Ewald Brochhausen, Jürgen Kielisch,
Jürgen Schnerring, Jens Staeck

mySAP® HR – Technische Grundlagen und Programmierung

Galileo Press

Inhalt

Vorwort

Seit der ersten Installation eines R/3 HR sind mittlerweile mehr als dreizehn Jahre vergangen. Während dieser Zeit wurde vieles weiterentwickelt und viel Funktionalität ergänzt. Ständig steigende Anforderungen an eine Human-Resources-Software führten zu Erweiterungen des Standards in Form von Customizing-Möglichkeiten und neuen Werkzeugen. Waren am Anfang oft noch Modifikationen des Systems erforderlich, um die spezifischen Besonderheiten der Unternehmen zu berücksichtigen, wurden im Lauf der Entwicklung immer mehr Möglichkeiten geschaffen, Kundenanpassungen gegenüber Änderungen im Standard zu schützen. Beispiele hierfür sind die Möglichkeiten, die Datenbasis beliebig zu ergänzen, die Standardbilder zu erweitern oder ABAP-Coding modifikationsfrei zu ersetzen oder hinzuzufügen.

Mit der Ausweitung der Anpassungsmöglichkeiten wurde es manchmal auch schwieriger, zu entscheiden, welche Vorgehensweise im konkreten Fall aus den verschiedenen Alternativen zu wählen ist. Um hierbei Hilfestellung zu geben, haben wir – auch auf Anregung vieler Anwender des SAP HR hin – die wichtigsten Grundlagen des HR in diesem Buch zusammengefasst. Hierbei geht es vor allem um die Besonderheiten des HR, weniger um allgemeine anwendungsübergreifende Grundlagen des SAP-Systems. Für die Nutzung dieses Buches sollten daher bereits grundlegende Kenntnisse der ABAP-Programmierung und von SAP HR vorhanden sein. Hierzu sei auf die im Anhang erwähnten Bücher verwiesen.

Wir freuen uns, dass der Verlag nach zwei Jahren die Neuauflage dieses Buches beschlossen hat. Der Erfolg der ersten Auflage gibt uns in unserem Bemühen Recht, zu einem spezifischen Segment der SAP-Software ein eigenes Programmier- und Technologie-Buch zu schreiben.

Es bleibt uns noch die angenehme Aufgabe des Dankens. Unser Vorhaben wurde von einer Vielzahl von Freunden und Kollegen unterstützt. Ihnen allen sei für ihren wertvollen Rat und ihre Hilfe gedankt. In diesem Zusammenhang seien Ulf Bangert, Klaus Billig, Bärbel Bohr, Udo Klein, Heiko Schultze und Matthias Wengner besonders erwähnt. Auch dem Verlag Galileo Press sei gedankt, hier insbesondere Florian Zimniak. Die

professionelle Unterstützung in Fragen des Layouts und in den verlagstechnischen Angelegenheiten hat zum Erscheinen dieses Buches wesentlich beigetragen.

Walldorf, im März 2005
Ewald Brochhausen – Jürgen Kielisch –
Jürgen Schnerring – Jens Staeck

6 Reporting im HR

In diesem Kapitel wird dargestellt, wie Reports im HR für die verschiedenen Applikationen aussehen können. An Beispielen wird gezeigt, welche grundlegenden Elemente bei der Erstellung der Reports im HR verwendet werden.

6.1 Stammdaten

Die Infotypen der Personaladministration sind in dem Bereich von 0000 bis 0999, die Infotypen der Zeitwirtschaft in dem Bereich von 2000 bis 2999 zu finden. Hinzu kommen eventuell kundeneigene Infotypen im Bereich ab 9000. Wie bei einer Auswertung der Stammdaten sinnvoll vorzugehen ist, wird an den nächsten Beispielen auf der Basis der logischen Datenbank PNP erläutert.

6.1.1 Aufbau eines Reports für Infotypen der Personaladministration

Erstellen einer einfachen Liste

Liste Bankverbindungen

In dem dargestellte Beispiel sollen die bereits in Kapitel 3 erläuterten Verfahren um weitere Hinweise bei der Erstellung von Reports ergänzt werden. Als erstes Beispiel soll eine einfache Liste erstellt werden, in der zum Monatsende die Bankverbindung der Mitarbeiter mit Kontonummer, Bankleitzahl und Name des Geldinstituts aufgeführt ist (siehe Abbildung 6.1).

Reportklasse

Für das Selektionsbild muss die geeignete Reportklasse verwendet werden. Sinnvoll ist hier die Stichtagsbetrachtung mit der Standard-Reportklasse `XX_10001`. Der Aufbau dieser Reportklassen wurde bereits in Abschnitt 3.1 erklärt. Ebenfalls bei den Eigenschaften des Reports wird die logische Datenbank `PNP` eingetragen. Hieraus ergibt sich das in Abbildung 6.2 dargestellte Selektionsbild.

Listüberschrift

In der Listüberschrift soll das Stichtagsdatum in der Form xx.xx.xxxx erscheinen. Hierzu wird die Variable `&0` mit ergänzenden Punkten in der Gesamtlänge (10) der anzuzeigenden Variablen in die Listüberschrift bei den Textelementen aufgenommen (siehe Abbildung 6.3).

Abbildung 6.1 Mitarbeiterliste mit Bankverbindung

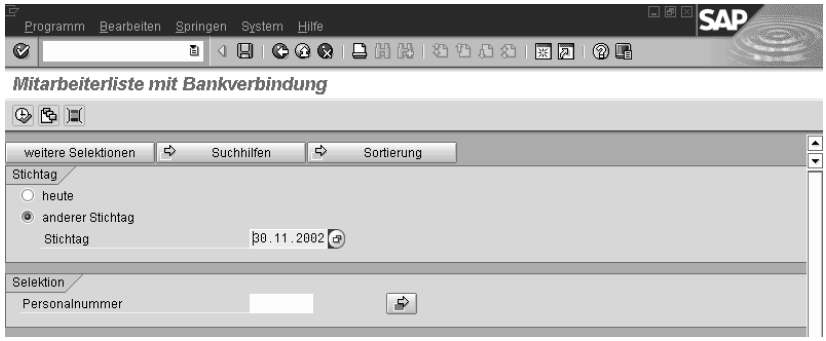

Abbildung 6.2 Selektionsbild mit Stichtag

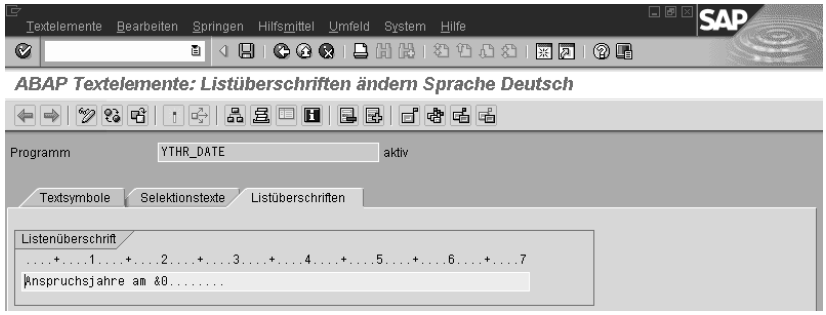

Abbildung 6.3 Textelemente zum Report »Anspruchsjahre«

Deklarationen

Am Anfang des Reports erfolgen die notwendigen Deklarationen. Für die logische Datenbank ist die Anweisung `TABLES: pernr` erforderlich. Verarbeitet werden die Infotypen 0002 (*Daten zur Person*) und 0009 (*Bank-*

verbindung). Weiterhin werden die notwendigen Variablen `coname` und `bankname` definiert.

```
REPORT  ythr_bank LINE-SIZE 132.
TABLES: pernr.
INFOTYPES: 0002, 0009.
DATA: coname(40),              "Nachname, Vorname
      bankname(40).            "Name der Bank
```

Vorschlagswerte

Die Ermittlung des Vorschlagswertes im Selektionsbild erfolgt zum Zeitpunkt `INITIALIZATION`. In dem Modul-Pool `sapfp500` befinden sich einige hilfreiche Routinen für das Berechnen von Zeitpunkten wie z. B. Addition einer Anzahl von Monaten zu einem gegebenen Datum. Hier wird mit dem externen Aufruf `last_day_in_month` aus dem Tagesdatum der Monatsletzte ermittelt. Dieser Wert wird als `pnpbegda` in dem Selektionsbild vorgeschlagen und kann bei Bedarf noch überschrieben werden.

```
INITIALIZATION.
  PERFORM last_day_in_month(sapfp500)
    USING sy-datum prop_date.
  pnpbegda = prop_date.
```

Für die Zuweisung des Datums in der Listüberschrift wird zum Zeitpunkt `INITIALIZATION` das eventuell veränderte Datum in die Systemvariable `syst-tvar0` geschrieben.

```
START-OF-SELECTION.
  WRITE pn-begda TO syst-tvar0 DD/MM/YYYY.
```

GET PERNR

Zum Zeitpunkt `GET pernr` erfolgt die Verarbeitung für jede Personalnummer. Zunächst werden alle Werte initialisiert, die im Report berechnet werden. Die aktuell gültigen Werte der Infotypen 0002 und 0009 (mit dem Subtyp 0) werden in die Kopfzeilen der internen Tabellen `P0002` und `P0009` gestellt. Der Name wird aus Nachname und Vorname zusammengesetzt und der Name der Bank gelesen. Schließlich erfolgt die Ausgabe in eine Liste.

```
GET pernr.
  PERFORM clear_all.
  rp_provide_from_last p0002 space pn-begda pn-begda.
  rp_provide_from_last p0009 '0' pn-begda pn-begda.
  PERFORM concat_name USING coname.
  PERFORM read_bankname USING bankname.
```

```
  WRITE: / pernr-pernr,
           coname,
           p0009-bankn(10),
           p0009-bankl(8),
           bankname.
END-OF-SELECTION.
```

