:=scen;
  ADOTable7.FieldByName('per').AsInteger:=per;
  ADOTable7.FieldByName('scenName').AsString:=scNa.Caption;
  ADOTable7.Post; ADOTable7.Close;
end;

procedure TMainForm.InitScenario;
begin
  ListBox3.Clear; nSc:=0;
  ADOTemp.TableName:='_Szenarios'; ADOTemp.Open;
  while not ADOTemp.eof do begin
    inc(nSc);
    AllScen[nSc]:=ADOTemp.FieldByName('scen').AsInteger;
    AllPer[nSc]:=ADOTemp.FieldByName('per').AsInteger;
    ListBox3.Items.Add(ADOTemp.FieldByName('Bezeichnung').AsString);
    ADOTemp.next;
  end;
  ADOTemp.Close;
  ListBox3.ItemIndex:=0;
end;

procedure TMainForm.SaveScenario(ask:boolean);
var i:integer; SQL:string;
begin
  if ask then
    if MessageDlg('Wollen Sie das aktuelle Scenario/Periode speichern',
       mtConfirmation,[mbYes,mbNo],0)=mrNo then exit;
  if ask then
    if MessageDlg('Sind Sie sicher?'#13#10+'Das gleichnamige Szenario in der '+
      'Szenariodatenbank wird vom aktuellen überschrieben.',
       mtConfirmation,[mbYes,mbNo],0)=mrNo then exit;
  ADOConnection2.BeginTrans;
  for i:=0 to AllTables.count-1 do begin
    SQL:='DELETE FROM '+AllTables[i]+
         ' WHERE scen='+IntToStr(scen)+' AND per='+IntToStr(per);
    ADOConnection2.Execute(SQL);
  end;
  ADOConnection2.CommitTrans;
  ADOConnection1.BeginTrans;
  for i:=0 to AllTables.count-1 do begin
    UpdateStatusBar('Tabelle '+AllTables[i]+' speichern');
    SQL:='INSERT INTO '+AllTables[i]+' (scen,per,'+AllFields[i]+') IN '''+MySCENDB+
         ''' SELECT '+IntToStr(scen)+', '+IntToStr(per)+', '+AllFields[i]
         +' FROM '+AllTables[i];
    ADOConnection1.Execute(SQL);
  end;
  ADOConnection1.CommitTrans;
  UpdateStatusBar('Scenario gespeichert');
end;

procedure TMainForm.SaveScen(Sender: TObject);
begin
  SaveScenario(true);
end;

procedure TMainForm.DeleteScen(Sender: TObject);
var i,k:integer; SQL:string;
begin
  k:=GetListBoxIndex(ListBox3);
  if (scen=AllScen[k]) and (per=AllPer[k]) then begin
    ShowMessage('Das aktuelle Scenario/Periode kann nicht gelöscht werden'); exit;
  end;
  for i:=1 to nSc do
    if (AllScen[i]=AllScen[k]) and (AllPer[i]>AllPer[k]) then begin
      ShowMessage('Nur die letzte Periode kann gelöscht werden!'); exit;
    end;
  if MessageDlg('Wollen Sie das Scenario/Periode "'
                +ListBox3.Items[ListBox3.ItemIndex]+
      '"'#10#13' wirklich aus der Scenariodatenbank löschen?',
      mtConfirmation,[mbYes,mbNo],0)=mrNo then exit;

  ADOConnection2.BeginTrans;
```

```
  for i:=0 to AllTables.count-1 do begin
    UpdateStatusBar('Löschen in '+AllTables[i]);
    SQL:='DELETE FROM '+AllTables[i]+
         ' WHERE scen='+IntToStr(AllScen[k])+' AND per='+IntToStr(AllPer[k]);
    ADOConnection2.Execute(SQL);
  end;
  SQL:='DELETE FROM _Szenarios WHERE scen='+IntToStr(AllScen[k])+' AND per='
      +IntToStr(AllPer[k]);
  ADOConnection2.Execute(SQL);
  ADOConnection2.CommitTrans;
  InitScenario;
  UpdateStatusBar('Scenario wurde gelöscht');
end;

