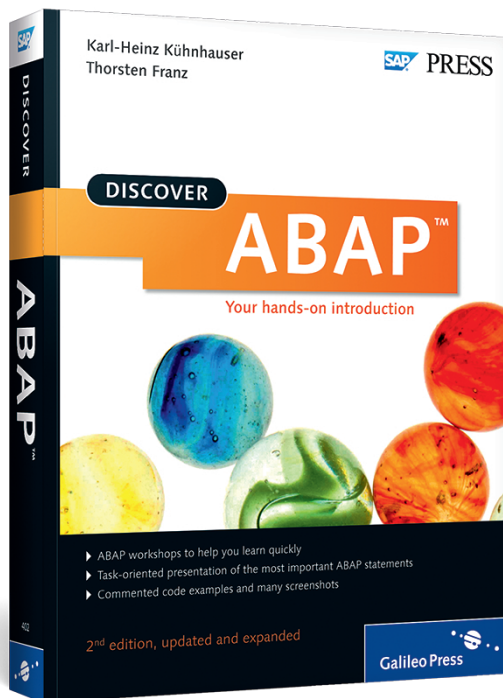


Karl-Heinz Kühnhauser and Thorsten Franz

Discover ABAP™




Galileo Press

Bonn • Boston

Contents at a Glance

1	ABAP and Getting Started with the SAP System	21
2	ABAP Dictionary	45
3	Programming in the ABAP Editor	71
4	Fields and Calculations	99
5	Modifying Character Strings	121
6	Debugging Programs	141
7	Modifying Transparent Database Tables	173
8	Calculating Dates, Times, Quantities, and Currencies	213
9	Using Data in a Database Table	243
10	Program Flow Control and Logical Expressions	275
11	Selection Screens	323
12	Internal Tables	381
13	Modularizing Programs	439
14	Advanced Topics	503

Contents

Preface to the Second Edition	15
-------------------------------------	----

1 ABAP and Getting Started with the SAP System ... 21

Overview of the Architecture of an SAP System	23
Technical Architecture	23
Business Organization Architecture	25
Platform-Independence	28
Application Programs and Runtime Environment	28
Work Processes	29
Structure of ABAP Programs	31
Logging On and Off the System	34
Overview of Business Components	36
ABAP Workbench	37
Logging off from the SAP System	42

2 ABAP Dictionary 45

Getting Started with the ABAP Dictionary	46
Database Tables	46
Creating and Maintaining Tables	47
Data Elements and Domains	52
Creating a Data Element	53
Creating Domains	57
Checking and Activating a Data Element	62
Maintaining the Technical Settings of the Table	64
Creating Data Records	66
Entering Data Records	67
Displaying the Contents of the Table	68

3 Programming in the ABAP Editor 71

Creating an ABAP Report	71
-------------------------------	----

ABAP Editor: Overview	75
Modes of the ABAP Editor	75
Controlling the ABAP Editor	77
Understanding and Editing ABAP Programs	82
Executing an ABAP Report	84
Reading and Outputting Database Tables	86
Formatting Lists	88
Chain Statement	89
Lines	89
Blank Lines	89
Writing and Editing Source Code	90
Notes on the Source Code	91
List Screen from Our Sample Source Code	96

4 Fields and Calculations 99

Preparing the Report	99
Declaring Fields	104
Declaring Variables	104
Declaring Constants	109
Basic Arithmetic Operations	110
Compatible and Convertible Data Objects	111
Conversion Rules	112
Special Features of Division Operations	114
Sample Code for Fields and Calculations	115
Notes on the Source Code	116
Improved List Format	119

5 Modifying Character Strings 121

Declaring Character Strings	121
String Operations	124
Shifting Character Strings	125
Replacing Character Strings	126
Condensing Character Strings	128
Concatenating String Fields	129
Splitting Character Strings	131

String Operations with Direct Positioning	132
Sample Code for String Operations	133
Notes on the Source Code	135
Program Output	139

6 Debugging Programs 141

Overview	141
Calling the ABAP Debugger	142
Working with the ABAP Debugger	147
Desktop 1	148
Structures Tab	152
Break-/Watchpoints Tab	154
Breakpoints Mode	157
Static Breakpoints	160
Layer for Layer: Layer-Aware Debugging	160
Sample Code for Layer-Aware Debugging	168
Notes on the Source Code	170

7 Modifying Transparent Database Tables 173

Copying a Database Table	175
Enhancing Non-Key Fields	180
Maintaining Fixed Values in Domains	180
Important Points for Currency and Quantity Fields	183
Maintaining Foreign Keys	186
Maintaining Append Structures	193
Maintaining an Include Structure	197
Manipulating Key Fields of Tables	203
Deleting Table Fields	207
Deleting Tables	209

8 Calculating Dates, Times, Quantities, and Currencies 213

Field Definitions	213
Using Date Fields in Arithmetic Operations	216
Using Time Fields in Arithmetic Operations	223

Using Quantity and Currency Fields in Arithmetic	
Operations	227
Sample Code for Date, Time, and Currency Fields	229
Notes on the Source Code	234
Program Output	240

9 Using Data in a Database Table 243

Authorization Concept	244
Lock Concept	246
OpenSQL Statements	249
Creating a New Data Record	250
Modifying an Existing Data Record	254
Modifying a Data Record	254
Deleting a Data Record	255
Comfortable Alternative: Object Services	257
Sample Code for INSERT	258
Notes on the Source Code	260
Program Output	262
Sample Code for UPDATE	263
Notes on the Source Code	264
Program Output	266
Sample Code for MODIFY	266
Notes on the Source Code	268
Program Output	269
Sample Code for DELETE	270
Notes on the Source Code	271
Program Output	272

10 Program Flow Control and Logical Expressions 275

Control Structures	276
Using Patterns	277
Branches	280
IF Structure	281
CASE Structure	284
Loops	287

SELECT Loop	287
DO Loop	287
WHILE Loop	290
Termination Statements for Loops	291
Logical Expressions	295
Simple Logical Expressions	295
Linked Logical Expressions	298
Sample Code for IF	302
Notes on the Source Code	304
Program Output	306
Sample Code for CASE	306
Notes on the Source Code	308
Program Output	309
Sample Code for DO and Termination Conditions	311
Notes on the Source Code	313
Program Output	315
Sample Code for WHILE and Logical Expressions	316
Notes on the Source Code	319
Program Output	321

11 Selection Screens 323

Events	325
Order of Events	326
Examples of Events	327
Simple Selections	328
PARAMETERS Statement	329
Additions to the PARAMETERS Statement	330
Complex Selections	336
SELECT-OPTIONS Statement	336
Multiple Selections	338
Additions to the SELECT-OPTIONS Statement	341
Using Selection Texts	342
Overview of Text Elements	342
Creating Selection Texts	342
Saving the Selection Screen	346
Creating Selection Variants	347
Starting a Report with a Variant	352

Completing Text Objects	354
Creating Text Symbols	354
Creating Messages	355
Free Layout of the Selection Screen	360
Formatting Single Lines	360
Formatting a Line Block	362
Sample Code for Selection Screen (Simple Form)	364
Notes on the Source Code	366
Program Output	368
Sample Code for Selection Screen (Extended Form)	369
Notes on the Source Code	373
Program Output	377

12 Internal Tables 381

Purpose of Internal Tables	382
Structure and Types of Internal Tables	384
Creating an Internal Standard Table	387
Object-Oriented Syntax with Work Area	387
Obsolete Syntax with Header Line	390
Filling an Internal Standard Table	392
Filling an Internal Table with a Work Area	392
Filling an Internal Table with a Header Line	396
Processing an Internal Table Line by Line	401
Processing an Internal Table with a Work Area	402
Processing an Internal Table with a Header Line	405
Deleting the Contents of Internal Tables	412
Deleting Work Areas and Internal Tables with a Work Area	412
Deleting an Internal Table with a Header Line	413
Sample Code for ITAB with Work Area	415
Notes on the Source Code	419
Tracing the Output of the Source Code in the ABAP Debugger	421
Sample Code for ITAB with Header Line	424
Notes on the Source Code	429
Tracing the Output of the Source Code in the ABAP Debugger	432

13 Modularizing Programs	439
Overview	440
Source Code Modules	442
Procedures	445
Subroutines	445
Function Modules	456
ABAP Classes	471
Memory Areas for Data Transfer	479
Global SAP Memory	479
Local SAP Memory	480
ABAP Memory	480
Shared Objects	481
Sample Code for Modularization	482
Notes on the Source Code	492
Program Output	495
Sample Code for Calling an External Report	496
Notes on the Source Code	499
Program Output	501
 14 Advanced Topics	 503
Interesting Times for the ABAP Programming Language	503
Programming Using Frameworks	504
Example of an Archiving Solution	506
Draft of a Possible Archiving Solution	508
Important Frameworks in the SAP Standard	508
Web Dynpro ABAP	509
Web Services	511
Frameworks for Enhancements	514
Discover!	517
 Index	 519

ABAP Dictionary

Even though the standard delivery of the SAP software contains thousands of tables, customers need their own tables to meet their own needs. That's why we'll take a look at the most important work involved, starting with the creation and maintenance of tables.