Die einzelnen Routinen haben die im Folgenden beschriebenen Aufgaben. Die reportspezifischen Felder `coname` und `bankname` werden initialisiert, bei umfangreicheren Reports könnte dies auch in einem eigenen Include erfolgen.

```
FORM clear_all.
  CLEAR: coname, bankname.
ENDFORM.
```

Aufbereitung Name

Aus den aktuell gültigen Feldern `p0002-nachn` und `p0002-vorna` wird der Name in der gewünschten Darstellung gebildet.

```
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna
    INTO p_conname SEPARATED BY ', '.
ENDFORM.
```

Interface FI

Der Name der Bank ist in Tabellen des FI gespeichert. Zum Lesen des Banknamens wählt man einen geeigneten Funktionsbaustein des HR-Interfaces. Diese beginnen mit »HRCA«. Für die Zwecke dieses Reports ist der Funktionsbaustein `HRCA_READ_BANK_ADDRESS_2` gut geeignet, der auf der Basis der Importparameter `bank_country` (Bankland) und `bank_number` (Bankleitzahl) eine Reihe Informationen der Bank in der Struktur `bankdata` zurückliefert. Benötigt wird hieraus das Feld `banka` (Name des Geldinstituts).

```
FORM read_bankname USING p_bankname.
  DATA : bankdata TYPE bnka_bf.
  CALL FUNCTION 'HRCA_READ_BANK_ADDRESS_2'
    EXPORTING
      bank_country = p0009-banks
      bank_number = p0009-bankl
    IMPORTING
      bank_data    = bankdata
    EXCEPTIONS
      not_found    = 1
```

```
      OTHERS        = 2.
  IF sy-subrc = 0.
    p_bankname = bankdata-banka.
  ELSE.
    p_bankname = 'Keine Bankverbindung'.
  ENDIF.
ENDFORM.
```

Auswertung von Wiederholungsstrukturen

In einigen Infotypen der Stammdaten können die Ausprägungen von Eigenschaften mehrfach in tabellarischer Form erfasst werden. Beispiele hierfür sind die Erfassung von Lohnarten in den Infotypen 0008 (*Basisbezüge*) und 0052 (*Verdienstsicherung*) oder die in Abbildung 6.4 dargestellten Datumsfelder des Infotyps 0041 (*Datumsangaben*).

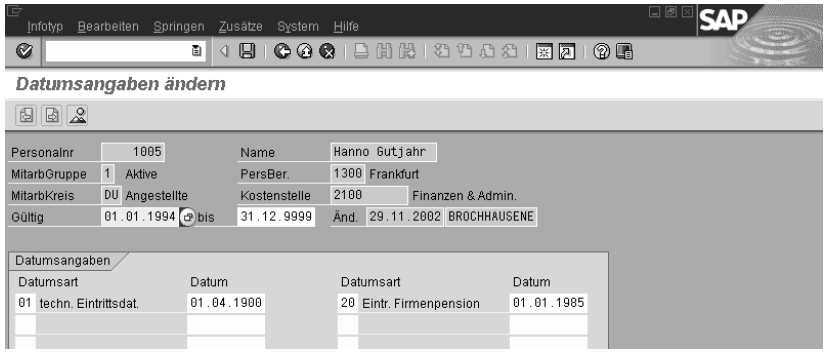

Abbildung 6.4 Datumsangaben in Infotyp 0041

Datumsangaben

Insgesamt sind in diesem Infotyp zwölf verschiedene Datumsarten möglich. Im Data Dictionary abgebildet sind diese Felder als eine Kette von Einzelfeldern, die sich durch die Nummern im Feldnamen unterscheiden. Abbildung 6.5 zeigt einen Ausschnitt aus der Datenstruktur des Infotyps 0041.

Der nachfolgend beschriebene Report zeigt die Verfahrensweise, wie auf diese Datenfelder zugegriffen werden kann. Es soll in diesem Beispiel die Zahl der Anspruchsjahre zum Monatsersten bestimmt werden. In der Liste sollen dabei das Eintrittsdatum und das Basisdatum (Datumsart 20) erscheinen. Die Anspruchsjahre werden als Differenz aus dem Selektionsdatum und dem jüngsten Datum aus Eintrittsdatum und Basisdatum berechnet. Es sollen nur die Personalnummern verarbeitet werden, bei

denen die Datumsart 20 zum Selektionszeitpunkt vorhanden ist. In Abbildung 6.6 ist die erzeugte Liste dargestellt.

Abbildung 6.5 Struktur Infotyp 0041 (Datumsangaben)

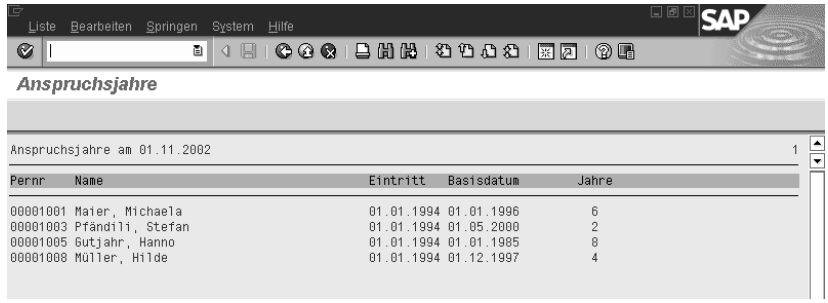

Abbildung 6.6 Listausgabe des Reports »Anspruchsjahre«

Datendeklarationen

Am Anfang des Reports erfolgen zunächst die notwendigen Datendeklarationen. Für das Lesen der Wiederholungsstrukturen werden die Felder `dar` und `dat` benötigt.

```
REPORT  yrhr_date LINE-SIZE 132.
TABLES: pernr.
INFOTYPES: 0002, 0041.
DATA: hire_date LIKE sy-datum, "Eintrittsdatum
      rel_date LIKE sy-datum,  "Berechnungsbasis
      rel_years TYPE i,        "Anspruchsjahre
      coname(40).              "Nachname, Vorname
DATA: dar LIKE p0041-dar01,    "Datumsart
      dat LIKE p0041-dat01.    "Datum
```

Zum Zeitpunkt INITIALIZATION wird aus dem Tagesdatum der Monatserste abgeleitet und für das Selektionsdynpro zur Verfügung gestellt. Die Systemvariable syst-tvar0 wird für die Listüberschrift gefüllt.

```
INITIALIZATION.
  pnpbegda = sy-datum.
  pnpbegda+6(2) = '01'.
START-OF-SELECTION.
  WRITE pn-begda TO syst-tvar0 DD/MM/YYYY.
```

GET PERNR

Zum Zeitpunkt GET pernr erfolgt wieder die Verarbeitung für jede Personalnummer. Zuerst werden alle Werte initialisiert, die im Report berechnet werden. Es wird geprüft, ob die interne Infotyptabelle P0041 gefüllt ist. In älteren Releases muss hier statt Verwendung von lines(p0041) noch eine Hilfsvariable mit DESCRIBE versorgt werden. Die aktuell gültigen Werte des Infotyps 0041 werden mit dem Makro RP_PROVIDE_FROM_LAST in die Kopfzeile der internen Tabellen P0041 gestellt. Ist kein gültiger Wert vorhanden, erhält das Feld pnp-sw-found einen Wert ungleich Null. Auch in diesem Fall erfolgt für die Personalnummer keine weitere Verarbeitung. Danach wird der aktuelle Wert des Infotyps 0002 gelesen. Der Name wird aus Nachname und Vorname zusammengesetzt. In den beiden Routinen read_hire_date und read_p0041 werden die interessierenden Datumswerte gelesen. Je nach Ausprägung dieser Werte werden die Anspruchsjahre berechnet und in der Liste ausgegeben.

```
GET pernr.
  PERFORM clear_all.
  CHECK lines( p0041 ) > 0.
  rp_provide_from_last p0041 space pn-begda pn-begda.
  CHECK pnp-sw-found <> 0.
  rp_provide_from_last p0002 space pn-begda pn-begda.
  PERFORM concat_name USING coname.
  PERFORM read_hire_date USING hire_date.
  PERFORM read_p0041 USING rel_date.
  IF rel_date < hire_date.
    rel_years = pn-begda+0(4) - hire_date+0(4).
    IF hire_date+4(4) >= pn-begda+4(4).
      rel_years = rel_years - 1.
    ENDIF.
  ELSE.
```

```
    rel_years = pn-begda+0(4) - rel_date+0(4).
    IF rel_date+4(4) => pn-begda+4(4).
      rel_years = rel_years - 1.
    ENDIF.
  ENDIF.
  WRITE: / pernr-pernr,
           coname,
           hire_date,
           rel_date,
           rel_years.
END-OF-SELECTION.
```

In Routine `clear_all` werden die reportspezifischen Variablen initialisiert:

```
FORM clear_all.
  CLEAR: coname, rel_date, hire_date, rel_years.
ENDFORM.
```

Die Aufbereitung des Namens erfolgt in der Routine `concat_name`:

```
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna
    INTO p_conname SEPARATED BY ', '.
ENDFORM.
```

