Anhang zur Diplomarbeit

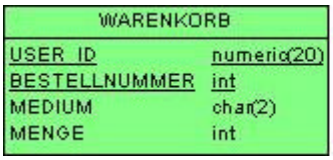
Michael Maretzke

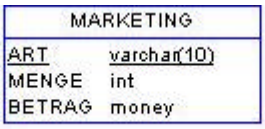
Evaluation einer Suchmaschine und Integration in eine existierende Internet-Shoppingapplikation

Anhang

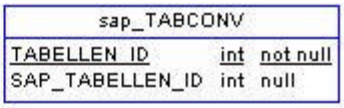
Anhang A – Datenmodellbilder

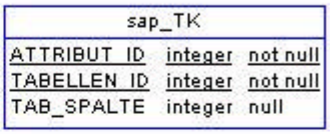
Datenmodell der Bestellung

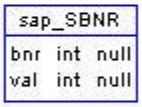


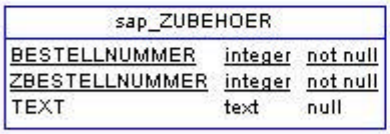


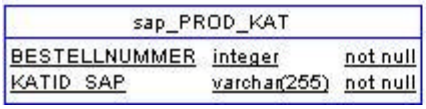
Datenmodell für den SAP-Import



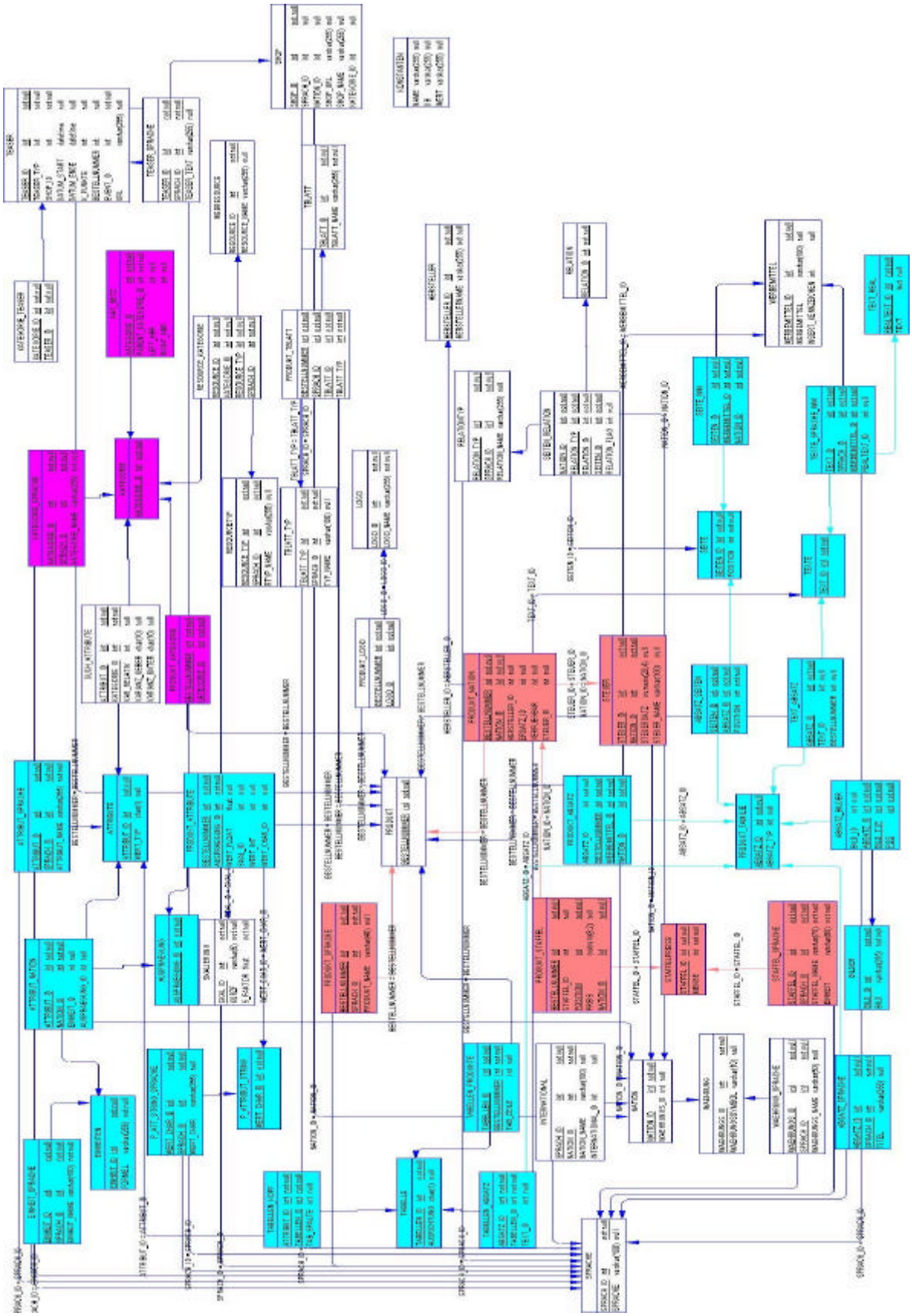




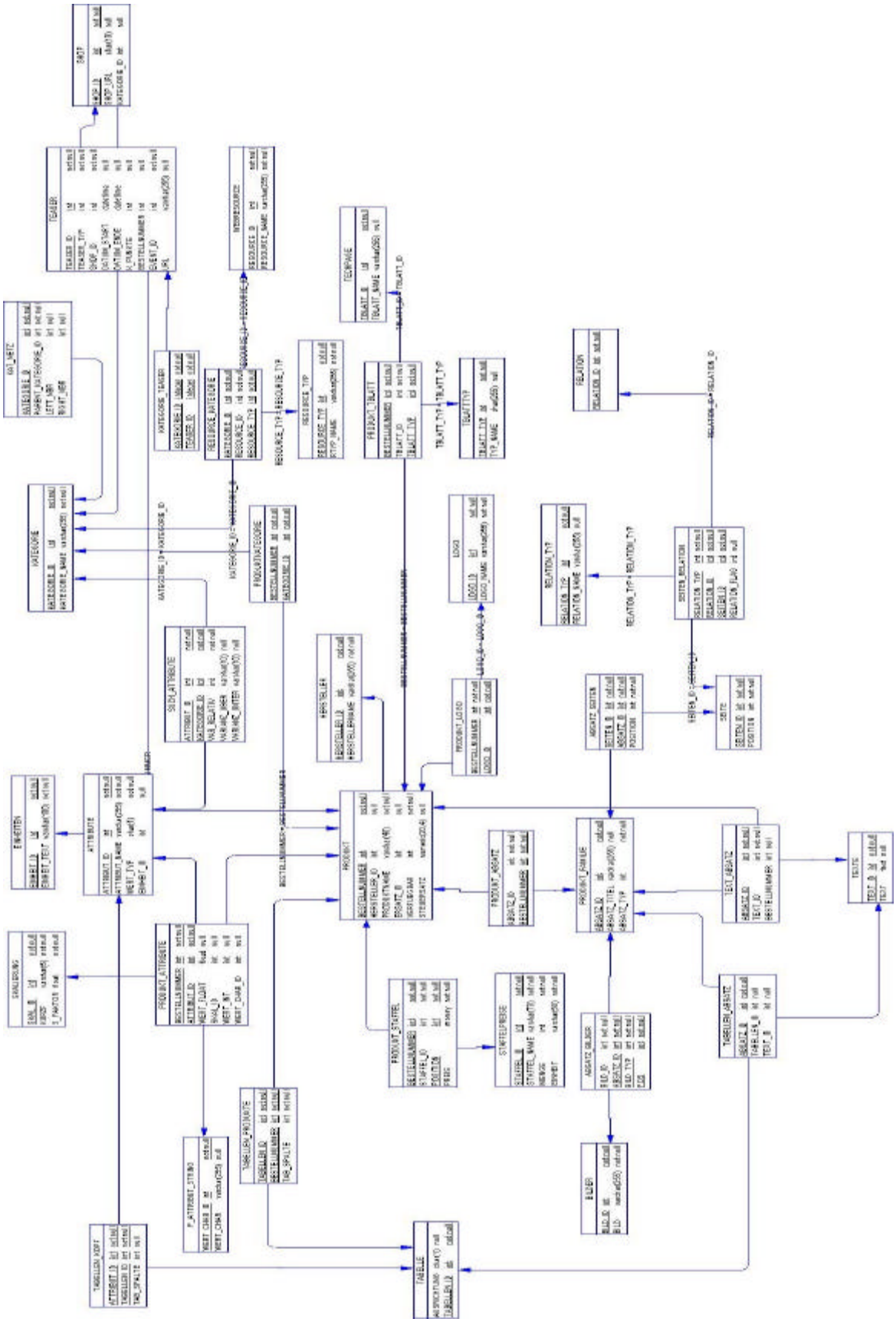




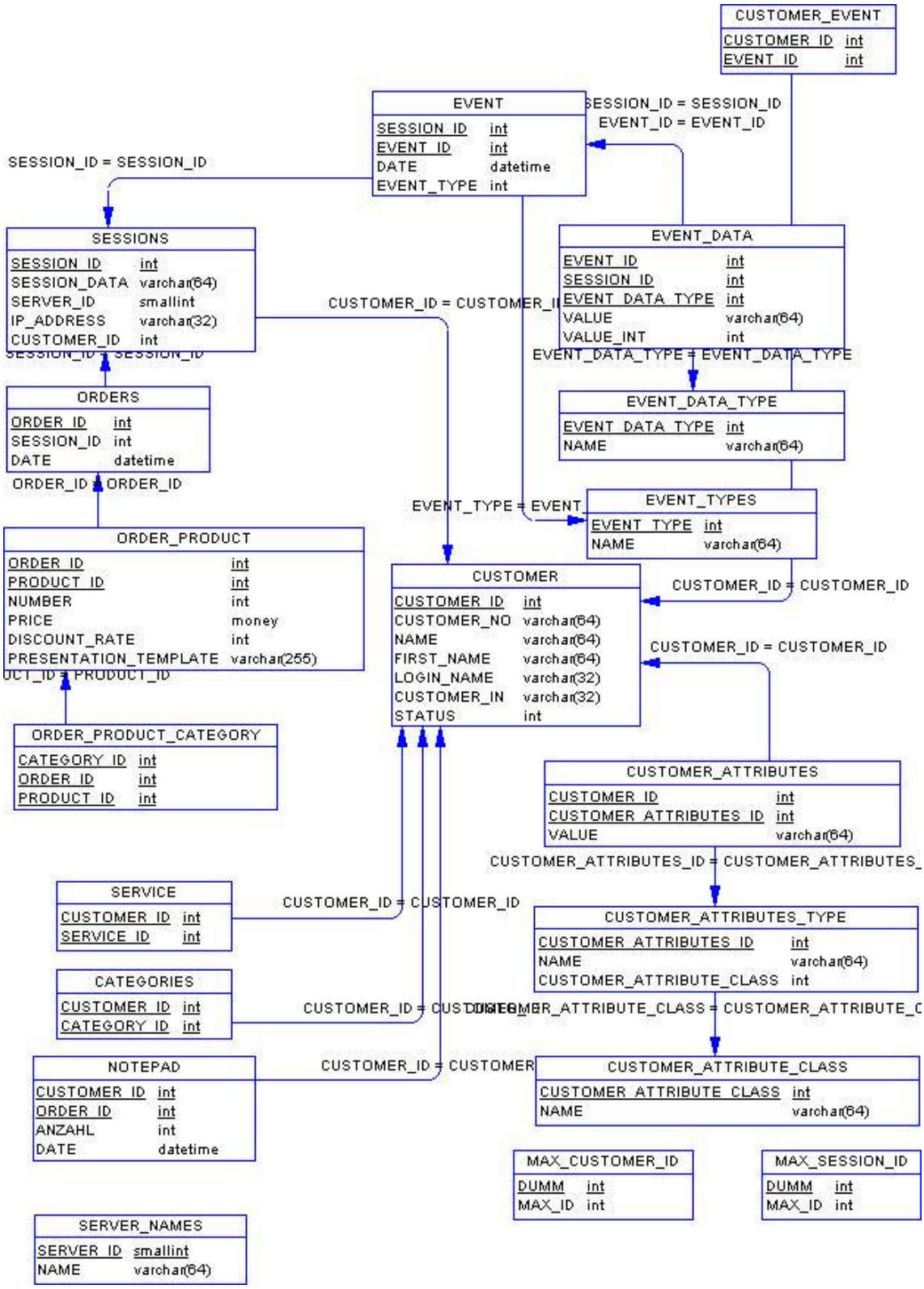
Datenmodell für die internationale Datenbank



Datenmodell für die nationale Datenbank

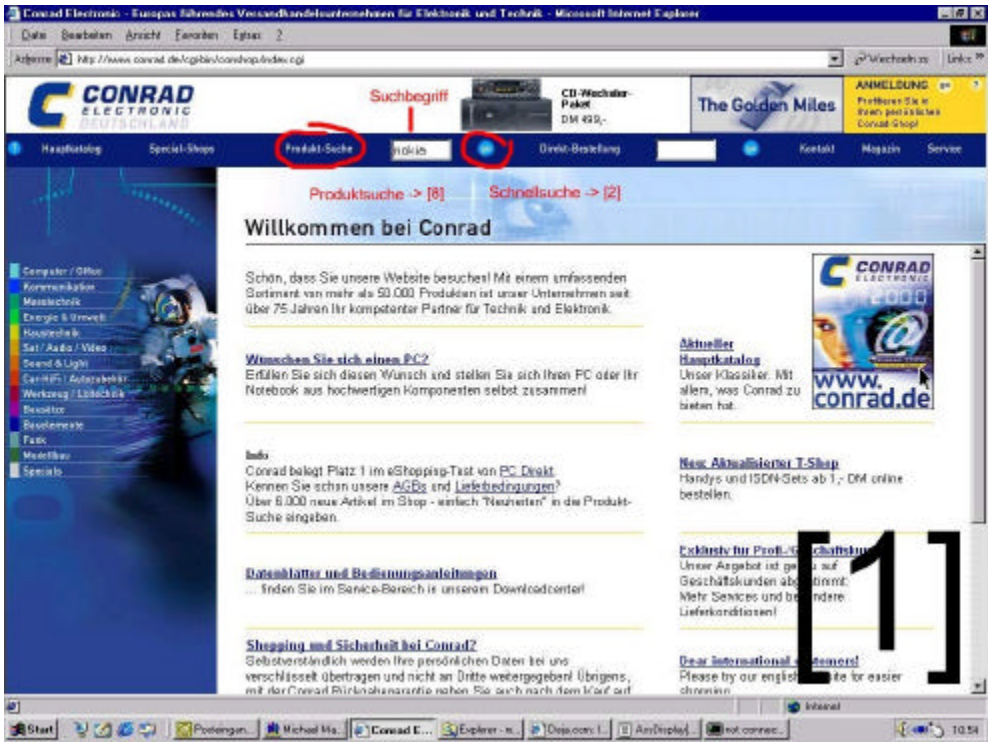


Datenmodell für die Verwaltung der Benutzer

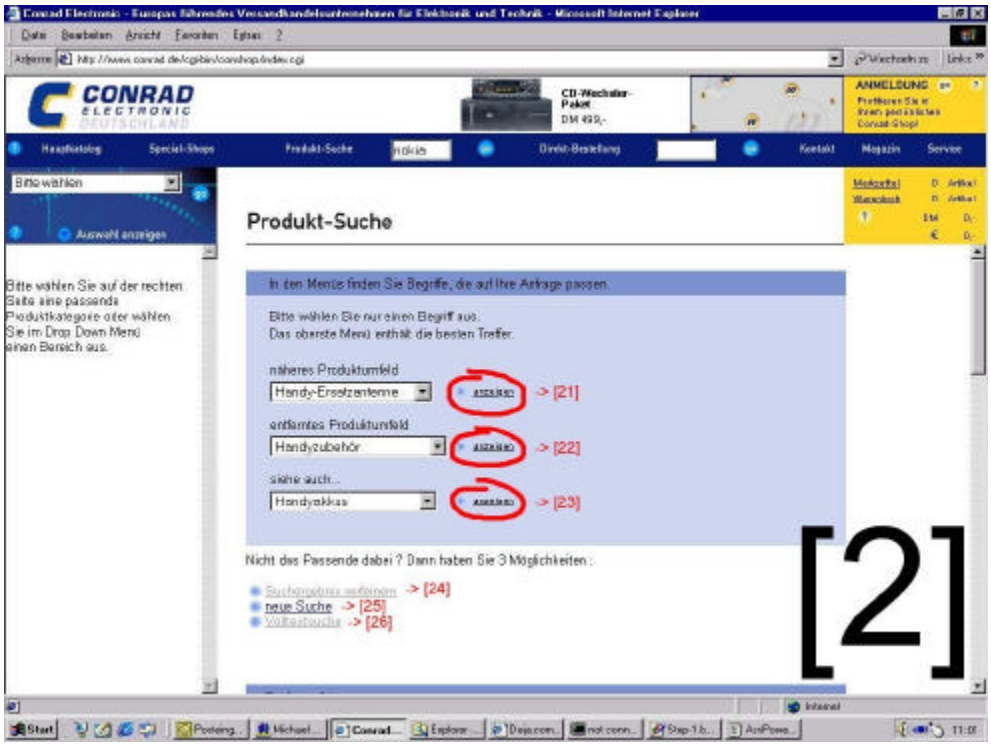


Anhang B – Bildschirmabzüge "Suche" in der aktuellen Shoppingapplikation

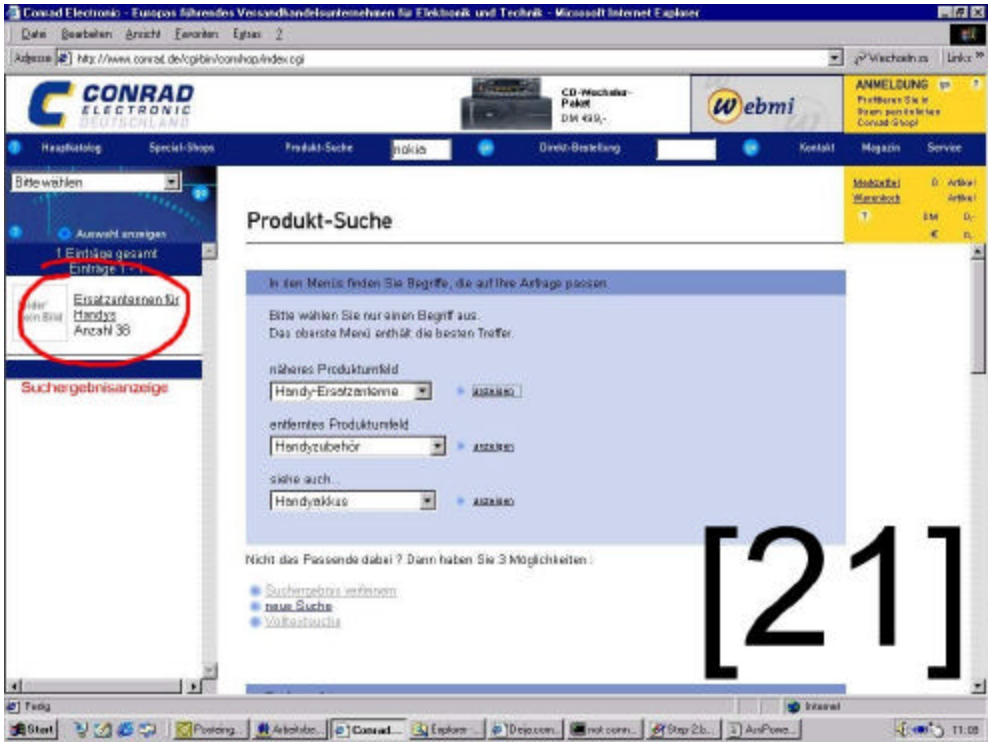
Schritt [1]



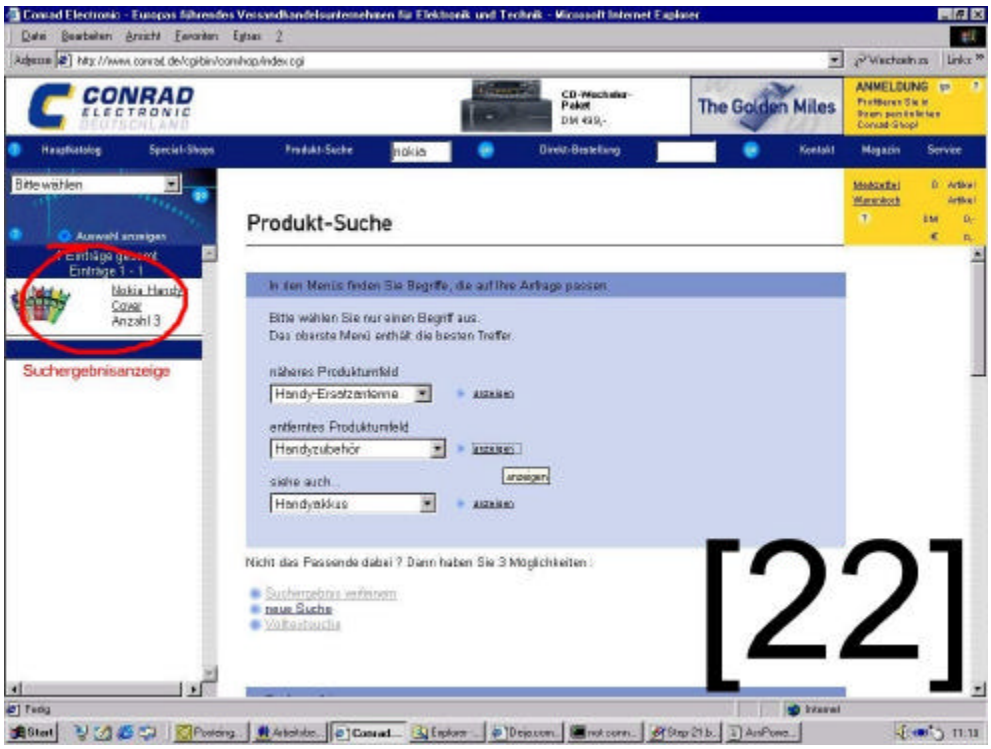
Schritt [2]



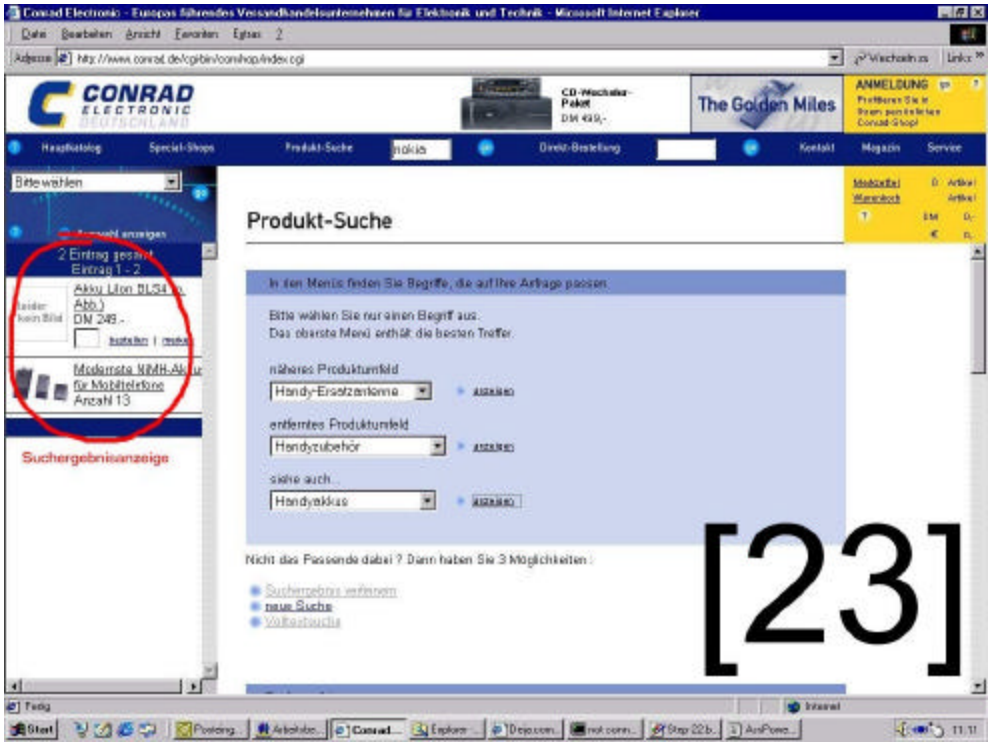
Schritt [21]



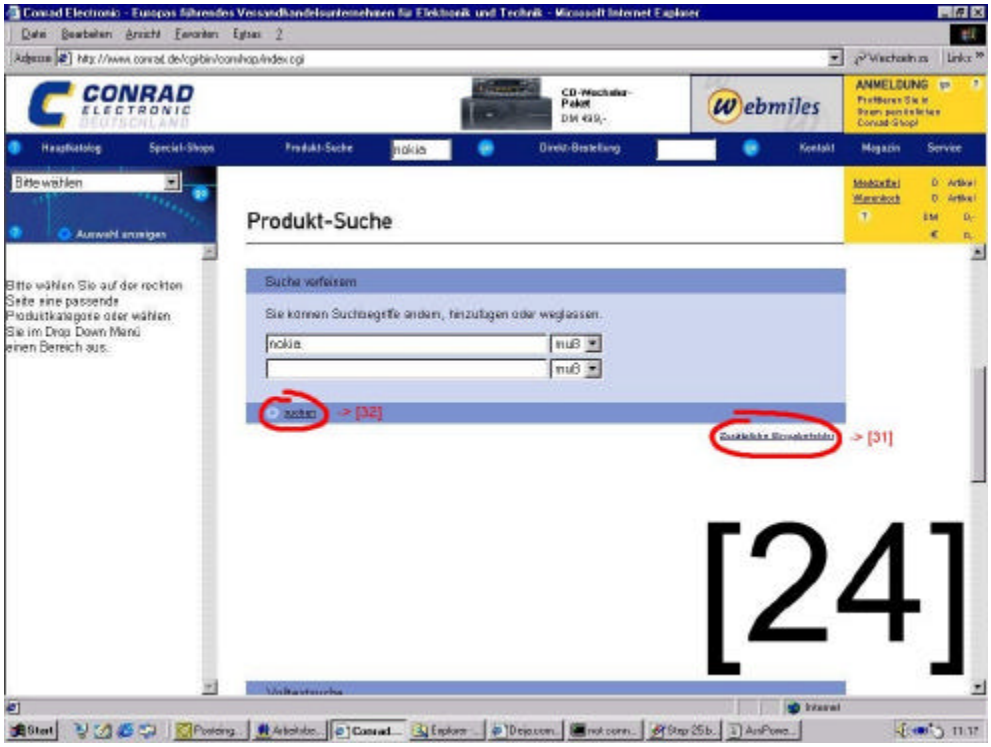
Schritt [22]



