ABAP Course

Chapter 4: Database access

Content
The fourth chapter focuses on the database accesses. You will learn how to use the data dictionary to gain information about data elements, domains and so on. Moreover, you will develop your first ABAP program, which uses database operations.

Prerequisites
Before starting the exercises you should be familiar with SQL and the basic concepts of ABAP programs.

Motivation
This chapter explains the basic access to the database by using OpenSQL. You will have a look into the hierarchy of data elements in the SAP system. Hence this chapter forms the fundament for developing more complex SAP applications later on.

Lecture notes
All students should be familiar with SQL and the basic concepts of a database as the chapter builds upon this knowledge. Students can go on with their account from chapter 1.

- **Product**: All
- **Level**: Beginner
- **Focus**: Programming
- **Version**: 1.0
- **Author**: UCC Technische Universität München

Task 1: Login into the SAP system

Short description: Use SAPGui to login into the SAP system with your username and password

Login

Start the SAPGui and login into the development system using the provided account and password. Please refer to chapter 1 for your username and your password.

Task 2: First steps in the data dictionary

Short description: Use the data dictionary to explore the structure of table 'SCARR'

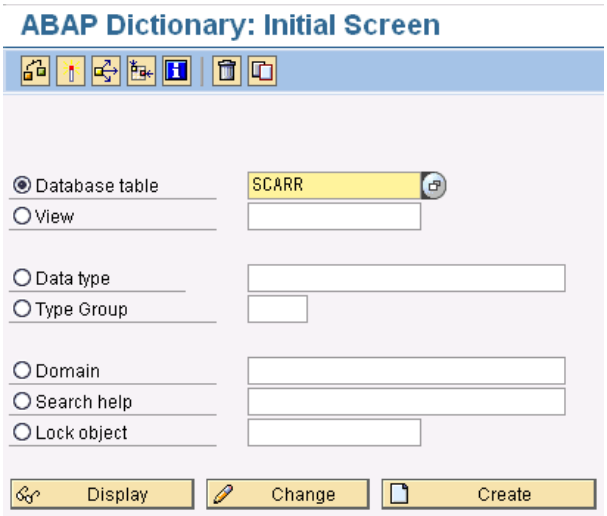
Please start the Data dictionary from the SAP Easy Access Menu by using the following path:

Menu path

Tools • ABAP Workbench • Development • Dictionary

You may also use the transaction code **SE11** for direct access.

Choose the option '**Database table**' and type in the table name '**SCARR**'.

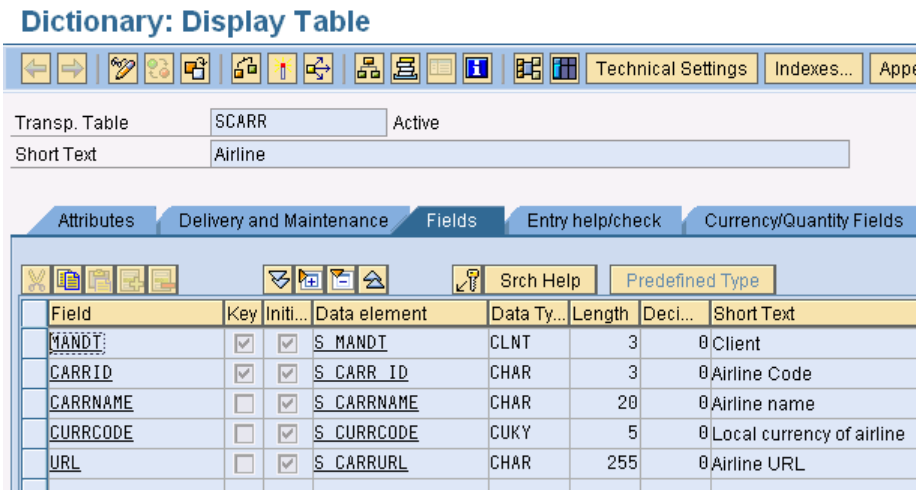


Display table

Then use the '**Display**' button to have a look at the table definition. The SAP system brings you directly to the field definition of the table. In this tab you see all the data elements which form the table. Moreover you see which data element is the key element. Beside the fields tab, there are also several other tabs available:

- Attributes: The attributes show you to which package the table is assigned and who did the last changes on which date.
- Delivery and maintenance: This is a very important tab as it shows you to which delivery class the table is assigned and if changes to the table are allowed or not.
- Entry help/check: This tab shows you if any 'check tables' are used and if there is a 'search help' available.
- Currency/quantity fields: The last tab is important when the table contains currency data as it is necessary to choose a reference currency from a customizing table then.