Eintrittsdatum

Das Eintrittsdatum wird mit dem Funktionsbaustein `HR_ENTRY_DATE` bestimmt, der bereits in Kapitel 3 vorgestellt wurde:

```
FORM read_hire_date USING p_hire_date.
  CALL FUNCTION 'HR_ENTRY_DATE'
    EXPORTING
      persnr               = pernr-pernr
    IMPORTING
      entrydate            = p_hire_date
    EXCEPTIONS
      entry_date_not_found = 1
      pernr_not_assigned   = 2
      OTHERS               = 3.
  IF sy-subrc <> 0.
* ...
```

```
  ENDIF.
ENDFORM.
```

In der Routine `read_p0041` wird der Wert für die Datumsart 20 ermittelt. Mit der Anweisung

DO-Schleife

```
DO 12 TIMES
  VARYING dar
    FROM p0041-dar01 NEXT p0041-dar02
```

wird die Struktur `p0041` verarbeitet. Bei jedem Durchlaufen der `DO`-Schleife wird das Feld `dar` gefüllt. Der Aufsetzpunkt wird durch das Feld `p0041-dar01` bestimmt, die Schrittweite durch die Differenz zum Feld `p0041-dar02`. Analog wird beim Feld `dat` verfahren. Wenn keine Einträge mehr gefunden werden oder die Datumsart 20 gelesen worden ist, wird die Schleife verlassen:

```
FORM read_p0041 USING p_rel_date.
  DO 12 TIMES
    VARYING dar
      FROM p0041-dar01 NEXT p0041-dar02
    VARYING dat
      FROM p0041-dat01 NEXT p0041-dat02
    IF dar IS INITIAL.
      EXIT.
    ELSEIF dar = '20'.
      p_rel_date = dat.
      EXIT.
    ENDIF.
  ENDDO.
ENDFORM.
```

Auswertung der Basisbezüge

Häufig ist die Ermittlung der Lohnarten und von deren Beträgen aus dem Infotyp 0008 (*Basisbezüge*) notwendig. Diese sind wie im vorangegangenen Beispiel in Wiederholungsgruppen des Infotyps 0008 gespeichert. Zudem werden die Beträge der indirekt bewerteten Lohnarten nicht in den Tabellen dauerhaft gespeichert, sondern erst zur Laufzeit ermittelt. Für die Auswertung dieser Lohnarten stehen Funktionsbausteine zur Verfügung, die die notwendige Funktionalität enthalten.

Entwicklung der Basisbezüge

In dem Beispielreport soll für die Mitarbeiter die Entwicklung der Basisbezüge der letzten zehn Jahre in einer Liste dargestellt werden. Als Selektionsbild wird das Standardselektionsbild verwendet. Als Vorschlagswerte werden das Jahresende und der Beginn des Auswertungszeitraums für den Datenauswahlzeitraum bestimmt (siehe Abbildung 6.7).

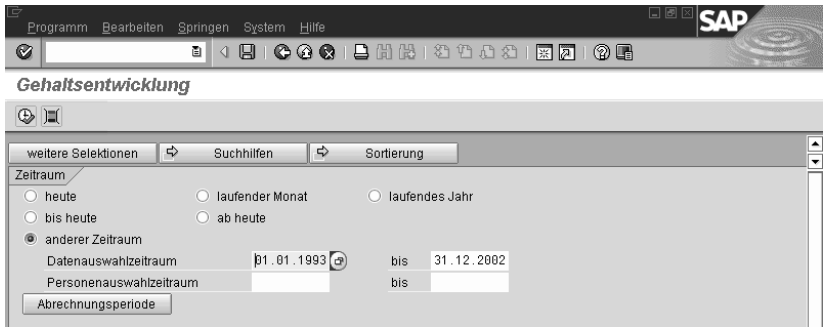

Abbildung 6.7 Selektionsbild des Reports »Gehaltsentwicklung«

In der Liste (siehe Abbildung 6.8) sollen neben Personalnummer und Name die in dem Zeitraum gültigen Gesamtbezüge dargestellt werden. Bei einem Wechsel der Währung sollen alle Beträge in die zuletzt gültige Währung umgerechnet werden. Zudem sollen die Zeilen mit den Namen farblich hervorgehoben werden.

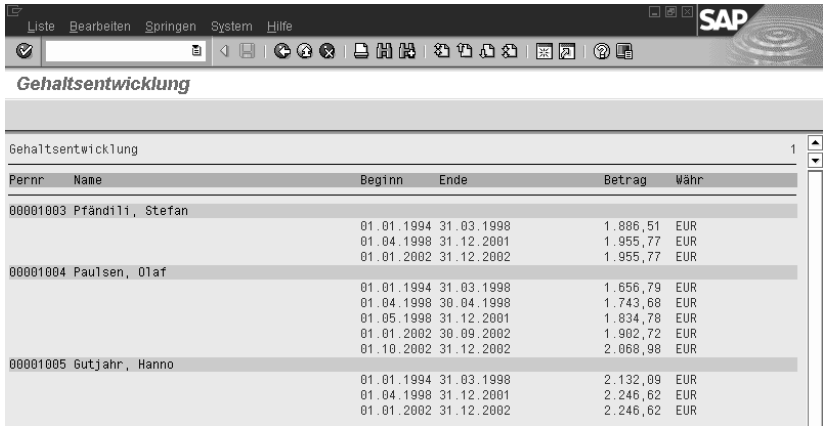

Abbildung 6.8 Listausgabe des Reports »Gehaltsentwicklung«

Bewertete Lohnarten

Die Datendeklarationen umfassen neben den für den später verwendeten Funktionsbaustein `RP_FILL_WAGE_TYPE_TABLE_EXT` notwendigen Infotypen die internen Tabellen der bewerteten Lohnarten `bewlart` und `t_`

bewlart und den Arbeitsbereich w_bewlart. Außerdem wird eine Variable last_curr für die zuletzt verwendete Währung benötigt.

```
REPORT  yrhr_p0008_list LINE-SIZE 132.
TABLES: pernr.
INFOTYPES: 0001, 0002, 0007, 0008.
DATA: bewlart TYPE TABLE OF pbwla,   "Lohnarten 1x
      t_bewlart TYPE TABLE OF pbwla, "Lohnarten gesamt
      w_bewlart TYPE pbwla,          "Lohnarten Work
      last_curr LIKE p0008-waers,    "Letzte Waehrung
      coname(40).                    "Nachname, Vorname
```

Datenauswahlzeitraum

Zum Zeitpunkt INITIALIZATION werden die Vorschlagswerte für den Datenauswahlzeitraum pnpbegda und pnpendda ermittelt.

```
INITIALIZATION.
  pnpendda = sy-datum.
  pnpendda+4(4) = '1231'.
  PERFORM day_minus_years(sapfp500)
    USING pnpendda '10' pnpbegda.
  pnpbegda = pnpbegda + 1.
START-OF-SELECTION.
```

Am Anfang der Verarbeitung für jede Personalnummer wird die Initialisierung vorgenommen und der gültige Name aus dem Infotyp 0002 bestimmt. Die Lohnarten werden mit der Routine read_p0008 aus dem Infotyp 0008 gelesen. Die Ausgabe von Personalnummer und Name wird farblich durch die FORMAT-Anweisung aufbereitet. Danach werden die Lohnarten pro Infotypzeitraum in dem Unterprogramm summiert und ausgeben.

```
GET pernr.
  PERFORM clear_all.
  rp_provide_from_last p0002 space pn-begda pn-endda.
  PERFORM concat_name USING coname.
  PERFORM read_p0008.
  FORMAT COLOR COL_GROUP INTENSIFIED ON.
  WRITE: / pernr-pernr,
           coname, 132 space.
  FORMAT COLOR OFF.
  PERFORM list_sum.
END-OF-SELECTION.
```

In der Routine `clear_all` werden die Variablen, Strukturen und internen Tabellen initialisiert:

```
FORM clear_all.
  CLEAR: coname, bewlart, t_bewlart,
    w_bewlart, last_curr.
ENDFORM.
```

Der Name wird aus Nachname und Vorname gebildet:

```
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna
    INTO p_conname SEPARATED BY ', '.
ENDFORM.
```

In dem Unterprogramm `read_p0008` wird in einer `PROVIDE`-Schleife die interne Infotyptabelle `p0008` für die Daten aus dem Datenauswahlzeitraum bearbeitet. Der Funktionsbaustein `RP_FILL_WAGE_TYPE_TABLE_EXT` füllt für den Betrachtungszeitraum die interne Tabelle der bewerteten Lohnarten `bewlart` aus den aktuellen Werten des Infotyps 0008. In der internen Tabelle `t_bewlart` werden die Ergebnisse aller Perioden für eine Personalnummer gesammelt. Zudem wird die letzte Währung des Infotyps in der Variablen `last_curr` festgehalten.

```
FORM read_p0008.
  PROVIDE * FROM p0008 BETWEEN pn-begda AND pn-endda.
    CALL FUNCTION 'RP_FILL_WAGE_TYPE_TABLE_EXT'
      EXPORTING
        begda                          = p0008-begda
        endda                          = p0008-endda
        infty                          = '0008'
        pernr                          = pernr-pernr
      TABLES
        pp0001                         = p0001
        pp0007                         = p0007
        pp0008                         = p0008
        ppbwla                         = bewlart
      EXCEPTIONS
        error_at_indirect_evaluation = 1
        OTHERS                         = 2.
    IF sy-subrc <> 0.
* ...
```

```
    ENDIF.
    APPEND LINES OF bewlart TO t_bewlart.
  ENDPROVIDE.
  last_curr = p0008-waers.
ENDFORM.
```