In this chapter, we will create and populate a table that you'll access again in later chapters. It is the basic table of our example, member management, in which you create, change, and maintain a data record for each member. With each step you'll build on what you learned from previous ones, so you'll be prepared for more difficult steps down the line.

Step by step



Tables in Real Life

You won't maintain data in SAP software in a single table, but in many tables that are linked by relationships to a complex data model. To learn the basic principles, however, working with one table will suffice.

Getting Started with the ABAP Dictionary

Transaction SE11 When you navigate to the ABAP Dictionary via SAP MENU • TOOLS • ABAP WORKBENCH • DEVELOPMENT • DICTIONARY (or using Transaction SE11), the system displays the initial screen of the ABAP Dictionary (see Figure 2.1).

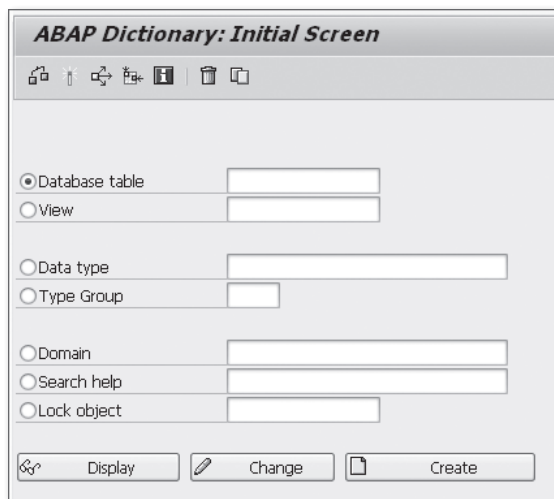


Figure 2.1 Getting Started with the ABAP Dictionary

Database Tables

Dictionary objects From the initial screen, you can display, change, and create a dictionary object such as a database table, view, data type, type group, domain, search help, or lock object. For now, we're interested only in working with database tables.

There are three types of database tables in SAP software:

- › Transparent tables
- › Pool tables
- › Cluster tables

Pool and cluster tables

With pool and cluster tables, the data is stored in an assigned table pool or cluster in the physical database. Both table types store special data (such as program parameters, dynpro sequences, and documentation),

and can store data from several tables in common in a pool or cluster. Pool and cluster tables are not normally used to store business application data; this is generally stored in transparent tables.



Transparent Tables

As noted, we will work only with SAP tools to create and maintain objects in the physical database. Transparent tables are the only type of tables that create a 1-1 correspondence between the table definition in the ABAP Dictionary and the physical table definition in the physical database.

A table pool consists of a single table in the physical database. All the tables assigned to the pool are called *pool tables*. The pool table stores all the records of the table in the pool, identified by the names of the pool table and the key fields.

A table cluster consists of several *cluster tables*. A cluster table combines data records with the same key in several cluster tables into a physical data record. This lowers the number of data records but lengthens a data record; the result is that you must often create continuation records.



Only with Open SQL

The 1-1 correspondence found in transparent tables does not apply to pool and cluster tables, so you don't need to maintain any technical settings or indices. The information stored in the ABAP Dictionary is absolutely essential in order to process these types of tables. In an emergency, these types of tables can be processed only with Open SQL, and never with native SQL.

Creating and Maintaining Tables

When you create a new table, you must first think about a name. A table name has a maximum of 16 characters and does not distinguish between uppercase and lowercase. Note that for almost all objects, you must adhere to various *namespaces* for standard SAP objects and customer-specific objects.

Naming conventions for customer objects

According to the naming conventions, the names of customer objects begin with a Z, Y, or a *prefix namespace*. A prefix namespace is a letter sequence enclosed in slashes and uniquely reserved at SAP. It is added as a prefix to the name of the development objects and cannot be used by other SAP customers in their customer-specific objects (for example, /SAPPRESS/). This ensures that object names are unique and that naming conflicts during mergers, for example, do not arise. However, for our training purposes, the well-known customer namespaces Z and Y are sufficient. Enter ZMEMBER01 as the name of your first DATABASE TABLE and click CREATE (see Figure 2.2).

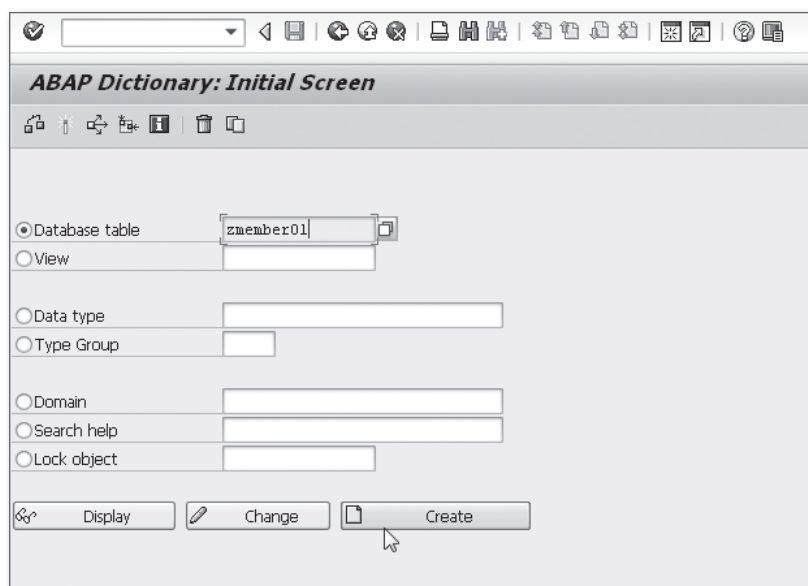


Figure 2.2 Creating a Database Table

Maintenance screen

The maintenance screen of the ABAP Dictionary contains five tabs: ATTRIBUTES, DELIVERY AND MAINTENANCE, FIELDS, ENTRY HELP/CHECK, and CURRENCY/QUANTITY FIELDS. In the SHORT DESCRIPTION on the PROPERTIES tab, enter an informative, explanatory text, such as "Table of Members." You must complete two fields in the DELIVERY AND MAINTENANCE tab:

› Delivery class

The delivery class regulates the transport of the table's data records during installations, upgrades, client copies, and transport into another SAP system. SAP and its customers have different write rights, depending on the delivery class.

Enter an "A" in this field for the application table of master and transaction data. Master data (such as material numbers and personnel numbers) changes only rarely. Transaction data, on the other hand, changes often; inventory line items are a good example.

The tables of the various delivery classes are handled differently during client copies, new installations, and upgrades. For example, a client copy generally does not copy master and transaction data.

› Data Browser/Table View Maintenance

The settings in the DATA BROWSER/TABLE VIEW MAINTENANCE field regulate the scope that you may use to display and maintain data records with standard SAP maintenance tools. If you limit these rights, you might be unable to create any new data records in the dictionary.

That's why you should select DISPLAY/MAINTENANCE ALLOWED here so that you have all the rights you need to work in the dictionary and can create, change, and delete data records.

But before you begin, save all your work up to this point. Click SAVE or press **Ctrl** + **S** (see Figure 2.3).

Saving a table

In the following dialog, you make an important decision: whether or not your table can be transported into another SAP system. If it can be transported, you must specify a package. A package bundles related objects of the ABAP Workbench and defines the transport layer. The transport layer determines whether objects will be assigned to a transportable change request for a target system, or will instead remain in the local SAP system for the time being.

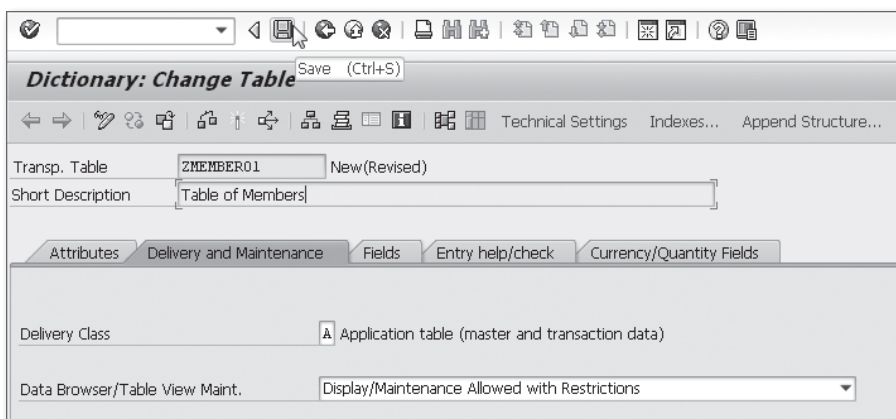


Figure 2.3 Saving the Table



To Transport or Not to Transport?