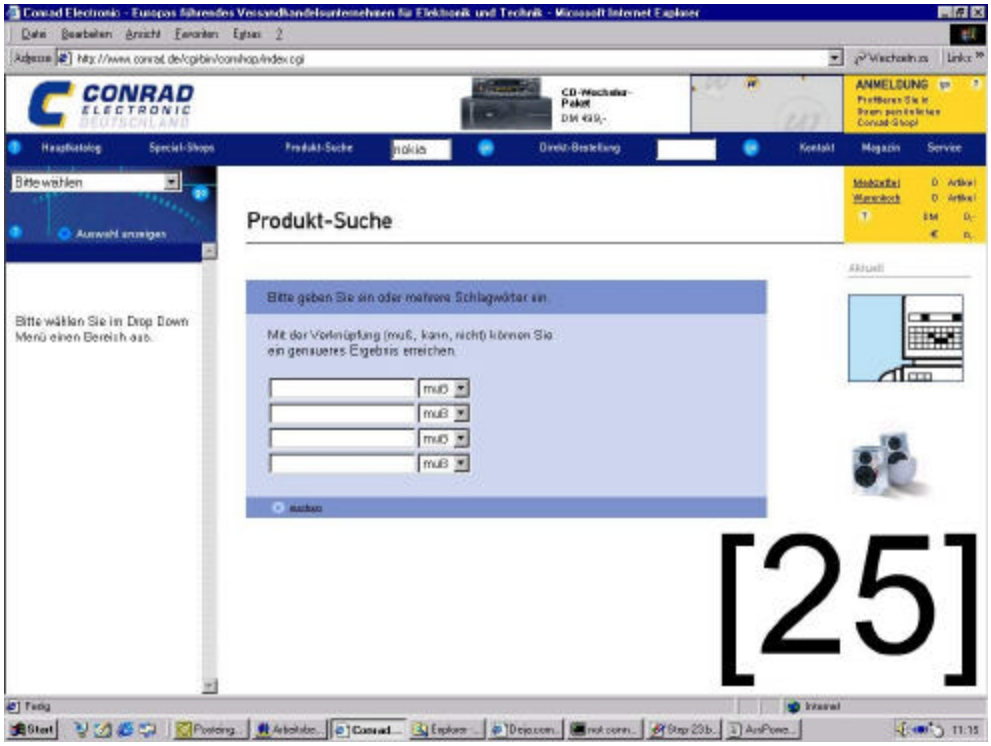
Schritt [23]



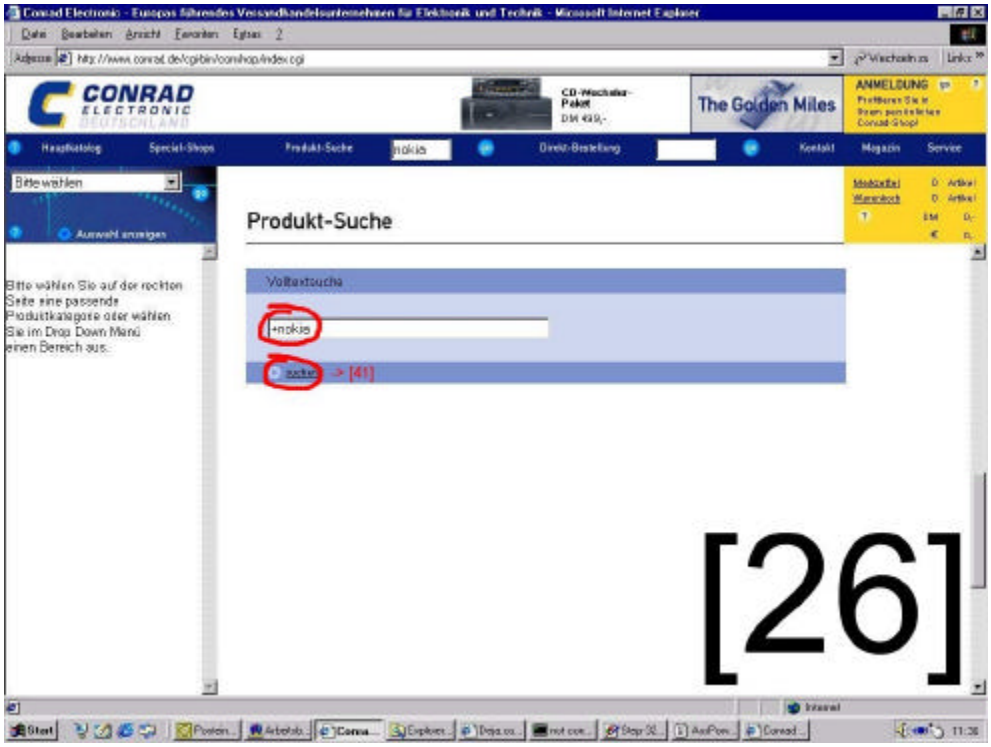
Schritt [24]



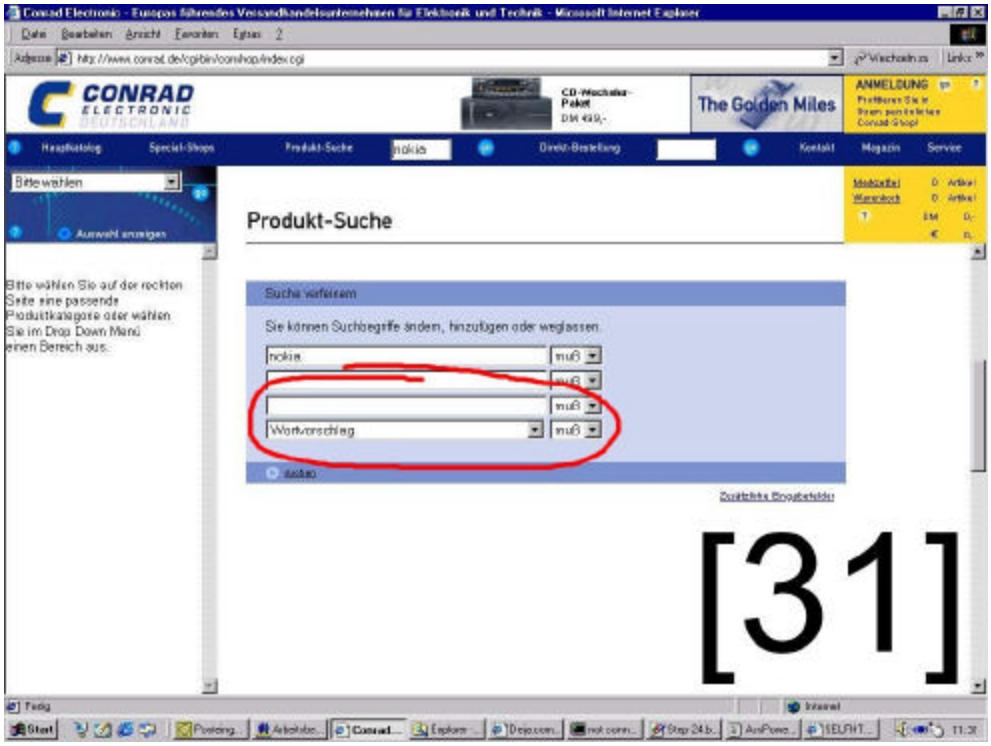
Schritt [25]



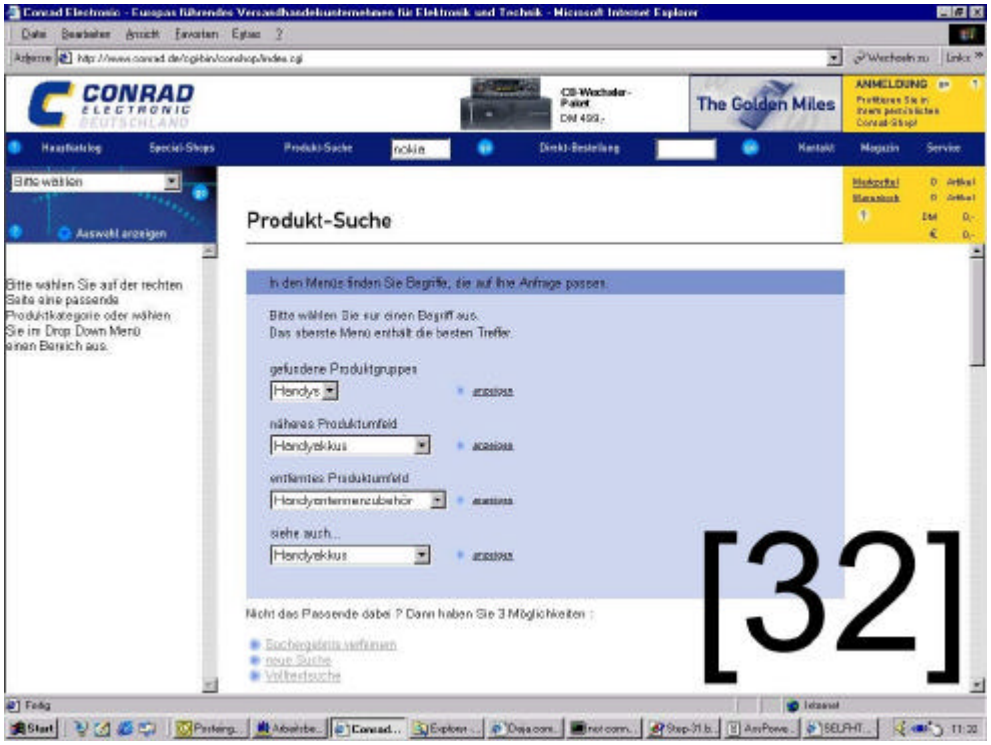
Schritt [26]



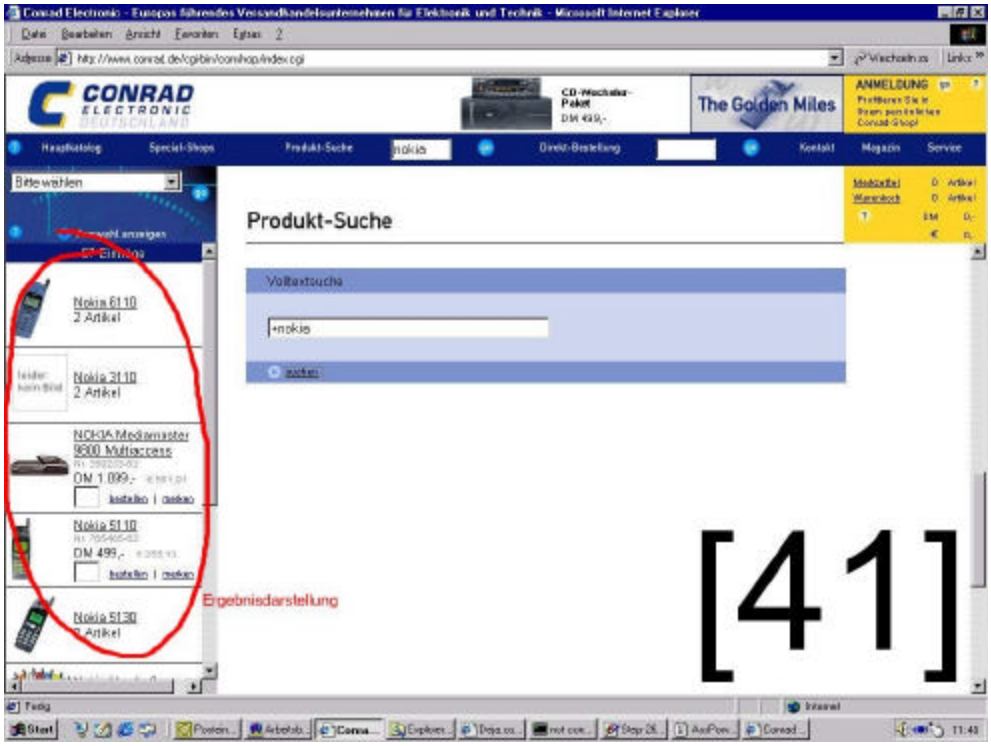
Schritt [31]



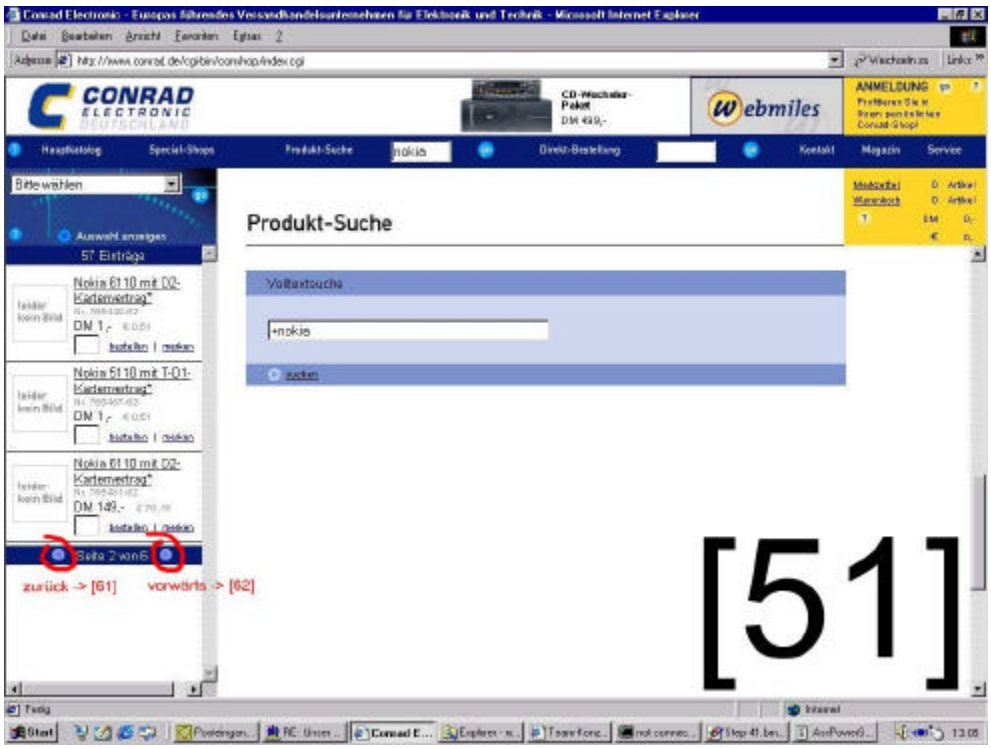
Schritt [32]



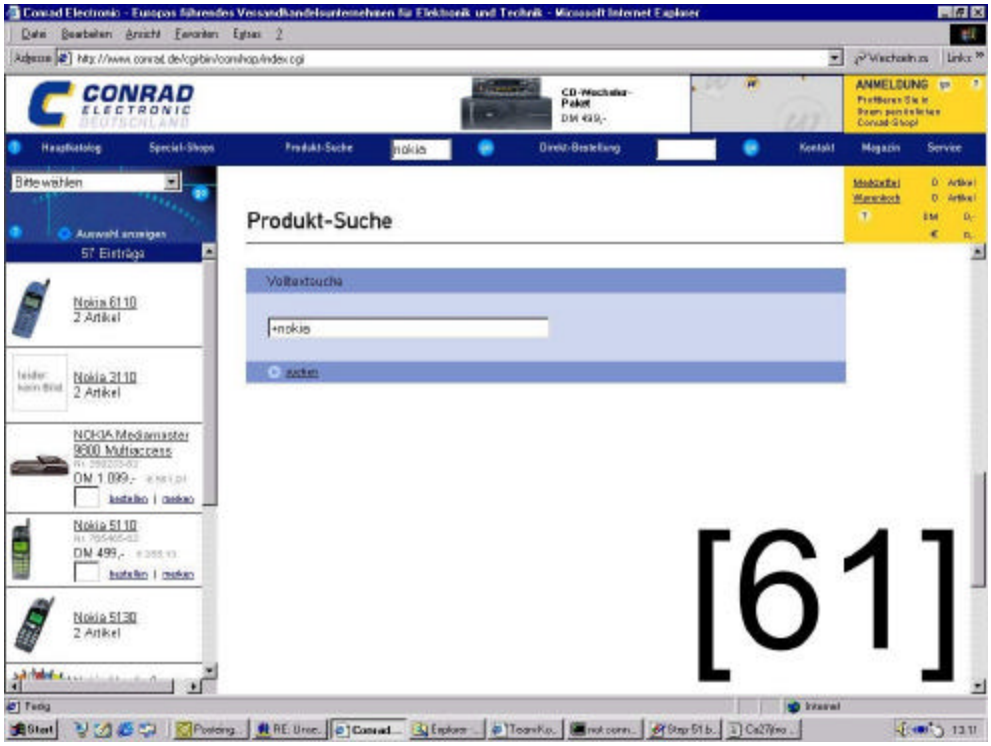
Schritt [41]



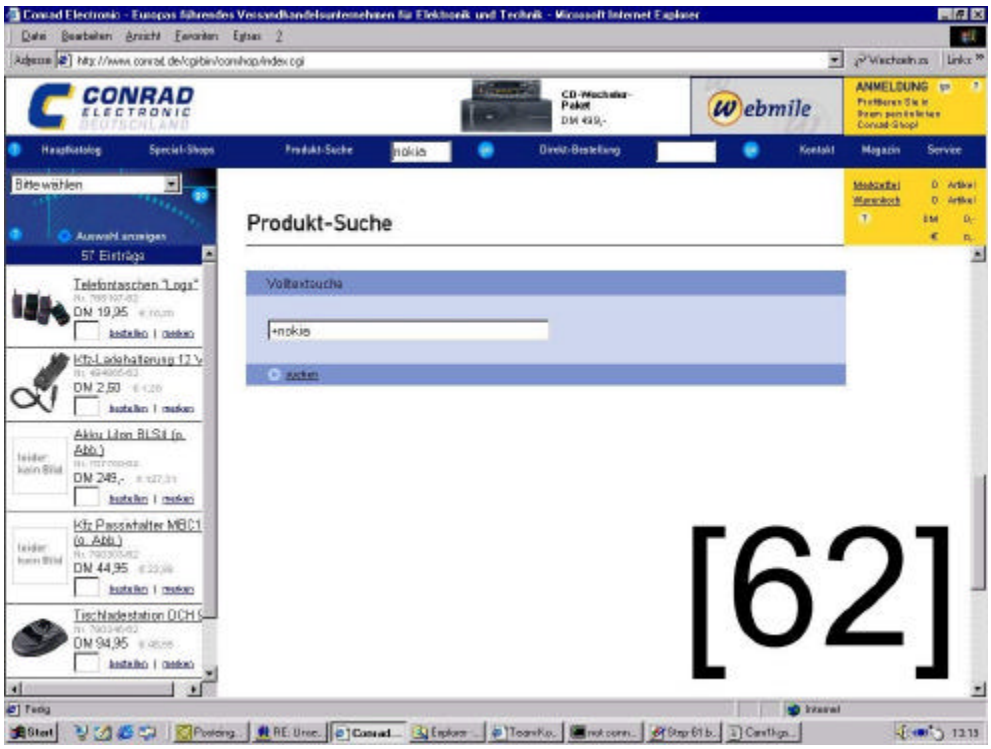
Schritt [51]



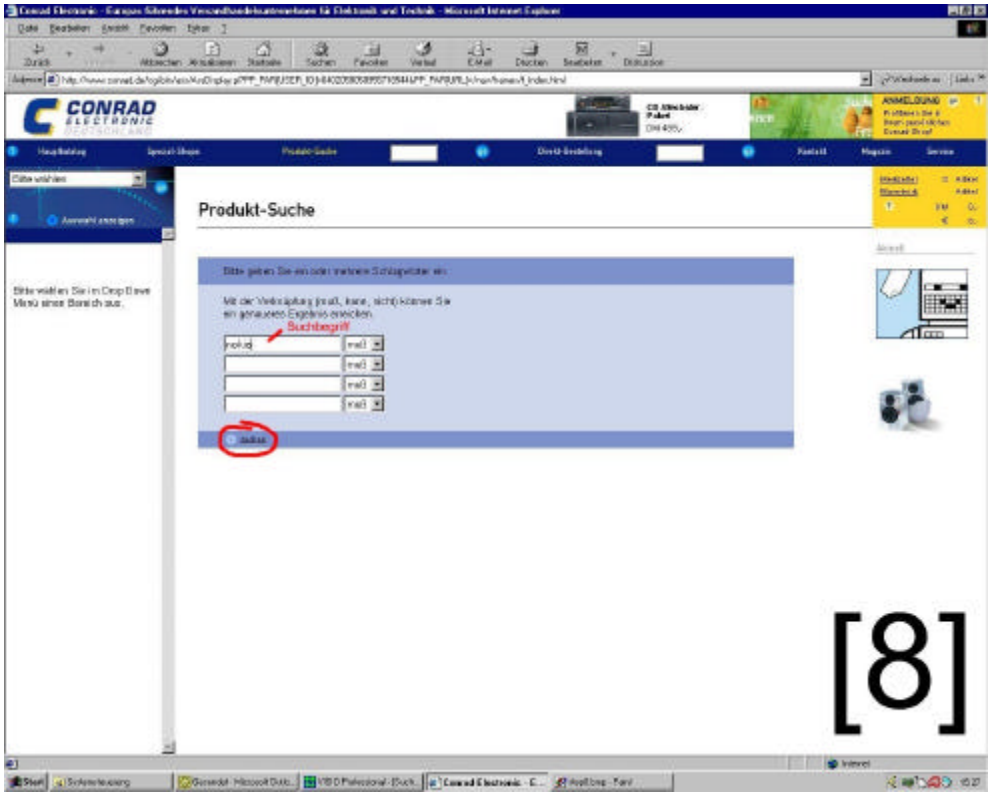
Schritt [61]



Schritt [62]



Schritt [8]