Umrechnung der Währung

In der Routine `list_sum` wird die interne Tabelle `t_bewlart` weiterverarbeitet. Wenn die Währung einer Lohnart in dieser Tabelle nicht mit der letzten Währung übereinstimmt, erfolgt die Umrechnung des Betrags mit dem Funktionsbaustein `CONVERT_TO_LOCAL_CURRENCY` und die Tabelle wird entsprechend geändert. Für gleiche Zeiträume werden die Werte summiert und in der Liste ausgegeben.

```
FORM list_sum.
  LOOP AT t_bewlart INTO w_bewlart.
    IF w_bewlart-waers <> last_curr.
      CALL FUNCTION 'CONVERT_TO_LOCAL_CURRENCY'
        EXPORTING
          date             = w_bewlart-endda
          foreign_amount   = w_bewlart-betrg
          foreign_currency = w_bewlart-waers
          local_currency   = last_curr
        IMPORTING
          local_amount     = w_bewlart-betrg
        EXCEPTIONS
          no_rate_found    = 1
          overflow         = 2
          no_factors_found = 3
          no_spread_found  = 4
          derived_2_times  = 5
          OTHERS           = 6.
      IF sy-subrc <> 0.
      ELSE.
        w_bewlart-waers = last_curr.
        MODIFY t_bewlart FROM w_bewlart.
      ENDIF.
    ENDIF.
    AT END OF endda.
      SUM.
      WRITE: /50 w_bewlart-begda, w_bewlart-endda,
                 w_bewlart-betrg, last_curr.
```

```
    ENDAT.
  ENDLOOP.
ENDFORM.
```

6.1.2 Aufbau eines Reports für Infotypen der Zeitwirtschaft

Bei Verwendung der logischen Datenbank `PNP` werden zum Zeitpunkt `GET PERNR` für alle mit `INFOTYPES` deklarierten Infotypen nnnn die internen Tabellen `Pnnnn` gefüllt. Bei der großen Menge an Sätzen bei den Infotypen der Zeitwirtschaft ist dieses Vorgehen hier nicht zu empfehlen. Wie dann zu verfahren ist, wird an dem Report »Übersicht Abwesenheiten« für den Infotyp 2001 (*Abwesenheiten*) erläutert. Für alle ausgewählten Mitarbeiter sollen die in den Reportparametern angegebenen Abwesenheitsarten des laufenden Kalenderjahres auswertet werden. Dabei soll es möglich sein, mit einen Schalter die Auflistung der einzelnen Abwesenheitssätze an- und auszuschalten. Die Zusammenfassung der Abwesenheiten soll für die Personengruppen und -kreise getrennt erfolgen. Das Ergebnis ist in Abbildung 6.9 zu sehen.

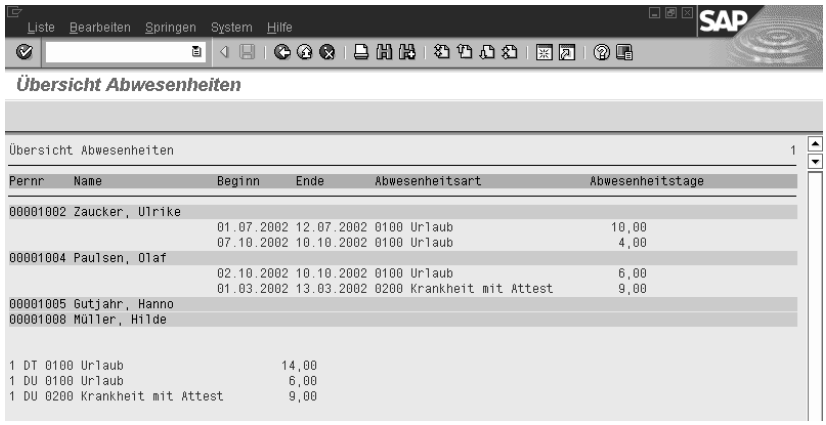

Abbildung 6.9 Listausgabe des Reports »Abwesenheiten«

MODE n Bei der Datendeklaration ist beim Infotyp 2001 der Zusatz `MODE n` angegeben. Dieser Zusatz verhindert das Füllen der internen Tabelle `P2001` zum Zeitpunkt `GET PERNR`. Für das Unterdrücken der Einzelliste ist der Parameter `s_name` eingerichtet, für die Auswahl der interessierenden Abwesenheiten die Selektion `s_abs`. Für die personenübergreifende Zusammenfassung wird die interne Tabelle `t_absence` mit dem Arbeitsbereich `w_absence` angelegt:

```
REPORT yrhr_p2001_list LINE-SIZE 132.
TABLES: pernr.
INFOTYPES: 0001, 0002,
           2001 MODE n.
PARAMETERS: s_name AS CHECKBOX.
SELECT-OPTIONS: s_abs FOR p2001-awart.
DATA: coname(40).  "Name, Vorname
TYPES: BEGIN OF absence,
         persg LIKE p0001-persg,  "Personengruppe
         persk LIKE p0001-persk,  "Personenkreis
         awart LIKE p2001-awart,  "Abwesenheitsart
         abwtg LIKE p2001-abwtg,  "Abwesenheitsname
       END OF absence.
DATA: w_absence TYPE absence,
      t_absence TYPE TABLE OF absence.
```

Zum Zeitpunkt `INITIALIZATION` werden die Vorschlagswerte des Datenauswahlzeitraums ermittelt:

```
INITIALIZATION.
  pnpbegda = pnpendda = sy-datum.
  pnpbegda+4(4) = '0101'.
  pnpendda+4(4) = '1231'.
START-OF-SELECTION.
```

GET PERNR

Für jede Personalnummer werden die bereits bekannten Verarbeitungen vorgenommen und der Infotyp 2001 wird mit der Routine `read_p2001` gelesen. Wenn der Parameter `s_name` aktiviert ist, wird die Einzelverarbeitung mit der Routine `list_name` durchgeführt. Nach der Verarbeitung aller Personalnummern wird die Zusammenfassung mit der Routine `list_sum` ausgegeben:

```
GET pernr.
  PERFORM clear_all.
  rp_provide_from_last p0002 space pn-begda pn-endda.
  PERFORM concat_name USING coname.
  PERFORM read_p2001.
  IF NOT s_name IS INITIAL.
    FORMAT COLOR COL_GROUP INTENSIFIED ON.
    WRITE: / pernr-pernr,
             coname, 132 space.
    FORMAT COLOR OFF.
```

```
    PERFORM list_name.
  ENDIF.
END-OF-SELECTION.
  PERFORM list_sum.
```

Die Verarbeitung der Routinen `clear_all` und `concat_name` erfolgt in der von den anderen Beispielen bereits bekannten Weise.

```
FORM clear_all.
  CLEAR: coname.
ENDFORM.
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna
    INTO p_conname SEPARATED BY ', '.
ENDFORM.
```

Füllen der Tabelle p2001

Zu Beginn der Routine `read_p2001` werden nur diejenigen Zeilen der internen Tabelle `p2001` gefüllt, die durch den Datenauswahlzeitraum festgelegt worden sind. Dies erledigt der Funktionsbaustein `rp_read_all_time_ity`. Aus der internen Tabelle `p2001` werden dann die Einträge gelöscht, die aufgrund der Angaben in den Selektionsoptionen nicht verarbeitet werden sollen. Wenn die Tabelle jetzt noch Werte enthält, werden die Abwesenheiten weiterverarbeitet. Die korrespondieren Inhalte des Infotyps p0001 werden in den Arbeitsbereich `w_absence` übertragen. Das Gleiche geschieht mit den verbliebenen Einträgen der internen Tabelle `p2001`.

```
FORM read_p2001.
  rp_read_all_time_ity pn-begda pn-endda.
  DELETE p2001 WHERE NOT ( awart IN s_abs ).
  IF lines( p2001 ) > 0.
    MOVE-CORRESPONDING p0001 TO w_absence.
    LOOP AT p2001.
      MOVE-CORRESPONDING p2001 TO w_absence.
      COLLECT w_absence INTO t_absence.
    ENDLOOP.
  ENDIF.
ENDFORM.
```

Abwesenheitsarten

Für die Auflistung der einzelne Abwesenheiten in dem Unterprogramm `list_name` wird innerhalb des Loops über die interne Tabelle `p2001` der Text der Abwesenheitsart mit der Routine `re554t` gelesen. Zur Vereinfachung wird angenommen, dass der zweite Parameter (Gruppierung der

Personalteilbereiche für Ab-/Anwesenheitsarten) den konstanten Wert »01« hat.

```
FORM list_name.
  DATA: atext LIKE t554t-atext.
  LOOP AT p2001.
    PERFORM re554t
      USING sy-langu '01' p2001-awart atext.
    WRITE: /30 p2001-begda, p2001-endda,
      p2001-awart, atext, p2001-abwtg.
  ENDLOOP.
ENDFORM.
```

In der Routine `re554t` wird die Texttabelle T554T der Abwesenheitsarten in der Anmeldesprache gelesen und zur Verfügung gestellt: **Texttabelle**

```
FORM re554t USING p_langu p_moabw p_awart p_atext.
  DATA: w_t554t LIKE t554t.
  SELECT SINGLE * FROM t554t
    INTO w_t554t
    WHERE sprsl = p_langu
      AND moabw = p_moabw
      AND awart = p_awart.
  IF sy-subrc = 0.
    p_atext = w_t554t-atext.
  ELSE.
    CLEAR p_atext.
  ENDIF.
ENDFORM.
```

Die Zusammenfassung wird in dem `LOOP` über die Tabelle `t_absence` ausgegeben.

```
FORM list_sum.
  DATA: atext LIKE t554t-atext.
  SKIP 2.
  LOOP AT t_absence INTO w_absence.
    PERFORM re554t
      USING sy-langu '01' w_absence-awart atext.
    WRITE: / w_absence-persg, w_absence-persk,
      w_absence-awart, atext, w_absence-abwtg.
```

```
  ENDLOOP.
ENDFORM.
```

6.2 Organisationsmanagement

Wie in Kapitel 3 dargestellt, sind bei der Verwendung der logischen Datenbank PCH sequenzielle und strukturelle Auswertungen möglich.