Carefully consider this. In the real world—that is, in a multilevel system landscape—you would request the name of the package from system administrators and then enter that name because you're creating objects on the test and development system that you'll later use in the production system. For this exercise, that's unnecessary. The table and the initial reports can remain on the local test and development system.

Save the table as a **LOCAL OBJECT** (see Figure 2.4). It is automatically assigned to package `$TMP`. Objects of package `$TMP` are never transported and are not subject to version management.

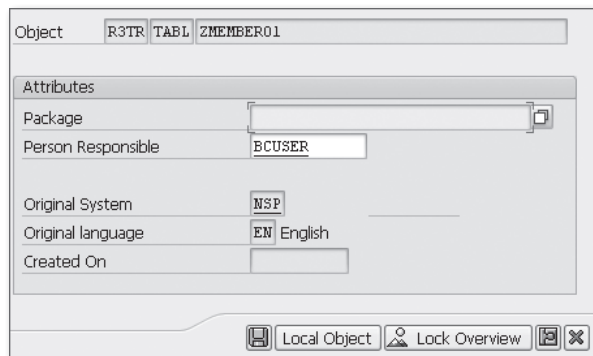


Figure 2.4 Creating an Object Directory Entry

Then enter the table's fields in the FIELDS tab (see Figure 2.5) and determine whether a field is a key field, whether it should have an initial value in the database, and which data element describes the business attributes of the field:

Maintaining
table fields

› Field

Though you don't have to take naming conventions into account when selecting a field name, you should make a meaningful selection. Because a table field is always addressed via the table (in the form of `Table_name-Field_name`), the difference between customer fields and SAP fields is not a problem here. The field name has a maximum of 30 characters.

› Key

Here you define whether a field is a key field. Key fields must be located in context at the start of the field and cannot have any gaps. Key fields uniquely identify a row and are used as sort and search criteria. A maximum of 16 key fields are allowed.

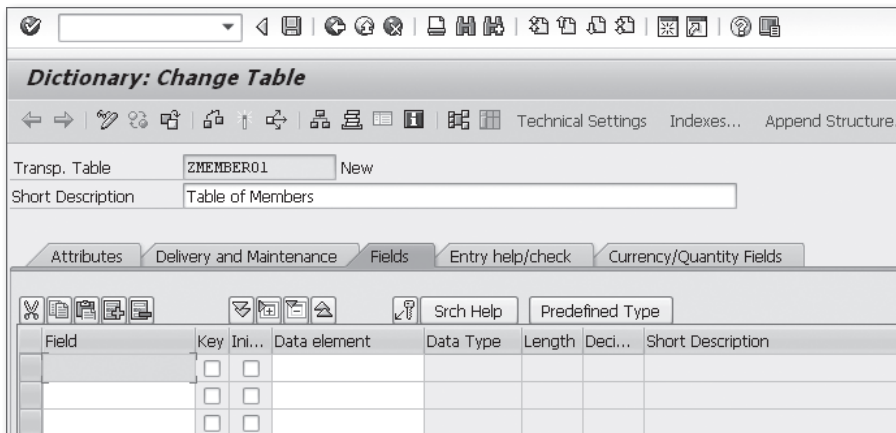


Figure 2.5 Maintaining a Table

› Initial values

This entry is interesting if the field is empty (that is, without a valid value) in a data record. You must make an entry here if you want the field to be populated according to its type and with an initial value (such as zero) in the database. Without an entry here, the field

is actually empty. You can populate the field contents with a value entry at any later time. You should usually always preassign initial values, because otherwise the field in the database may contain null values. In programming, null values can be easily confused with initial values, which can lead to unpleasant errors.

› Data element

Here you either enter the name of the data element or select a default data element. The data element stores some of the attributes of a field, including length, type, and so on.



Business and Technical Attributes

One characteristic of the ABAP Dictionary is that field attributes are not normally stored directly with the field. Instead, they are stored separately in their own dictionary objects. Notice the distinction between business and technical attributes. The business attributes of a field are found in the related data element, whereas the technical attributes of a field are found in the related domain. Enter the name of the data elements here, and the name of the domains with the data element. We'll deal with this procedure in more detail shortly.

The separation between business and technical attributes provides an advantage: data elements and domains can be reused multiple times. The fields of various tables can point to the same data element, and changes to field attributes can be maintained at a central location. Dependent objects are automatically generated at runtime.

Data Elements and Domains

Using existing
elements and
domains

In this section, we review the sequence of *field*, *data element*, and *domain*, and the creation of all the objects. Let's begin with a field as an example. The field stores the name of the system client—the client number that you entered on the logon screen. Objects like tables are stored independently of the client and centrally in the database. They can be addressed from multiple clients. Using the correct client is quite helpful when trying to identify the correct data record.

Creating a Data Element

The first field should be called `CLIENT` and should be a key field. Use the existing element, `MANDT`, as the data element. Make these entries and click **SAVE** or press **[Ctrl] + [S]**. Then note the gray column of the first row (see Figure 2.6).

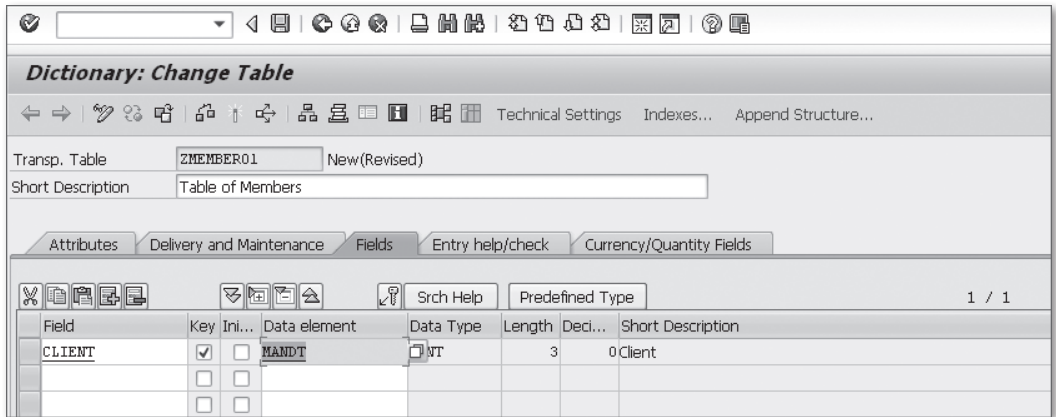


Figure 2.6 Creating a Table Field with an Existing Data Element

Because the `MANDT` data element already exists and is active, all its attributes and the related domains are accepted automatically. For our exercise, that includes the data type, length, number of decimal places, and a short description. You don't have to do any more work for this table field.

It gets somewhat more complex when you try to create a new data element and a new domain. But remember that you don't have to re-invent the wheel. A sensible alternative is to use existing objects (data elements and domains).

Creating your own elements and domains

The second field of the table should be a unique member number between 1 and 99,999. Enter `MNUMBER` as the field name. Though you must consider the customer namespace when you work with data elements, you can choose any name you wish—`ZMNUMBER`, for example—within those limits. Make both entries, and then define the field as a key field and save your work.

Forward navigation Because the data element does not yet exist, the gray areas behind the data element are not populated at this time. To avoid navigating through the ABAP Workbench when you create a data element, use *forward navigation*. When you double-click on the object that interests you (in this case, the ZMNUMBER data element), the system asks if you want to create a data element (see Figure 2.7). Clicking YES will prompt a dialog for maintaining the data element to be displayed (see Figure 2.8).

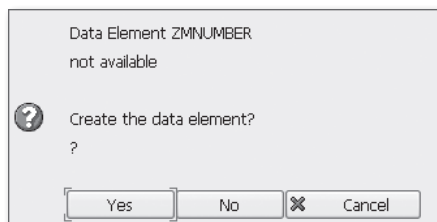


Figure 2.7 Creating the Object

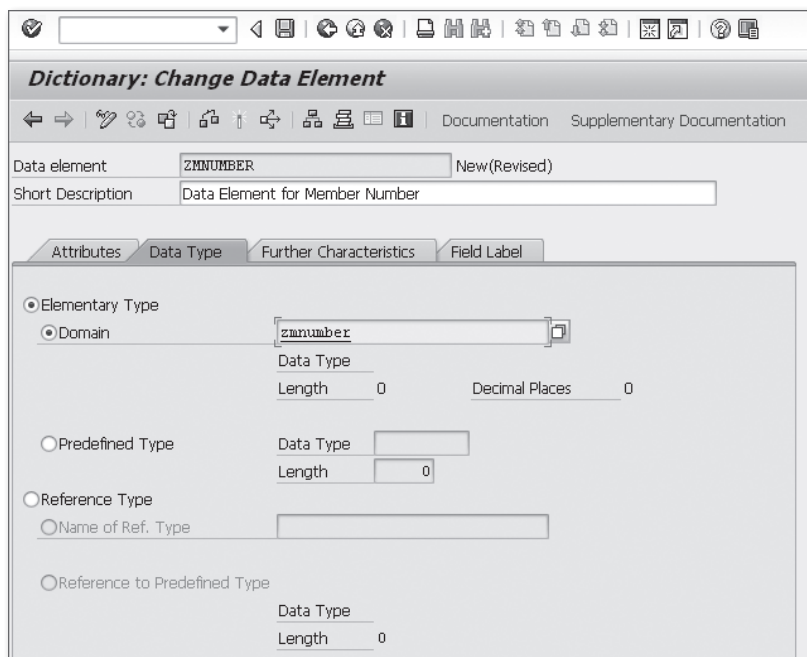


Figure 2.8 Maintaining the Data Element

Data Type

Enter the name of the linked domain in the DATA TYPE tab. Remember that data elements and domains are different types of objects. You can either select a new name for the new domain or repeat the same name that you used for the data element.