Anhang C – Sourcen zum Thema "Suche" in der aktuellen Shoppingapplikation

sw_indexer.pl

```
#!/opt/local/bin/perl

##############################################################################
# sw_indexer.pl                                                              #
# CONRAD Schlagwortsuche Indexer                                              #
# Author: Martin Bernd Schmeil schmeil@pixelpark.com 1999, 2000               #
#                                                                             #
# Erzeugt aus den im current dirotory befindlichen Schlagwortsuche-Dateien    #
# bcp-files, die dem Schlagwortusche Datenmodell entsprechen und kopiert      #
# diese in die Datenbank.                                                     #
#                                                                             #
# v1.0                                                                        #
# v1.1 SW_SCHLAGWORT Anzeigeflag                                              #
# v1.2 bcp batch size 1000                                                   #
# v1.3 truncate table / rename SPs                                            #
# v1.4 bcp-files receycling / mehr Kommentare                                 #
###############################################################################

##############################################################################
# usage: $0 Server DB conradi-password [bcp-file-receycling]                 #
##############################################################################

die "usage: $0 Server DB conradi-password [bcp-file-receycling]\n"
  if ($#ARGV < 2 or $ARGV > 3);

##############################################################################
# ---------------------------- config ------------------------------------- #
##############################################################################
```

```
# Status: test

$debug = 1;

# DB-config
$sybase = "/opt/sybase/";
$isql   = "$sybase/bin/isql";
$bcp    = "$sybase/bin/bcp";
$server = $ARGV[0];
$DB     = $ARGV[1];
$user   = "conradi";
$pwd    = $ARGV[2];

# filenames
$fgruppe = "gruppe.txt";
$fhomonym = "homonym.txt";
$fprodgr = "produkt_gruppe.txt";
$fprodsw = "produkt_schlagwort.txt";
$fsw = "schlagwort.txt";
$fanzeige = "wort_anzeige.txt";


%swid;                        # flag: swid{Wort-ID}
%swklar;                      # SW_klar{Schlagwort-ID} = Klartext

################################################################################
# bcp-file-receycling: bereits generierte bcp-files nutzen.                    #
################################################################################

if ($#ARGV == 3) {
  # bcp-Daten direkt in DB kopieren
  &DB_stuff;
  exit (0);
}

################################################################################
# WORT_ANZEIGE: Anzuzeigende Schlagworte werden im Hash %anzeige gehalten.     #
################################################################################

%anzeige;                       # flag: Schlagwort-ID, wird angezeigt

open (ANZEIGE, $fanzeige)
  or die "cant open $fanzeige: $!\n";

$line = <ANZEIGE>;            # header-Zeile

while ($line = <ANZEIGE>) {
  chomp $line;
  $line =~ s/\r//g;
  my ($id, $anz) = split /\t/, $line;

  $anzeige{$id} = 1
    if ($anz eq "1");

  #print "\$anzeige{$id} = $anzeige{$id}\n";
}

close ANZEIGE;

################################################################################
# PRODUKT_SCHLAGWORT: Produkt-Schlagwort / Level-Relation wird im Hash         #
# %prodsw gehalten. Der 3-stellige Level wird auf den 1-stelligen Hauptlevel   #
# gekürzrt, existieren zu einer Relation mehrere Levels wird der niedrigste    #
# ausgewählt. Der Inhalt des Hashs wird in das file prodsw.bcp geschrieben.    #
################################################################################

%prodsw;

open (PRODSW, $fprodsw)
  or die "cant open $fprodsw: $!\n";

$line = <PRODSW>;             # header-Zeile

while ($line = <PRODSW>) {
```

```
  chomp $line;
  $line =~ s/\r//g;
  my ($bnr, $id, $level) = split /\t/, $line;
  next if (($bnr eq "") or ($id eq "") or ($level eq ""));
  $level =~ s/(\d)\d\d/$1/;   # keine Sublevel
  if ((!(exists ($prodsw{"$bnr\t$id"})))
|| ($prodsw{"$bnr\t$id"} > $level)) {
    $prodsw{"$bnr\t$id"} = $level; # doppelte eleminieren
  }
  $swid{$id} = 1;            # Schlagwort-ID festhalten
}

close PRODSW;

open (PRODSWOUT, ">prodsw.bcp")
  or die "cant open prodsw.bcp: $!\n";

foreach $key (keys %prodsw) {
  print PRODSWOUT "$key\t$prodsw{$key}\n";
}

close PRODSWOUT;

##############################################################################
# HOMONYM: Homonym (für lookup gewandelt) - Schlagwort-ID - Relation wird im  #
# Hash %homonym, Schlagwort-IDs im Hash %swid gehalten.                       #
##############################################################################

%homonym;                    # Homonym{HOMONYM} = Wort-ID

open (HOMONYM, $fhomonym)
  or die "cant open $fhomonym: $!\n";

$line = <HOMONYM>;           # header-Zeile

while ($line = <HOMONYM>) {
  chomp $line;
  $line =~ s/\r//g;
  my ($sw, $id) = split /\t/, $line;
  next if (($sw eq "") or ($id eq ""));
  $sw =~ s/\s+$//;           # trim
  $homonym{&lookupize($sw)} = $id;
  $swid{$id} = 1;            # Schlagwort-ID festhalten
}

close (HOMONYM);

##############################################################################
# SCHLAGWORT: Homonymliste, d.h. Hash %homonym, um Schlagworte (gewandelt für #
# lookup) ergänzen, Schlagworte (nicht gewandelt!) im Hash %swklar halten.    #
##############################################################################

open (SW, $fsw)
  or die "cant open $fsw: $!\n";

$line = <SW>;                # header-Zeile

while ($line = <SW>) {
  chomp $line;
  $line =~ s/\r//g;
  my ($sw, $id) = split /\t/, $line;
  next if (($sw eq "") or ($id eq ""));
  $sw =~ s/\s+$//;           # trim
  $homonym{&lookupize($sw)} = $id; # Schlagwort-ID
  $swklar{$id} = $sw;             # Schlagwort Klartext
  $swid{$id} = 1;                 # Schlagwort-ID festhalten
}

close (SW);

##############################################################################
# Der Inhalt des Hashs %homonym wird in die Datei homonym.bcp geschrieben.    #
# Dabei werden die Homonyme darstellbarer Schlagworte in Trigramme zerlegt    #
# (Trigramme und deren IDs im Hash %gramm3 gespeichert) und die Relation      #
```

```
# Homonym - Trigramm im Hash %hgramm gehalten.                                #
###############################################################################

open (HOMONYMOUT, ">homonym.bcp")
  or die "cant open homonym.bcp: $!\n";

$i = 0; # homonym-ID
$g3i = 1;

foreach $word (keys %homonym) {
  $i++; # homonym-ID

  print HOMONYMOUT &lookupize ($word) . "\t" # Homonym
    . "$homonym{$word}\t" # Wort_ID
    . "$i\t" # Homonym_ID
    . "0\n";  # Statistik

  # kein Ngramm-Index, wenn Schlagwort nicht anzuzeigen ist
  next if (!(exists ( $anzeige{($homonym{$word})} )));

  $word = " " . $word . " ";
  my $pos = 0;
  my $len = length ($word) - 2;

  while ($pos < $len) {
    $g3 = substr ($word, $pos, 3);

    if (!(exists ($gramm3{$g3}))) {
      $gramm3{$g3} = $g3i;
      $g3i++;
    }

    # homonym-id / gramm-id
    $hgramm3{$i}{$gramm3{$g3}} = 1;

    $pos++;
  }
}

close (HOMONYMOUT);

###############################################################################
# Schlagworte, IDs und das Anzeige-flag wird in das file sw.bcp geschrieben.  #
###############################################################################

open (SWOUT, ">sw.bcp")
  or die "cant open sw.bcp: $!\n";

foreach $id (keys %swid) {
  next if (!(exists($swklar{$id})));

  print SWOUT "$id\t" # Schlagwort_ID
              . $swklar{$id} . "\t" # Schlagwort
              . $anzeige{$id} . "\t" # Anzeige-Flag
              . "0\n"; # Statistik
}

close (SWOUT);

###############################################################################
# PRODUKT_GRUPPE: Bestellnummer - Produktgruppe - Relation wird in das file   #
# prodgr.bcp geschrieben.                                                     #
###############################################################################

open (PRODGR, $fprodgr)
  or die "cant open $prodgr: $!\n";
open (PRODGROUT, ">prodgr.bcp")
  or die "cant open prodgr.bcp: $!\n";

$line = <PRODGR>;            # header-Zeile

while ($line = <PRODGR>) {
  chomp $line;
  $line =~ s/\r//g;
```

```
  my ($bnr, $grid) = split /\t/, $line;
  last if (($bnr eq "") or ($grid eq ""));
  print PRODGROUT "$bnr\t$grid\n";
}

close (PRODGR);
close (PRODGROUT);

##############################################################################
# GRUPPE: Gruppenname / ID wird in das file gruppe.bcp geschrieben.          #
##############################################################################

open (GRUPPE, $fgruppe)
  or die "cant open $fgruppe: $!\n";
open (GRUPPEOUT, ">gruppe.bcp")
  or die "cant open gruppe.bcp: $!\n";

$line = <GRUPPE>;            # header-Zeile

while ($line = <GRUPPE>) {
  chomp $line;
  $line =~ s/\r//g;
  my ($name, $grid) = split /\t/, $line;
  last if (($grid eq "") or ($name eq ""));
  $name =~ s/\s+$//;         # trim
  print GRUPPEOUT "$grid\t$name\n";
}

close (GRUPPE);
close (GRUPPEOUT);

##############################################################################
# "FUZZY"-Daten in die files homonym_gramm3.bcp und gramm3.bcp schreiben.    #
##############################################################################

open (HG3OUT, ">homonym_gramm3.bcp")
  or die "cant open homonym_gramm3.bcp: $!\n";
open (G3OUT, ">gramm3.bcp")
  or die "cant open gramm3.bcp: $!\n";

foreach $g3 (keys %gramm3) {
  print G3OUT "$g3\t$gramm3{$g3}\n";
}

foreach $i (keys %hgramm3) {
  foreach $g3i (keys %{$hgramm3{$i}}) {
    print HG3OUT "$i\t$g3i\n";
  }
}

close (HG3OUT);
close (G3OUT);

##############################################################################
# bcp-Daten in DB kopieren.                                                  #
##############################################################################
&DB_stuff;

##############################################################################
# ---------------------------- Ende ------------------------------------- #
##############################################################################

##############################################################################
# DB_stuff                                                                   #
#                                                                            #
# IN: -                                                                      #
# OUT: -                                                                     #
#                                                                            #
# Datenbank-Aktionen: stored procedures umbenennen, SWS-Tables truncaten,    #
# Inhalt der bcp-files in DB kopieren, stored procedures wiederherstellen.   #
##############################################################################

sub DB_stuff {
$cmd = <<EOT
```

```
use $DB
go
exec sp_rename sp_schlagwortsuche, xx_schlagwortsuche
go
exec sp_rename sp_schlagwortprodukt, xx_schlagwortprodukt
go
truncate table SW_HOMONYM_GRAMM3
go
truncate table SW_GRAMM3
go
truncate table SW_PRODUKT_SCHLAGWORT
go
truncate table SW_MAP
go
truncate table SW_HOMONYM
go
truncate table SW_SCHLAGWORT
go
truncate table SW_PRODUKT_GRUPPE
go
truncate table SW_GRUPPE
go
truncate table SW_SUCHINDEX
go
EOT
  ;

system "$isql -U$user -S$server -P$pwd<<EOT\n$cmd\nEOT" ;

system "bcp $DB..SW_SCHLAGWORT in sw.bcp -U$user -S$server -P$pwd -b1000 -c";
system "bcp $DB..SW_HOMONYM in homonym.bcp -U$user -S$server -P$pwd -b1000 -c";
system "bcp $DB..SW_PRODUKT_SCHLAGWORT in prodsw.bcp -U$user -S$server -P$pwd -b1000 -
c";
system "bcp $DB..SW_PRODUKT_GRUPPE in prodgr.bcp -U$user -S$server -P$pwd -b1000 -c";
system "bcp $DB..SW_GRUPPE in gruppe.bcp -U$user -S$server -P$pwd -b1000 -c";
system "bcp $DB..SW_HOMONYM_GRAMM3 in homonym_gramm3.bcp "
  ."-U$user -S$server -P$pwd -b1000 -c";
system "bcp $DB..SW_GRAMM3 in gramm3.bcp -U$user -S$server -P$pwd -b1000 -c";


$cmd = <<EOT
use $DB
go
exec sp_rename xx_schlagwortsuche, sp_schlagwortsuche
go
exec sp_rename xx_schlagwortprodukt, sp_schlagwortprodukt
go
exec sp_recompile SW_HOMONYM
go
EOT
  ;

system "$isql -U$user -S$server -P$pwd<<EOT\n$cmd\nEOT" ;
}

##############################################################################
# lookupize                                                                  #
#                                                                            #
# IN / OUT: $word: zu wandelndes / gewandeltes Schlagwort                    #
#                                                                            #
# Wandelt string für lookup (uppercase, Umlaute ersetzt)                     #
##############################################################################

sub lookupize {
  my ($word) = @_;

  $word =~ s/(Ä|ä)/AE/g;
  $word =~ s/(Ö|ö)/OE/g;
  $word =~ s/(Ü|ü)/UE/g;
  $word =~ s/ß/SS/g;

  $word =~ tr/a-z/A-Z/;

  return ($word);
```

```
}
```

sp_schlagwortsuche

```
create proc sp_schlagwortsuche (@GRUPPE_ID int = null) as
begin
	/* Schlagwortsuche */
	/* liesst aus vom aufrufenden Prozess erzeugter #SCHLAGWORT */
	/* schreibt nach #SCHLAGWORT_RESULT                         */
	/* filtert optional nach AND-/NOT-Schlagworten / GRUPPE_ID  */

	/* Rueckgabe: LEVEL / ID / NAME / ANZAHL                    */
	/* Level < 1: Vorschlag fuer nicht gefundenes Schlagwort    */
	/*            -n / WORT_ID / HOMONYM / POS                  */
	/* Level = 0: Vorschlag weiterer Suchworte                  */
	/*            0 / WORT_ID / WORT (Klartext) / null     */
	/* Level > 0: Rueckgabe der Gruppen, Produkte pro Level     */
	/*            LEVEL / GRUPPE_ID / GRUPPE_NAME / ANZAHL      */

	/*
	 * table #SCHLAGWORT (
	 *     WORT_ID         int     null,
	 *     SCHLAGWORT      varchar(100)    null,
	 *     BOOL_MOD        char(1) null
	 * )
	 *
	 * table #SCHLAGWORT_RESULT (
	 *     SW_LEVEL        int,
	 *     ID              int,
	 *     NAME            varchar(100) null,
	 *     ANZAHL          int null
	 * )
	  */

	/* MBS 990512: Statistik */
	/* MBS 990514: minor bugs beseitigt */
	/* MBS 990618: "fuzzy"-Suche */
	/* MBS 990705: update WORT_ID in #SCHLAGWORT */
	/* MBS 990822: "fuzzy"-Buffer eingerichtet */
	/* MBS 990915: Statistik integriert */
	/* MBS 990915: Anzeigeflag für "fuzzy"-Suche */
	 /* MBS 991004: @TEXT-Bug beseitigt (varchar -> char) */
	 /* MBS 991109: SEARCHINDEX-Bug beseitigt (Eintrag von existierenenden SW) */

	declare @NO_AND       int /* Anzahl AND-Schlagworte */
	declare @I            int  /* Wortvorschlag Nummer -@I */
	declare @SID       int /* Suchwort-ID */
	declare @LEN   int /* zur Zerlegung in 3Gramme */
	declare @POS   int /* zur Zerlegung in 3Gramme */
	declare @GID    int /* 3Gramm-ID */
	declare @WORT  varchar(100) /* Suchwort */
	declare @TEXT   char(255) /* scratch, zur Zerlegung in 3Gramme */
	declare @MAX_RES int  /* Ideal-Anzahl der Ergebnisse der fuzzy-Suche */
	declare @REQ   int /* Mindestanzahl uebereinstimmender 3Gramme */
	 declare @NEW  int /* neuer Suchindex-Eintrag */
	declare @WID int /* Wort-ID */

	select @MAX_RES = 20

	/* Wort-IDs finden: Homonym-Lookup */
	update #SCHLAGWORT
	set    sw.WORT_ID = h.WORT_ID
	from   #SCHLAGWORT sw, SW_HOMONYM h
	where  sw.WORT_ID is null
	and    sw.SCHLAGWORT = h.HOMONYM

	/* gefundene Produkte */
	create table #PRODUKT_SW (
		BESTELLNUMMER   int,
		WORT_ID         int,
		SW_LEVEL                int
	)
```

```
/* Produkte zu Schlagworten finden */
insert into #PRODUKT_SW
select distinct ps.BESTELLNUMMER, ps.WORT_ID, ps.SW_LEVEL
from   SW_PRODUKT_SCHLAGWORT ps, #SCHLAGWORT sw
where  ps.WORT_ID = sw.WORT_ID
and    exists (
       select *
       from   PRODUKT_ABSATZ
       where  BESTELLNUMMER = ps.BESTELLNUMMER)

select @I = 0
select @WORT = (select min(SCHLAGWORT)
                from #SCHLAGWORT)

/* loop ueber #SCHLAGWORT.WORT */
while (@WORT is not null)
begin
       /* Wort-ID feststellen */
       select @WID = (select WORT_ID
                      from #SCHLAGWORT
                      where SCHLAGWORT = @WORT)

       /* Suchwort-ID feststellen */
       select @SID = (select SUCHWORT_ID
                       from  SW_SUCHINDEX
                       where SUCHWORT = @WORT)

       /* Eintrag in Suchindex */
       if (@SID is null)
       begin
        begin transaction
              select @SID = (select isnull (max(SUCHWORT_ID) + 1, 1)
                                    from   SW_SUCHINDEX)
              insert into SW_SUCHINDEX values (@WORT, @SID, 0)
              select @NEW = 1
         commit transaction
       end
       else
              select @NEW = 0

       /* nicht gefundene Suchworte */
       if (@WID is null)
       begin
              /* Vorschlaege ermitteln */
              if (@NEW = 1)
              begin
                     create table #GRAMM3 (
                            GRAMM_ID int
                     )

                     create table #FOUND (
                            HOMONYM_ID     int,
                            ANZAHL  int,
                            H_LEN          int
                     )

                     create table #RESULT (
                            HOMONYM_ID     int,
                            ANZAHL  int,
                            H_LEN          int,
                            POS            numeric(4) identity
                     )

                     select @LEN = datalength (@WORT)
                     select @TEXT = (select " " + @WORT + " ")
                     select @POS = 1
                     select @I = @I - 1

                     /* Suchbegriff in 3Gramme zerlegen */
                     begin transaction

                     while (@POS <= @LEN)
                     begin
```

```
                                        select @GID = (select GRAMM_ID
                                                       from   SW_GRAMM3
                                                       where  GRAMM = substring(@TEXT,
@POS, 3))

                                        if ((@GID is not null)
                                            and(not (exists (select *
                                                             from   #GRAMM3
                                                             where  GRAMM_ID = @GID))))
                                        begin
                                               insert into #GRAMM3 values (@GID)
                                        end

                                        select @POS = (select @POS + 1)
                                end

                                /* effektive Laenge (unique 3Gramme) und Mindest-
Uebereinstimmung */
                                select @LEN = (select count(*) from #GRAMM3)
                                if (@LEN <= 3)
                                        select @REQ = @LEN
                                else
                                        select @REQ = @LEN + 2 - ROUND (@LEN / 1.5, 0)

                                /* Ergebnis filtern und sortieren */
                                insert into #FOUND
                                select h.HOMONYM_ID, count(*),
                                        H_LEN=(select  count(*)
                                               from   SW_HOMONYM_GRAMM3
                                               where  HOMONYM_ID = h.HOMONYM_ID)
                                from   SW_HOMONYM_GRAMM3 h, #GRAMM3 g
                                where  g.GRAMM_ID = h.GRAMM_ID
                                group by h.HOMONYM_ID
                                having count(*) >= @REQ

                                insert into #RESULT
                                select f.HOMONYM_ID, f.ANZAHL, f.H_LEN
                                from   #FOUND f, SW_HOMONYM h
                                where  f.HOMONYM_ID = h.HOMONYM_ID
                                order  by f.ANZAHL desc, f.H_LEN

                                /* ~MAX_RES beste Treffer */
                                insert into SW_MAP
                                select @SID, HOMONYM_ID, POS
                                from   #RESULT
                                where  POS <= (select max(POS)
                                               from   #RESULT
                                               where  ANZAHL = (select      ANZAHL
                                                                from   #RESULT
                                                                where  POS = @MAX_RES)
                                               and    H_LEN = (select       H_LEN
                                                                from   #RESULT
                                                                where  POS = @MAX_RES))
                                commit transaction

                                drop table #GRAMM3
                                drop table #FOUND
                                drop table #RESULT
                        end /* Eintrag in Suchindex */

                        /* Return: Schlagwort-Vorschlaege */
                        insert into #SCHLAGWORT_RESULT
                        select @I, h.WORT_ID, s.SCHLAGWORT, m.POS
                        from   SW_HOMONYM h, SW_MAP m, SW_SCHLAGWORT s
                        where  m.SUCHWORT_ID = @SID
                        and    m.HOMONYM_ID = h.HOMONYM_ID
                        and    h.WORT_ID = s.WORT_ID

                        /* update #SCHLAGWORT: Vorschlag kennzeichnen */
                        update #SCHLAGWORT
                        set    WORT_ID = @I
                        where  SCHLAGWORT = @WORT
                end /* loop ueber nicht gefundene Schlagworte */
```

```
                /* fetch next */
                select @WORT = (select min(SCHLAGWORT)
                                from   #SCHLAGWORT
                                where  @WORT < SCHLAGWORT)
        end /* loop ueber #SCHLAGWORT.WORT */

        /* Statistik (gefunden) */
        update SW_HOMONYM
        set    STATISTIK = STATISTIK + 1
        where  HOMONYM
               in      (
                       select SCHLAGWORT
                       from    #SCHLAGWORT
                       )

        /* Statistik (nicht gefunden) */
        update SW_SUCHINDEX
        set    STATISTIK = STATISTIK + 1
        where  SUCHWORT
               in      (
                       select SCHLAGWORT
                       from    #SCHLAGWORT
                       )

        /* Gruppen-Filter */
        if (@GRUPPE_ID is not null)
        begin
               delete #PRODUKT_SW
               where  BESTELLNUMMER
               not in (
                       select BESTELLNUMMER
                       from   SW_PRODUKT_GRUPPE
                       where  GRUPPE_ID = @GRUPPE_ID
                       )
        end

        /* AND-Filter */
        select @NO_AND = (select count(*) from #SCHLAGWORT
                          where  BOOL_MOD = "+")
        if (@NO_AND > 0)
        begin
               create table #AND (
                       BESTELLNUMMER  int,
                       ANZAHL         int
               )

               insert into #AND
               select ps.BESTELLNUMMER, count(distinct(sw.WORT_ID))
               from   #SCHLAGWORT sw, #PRODUKT_SW ps
               where  sw.BOOL_MOD = "+"
               and    sw.WORT_ID = ps.WORT_ID
               group  by ps.BESTELLNUMMER

               delete #PRODUKT_SW
               where  BESTELLNUMMER
               in     (
                       select BESTELLNUMMER
                       from   #AND
                       where  ANZAHL < @NO_AND
                       )

               drop table #AND
        end

        /* NOT-Filter */
        delete #PRODUKT_SW
        where  BESTELLNUMMER
        in     (
               select ps.BESTELLNUMMER
               from   #SCHLAGWORT sw, #PRODUKT_SW ps
               where  sw.BOOL_MOD = "-"
               and    sw.WORT_ID = ps.WORT_ID
               )
```

```
	/* Return: andere Schlagworte zu den Produkten */
	insert into #SCHLAGWORT_RESULT
	select distinct 0, s.WORT_ID, s.SCHLAGWORT, null
	from   #PRODUKT_SW tps, SW_PRODUKT_SCHLAGWORT ps,
	       SW_SCHLAGWORT s
	where  tps.BESTELLNUMMER = ps.BESTELLNUMMER
	and    ps.WORT_ID = s.WORT_ID
	and    s.WORT_ID
	not in (
	       select WORT_ID
	       from   #SCHLAGWORT
	       )

	/* Return: Gruppen der Produkte */
	insert into #SCHLAGWORT_RESULT
	select distinct tps.SW_LEVEL, g.GRUPPE_ID,
	       g.GRUPPE_NAME, count(distinct(pg.BESTELLNUMMER))
	from   #PRODUKT_SW tps, SW_PRODUKT_GRUPPE pg, SW_GRUPPE g
	where  tps.BESTELLNUMMER = pg.BESTELLNUMMER
	and    pg.GRUPPE_ID = g.GRUPPE_ID
	group by g.GRUPPE_ID, tps.SW_LEVEL

	drop table #PRODUKT_SW
end
```

sp_schlagwortprodukt

```
create proc sp_schlagwortprodukt (@SW_LEVEL int = null, @GRUPPE_ID int = null) as
begin
	/* Schlagwortsuche Produkt */
	/* liesst aus vom aufrufenden Prozess erzeugter #SCHLAGWORTID  */
	/* schreibt nach #SCHLAGWORT_PRODUKT                          */
	/* filtert optional nach AND-/NOT-SWs und SW_LEVEL / GRUPPE_ID */
	/* Rueckgabe: BESTELLNUMMER                                   */

	/* MBS 990512: Statistik */
	/* MBS 990915: Statistik integriert */

	declare @NO_AND       int /* Anzahl AND-Schlagwörter */

	/* update Statistik */
	update  SW_SCHLAGWORT
	set     STATISTIK = STATISTIK + 1
	where   WORT_ID
	        in     (
	               select WORT_ID
	              from   #SCHLAGWORTID
	               )

	/* gefundene Produkte */
	create table #PRODUKT_SW (
	       BESTELLNUMMER   int,
	       WORT_ID         int,
	       SW_LEVEL                int
	)

	/* Produkte zu Schlagwörtern finden */
	insert into #PRODUKT_SW
	select distinct ps.BESTELLNUMMER, ps.WORT_ID, ps.SW_LEVEL
	from   SW_PRODUKT_SCHLAGWORT ps, #SCHLAGWORTID sw
	where  ps.WORT_ID = sw.WORT_ID

	/* Gruppen-Filter */
	if (@GRUPPE_ID is not null)
	begin
	       delete #PRODUKT_SW
	       where  BESTELLNUMMER
	       not in (
	              select BESTELLNUMMER
```

```
                        from   SW_PRODUKT_GRUPPE
                        where  GRUPPE_ID = @GRUPPE_ID
                        )
        end

        /* AND-Filter */
        select @NO_AND = (select count(*) from #SCHLAGWORTID
                          where  BOOL_MOD = "+")
        if (@NO_AND > 0)
         begin
                create table #AND (
                        BESTELLNUMMER  int,
                        ANZAHL         int
                )

                insert into #AND
                select ps.BESTELLNUMMER, count(distinct(sw.WORT_ID))
                from   #SCHLAGWORTID sw, #PRODUKT_SW ps
                where  sw.BOOL_MOD = "+"
                and    sw.WORT_ID = ps.WORT_ID
                group  by ps.BESTELLNUMMER

                delete #PRODUKT_SW
                where  BESTELLNUMMER
                in     (
                        select BESTELLNUMMER
                        from   #AND
                        where  ANZAHL < @NO_AND
                        )

                drop table #AND
        end

        /* NOT-Filter */
        delete #PRODUKT_SW
        where  BESTELLNUMMER
        in     (
                select ps.BESTELLNUMMER
                from   #SCHLAGWORTID sw, #PRODUKT_SW ps
                where  sw.BOOL_MOD = "-"
                and    sw.WORT_ID = ps.WORT_ID
                )

        /* Kontext-Level-Filter */
        if (@SW_LEVEL is not null)
        begin
                delete #PRODUKT_SW
                where  SW_LEVEL <> @SW_LEVEL
        end

        /* Return: Produkte */
        insert into #SCHLAGWORT_PRODUKT
        select distinct BESTELLNUMMER
        from   #PRODUKT_SW
        order by BESTELLNUMMER

        drop table #PRODUKT_SW
end
```