6.2.1 Sequenzielle Auswertungen

Stellenbeschreibung

Die Anwendung einer sequenziellen Auswertung wird in dem folgenden Report näher erläutert. Die verschiedenen Objekttypen können im Infotyp 1002 eine verbale Beschreibung erhalten. In dem Beispiel soll die Beschreibung der Stelle (Objekttyp C) zu einem Stichtag ausgegeben werden. Die Stellen-ID und die Stellenbezeichnung sollen farblich hervorgehoben werden (siehe Abbildung 6.10).

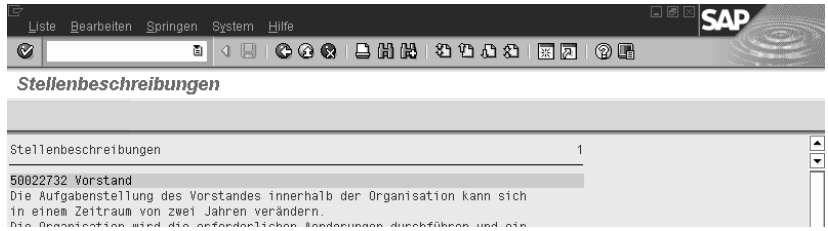

Abbildung 6.10 Sequenzielle Auswertung: Stellenbeschreibung

Im Deklarationsteil des Reports wird bei der `TABLES`-Anweisung *nicht* `GDSTR` angegeben. Hierdurch wird erkannt, dass *keine* strukturelle, sondern eine sequenzielle Auswertung erfolgen soll. Unter `INFOTYPES` wird der Infotyp 1002 (*Verbale Beschreibung*) deklariert. Dieser Infotyp ist als Tabelleninfotyp ausgeprägt. Deshalb ist zusätzlich die Definition der internen Tabelle `t_pt1002` erforderlich, die im weiteren Verlauf von einem Makro gefüllt wird.

```
REPORT  ythr_org_seq LINE-SIZE 80.
TABLES: objec.
INFOTYPES: 1002.
DATA: t_pt1002 TYPE TABLE OF pt1002 WITH HEADER LINE.
```

Stichtag

Zum Zeitpunkt `INITIALIZATION` wird durch das Makro `rh-sel-key-date` erreicht, dass kein Auswahlzeitraum, sondern ein Stichtag in dem Selektionsbild angezeigt wird. Als Vorschlagswert für dieses Datum wird der 1. Januar des laufenden Jahres bereitgestellt. Weiterhin werden Vor-

schlagswerte für die Planvariante `pchplvar` und den Objekttyp `pchotype` erzeugt.

```
INITIALIZATION.
  rh-sel-keydate.
  pchplvar = '01'.
  pchotype = 'C'.
  pchobeg = sy-datum.
  pchobeg+4(4) = '0101'.
START-OF-SELECTION.
```

GET objec

Bei `GET objec` stehen hier die Inhalte des Infotyps 1000 zum Stichtag zur Verfügung. Ausgegeben werden die Objekt-ID und die Bezeichnung des Objekts.

```
GET objec.
  FORMAT COLOR COL_GROUP INTENSIFIED ON.
  WRITE: / objec-objid, objec-stext, 80 space.
  FORMAT COLOR OFF.
```

Tabelleninfotyp

Bei der `PROVIDE`-Schleife über den Infotyp 1002 wird auf die Werte der Anmeldesprache eingeschränkt. Mit dem Makro `rh-get-tbdat` werden für den Tabelleninfotyp die Tabelleneinträge in die interne Tabelle `t_pt1002` eingelesen. Der Inhalt dieser internen Tabelle kann dann in einer `LOOP`-Schleife ausgegeben werden.

```
  PROVIDE * FROM p1002
   BETWEEN pc-begda AND pc-endda
    WHERE p1002-langu = sy-langu.
    rh-get-tbdat p1002-infty p1002-tabnr t_pt1002.
    LOOP AT t_pt1002.
      WRITE: / t_pt1002-tline.
    ENDLOOP.
  ENDPROVIDE.
```

6.2.2 Strukturelle Auswertung

Bedeutend häufiger werden bei der Verwendung der logischen Datenbank PCH strukturelle Auswertungen durchgeführt.

Stellenbesetzungsplan

Es soll ein Stellenbesetzungsplan mit folgenden Informationen erstellt werden:

- Organisationseinheit, -bezeichnung
- Planstelle, Bezeichnung
- Personalnummer, Name (Nachname, Vorname)

Die Organisationseinheiten (Objekttyp »O«) werden ab Stelle 1 angedruckt. Entsprechend der Hierarchie werden Einrückungen vorgenommen. Die Planstellen (Objekttyp »S«) werden ab Stelle 20, die Personalnummern (Objekttyp »P«) ab Stelle 30 angedruckt. Für Organisationseinheiten, Planstellen und Personendaten werden unterschiedliche Hintergrundfarben verwendet.

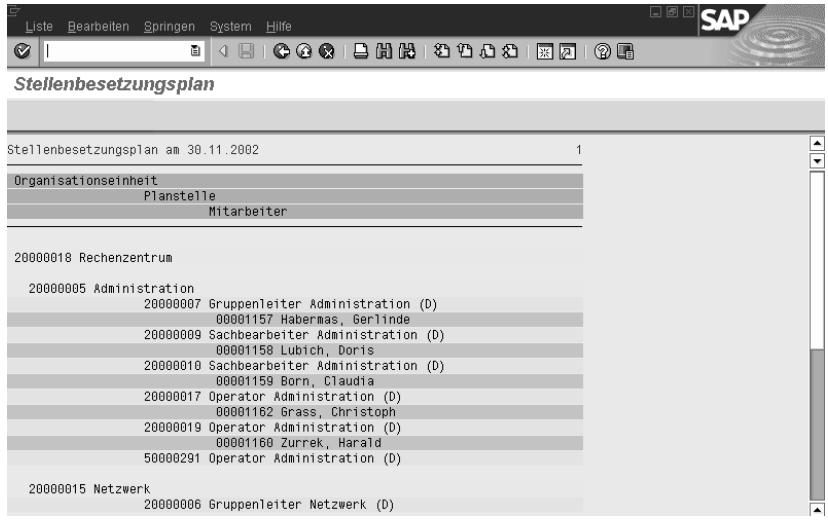

Abbildung 6.11 Strukturelle Auswertung: Stellenbesetzungsplan

Anzeige Auswertungsweg

Bei einer strukturellen Auswertung mit der logischen Datenbank PCH werden alle Objekte gelesen, die über den in den Selektionsparametern angegebenen Auswertungsweg erreichbar sind. Diese Auswertungswege werden im Customizing eingerichtet. In diesen Report soll der Auswertungsweg nicht bei den Selektionsparametern erscheinen, sondern fest vorgegeben werden und nicht änderbar sein. Aus diesem Grund ist es erforderlich, bei den Reporteigenschaften die Selektionsbildversion 900 (Strukturauswertung ohne Strukturparameter) zu nutzen.

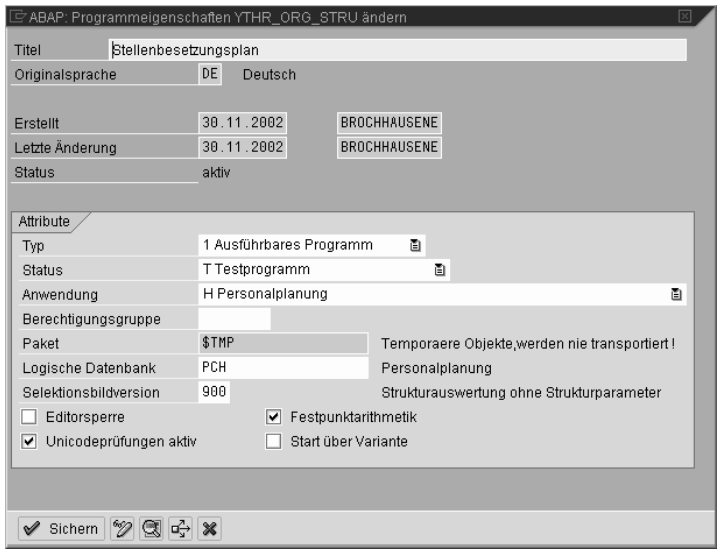

Abbildung 6.12 Eigenschaften des Reports »Stellenbesetzungsplan«

Im Deklarationsteil des Reports wird anhand der `TABLES`-Eintragung `gdstr` erkannt, dass eine strukturelle Auswertung vorliegt. Für den Andruck des Namens ist der Infotyp 0002 zu deklarieren. Für die Einrückung bei den Organisationseinheiten wird die Hilfsvariable `col` verwendet:

```
REPORT ythr_org_stru LINE-SIZE 80.
TABLES: objec,
        gdstr.
INFOTYPES: 0002.
DATA: col LIKE struc-level,
      coname(40).
```