However, you must consider the customer namespace. The DOMAIN object types make it easy for the system to identify the correct object. For the sake of simplicity and uniformity in this example, the domain will also be named ZMNUMBER. Enter the domain in the appropriate field and save the data element.

Enter another SHORT DESCRIPTION of the object. (Use "Data Element for Member Number".) Because the data element is its own object, the system wants to know whether to transport the object. You must enter a package name again, but since you won't transport this initial exercise, save the data element as a LOCAL OBJECT (see Figure 2.9).

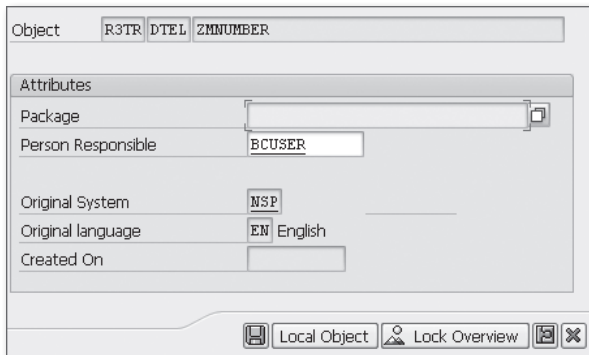


Figure 2.9 Saving the Data Element

In the FIELD LABEL tab, specify the identifier that will later be output in dialog programs and lists. In the screen templates of later dialog programs, you can set up the table field with its short, medium, or long text. The column for the field has a title in lists. Maintain these texts here and populate the related fields of the FIELD LABEL, as shown in Figure 2.10. After you save your entries, the system calculates the

Field label

length of the texts and populates the LENGTH field. You have now done the minimum required to create a data element.

Dictionary: Change Data Element

Data element: ZMNUMBER New(Revised)

Short Description: Data Element for Member Number

Attributes | Data Type | Further Characteristics | **Field Label**

	Length	Field Label
Short	10	MNO
Medium	15	MNumber
Long	20	Member Number
Heading	3	[MNo]

Figure 2.10 Maintaining the Field Label



Documentation

Developers and users often become frustrated with the poor documentation of programs provided by other developers. We want to create an explanatory text for our data element so that if, later on, someone doesn't know the reason behind the field (even though the name is intuitive), the user can access an explanation and help for the entry or use in custom programs.

All documentation and texts are maintained in the logon language, which is English in this case. When necessary, the texts and documentation are translated into the target language with a separate tool.

Click DOCUMENTATION (see Figure 2.10) to display the screen of another SAP tool, the *SAPscript Editor*. In the world of SAP software, SAPscript is used to design forms; it has its own editor and set of commands. Further detail about the SAPscript Editor is beyond the scope of this book, so just click DEFINITION and enter "Please insert the number of the member (valid from 1 to 99999)" as the help text in the row

beneath the entry (see Figure 2.11). Save your entry and press **F3** to return to maintenance of the data element.

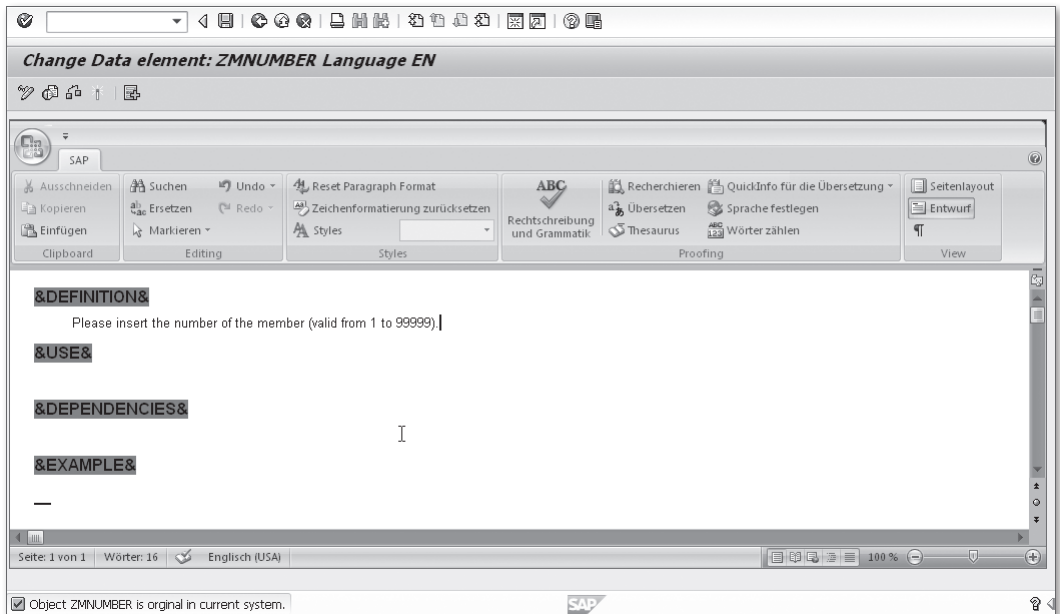


Figure 2.11 Maintaining the Help Text

Creating Domains

You've already maintained the domain name in the DATA TYPE tab of the maintenance screen. So now that we've completed our initial work on the data element, we can focus on creating the domain. We encourage you to work with forward navigation, so just double-click on the domain name, ZMNUMBER.

The system asks if you want to create a domain. If you click Yes, the maintenance screen for domains is displayed (see Figure 2.12). But before we look at the tabs of the domain, we must enter a short description: "Domain for Member Number".

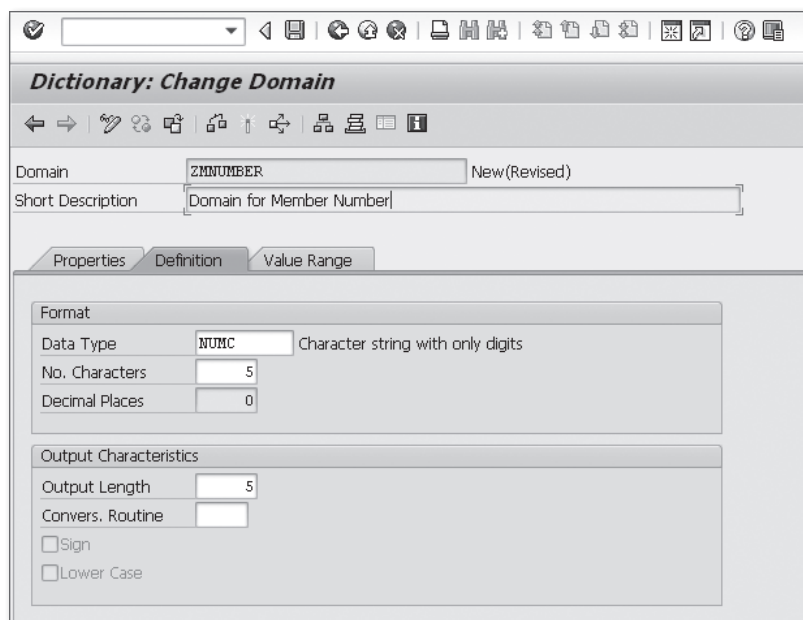


Figure 2.12 Maintaining Domains

Definition In the **DEFINITION** tab, you must first specify the data type of the domain. In the real world, decisions about all the attributes of the field were made and documented when the table was designed. But for your information, you can press **F4** to see the data types contained in the dictionary and their attributes. For this exercise, select **NUMC** as the data type—a numeric character string with an output length of 5. Save the domain object as a **LOCAL OBJECT**.

Value range According to the previous declaration of the domain, it might be helpful to limit the range of valid values. The maintenance screen for the domain contains the **VALUE RANGE** (see Figure 2.13) for this purpose.

In principle, three different types of limitation are supported. If any entries are made here, the system accepts only the value defined here as valid and rejects all other entries. But please note that how the system responds will depend on the data type. Fixed values (single values and intervals) cannot be entered for all data types. Checking entries for dialog screens also depends on the data type:

Dictionary: Change Domain

Domain: New(Revised)

Short Description:

Properties | Definition | Value Range

Single Vals

I	Fix.Val.	Short Descript.