procedure TMainForm.LoadScen(Sender: TObject);
var i,k:integer; SQL:string;
begin
  SaveScenario(true);
  scNa.Caption:=ListBox3.Items[ListBox3.ItemIndex];
  k:=GetListBoxIndex(ListBox3);
  scen:=AllScen[k]; per:=AllPer[k];
  PutScenarioName;
  ADOConnection1.BeginTrans;
  for i:=AllTables.count-1 downto 0 do begin
    SQL:='DELETE * FROM '+AllTables[i];
    ADOConnection1.Execute(SQL);
  end;
  ADOConnection1.CommitTrans;
  ADOConnection2.BeginTrans;
  for i:=0 to AllTables.count-1 do begin
    UpdateStatusBar('Tabelle '+AllTables[i]+' laden');
    SQL:='INSERT INTO '+AllTables[i]+' ('+AllFields[i]+') IN '''+MyDB+
         ''' SELECT '+AllFields[i]+' FROM '+AllTables[i]+
         ' WHERE scen='+IntToStr(scen)+' AND per='+IntToStr(per);
    ADOConnection2.Execute(SQL);
  end;
  ADOConnection2.CommitTrans;
                                //exit; ////////////////////////////////////
  ShowMessage('Warten Sie 2-3 Sekunden!');
  Refresh1Click(Sender);
  UpdateStatusBar('Szenario geladen');
end;

procedure TMainForm.NewScen(Sender: TObject);
begin
  Randomize;
  scen:=trunc(Random*100000);
  per:=1;
  Label22.Visible:=true;
  Edit1.Visible:=true;
  ScenGenerate.Visible:=true;
end;

procedure TMainForm.ScenGenerateClick(Sender: TObject);
var i,j:integer; SQL:string;
begin
  if Edit1.Visible then scNa.Caption:=Edit1.Text+', Periode 1';
  ADOConnection2.BeginTrans;
  PutScenarioName;
  SQL:='INSERT INTO _Szenarios (scen,per,Bezeichnung) VALUES ( '''+
       IntToStr(scen)+''' , '''+IntToStr(per)+''' , '''+scNa.Caption+''')';
  ADOConnection2.Execute(SQL);
  ADOConnection2.CommitTrans;
  SaveScenario(false);
  InitScenario;
  Label22.Visible:=false;
  Edit1.Visible:=false;
  ScenGenerate.Visible:=false;
end;

procedure TMainForm.NextPeriod(Sender: TObject);
var i,k:integer; SQL:string;
begin
  k:=GetListBoxIndex(ListBox3);
  for i:=1 to nSc do
    if (AllScen[i]=AllScen[k]) and (AllPer[i]>AllPer[k]) then begin
```

```
      ShowMessage('Aktuelle Periode muss die letzte sein!');
      exit;
    end;
  inc(per);
  scNa.Caption:=copy(scNa.Caption,1,length(scNa.Caption)-1)+' '+IntToStr(per);

  {Button12Click(Sender);}
end;

procedure TMainForm.ScenExportClick(Sender: TObject);
begin
  CopyFile('rap.mdb','export.mdb',false);
  ShowMessage('Aktuelles Szenario als EXPORT.MDB abgespeichert!');
end;

procedure TMainForm.ScenImportClick(Sender: TObject);
begin
  {SaveScenario(true);
  CloseRapDB;
  copyFile('export.mdb','rap.mdb',false); }

end;

// ------------------------------ OPTIMIEREN ---------------------------------

procedure TMainForm.OptimierenClick(Sender: TObject);
begin
  if Optimieren.Caption='Optimieren' then Optimize else ErrorOnExit:=true;
  ReOpenRapDB;
end;

procedure TMainForm.LPLModellClick(Sender: TObject);
const prog='rap.lpl';
var i:THandle;
begin
  i:=ShellExecute(Application.MainForm.Handle, nil,prog,'','',3);
  if i<=31 then ShowMessage('LPLW.EXE nicht gefunden');
end;

// ------------------------------ RESULTATE ----------------------------------

procedure TMainForm.ReadNOMHeaders;
var fil:TextFile; s:string[255];
begin
  IntSchema.Clear;
  if FileExists('rap.nom') then begin
    assignFile(fil,'rap.nom'); reset(fil);
    while not eof(fil) do begin
      readln(fil,s);
      if (pos('>',s)=1) and (s[2]<>' ') then begin
        delete(s,1,1);
        IntSchema.Items.Add(s);
      end;
    end;
    closeFile(fil)
  end else
    IntSchema.Items.Add('Report-Datei existiert nicht');
end;

procedure TMainForm.IntSchemaClick(Sender: TObject);
var fil:TextFile; s,s1:string[255]; f:boolean;
begin
  RichEdit1.Clear; f:=false;
  s1:=IntSchema.Items[IntSchema.ItemIndex];
  if FileExists('rap.nom') then begin
    assignFile(fil,'rap.nom'); reset(fil);
    while not eof(fil) do begin
      readln(fil,s);
      if (pos('>',s)=1) and (s[2]<>' ') then begin
        delete(s,1,1);
        if s1=s then repeat
          RichEdit1.Lines.Add(s);
          readln(fil,s);  f:=true;
        until eof(fil) or (pos('>',s)=1);
        if f then break;
      end;
```