sp_search

```
create proc sp_search
           (@buffer    varchar(255),
            @kat_id    int   = null,
            @min_preis money = null,
            @max_preis money = null)
as

/* v1.0: unterstützt + - . (AND, NOT, EXACT, default: OR)         */
/* v1.1: optionale Filter! - Schnittstelle geändert:              */
/*                           (Kategorie-ID, min-Preis, max-Preis) */
/* v1.?: Performance optimiert?                                    */
```

```
/* v1.4: Logik korrigiert                                          */
/* v1.5: explizites Löschen der temp-tables                        */
/*       index auf #ABSATZ                                         */
/* v1.6: neue Statistik                                            */
/*       nonclustered index auf #ABSATZ                            */
/* v1.7: Spezialbehandlung kurzer Suchbegriffe                     */
/* v1.8: neues DM: integrierte Statistik                           */

declare @word     varchar(100)
declare @pos      int
declare @buflen   int
declare @words    int
declare @c        char
declare @logik    int /* AND-Bedingung = 1, NOT = -1, OR = 0 */
declare @match    int /* exakter match = 1, Teilmatch = 0 */

declare @wort_id  int
declare @swort_id int

/*** config ***/

/* Suchwörter der Länge kurz werden nur noch exakt ge-map-t */
declare @kurz  int
select @kurz = 3

create table #LOGIK (
       SUCHWORT_ID int,
       LOGIK       int,
       MATCH       int
)

create table #ABSATZ (
       SUCHWORT_ID int,
       ABSATZ_ID   int,
       GEWICHTUNG  int,
       MATCH       int
)

create nonclustered index TMPABSATZ
       on #ABSATZ (ABSATZ_ID, SUCHWORT_ID)

create table #AND (
       ABSATZ_ID   int,
       ANZAHL      int
)

create table #RESULT (
       ABSATZ_ID   int,
       MATCH       int,
       GEWICHTUNG  int
)

create table #BESTELLNUMMER (
       BESTELLNUMMER int
)

create table #VBESTELLNUMMER (
       BESTELLNUMMER int
)

/* init */
select @buflen = datalength (@buffer)
select @words = 0

/* in Worte zerlegen */
while (@buflen > 0)
begin
    select @pos = charindex (" ", @buffer)
    if (@pos = 0 and @buflen > 1) /* letztes Wort, aber kein blank */
        select @pos = @buflen + 1

    if (@pos > 1)
    begin
        select @word = substring (@buffer, 1, @pos - 1)
```

```
        if (@buflen > @pos)
            select @buffer = substring (@buffer, @pos + 1,
                                        @buflen - @pos)
        else
            select @buffer = ""

               /* logik lesen */
               select @c = substring (@word, 1, 1)
               if (@c = "+")
                      select @logik = 1
               else if (@c = "-")
                      select @logik = -1
        else
                      select @logik = 0

               /* +/- weg */
               if (@logik <> 0)
                      select @word = substring (@word, 2, datalength (@word) - 1)

               /* exact match lesen */
               select @c = substring (@word, datalength (@word), 1)

               if (@c = ".")
                      begin
                             select @match = 1
                             select @word = substring (@word, 1, datalength (@word) - 1)
                      end
               else
                      select @match = 0

               /* lookup in VTS_SUCHINDEX */
               select @swort_id = (select SUCHWORT_ID
                                                  from VTS_SUCHINDEX
                                                 where SUCHWORT = @word)

               begin transaction
                      /* existiert nicht: Eintrag */
                      if (@swort_id is null)
                      begin
                             select @swort_id = (select max(SUCHWORT_ID) + 1
                                                           from VTS_SUCHINDEX)

                             if (@swort_id is null)
                                select @swort_id = 1

                             insert into VTS_SUCHINDEX
                             values (@word, @swort_id, 0)

                             select @wort_id = (select WORT_ID
                                                           from VTS_INDEX
                                                          where WORT = @word)

                             if (@wort_id is null)
                                    select @wort_id = 0 /* existiert nicht! */
                             else
                                    insert into VTS_MAP /* exakter match */
                                    values (@swort_id, @wort_id, 1)

                             /* Teilmatch: nur bei Mindestlaenge eintragen! */
                             if (datalength (@word) > @kurz)
                             begin
                                    insert into VTS_MAP
                                    select distinct @swort_id, a.WORT_ID, 0
                                    from VTS_INDEX i, VTS_ABSATZ a
                                    where i.WORT like "%" + @word + "%"
                                    and i.WORT_ID = a.WORT_ID
                                    and a.WORT_ID <> @wort_id
                             end
                      end
               commit transaction

               /* Statistik */
               update VTS_SUCHINDEX
               set    STATISTIK = STATISTIK + 1
```

```
               where  SUCHWORT_ID = @swort_id

               /* Eintrag in Logik-Tabelle */
               insert into #LOGIK
                   values (@swort_id, @logik, @match)

               /* Eintrag in Absatz-Liste */
               if (@match = 1) /* exakter match */
                      insert into #ABSATZ
                      select @swort_id, a.ABSATZ_ID, a.GEWICHTUNG, 1
                        from VTS_ABSATZ a, VTS_MAP m
                       where m.SUCHWORT_ID = @swort_id
                         and m.MATCH = 1 /* exakt */
                         and m.WORT_ID = a.WORT_ID
               else /* Teilmatch */
                      insert into #ABSATZ
                      select @swort_id, a.ABSATZ_ID, a.GEWICHTUNG, m.MATCH
                        from VTS_ABSATZ a, VTS_MAP m
                       where m.SUCHWORT_ID = @swort_id
                         and m.WORT_ID = a.WORT_ID
               end
    else /* Leerzeichen */
    begin
        select @buffer = substring (@buffer, 2, @buflen - 1)
    end
        select @buflen = datalength (@buffer)
end


/* Filter: AND-Bedingung */
if (exists (select *
             from #LOGIK
             where LOGIK = 1)) /* AND */
begin
       insert into #AND
       select ABSATZ_ID, count(distinct (SUCHWORT_ID))
      from #ABSATZ
     where SUCHWORT_ID
        in (select SUCHWORT_ID
              from #LOGIK
             where LOGIK = 1)
     group by ABSATZ_ID

       /* alle Absätze, die mindestens ein AND-Wort nicht enthalten */
       delete #ABSATZ
     where ABSATZ_ID
    not in (select ABSATZ_ID
              from #AND
             where ANZAHL =
                   (select count(SUCHWORT_ID)
                      from #LOGIK
                     where LOGIK = 1))
end

drop table #AND

/* Filter: NOT-Bedingung */
if (exists (select *
             from #LOGIK
             where LOGIK = -1)) /* NOT */
begin
       /* alle Absätze, die mindestens ein NOT-Wort enthalten */
       delete #ABSATZ
         where ABSATZ_ID
        in (select ABSATZ_ID
              from #ABSATZ
             where SUCHWORT_ID
                in (select SUCHWORT_ID
                      from #LOGIK
                     where LOGIK = -1))
end

drop table #LOGIK
```

```
/* Ergebnisliste */
insert into #RESULT
select ABSATZ_ID, 99 + max(MATCH), sum(GEWICHTUNG)
  from #ABSATZ
 group by SUCHWORT_ID, ABSATZ_ID

drop table #ABSATZ

/* Einschränkung für Filter (Performance) */
if (@kat_id is not null
    or @min_preis is not null
    or @max_preis is not null)
begin
       insert into #VBESTELLNUMMER
       select BESTELLNUMMER
      from TEXT_ABSATZ
     where ABSATZ_ID
        in (select ABSATZ_ID
              from #RESULT)

       insert into #VBESTELLNUMMER
    select tp.BESTELLNUMMER
      from TABELLEN_PRODUKTE tp, TABELLEN_ABSATZ ta
     where ta.ABSATZ_ID
        in (select ABSATZ_ID
              from #RESULT)
       and ta.TABELLEN_ID = tp.TABELLEN_ID
end


/* Filter: KATEGORIE_ID */
if (@kat_id is not null)
begin
       /* Bestellnummern aus Kategorie(sub)baum */
       insert into #BESTELLNUMMER
    select BESTELLNUMMER
      from PRODUKTKATEGORIE
     where BESTELLNUMMER
        in (select BESTELLNUMMER
              from #VBESTELLNUMMER)
       and KATEGORIE_ID
        in (select KATEGORIE_ID
              from KAT_NETZ
             where LEFT_NBR >= (select LEFT_NBR
                                  from KAT_NETZ
                                 where KATEGORIE_ID = @kat_id)
               and RIGHT_NBR <= (select RIGHT_NBR
                                                                  from KAT_NETZ
                                                                 where KATEGORIE_ID =
@kat_id))

       /* Einzelprodukte */
       delete #RESULT
      from #RESULT r, TEXT_ABSATZ ta
        where r.ABSATZ_ID = ta.ABSATZ_ID
          and ta.BESTELLNUMMER
       not in (select BESTELLNUMMER
                         from #BESTELLNUMMER)

       /* Tabellenprodukte */
       delete #RESULT
         from #RESULT r, TABELLEN_ABSATZ ta
     where r.ABSATZ_ID = ta.ABSATZ_ID
       and ta.TABELLEN_ID
       not in (select TABELLEN_ID
                               from TABELLEN_PRODUKTE tp, #BESTELLNUMMER b
                              where tp.BESTELLNUMMER = b.BESTELLNUMMER)

       delete #BESTELLNUMMER
end

/* Filter: PREIS */
if ((@min_preis is not null) or (@max_preis is not null))
begin
```

```
       /* Extremwerte initialisieren */
       if (@min_preis is null)
       begin
              select @min_preis = 0.0
       end
       if (@max_preis is null)
       begin
              select @max_preis = (select max(PREIS)
                                   from PRODUKT_STAFFEL)
       end

       /* Bestellnummern */
       insert into #BESTELLNUMMER
    select BESTELLNUMMER
      from PRODUKT_STAFFEL
     where BESTELLNUMMER
        in (select BESTELLNUMMER
              from #VBESTELLNUMMER)
       and POSITION = 1
       and PREIS between @min_preis and @max_preis

       /* Einzelprodukte */
       delete #RESULT
      from #RESULT r, TEXT_ABSATZ ta
        where r.ABSATZ_ID = ta.ABSATZ_ID
          and ta.BESTELLNUMMER
       not in (select BESTELLNUMMER
                         from #BESTELLNUMMER)

       /* Tabellenprodukte */
       delete #RESULT
         from #RESULT r, TABELLEN_ABSATZ ta
     where r.ABSATZ_ID = ta.ABSATZ_ID
       and ta.TABELLEN_ID
       not in (select TABELLEN_ID
                               from TABELLEN_PRODUKTE tp, #BESTELLNUMMER b
                              where tp.BESTELLNUMMER = b.BESTELLNUMMER)
end

drop table #VBESTELLNUMMER
drop table #BESTELLNUMMER

/* Ergebnis */
select ABSATZ_ID, "MATCH" = sum(MATCH), "GEWICHTUNG" = sum(GEWICHTUNG)
  into #RESULT2
  from #RESULT
 group by ABSATZ_ID
 order by MATCH desc, GEWICHTUNG desc

drop table #RESULT

/* in #SEARCHTEMP kopieren */
insert into #SEARCHTEMP
select ABSATZ_ID from #RESULT2

drop table #RESULT2
```

sp_text_browse

```
create procedure sp_text_browse
 (@TEXT varchar(255), @START int, @COUNT int, @CATID int, @PRICEMIN money, @PRICEMAX
money, @debug int = null)
as
       create table #SEARCHTEMP
       (
           POS               numeric(5,0)          identity,
           ABSATZ_ID  int            null,
               constraint SEARCHTEMP primary key (POS)
       )

       execute sp_search @TEXT, @CATID, @PRICEMIN, @PRICEMAX, @debug
```

```
        declare @TOTAL int

        select @TOTAL = count(*) from #SEARCHTEMP

        select distinct
                SA.SEITEN_ID,
                AI.ABSATZ_ID,
                AI.ABSATZ_TITEL,
                AI.THUMBNAIL,
                AI.PROD_ANZ,
                AI.BESTELLNUMMER,
                AI.PREIS,
                @TOTAL
        from
                ABSATZ_INFO AI,
                ABSATZ_SEITEN SA,
                #SEARCHTEMP T
        where
                AI.ABSATZ_ID = T.ABSATZ_ID
        and
                SA.ABSATZ_ID = T.ABSATZ_ID
        and
                T.POS between @START and (@START+@COUNT-1)
        and
                T.POS <= @TOTAL
        order by
                T.POS
```