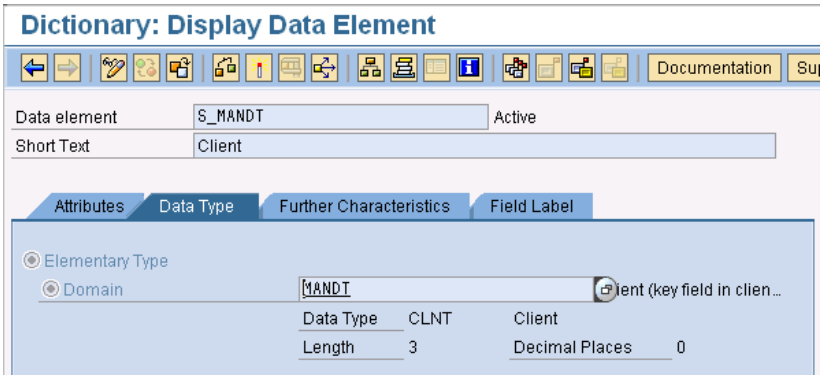
Please switch back to the '**Fields**' tab so you can have a look at the table fields.



You see, that the table uses field '**MANDT**' and '**CARRID**' as key fields. This is indicated by the Key checkbox. Moreover, you see all the different data elements. For example the field '**MANDT**' uses the data element '**S_MANDT**' which is linked to the data type '**CLNT**' and has a length of 3.

Forward navigation

If you want to explore the data element '**S_MANDT**' now, use the forward navigation of the SAP system and double click on the data element. The data dictionary now jumps directly to the definition of the data element and comes up with the details of the definition.

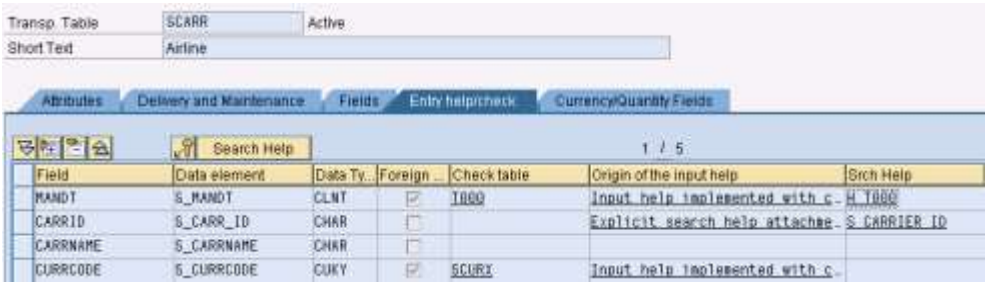


Hint

Hint:
Please note that the data element and the table are both in the active state. This is indicated by the '**Active**' text next to the table name and the data elements name. When changing a table or data element in the dictionary the state is changed to '**Inactive**'. If an active version of the table or data element is still in the dictionary, all programs use the active version until the inactive version is activated successfully.

Entry help

Please return to the table definition using the '**Back**' button and switch to the tab '**Entry help/check**'. This tab gives you information about implemented entry help and check tables. When working with the table later on, the user may choose an appropriate value to be inserted into the table. To ensure the table integrity the entry help only shows appropriate values. The entry help is built upon the check table. This table only contains appropriate values.