```
    end;
    closeFile(fil);
    RichEdit1.Lines[0]:=RichEdit1.Lines[0]+' // '
        +scNa.Caption+ ' ('+GetDateAndTime+')';
  end;
end;

procedure TMainForm.StartEditorClick(Sender: TObject);
const prog='temp.doc';
var i:THandle;
begin
  RichEdit1.Lines.SaveToFile('temp.doc');
  i:=ShellExecute(Application.MainForm.Handle, nil,prog,'','',3);
  if i<=31 then ShowMessage('Kann kein Editor für *.doc Dateien finden');
end;

procedure TMainForm.AusdruckenClick(Sender: TObject);
begin
  RichEdit1.Lines.SaveToFile('temp.doc');
  RichEdit1.Print('');
end;

procedure TMainForm.TabSheet7Show(Sender: TObject);
begin
  ReadNOMHeaders;
end;

procedure TMainForm.ExtSchemaClick(Sender: TObject);
var i,k:integer;
begin
  k:=GetListBoxIndex(ExtSchema)-1;
  case k of
    0 : QuickReport1.Preview;
    1 : QuickReport2.Preview;
    2 : begin QuickReport3.ADOTable1.Open;
          QuickReport3.Preview; QuickReport3.ADOTable1.Close; end;
    3 : QuickReport4.Preview;
    4 : QuickReport5.Preview;
    5 : QuickReport6.Preview;
    6 : QuickReport7.Preview;  {Tab 7}
    7 : QuickReport8.Preview;  {Tab 9}
    8 : QuickReport9.Preview;
    9 : QuickReport10.Preview;
  end;
end;