AssPowersuche.pl

```
#!/usr/bin/perl

# THIS VERSION OF "AssPowerSuche.pl" is not the original one extracted
# on 25.04.2000 by Michael Maretzke
# This one is a edited one

use lib "$ENV{'DOCUMENT_ROOT'}/cgi-bin/conshop";
use strict;           # modperl-konforme Programmierung forcieren
use CGI;              # Parameteruebergabe
use TKSqlDB;          # Team-Konzept Modul
use Apache;
use IniConf;


use def; ## hole_shopname

### Serverkonfiguration aus INI-Datei holen
######################################################################

my $ini                =       IniConf->new( -file => "$ENV{'DOCUMENT_ROOT'}/cgi-
bin/ini/AssAddOn.ini");

my $server            =       $ini->val("DB_LOGIN","DB_SERVER1");
my $login             =       $ini->val("DB_LOGIN","DB_LOGIN1");
my $passwd            =       $ini->val("DB_LOGIN","DB_PASSWD1");
my $dbname            =       $ini->val("DB_LOGIN","DB_NAME1");

my $docroot           =       $ENV{'DOCUMENT_ROOT'};

my $SERVER_NAME =       $ENV{'SERVER_NAME'};
my $SHOP               =       def->hole_shopname();

my $locallogin         =       $ini->val("LOCAL_DB_LOGIN","DB_LOGIN_$SHOP");
my $localpasswd        =       $passwd;
my $localdbname        =       $ini->val("LOCAL_DB_LOGIN","DB_NAME_$SHOP");
my $medium_id          =       $ini->val("MEDIUM_ID","MEDIUM_$SHOP");

$locallogin            =       $login  if (!$locallogin);
$localdbname           =       $dbname if (!$localdbname);

my $cgi               =      new CGI;
my $sql                =        "";
```

```
my $file              =       $cgi->param('URL');
my $UID                =        $cgi->param("PP_PAR[USER_ID]");
   $UID                =        $cgi->param("TK_PAR[USER_ID]") if (!$UID);
   $UID                =        $cgi->param("UID") if (!$UID);

my $tmpl              =       "suche/tmpl/TextSearch2.tmpl";
my $result_tmpl       =       "suche/tmpl/TextSearch_result.tmpl";

my @result            =       undef;
my @result2           =       undef;
my @result3           =       undef;
my @result4           =       undef;
my @result5           =       undef;
my @vorschlag         =       undef;
my @level             =       ("gefundene Produktgruppen","n&auml;heres
Produktumfeld","entferntes Produktumfeld","siehe auch...","Technik/Features");
my @worte             =       undef;
my @such_worte        =       undef;
my @such_modes        =       undef;
my @notfound          =       undef;
my %found             =       undef;

my $content           =       undef;
my $line              =       undef;
my $anzahl            =       undef;
my $i                 =       undef;
my $j                 =       undef;
my $k                 =       undef;
my $suchwort          =       undef;
my $mode              =       undef;
my $ausgabe           =       undef;
my $loop              =       undef;
my $loop2             =       undef;
my $flag              =       undef;
my $flag2             =       undef;
my $findflag          =       undef;
my $dbh               =       undef;
my $value             =       undef;
my $value_vor         =       undef;
my $code              =       undef;
my $code1             =       undef;
my $code2             =       undef;
my $code_neu          =       undef;
my $code_paste        =       undef;
my $level             =       undef;
my $gruppe            =       undef;
my $suche             =       undef;
my $suche_gesamt      =       undef;
my $weg               =       undef;
my $count             =       undef;
my $parameter         =       undef;
my $add_par           =       undef;
my $pos               =       $cgi->param("pos");
my $nav               =       $cgi->param("navi.x");
my $wert              =       undef;
my $vor               =       undef;
my $vorflag           =       undef;
my $vorschlag         =       undef;
my $vorschlag_mode    =       undef;
my $paste             =       undef;
my $paste2            =       undef;
my $paste3            =       undef;
my $notfound_flag     =       undef;
my $display           =       undef;
my $display_bak       =       undef;
my $leer              =       undef;
my $bestellnr         =       undef;
my @ergebnisse        =       undef;
my @ergebnisse2       =       undef;
my $counter           =       undef;
my $anfrage           =       undef;
my @alles             =       undef;
my $tmp               =       undef;
```

```
my $language           =       undef;
my $text1              =       undef;
my $text2              =       undef;
my $text3              =       undef;
my $text4              =       undef;
my $satz               =       undef;

#Texte
$language = $cgi->param("language");

if ($language eq "deutsch")
{
       @level=("gefundene Produktgruppen","n&auml;heres Produktumfeld","entferntes
Produktumfeld","siehe auch...","Technik/Features");
       $text1="nicht gefunden.Meinen sie vielleicht";
       $text2="Leider keine passenden Produkte gefunden.";
       $text3="<BODY BGCOLOR=\"#ffffff\"><br><font face=\"arial,helvetica,lucida\"
size=\"2\">Bitte w&auml;hlen Sie auf der rechten Seite eine passende Produktkategorie
oder w&auml;hlen<br>Sie im Drop Down Men&uuml;<br>einen Bereich aus.</font></BODY>";
       $text4 ="Anzahl";
       $satz = "Wortvorschlag";
}
elsif ($language eq "englisch")
{
       @level=("Found product groups","Closely related product range","Remotely related
product range","See also...","Technical/Features");
       $text1="Not found. Do you think that";
       $text2="Sorry, no suitable products found.";
       $text3="<BODY BGCOLOR=\"#ffffff\"><br><font face=\"arial,helvetica,lucida\"
size=\"2\">Please select a suitable product category on the right or mark a<br>field in
the dropdown menu.</font></BODY>";
       $text4="Quantity";
       $satz="Suggested word";
}
elsif ($language eq "franzoesisch")
{
       @level=("cat&eacute;goriees de produits trouv&eacute;es","cat&eacute;gorie
proche","cat&eacute;gorie plus &eacute;loign&eacute;e","voir
aussi...","technique/fonctionnalit&eacute;s");
       $text1="introuvable. Nous vous proposons";
       $text2="Aucun produit correspondant n'a &eacute;t&eacute; trouv&eacute;.";
       $text3="<BODY BGCOLOR=\"#ffffff\"><br><font face=\"arial,helvetica,lucida\"
size=\"2\">Choisissez une cat&eacute;gorie de<br>produits (&agrave; droite) ou
bien<br>s&eacute;lectionnez une rubrique dans<br>le menu d&eacute;roulant (&agrave;
gauche).</font></BODY>";
       $text4="Quantit&eacute;";
       $satz="Proposition de produit";
}
elsif ($language eq "hollaendisch")
{
       @level=("gevonden productgroepen","kleinere productselectie","Grotere
productselectie","zie ook...","technique/features");
       $text1="niet gevonden. Bedoelde u misschien...";
       $text2="Helaas werd er geen passend artikel gevonden.";
       $text3="<BODY BGCOLOR=\"#ffffff\"><br><font face=\"arial,helvetica,lucida\"
size=\"2\">Kies de juiste productgroep op<br> de rechterpagina of maak uw<br> keuze uit
het Drop Down menu.</font></BODY>";
       $text4="Aantal";
       $satz="Woordsuggestie";
}
else
{
       @level=("gefundene Produktgruppen","n&auml;heres   Produktumfeld","entferntes
Produktumfeld","siehe auch...","Technik/Features");
       $text1="nicht gefunden.Meinen sie vielleicht";
       $text2="Leider keine passenden Produkte gefunden.";
       $text3="<BODY BGCOLOR=\"#ffffff\"><br><font face=\"arial,helvetica,lucida\"
size=\"2\">Bitte w&auml;hlen Sie auf der rechten Seite eine passende Produktkategorie
oder w&auml;hlen<br>Sie im Drop Down Men&uuml;<br>einen Bereich aus.</font></BODY>";
       $text4="Anzahl";
       $satz = "Wortvorschlag";
}
```

```
if ($cgi->param("navi2.x"))
{
	$nav = "weiter";
}
else
{
	$nav= "zurück";
}

if ($cgi->param("PP_PAR[QUICK_LIST]") eq 1)
{
	# PP_PAR[QUICK_LIST] is 1
	# QUICK_LIST stands for quick search via "go"-button
	$bestellnr = $cgi->param("PP_PAR[SUCHWORT]");
	if (!$bestellnr)
	{
		# $bestellnr is empty -> no PP_PAR[SUCHWORT] in cgi's
		print "location: /cgi-
bin/ass/AssDisplay.pl?URL=/nav/frames/f_unten_powersuche.html&PP_PAR[USER_ID]=$UID&langu
age=$language\n\n";
		Apache::exit();
	}
	if (($bestellnr =~ /\A(\d+){6}/) || (($bestellnr =~ /\A(\d+){7}/) &&
($SHOP="FR")))
	{
		# $bestellnr is at least 6 digits long or if shop is france shop
$bestellnr is at least 7 digits long
		if (($bestellnr =~ /\A(\d+){7}/) && ($SHOP="FR"))
		{
			$bestellnr=substr($bestellnr,0,7);
		}
		else
		{
			$bestellnr=substr($bestellnr,0,6);
		}

		#    $sql="SELECT     AN.SEITEN_ID
		#          FROM              ABSATZ_INFO AI, ABSATZ_SEITEN AN
		#          WHERE     AI.BESTELLNUMMER=$bestellnr
		#          AND       AN.ABSATZ_ID = AI.ABSATZ_ID";
		$sql="exec sp__ass_psuche_sql1 $bestellnr";

		#    $anfrage="SELECT KATEGORIE_ID
		#              FROM   PRODUKTKATEGORIE
		#              WHERE  BESTELLNUMMER=$bestellnr";
		$anfrage="exec sp__ass_psuche_sql2 $bestellnr";

		# connect to database
		$dbh = TKSqlDB->new($server, $locallogin, $localpasswd);
		# executing two sql-statements with the same connection
		@result  = &dblquery($dbh,$sql);
		@result2 = &dblquery($dbh,$anfrage);
		# close connection
		$dbh->close();


		print "location: /cgi-
bin/conshop/ConShop.pl?TK_EV[SHOPINIT]=&TK_PAR[USER_ID]=$UID&TK_PAR[CATID]=$result2[0]&T
K_PAR[PAGEID]=$result[0]\n\n";
		Apache::exit();
	}
	else
	{
		# $bestellnr is not at least 6 digits or 7 digits in france shop
		$bestellnr =~ s/\s/\%24%24/igs;

		print "location: /cgi-
bin/ass/AssDisplay.pl?URL=/nav/frames/f_unten_suche.html&TK_PAR[USER_ID]=$UID&PP_PAR[QUI
CK_LIST]=2&PP_PAR[QUICK_CONTENT]=1&PP_PAR[SUCHWORT]=$bestellnr&PP_PAR[language]=$langua
e\n\n";
		Apache::exit();
	}
}
```

```
# full text search feature
if ($cgi->param('volltext_submit.x'))
{
       $suchwort = $cgi->param("volltext");
       $weg=pack ("c",32);
       $suchwort =~ s/([a-z][A-Z]*)/$1/igs;
       $suchwort =~ s/\+\Z//igs;
       $suchwort =~ s/\+/\%2b/igs;
       $suchwort =~ s/$weg/+/igs;

       print "location: /cgi-
bin/conshop/ConShop.pl?TK_PAR[SUCHTEXT]=$suchwort&TK_PAR[USER_ID]=$UID&TK_EV[TEXTSEARCH]
=1\n\n";
       Apache::exit();
}

# end of simple redirections
# from now on html-page generation starts
print "content-type:text/html\n\n";

if ($cgi->param("PP_PAR[QUICK_LIST]") eq 2)
{
       # seems to be the content of left navigation bar
       # "Please select a suitable product category on the right or mark a<br>field in
the dropdown menu."

       #&level_auswertung("0",0,0,0,$cgi,$SHOP,$medium_id);

       print "<BODY BGCOLOR=\"#ffffff\"><br><font face=\"arial,helvetica,lucida\"
size=\"2\">$text3</font></BODY>";
       Apache::exit();
}

if ($cgi->param("PP_PAR[QUICK_CONTENT]") eq 1)
{
       push @such_worte,$cgi->param("PP_PAR[SUCHWORT]");
       $such_worte[1] =~ s/\$\$/ /igs;
       push @such_modes, "+";
       $vorschlag = $satz;
       if (!$such_worte[0])
       {
              $notfound_flag=2;
       }
}
else
{
       @result = $cgi->param;
       foreach (@result)
       {
              if ($_ =~ /suchbegr/)
              {
                     push @such_worte,$cgi->param($_);
                     $_=~ s/suchbegr(\d*)/$1/igs;
                     push @such_modes, $cgi->param("suchmode$_");
              }
       }

       $vorschlag = $cgi->param("vorschlag");
       $vorschlag_mode = $cgi->param("suchmode_vor");
}

if ($cgi->param('level1.x'))
{
       $gruppe = $cgi->param('art1');
       &level_auswertung("1",$gruppe,$pos,$nav,$cgi,$SHOP,$medium_id);
}
elsif ($cgi->param('level2.x'))
{
       $gruppe = $cgi->param('art2');
       &level_auswertung("2",$gruppe,$pos,$nav,$cgi,$SHOP,$medium_id);
}
elsif ($cgi->param('level3.x'))
{
       $gruppe = $cgi->param('art3');
```

```
        &level_auswertung("3",$gruppe,$pos,$nav,$cgi,$SHOP,$medium_id);
}
elsif ($cgi->param('level4.x'))
{
        $gruppe = $cgi->param('art4');
        &level_auswertung("4",$gruppe,$pos,$nav,$cgi,$SHOP,$medium_id);
}
elsif ($cgi->param('level5.x'))
{
        $gruppe = $cgi->param('art5');
        &level_auswertung("5",$gruppe,$pos,$nav,$cgi,$SHOP,$medium_id);
}
elsif ($cgi->param('level0.x') eq "Anzeigen")
{
        &level_auswertung("-1",0,$pos,$nav,$cgi,$SHOP,$medium_id);
}

open (handle,"$docroot/$tmpl");
while ($line=<handle>)
{
        $content .= $line;
}


$content =~ s/<TK_USER_ID>/$UID/igs;
$content =~ s/<PP_USER_ID>/$UID/igs;

$flag=0;
$i=1;

while ($flag eq 0)
{
        if (!($such_worte[$i] =~ /\[VALUE/i)&& ($such_worte[$i]))
        {
                $suchwort= $such_worte[$i];
                if ($suchwort =~ /Meinen sie vielleicht/i)
                {
                        #$suchwort =~ s/\A(.*?)\s(.*)/$1/igs;
                        #$such_worte[$i] = $suchwort;
                        splice (@such_worte,$i,1);
                        splice (@such_modes,$i,1);
                        next;
                }
                $mode = $such_modes[$i];

                $suchwort= uc $suchwort;
                $suchwort=~ s/ö/OE/igs;
                $suchwort=~ s/ü/UE/igs;
                $suchwort=~ s/ä/AE/igs;
                $suchwort=~ s/Ö/OE/igs;
                $suchwort=~ s/Ü/UE/igs;
                $suchwort=~ s/Ä/AE/igs;
                $suchwort=~ s/ß/SS/igs;
                $suchwort=~ s/_/ /igs;
                push @ergebnisse, "$suchwort";
                push @ergebnisse, "$mode";
        }
        else
        {
                if (!($vorschlag =~ /$satz/)&&($vorschlag))
                {
                        $i++;
                        $suchwort = $vorschlag;
                        $suchwort= uc $suchwort;
                        $suchwort=~ s/ö/OE/igs;
                        $suchwort=~ s/ü/UE/igs;
                        $suchwort=~ s/ä/AE/igs;
                        $suchwort=~ s/Ö/OE/igs;
                        $suchwort=~ s/Ü/UE/igs;
                        $suchwort=~ s/Ä/AE/igs;
                        $suchwort=~ s/ß/SS/igs;
                        $suchwort=~ s/_/ /igs;
                        $mode = $vorschlag_mode;
                        push @ergebnisse, "$suchwort";
```

```
                        push @ergebnisse, "$mode";
                        $flag = 1;
                }
                else
                {
                        $flag = 1;
                }
        }
        $i++;
}

$anfrage = "   create table #SCHLAGWORT (
               WORT_ID     int             null,
               SCHLAGWORT  varchar(100)    null,
               BOOL_MOD    char(1)         null)

               create table #SCHLAGWORT_RESULT (
               SW_LEVEL    int,
               ID          int,
               NAME        varchar(100)    null,
               ANZAHL      int             null)";

$counter = 1;
$count = 1;

while ($counter <= $#ergebnisse)
{
        #if ($cgi->param("suchcode$count")>0)
        #{
        #       $tmp = $cgi->param("suchcode$count");
        #       $anfrage .= "insert into #SCHLAGWORT values
($tmp,\"$ergebnisse[$counter]\",\"$ergebnisse[$counter+1]\")";
        #}
        #else
        #{

        $anfrage .= "insert into #SCHLAGWORT values
(null,\"$ergebnisse[$counter]\",\"$ergebnisse[$counter+1]\")";

        #}

        $counter += 2;
        $count++;
}


$anfrage .= "
        exec sp_schlagwortsuche

        select *
        from #SCHLAGWORT_RESULT";

$sql =<< EOS;
        SELECT *
        FROM #SCHLAGWORT

        drop table #SCHLAGWORT
        drop table #SCHLAGWORT_RESULT
        EOS


$dbh = TKSqlDB->new($server, $locallogin, $localpasswd);
@alles = &dblquery($dbh,$anfrage);
@ergebnisse2 = &dblquery($dbh,$sql);
$dbh->close();

# Hier abfrage, ob Ergebnis gefunden

$counter=0;
$#result2 = -1;
$#result4 = -1;
while ($counter <= $#ergebnisse2 )
{
        push @result2, $ergebnisse2[$counter];
```

```
        push @result2, $ergebnisse2[$counter+2];
        push @result4, $ergebnisse2[$counter];
        push @result4, $ergebnisse2[$counter+1];
        $counter += 3;
}

$counter=1;
@ergebnisse = undef;
$#vorschlag = -1;
$#result3 = -1;

while ($counter <= $#alles+1)
{
        if ($alles[$counter] > 0)
        {
                push @ergebnisse,$alles[$counter];
                push @ergebnisse,$alles[$counter+1];
                push @ergebnisse,$alles[$counter+2];
                push @ergebnisse,$alles[$counter+3];
                push @result3, $alles[$counter+1];
                $counter += 4;
        }
        elsif ($alles[$counter] eq 0)
        {
                push @vorschlag,$alles[$counter+1];
                push @vorschlag,$alles[$counter+2];
                $counter += 4;
        }
        else
        {
                $counter += 4;
        }
}


@result = @ergebnisse;

if ($#result < 0)
{
        $leer = 1;
}

$flag = 0;
$i = 0;
while ($flag eq 0)
{
        if ($result4[$i+1])
        {
                $found{$result4[$i+1]} = $result4[$i];
                $i += 2;
        }
        else
        {
                $flag=1;
        }
}

$anzahl = $#result/4;

$ausgabe = $content;

$code1 = $ausgabe;
$code2 = $ausgabe;
$code1 =~ s/(.*?)\[formular\](.*?)\[\/formular\](.*)/$2/igs;
$code2 =~ s/(.*?)\[formular2\](.*?)\[\/formular2\](.*)/$2/igs;

$code="";
$i=1;
$flag=0;
$j=0;
$flag2=0;

while ($flag == 0)
{
```

```
        if ($such_worte[$i])
        {
                #print $such_worte[$i];
                $wert = $such_worte[$i];
                $wert = uc $wert;
                $wert=~ s/ö/OE/igs;
                $wert=~ s/ü/UE/igs;
                $wert=~ s/ä/AE/igs;
                $wert=~ s/Ö/OE/igs;
                $wert=~ s/Ü/UE/igs;
                $wert=~ s/Ä/AE/igs;
                $wert=~ s/ß/SS/igs;
                $wert=~ s/_/ /igs;

                if ($found{$wert} > 0)
                {
                        $code_paste = $code1;
                        $code_paste =~ s/\[suchbegr\]/suchbegr$i/igs;
                        $code_paste =~ s/\[value\]/\[value$i\]/igs;
                        $code_paste =~ s/\[suchmode\]/suchmode$i/igs;
                        $code_paste =~ s/\[set(.*?)\]/[set$i$1]/igs;
                        $code .= $code_paste;
                }
                else
                {
                        $notfound_flag = 1;
                        $counter = 0;
                        $#notfound = -1;
                        while ($counter <= $#alles)
                        {
                                if ($alles[$counter+1] eq $found{$wert})
                                {
                                        if ($alles[$counter+4])
                                        {
                                                $notfound[$alles[$counter+4]-1] =
$alles[$counter+3];
                                        }
                                        else
                                        {
                                                push @notfound, $alles[$counter+3];
                                        }
                                        #print $alles[$counter+4],",";
                                }
                                $counter += 4;
                        }

                        $code_paste = $code2;
                        $paste = $code2;
                        $paste =~ s/(.*)\[notfound\](.*?)\[\/notfound\](.*)/$2/is;
                        $paste3 = "";
                        $j=0;
                        $flag2=0;
                        $vorschlag[0] = "[value$i] $text1";

                        if ($#notfound < 0)
                        {
                                $code_paste = $code1;
                                $code_paste =~ s/\[suchbegr\]/suchbegr$i/igs;
                                $code_paste =~ s/\[value\]/[value$i]/igs;
                                $code_paste =~ s/\[suchmode\]/suchmode$i/igs;
                                $code_paste =~ s/\[set(.*?)\]/[set$i$1]/igs;
                                $code .= $code_paste;
                                $ausgabe =~ s/<\!--$satz(.*?)-->//igs;
                        }
                        else
                        {
                                while ($flag2 ne 2)
                                {
                                        if ($j<=$#notfound)
                                        {
                                                if ($flag2 eq 0)
                                                {
                                                        $flag2 = 1;
                                                        $paste2 = $paste;
```

```
                                                $paste2 =~
s/\[VALUE\]/$vorschlag[0]/igs;
                                                $paste3 .= $paste2;
                                        }
                                        else
                                        {
                                                $paste2 = $paste;
                                                $paste2 =~
s/\[VALUE\]/$notfound[$j]/igs;
                                                $paste3 .= $paste2;
                                                $j += 1;
                                        }
                                }
                                else
                                {
                                        $flag2=2;
                                }
                        }
                        $code_paste =~
s/\[notfound\](.*?)\[\/notfound\]/$paste3/is;
                        $code_paste =~ s/\[suchbegr\]/suchbegr$i/igs;
                        $code_paste =~ s/\[suchmode\]/suchmode$i/igs;
                        $code_paste =~ s/\[set(.*?)\]/[set$i$1]/igs;
                        $code .= $code_paste;
                        $ausgabe =~ s/<\!--$satz(.*?)-->/$1/igs;
                }
        }
        else
        {
                if ($cgi->param("eingabefelder.x"))
                {
                        $code_paste = $code1;
                        $code_paste =~ s/\[suchbegr\]/suchbegr$i/igs;
                        $code_paste =~ s/\[value\]//igs;
                        $code_paste =~ s/\[suchmode\]/suchmode$i/igs;
                        $code_paste =~ s/\[set(.*?)\]/[set$i$1]/igs;
                        $code .= $code_paste;
                        $i++;
                }
                if (!($vorschlag =~ /$satz/) && ($vorschlag))
                {
                        $wert = $vorschlag;
                        $code_paste = $code1;
                        $code_paste =~ s/\[suchbegr\]/suchbegr$i/igs;
                        $code_paste =~ s/\[value\]/$wert/igs;
                        $code_paste =~ s/\[suchmode\]/suchmode$i/igs;
                        $code_paste =~ s/\[set(.*?)\]/[set$i$1]/igs;
                        $code .= $code_paste;
                        $code_paste = $code1;

                        $i++;

                        if ($notfound_flag ne 1)
                        {
                                $code_paste =~ s/\[suchbegr\]/suchbegr$i/igs;
                                $code_paste =~ s/\[value\]//igs;
                                $code_paste =~ s/\[suchmode\]/suchmode$i/igs;
                                $code_paste =~ s/\[set(.*?)\]/[set$i$1]/igs;
                                $code .= $code_paste;
                        }
                }
                else
                {
                        #if ($notfound_flag ne 1){
                        $code_paste = $code1;
                        $code_paste =~ s/\[suchbegr\]/suchbegr$i/igs;
                        $code_paste =~ s/\[value\]//igs;
                        $code_paste =~ s/\[suchmode\]/suchmode$i/igs;
                        $code_paste =~ s/\[set(.*?)\]/[set$i$1]/igs;
                        $code .= $code_paste;
                }
                $flag=1;
        }
```

```
	$i++;
}

$ausgabe =~ s/\[formular\](.*?)\[\/formular2\]/$code/igs;
$code = $ausgabe;
$ausgabe =~ s/(.*)\[start\](.*)/$1/igs;

$code =~
s/(.*)<INPUT(.*?)\[sc_value\]\">(.*)<INPUT(.*?)\[sm\](.*?)>(.*)/<INPUT$2\[sc_value\]\">$
3<INPUT$4\[sm\]\">/igs;
$flag=0;
$i=1;
$j=0;

while ($flag eq 0)
{
	if ($ausgabe =~ /\[value$i\]/i)
	{
		$value = $such_worte[$i];
		$value =~ s/_/ /igs;
		if (!($value =~ /\[VALUE/i)&& $value ne "")
		{
			$ausgabe =~ s/\[VALUE$i\]/$value/igs;
			$code_neu = $code;
			$code_neu =~ s/suchcode/suchcode$i/igs;
			$code_neu =~ s/suchmode/suchmode$i/igs;
			$code_neu =~ s/\[sc_value\]/$result2[$j]/igs;
			$mode = $result2[$j+1];
			$code_neu =~ s/\[sm\]/$mode/igs;

			$ausgabe =~
s/(.*)<(.*?)\[sc_value\]\">(.*)<.*?\[sm\]\">/$1$code_neu\n$code/igs;
			$ausgabe =~
s/\[PARAM\]<(.*?)\[sc_value\]\">(.*?)<.*?\[sm\]\">/$code_neu\n\[PARAM\]$code/igs;
			$mode = $such_modes[$i];

			if ($mode eq "+")
			{
				$ausgabe =~ s/\[set$i+\]/selected/igs;
			}
			elsif ($mode eq "-")
			{
				$ausgabe =~ s/\[set$i-\]/selected/igs;
			}
			else
			{
				$ausgabe =~ s/\[set$i\o\]/selected/igs;
			}
		}
		else
		{
			$ausgabe =~ s/\[VALUE$i\]//igs;
		}
	}
	else
	{
		if (!($vorschlag =~ /$satz/))
		{
			$value = $vorschlag;
			$value =~ s/_/ /igs;
			if (!($value =~ /\[VALUE/i)&& $value ne "")
			{
				$ausgabe =~ s/\[VALUE$i\]/$value/igs;
				$code_neu = $code;
				$code_neu =~ s/suchcode/suchcode$i/igs;
				$code_neu =~ s/suchmode/suchmode$i/igs;
				$code_neu =~ s/\[sc_value\]/$result2[$j]/igs;

				$mode = $vorschlag_mode;
				$code_neu =~ s/\[sm\]/$mode/igs;
				$ausgabe =~
s/(.*)<(.*?)\[sc_value\]\">(.*)<.*?\[sm\]\">/$1$code_neu\n$code/igs;

				if ($mode eq "+")
```

```
                                {
                                        $ausgabe =~ s/\[set$i+\]/selected/igs;
                                }
                                elsif ($mode eq "-")
                                {
                                        $ausgabe =~ s/\[set$i-\]/selected/igs;
                                }
                                else
                                {
                                        $ausgabe =~ s/\[set$i\o\]/selected/igs;
                                }
                        }
                        $flag=1;
                }
                else
                {
                        $flag=1;
                }
        }
        $i += 1;
        $j += 2;
}

$ausgabe =~ s/\[PARAM\]//igs;
$code = $ausgabe;
$code =~ s/(.*)\[vorschlag\](.*?)\[\/vorschlag\](.*)/$2/is;
$code_neu = $code;
$code_paste = $code;
$code="";

$i=0;
$flag=0;
$vorschlag[0] = $satz;
while ($flag ne 2)
{
        if ($i<=$#vorschlag)
        {
                if ($flag eq 0)
                {
                        $flag = 1;
                        $code_neu =~ s/\[vorschlag_name\]/$vorschlag[$i]/igs;
                        $code .= $code_neu;
                        $code_neu = $code_paste;
                }
                else
                {
                        $code_neu =~ s/\[vorschlag_name\]/$vorschlag[$i+1]/igs;
                        $code .= $code_neu;
                        $code_neu = $code_paste;
                        $i += 2;
                }
        }
        else
        {
                $flag=2;
        }
}

if ($notfound_flag > 0)
{
        $ausgabe =~ s/\[formular3\](.*?)\[\/formular3\]//igs;
}
else
{
        $ausgabe =~ s/\[formular3\](.*?)\[\/formular3\]/$1/igs;
}

$ausgabe =~ s/\[vorschlag\](.*?)\[\/vorschlag\]/$code/igs;
#print $ausgabe;
$display .= $ausgabe;

$k=0;
for ($j=;$j<=5;$j++)
{
```

```
        $ausgabe = $content;
        $ausgabe =~ s/(.*)\[start\](.*?)\[loop\](.*)/$2/igs;
        $loop = $content;
        $loop =~ s/(.*)\[loop\](.*)\[\/loop\](.*)/$2/igs;
        $flag = 0;
        for ($i=1;$i<=$anzahl;$i++)
        {
                if ($result[$i*4-3] eq $j)
                {
                        if ($flag ne 1)
                        {
                                $flag = 1;
                                $ausgabe =~ s/\[level\]/$level[$j-1]/igs;
                                $ausgabe =~ s/\[art\]/art$j/igs;
                                #print $ausgabe;
                                $display .= $ausgabe;
                        }
                        $loop =~ s/\[art_value\]/$result3[$k]/igs;
                        $loop =~ s/\[art_name\]/$result[$i*4-1]/igs; #($result[$i*4]x)
                        #print $loop;
                        $display .= $loop;
                        $loop = $content;
                        $loop =~ s/(.*)\[loop\](.*)\[\/loop\](.*)/$2/igs;
                        $k++;
                }
        }
        if ($flag eq 1)
        {
                $ausgabe = $content;
                $ausgabe =~ s/(.*)\[\/loop\](.*)\[ende\](.*)/$2/igs;
                $ausgabe =~ s/\[submit\]/level$j/igs;
                $display .= $ausgabe;
                #print $ausgabe;
        }
}

$ausgabe = $content;
$ausgabe =~ s/(.*)\[ende\](.*)/$2/igs;

$i=1;
$flag=0;
while ($flag == 0)
{
        $suche = $such_worte[$i];
        if ($suche)
        {
                $mode = $such_modes[$i];
                @worte = split / /,$suche;
                $suche = undef;
                foreach (@worte)
                {
                        if (length ($_) >3 )
                        {
                                $suche .= $_." ";
                        }
                }
                if ($suche)
                {
                        $suche=~ s/\s\Z//igs;
                        $suche=~ s/\s/ +/igs;
                        $suche_gesamt .= "$mode$suche ";
                }
        }
        else
        {
                if (!($vorschlag =~ /$satz/))
                {
                        $mode = $vorschlag_mode;
                        $suche = $vorschlag;
                        @worte = split / /,$suche;
                        $suche = undef;
                        foreach (@worte)
                        {
                                $suche .= $_." ";
```

```
                        }
                        if ($suche)
                        {
                                $suche=~ s/\s\Z//igs;
                                $suche=~ s/\s/ +/igs;
                                $suche_gesamt .= "$mode$suche ";
                        }
                }
                $flag=1;
        }
        $i++;
}

$ausgabe =~ s/\[VOLLTEXT\]/$suche_gesamt/igs;
#print $ausgabe;
$display .= $ausgabe;

if (!$leer)
{
        $display =~ s/<\!--leer(.*?)-->//igs;
        $display =~ s/<\!--header2(.*?)-->//igs;
        $display =~ s/<\!--header1(.*?)-->/$1/igs;
        $display_bak = $display;
        $display =~ s/\[part1\](.*?)\[\/part1\]//igs;
        $display_bak =~ s/(.*?)\[part1\](.*?)\[\/part1\](.*)/$2/igs;
        $display =~ s/\[\/part2\]/$display_bak/;
}
else
{
        $display =~ s/<\!--leer(.*?)-->/$1/igs;
        $display =~ s/<\!--header1(.*?)-->//igs;
        $display =~ s/<\!--header2(.*?)-->/$1/igs;
        $display =~ s/\[\/part1\](.*)\[\/part2\]//igs;
        $display =~ s/\[part1\]//igs;
}

print $display;
$cgi=undef;
Apache::exit();


sub level_auswertung
{
  my $anzahl      =       0;
  my $ergebnisse  =       5;
  my $anz         =       undef;
  my $dbh         =       undef;
  my $flag        =       undef;
  my $i           =       undef;
  my $suchwort    =       undef;
  my $mode        =       undef;
  my $wort        =       undef;
  my @such_worte  =       undef;
  my @such_modes  =       undef;
  my @aus1        =       undef;
  my @aus2        =      undef;
  my @result      =       undef;
  my @result2     =       undef;
  my @result3     =       undef;
  my @result4     =       undef;
  my @result5     =       undef;
  my $level       =       shift;
  my $gruppe      =       shift;
  my $pos         =      shift;
  my $nav         =       shift;
  my $cgi         =       shift;
  my $shop        =       shift;
  my $medium_id   =       shift;
  my $pos2        =       undef;
  my $anfrage     =       undef;
  my $sql         =       undef;
  my $sql2        =       undef;
  my $sql3        =       undef;
```

```
  my $UID         = $cgi->param("PP_PAR[USER_ID]");
     $UID         = $cgi->param("TK_PAR[USER_ID]") if (!$UID);
     $UID         = $cgi->param("UID") if (!$UID);

$anfrage = "create table #SCHLAGWORTID (
                WORT_ID         int     null,
                BOOL_MOD        char(1) null
        )
        create table #SCHLAGWORT_PRODUKT (
               BESTELLNUMMER    int,
                POS             numeric (5,0) identity,
                constraint POS   primary key (POS)
        )";

if ($level eq 0)
{
        push @such_worte,$cgi->param("PP_PAR[SUCHWORT]");
        push @such_modes, "+";
        $such_worte[1]= uc($such_worte[1]);
        $such_worte[1]=~ s/ö/OE/igs;
        $such_worte[1]=~ s/ü/UE/igs;
        $such_worte[1]=~ s/ä/AE/igs;
        $such_worte[1]=~ s/Ö/OE/igs;
        $such_worte[1]=~ s/Ü/UE/igs;
        $such_worte[1]=~ s/Ä/AE/igs;
        $such_worte[1]=~ s/ß/SS/igs;
        $such_worte[1]=~ s/_/ /igs;
        $such_worte[1]=~ s/\$\$/ /igs;

        $sql=<<EOS;
                SELECT WORT_ID
                FROM SW_HOMONYM
                WHERE
                HOMONYM=\"$such_worte[1]\"
                EOS

        @result=&dblquery2($sql);
        $such_worte[1]=$result[0];

        $vorschlag = "$satz";
}
else
{
        if ($level eq "-1")
        {
                $level = 0;
        }
        @result = $cgi->param;
        foreach (@result)
        {
                if ($_ =~ /suchcode/)
                {
                        push @such_worte,$cgi->param($_);
                        $_=~ s/suchcode(\d*)/$1/igs;
                        push @such_modes, $cgi->param("suchmode$_");
                }
        }
        $vorschlag = $cgi->param("vorschlag");
        $vorschlag_mode = $cgi->param("suchmode_vor");
}

$flag=0;
$i=1;

while ($flag eq 0)
{
        if (!($such_worte[$i] =~ /\[sc_value/i)&& ($such_worte[$i]))
        {
                if ($such_worte[$i] >= 0)
                {
                        $suchwort= $such_worte[$i];
                        $mode = $such_modes[$i];
                        #print "$suchwort, $such_modes[$i]";
```

```
                        $anfrage .="insert into #SCHLAGWORTID
values($suchwort,\"$mode\")";
                }
        }
        else
        {
                $flag = 1;
        }
        $i++;
}

if ($level == 0)
{
        $anfrage .= "exec sp_schlagwortprodukt";
}
else
{
        $anfrage .= "exec sp_schlagwortprodukt $level,$gruppe";
}

$sql =     "SELECT distinct
            SA.SEITEN_ID,
            AI.ABSATZ_ID,
            AI.ABSATZ_TITEL,
            AI.THUMBNAIL,
            AI.PROD_ANZ,
            AI.BESTELLNUMMER,
            AI.PREIS
    FROM
            ABSATZ_INFO AI,
            ABSATZ_SEITEN SA,
            #SCHLAGWORT_PRODUKT T
    WHERE
            T.BESTELLNUMMER = AI.BESTELLNUMMER
    AND
           AI.ABSATZ_ID = SA.ABSATZ_ID
    AND
           AI.ABSATZ_ID < 200000000
    order by T.POS";

$sql2 = "select distinct
            SA.SEITEN_ID,
            AI.ABSATZ_ID,
            AI.ABSATZ_TITEL,
            AI.THUMBNAIL,
            AI.PROD_ANZ,
            AI.BESTELLNUMMER
    from
            ABSATZ_INFO AI,
            ABSATZ_SEITEN SA,
            #SCHLAGWORT_PRODUKT T,
            PRODUKT_ABSATZ PA
    where
            AI.ABSATZ_ID = PA.ABSATZ_ID
    and
           AI.ABSATZ_ID = SA.ABSATZ_ID
    and
            PA.BESTELLNUMMER in (SELECT BESTELLNUMMER FROM #SCHLAGWORT_PRODUKT)
    and
           PA.ABSATZ_ID > 200000000
    drop table #SCHLAGWORTID
    drop table #SCHLAGWORT_PRODUKT";

$dbh = TKSqlDB->new($server, $locallogin, $localpasswd);
@result=&dblquery($dbh,$anfrage);
@result2=&dblquery($dbh,$sql);
@aus2=&dblquery($dbh,$sql2);
$dbh->close();

if ($#result2 ne "-1")
{
        push @aus1, @result2;            #Alle Produkte aus Absatz_info
}
```

```
$anz = $#aus1/7+($#aus2+1)/6;

$content="";
open (handle, "$docroot/$result_tmpl");
while ($line =<handle>)
{
	$content .= $line;
}
close(handle);

$content =~ s/<TK_USER_ID>/$UID/igs;
$content =~ s/\[level_name\]/level$level/igs;
$content =~ s/\[art_name\]/art$level/igs;
$content =~ s/\[art_value\]/$gruppe/igs;

if ($pos =~ /\[pos\]/)
{
	$pos = 0;
}

if (!$pos)
{
	$pos = 0;
}
if ($nav eq "weiter")
{
	$pos += 5;
}
else
{
	if ($pos ne 0)
	{
		$pos -= 5;
	}
}
$pos2 = $pos+$ergebnisse;
$content =~ s/\[pos\]/$pos/igs;

if ($anz <= $ergebnisse)
{
	if ($anz)
	{
		$content =~ s/\[nummer\]/1 - $anz/igs;
	}
	else
	{
		$content =~ s/\[nummer\]/0 - $anz/igs;
	}
}
elsif ($pos+$ergebnisse >= $anz)
{
	$anz = int $anz;
	$pos++;
	$content =~ s/\[nummer\]/$pos -$anz/igs;
	$pos--;
}
else
{
	if ($pos eq 0)
	{
		$content =~ s/\[nummer\]/1 - $pos2/igs;
	}
	else
	{
		$pos++;
		$content =~ s/\[nummer\]/$pos - $pos2/igs;
		$pos--;
	}
}

$ausgabe = $content;
$code = $ausgabe;
```

```
$code =~
s/(.*)<INPUT(.*?)\[sc_value\]\">(.*)<INPUT(.*?)\[sm\](.*?)>(.*)/<INPUT$2\[sc_value\]\">$
3<INPUT$4\[sm\]\">/igs;

$flag=0;
$i=1;

while ($flag eq 0)
{
        if ($such_worte[$i])
        {
                $value = $such_worte[$i];
                $code_neu = $code;
                $code_neu =~ s/suchcode/suchcode$i/igs;
                $code_neu =~ s/suchmode/suchmode$i/igs;
                $code_neu =~ s/\[sc_value\]/$value/igs;

                $mode = $such_modes[$i];
                $code_neu =~ s/\[sm\]/$mode/igs;
                $ausgabe =~
s/(.*)<(.*?)\[sc_value\]\">(.*)<.*?\[sm\]\">/$1$code_neu\n$code/igs;

                if ($mode eq "+")
                {
                        $ausgabe =~ s/\[set$i+\]/selected/igs;
                }
                elsif ($mode eq "-")
                {
                        $ausgabe =~ s/\[set$i-\]/selected/igs;
                }
                else
                {
                        $ausgabe =~ s/\[set$i\o\]/selected/igs;
                }
        }
        else
        {
                $flag=1;
        }
        $i++;
}

$content = $ausgabe;

$loop = $content;
$loop =~ s/(.*?)\[loop\](.*)\[\/loop\](.*)/$1/igs;

$anzahl = $anzahl+($#aus1/7);
$anz = int $anz;
$loop =~ s/\[ANZAHL\]/$anz/igs;

if ($anzahl eq 1)
{
        $loop =~ s/Eintr&auml;ge/Eintrag/igs;
}
print $loop;

@result = @aus1;
shift @result;

$loop = $content;
$loop =~ s/(.*)\[loop\](.*)\[\/loop\](.*)/$2/igs;
$flag = 0;
$findflag = 0;
$i = 0;
$count = 0;

if ($#result>0)
{
        $findflag=1;
        while ($flag eq 0)
        {
                if ($count >= $pos)
                {
```

```
                        if (!$result[$i+2])
                        {
                                $flag=1;
                        }
                        else
                        {
                                $loop2= $loop;
                                $loop2 =~ s/\[NAME\]/$result[$i+2]/igs;
                                $loop2 =~ s/\[BESTELLNUMMER\]/$result[$i+5]/igs;
                                if (!($result[$i+6] =~ /\.(\d+)\Z/))
                                {
                                        $result[$i+6] .= ".-";
                                }
                                elsif ($result[$i+6] =~ /\.(\d)\Z/)
                                {
                                        $result[$i+6] .= "0";
                                }
                                $loop2 =~ s/\[PREIS\]/$result[$i+6]/igs;
                                if ($result[$i+3])
                                {
                                        $loop2 =~ s/\[BILD\]/\/$result[$i+3]/igs;
                                }
                                else
                                {
                                        $loop2 =~
s/\[BILD\]/\/shopping\/images\/thumbnail.gif/igs;
                                }

                                $loop2 =~ s/<TK_PAGEID>/$result[$i]/igs;
                                $loop2 =~ s/<TK_MEDIUM>/$medium_id/igs;
                                $loop2 =~ s/<TK_BESTELLNUMMER>/$result[$i+5]/igs;
                                $loop2 =~ s/<!--PP_PAR\[BESTELLEN\](.*?)-->/$1/igs;

                                print $loop2;
                                $count++;
                                $i += 7;

                                if (($i-1 >= $#result) || ($count >= $pos+$ergebnisse))
                                {
                                        $flag = 1;
                                }
                        }
                }
                else
                {
                        if ($i-1 >= $#result )
                        {
                                $flag=1;
                        }
                        else
                        {
                                $count++;
                                $i +=7;
                        }
                }
        }
}

if (($aus2[5])&&($#aus2>0))
{
        $findflag=1;
        $flag = 0;
        if ($count >= $pos+$ergebnisse)
        {
                $flag=1;
        }
        $i = 0;

        @result = @aus2;
        $loop = $content;
        $loop =~ s/(.*)\[loop\](.*)\[\/loop\](.*)/$2/igs;
        while ($flag eq 0)
        {
                if ($count >= $pos)
```

```
                {
                        $loop2= $loop;
                        $loop2 =~ s/\[NAME\]/$result[$i+2]/igs;
                        $loop2 =~ s/\[BESTELLNUMMER\]/$result[$i+5]/igs;
                        $loop2 =~ s/\[PREIS\]/$result[$i+4]/igs;
                        if ($result[$i+3])
                        {
                                $loop2 =~ s/\[BILD\]/\/$result[$i+3]/igs;
                        }
                        else
                        {
                                $loop2 =~ s/\[BILD\]/\/shopping\/images\/thumbnail.gif/igs;

                        }
                        $loop2 =~ s/<TK_PAGEID>/$result[$i]/igs;
                        $loop2 =~ s/<TK_MEDIUM>/62/igs;
                        $loop2 =~ s/DM/$text4/gs;
                        $loop2 =~ s/ÖS/$text4/gs;
                        $loop2 =~ s/FF/$text4/gs;
                        $loop2 =~ s/Hfl./$text4/gs;
                        $loop2 =~ s/CHF/$text4/gs;
                        print $loop2;

                        $count++;
                        $i += 6;
                        if (($i-1 >= $#result) || ($count >= $pos+$ergebnisse))
                        {
                                $flag = 1;
                        }
                }
                else
                {
                        $count++;
                        $i +=6;
                }
        }
}

if ($findflag eq 0)
{
        print "<tr><td colspan=2><font face=\"arial,helvetica,lucida\" size=\"2\"
color=\"#000000\">$text2</font></td></tr>";
}

$loop = $content;
$loop =~ s/(.*?)\[loop\](.*?)\[\/loop\](.*)/$3/igs;

if ($anz <=$ergebnisse)
{
        $loop =~ s/<\[nav1\]>(.*?)<\/\[nav1\]>//igs;
        $loop =~ s/<\[nav2\]>(.*?)<\/\[nav2\]>//igs;
}
elsif ($pos eq 0)
{
        $loop =~ s/<\[nav1\]>(.*?)<\/\[nav1\]>//igs;
        $loop =~ s/<\[nav2\]>(.*?)<\/\[nav2\]>/$1/igs;
}
elsif (($pos+$ergebnisse)>=$anz)
{
        $loop =~ s/<\[nav1\]>(.*?)<\/\[nav1\]>/$1/igs;
        $loop =~ s/<\[nav2\]>(.*?)<\/\[nav2\]>//igs;
}
else
{
        $loop =~ s/<\[nav1\]>(.*?)<\/\[nav1\]>/$1/igs;
        $loop =~ s/<\[nav2\]>(.*?)<\/\[nav2\]>/$1/igs;
}

print $loop;
Apache::exit();

}

# Datenbankabfrage ############
```

```
##############################

sub dblquery
{
        my $dbh = shift;
         my ($cmd) = @_;
         my @data;
        my $row;
        my $col;
         #my $dbh      = TKSqlDB->new($server, $login, $passwd);

         my $res = $dbh->query ($cmd);

        foreach $row (@$res)
        {
                foreach $col (@$row)
                {
                        push(@data,$col);
                }
        }
        #$dbh->close();

        return (@data);

}

# Produktdatenabfrage ########
##############################
sub dblquery2
{
         my ($cmd) = @_;
         my @data;
        my $row;
        my $col;
         my $dbh       = TKSqlDB->new($server, $locallogin, $localpasswd);

         my $res = $dbh->query ($cmd);

        foreach $row (@$res)
        {
                foreach $col (@$row)
                {
                        push(@data,$col);
                }
        }
        $dbh->close();

        return (@data);
}

# EOF
```