Display graphic

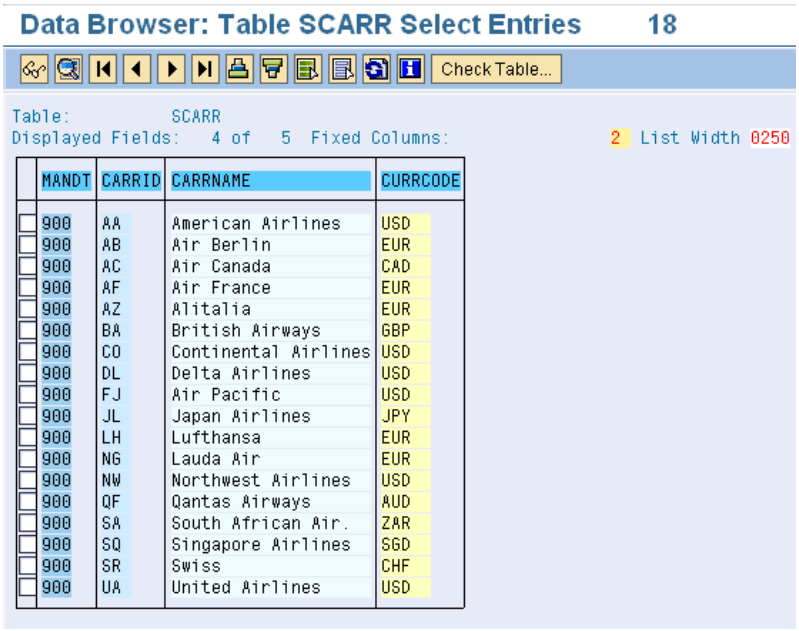
In the next step you want to check the foreign key relations. To do so you will consult the graphical representation of the table. The graphical representation can be called by pressing the '**Graphic**' button  . This button can be found in the toolbar. By pressing the button an additional SAPGui program is started, which is called 'SAP Graphical Editor'. If you do not see a graphic you have to install the 'SAP Graphical Editor' first. To return to SAPGui please use the '**Back**' button (F3).

Checking the table content is very important from time to time. You can do this very easily by choosing the following menu path (from the Data Dictionary screen):

Menu path

Utilities • Table Contents • Display

This menu path does not bring you directly to the table content but jumps into the data browser transaction (transaction code **SE16**). All you have to do now is pressing the '**Execute**' button (F8). This triggers the SAP system to read all the data from table '**SCARR**' and display it. The result should look similar to this:



Task 3: Read and display data from table

Short description: Use SQL statements to read data from a table and display the data using an ABAP program

Before starting programming, have a look at the table '**SPFLI**'. The table contains data about flights, which will be displayed in your program later. Please ensure you are familiar with the table definition, especially the primary and foreign keys.
Start the Object Navigator from the SAP Easy Access Menu by using the following path:

Tools • ABAP Workbench • Overview • Object Navigator

Menu path

Create a new program called '**ZY_##_FLIGHTS**' in your existing package **ZY_##**. Do not use TOP INCLUDE, assign the status test program and use your existing transport request.

In the first section of your new program you want to define the variables which are needed later on. You need two variables: it_flights as an internal table and wa_flight as a work area. The internal table it_flights will contain the entire content of the database table '**SPFLI**' whereas the work area wa_flight only contains one single data set from the internal table.

Define variables

```
*&---------------------------------------
*& Report  ZY_99_FLIGHTS
*&
*&---------------------------------------
*&
*&
*&---------------------------------------

REPORT  zy_99_flights.

DATA it_flights TYPE TABLE OF spfli.
DATA wa_flight TYPE spfli.
```

As you can see you define it_flights as a type table of '**SPFLI**'. This means the SAP system consults the data dictionary for the structure of the database table SPFLI and creates an internal table with the same structure like SPFLI. The work area wa_flight has the same structure like SPFLI but is not a table, only a single record.

As you want to display the content of the table in your program you have to read all the content from the table. This is done by implementing a simple SQL statement which reads the content from SPFLI into your internal table it_flights.

SQL statement

```
SELECT * FROM spfli INTO TABLE it_flights.
```

In the last step we want to display the content from it_flights and therefore we loop through the internal table, read the current data record and write it into our work area. The content from the work area will be written to the console.