end.
```

Anhang C : Portierung Access 2 nach Access 97

Hier noch ein kurzer Erfahrungsbericht über die fehlgeschlagenen Portierung der Access2 Implementation in die Access Version 97. Als erster Schritt musste also die Datenbank, die sich in der Version von Access 2 befand, in eine Version, die unter Access97 lauffähig ist “portiert” werden. Man könnte natürlich annehmen, dass eine solche Portierung gar nicht nötig ist, da ja das Produkt Access 2 und Access 97 dasselbe Produkt sind, nur eben zwei verschiedene Versionen. Weit gefehlt! Wenn die Datenbankensysteme schon nicht kompatibel sind, so sollte es wenigstens Uebersetzungsprogramme geben. Gewiss doch! Wenn die alte unter Access 2 erstellte Datenbank mit Access 97 geöffnet wird, so wird automatisch eine Portierung vorgeschlagen: “Diese Datenbank ist in einem älteren Format, wollen sie diese ins neuere Format übersetzen? (Ja/Nein).” Click: Ja, natürlich! Es läuft flott. Keine Fehlermeldungen, nichts. Wenn die Datenbank, beziehungsweise sein Hauptmodul, jedoch gestartet wird, so erscheint eine ganze Reihe sinnloser Fehlermeldungen der Art: “DLL-Datei XYZ nicht gefunden”, “Fehler 234 im Module ABC”.

So, nun muss Hand angelegt werden. Mit Hilfe der kompetenten Hand von Marco Moresino gelang es schiesslich nach Abänderungen einiger Module, das Programm unter Access 97 zum Laufen zu bringen: Investition; 4 Mann-Tage – dabei kann nicht gesagt werden, dass es sich bei der Datenbank um ein kompliziertes Gebilde handelt.

Die Optimierung (das heisst, ein Teil des Kodes) lief dann doch nicht! Und hier komme ich zu einem der bemerkenswertesten Detail dieser Uebersetzung, die eigentlich automatisch hätte erfolgen sollen. Ich möchte damit einen Bug von Mirkosoft dokumentieren, der mich zusätzlich 5 Stunden gekostet hat, bis der Bug seine Essenz preisgab. Die Dokumentation lohnt sich, denn es ist so absurd wie es traurig ist, wie schluderig Software – auch von den ganz Grossen –geschrieben werden kann. Natürlich ist der Bug in der Access Dokumentation nirgends dokumentiert. Aus Sicht von Mikrosoft handelt es sich möglicherweise gar nicht um einen Bug.

Die Essenz des Bugs ist schnell erzählt: Die Funktion *InStr()* in VisualBasic nimmt zwei String-Parameter (nennen wir sie *a* und *b*) entgegen und liefert einen Integer zurück. Wenn *a* als Unterstring im String *b* vorkommt, liefert die Funktion die Position des ersten Zeichens des ersten Vorkommens von *a* in *b* von links nach rechts. Wenn *a* in *b* nicht vorkommt, liefert die Funktion 0 (null) zurück. Soweit so gut, eine einfache Stringfunktion, könnte man meinen! Die Funktion reagiert jedoch anders in den beiden Access Versionen.
Wenn beispielsweise *a=chr(10)* (ein Linefeed) ist, so wird korrekt in *b,* in der mindestens ein *chr(10)* enthalten ist, die Position in beiden Versionen zurückgeliefert. Wenn aber *a=chr(10)* und *b* ein *chr(13)* enthält, so liefert die *InStr()* Version von Access 2 ebenfalls die Position von *chr(13)* zurück. Alle Zeichen von *chr(0)* bis *chr(31)* sind in der Access 2 *InStr()* Funktion dasselbe Zeichen mit Ausnahme von *chr(9)* (der Tabulator). Wird also ein *chr(7)* (bell) in b gesucht der ein chr(15), beispielsweise, enthält (aber keine andern Escape-Zeichen, so wird die Position von chr(15) zurückgeliefert.
In der *Instr()* Version von Access 97 werden alle Escape-Zeichen voneinander unterschieden; d.h. wird, z.B., ein chr(7) in b, welches ein chr(15) enthält, gesucht, so liefert *InStr()* Null zurück (sofern kein chr(7) in b existiert.
So einfach kann das Verhalten einer Basisfunktion verändert werden, ohne dass ihr Verhalten dokumentiert wird!

Der Bug manifestierte sich jedoch – wie dem so üblich ist – an einem ganz andern Ort der Implementation. Die Implementation von RAP muss als Zwischenschritt der Optimierung die Text-Dateien *Tab-1.dat* bis *Tab-11.dat* generieren. Diese dienen als Input für das Turbo-Pascal *RAP-SS.EXE,* welches seinerseits die drei LPL-Dateien *modset.dat, modcoef.dat* und *smcoef.dat* vor einer Optimierung produziert. Die Manifestation war, dass das Dos-Programm RAP-SS.EXE in einen unendlichen Loop hing. Die Pascal-Source-Datei RAP-SS.PAS musste als ausgegraben werden. Turbo-Pascal neu installiert werden. Schliesslich konnte die Source-Code Position eruiert werden. Es stellte sich schliesslich heraus, dass die *READLN*-Funktion zwei Zeilen aufs Mal in der *Tab-4.dat* Datei liest. Wie bitte? Wie ist das möglich? Eine nähere Analyse ergab schliesslich, dass die beiden Zeilen nur durch ein chr(13) , statt wie üblich im Dos/Windows durch die Kombination chr(13)+chr(10) voneinander getrennt waren. Die READLN Funktion liest aber bis zum nächsten chr(10), dies ist als EOLN-Zeichen definiert.

Soweit war die Sache jetzt klar: In einigen Dateien *Tab-1.dat* bis *Tab-11.dat* waren an einigen Stellen ein einfacher chr(13) anstatt ein chr(13)+chr(10) produziert worden. Aber wo? Da diese Zeichen nicht sichtbar sind, ist es gar nicht so leicht festzustellen, wo ein chr(13) anstatt ein chr(13)+chr(10) steht.[1] Schliesslich stellte ich eher durch Zufall als durch Kenntnis fest, dass der einfach Editor *Notepad* unter Windows, ein einzelner chr(13) als □ darstellt. Damit wurde festgestellt, dass an genau drei Stellen in verschiedenen *Tab-??.dat* Dateien dies vorkommt. Wieso hier und nicht systematisch überall?

Nach längerer Suche im Access-Code konnte schliesslich eruiert werden, dass in der Funktion *Function tab_4_5_transformation(tabname)* im Modul *Export* es einen vom oben beschriebenen Bug gibt. Die folgende Zeile in der einem Fuktionsaufruf ist fehlerhaft:

```
fin = InStr(zeichen, Chr(10))
```

Sie müsste heissen:

```
fin = InStr(zeichen, Chr(13) & Chr(10))
```

Auch an einer ganz andern Stelle des Code kommt dieser Bug vor. Nachdem dieser Fehler behoben war, wurden in beiden Versionen die korrekten *Tab-??.dat* Dateien generiert. Dabei hätte man es belassen können. Schliesslich funktioniert's jetzt ja!! Aber warum das Programm jetzt korrekt lief und vorher nicht, dafür konnte keine Erklärung gegeben werden. *Solange auf eine solche Frage keine befiedigende Antwort gegeben werden kann, nagt der Zahn der Ungewissheit, ob das Programm nun auch in Zukunft richtig laufen wird*. Der Warum-Frage muss daher *immer* nachgegangen werden. Detail-Analysen, auf die hier jetzt verzichtet wird, sie zu geschrieben, ergaben schiesslich, dass an genau drei Stellen in einen Loop auf Grund des zweiten Fehlers in der Access 2 Version die Position des Zeichens chr(13) anstatt von chr(10) zurückgegeben wurde, das heisst, da chr(13) jeweils ein Zeichen vor chr(10) in einer Zeile vorkommt war die Variable *fin* um eine Einheit kleiner als vom Programmierer (Horner) angenommen. In der Access 97 Version aber wurde nicht die Position von chr(13) sondern die von chr(10) zurückgegeben. Da der String abgeschnitten wurde, ist er in Access 97 mit zusätzlich einem chr(13) versehen, was das einzelne chr(13) in den Dateien erklärt.

An diesem Beispiel sieht man: (1) wie scheinbar Unwahrscheinliches zusammenwirken muss, bis ein subtil verborgener Fehler sich manifestiert, (2) dass das scheinbar Unwahrscheinliche in der Softwareprogrammierung viel wahrscheinlicher als gemeinhin angenommen ist, (3) dass Fehler sich gegenseitig aufheben und Schachmatt setzen können, so nach dem Motto: *Ein Fehler + ein Fehler = Kein Fehler!*

Nachdem all diese Fehler behoben waren, konnten fast alle Funktion in der neuen Access 97 Implementation ausgeführt werden. Nach einiger Zeit stellte sich jedoch

---

[1] . Unter Unix könnte man jetzt *grep* einsetzen, aber unter den armseligen Systemen wie Dos/Windows ist man natürlich weit von solch mächtigen Tools entfernt. Man hätte natürlich jetzt eine Internet-Suche für so ein Tool veranstalten können. Das gibt es bestimmt.

heraus, dass das Module Szenario-laden, speichern usw. nicht richtig funktionierte und das Programm weiterhin abstürtzte. Daraufhin wurde beschlossen, eine von Grund auf neu konzipierte Re-implementation durchzuführen.

Moral der Geschicht: Benütze nie Access zur Implementation von Applikationen. Der Code wird so unübersichtlich und unwartbar, dass leichte Veränderungen das ganze System unstabil werden lassen. Access mag als Kleinserver-Datenbank und zur Implementation von kleinen, übersichtlichen Modulen genügend sein. Aber eine vollständig Applikation damit durchzuführen ist ein Nightmare. Ganz abgesehen von den Portierungsschwierigkeiten.

Anhang D: Portierung LPL 3.5 nach LPL 4.40

Als zweiter Schritt, ohne die Architektur von Horner im wesentlichen zu verändert, war die Portierung der mathematischen Modells von LPL Version 3.5 nach LPL 4.40. Eine solche Portierung war notwendig, da die beiden LPL Sprachversionen sich sehr voneinander unterscheiden. Dazu ist zu sagen, dass LPL selber kein kommerzielles Produkt, sondern ein Forschungsprojekt ist. Das Sprachelement der hierarchischen Sets wurde aus der Sprache entfernt, dafür kamen eine ganze Reihe neuerer Sprachkonstrukte hinzu. Der Portierungsaufwand war somit beträchtlich, das insbesondere, als das Rap-Modell der alten Version, stark vom Sprachkonstrukt der hierarchischen Indexsets abhängt.
Bei der Portierung wurde auch eine ganze Reihe von Ausdrücken vereinfacht und der gesamte Code übersichtlicher gestaltet. Dadurch wurden die Ausdücke um einiges kürzer und transparenter. Dies konnte hauptsächlich auf Grund der Aliasnamen und dank einer intelligenteren Bindung der passiven an die aktiven Indexset – wie sie im LPL 4.40 jetzt realisiert wurde – erreicht werden. Da Zusammenwirken der Aliasnamen mit einem neueren Verfahren der Bindung kann an der folgenden Beschränkung gesehen werden. Die lineare Beschränkung *RSLG*, beispielsweise, musste in LPL 3.5 folgendermassen geschrieben werden:

```
RSLG: SL=SUM(k=Lager,i=Lager |KALKOR(k,i))
  (SL1L2P(k,i) + SL1L2N(k,i) + 3 * (XL1L2P(k,i) + XL1L2N(k,i))
   + 10 * (ZL1L2P(k,i) + ZL1L2N(k,i))) * KALKOR(k,i);
```

In LPL 4.40 kann dieselbe Beschränkung verkürzt werden zu:

```
RSLG:
  SL=SUM{p,p1|K} (SL1L2P+SL1L2N+3*(XL1L2P+XL1L2N)+10*(ZL1L2P+ZL1L2N))*K;
```

Alle Indexlisten entfallen. Die Aliasnamen ersetzen die korrekte Position und die entsprechend richtige Bindung.

Noch frappanter wird die Vereinfachung im Report-Generator. Ein Ausdruck in LPL Version 3.5:

```
PRINT rtab3g
(IF(SUM(l=Bezuegergruppen|l>5 AND l<11)1>0;313;0),  'Gruppen',  'f ',
```

```
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
COL(l=Bezuegergruppen|l>5 AND l<11) (l,'I'),
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
COL(b=Bezuegergruppen|b>5 AND
b<11)(SUM(k=Karten)KARTENZAHL(b,k)*KxGy(k,1)*1000/30,'I'),
COL(b=Bezuegergruppen|b>5 AND
b<11)(SUM(k=Karten)KARTENZAHL(b,k)*KxGy(k,2)*1000/30,

SUM(k=Karten)KARTENZAHL(b,k)*KxGy(k,5)*1000/30,'I'),
COL(b=Bezuegergruppen|b>5 AND
b<11)(SUM(k=Karten)KARTENZAHL(b,k)*KxGy(k,3)*1000/30,

SUM(k=Karten)KARTENZAHL(b,k)*KxGy(k,6)*1000/30,'I'),
COL(b=Bezuegergruppen|b>5 AND
b<11)(SUM(k=Karten)KARTENZAHL(b,k)*KxGy(k,4)*1000/30,'I'),
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
ROW(k=Kartenbelegung|NOT (k in &Gruppe) AND RATIONIERT)
(k, COL(l=Bezuegergruppen|l>5 AND l<11)
(SUM(x=Karten)KARTENZAHL(l,x)*KxProd(x,k)*1000,'I')),
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
ROW(k=Kartenbelegung|NOT (k in &Gruppe) AND NOT RATIONIERT)
(k, COL(l=Bezuegergruppen|l>5 AND l<11)
(SUM(x=Karten)KARTENZAHL(l,x)*KxProd(x,k)*1000,'I')),
COL(l=Bezuegergruppen|l>5 AND l<11) '---------I',
IF(SUM(l=Bezuegergruppen|l>5 AND l<11)1>0;314;0),
COL(l=Bezuegergruppen|l>5 AND l<11)l,
COL(l=Bezuegergruppen|l>5 AND l<11)'Gramm   ',
ROW(i=#Gruppe)
(i,COL(l=Bezuegergruppen|l>5 AND l<11)
SUM(x=Karten)KARTENZAHL(l,x)*KxProd(x,i)*1000,
ROW(j=&i)
(j,COL(l=Bezuegergruppen|l>5 AND
l<11)SUM(x=Karten)KARTENZAHL(l,x)*KxProd(x,j)*1000)));
```

kann jetzt in LPL4.40 wie folgt verkürzt und übersichtlicher geschrieben werden:

```
WRITE "@rtab3g" :
 311,  'Gruppen',  ' ',
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} (bg,'I'),
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,1]*1000/30,'I'),
 COL{bg|bg<6}
   (SUM{k} KZ*KxGy[k,2]*1000/30,SUM{k} KZ*KxGy[k,5]*1000/30,'I'),
 COL{bg|bg<6}
   (SUM{k} KZ*KxGy[k,3]*1000/30,SUM{k} KZ*KxGy[k,6]*1000/30,'I'),
 COL{bg|bg<6} (SUM{k} KZ*KxGy[k,4]*1000/30,'I'),
 COL{bg|bg<6} '---------I',