Zum Zeitpunkt `INITIALIZATION` erfolgt neben den aus dem vorherigen Beispiel bekannten Anweisungen die Bestimmung des Auswertungswegs. Für die Überschrift wird die Systemvariable `syst-tvar0` mit dem Stichtagsdatum gefüllt:

```
  rh-sel-keydate.
  pchotype = 'O'.
  pchwegid = 'O-S-P'.
START-OF-SELECTION.
  WRITE pc-begda TO syst-tvar0 DD/MM/YYYY.
```

GET objec Bei `GET objec` werden jetzt alle Objekte auf dem Auswertungsweg mit ihren Infotypen zur Verfügung gestellt. Für die objektspezifische Aufbereitung wird auf den Objekttyp abgefragt. Bei dem Objekttyp »O« wird entsprechend der Auswertungsebene `struc-level` die Andruckposition bestimmt. Für die beiden Objekttypen »O« und »S« werden für die Ausgabe die Werte der Struktur `objec` verwendet.

```
GET objec.
  IF objec-otype = 'O'.
    col = struc-level * 2 .
    SKIP.
    FORMAT COLOR 2.
    WRITE:  AT col objec-objid, objec-stext, 80 space.
  ENDIF.
  IF objec-otype = 'S'.
    FORMAT COLOR 3.
    WRITE: /20 objec-objid, objec-stext, 80 space.
  ENDIF.
```

Für den Personalstamm stehen die in `INFOTYPES` deklarierten internen Infotyptabellen zur Verfügung. Hier wird der Infotyp 0002 ausgewertet und der Name mit der Routine `concat_name` aufbereitet:

```
  IF objec-otype = 'P'.
    PROVIDE vorna nachn FROM p0002
      BETWEEN pc-begda AND pc-endda.
      PERFORM concat_name USING coname.
      FORMAT COLOR 4.
      WRITE: /30  objec-objid, coname, 80 space.
    ENDPROVIDE.
  ENDIF.
END-OF-SELECTION.
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna
    INTO p_conname SEPARATED BY ', '.
ENDFORM.
```

6.3 Abrechnungsdaten

In Abschnitt 3.5 wurde bereits der Zugriff auf die Abrechnungsdaten dargestellt. An dem folgenden Beispiel sollen weitere Funktionsbausteine, die die Auswertung der Daten der Personalabrechung unterstützen, erläutert werden.

Als Ergebnis des Beispielreports sieht man die Liste in Abbildung 6.13. Aufgeführt sind unter den Namen der Mitarbeiter die Abrechnungsperioden und die gezahlten Nettogehälter.

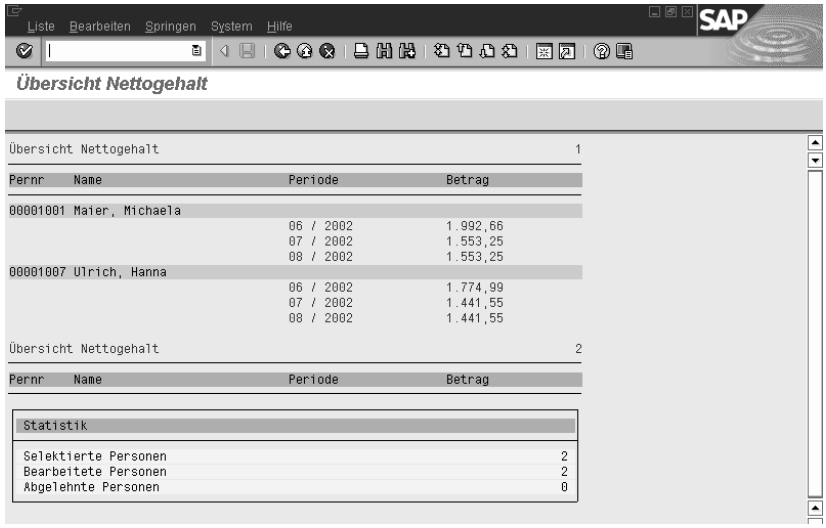

Abbildung 6.13 Report »Übersicht Nettogehalt«

Am Ende der Liste wird eine Zusammenfassung ausgegeben. Falls Fehler bei der Bearbeitung aufgetreten sind, wird eine Fehlerliste angezeigt (Abbildung 6.14).

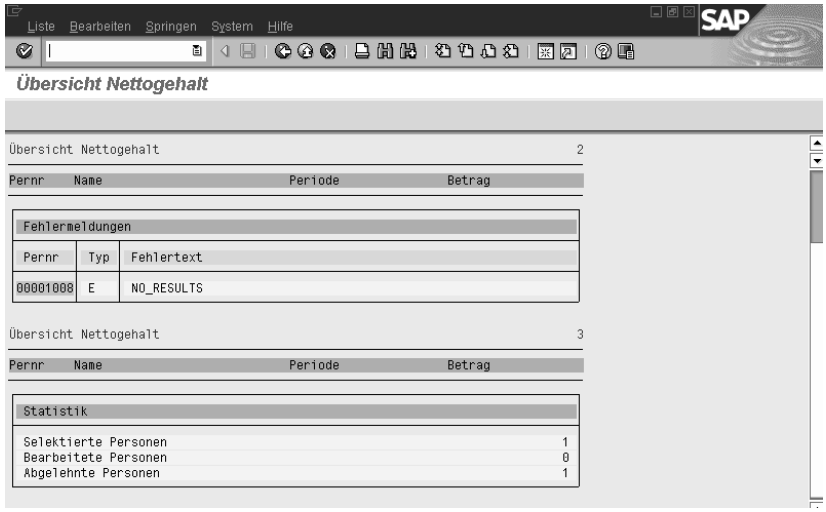

Abbildung 6.14 Fehlermeldungen im Report »Übersicht Nettogehalt«

Auswertung der Abrechnungsergebnisse

Für die Auswertung der Abrechnungsergebnisse ist die Definition der internen Tabelle `t_result` mit dem Arbeitsbereich `w_result` erforderlich. Notwendig für die Bearbeitung der Lohnarten aus der Tabelle RT des Abrechnungsclusters ist die Deklaration des Arbeitsbereichs `w_rt`. Die weiteren Datendefinitionen werden für die Reportstatistik und die Ausgabe der Fehlertabelle benötigt.

```
REPORT  ythr_pay_net LINE-SIZE 80.
TABLES: pernr.
INFOTYPES: 0002.
DATA: coname(40).
DATA: t_result TYPE TABLE OF payde_result,
      w_result TYPE payde_result,
      w_rt TYPE pc207.
DATA:  g_select TYPE i,
       g_proces TYPE i,
       g_reject TYPE i.
DATA: error_int TYPE TABLE OF hrerror.
```

Am Anfang der Verarbeitung müssen das ABAP Memory für die Fehlerliste und die Reportstatistik mit den beiden aufgeführten Funktionsbausteinen initialisiert werden:

```
START-OF-SELECTION.
  CALL FUNCTION 'HR_REFRESH_ERROR_LIST'.
  CALL FUNCTION 'HR_REFRESH_STAT_LIST'.
```

GET pernr

Zum Zeitpunkt `GET pernr` werden zunächst bereits bekannte Verarbeitungen durchgeführt. Die Anzahl der selektierten Mitarbeiter wird in der Variablen `g_select` fortgeschrieben:

```
GET pernr.
  rp_provide_from_last p0002 space pn-begda pn-begda.
  PERFORM concat_name USING coname.
  FORMAT COLOR COL_GROUP INTENSIFIED ON.
  WRITE: / pernr-pernr,
           coname, 80 space.
  FORMAT COLOR OFF.
  g_select = g_select + 1.
```

Der Import der aktuellen Abrechnungsergebnisse des Datenauswahlzeitraums in die Tabelle `t_result` wird mit dem Funktionsbaustein `HR_GET_PAYROLL_RESULTS` durchgeführt:

```
CALL FUNCTION 'HR_GET_PAYROLL_RESULTS'
  EXPORTING
    pernr                          = pernr-pernr
    pabrj                          = pn-begda(4)
    pabrp                          = pn-begda+4(2)
    pabrj_end                      = pn-endda(4)
    pabrp_end                      = pn-endda+4(2)
    actual                         = 'A'
  TABLES
    result_tab                     = t_result
  EXCEPTIONS
    no_results                     = 1
    error_in_currency_conversion   = 2
    t5001_entry_not_found          = 3
    period_mismatch_error          = 4
    t549q_entry_not_found          = 5
    internal_error                 = 6
    wrong_structure_of_result_tab = 7.
```

Fehlerliste

Im Fehlerfall wird der Funktionsbaustein `HR_APPEND_ERROR_LIST` aufgerufen, der die Fehlerliste `error_int` im ABAP Memory mit der ID »HRERRLIST« fortschreibt. Für jede Personalnummer erfolgt die Ausgabe der Liste der Entwicklung des Nettoentgelts mit der Routine `list`.

```
IF sy-subrc NE 0.
  CALL FUNCTION 'HR_APPEND_ERROR_LIST'
    EXPORTING
      pernr = pernr-pernr
      arbgb = sy-msgid
      msgty = 'E'
      msgno = sy-msgno
      msgv1 = sy-msgv1
      msgv2 = sy-msgv2
      msgv3 = sy-msgv3
      msgv4 = sy-msgv4.
  g_reject = g_reject + 1.
ELSE.
```

```
    g_proces = g_proces + 1.
  ENDIF.
  PERFORM list.
END-OF-SELECTION.
```