Intervals

Lower limit	UpperLimit	Short Descript.

Value Table:

Figure 2.13 Defining the Value Range of a Domain

› Single values

When you maintain single values, you list and describe every valid value. Maintenance of single values in the domain makes sense when the number of permitted entries is relatively small.

› Intervals

If you maintain valid intervals in the domain, the smallest value and the largest value of an interval are specified. All values in between are also valid, including the values at both ends of the interval. Several intervals can be specified for one domain.

→ Chapter 7 deals with foreign keys in more detail.

› Value table

We recommend that you work with a value table when large quantities of data are involved if you expect input checks for the field to work with a table. Please note that simply populating the value table does not set up a check; this requires implementation of a foreign key on the table field.

Checking the domain

In this example, we don't cover entering fixed values or value tables, but instead focus on saving domains. To proceed safely and avoid formal errors, you should check the domain in the next step. Follow the menu path DOMAIN • CHECK • CHECK (see Figure 2.14), press **Ctrl** + **F2**, or the CHECK button. If you're lucky, the system notifies you that it has not found any inconsistencies. You can then activate the domain.

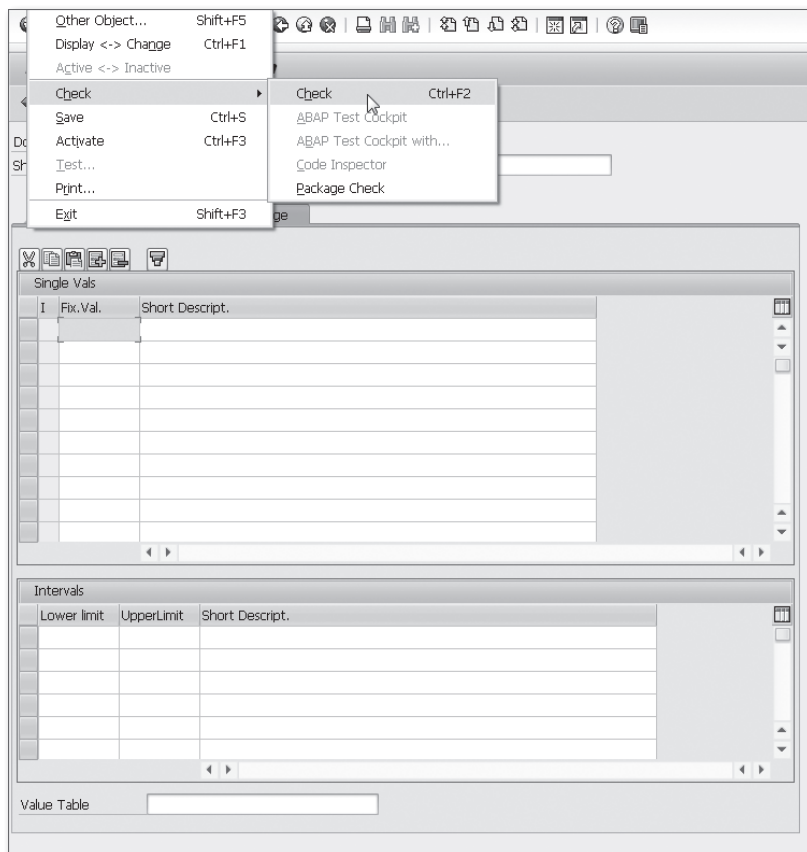


Figure 2.14 Checking the Domain

The domain still has a status of *new* (see Figure 2.15). It must have a status of *active* for you to use it in an active data element; it will become active when you activate it. Navigate via the menu (DOMAIN • ACTIVATE), **Ctrl** + **F3**, or the ACTIVATE button.

Activating the domain

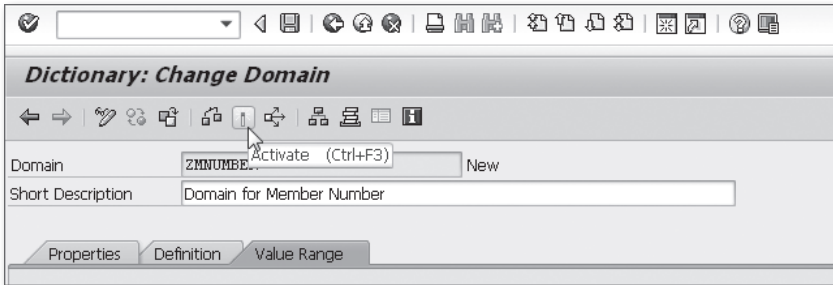


Figure 2.15 Activating the Domain

The following dialog window displays an overview of all the currently inactive objects; the domain is already highlighted (see Figure 2.16). If necessary, you can also highlight and activate the other inactive objects. But if the work on these objects has not yet been finalized, you should avoid unnecessary risks and activate only the desired domain.

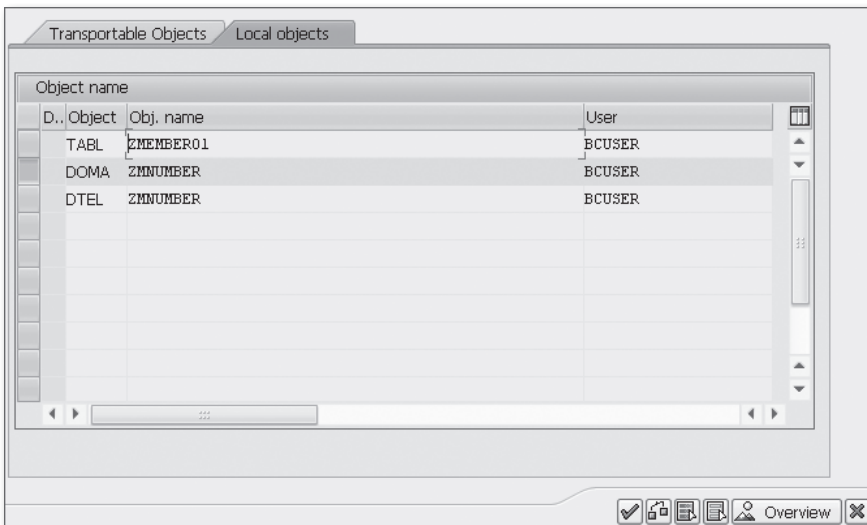


Figure 2.16 Activating Objects

Systemwide use Confirm your selection with the ENTER button at the lower left or with the appropriate key. The status line notifies you that the object has been activated. To double-check, view the status again and ensure that it is set to active (see Figure 2.17). The domain can now be used globally in the system. Press **F3** to leave the maintenance screen and return to the data element.

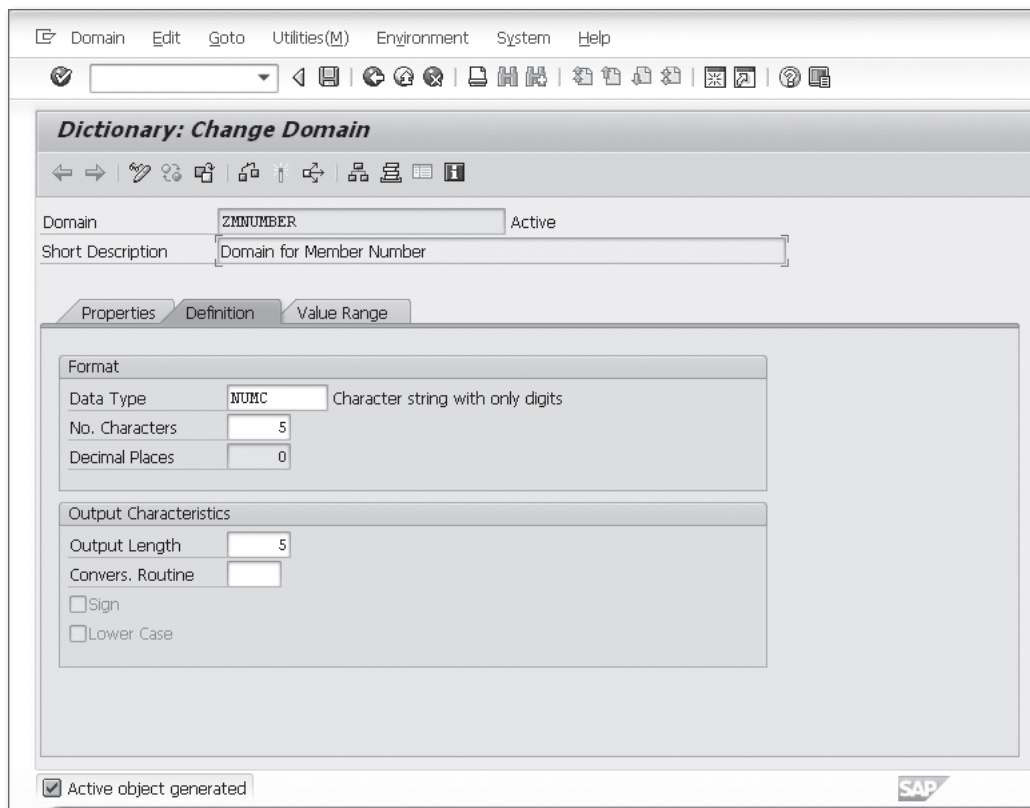


Figure 2.17 Activated Domain

Checking and Activating a Data Element

You should also perform a consistency check before activating a data element. Again, you can navigate via the menu, the key combination, or the CHECK button (see Figure 2.18). Ideally, everything is correct and you can proceed with activating the data element.