 COL{bg|bg<6} '---------I',
 ROW{kb|~Gr AND RATIONIERT}
   (kb, COL{bg|bg<6} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg<6} '---------I',
 COL{bg|bg<6} '---------I',
 ROW{kb|~Gr AND ~RATIONIERT}
   (kb, COL{bg|bg<6} (SUM{k} KZ*Kxprod*1000,'I')),
 COL{bg|bg<6} '---------I',
 312,  COL{bg|bg<6} bg,
 COL{bg|bg<6} 'Gramm   ',
 ROW{mG} (mG,COL{bg|bg<6} SUM{k} KZ*Kxprod*1000,
   ROW{p|Gruppe[mG,p]} (p,COL{bg|bg<6} SUM{k} KZ*Kxprod*1000));
```

Eine solche Schreibweise schafft Uebersicht und ihre Notation ist einfacher zu lesen. Natürlich hätte die alte Schreibweise unter Verwendung der lokalen Indizes auch beibehalten werden können.
Da LPL 4.40 nun auch qualifizierte Kommentare erlaubt, wurden diese von der Rap.doc Word-Datei übernommen. Diese Rap.doc Source Datei ist nun in der LPL 4.40 nicht mehr nötig. Eine Uebersicht über das Modell und ein Browser im Modell wird von der integrierten Umgebung von LPL (LPLW.EXE) besser erreicht. Nur noch die Text-Datei RAP.LPL (sowie die RAPRAPO.LPL Datei) ist notwendig.

Anhang E: Re-Engeneering der Datenbank

Die Access Datenbank in ihrer vorherigen Version, wie sie von A. Horner übernommen habe, besteht aus:

23 Modulen
69 Makros
16 Reports
88 Formularen
86 Queries
und 22 Tabellen

Auf den ersten Blick scheint diese Zerlegung in verschiedene überschaubare Code- und Datenkomponeneten eine gelungene Sache zu sein. Wird doch dadurch nicht ein grosser, schwer zu lesender Spaghetti-Code produziert, sondern das ganze Projekt in kleine von einander unabhängigen und individuell wartbarem Codesegmenten aufgeteilt. Die Forderungen eines modernen Software-Engeneerings scheinen hier optimal erfüllt. Leider ist das nur Theorie. Die Abhängigkeiten zwischen den Teilen sind so vielfältig und intransparent, dass es enorm schwierig wird, den Code überhaupt zu verstehen, geschweige denn zu warten. Das liegt nicht etwa nur an einem schlechten Design des Implementierers, sondern es ist immanent durch die Access-Software vorprogrammiert. Es gibt beispielsweise gerade dadurch, dass es Makros *und* Module gibt, keine Möglichkeit herauszufinden, wo überall eine Funktion (definiert in einem Modul) aufgerufen wird: Wird sie in einem Makro oder vielleicht noch in einem Formular aufgerufen? Keine Ahnung! Dadurch wird es extrem schwierig, ein solches System überhaupt zu verstehen. Es ist quasi unmöglich, das System zu warten.
Wenn es unmöglich das System zu warten – man es aber von Zeit zu Zeit trotzdem tun muss – verhalten sich die Programmierer extrem konservativ: “Auf keinen Fall etwas berühren, was funktioniert.” Die Konsequenz ist – wenn eine neue Funktionalität eingebaut werden muss – möglichst den bestehenden Kode unberührt lassen und das Neue quasi als Nebengeleise zu implementieren. Dadurch werden Teile redundant und der Kode ist erst recht nicht mehr wartbar. Dies war genau der Fall in dieser Access 2 Implementation.

Die Konsequenz war, auch die Datenbank vollständig neu zu implementieren. Alle Modulen, Makros, Reports, Formulare und Queries wurden daher entfernt und durch eine einzige Delphi Applikation ersetzt, wie oben beschrieben. Die Tabellen wurden auf 13 Tabellen zusammengestrichen und in eine konsequente 3-normierte relationale Datenbank ersetzt. Ausserdem wurden die 77 Text-Dateien (im alten

Ordner *Archiv*, welche alle 7 Szenarien/Perioden Daten representierten) in die Szenario-Datenbank (SCEN.MDB) überführt. Dazu wurde zuerst ein Pascal-Programm geschrieben, welches die 11 Datentabellen der verschiedenen Szenarien je in eine Textdatei zusammenfasst. Der Kode ist im folgenden widergegeben (Prgramm Compact.pas):