Nach Ende der Selektion wird die Fehlertabelle `error_int` aus dem ABAP Memory gelesen und mit dem Funktionsbaustein `HR_DISPLAY_ERROR_LIST` ausgegeben.

```
  NEW-PAGE.
  IMPORT error_int FROM MEMORY ID 'HRERRLIST'.
  CALL FUNCTION 'HR_DISPLAY_ERROR_LIST'
    EXPORTING
      no_popup         = 'X'
      no_print         = ' '
    TABLES
      error            = error_int
    EXCEPTIONS
      invalid_linesize = 1
      OTHERS           = 2.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
```

Reportstatistik

Die Verarbeitung und Ausgabe der Reportstatistik wird von den Funktionsbausteinen `HR_APPEND_STAT_LIST` und `HR_DISPLAY_STAT_LIST` durchgeführt:

```
  CALL FUNCTION 'HR_APPEND_STAT_LIST'
    EXPORTING
      selected  = g_select
      processed = g_proces
      rejected  = g_reject.
  CALL FUNCTION 'HR_DISPLAY_STAT_LIST'
    EXPORTING
      no_popup         = 'X'
      no_print         = ' '
    EXCEPTIONS
      invalid_linesize = 1
      OTHERS           = 2.
  IF sy-subrc <> 0.
```

```
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.
```

Die Aufbereitung des Namens erfolgt mit der Routine `concat_name`:

```
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna
    INTO p_conname SEPARATED BY ', '.
ENDFORM.
```

Lohnartentabelle

Die Ausgabe der Abrechnungsergebnisse wird pro Abrechnungsperiode in dem `LOOP` über die interne Tabelle `t_result` durchgeführt. Die Werte der Lohnartentabelle RT werden einer tieferen Ebene in den Arbeitsbereich `w_rt` eingelesen. Das Nettoentgelt ist in der technischen Lohnart »/550« zu finden.

```
FORM list.
  LOOP AT t_result INTO w_result.
    WRITE : /40 w_result-evp-fpper+4(2) ,
      '/', w_result-evp-fpper(4).
    LOOP AT w_result-inter-rt INTO w_rt
      WHERE lgart = '/550'.
      WRITE: w_rt-betrg.
    ENDLOOP.
  ENDLOOP.
ENDFORM.
```

Bei der Auswertung der Ergebnisse wird hier vereinfachend davon ausgegangen, dass dieselben Lohnarten nicht mit mehreren Zeitraumkennzeichen vorliegen. Dieser Sachverhalt ist noch entsprechend zu berücksichtigen.

6.4 Zeitwirtschaft

Ergebnisse der Zeitauswertung

In der Zeitwirtschaft sind die Ergebnisse der Zeitauswertung wie die Abrechnungsergebnisse in Clustern gespeichert. Wie diese zu lesen sind, wird in diesem Kapitel dargestellt. In der Abbildung 6.15 sieht man die Liste der erzeugten Zeitpaare und der Zeitart »0003«, Rahmenzeit. Das Ende jeder Woche wird durch eine horizontale Linie verdeutlicht.

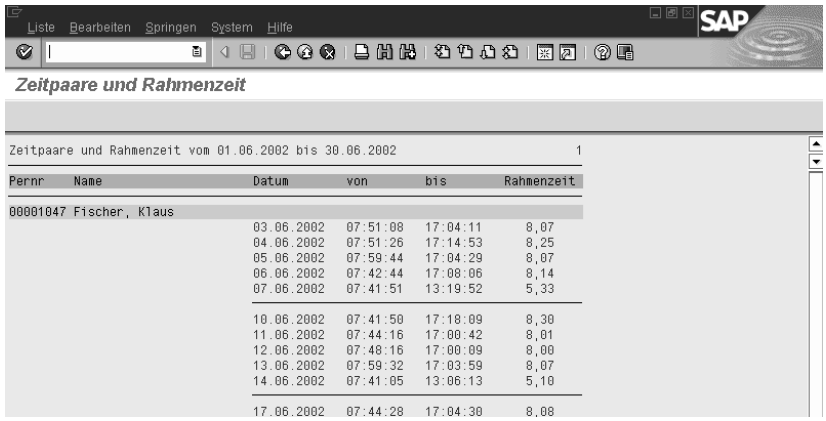

Abbildung 6.15 Report der Zeitauswertung

Datendeklaration

Bei der Datendeklaration sind die notwendigen Includes für die Pufferung und die Datenbeschreibungen der Cluster aufgeführt. Die Konventionen hierfür wurden in Abschnitt 3.5 dargestellt.

```
REPORT  yrhr_time LINE-SIZE 80.
TABLES: pernr,
        pcl1,
        pcl2.
INFOTYPES: 0002.
INCLUDE rpppxd00.       "Daten fuer PCL1/2-Puffer
DATA:   BEGIN OF COMMON PART buffer.
INCLUDE rpppxd10.
DATA:   END OF COMMON PART.
INCLUDE: rpc2b200.
DATA: coname(40),              "Nachname, Vorname
      odate LIKE pt-ldate,     "Wochenerster
      odate_old LIKE pt-ldate. "Wochenerster alt
```

Vorschlagswerte

Als Vorschlagswerte für den Datenauswahlzeitraum werden Monatserster und Monatsletzter des Vormonats erzeugt. Für die Listüberschrift werden die Systemvariablen `syst-tvar0` und `syst-tvar1` gefüllt.

```
INITIALIZATION.
  pnpbegda = sy-datum.
  pnpbegda+6(2) = '01'.
  PERFORM day_minus_months(sapfp500)
    USING pnpbegda '1' pnpbegda.
```

```
  PERFORM last_day_in_month(sapfp500)
    USING pnpbegda pnpendda.
START-OF-SELECTION.
  WRITE pn-begda TO syst-tvar0 DD/MM/YYYY.
  WRITE pn-endda TO syst-tvar1 DD/MM/YYYY.
```

Neu zum Zeitpunkt `GET pernr` im Vergleich zu den bisher behandelten Beispielen ist die Routine `read_time`, die die Daten der Zeitauswertung listet und ausgibt:

```
GET pernr.
  PERFORM clear_all.
  rp_provide_from_last p0002 space pn-begda pn-begda.
  PERFORM concat_name USING coname.
  FORMAT COLOR COL_GROUP INTENSIFIED ON.
  WRITE: / pernr-pernr,
           coname, 80 space.
  FORMAT COLOR OFF.
  PERFORM read_time.
END-OF-SELECTION.
```

Das Include `rpppxm00` enthält die Standardroutinen für die Pufferung der Daten bei der Benutzung von Makros:

```
  INCLUDE rpppxm00.
```

Die Routine `concat_name` bereitet den Namen auf:

```
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna INTO p_conname
SEPARATED BY ', '.
ENDFORM.
```

Cluster B2

In der Routine `read_time` wird der Key für das Lesen des Clusters B2 der Clustertabelle PCL2 gefüllt. Das Lesen des Clusters erfolgt mit dem Makro `rp-imp-c2-b2`. Falls der Import erfolgreich war, werden die Zeitpaare aus der importierten Tabelle PT gelesen und ausgegeben. Bei einem Wochenwechsel, der mit der Routine `get_first_day_in_week` erkannt wird, erfolgt die Ausgabe einer horizontalen Linie. Für jeden Tag wird die Zeitart »0003« (Rahmenzeit) aus der importierten Tabelle ZES angelistet.

```
FORM read_time.
  b2-key-pernr = pernr-pernr .    "Personalnummer
```

```
  b2-key-pabrp = pnpbegda+4(2).   "Abrechnungsperiode
  b2-key-pabrj = pnpbegda(4).     "Abrechnungsjahr
  b2-key-cltyp = '1'.             "original
  rp-imp-c2-b2.
  IF sy-subrc EQ 0.
    LOOP AT pt.
      PERFORM get_first_day_in_week(sapfp500)
        USING pt-ldate '01' odate.
      IF ( NOT odate_old IS INITIAL )
        AND ( odate_old <> odate ).
        NEW-LINE.
        ULINE AT 35.
      ENDIF.
      odate_old = odate.
      WRITE: /35(12) pt-ldate,
        (10) pt-begtm USING EDIT MASK '__:__:__',
        (10) pt-endtm USING EDIT MASK '__:__:__'.
      LOOP AT zes WHERE reday = pt-ldate+6(2)
        AND ztart = '0003'.
        WRITE: zes-anzhl.
        EXIT.
      ENDLOOP.
    ENDLOOP.
  ELSE.
* ...
  ENDIF.
ENDFORM.
```

6.5 Aufbereitung der Ausgabe mit dem ABAP List Viewer

Die Aufbereitung der in den bisher in diesem Kapitel dargestellten Listen kann durch den Einsatz des ABAP List Viewers (ALV) um weitere Funktionen ergänzt werden. Zum Beispiel können Sortierungen der angezeigten Liste oder Downloads nach Excel durchgeführt werden. Die Anzeige kann individuell konfiguriert und als Anzeigevariante abgespeichert werden. In Abbildung 6.16 wird die aus Abschnitt 6.1 bekannte Liste mit dem Funktionsbaustein `REUSE_ALV_LIST_DISPLAY` aufbereitet.