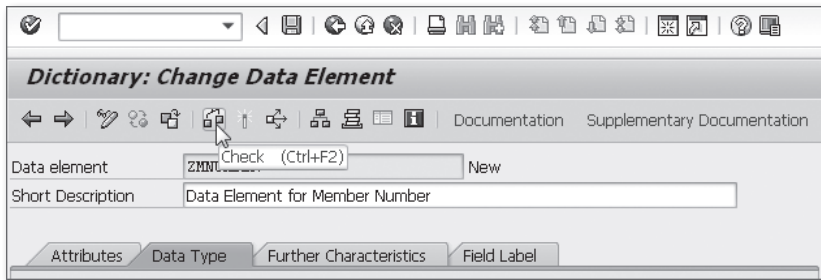


Figure 2.18 Checking the Data Element

The work here corresponds to the work for domains. Activate the data element with the menu or button, confirm your action in the following dialog, and then check the success of the activation by looking for either a status of active for the data element or a notification in the status line. Then use **[F3]** to return to editing the table fields.

Activating the data element

In order to store data records meaningfully in the table, our exercise requires that you enhance the table with additional fields. You can decide on your own which fields to include in the sample table. You're just practicing here. For our needs, four additional fields are enough to maintain a small data record in the table. The result should look like Figure 2.19.

Completing the table

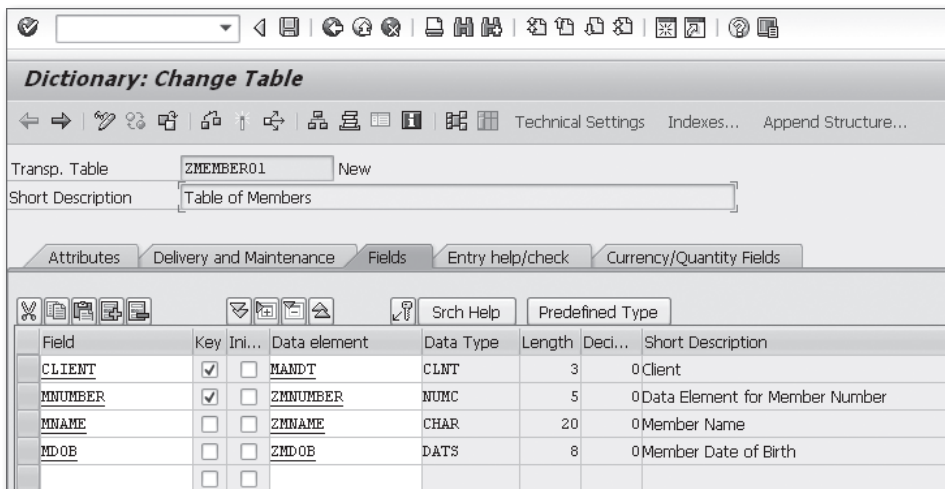


Figure 2.19 Sample Table with Four Fields



Checking the Table

Don't forget to check the table for inconsistencies with the **CHECK** button, **Ctrl** + **F2**, or the menu. You should make this important step part of your normal routine and your basic approach.

Maintaining the Technical Settings of the Table

The table is still located only in the dictionary, which is why you have to maintain the technical settings before you activate the table. That's where you define memory parameters, or how (if at all) the data record is stored in a read buffer when the database reads it.

Navigate to the technical settings from the table maintenance screen of the dictionary via **GOTO • TECHNICAL SETTINGS** or use the **TECHNICAL SETTINGS** button. The required screen is displayed (see Figure 2.20).

Dictionary: Maintain Technical Settings

Revised <-> Active

Name: ZMEMBER01 Transparent Table

Short text: Table of Members

Last Change: BCUSER 26.07.2011

Status: New Not saved

Logical storage parameters

Data class: APPL0

Size category: 0

Buffering

☒ Buffering not allowed

☐ Buffering allowed but switched off

☐ Buffering switched on

Buffering type

☐ Single records buff.

☐ Generic Area Buffered

☐ Fully Buffered

No. of key fields: 1

☐ Log data changes

☐ Write access only with JAVA

Figure 2.20 Maintaining the Technical Settings of the Table

For the DATA CLASS, use `APPL0` (master data in transparent tables). By using this value, you're defining the correct physical area for the table in the database. The physical area, also called the *tablespace*, is an assignment to the directories or disks of the database. Depending on the database system, the data volume, and the number of accesses, the settings made here have an effect on later runtime behavior and should optimize read and write access. Master data or customizing data for the system settings is stored in a separate, dedicated database area.

Data class

Enter "0" as the SIZE CATEGORY because we don't expect a lot of data records for our exercise. The size category sets the size of the initial storage area for the table in the database. If the area you choose is too large, you will reserve unnecessary space. If the area you select is too small, you will have to reorganize more often.

Size category



Smallest Category

Don't be perplexed by the numbers in the help window. These suggested values depend on the database system used. The smallest of the four size categories is sufficient for our needs.

For buffering, select `BUFFERING NOT ALLOWED`. In general, buffering means that the table is completely or partially loaded into working memory for reading. In other words, it's read in advance. For good performance, we recommend buffering tables or data records for large tables with many read accesses and relatively few write accesses. Our exercise involves only a very small table.

Buffering

After you make these settings, use `F3` to return to table maintenance and activate the table. Success is indicated by a notification in the status line ("Object activated"); the status of the table is now displayed as `ACTIVE` (see Figure 2.21).

Activating the table

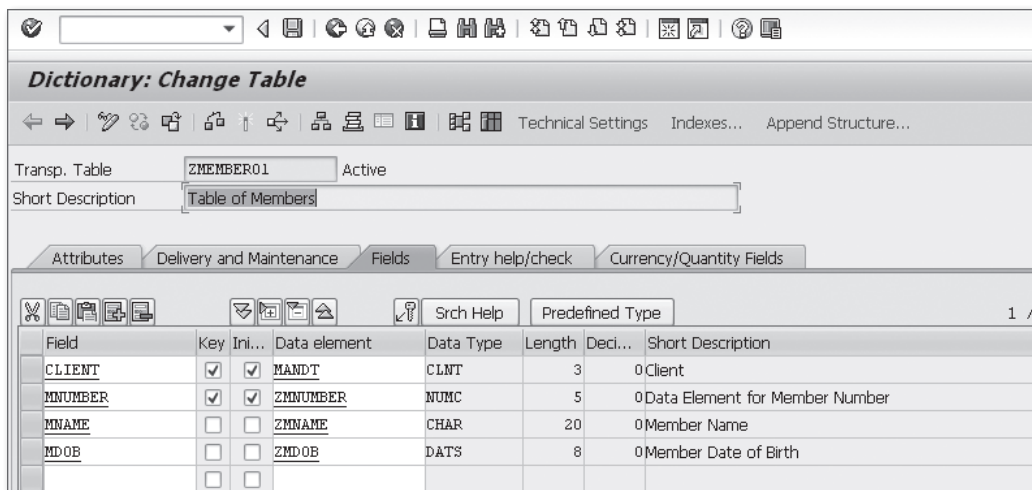


Figure 2.21 Activated Table



Default Values Sufficient

You can classify every table and structure in the ABAP Dictionary using an *enhancement category*, which defines how the table can be enhanced in the future. Don't get frustrated by the warning that is output after the successful activation. It only indicates that the default values from the system have been used for this purpose, which is absolutely sufficient for our case.

Creating Data Records

You can create data records for the table from the maintenance dialog. To write a few data records to the table, navigate via UTILITIES • TABLE CONTENTS • CREATE ENTRIES (see Figure 2.22) to a maintenance dialog in which you can create and save data records (see Figure 2.23).