Generate output

```
LOOP AT it_flights INTO wa_flight.
  WRITE:/ wa_flight-connid, wa_flight-cityfrom, wa_flight-countryfr,
          wa_flight-cityto, wa_flight-countryto.
ENDLOOP.
```

Save, check and activate your program now. The output should look similar to this:

Save, check, activate

Program ZY_99_FLIGHTS

```
Program ZY_99_FLIGHTS
-------------------------------------------------------------------------------
0017 NEW YORK             US  SAN FRANCISCO       US
0064 SAN FRANCISCO        US  NEW YORK            US
0555 ROME                 IT  FRANKFURT           DE
0788 ROME                 IT  TOKYO               JP
0789 TOKYO                JP  ROME                IT
0790 ROME                 IT  OSAKA               JP
0106 NEW YORK             US  FRANKFURT           DE
1699 NEW YORK             US  SAN FRANCISCO       US
1984 SAN FRANCISCO        US  NEW YORK            US
0407 TOKYO                JP  FRANKFURT           DE
0408 FRANKFURT            DE  TOKYO               JP
0400 FRANKFURT            DE  NEW YORK            US
0401 NEW YORK             US  FRANKFURT           DE
0402 FRANKFURT            DE  NEW YORK            US
2402 FRANKFURT            DE  BERLIN              DE
2407 BERLIN               DE  FRANKFURT           DE
0005 SINGAPORE            SG  FRANKFURT           DE
0006 FRANKFURT            DE  SINGAPORE           SG
0002 SINGAPORE            SG  SAN FRANCISCO       US
0015 SAN FRANCISCO        US  SINGAPORE           SG
0158 SINGAPORE            SG  JAKARTA             ID
0988 SINGAPORE            SG  TOKYO               JP
0941 FRANKFURT            DE  SAN FRANCISCO       US
3504 SAN FRANCISCO        US  FRANKFURT           DE
3516 NEW YORK             US  FRANKFURT           DE
3517 FRANKFURT            DE  NEW YORK            US
```

sy-subrc

The program seems to work fine. But how can you determine whether the SQL statement was executed successfully? For this purpose SAP provides you with the system variable '**sy-subrc**'. You may use the variable to determine if the last action/step in your program was successful or not. You will do this now in your program using an **if-else**-branch. Please use the if-branch to determine if the SQL statement was executed successfully. In this case the program writes all the data to the output whereas when the SQL statement was not executed successfully, the program should return an error message.

```
SELECT * FROM spfli INTO TABLE it_flights.

IF sy-subrc = 0.
  LOOP AT it_flights INTO wa_flight.
    WRITE:/ wa_flight-connid, wa_flight-cityfrom, wa_flight-countryfr,
            wa_flight-cityto, wa_flight-countryto.
  ENDLOOP.
ELSE.
  WRITE: 'The SQL statement was not executed successfully. Please try again later.'.
ENDIF.
```

It is necessary to place the if-branch directly after the SQL statement as the variable '**sy-subrc**' only contains the return code of the last instruction. Of course, this is just a very simple error handling but it shows you how to use the system variable to react on failures during runtime. Another one is using exceptions which will be discussed in a later chapter.

Task 4: Use domains, data elements and entry helps

Short description: Use the data dictionary to create an entry help for a new table

Please start the ABAP dictionary from the SAP Easy Access Menu by using the following menu path:

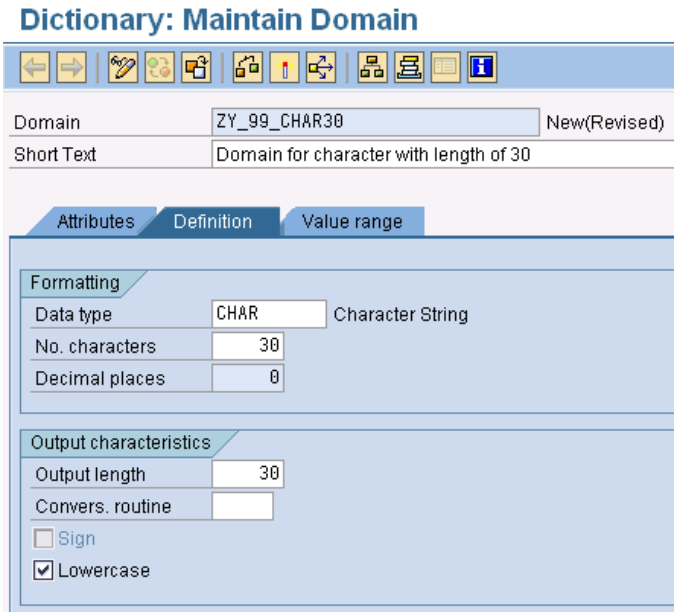
Menu path

Tools • ABAP Workbench • Development • ABAP Dictionary

You may also use transaction code **SE11** for direct access.