AssDisplay.pl

```
#!/usr/bin/perl

#Name:         DisplayHTML.pl
#Params:       URL (absolute path)
#              UID (user id)
#Author:       Joerg Luettgau
#Date:         10/11/1998
#              (c) Pixelpark GmbH
#               Version: 1.2 (990810 - Added "use IniConf")

use lib "../conshop";
use strict;
use CGI;

use TKSqlDB;
use Apache;
use IniConf;
```

```
use vars qw(%tmpls);

### Serverkonfiguration aus INI-Datei holen
#######################################################################

my $ini         =  IniConf->new( -file => "../ini/AssAddOn.ini");

my $server     =  $ini->val("DB_LOGIN","DB_SERVER1");
my $login      =  $ini->val("DB_LOGIN","DB_LOGIN1");
my $passwd     =  $ini->val("DB_LOGIN","DB_PASSWD1");
my $dbname     =  $ini->val("DB_LOGIN","DB_NAME1");


my $query      =      new CGI;

my $file       =      $query->param('PP_PAR[URL]');
my $file_user   =       $query->param('PP_PAR[URL_USER]');
my $UID         =       $query->param("PP_PAR[USER_ID]");


# !!! Die folgenden beiden Zeilen sollten nur uebergangsweise
#     beibehalten werden, ein sauberer Shop soll diese
#     Parameter nicht mehr beinhalten !!!

   $file        =       $query->param('URL') if (!$file);
   $UID         =       $query->param("TK_PAR[USER_ID]") if (!$UID);
   $UID         =       $query->param("UID") if (!$UID);

my $valid       =       1;
$valid = 0 if ($file =~ /\.\.+/);
#$valid = 0 if ($file =~ /[\;\<\>\*\|\`\&\$\!\#\(\)\[\]\{\}\:\'\"]/);

if ( !$valid ) {
  print "content-type: text/html\n\n";
  print "invalid URL requested by $ENV{REMOTE_ADDR}<br>webmaster has been informed due
to security reasons<br>\n\n";

  warn "-----------------------------------------------\n";
  warn "$0:\n";
  warn "Illegal URL requested by $ENV{REMOTE_ADDR}\n";
  warn "-----------------------------------------------\n";
  Apache::exit();
}


my $server_name       =      $ENV{'SERVER_NAME'};
my $docroot    =      $ENV{'DOCUMENT_ROOT'};

my $rel        =      undef;
my $href       =      undef;
my $line       =      undef;
my $content     =       undef;
my $wert        =       undef;
my @parameter   =       undef;

my @result      =       ();
my $result      =       undef;

if ($file =~ /http\:\/\//) {
  print "location: $file\n\n";
  Apache::exit();
}
elsif ($file =~ /\/cgi\-bin/igs) {
  print "location: $file\n\n";
  Apache::exit();
}

if (!(-e $docroot.$file)) {
  $file =~ s/(.*\/)*(.*)/$1c_index\.html/;
}

#if ($file_user ne "") {
#  @result=&dbquery("SELECT NAME,LOGIN_NAME FROM CUSTOMER WHERE CUSTOMER_ID=(SELECT
CUSTOMER_ID FROM SESSIONS WHERE SESSION_DATA=\"$UID\")");
```

```
#}

#$file = $file_user if (($result[0]) && ($result[1]));

print "content-type:text/html\n\n";

#if (!($UID =~ /(\d+)/) ) {
 #print" ";
 #exit(0);
#}

$rel = $file;
$rel =~ s/(.*)\/.*/$1\//;
$rel = "/".$rel if (! ($rel =~ /^\//));


### parsen des Templates, falls noch nicht als globale Variable
### vorhanden.

if (!$tmpls{"$docroot$file"}) {
  open (PARSE, "<$docroot$file") || die "Aus AssDisplay.pl: Kann $docroot$file nicht
oeffnen!";
  while ($line = <PARSE>) {
    $content .= $line;
  }
  close (PARSE);
  if ($server_name =~ /www/) {
    $tmpls{"$docroot$file"} = $content;
  }

#  warn ("NEUES TEMPLATE");
} else {
  $content = $tmpls{"$docroot$file"};
#  warn ("ALTES TEMPLATE");
}


### USER_ID ersetzen

    $content =~ s/\<TK_USER_ID\>/$UID/sgi;
    $content =~ s/\<PP_USER_ID\>/$UID/sgi;

### relativen Links Pfad voranstellen

    $content =~ s/(HREF|SRC|ACTION|BACKGROUND)(=\"?)(\.{1,2}|([\w\-]+\.)|([\w\-
]+\/))/$1$2$rel$3/ig;

### Entfernen der /ordner/../file - Konstrukte

    $content =~ s/\/[\w\-]*\/\.\.\//\//g;

    if (!($server_name =~ /www/i) ) {
      $content =~ s/<PP_SSL>//igs;
      $content =~ s/<PP_HTTP>//igs;
    }
    else {
      $content =~ s/<PP_SSL>/https\:\/\/$server_name/igs;
      $content =~ s/<PP_http>/http\:\/\/$server_name/igs;
    }

### PP_PAR - Parameter durchschleifen

    @parameter = $query->param;

    foreach (@parameter) {
      if ($_ =~  /PP_PAR/igs) {
        $wert = $query->param($_);
        $_ =~ s/PP_PAR\[(.*?)\]/$1/igs;
        $content =~ s/<PP_PAR_$_>/$wert/igs;
      }
    }

### TK_PAR - Parameter durchschleifen (z. Zt. nur gebraucht fuer statischen
Bannereinstieg, f_unten_bestell.html)
```

```
    @parameter = $query->param;

    foreach (@parameter) {
      if ($_ =~  /TK_PAR/igs) {
        $wert = $query->param($_);
        $_ =~ s/TK_PAR\[(.*?)\]/$1/igs;

        #$content =~ s/<TK_PAR_$_>/$wert/igs;
        $content =~ s/<TK_BESTNR>/$wert/igs;
      }
    }

### Ausgabe der Zeile an STDOUT
  print $content;

# Datenbankabfrage ###########
################################

sub dbquery {


        my ($cmd) = @_;
        my @data;
       my $row;
       my $col;
        my $dbh       = TKSqlDB->new($server, $login, $passwd);

        my $res = $dbh->query ($cmd);

       foreach $row (@$res){
         foreach $col (@$row){
           push(@data,$col);
         }
       }
       $dbh->close();

       return (@data);

}
# EOF
```

Conshop.pl

```
#!/usr/bin/perl

package main;

use strict;

# Fuer Signal Handler
use Carp ();

use vars qw(   $gRequest $gUserId
               $par @par %par
       );

use TKLib;
use TKTemplate;
use TKForm;
use TKCgiParams;
use TKCookie;
use ConConf;
use ConDB;
use ConLib;
use ConTemplate;
use ConSearchForm;
use ConSearch;
use ConShow;
use ConBasket;
#use ConGame;
use ConJubel;
```

```
use ConCyber;

use AssShop;
use NullShop;

$gRequest                           = 'not yet set';
$gUserId                            = undef;
$par                                = undef;
@par                                = ();
%par                                = ();

my $gUserCookieName   = 'ConShopUser';

my $tmplStartFrame           = 'StartFrame.tmpl';
my $tmplNaviRegular          = 'NaviRegular.tmpl';
my $tmplNaviSafe             = 'NaviSafe.tmpl';
my $tmplCatInitFrame  = 'CatInitFrame.tmpl';
my $tmplCatSearch            = 'CatSearch.tmpl';
my $tmplTextFrame            = 'TextFrame.tmpl';
my $tmplTextSearch           = 'TextSearch.tmpl';
my $tmplProfiSearch          = 'ProfiSuche.tmpl';
my $tmplBasket               = 'Basket.tmpl';
my $tmplBrowseFrame          = 'BrowseFrame.tmpl';
my $tmplResultList           = 'ResultList.tmpl';
my $tmplProductPage          = 'ProductPage.tmpl';
my $tmplRefineInitFrame      = 'RefineInitFrame.tmpl';
my $tmplSeachInitFrame= 'SearchInitFrame.tmpl';
my $tmplOrderForm            = 'OrderForm.tmpl';
my $tmplOrderConfirm  = 'OrderConfirm.tmpl';
my $tmplCreditPayStep1= 'CreditPayStep1.tmpl';
my $tmplCreditPayStep2= 'CreditPayStep2.tmpl';
my $tmplWarnLeave            = 'WarnLeave.tmpl';
my $tmplExitFrame            = 'ExitFrame.tmpl';
my $tmplExitWithCookie= 'ExitWithCookie.tmpl';
my $tmplGotCookie            = 'GotCookie.tmpl';
my $tmplBigPict                     = 'BigPict.tmpl';

my $htmlEmptyProcuct  = 'EmptyProduct.html';

my $eventExpireHash = { 'OPENCAT' => 24,
                        'CLOSECAT' => 24,
                        'CATINIT' => 24,
                        'TEXTINIT' => 24,
                        'CATSTART' => 24,
                        'TEXTSTART' => 24,
                        'TEXTFORM' => 24,
                        'BROWSEINIT' => 24,
                        'BROWSESTART' => 24,
                        'CATSEARCH' => 24,
#                       'SHOWPAGE' => 24,
                        'BROWSELIST' => 24,
                        'BASKETSTATE' => 0,
                        'ADDNOTEPAD' => 0,
                        'CATBROWSENEW' => 24,
                        'CATLISTNEW' => 24
                      };

# Jonas:

open D, ">>/tmp/condebug.txt";

print D "GATEWAY_INTERFACE: ".$ENV{GATEWAY_INTERFACE}."\n";
print D $$."\n";

#while ( my ($k,$v) = each %ENV ) {
#       print D "$k => $v\n";
#}
close D;


# :sanoJ


# ------------------------------------------------------------------
```

```
=item doStartFrame( $params, $isCatSearch )
=cut
# ============================================================================

sub doStartFrame($$)
{
	my $params = shift;
	my $event = shift;

	$event =~ s/START$/INIT/;

	my $request = &cgi2ANSI( $gRequest );
	$request =~ s/TK_EV\[[^\]]*\]&?//g;
	$request = '&' . $request if ($request);

	my $tmpl = ConTemplate->new( $tmplStartFrame );
	$tmpl->setSubstRef( {
		'WHICH' => $event,
		'ADDIT' => $request
	} );
	$tmpl->doTagSubstitution();
	$tmpl->printTemplate();
}

# ----------------------------------------------------------------------------
=item doNaviTest( $params )
=cut
# ============================================================================

sub doNaviTest($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new(
		( &getCookie( $gUserCookieName ) eq $gUserId )
		? $tmplNaviRegular
		: $tmplNaviSafe
		);
	$tmpl->doTagSubstitution();
	$tmpl->printTemplate();
}

# ----------------------------------------------------------------------------
=item doCatInit( $params )
=cut
# ============================================================================

sub doCatInit($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplCatInitFrame );
#	my $tmpl = ConTemplate->new( 'Alt'.$tmplCatInitFrame );
	$tmpl->doTagSubstitution();
	$tmpl->printTemplate();
}

# ----------------------------------------------------------------------------
=item doTextInit( $params )
=cut
# ============================================================================

sub doTextInit($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplTextFrame );
	$tmpl->doTagSubstitution();
	$tmpl->printTemplate();
}


# ----------------------------------------------------------------------------
=item doTextSearchForm( $params )
```

```
=cut
# ============================================================================

sub doTextSearchForm($)
{
       my $params = shift;

       my $tmpl = ConTemplate->new( $tmplTextSearch );
       $tmpl->doTagSubstitution();
       $tmpl->doCleanup();
       $tmpl->printTemplate();
}


# ----------------------------------------------------------------------------
=item doTextSearch( $params )
=cut
# ============================================================================

sub doTextSearch($)
{
       my $params = shift;
       my $mediaId = $params->{'PAR'}{'MEDIUM'}
                                    || $gDBCurrSection->{'MEDIA_ID'};

       my $tmpl = ConTemplate->new( $tmplResultList );
       $tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
       my $form = ConSearch->new( $params, $tmpl );

       $form->doTextSearch();
}

# ----------------------------------------------------------------------------
=item doProfiSearchForm( $params )
=cut
# ============================================================================

sub doProfiSearchForm($)
{
       my $params = shift;
       my $mediaId = $params->{'PAR'}{'MEDIUM'}
                                    || $gDBCurrSection->{'MEDIA_ID'};

       my $tmpl = ConTemplate->new( $tmplProfiSearch );
       $tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
       my $form = ConSearch->new( $params, $tmpl );

       $form->doProfiSearchForm();
}

# ----------------------------------------------------------------------------
=item doProfiSearch( $params )
=cut
# ============================================================================

sub doProfiSearch($)
{
       my $params = shift;

       my $mediaId = $params->{'PAR'}{'MEDIUM'}
                                    || $gDBCurrSection->{'MEDIA_ID'};

       my $tmpl = ConTemplate->new( $tmplResultList );
       $tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
       my $form = ConSearch->new( $params, $tmpl );

       $form->doProfiSearch();
}

# ----------------------------------------------------------------------------
=item doTextBrowse( $params )
=cut
# ============================================================================
```

```
sub doTextBrowse($)
{
	my $params = shift;
	my $mediaId = $params->{'PAR'}{'MEDIUM'}
				|| $gDBCurrSection->{'MEDIA_ID'};

	my $tmpl = ConTemplate->new( $tmplResultList );
	$tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
	my $form = ConSearch->new( $params, $tmpl );

	$form->printTextSearchResult(
		$params->{'POS'}{'TEXTBROWSE'},
		&cgi2ANSI($params->{$tkPARAM}{'START_POS'}),
		$params->{$tkPARAM}{'SUCHCATID'},
		$params->{$tkPARAM}{'PREISMIN'},
		$params->{$tkPARAM}{'PREISMAX'}
	);
}

# ---------------------------------------------------------------------
=item doSaleInit( $params )
=cut
# =====================================================================

sub doSaleInit($)
{
	&logError("illegaler Aufruf eines eliminierten EVENT SALEINIT");
}

# ---------------------------------------------------------------------
=item doSpecialInit( $params )
=cut
# =====================================================================

sub doSpecialInit($)
{
	&logError("illegaler Aufruf von eliminierten EVENT SPECIALINIT");
}

# ---------------------------------------------------------------------
=item doBrowseInit( $params )
=cut
# =====================================================================

sub doBrowseInit($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplBrowseFrame );

	my $catId = $params->{'PAR'}{'CATID'};
	my $icon = $params->{'PAR'}{'ICON'};
	my $mediaId = $params->{'PAR'}{'MEDIUM'}
				|| $gDBCurrSection->{'MEDIA_ID'};
	my $pageId = $params->{'PAR'}{'PAGEID'};

	if( !$pageId ) {
		my $sql = <<EOS;
			select
				SEITEN_ID
			from
				KATEGORIEN_ABSAETZE
			where
				KATEGORIE_ID = $catId
			group by
				KATEGORIE_ID
			having
				LIST_POS = min( LIST_POS )
EOS
		my $result = $conDB->query( $sql );
		$pageId = $result->[0][0] if $#{$result} >= 0;
	}

	$tmpl->addSubstRef( {'CATID' => $catId} ) if $catId;
```

```
        $tmpl->addSubstRef( {'ICON' => $icon} ) if $icon;
        $tmpl->addSubstRef( {'PAGEID' => $pageId} ) if $pageId;
        $tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
        $tmpl->doTagSubstitution();
        $tmpl->doCleanup();
        $tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item doBasketInit( $params, $catCount )
=cut
# ========================================================================

sub doBasketInit($;$)
{
        my $params = shift;
        my $catCount = shift || 0;

        my $tmpl = ConTemplate->new( $tmplBasket );
        my $form = ConBasket->new( $params, $tmpl );

        $form->addCatalog( $catCount );
        $form->showBasket();
}

# ------------------------------------------------------------------------
=item doBasketAdd( $params )
=cut
# ========================================================================

sub doBasketAdd($)
{
        my $params = shift;

        my $tmpl = ConTemplate->new( $tmplProductPage );
        my $form = ConShow->new( $params, $tmpl );

        if( $form->checkForm() ) {
                $form = ConBasket->new( $params, ConTemplate->new( $tmplBasket ) );
                $form->actualizeBasket(0);
                $form->addCatalog();
                $form->showBasket();
        };
}


# ------------------------------------------------------------------------
=item doNewOrder( $params )
=cut
# ========================================================================

sub doNewOrder($)
{
        my $params = shift;

        my $tmpl = ConTemplate->new( $tmplBasket );
        my $form = ConBasket->new( $params, $tmpl );

        $form->actualizeBasket(0);
        $form->addCatalog();
        $form->showBasket();
}

# ------------------------------------------------------------------------
=item doBasketRefine( $params )
=cut
# ========================================================================

sub doBasketRefine($)
{
        my $params = shift;

        my $tmpl = ConTemplate->new( $tmplBasket );
        my $form = ConBasket->new( $params, $tmpl );
```

```
	if( $form->actualizeBasket(0) ) {
		$form->showBasket();
	};
}

# ------------------------------------------------------------------------
=item doBasketDelete( $params )
=cut
# ========================================================================

sub doBasketDelete($$)
{
	my $event = shift;
	my $params = shift;

	if($event =~ /(\d+)/) {
		$params->{PAR}{"A" . $1} = 0;
	}

	&doBasketRefine($params);
}


# ------------------------------------------------------------------------
=item doOrderForm( $params )
=cut
# ========================================================================

sub doOrderForm($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplBasket );
	my $form = ConBasket->new( $params, $tmpl );

	if( $form->actualizeBasket(1) ) {
		$form->{'tmpl'} = ConTemplate->new( $tmplOrderForm );
		$form->initOrderForm();
		$form->showBasket();
	};
}

# ------------------------------------------------------------------------
=item doSendOrder( $params )
=cut
# ========================================================================

sub doSendOrder($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplOrderForm );
	my $form = ConBasket->new( $params, $tmpl );

	if( $form->checkForm() ) {
		$form->{'tmpl'} = ConTemplate->new( $tmplOrderConfirm );
		if($form->preAction()) {
			$form->postAction();
		}
	};
}

# ------------------------------------------------------------------------
=item doSendOrderNoCheck( $params )
=cut
# ========================================================================

sub doSendOrderNoCheck($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplOrderConfirm );
	my $form = ConBasket->new( $params, $tmpl );
```

```
	if($form->preAction()) {
		$form->{'tmpl'} = ConTemplate->new( $tmplOrderConfirm );
		$form->postAction();
	}
}

# ------------------------------------------------------------------------
=item doCatSearch( $params )
=cut
# ========================================================================

sub doCatSearch($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplCatSearch );
	my $form = ConSearchForm->new( $params, $tmpl );
	$form->fillSearchForm();
	$tmpl->doCleanup();
	$tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item doBrowseList( $params )
=cut
# ========================================================================

sub doBrowseList($)
{
	my $params = shift;
	my $catId = $params->{'PAR'}{'CATID'};
	my $icon = $params->{'PAR'}{'ICON'};
	my $mediaId = $params->{'PAR'}{'MEDIUM'} || $gDBCurrSection->{'MEDIA_ID'};

	my $tmpl = ConTemplate->new( $tmplResultList );
	$tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
	my $form = ConSearch->new( $params, $tmpl );
	$form->doCatBrowseStart( $catId, $icon );
}

# ------------------------------------------------------------------------
=item doShowPage( $params )
=cut
# ========================================================================

sub doShowPage($)
{
	my $params = shift;

	if( !$params->{'PAR'}{'PAGEID'} ) {
		my $tmpl = ConTemplate->new( $htmlEmptyProcuct );
		$tmpl->printTemplate();
		return;
	}

	my $tmpl = ConTemplate->new( $tmplProductPage );
	my $form = ConShow->new( $params, $tmpl );

	$form->showProductPage();
}

# ------------------------------------------------------------------------
=item doShowBigPict( $params )
=cut
# ========================================================================

sub doShowBigPict($) {
	my $params = shift;
	my $tmpl = ConTemplate->new( $tmplBigPict );
	$tmpl->addSubstRef( {
		'PICREF' => $params->{'PAR'}->{'PICREF'}
		});
	$tmpl->doTagSubstitution();
```

```
	$tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item doProductSearchInit( $params )
=cut
# ========================================================================

sub doProductSearchInit($)
{
	my $params = shift;

	my $tmpl = ConTemplate->new( $tmplSeachInitFrame );
	my $searchQuery = &cgi2ANSI( $gRequest );
	my $refineQuery = $searchQuery;
	$searchQuery =~ s/TK_EV\[.*?\]/TK_EV[SEARCH]/ig;
	$refineQuery =~ s/TK_EV\[.*?\]/TK_EV[REFINE]/ig;
	$tmpl->setSubstRef( {
		'SEARCH_QUERY' => $searchQuery,
		'REFINE_QUERY' => $refineQuery
	} );
	$tmpl->doTagSubstitution();
	$tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item doSearchRefineInit( $params, $doOpen )
=cut
# ========================================================================

sub doSearchRefineInit($$)
{
	my $params = shift;
	my $doOpen = shift;

	my $tmpl = ConTemplate->new( $tmplRefineInitFrame );
	my $refineQuery = &cgi2ANSI( $gRequest );
	$refineQuery =~
		s/CATINIT/CAT/ig;
	$tmpl->setSubstRef( {
		'REFINE_QUERY' => $refineQuery
	} );
	$tmpl->doTagSubstitution();
	$tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item doSearchRefine( $params, $redraw )
=cut
# ========================================================================

sub doSearchRefine($$)
{
	my $params = shift;
	my $redrawMode = shift;

	my $posParam = &cgi2ANSI( ($redrawMode == 0
		? $params->{$tkPARAM}{'OFFEN'}
		: $params->{'POS'}{ ($redrawMode < 0 ? 'CLOSECAT' : 'OPENCAT' ) }
	) );

	my $left;
	my $right;

	($left, $right) = split /,/, $posParam;

	my $tmpl = ConTemplate->new( $tmplCatSearch );
	my $form = ConSearchForm->new( $params, $tmpl );

	$form->fillSearchForm( $redrawMode, $left, $right );
	$tmpl->doCleanup();
	$tmpl->printTemplate();
}
```

```
# ---------------------------------------------------------------------
=item doProductSearch( $params )
=cut
# =====================================================================

sub doProductSearch($)
{
	my $params = shift;
	my $mediaId = $params->{'PAR'}{'MEDIUM'} || $gDBCurrSection->{'MEDIA_ID'};

	my $tmpl = ConTemplate->new( $tmplResultList );
	$tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
	my $form = ConSearch->new( $params, $tmpl );
	$form->doSearch();
}

# ---------------------------------------------------------------------
=item doBrowseCatList( $params )
=cut
# =====================================================================

sub doBrowseCatList($)
{
	my $params = shift;
	my $mediaId = $params->{'PAR'}{'MEDIUM'} || $gDBCurrSection->{'MEDIA_ID'};

	my $tmpl = ConTemplate->new( $tmplResultList );
	$tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
	my $form = ConSearch->new( $params, $tmpl );
	$form->doBrowseCont( $params->{'POS'}{'CATBROWSE'} );
}

# ---------------------------------------------------------------------
=item doBrowseSearchResult( $params )
=cut
# =====================================================================

sub doBrowseSearchResult($)
{
	my $params = shift;
	my $mediaId = $params->{'PAR'}{'MEDIUM'} || $gDBCurrSection->{'MEDIA_ID'};

	my $tmpl = ConTemplate->new( $tmplResultList );
	$tmpl->addSubstRef( {'MEDIUM' => $mediaId} );
	my $form = ConSearch->new( $params, $tmpl );
	$form->printSearchResult(
		$params->{$tkPARAM}{'START_POS'},
		$params->{'POS'}{'SEARCHBROWSE'},
		$params->{$tkPARAM}{'END_POS'}
	);
}

# ---------------------------------------------------------------------
=item doWarnLeaving( $params )
=cut
# =====================================================================

sub doWarnLeaving($)
{
	my $params = shift;
	my $refPar = &cgi2ANSI( $params->{$tkPARAM}{'REF'} );

	my $tmpl = ConTemplate->new( $tmplWarnLeave );
	if( $params->{$tkPARAM}{'TOP'} ) {
		$tmpl->setSubstRef( {
			'REF' => $refPar,
			'DEST'=> $refPar,
			'TOP' => '1'
		} );
	}
	else {
		$tmpl->setSubstRef( {
			'REF' => $gSelf,
			'DEST' => $refPar,
```

```
                    'TOP' => ''
                } );
        }

        $tmpl->doTagSubstitution();
        $tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item doExitShop( $params )
=cut
# ========================================================================

sub doExitShop($)
{
        my $params = shift;
    my $refPar = &cgi2ANSI( $params->{$tkPARAM}{'REF'} );

        my $tmpl = ConTemplate->new( $tmplExitFrame );
        $tmpl->setSubstRef( {
                'REF' => $refPar
        } );
        $tmpl->doTagSubstitution();
        $tmpl->printTemplate();
}

sub doShowStatic($)
{
        my $params = shift;
        my $fn = $params->{$tkPARAM}{'URL'};
        my $base = $params->{$tkPARAM}{'URLBASE'};

        my $tmpl = ConTemplate->new( $fn, $base );
        $tmpl->doTagSubstitution();
        $tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item doGotCookie( $params )
=cut
# ========================================================================

sub doGotCookie($)
{
        my $params = shift;
        my $where = shift;
        my $tmplName;

        my $refPar = &cgi2ANSI( $params->{$tkPARAM}{'REF'} );

        if($where) {
                $tmplName = $params->{$tkPARAM}{'TOP'} ?
                                                        $tmplExitWithCookie :
$tmplExitFrame;
        } else {
                $tmplName = $tmplGotCookie;
        }
        my $tmpl = ConTemplate->new( $tmplName );
        $tmpl->setSubstRef( {
                'REF' => $refPar
        } );


        $tmpl->doTagSubstitution();
        $tmpl->printTemplate();
}

# ------------------------------------------------------------------------
=item newUserId()
=cut
# ========================================================================

sub newUserId()
{
```

```
        return &uniqueId();
}

# ------------------------------------------------------------------------
=item printConExpireDate()
=cut
# ========================================================================

sub getConExpireDate($)
{
        my $event = shift;

        my $expireHours =  $eventExpireHash->{ $event };

        if (defined $expireHours) {
                return (&getExpireDate(time(), $expireHours*3600) . "\n")
        }
        #       return (&getExpireDate(time(), 24*3600) . "\n");
   return "";
}

# ------------------------------------------------------------------------
=item main()
=cut
# ========================================================================

sub main() {

        &parseInput( *par );
        $gRequest = $par;
        my $params = &scanParams( \%par );

        my $event = (keys %{$params->{'EV'}})[0];
        my $isCatSearch;

        print "Content-type: text/html\n";
        print &getConExpireDate($event);

        &ConDB::conDBInit();

        if( $event =~ /^GETCOOKIE(.*)$/ )  {
                my $where = $1;
                if( !(  $gUserId = &getCookie( $gUserCookieName ) ) ) {
                        if( ! ($gUserId = $params->{$tkPARAM}{'USER_ID'}) ) {
                                $gUserId = &newUserId();
                        }
                        print &setCookieString( $gUserCookieName, $gUserId );
                }
                print "\n";
                &doGotCookie( $params, $where );

        }
        elsif( $event =~ /START$/ )  {

                if( !( $gUserId = &getCookie( $gUserCookieName ) ) ) {
                        $gUserId = &newUserId();
                }
                print "\n";
                &doStartFrame($params, $event);

        }
        else {
                $gUserId = $params->{$tkPARAM}{'USER_ID'};
                print "\n";

                if( $event eq 'NAVITEST' )          { &doNaviTest( $params ) }
                elsif( $event eq 'CATINIT' ) { &doCatInit( $params ) }
                elsif( $event eq 'TEXTINIT' ){ &doTextInit( $params ) }
                elsif( $event eq 'TEXTFORM' ){ &doTextSearchForm( $params ) }
                elsif( $event eq 'TEXTSEARCH' )      { &doTextSearch( $params ) }
                elsif( $event eq 'PROFIFORM' )       { &doProfiSearchForm( $params ) }
                elsif( $event eq 'PROFISEARCH' ){ &doProfiSearch( $params ) }
                elsif( $event eq 'TEXTBROWSE' )      { &doTextBrowse( $params ) }
                elsif( $event eq 'ADDORDER' ){ &doBasketAdd( $params ) }
```

```
                elsif( $event eq 'BASKETINIT' ) { &doBasketInit( $params ) }
                elsif( $event eq 'CATORDERINIT' ) { &doBasketInit( $params, 1 ) }
                elsif( $event eq 'NEWORDERINIT' ) { &doNewOrder( $params ) }
                elsif( $event eq 'BASKETREFINE' ) { &doBasketRefine( $params ) }
                elsif( $event =~ /BASKETDELETE/) { &doBasketDelete( $event, $params ) }
                #Saleinit und SpecialInit koennen raus
#               elsif( $event eq 'SALEINIT' ){ &doSaleInit( $params ) }
#               elsif( $event eq 'SPECIALINIT' ){ &doSpecialInit( $params ) }
                elsif( $event eq 'BROWSEINIT' ){ &doBrowseInit( $params ) }
                elsif( $event eq 'CATSEARCH' )      { &doCatSearch( $params ) }
                elsif( $event eq 'BROWSELIST' ){ &doBrowseList( $params ) }
                elsif( $event eq 'CATBROWSE' )      { &doBrowseCatList( $params ) }
                # SEARCHBROWSE, SEARCHINT,  kann raus REFINE wahrscheinlich auch
                elsif( $event eq 'SEARCHBROWSE' ){ &doBrowseSearchResult( $params ) }
                elsif( $event eq 'SHOWPAGE' ){ &doShowPage( $params ) }
                elsif( $event eq 'SHOWBIGPICT' )    { &doShowBigPict( $params ) }
                elsif( $event eq 'SEARCHINIT' )     { &doProductSearchInit( $params ) }
                elsif( $event eq 'OPENCATINIT' ){ &doSearchRefineInit( $params, 1 ) }
                elsif( $event eq 'CLOSECATINIT' ){ &doSearchRefineInit( $params, 0 ) }
                elsif( $event eq 'SEARCH' )         { &doProductSearch( $params ) }
                elsif( $event eq 'OPENCAT' ) { &doSearchRefine( $params, 1 ) }
                elsif( $event eq 'CLOSECAT' ){ &doSearchRefine( $params, -1 ) }
                elsif( $event eq 'REFINE' )         { &doSearchRefine( $params, 0 ) }
                elsif( $event eq 'ORDER' )          { &doOrderForm( $params ) }
                elsif( $event eq 'SENDORDER' )      { &doSendOrder( $params ) }
                elsif( $event eq 'SENDNOCHECK') { &doSendOrderNoCheck( $params ) }
                elsif( $event eq 'WARNLEAVE' )      { &doWarnLeaving( $params ) }
                elsif( $event eq 'EXITSHOP' ){ &doExitShop( $params ) }

                elsif( $event eq 'JUBELINIT' )      { &doJubelInit( $params ) }
                elsif( $event eq 'JUBELSUBMIT' ){ &doJubelSubmit( $params ) }

                elsif( $event eq 'ASSINIT' ){ &doAssInit( $params ) }
                elsif( $event eq 'ASSNAVI' ){ &doAssNavi( $params ) }
                elsif( $event eq 'ASSNAVI_NEW' ){ &doAssNaviNew( $params ) }
                elsif( $event eq 'ASSLIST' ){ &doAssList( $params ) }
                elsif( $event eq 'ASSPAGE' ){ &doAssShowPage( $params ) }

                elsif( $event eq 'STATICINIT' ){ &doShowStatic( $params ) }

                elsif( $event eq 'SHOPINIT' ){ &doShoppingInit( $params ) }
                elsif( $event eq 'ADDNOTEPAD' ){ &doNotePadAdd( $params ) }
                elsif( $event eq 'CATBROWSENEW' ){ &doCatBrowseNew( $params ) }
                elsif( $event eq 'CATLISTNEW' ){ &doCatListNew( $params ) }
#               elsif( $event eq 'BASKETSTATE' ){ &doBasketStateEmpty( $params ) }
                elsif( $event eq 'BASKETSTATE' ){ &doBasketState( $params ) }
                elsif( $event eq 'ADDORDER_NEW' ){ &doBasketAddNew( $params ) }

#               elsif( $event eq 'GAMEINIT' ){ &doGameInit( $params ) }
#               elsif( $event eq 'GAMENEW' ) { &doGameAnswer( $params,1 ) }
#               elsif( $event eq 'GAMEANSWER' )     { &doGameAnswer( $params ) }
#               elsif( $event eq 'GAMEQUESTION' ){ &doGameQuestion( $params ) }
#               elsif( $event eq 'GAMEENTER' )      { &doGameEnter( $params ) }
#               elsif( $event eq 'GAMELIST' ){ &doGameList( $params ) }
#               elsif( $event eq 'GAMEGIVEUP' )     { &doGameGiveup( $params ) }
        }

        &logEvent2Cyberbrand($event, $params, $gUserId);
        $conDB->close();
}

$SIG{'USR1'} = \&conDumpStack;
sub conDumpStack {
        my $sig=@_;
        &Carp::confess("caught SIG$sig!");
}

&main();

1;
```