```
program CollectScenarios;
uses dos;

type TextFile = text;
     str255   = string[255];
     str12    = string[12];
     TDirList = array[0..500] of str12;
var
  files:TDirList; fil,fil1,fil2:text;
  i,j,n,k,h,p:integer;
  s,s1,s2,s3,line:string;

function DirList(mask:str255; var A:TDirList):integer;
{ return a list of filename corresponding to the mask }
{ mask is: 'path\filePattern example: mask='c:\source\*.pas'}
var
  Info : SearchRec;
  i,n: integer;
begin
  n:=0;
  FindFirst(mask,Archive{Directory},Info);
  while DosError=0 do begin
    A[n]:=Info.name;
    {i:=pos('.',DirList[n]);
    if i>0 then DirList[n]:=copy(DirList[n],1,i-1);}
    FindNext(Info); inc(n);
  end;
  {FindClose(Info);}
  DirList:=n-1;
end;

function islower(c:char):boolean;
begin islower:= (c>='a') and (c<='z'); end;

procedure DoUpper(var s:str12);
{ translate all lowercase letters in s to uppercase letters }
var i:integer;
begin
  for i:=1 to length(s) do
    if islower(s[i]) then s[i]:=chr(ord(s[i])-32);
end;

begin
  n:=DirList('*.txt',files);
  for i:=0 to n do DoUpper(files[i]);
  for i:=1 to 11 do begin
    str(i,s); s1:=s+'.dat'; s:='_'+s+'.';
    assign(fil1,s1); rewrite(fil1);
    for j:=0 to n do if pos(s,files[j])>0 then begin
      k:=pos('_',files[j]); s1:=copy(files[j],1,k-1);
      h:=pos(s,files[j]); s2:=copy(files[j],length(s1)+2,1);
      assign(fil,files[j]); reset(fil);
      while not eof(fil) do begin
        readln(fil,line);

        if i=11 then begin
          if line[3]=#9 then begin
            s3:=#9+'10'+copy(line,1,2);
            line:=copy(line,1,2)+s3+copy(line,3,length(line));
          end else begin
```

```
            s3:=#9+copy(line,1,3);
            line:=copy(line,1,3)+s3+copy(line,4,length(line));
          end;
        end;
        if i=5 then begin
          h:=pos(#9,line); line:=copy(line,h+1,length(line));
        end;

        if i<>6 then line:=s1+chr(9)+s2+chr(9)+line;
        if (i=8) or (i=10) then begin
          h:=pos('1010',line); if h>0 then delete(line,h,1);
          h:=pos('1020',line); if h>0 then delete(line,h,1);
          h:=pos('1030',line); if h>0 then delete(line,h,1);
          h:=pos('1040',line); if h>0 then delete(line,h,1);
        end;
        writeln(fil1,line);
      end;
      close(fil);
    end;
    close(fil1);
  end;

  assign(fil,'3.dat'); reset(fil);
  assign(fil1,'6.dat'); reset(fil1);
  assign(fil2,'3x.dat'); rewrite(fil2);
  while not eof(fil) do begin
    readln(fil,line); readln(fil1,s);
    s:=copy(s,4,length(s));
    writeln(fil2,line,#9,s);
  end;
  close(fil2); close(fil1); close(fil);
end.
```

Die elf Tabellen wurden dann zuerst manuell in Excel eingelesen und als Tabelle in Access kopiert, wo verschiedenen Datenkonsistenz-Prüfungen vorgenommen wurden, bevor die Relationen über Fremdschlüssel bewerkstelligt werden konnten.

Bibliographie

HORNER A. [1994], Le Réengineering du Système d'Aide à la Décision DSS-RAP, Travail de diplôme, Institut für Informatik, Universität Freiburg.

HÜRLIMANN T., [2000], Reference Manual for the LPL Modeling Language, Version 4.40, Working Paper 0-05, Institute for Informatics, April, Fribourg, file *man440.pdf* at the LPL-site.

HÜRLIMANN T., [2000b], LPL: A mathematical programming language, Working Paper 0-06, Institute for Informatics, April, Fribourg,, file *arte44.pdf* at the LPL-site.

LANGENEGGER P. [1991], Formale Modellierung der Kunsumlenkung für die schweizerische Lebensmittelrationierung mit LPL 3.5, Diplomarbeit am Institut für Informatik, Universität Freiburg.

LPL-Site : HOMEPAGE at: http://www2-iiuf.unifr.ch/tcs/lpl/ and ARCHIVE at: ftp://ftp-iiuf.unifr.ch/pub/lpl/