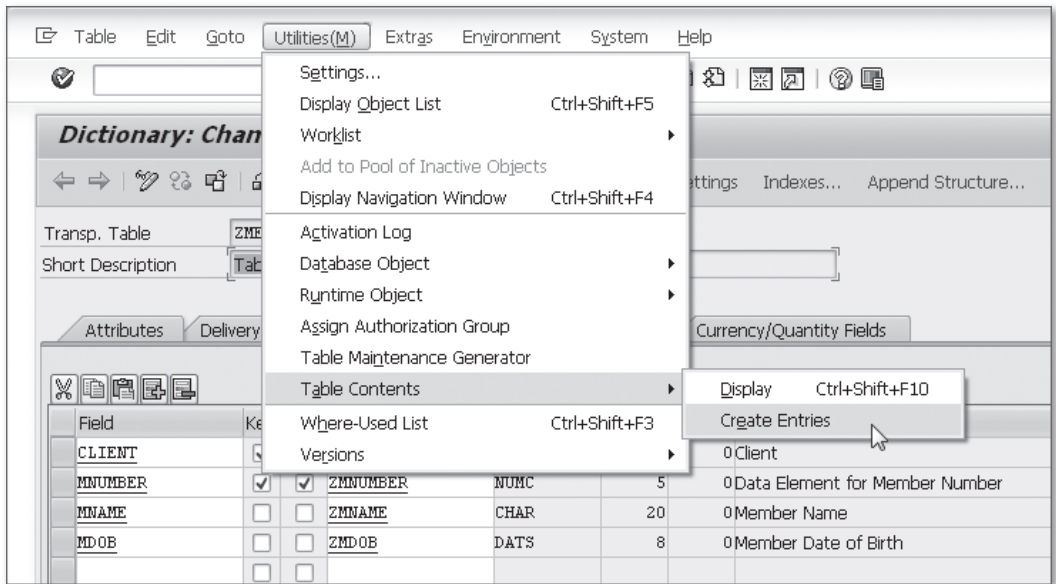


Figure 2.22 Creating Entries in the Table

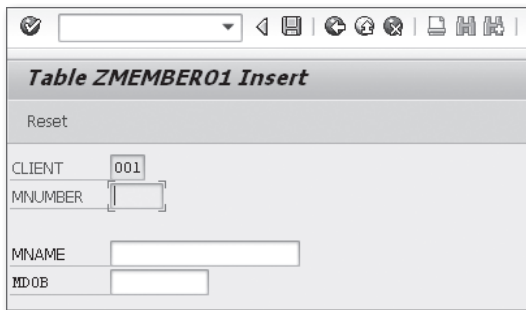


Figure 2.23 Maintenance Dialog for Creating Data Records

Entering Data Records

As a test, enter three data records: a member number (MNUMBER), a name (MNAME), and a birth date (MDOB). The CLIENT field is grayed out and cannot accept entries because automatic client management (which we'll discuss later) has automatically populated this field with the logon client. Key fields (in this example, the member number) appear in yellow. The remaining fields are white.

Gray, yellow, and white

Save each data record individually using the corresponding button. After each save, the status line displays a message: "Database record successfully created."

Monitoring To check that the documentation is built into the data element, focus on the entry field for the birth date. Click it and press **F1**. If you have done everything correctly, the business help appears in a new window. If you have anchored fixed values in the domain, you can check them with **F4**. Then return to the maintenance dialog for the table with **BACK** or **F3**.

Displaying the Contents of the Table

From the maintenance dialog, you can view and check the data records of the table. Navigate via **UTILITIES • TABLE CONTENTS** and select **DISPLAY**. You can limit selection to specific records, but in this case you want to view all the data records that you have maintained, so click on **EXECUTE** (see Figure 2.24).

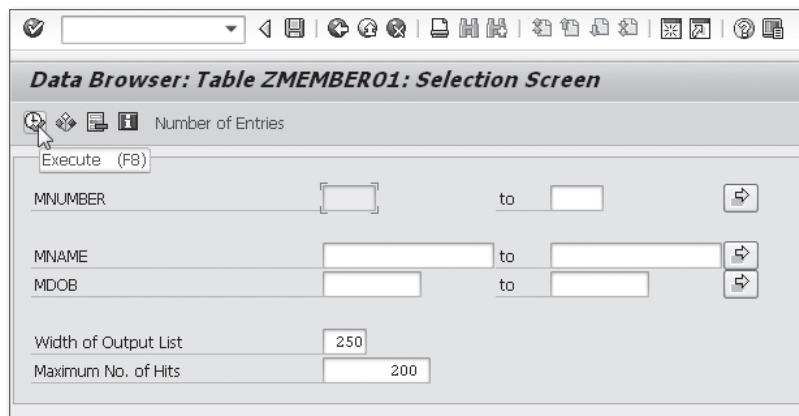
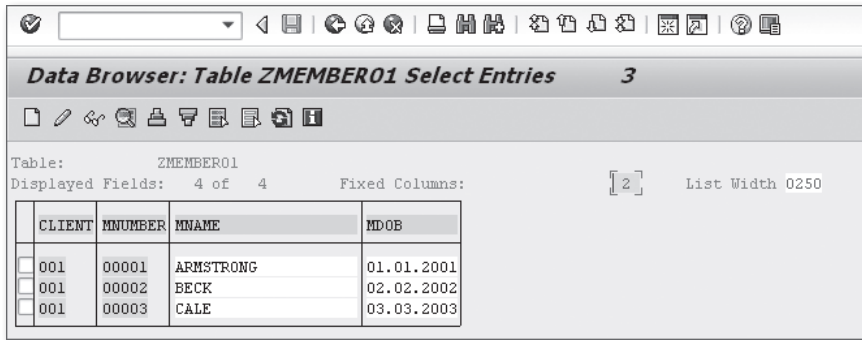


Figure 2.24 Displaying the Table Contents

Initial success When you see the list shown in Figure 2.25, you'll know that everything has worked properly. Of course, tables in the real world are more comprehensive and involve more effort. We have not explained many options available in the dictionary, but as far as getting starting with

ABAP goes, the worst is already behind you. You have created a table and populated it with data. Now you can devote all your attention to ABAP programming.



The screenshot shows the SAP Data Browser interface for table ZMEMBER01. The title bar reads "Data Browser: Table ZMEMBER01 Select Entries 3". Below the title bar is a toolbar with various icons. The main area displays the table structure and data. The table has four columns: CLIENT, MNUMBER, MNAME, and MDOB. There are three data rows, each with a checkbox in the first column. The status bar at the bottom indicates "Table: ZMEMBER01", "Displayed Fields: 4 of 4", "Fixed Columns:", and "List Width 0250".

	CLIENT	MNUMBER	MNAME	MDOB
<input type="checkbox"/>	001	00001	ARMSTRONG	01.01.2001
<input type="checkbox"/>	001	00002	BECK	02.02.2002
<input type="checkbox"/>	001	00003	CALE	03.03.2003

Figure 2.25 Displaying Data Records

Index

\$TMP, 50, 177

/NEX, 44

A

A, 131

ABAP, 21

Command, 83

Developer community, 504

Keyword documentation, 83

Origin, 503

Processor, 30

Programming, 34

Statement, 83

Trial version, 161, 517

ABAP Debugger, 141, 147, 158, 421, 432

Call, 142

Classic, 142

Close, 158

Configure layer, 164

Execution level, 148

New, 142, 151

System field, 152

Tablespace, 153

ABAP Dictionary, 40, 45, 46, 48, 175, 200, 215, 227, 244, 249

ABAP Editor, 41, 71, 75, 90

Control, 77

Dynamic breakpoint, 158

Icon, 77

Toolbar, 77

ABAP memory, 480, 481

ABAP Objects, 32, 387, 503

Compatibility, 387

ABAP program

Modular structure, 33

Structure, 31

ABAP report

Create, 71

Execute, 84

ABAP runtime environment, 31

ABAP Workbench, 37, 85, 443

Abstraction, 513

Backend abstraction, 513

Platform abstraction, 513

Access effort, 409

Access key, 38, 385

Access time, 385, 386

Accounting, 36

Acrobat Reader, 464

Activation sequence, 180

ADD, 111

Addition, 110

ADJACENT DUPLICATES, 404

Alphanumeric character, 122

ALV Grid, 162

AND, 299

Logical, 299

AND RETURN, 455

APPEND, 394, 397, 420, 423

Append structure, 173, 194

Activate, 196

Assign, 194

Component, 195, 196

Create, 194

Maintain, 193

Name, 195

APPL0, 65

Application layer, 23, 24

Application log, 359

Application server, 23

Archive index, 507

Archiving, 506

Arithmetic operation, 223

Basic, 104, 110

Arithmetic operator, 111

-, 111

., 83

**, 82, 111*

/, 88, 111, 114

+, 110

Array fetch, 401, 429
 Article number, 123
 AS CHECKBOX, 333
 Assigning names
 To variable, 104
 Asterisk, 469
 AS TEXT, 412
 AT SELECTION-SCREEN, 326, 327, 358
 AT SELECTION-SCREEN ON, 330
 Attribute
 Semantic, 191
 AUTHORITY-CHECK, 246
 Authorization
 Group, 73
 Program execution, 245
 Transaction, 245
 Authorization check in source code, 246
 Authorization concept, 244
 Authorization object, 244
 Automatic client management, 67
 Automatic type conversion, 296, 399