Anhang D – Auszug aus der Artikeldatenbank

pos_name	lay_header_06	lay_header_05	lay_header_04	lay_header_03	lay_header_02	lay_header_01	lay_text_01	lay_ftext_01	pos_unit	pos_header	pos_text	pos_acc01
282960		conrad.de	Kommunikation	ISDN Technik	ISDN-Bundles				ST	Auch der Anrufbeantworter und das Fax... SET 00 Schnurloses ISDN-Komfortte	Leistungsmerkmale:Mobilteil:siehe SET 53, Leistungsmerkmale Basisstation wie SET 53 zusätzlich:Anschluß für 2 zusätzliche Geräte, z.B. Fax, ...	
282960		conrad.de	Kommunikation	ISDN Technik	ISDN-Bundles				ST	ISDN-Set 58	DeTeWe Eurix 245*<P> - Alle ISDN-Leistungsmerkmale im Mobilteil verfügbar - Telefonbuch für 40 Einträge mit Namen pro Mobilteil ...	
				Technik	Bundles					245	Leistungsmerkmale wie Eurix 240, zusätzlich<P>-Anschluß für 2 zusätzliche analoge Geräte z.B. Fax/Anrufbeantworter/Modem usw. - Programmierung der Geräte über das Mobilteil - Gebührenerfassung für alle angeschlossenen	

pos_name	lay_header_06	lay_header_05	lay_header_04	lay_header_03	lay_header_02	lay_header_01	lay_text_01	lay_ftext_01	pos_unit	pos_header	pos_text	pos_acc01
											Geräte	
282979		conrad.de	Kommunikation	ISDN Technik	ISDN-Bundles				ST	ISDN pur - ganz ohne Schnur... SET 53 Schnurloses ISDN-Telefon Eurix 240 p	Sämtliche Bedienschritte werden klar im Display beschrieben. Eine auf dem Mobilteil vorhandene “Hilfe”-Taste ist mit einer Notrufnummer programmierbar. Laut- und Leise-Tasten	
282979		conrad.de	Kommunikation	ISDN Technik	ISDN-Bundles				ST	DeTeWe Eurix 240	<B>DeTeWe<P>Eurix 240</B><P>- Telefonbuch für 40 Einträge mit Namen pro Mobilteil - Hilfetaste am Mobilteil - Alle ISDN-Leistungsmerkmale im Mobilteil verfügbar - ...	281786 <TAB>
283053		conrad.de	Kommunikation	Analog Telefontechnik	Anrufbeantworter				ST	Anrufbeantworter Zettler Zet-Com 130	Dieser Anrufbeantworter besitzt 2 Kassettenlaufwerke, getrennt in je 1 Laufwerk für die Ansagen und 1 Laufwerk für die Nachrichten. Die Bandaufzeichnung eignet sich ausgezeichnet, um Gespräche zu ...	281646 <TAB> Eine kleiner Reserve schadet nie!

Anhang E – Sourcen zum selbstentwickelten Prototypen

firststepsearch.pl

```
#! d:\perl\bin\perl.exe

use Win32::ODBC;

print "Content-type: text/html\n\n";
print "<html><head><title>CGI-Feedback</title></head>\n";

$daten = $ENV{'QUERY_STRING'};

undef $suchwort;
undef $DSN;

print "<body>\n";

@form = split(/&/, $daten);
foreach $feld (@form)
 {
  ($name, $value) = split(/=/, $feld);
  $value =~ tr/+/ /;
  $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
  $value =~ s/<!--(.|\n)*-->//g;
  $form_data[$i] = $name;
  $i = $i + 1;
  $form_data[$i] = $value;
  $i = $i + 1;
        if ($name eq "suchwort")
        {
                $suchwort = $value;
        }

        if ($name eq "DSN")
        {
                $DSN = $value;
        }

        if ($name EQ "phonet")
        {
                $phonet = $value;
        }

        if ($name EQ "phoneticsearch")
        {
                $searchalgo = $value;
        }
 }

if ($suchwort eq "") { $suchwort="acku"; };

if ($DSN eq "") { $DSN = "efeu"; };

    print "\n\tOpening ODBC connection for \"$DSN\"...\n<br>";
    if (!($db = new Win32::ODBC($DSN)))
    {
        print "Failure. \n\n";
        $Failed{'odbc-connection'} = "new(): " . Win32::ODBC::Error();
        exit();
    }
    else
    {
        print "Success (connection #", $db->Connection(), ")\n\n";
    }


       if (! $db->Sql("SELECT wort, wort_id FROM schlagwort where wort=\'$suchwort\'"))
       {
                while($db->FetchRow())
```

```
                {
                        undef %data;
                        %data = $db->DataHash();

                        $wort_id = $data{"wort_id"};
                }

                $sql = "SELECT gruppe_name, level1, gruppe_id FROM suchtabelle WHERE
wort_id=$wort_id ORDER BY level1";
        }
        else
        {
                $error = $db->Error();
                print "SQL failed ... $error";
        }

        if ($wort_id EQ "")
        {
                # if $wort_id is empty -> start phonetic search ;)

                print "<br><h1>phonetic search<BR>";

                if ($searchalgo eq "soundex")
                {
                        print "using SoundEx - Algorithm</h1><br>";

                        $soundex = SoundEx($suchwort);

                        if (length($soundex) < 2)
                        {
                                $soundex = 0;
                        }
                        print "<b>phonical [ SoundEx ] representation of $suchwort is
$soundex.</b><br>";

                        if ($soundex EQ 0)
                        {
                                print "<b>No Products found ... Sorry !</b>";
                                print "<br>Closing Connection: ", (($db->Close())?
"Successful.":"Failure."), "\n";

                                print "</body></html>\n";
                                exit;
                        }

                        $searchsql = "SELECT wort, wort_id FROM schlagwort where
soundex=$soundex";
                }
                else
                {
                        print "using Phonet - Algorithm</h1><br>";

                        print "<b>phonical [ Phonet ] representation of $suchwort is
$phonet.</b><br>";

                        $searchsql = "SELECT wort, wort_id FROM schlagwort where
phonet=\'$phonet\'"
                }

                print "<br>SearchSQL fired to DB:<br>$searchsql<br>";

                if (! $db->Sql($searchsql))
                {
                        $sql = "SELECT gruppe_name, level1, gruppe_id FROM suchtabelle
WHERE ";

                        while($db->FetchRow())
                        {
                                undef %data;
                                %data = $db->DataHash();

                                $wort = $data{"wort"};
                                $wort_id = $data{"wort_id"};
                                $sql = $sql . "wort_id=$wort_id OR ";
```

```
                              #print "<br>wort is: $wort<br>wort_id is:
$wort_id<br>soundex is: $sound_fromdb<br>";
                       }

                       $sql = substr($sql, 0, length($sql) - 3);
                       $sql = $sql . "ORDER BY level1";
                }
               else
               {
                       $error = $db->Error();
                       print "SQL failed ... $error";
               }


       }
       else
       {
               print "<br><h1>non phonetic search</h1>";
       }


       print "<br>SQL fired to DB:<br>\n";
       print "$sql<br>\n";


       if (! $db->Sql($sql))
       {
               @fields = $db->FieldNames();
               $Cols = $#fields + 1;

               # now creating the great magic ...

               $oldlevel = "init";

               print "<table width=\"50%\"><form target=\"left\" action=\"/cgi-
bin/scndstepsearch.pl\"><tr><td>\n";

               while($db->FetchRow())
               {
                       undef %data;
                       %data = $db->DataHash();

                       if ($oldlevel ne $data{"level1"})
                       {
                               if ($oldlevel ne "init") { print "</select><td><input
type=submit name=\"submit$oldlevel\" value=\"Auswahl\"></form><form target=\"left\"
action=\"/cgi-bin/scndstepsearch.pl\"><tr><td>\n"; };
                               $oldlevel = $data{"level1"};
                               print "<select name=\"$oldlevel\" size=1>\n";
                       }
                       $dummy = $data{"gruppe_name"};
                       $temp = $data{"gruppe_id"};
                       print "<option value=\"$temp\"> $dummy\n";
               }

               print "</select><td><input type=submit name=\"submit$oldlevel\"
value=\"Auswahl\"></td></form></table>\n";

       }
       else
       {
               $error = $db->Error();
               print "SQL failed ... $error";
       }



       print "<br>Closing Connection: ", (($db->Close())? "Successful.":"Failure."),
"\n";



print "</body></html>\n";
```

```
exit;

##############################################################################################
# author:      Michael Maretzke
# version:     1.01
# template:    SoundEx C-Implementation from Kevin Setter '97
# functionality:
#   SoundEx is a algorithm for phonetic search support. It's
#   used to transform entered words into a numeric
#   representation.
#   The implementation is based on the soundex algorithm
#   published in several algorithm books.
#
# call: $phonetic = SoundEx($word);
#
##############################################################################################
sub SoundEx
{
       my $SOUNDEX_LEN = 8;
       my $Input = $_[0];    # get first parameter from param stack
       my $Output;           # the phonetic representation will be stored here

       my $iIn = 0;          # counter for $Input string handling
       my $iOut = 0;         # counter for $Output string handling
       my $PrevDig = '*';    # flag to store last character meaning

       my $c;                # result to add to output string

       my $InputLen = length($Input);      # length of input string

       # work on every character in input string and output string is not longer
       # than $SOUNDEX_LEN characters
       while (($iIn  < $InputLen) && ($iOut < $SOUNDEX_LEN))
       {
              # transform character to lower character
              my $SingleInput = lc(substr($Input,$iIn,1));

              # decide what to do depends on character
              SWITCH: {
                            $SingleInput EQ 'b' && do { $c = '1'; last SWITCH;};
                            $SingleInput EQ 'p' && do { $c = '1'; last SWITCH;};
                            $SingleInput EQ 'f' && do { $c = '1'; last SWITCH;};
                            $SingleInput EQ 'v' && do { $c = '1'; last SWITCH;};
                            $SingleInput EQ 'c' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 's' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 'k' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 'g' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 'j' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 'q' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 'x' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 'z' && do { $c = '2'; last SWITCH;};
                            $SingleInput EQ 'd' && do { $c = '3'; last SWITCH;};
                            $SingleInput EQ 't' && do { $c = '3'; last SWITCH;};
                            $SingleInput EQ 'l' && do { $c = '4'; last SWITCH;};
                            $SingleInput EQ 'm' && do { $c = '5'; last SWITCH;};
                            $SingleInput EQ 'n' && do { $c = '5'; last SWITCH;};
                            $SingleInput EQ 'r' && do { $c = '6'; last SWITCH;};
                            $c = '*';
                     };

              if (($c NE $PrevDig) && ($c NE '*'))
              {
                     $Output = $Output . $c;
                     $PrevDig = $c;
                     $iOut++;
              }
              $iIn++;
       }

       if ($iOut < $SOUNDEX_LEN)
       {
              for ($iIn = $iOut; $iIn < $SOUNDEX_LEN; $iIn++)
              {
                     $Output[iIn] = '0';
```

```
                }
        }
        return $Output;
}
```

scndstepsearch.pl

```
#! d:\perl\bin\perl.exe

use Win32::ODBC;


print "Content-type: text/html\n\n";
print "<html><head><title>Search Results</title></head>\n";

$daten = $ENV{'QUERY_STRING'};

print "<body>\n";

@form = split(/&/, $daten);
foreach $feld (@form)
 {
  ($name, $value) = split(/=/, $feld);
  $value =~ tr/+/ /;
  $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
  $value =~ s/<!--(.|\n)*-->//g;
  $form_data[$i] = $name;
  $i = $i + 1;
  $form_data[$i] = $value;
  $i = $i + 1;

        if ( ! ($name =~ /submit/) )
        {
                $group = $value;
                print "group to search = $group";
                #print "$name//$value<br>";
        }
 }

if ($DSN eq "") { $DSN = "efeu"; };

    print "\n\tOpening ODBC connection for \"$DSN\"...\n<br>";
    if (!($db = new Win32::ODBC($DSN)))
    {
        print "Failure. \n\n";
        $Failed{'odbc-connection'} = "new(): " . Win32::ODBC::Error();
        exit();
    }
    else
    {
        print "Success (connection #", $db->Connection(), ")\n\n";
    }


        if (! $db->Sql("select bestellnummer from produkt_gruppe where gruppe_id =
$group"))
        {
                @fields = $db->FieldNames();
                while($db->FetchRow())
                {
                        undef %data;
                        %data = $db->DataHash();

                        $bestellnummer = $data{"bestellnummer"};
                        print "$bestellnummer<br>\n";
                }

        }
        else
        {
                $error = $db->Error();
                print "SQL failed ... $error";
```

```
        }



        print "<br>Closing Connection: ", (($db->Close())? "Successful.":"Failure."),
"\n";



print "</body></html>\n";
```

soundex.c

```
// Soundex code as implemented by Kevin Setter, 8/27/97
// Code reimplemented by Michael Maretzke to be used in a PERL
// environment

#include "SoundEx.h"

// Returns the soundex equivalent to pszInput
//void Soundex(char* pszInput, char** rpszOutput)
void SoundEx(char* pszInput, char** rpszOutput)
{
        int iIn = 0;
        int iOut = 0;
        char cPrevDig = '*';

        char c;

        while ((pszInput[iIn] != (char)NULL) && (iOut <= 4))
        {
                pszInput[iIn] = tolower(pszInput[iIn]);
                switch (pszInput[iIn])
                {
                        case 'b' : c = '1'; break;
                        case 'p' : c = '1'; break;
                        case 'f' : c = '1'; break;
                        case 'v' : c = '1'; break;
                        case 'c' : c = '2'; break;
                        case 's' : c = '2'; break;
                        case 'k' : c = '2'; break;
                        case 'g' : c = '2'; break;
                        case 'j' : c = '2'; break;
                        case 'q' : c = '2'; break;
                        case 'x' : c = '2'; break;
                        case 'z' : c = '2'; break;
                        case 'd' : c = '3'; break;
                        case 't' : c = '3'; break;
                        case 'l' : c = '4'; break;
                        case 'm' : c = '5'; break;
                        case 'n' : c = '5'; break;
                        case 'r' : c = '6'; break;
                        default : c = '*';
                }

                if ((c != cPrevDig) && (c != '*'))
                {
                        (*rpszOutput)[iOut] = c;
                        cPrevDig = (*rpszOutput)[iOut];
                        iOut++;
                }
                iIn++;
        }

        if (iOut < 4)
                for (iIn = iOut; iIn < 4; iIn++)
                        (*rpszOutput)[iIn] = '0';

        (*rpszOutput)[4] = NULL;
```

```
}
```