In the first step you will create a new domain as this is the highest hierarchy level in the ABAP dictionary. Your new domain is named '**ZY_##_CHAR30**'. Select the '**Domain**' radio button and type in the name into the input field. Then click on the '**Create**' button. The SAP system will bring you directly to the domain details where you have to define a short text and a data type. You can choose a short text on your

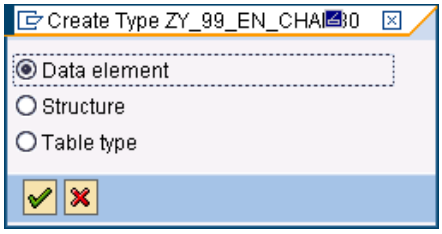
own. We want to use the data type '**CHAR**' in our domain and the number of characters is limited to '**30**'.



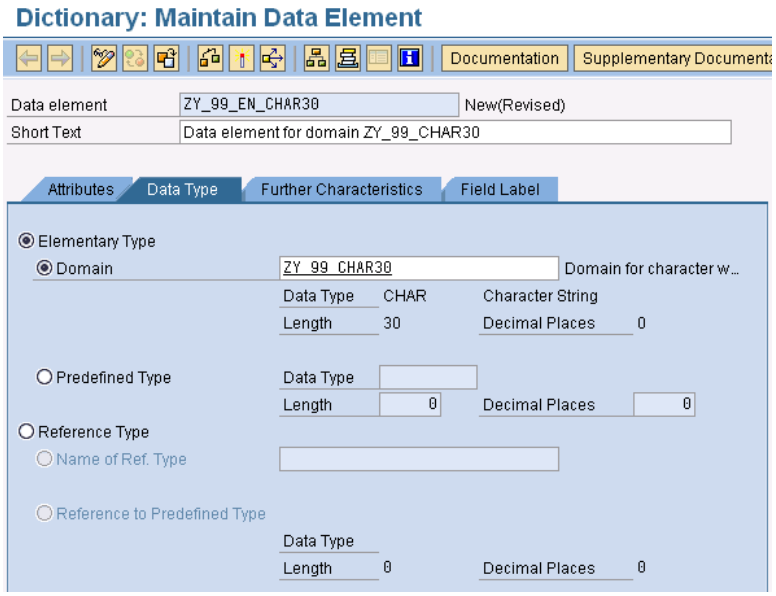
Save and activate

Now save and activate your new domain. Return to the ABAP dictionary by pressing the '**Back**' button (F3). In order to avoid problems please leave the SE11 transaction and re-enter it.

In the next step you want to create a new data type named '**ZY_##_EN_CHAR30**'. Select the radio button '**Data type**' and type in the name of your new data type.

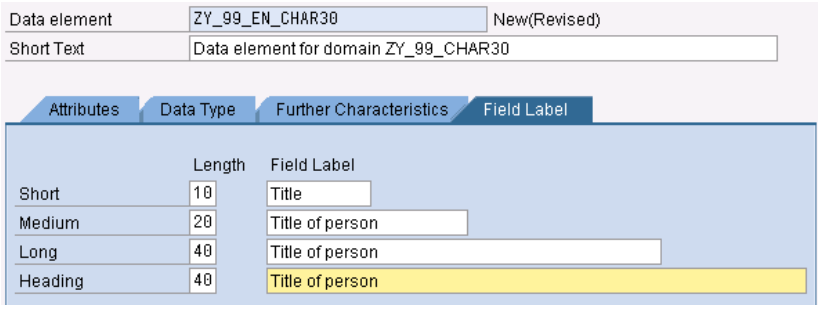


The system asks you now if you want to create a Data element or a structure or a table type. Please select the first option '**Data element**'. Maintain the short text and the domain of your new data element. Choose the domain you created in the first step of this task.