PersNr	Name	Bankkonto	Bankschl.	Name des Geldinstituts
1001	Maier, Michaela	1258693	20050000	Deutsche Bank Hamburg
1002	Zaucker, Ulrike	253571587	23984899	Vereinsbank München
1003	Pfändili, Stefan	258369741	12388823	Deutsche Bank Paderborn
1004	Paulsen, Olaf	741852963	20050000	Deutsche Bank Hamburg
1005	Gutjahr, Hanno	258963147	23984899	Vereinsbank München
1006	Awad, Yasmin			
1007	Ulrich, Hanna	582369741	23022200	Volksbank Marschfelden
1008	Müller, Hilde	123456987	32902212	Raifeisenbank Breitenrunn
1009	Braunstein, Herbert	564789123	10050033	Dresdner Bank Berlin
1010	Schmidtrohr, Frank			
1011	Förster, Claudia	123123	67292200	Volksbank Wiesloch
1014	Hintze, Gudrun	12581541	20050000	Deutsche Bank Hamburg
1015	Rickes, Alexander	1569874256	23022200	Volksbank Marschfelden
1016	Kaufman, Mike	21343214	12388823	Deutsche Bank Paderborn
1017	Sturm, Annette	1234145252	23022200	Volksbank Marschfelden

Abbildung 6.16 Ausgabe der Liste mit REUSE_ALV_LIST_DISPLAY

Eine weitere Möglichkeit bietet der Funktionsbaustein `REUSE_ALV_GRID_DISPLAY`. Das Ergebnis ist in Abbildung 6.17 zu sehen.

PersNr	Name	Bankkonto	Bankschl.	Name des Geldinstituts
1001	Maier, Michaela	1258693	20050000	Deutsche Bank Hamburg
1002	Zaucker, Ulrike	253571587	23984899	Vereinsbank München
1003	Pfändili, Stefan	258369741	12388823	Deutsche Bank Paderborn
1004	Paulsen, Olaf	741852963	20050000	Deutsche Bank Hamburg
1005	Gutjahr, Hanno	258963147	23984899	Vereinsbank München
1006	Awad, Yasmin			
1007	Ulrich, Hanna	582369741	23022200	Volksbank Marschfelden
1008	Müller, Hilde	123456987	32902212	Raifeisenbank Breitenrunn
1009	Braunstein, Herbert	564789123	10050033	Dresdner Bank Berlin
1010	Schmidtrohr, Frank			
1011	Förster, Claudia	123123	67292200	Volksbank Wiesloch
1014	Hintze, Gudrun	12581541	20050000	Deutsche Bank Hamburg
1015	Rickes, Alexander	1569874256	23022200	Volksbank Marschfelden
1016	Kaufman, Mike	21343214	12388823	Deutsche Bank Paderborn

Abbildung 6.17 Ausgabe der Liste mit REUSE_ALV_GRID_DISPLAY

TYPE-POOLS

Für die Verwendung des ALV ist die Deklaration des `TYPE-POOLS: slis` notwendig. Die im Report definierte Struktur der Ausgabe `display` wird verwendet, um die interne Tabelle `t_display` und den Arbeitsbereich `w_display` zu definieren. Weiterhin werden der Feldkatalog `alv_fieldcat` und Angaben für das Layout und die Variantenspeicherung definiert:

```
REPORT  yrhr_bank_alv.
TYPE-POOLS: slis.
```

```
TABLES: pernr.
INFOTYPES: 0002, 0009.
DATA: coname(40),      "Name, Vorname
      bankname(40).    "Bankname
TYPES: BEGIN OF display,
         pernr LIKE pernr-pernr,
         coname(40),
         bankn LIKE p0009-bankn,
         bankl LIKE p0009-bankl,
         bankname(40),
       END OF display.
DATA: t_display TYPE TABLE OF display,
      w_display TYPE display.
DATA: alv_fieldcat TYPE slis_t_fieldcat_alv,
      w_alv_fieldcat TYPE LINE OF slis_t_fieldcat_alv,
      alv_layout TYPE slis_layout_alv,
      alv_variant TYPE disvariant.
INITIALIZATION.
  PERFORM last_day_in_month(sapfp500)
    USING sy-datum pnpbegda.
START-OF-SELECTION.
```

GET pernr Zum Zeitpunkt `GET pernr` werden die bekannten Verarbeitungen durchgeführt. Im Unterschied zu der einfachen Liste wird die Ausgabe in die interne Tabelle `t_display` unter Verwendung des Arbeitsbereiches `w_display` vorgenommen. Zur Verdeutlichung wird hier die Zuweisung für jedes einzelne Feld statt `MOVE-CORRESPONDING` verwendet:

```
GET pernr.
  PERFORM clear_all.
  rp_provide_from_last p0002 space pn-begda pn-begda.
  rp_provide_from_last p0009 '0' pn-begda pn-begda.
  PERFORM concat_name USING coname.
  PERFORM read_bankname USING bankname.
  MOVE: pernr-pernr TO w_display-pernr,
        coname TO w_display-coname,
        p0009-bankn TO w_display-bankn,
        p0009-bankl TO w_display-bankl,
        bankname TO w_display-bankname.
  APPEND w_display TO t_display.
END-OF-SELECTION.
```

Am Ende der Verarbeitung ist die Tabelle `t_display` gefüllt und kann mit der Routine `show_liste` ausgegeben werden:

Ausgabe

```
  PERFORM show_liste.
```

Die unten aufgeführten Routinen sind bereits bekannt.

```
FORM clear_all.
  CLEAR: coname, bankname.
ENDFORM.
FORM concat_name USING p_conname.
  CONCATENATE p0002-nachn p0002-vorna
    INTO p_conname SEPARATED BY ', '.
ENDFORM.
FORM read_bankname USING p_bankname.
  DATA : bankdata TYPE bnka_bf.
  CALL FUNCTION 'HRCA_READ_BANK_ADDRESS_2'
    EXPORTING
      bank_country = p0009-banks
      bank_number  = p0009-bankl
    IMPORTING
      bank_data    = bankdata
    EXCEPTIONS
      not_found    = 1
      OTHERS       = 2.
  IF sy-subrc = 0.
    p_bankname = bankdata-banka.
  ELSE.
    CLEAR p_bankname.
  ENDIF.
ENDFORM.
```

Neu ist die Verarbeitung in der Routine `show_liste`. Für die Anzeige des Titels werden die Variablen `grid_titel` und `list_begda` definiert und gefüllt:

```
FORM show_liste.
  DATA: grid_title TYPE lvc_title,
        list_begda(10).
  WRITE pn-begda TO list_begda DD/MM/YYYY.
  CONCATENATE 'Bankverbindungen der Mitarbeiter am'
    list_begda INTO grid_title SEPARATED BY space.
```

Layout Für das gewünschte Layout werden die geeigneten Übergabeparamter der Strukur `alv_layout` angegegeben.

```
alv_layout-colwidth_optimize = 'X'.
alv_layout-zebra = 'X'.
```

Feldkatalog Für die einzelnen Spalten der Ausgabe werden die speziellen Eigenschaften definiert und in den Feldkatalog aufgenommen. Die Bedeutung der Felder ist der Dokumentation des weiter unten aufgerufenen Funktionsbausteins zu entnehmen.

```
CLEAR w_alv_fieldcat.
w_alv_fieldcat-fieldname = 'PERNR'.
w_alv_fieldcat-ref_tabname = 'PERNR'.
w_alv_fieldcat-key = 'X'.
APPEND w_alv_fieldcat TO alv_fieldcat.
CLEAR w_alv_fieldcat.
w_alv_fieldcat-fieldname = 'CONAME'.
w_alv_fieldcat-reptext_ddic = 'Name'.
APPEND w_alv_fieldcat TO alv_fieldcat.
CLEAR w_alv_fieldcat.
w_alv_fieldcat-fieldname = 'BANKN'.
w_alv_fieldcat-ref_tabname = 'P0009'.
APPEND w_alv_fieldcat TO alv_fieldcat.
CLEAR w_alv_fieldcat.
w_alv_fieldcat-fieldname = 'BANKL'.
w_alv_fieldcat-ref_tabname = 'P0009'.
APPEND w_alv_fieldcat TO alv_fieldcat.
CLEAR w_alv_fieldcat.
w_alv_fieldcat-fieldname = 'BANKNAME'.
w_alv_fieldcat-ref_tabname = 'BNKA_BF'.
w_alv_fieldcat-ref_fieldname = 'BANKA'.
APPEND w_alv_fieldcat TO alv_fieldcat.
```

Nachdem der Feldkatalog vollständig gefüllt worden ist, kann jetzt der Funktionsbaustein `REUSE_ALV_GRID_DISPLAY` aufgerufen werden und die Ausgabe übernehmen. In der Parameterliste sind auch die Parameter `i_save` und `is_variant` aufgeführt, die für die Abspeicherung der Listvarianten erforderlich sind.

```
CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
```

```
      i_callback_program = 'YTHR_BANK_ALV'
      i_grid_title       = grid_title
      is_layout          = alv_layout
      it_fieldcat        = alv_fieldcat
      i_save             = 'A'
      is_variant         = alv_variant
    TABLES
      t_outtab           = t_display
    EXCEPTIONS
      program_error      = 1
      OTHERS             = 2.
  IF sy-subrc <> 0.
    WRITE: / 'Fehler bei Aufbereitung'.
  ENDIF.
ENDFORM.                    "show_liste
```

Ähnliche Aufbereitungsmöglichkeiten bietet der Funktionsbaustein `HR_ALV_LIST_DISPLAY`. Dieser Funktionsbaustein kapselt den Aufruf des Funktionsbausteins `REUSE_ALV_LIST_DISPLAY` für Anwendungen im HR.

Index

Q

R

S

T

U