phonet.c

```
/*
 * phonet.c
 * --------
 *
 * Program for phonetic string conversion.
 *
 * Copyright (c):
 * 1999,2000:  Joerg MICHAEL, Adalbert-Stifter-Str. 11, 30655 Hannover, Germany
 *     and
 * (version 1.0) 1999:  Heise Verlag, Helstorfer Str. 7, 30625 Hannover, Germany
 *
 * SCCS: @(#) phonet.c  1.1  2000-01-10
 *
 * This program is subject to the GNU Library General Public License (LGPL)
 * as published by the Free Software Foundation; either version 2 of the
 * License, or (at your option) any later version.
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * You should have received a copy of the GNU Library General Public License
 * along with this program; if not, write to the
 * Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 *
 * Actually, the LGPL is __less__ restrictive than the better known GNU General
 * Public License (GPL). See the GNU Library General Public License or the file
 * COPYING.LIB for more details and for a DISCLAIMER OF ALL WARRANTIES.
 *
 * There is one important restriction: If you modify this program in any way
 * (e.g. add or change phonetic rules or modify the underlying logic or
 * translate this program into another programming language), you must also
 * release the changes under the terms of the LGPL.
 * That means you have to give out the source code to your changes,
 * and a very good way to do so is mailing them to the address given below.
 * I think this is the best way to promote further development and use
 * of this software.
 *
 * If you have any remarks, feel free to e-mail to:
 *     ct@ct.heise.de
 * The author's e-mail address is:  gerhard.michael@nienburg-weser.de
 * (this address is correct, and yes, my name is  Joerg Michael).
 */


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

void *malloc (size_t);

#include "ph_ext.h"
#include "phonet.h"


#define  TEST_CHAR    '\004'

/****  Macros for "phonet_init" and "run_mode":  ****/
#define  IS_INITIALIZED    1
#define  DO_TRACE          2
#define  CHECK_RULES       4


static int  run_mode = 0;
static int  last_rule_set = -SECOND_RULES;
static CHAR upperchar[256];
static int  isletter[256];
```

```
static void trace_info (CHAR text[], int n, CHAR err_text[])
/****  output trace info  ****/
{
  CHAR *s,*s2,*s3;
  s  = (phonet_rules[n] == NULL)  ?  (CHAR *) "(NULL)" : phonet_rules[n];
  s2 = (phonet_rules[n+1] == NULL) ? (CHAR *) "(NULL)" : phonet_rules[n+1];
  s3 = (phonet_rules[n+2] == NULL) ? (CHAR *) "(NULL)" : phonet_rules[n+2];

  printf ("%s %d:  \"%s\"%s\"%s\" %s\n", text, ((n/3)+1), s, s2, s3, err_text);
}



static int initialize_phonet (void)

/****  language dependant initializations  ****/
/****  resut:  0 : success                 ****/
/****         -1 : an error occured        ****/
{
  int i,k;
  CHAR *s;

  if (! (run_mode & IS_INITIALIZED))
    {
     /****  generate arrays "upperchar" and "isletter"  ****/
     run_mode = run_mode | IS_INITIALIZED;

     for (i=0; i<256; i++)
       {
       upperchar[i] = (islower (i)) ?  (CHAR) toupper(i) : (CHAR) i;
       isletter[i] = (isalpha (i)) ?  1 : 0;
       }
     for (i=0; umlaut[i] != '\0'; i++)
       {
       upperchar [umlaut[i]] = umlaut_upper[i];
       isletter [umlaut[i]] = 1;
       }
    }

  if (phonet_init != NULL  &&  phonet_hash != NULL  &&  phonet_rules != NULL)
    {
     *phonet_init = *phonet_init | IS_INITIALIZED;

     for (i=0; i< 2*HASH_COUNT; i++)
       {
       phonet_hash[i] = -1;
       }

     for (i=0; phonet_rules[i] != PHONET_END; i++)
       {
       s = phonet_rules[i];
       if (s != NULL  &&  i % 3 == 0)
         {
          /****  calculate hash value  ****/
          k = (int) *s;

          if (phonet_hash[k] < 0  &&  phonet_rules[i+1] != NULL)
            {
             /****  first rules  ****/
             phonet_hash[k] = i;
            }
          if (phonet_hash [k+HASH_COUNT] < 0  &&  phonet_rules[i+2] != NULL)
            {
             /****  second rules  ****/
             phonet_hash [k+HASH_COUNT] = i;
            }
         }
       }
     return (0);
    }
```

```
  return (-1);
}



static void string_prepare (CHAR *text, CHAR *s, CHAR *s2)

/****  Auxiliary function for "check_rules":  ****/
/****  "strcpy (text,s)" plus inclusion of    ****/
/****  'TEST_CHAR' and '-', if necessary      ****/
{
  if (*s != '\0')
    {
     *text = *s;
     text++;
     s++;
    }
  while (*s != '\0'  &&  ! isdigit (*s)
  &&  strchr ("-<^$", *s) == NULL)
    {
     *text = *s;
     text++;
     s++;
    }
  if (strchr (s2,'-') != NULL  ||  strchr (s2,'$') == NULL)
    {
     *text = TEST_CHAR;
     text++;
     *text = '-';
     text++;
    }
  strcpy (text, s);
}



int phonet (CHAR src[], CHAR dest[], int len, int mode)

/****  phonetic conversion                         ****/
/****  ("dest" == "src" is allowed).               ****/
/****  "len" = max. length of "dest" incl. '\0'.  ****/
/****  mode = <language> + FIRST_RULES   :         ****/
/****           Use <language> and first rules     ****/
/****  mode = <language> + SECOND_RULES  :         ****/
/****           Use <language> and second rules    ****/
/****  result:  >= 0 :  string length of "dest"    ****/
/****           < 0  :  an error occured           ****/
{
 int  i,j,k,n,p,z;
 int  k0,n0,p0,z0;
 CHAR c,c0,*s;
 CHAR *src_2,text[51];

 if (dest == NULL  ||  src == NULL  ||  len <= 0)
   {
    /****  wrong arg's  ****/
    if (run_mode & DO_TRACE)
      {
       printf ("Error: wrong arguments.\n");
      }
    return (-1);
   }

 /****  select language  ****/
 i = 0;
 if (mode != last_rule_set)
   {
    i = set_phonet_language (mode % SECOND_RULES);
    last_rule_set = mode;
   }
 if (i < 0)
   {
    s = "Warning: language not found, use current language";
```

```
    if (phonet_init == NULL
    ||  phonet_hash == NULL  ||  phonet_rules == NULL)
      {
       i = set_phonet_language (PHONET_DEFAULT_LANGUAGE);
       s = "Warning: language not found, use default language";
       if (i < 0)
         {
          s = "Error: language not found; default language could not be set";
         }
      }
    if (run_mode & DO_TRACE)
      {
       printf ("%s\n", s);
      }
    if (phonet_init == NULL  ||  phonet_hash == NULL  ||  phonet_rules == NULL)
      {
       strcpy (dest,"");
       return (-2);
      }
  }

if (phonet_init == NULL  ||  ! (*phonet_init & IS_INITIALIZED)
||  phonet_hash == NULL  ||  phonet_rules == NULL
||  ! (run_mode & IS_INITIALIZED))
  {
   /****  initialization  (must be done           ****/
   /****  BEFORE converting "src" to upper char)  ****/
   i = initialize_phonet();
   if (i < 0)
     {
      if (run_mode & DO_TRACE)
        {
         printf ("Error: initialization failed\n");
        }
      strcpy (dest,"");
      return (-3);
     }
  }

src_2 = text;
i = (int) strlen (src);
if (i > 50)
  {
   /****  "oversized" string  ****/
   src_2 = malloc ((size_t) (i+1));
   if (src_2 == NULL)
     {
      /****  "malloc" failed  ***/
      if (run_mode & DO_TRACE)
        {
         printf ("Error: \"malloc\" for %d Bytes failed.\n", i+1);
        }
      strcpy (dest,"");
      return (-4);
     }
  }

/****  "strcpy" plus conversion to upper char  ****/
i = 0;
while (src[i] != '\0')
  {
   src_2[i] = upperchar [src[i]];
   i++;
  }
src_2[i] = '\0';
src = src_2;

if (mode < SECOND_RULES)
  {
   mode = 0;
   s = "first";
  }
else
  {
```

```
    mode = HASH_COUNT;
    s = "second";
   }
if (run_mode & DO_TRACE)
   {
    printf ("\n\nphonetic conversion for  :  \"%s\"\n", src_2);
    printf ("(%s rules)\n", s);
   }

/****  check "src"  ****/
i = 0;
j = 0;
z = 0;
while ((c = src[i]) != '\0')
   {
    if (run_mode & DO_TRACE)
      {
       printf ("\ncheck position %d:  src = \"%s\",", j, src+i);
       printf ("  dest = \"%.*s\"\n", j, dest);
      }
    n = phonet_hash [(int) c + mode];
    z0 = 0;

    if (n >= 0)
      {
       /****  check all rules for this char  ****/
       while (phonet_rules[n] == NULL  ||  phonet_rules[n][0] == c)
        {
         if (phonet_rules [n] == NULL
         ||  phonet_rules [n+1+(mode/HASH_COUNT)] == NULL)
           {
            /****  no conversion rule  ****/
            n += 3;
            continue;
           }
         if (run_mode & DO_TRACE)
           {
            trace_info ("> rule no.", n, "is being checked");
           }

         /****  check whole string  ****/
         k = 1;   /****  no. of matching letters  ****/
         p = 5;   /****  default priority  ****/
         s = phonet_rules[n];
         s++;     /****  needed by "*(s-1)" below  ****/

         while (*s != '\0'  &&  src[i+k] == *s
         &&  ! isdigit (*s)  &&  strchr ("(-<^$", *s) == NULL)
           {
            k++;
            s++;
           }
         if (run_mode & CHECK_RULES)
           {
            /****  we do "check_rules"  ****/
            while (*s != '\0'  &&  src[i+k] == *s)
              {
              k++;
              s++;
              }
           }
         if (*s == '(')
           {
            /****  check an array of letters  ****/
            if (isletter [src[i+k]]
            &&  strchr (s+1, src[i+k]) != NULL)
              {
              k++;
              while (*s != '\0'  &&  *s != ')')
                {
                 s++;
                }
              if (*s == ')')
                {
```

```
            s++;
           }
        }
     }
   p0 = (int) *s;
   k0 = k;
   while (*s == '-'  &&  k > 1)
     {
      k--;
      s++;
     }
   if (*s == '<')
     {
      s++;
     }
   if (isdigit (*s))
     {
      /****  read priority  ****/
      p = *s - '0';
      s++;
     }
   if (*s == '^'  &&  *(s+1) == '^')
     {
      s++;
      if ((run_mode & CHECK_RULES)
      &&  ! isletter [src[i+k0]])
        {
        /****  we do "check_rules"  ****/
        s = s-2;
        }
     }

   if (*s == '\0'
   || (*s == '^'  &&  (i == 0  ||  ! isletter [src[i-1]])
    && (*(s+1) != '$'
    || (! isletter [src[i+k0]]  &&  src[i+k0] != '.')))
   || (*s == '$'  &&  i > 0  &&  isletter [src[i-1]]
    && (! isletter [src[i+k0]]  &&  src[i+k0] != '.')))
     {
      /****  look for continuation, if:         ****/
      /****  k > 1  and  NO '-' in first string ****/
      c0 = src [i+k-1];
      n0 = phonet_hash [(int) c0 + mode];

      if (k > 1  &&  n0 >= 0
      &&  p0 != (int) '-'  &&  src[i+k] != '\0')
        {
        /****  check continuation rules for "src[i+k]"  ****/
        while (phonet_rules[n0] == NULL
        ||  phonet_rules[n0][0] == c0)
          {
           if (phonet_rules [n0] == NULL
           ||  phonet_rules [n0+1+(mode/HASH_COUNT)] == NULL)
             {
              /****  no conversion rule  ****/
              n0 += 3;
              continue;
             }
           if (run_mode & DO_TRACE)
             {
              trace_info ("> > continuation rule no.",
                  n0, "is being checked");
             }

           /****  check whole string  ****/
           k0 = k;
           p0 = 5;
           s = phonet_rules[n0];
           s++;
           while (*s != '\0'  &&  src[i+k0] == *s
           &&  ! isdigit (*s)  &&  strchr("(-<^$",*s) == NULL)
             {
              k0++;
              s++;
```

```
    }
  if (*s == '(')
    {
     /****  check an array of letters  ****/
     if (isletter [src[i+k0]]
     &&  strchr (s+1, src[i+k0]) != NULL)
      {
       k0++;
       while (*s != '\0'  &&  *s != ')')
         {
          s++;
         }
       if (*s == ')')
         {
          s++;
         }
      }
    }
  while (*s == '-')
    {
     /****  "k0" is NOT decremented     ****/
     /****  because of  "if (k0 == k)"  ****/
     s++;
    }
  if (*s == '<')
    {
     s++;
    }
  if (isdigit (*s))
    {
     p0 = *s - '0';
     s++;
    }

  if (*s == '\0'
    /****  *s == '^' is not possible here  ****/
  || (*s == '$'  &&  ! isletter [src[i+k0]]
   &&  src[i+k0] != '.'))
    {
     if (k0 == k)
      {
       /****  this is only a partial string  ****/
       if (run_mode & DO_TRACE)
         {
          trace_info ("> > continuation rule no.",
             n0, "not used (too short)");
         }
       n0 += 3;
       continue;
      }

     if (p0 < p)
      {
       /****  priority is too low  ****/
       if (run_mode & DO_TRACE)
         {
          trace_info ("> > continuation rule no.",
             n0, "not used (priority)");
         }
       n0 += 3;
       continue;
      }
     /****  rule found; no further search  ****/
     break;
    }

  if (run_mode & DO_TRACE)
    {
     trace_info ("> > continuation rule no.",
        n0, "not used");
    }
  n0 += 3;
 } /****  end of "while"  ****/
```

```
              if (p0 >= p
              && (phonet_rules[n0] != NULL  &&  phonet_rules[n0][0] == c0))
                {
                 if (run_mode & DO_TRACE)
                   {
                    trace_info ("> rule no.", n,"");
                    trace_info ("> not used because of continuation",n0,"");
                   }
                 n += 3;
                 continue;
                }
              }

            /****  replace string  ****/
            if (run_mode & DO_TRACE)
              {
              trace_info ("Rule no.", n, "is applied");
              }
            s = phonet_rules [n+1+(mode/HASH_COUNT)];
            p0 = (phonet_rules[n][0] != '\0'
              &&  strchr (phonet_rules[n]+1,'<') != NULL) ?  1 : 0;

            if (p0 == 1  &&  z == 0)
              {
              /****  rule with '<' is applied  ****/
              if (j > 0  &&  *s != '\0'
              && (dest[j-1] == c  ||  dest[j-1] == *s))
                {
                 j--;
                }
              z0 = 1;
              z++;
              k0 = 0;
              while (*s != '\0'  &&  src[i+k0] != '\0')
                {
                 src[i+k0] = *s;
                 k0++;
                 s++;
                }
              if (k > k0)
                {
                 strcpy (src+i+k0, src+i+k);
                }
              if ((run_mode & CHECK_RULES)
              &&  (*s != '\0'  ||  k < k0))
                {
                 /****  we do "check_rules":            ****/
                 /****  replacement string is too long  ****/
                 dest[j] = '\0';
                 return (-200);
                }
              /****  new "current char"  ****/
              c = src[i];
              }
            else
              {
              if ((run_mode & CHECK_RULES)
              &&  p0 == 1  &&  z > 0)
                {
                 /****  we do "check_rules":      ****/
                 /****  recursion found -> error  ****/
                 dest[j] = '\0';
                 return (-100);
                }
              i = i+k-1;
              z = 0;
              while (*s != '\0'
              &&  *(s+1) != '\0'  &&  j < len-1)
                {
                 if (j == 0  ||  dest[j-1] != *s)
                   {
                    dest[j] = *s;
                    j++;
                   }
```

```
                  s++;
                 }
               /****  new "current char"  ****/
               c = *s;

               if (phonet_rules[n][0] != '\0'
               &&  strstr (phonet_rules[n]+1, "^^") != NULL)
                 {
                  if (c != '\0')
                    {
                     dest[j] = c;
                     j++;
                    }
                  strcpy (src, src+i+1);
                  i = 0;
                  z0 = 1;
                 }
               }

             break;
            }
          n += 3;
         }
       }

    if (z0 == 0)
      {
       if (j < len-1  &&  c != '\0'
       && (j == 0  ||  dest[j-1] != c))
         {
          /****  delete multiple letters only  ****/
          dest[j] = c;
          j++;
         }
       i++;
       z = 0;
      }
   }

 if (src_2 != text)
   {
    free (src_2);
   }
 dest[j] = '\0';
 return (j);
}



int check_rules (int language, int trace_only)

/****  Check all phonetic rules of the current    ****/
/****  language.                                  ****/
/****  ("trace_only" > 0:  trace this rule only)  ****/
/****  Result:  Number of errors                  ****/
{
 int  i,k,n,n0;
 int  errors = 0;
 CHAR *r,*r0,rule[35];
 CHAR *s,err_text[201];
 CHAR orig[35],orig2[35];
 CHAR text[35],text2[35];

 /****  initialization  ****/
 i = set_phonet_language (language);
 if (i >= 0)
   {
    i = initialize_phonet();
   }
 if (i < 0)
   {
    printf ("Error: initialization for language %d failed\n", language);
    return (-1);
```

```
  }

isletter [(int) TEST_CHAR] = 1;
run_mode = run_mode | CHECK_RULES;
i = 0;

while (phonet_rules[i] != PHONET_END)
  {
   /****  syntax check for all strings  ****/
   if ((i/3)+1 == trace_only)
     {
      run_mode = run_mode | DO_TRACE;
     }
   else if (trace_only > 0)
     {
      run_mode = run_mode & (~DO_TRACE);
     }

   strcpy (err_text,"");
   k = 0;
   if (i % 3 == 0  &&  (phonet_rules[i] == NULL
   || (phonet_rules[i+1] == NULL  &&  phonet_rules[i+2] == NULL)))
     {
      strcpy (err_text,"  Forbidden null pointer");
      k = -10;
     }

   if (k >= 0)
     {
      if (phonet_rules[i] == NULL)
       {
        i++;
        continue;
       }

      if (i % 3 == 0)
       {
        /****  check first letter  ****/
        s = phonet_rules[i];
        n = phonet_hash [(int) *s];
        if (i >= n+3  &&  n >= 0
        &&  *s != phonet_rules[i-3][0])
          {
           strcpy (err_text,"  Wrong first char");
           k = -10;
          }
        n = phonet_hash [(int) *s + HASH_COUNT];
        if (i >= n+3  &&  n >= 0
        &&  *s != phonet_rules[i-3][0])
          {
           strcpy (err_text,"  Wrong first char");
           k = -10;
          }

        if (k >= 0)
          {
           /****  check length of search string  ****/
           k = 0;
           while (*s != '\0'  &&  ! isdigit (*s)
           &&  strchr ("()<^$", *s) == NULL)
             {
             k++;
             s++;
             }
           if (k == 0)
             {
             strcpy (err_text,"  Search string is empty");
             if (*s != '\0'  &&  strchr ("()<^$", *s) == NULL)
               {
                strcpy (err_text,"  First char is meta char");
               }
             k = -10;
             }
          }
```

```
        }
      }

    if (k >= 0)
      {
       /****  syntax check for string  ****/
       k = 0;
       s = phonet_rules[i];
       n = 0;
       if (*s != upperchar [(int)*s])
        {
         /****  forbidden lower-case char  ****/
         k = -100;
        }
       if (i % 3 == 0  &&  *s != '\0')
        {
         s++;
         n++;
        }
       while (*s != '\0'  &&  k >= 0)
        {
         if (*s != upperchar [(int) *s])
           {
            /****  forbidden lower-case char  ****/
            k = -100;
            break;
           }
         if (*s == '(')
           {
            if (k >= 1  ||  ! isletter [*(s+1)])
              {
              k = -10;
              break;
              }
            s++;
            n++;
            while (isletter[*s])
              {
              s++;
              }
            if (*s != ')')
              {
              k = -10;
              break;
              }
            k = 1;
           }
         else if (*s == '-')
           {
            /****  "k > 2" is correct              ****/
            /****  (more than one '-' is allowed)  ****/
            n--;
            if (k > 2  ||  n <= 0)
              {
              k = -10;
              break;
              }
            k = 2;
           }
         else if (*s == '<')
           {
            if (k >= 3)
              {
              k = -10;
              break;
              }
            k = 3;
           }
         else if (isdigit (*s))
           {
            if (k >= 4)
              {
              k = -10;
              break;
```

```
            }
          k = 4;
        }
      else if (*s == '^')
        {
         if (k >= 5)
           {
           k = -10;
           break;
           }
         if (*(s+1) == '^')
           {
           s++;
           }
         k = 5;
        }
      else if (*s == '$')
        {
         if (k >= 6  ||  *(s+1) != '\0')
           {
           k = -10;
           break;
           }
         k = 6;
        }
      else if (k > 0  ||  *s == ')')
        {
         k = -10;
         break;
        }
      else
        {
         n++;
        }
      s++;
     }

   if (k > 0  &&  i % 3 != 0)
    {
     sprintf (err_text,"  Meta char in replacement string");
     k = -10;
    }
   else if (k <= -100)
    {
     sprintf (err_text,"  Lower-case letter in string");
    }
   else if (k < 0)
    {
     sprintf (err_text,"  Syntax error in search string");
    }
   else if ((int) strlen (phonet_rules[i]) > 30)
    {
     sprintf (err_text,"  String very long ( > 30 chars)");
     k = -1;
    }
   s = phonet_rules[i];

   if (k >= 0  &&  i % 3 == 0
   &&  n > 0  &&  strchr (s,'<') != NULL)
    {
     /****  check lengths of search and replacement string  ****/
     if ((phonet_rules[i+1] != NULL
      &&  strcmp (s,phonet_rules[i+1]) == 0)
     || (phonet_rules[i+2] != NULL
      &&  strcmp (s,phonet_rules[i+2]) == 0))
        {
         strcpy (err_text,"  Replacement string too long due to '<'");
         k = -10;
        }
     if ((phonet_rules[i+1] != NULL
      && (int) strlen (phonet_rules[i+1]) > n)
     || (phonet_rules[i+2] != NULL
      && (int) strlen (phonet_rules[i+2]) > n))
        {
```

```
            strcpy (err_text,"  Replacement string too long due to '<'");
            k = -10;
           }
       }
     }

  if (k < 0)
    {
     /****  output error message  ****/
     s = "Possible error in rule";
     if (k < -1)
      {
       s = "Error in rule";
      }
     trace_info (s, i-(i%3), err_text);
     errors++;
    }


  if (k >= 0  &&  i % 3 != 0)
    {
     /****  do phonetic conversion and check result  ****/
     n = i % 3;
     n0 = (i%3 == 1) ?  FIRST_RULES : SECOND_RULES;
     r = strchr (phonet_rules[i-n], '(');
     if (r == NULL)
      {
       /****  There is no regular expression in search string  ****/
       r = "  ";
      }
     r++;

     while (*r != ')'  &&  *r != '\0')
      {
       /****  Split regular expression (e.g. "GS(CH)--")  ****/
       /****  into simple rules and check each of them.   ****/
       r0 = phonet_rules[i-n];
       strcpy (rule, r0);
       phonet_rules[i-n] = rule;
       s = strchr (rule,'(');

       if (s != NULL)
         {
          *s = *r;
          s++;
          while (*s != ')'  &&  *s != '\0')
            {
            strcpy (s,s+1);
            }
          if (*s == ')')
            {
            strcpy (s,s+1);
            }
         }

       /****  do the check  ****/
       sprintf (orig, "%c%s", TEST_CHAR, phonet_rules[i-n]);
       sprintf (orig2, "%c%s", TEST_CHAR, phonet_rules[i]);

       if (strchr (phonet_rules[i-n],'^') != NULL)
         {
          sprintf (orig, orig+1);
          sprintf (orig2,orig2+1);
         }
       if (strchr (phonet_rules[i-n],'-') != NULL
       ||  strchr (phonet_rules[i-n],'$') == NULL)
         {
          sprintf (orig, "%s%c", orig, TEST_CHAR);
          sprintf (orig2,"%s%c", orig2,TEST_CHAR);
         }
       if (orig2[0] == orig2[1]  &&  orig2[2] == '\0')
         {
          /****  e.g. orig2 == "<TEST_CHAR><TEST_CHAR>"  ****/
          orig2[1] = '\0';
```

```
          }

        /****  check conversion result  ****/
        k = phonet (orig,text, 33,n0);
        if (k > -100)
          {
           k = phonet (orig2,text2, 33,n0);
          }

        if (k <= -100)
          {
           /****  error found  ****/
           phonet_rules[i-n] = r0;
           strcpy (err_text,"  Recursion found");
           if (k == -200)
             {
             strcpy (err_text,"  Replacement string too long due to '<'");
             }
           trace_info ("Error in rule", i-(i%3), err_text);
           errors++;
           break;
          }

        /****  second rule check  ****/
        if (strcmp (text,orig2) != 0)
          {
           string_prepare (err_text+80, rule,rule);
           string_prepare (err_text, orig,orig);

           phonet_rules[i-n] = err_text+80;
           (void) phonet (err_text, err_text+40, 33,n0);
           phonet_rules[i-n] = rule;
           err_text[0] = '\0';
           if (strcmp (err_text+40, orig2) == 0)
             {
             strcpy (text,orig2);
             }
          }

        if (strcmp (text2,orig2) != 0
        && ((strcmp (phonet_rules[i-n],"AVIER$") == 0  &&  n==1
         &&  strcmp (phonet_rules[i],"AWIE") == 0)
        || (strcmp (phonet_rules[i-n],"GH") == 0  &&  n == 1
         &&  strcmp (phonet_rules[i],"G") == 0)
        || (strcmp (phonet_rules[i-n],"HEAD-") == 0  &&  n == 1
         &&  strcmp (phonet_rules[i],"HE") == 0)
        || (strcmp (phonet_rules[i-n],"IERRE$") == 0
         &&  strcmp (phonet_rules[i],"IER") == 0)
        || (strcmp (phonet_rules[i-n],"IVIER$") == 0  &&  n == 1
         &&  strcmp (phonet_rules[i],"IWIE") == 0)
        || (strcmp (phonet_rules[i-n],"SHST") == 0  &&  n == 1
         &&  strcmp (phonet_rules[i],"SHT") == 0)))
          {
           /****  these are exceptions  ****/
           strcpy (text2, orig2);
          }

    #ifdef PHONET_GERMAN
        if (strcmp (text2,orig2) != 0
        &&  language == PHONET_GERMAN
        && ((strncmp (phonet_rules[i-n],"GEGEN",5) == 0  &&  n == 1
         &&  strncmp (phonet_rules[i],"GEGN",4) == 0)
        || (strcmp (phonet_rules[i-n],"GGF.") == 0  &&  n == 1
         &&  strcmp (phonet_rules[i],"GF.") == 0)
        || (strcmp (phonet_rules[i-n],"HAVEN7$") == 0  &&  n == 1
         &&  strcmp (phonet_rules[i],"HAFN") == 0)))
          {
           /****  exceptions in German  ****/
           strcpy (text2, orig2);
          }
    #endif

        if (strcmp (text2,orig2) != 0
        && (s = strchr (orig2,'I')) != NULL)
```

```
            {
             /****  extra check for replacement strings with an 'I'  ****/
             if (strchr (s+1,'I') != NULL)
               {
               /****  take second 'I', if found  ****/
               s = strchr (s+1,'I');
               }
             *s = 'J';
             (void) phonet (orig2,text2, 33,n0);
             *s = 'I';
            }

          /****  extra check for search strings with a '-'  ****/
          s = orig;
          k = 0;
          while (*s != '\0'  &&  ! isdigit (*s)
          &&  strchr ("-<^$",*s) == NULL)
            {
             s++;
             k++;
            }
          while (*s != '\0')
            {
             if (*s == '-')
               {
               k--;
               }
             s++;
            }

          if (strcmp (text2,orig2) != 0
           && ((strchr (orig,'-') != NULL  &&  k > 0)
          || (phonet_rules[i-n][0] == phonet_rules[i-n][1]
           &&  phonet_rules[i-n][0] == phonet_rules[i][0])
          || (strncmp (phonet_rules[i-n],"AI",2) == 0
           &&  phonet_rules[i][0] == 'E'
           &&  k > 1  &&  strncmp (s-2,"E$",2) == 0)))
            {
             s = orig + k;
             k = (int) strlen (orig2);
             if (k > 0)
               {
               if (orig2[k-1] == TEST_CHAR)
                 {
                  k--;
                 }
               strcpy (err_text+1, orig2);
               strcpy (err_text+1+k, s);
               k = 1;

               if (phonet_rules[i-n][0] == phonet_rules[i-n][1]
               &&  phonet_rules[i-n][0] == phonet_rules[i][0]
               &&  phonet_rules[i][1] == '\0')
                 {
                  /****  extra check for double letters  ****/
                  err_text[0] = TEST_CHAR;
                  err_text[1] = phonet_rules[i][0];
                  k = 0;
                 }
               if (phonet_rules[i-n][0] == 'H'
               &&  phonet_rules[i-n][1] != '\0'
               &&  phonet_rules[i-n][2] == 'H'
               &&  phonet_rules[i-n][1] == phonet_rules[i][0]
               &&  phonet_rules[i-n][2] == phonet_rules[i][1])
                 {
                  /****  special case "H?H"  ****/
                  err_text[0] = TEST_CHAR;
                  err_text[1] = 'H';
                  k = 0;
                 }
               if (strncmp (phonet_rules[i-n],"LV",2) == 0
               &&  strncmp (phonet_rules[i], "LW",2) == 0)
                 {
                  /****  special case "LV*"  ****/
```

```
                err_text[3] = 'V';
               }
             if (strncmp (phonet_rules[i-n],"AI",2) == 0
             &&  phonet_rules[i][0] == 'E')
               {
                /****  special case "AI*E$"  ****/
                err_text[0] = TEST_CHAR;
                err_text[1] = err_text[2];
                strcpy (err_text+2, phonet_rules[i]);
                k = 0;
               }

             (void) phonet (err_text+k, err_text+40, 33,n0);

             if (strcmp (err_text+40, orig2) != 0)
               {
                string_prepare (err_text+80, err_text+k,rule);
                string_prepare (err_text, rule,rule);

                phonet_rules[i-n] = err_text;
                (void) phonet (err_text+80, err_text+40, 33,n0);
                phonet_rules[i-n] = rule;
               }
             err_text[0] = '\0';
             if (strcmp (err_text+40, orig2) == 0)
               {
                strcpy (text2, orig2);
               }
             }
           }

         phonet_rules[i-n] = r0;

         if (strcmp (text, orig2) != 0
         ||  strcmp (text2,orig2) != 0)
           {
            orig[0] = '\0';
            if (*r != ' ')
              {
              sprintf (orig," for '%c'", *r);
              }
            sprintf (err_text, "  result %d%s: \"%s\"%s\"",
                n,orig, text,text2);

            /****  delete 'TEST_CHAR' from "error" string  ****/
            s = err_text;
            while (*s != '\0')
              {
              while (*s == TEST_CHAR)
                {
                 strcpy (s,s+1);
                }
              s++;
              }

            /****  output error message  ****/
            s = "Possible error in rule";
            if (strcmp (text,orig2) != 0)
              {
              s = "Error in rule";
              }
            trace_info (s, i-(i%3), err_text);
            errors++;
           }
         r++;
        }
      }
    i++;
   }

if (i % 3 != 0)
  {
   printf ("Error: string count is not a multiple of 3.\n");
   errors++;
```

```
   }
 isletter [(int) TEST_CHAR] = 0;
 run_mode = run_mode & (~CHECK_RULES);

 printf ("Language \"%s\":\n", phonet_language);
 printf ("Check of all phonetic rules:  ");

 if (errors == 0)
   {
    printf ("No syntax error or inconsistency found.\n");
   }
 else
   {
    printf ("%d errors have been found.\n\n", errors);
    printf ("Remarks:\n");
    printf ("a) The correct syntax for search strings is:\n");
    printf ("      <word> [<->..] [<] [<0-9>] [^[^]] [$]\n");
    printf ("   The end of \"word\" may contain as a simple regular expression\n");
    printf ("   one array of letters that must be enclosed in '(' and ')'.\n");
    printf ("b) Rules with a '<' demand that the replacement string may not\n");
    printf ("   be longer than the search string.\n");
    printf ("c) The placement of rules determines their priority.\n");
    printf ("   Therefore, the rules for \"SH\" must be placed before the rules\n");
    printf ("   for \"S\" (otherwise, a conversion error will occur for \"SH\").\n");
    printf ("d) Another common source of errors is ignorance of dependencies.\n");
    printf ("   For example, in German the replacement string \"NJE\" would be
wrong,\n");
    printf ("   because the 'J' is subject to another phonetic rule.\n");
   }
 return (errors);
}

int main (int argc, char *argv[])
{
       char* pszQuery;
       char* pszWord;
       char* pcAnd;
       int iCounter = 0;
       int iDebug = 0;

       CHAR *s,text[101];
       int  n=0,i=-1,r=-1;

       printf("Content-type: text/html\n\n");

       printf("<html><body>");

       pszQuery = getenv("QUERY_STRING");

       if (!pszQuery)
       {
              printf("<b>CGI-QueryString is empty !</b>");
              exit (1);
       }

       if (iDebug) printf("pszQuery is: %s\n<br>", pszQuery);

       pszWord = strstr(pszQuery, "suchwort");

       if (!pszWord)
       {
              printf("<b>CGI-Querystring contains no \"suchwort\" - Parameter !</b>");
              exit(1);
       }

       if (iDebug) printf("pszWord is: %s\n<br>", pszWord);

       pszWord = (strchr(pszWord, '=') + 1);

       if (iDebug) printf("pszWord is %s%\n<br>", pszWord);


       pcAnd = strchr(pszWord, (char) '&');
```

```
        if (iDebug) printf("pcAnd is %s%\n<br>", pcAnd);

        if (!pcAnd) iCounter = strlen(pszWord);
        else
        {
                if (iDebug) printf("in else part\n<br>");
                iCounter = (int) (pcAnd - pszWord);
                if (iCounter > 100) iCounter = 100;
        }

        if (iDebug) printf("iCounter is %d\n<br>", iCounter);


        pszWord = strncpy(text, pszWord, iCounter);

        text[iCounter] = '\0';

        //phonet (pszSrc, pszDest, 101, 10000);
        phonet (text, text, 101, SECOND_RULES);

        if (iDebug) printf("phonetic representation of %s is: %s\n<br>", text, text);
        if (!iDebug) printf("<html><meta http-equiv=\"refresh\" content=\"0;
URL=http://127.0.0.1/cgi-bin/firststepsearch.pl?%s&phonet=%s\"></html>", pszQuery,
text);
        if (iDebug) printf("<html>URL=http://127.0.0.1/cgi-
bin/firststepsearch.pl?%s&phonet=%s\"><br>", pszQuery, text);
        printf("</html>");

  return (0);
}
```