Maintain 'Field Label'

Switch to the '**Field Label**' tab and maintain the field labels, too. You can define the maximum length of each field label.



Save and activate

Save and activate your new data element using your transport request and package. In order to avoid problems please leave the SE11 transaction and re-enter it.

Create table

The next step is the creation of your database table. The table will contain some exemplary titles of persons and will therefore function as a check table. The name of the new table is '**ZY##_TITLE'**. Please note that because of naming conventions there is no underscore after ZY. Maintain the short text and then choose '**Application table (master and transaction data)**' as the '**Delivery Class**' and '**X Display/Maintenance Allowed**' as the value for '**Data Browser/Table View Maint.**'.

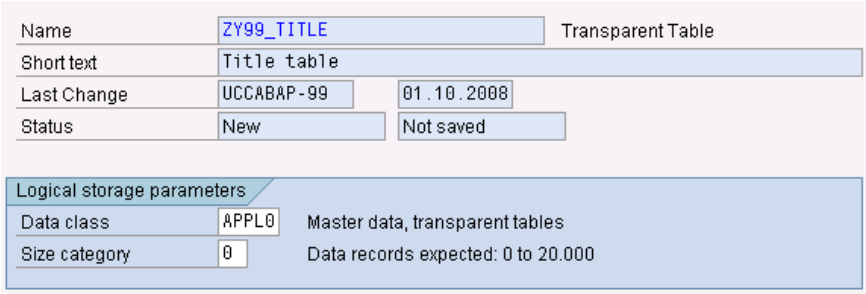
Tab Fields

Switch to tab '**Fields**' to maintain the fields of your table and define the first field '**Title**'. The title field is also used as the primary key. So select the checkbox '**Key**'. For the first field you have to define the data element. Here you choose the created data element '**ZY_##_EN_CHAR30'**. Please use the F4 help to avoid any typing failures.

Technical settings

Now switch to the technical settings of your table by using the button: Technical Settings

Now you are asked if you would like to save your table. You can confirm this and assign your existing transport request.
The technical settings describe how the table will be stored later on and if data records of the table are buffered or not. The maintenance of the data class is mandatory: please choose '**APPL0**' and the size category '**0**'.



Leave the rest of the settings as they are and save the technical settings. After you saving the settings, go back (F3) to the table maintenance.

In order to classify your table in terms of possible enhancements you need to choose

Extras • Enhancement Category

Menu path

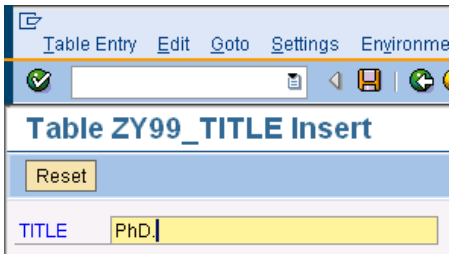
Confirm the information dialog that the table is not classified yet. Choose "cannot be enhanced" and confirm the dialog. Save and activate your table now.

In the next step you will enter some data records into the table. This can be done easily using the '**Data browser**'. The Data browser can be accessed via the menu path:

Utilities • Table Contents • Create Entries

Menu path

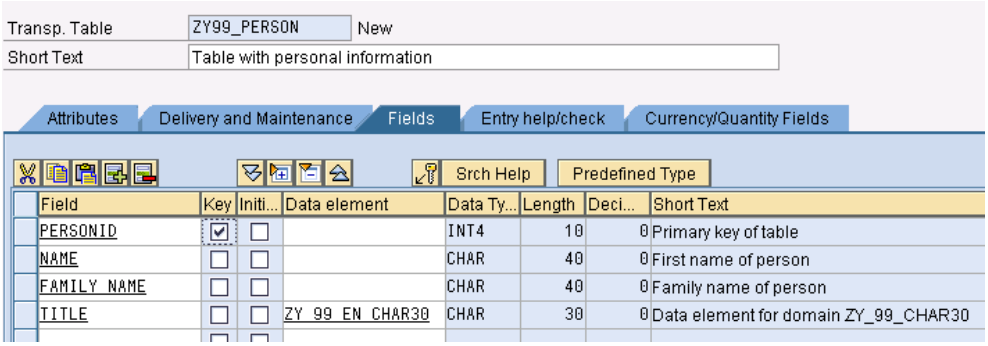
The SAP system jumps directly to a new program where you can add new data records into the table. To add a new entry into the table, type in the title into the input field and then press the '**Save**' button.



Insert data record

Add the following titles: PhD., Dr., Prof. Dr., Prof. Dr. h.c. mult. After you entered all titles please go back to the data dictionary. In order to avoid problems please leave the SE11 transaction and re-enter it.

In the next step we want to create another table named '**ZY##_PERSON**'. The table will contain data about persons. You will integrate the previous generated table '**ZY##_TITLE**' into the new table by using a foreign key. Choose the same Delivery and Maintenance settings as in the previous table. Create the table using the following fields: PERSONID, NAME, FAMILY_NAME and TITLE. Refer to the following figure:

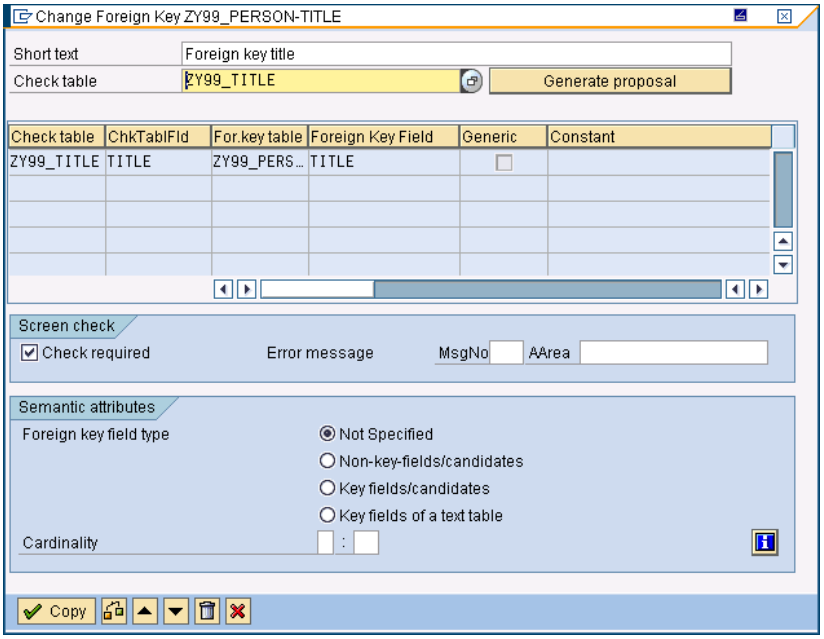


Hint:
When defining table fields you may either use predefined data types or data elements. The best way to define your table fields is to define all fields using predefined data types at first and then define table fields using data elements. Use the button Predefined Type / Data Element to switch the modus.

Hint

Now you want to define the foreign key connection between both tables whereas the '**TITLE**' field should be linked to the table '**ZY##_TITLE**'. Therefore select the field '**TITLE**' and press the '**Foreign Key**' button in the toolbar. The SAP system comes up with a pop-up and asks you for the check table. Please select your title table '**ZY99_TITLE**' and press '**Enter**'. The system will automatically read the repository information and will propose a foreign key definition using the field names from both tables.

Foreign key



Please choose the same technical settings and the same enhancement category as for the title table.

Save and activate your new table after you maintained the technical settings. To prove if the check table was defined successfully you will now create a program. Leave the data dictionary.

Please start the Object Navigator from the SAP Easy Access Menu by using the following path:

Tools • ABAP Workbench • Overview • Object Navigator

Menu path

You may also use the transaction code **SE80** for direct access.

Create a new program which is named '**ZY_##_PERSON**'. Use the following code fragment to define parameters in your program:

```
*&---------------------------------------------------------------------*
*& Report  ZY_99_PERSON
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

REPORT  ZY_99_PERSON.

parameters: pa_title type zy99_person-title, pa_name type zy99_person-name, pa_fam type zy99_person-family_name.
```

Save, check, activate, test

Save, check and activate your new program. Now when testing your new program the SAP system comes up with an entry help on the '**PA_TITLE**' input field. This is because of the foreign key definition. All the values come from the title table.