B

Backend mode, 75, 76
 Background processing, 350
 Back quotes, 126
 BAdI, 515
 Batch job, 73
 BEGIN OF, 390
 BEGIN OF LINE, 360
 BETWEEN, 297
 Blank character, 110
 Remove, 129
 Block
 Buffer, 103
 Duplicate, 81
 Move, 81
 Name, 363
 Block-by-block transfer, 401
 Block of lines
 Delete, 103
 Blog, 517
 Body line, 481

Change, 406
 Delete, 404, 412, 414
 BPM, 510
 Branch, 280
 Breakpoint, 79, 142, 144, 154
 Activated, 157
 Delete, 157, 159
 Dynamic, 145, 159, 160
 External, 79
 Manage, 158
 Mode, 157
 Save, 158
 Session, 145
 Set, 79, 144, 157
 BREAK-POINT, 144, 160
 Buffering, 65, 81, 244
 Not allowed, 65
 Business Add-In, 515
 BY, 411

C

Calculation, 99
 Accuracy, 113
 Age of a person, 221
 Beginning of a month, 217, 218
 Dimension of time periods, 222
 Invoice amount, 229
 Invoice date, 217
 Key date, 216, 217
 Passed days, 220
 Payment date, 217
 Remaining days, 221
 Time, 216
 Time period containing date change, 225
 Time period in days, 222
 Time period in hours, 223
 Time period in minutes, 223
 Time period in seconds, 223
 Time period in years, 222
 Time until midnight, 224
 Ultimo, 219
 Using date field, 216
 Using time field, 223
 With currency field, 227

- With quantity field*, 227
- Callback, 163
- Callback function, 163
- CALL FUNCTION, 466, 470
- CALL METHOD, 478
- Cardinality, 191
- CASE, 284, 285, 287, 309
 - Structure*, 470
- Case distinction, 275, 285
- Case sensitivity, 313
- Central control component, 502
- Chain statement, 89, 95, 119, 388, 390, 391
- Change mode, 76
- Character
 - Alphanumeric*, 122
 - Cut*, 124
 - Numeric*, 123
- Character string, 93, 121
 - Complement*, 124
 - Concatenate*, 129
 - Condense*, 128
 - Modify*, 121
 - Replace*, 124, 126
 - Search*, 124
 - Shift*, 124, 125
 - Split*, 131
- CHECK, 292, 449
- Checkbox, 455
- Check inconsistency, 192
- Checksum process, 386
- Check table, 189, 207
- Class, 471
 - Attribute*, 471
 - Global*, 472
 - Local*, 472
 - Method*, 471, 473
 - Object instance*, 471
 - Scope*, 472
 - Test*, 475
 - Visibility*, 471
- Class Builder, 473
 - Adapt presentation*, 474
 - Call*, 473
- CLASS DEFINITION, 478
- CLASS IMPLEMENTATION, 478
- CLEAR, 251, 253, 254, 412, 414, 421, 423, 432, 437, 452
- Client, 36
- Client handling
 - Automatic*, 249
- Client management, automatic, 67
- Client number, 249
- Client-server architecture, 23, 243
- Clipboard, 76
- Cluster, 480, 481
 - Table*, 47
- Code table, 296
- Collective shaft, 283
- Column distribution, 119
- Command, 83
 - Cancel*, 80
- Comment
 - Asterisk*, 82
 - In the screen*, 360
 - Quotation mark*, 82
- COMMENT, 361
- Comment line, 103, 278
 - Convert blocks of lines*, 102
- Company code, 36
- Compatible data object, 111
- Component
 - Cross-application*, 37
- Composite profile, 245
- Computational accuracy, 114, 227
- COMPUTE, 113
- CONCATENATE, 129, 139
 - SEPARATED BY*, 130
- CONDENSE, 128, 139
 - NO-GAPS*, 129
- Condition, 290
 - Complex*, 280, 298
 - Resolve*, 298
- Consolidation system, 25, 26
- Constant, 104
- CONSTANTS, 109
- Context-sensitive, 358
- CONTINUE, 291
- Control component
 - Central*, 502
- Control structure, 276, 281
- Conversion rule, 111, 112

- Convertible data object, 111
 - Copy, 203
 - Core area, 36
 - Correction system, 27
 - Create line, 89
 - Cross-application component, 37
 - Cross-client table, 249
 - Cryptology, 386
 - Currency
 - Amount*, 184
 - Field*, 183
 - Unit*, 184, 187
 - Currency field, 215
 - Define*, 215
 - Currency key, 215
 - Currency name, 215
 - Customer exit, 515
 - Customer namespace, 195, 196, 199
 - Customizing, 26, 514
 - Cut, 203
- D**
-
- Data
 - Browser*, 49, 179, 202
 - Consistency*, 456
 - Declaration*, 443
 - Definition*, 445
 - Interface*, 462
 - Loss*, 204
 - Object*, 448
 - Temporary*, 336
 - Transfer*, 480
 - DATA, 105, 108, 122, 213, 234, 250, 251, 390, 419, 432, 448
 - BEGIN OF*, 390
 - DECIMALS*, 106
 - END OF*, 391
 - VALUE*, 214
 - Database
 - Adjust*, 174, 206
 - Interface*, 30
 - Layer*, 23
 - Lock*, 246
 - Server*, 25
 - Utility*, 206
 - Database table, 479
 - Modification*, 173
 - Output*, 86
 - Read*, 86
 - Transparent*, 173
 - Data block, 400
 - Data class, 65
 - Data Dictionary (see ABAP Dictionary), 45
 - Data element, 52
 - Activate*, 63
 - Check*, 62
 - Create*, 53
 - DATA LIKE, 107, 123
 - Data object, 104, 387, 393
 - Compatible*, 111
 - Convertible*, 111
 - DATA OCCURS, 391
 - Data record, 383
 - Check*, 178
 - Create*, 66
 - Delete*, 255
 - Enter*, 67
 - Data type, 55, 58, 105, 113, 387, 448
 - c*, 122, 125, 132, 216, 227, 251, 253, 296
 - CHAR*, 180, 183
 - Complex*, 387
 - CUKY*, 184, 185, 215, 227
 - CURR*, 183, 215, 216, 227
 - d*, 213, 253
 - DEC*, 183, 227
 - Difference*, 125
 - Elementary*, 106
 - f*, 329
 - Generic*, 122, 123
 - i*, 106, 112, 222
 - n*, 123, 125, 132
 - NUMC*, 58, 183
 - p*, 106, 109, 112, 216, 222, 227, 228
 - QUAN*, 183, 228
 - t*, 214
 - UNIT*, 184, 228
 - DATA TYPE, 106, 123
 - DATA VALUE, 123

- Date field, 213
 - Arithmetic operation, 216*
 - Define, 213*
 - Fill, 214, 216*
 - Processing, 216*
- Date format, 96
- Debugging, 141
 - Mode, 142, 146*
- Decimal number, 105
- Decimal place, 106, 108
- DECIMALS, 106
- Decimal separator, 108
- Declaration, 107
- Declaration part, 31
- DEFAULT, 330, 341
- DELETE, 249, 255, 256, 272, 275, 404, 409, 410, 421, 436
- DELETE ADJACENT DUPLICATES, 404
- Delivery class, 49
- DESCENDING, 412
- DESCRIBE, 407
 - LINES, 407*
- Development system, 25, 26
- Dialog
 - Module, 32*
- Dictionary object, 52
- Digital signature, 386
- Dispatcher, 29
- DIV, 115, 222
- DIVIDE, 111
- Division, 110, 114, 222
 - Different cases, 114*
 - Integer, 320*
 - Integral, 115*
- Division of labor, 244
- DO, 287, 288
 - Loops, 289*
- Documentation, 56
- Domain, 52, 107
 - Activate, 61*
 - Check, 60*
 - Create, 53, 57*
 - WAERS, 186, 188*
- Dump, 85
- Dynamic name assignment, 251, 256

- Dynamic program, 324
- Dynpro, 28, 324
 - Processor, 30*

E

- Editor lock, 74
- EHP, 516
- Element
 - Copy, 101*
 - Create your own, 53*
- Elementary data type, 106
- ELSE, 282, 283
- ELSEIF, 282, 283
- ENDCASE, 285
- ENDDO, 287
- ENDFORM, 447, 450, 452
- ENDIF, 276, 281
- Endless loop, 288
- ENDLOOP, 402
- END OF, 391
- END OF BLOCK, 363
- END OF LINE, 360
- ENDSELECT, 88
- ENDWHILE, 290
- Enhance, 77
- Enhancement, 514
 - Category, 66*
 - Package, 516*
- Enhancement Framework, 515
- Enterprise service, 514
- Entry check, 183, 186, 188, 191
- Error
 - Analysis, 85*
 - Case, 286, 309*
 - Exit, 470*
 - Handling, 261, 328, 358*
 - Log, 205*
 - Message, 191*
- Event, 32, 325, 375
 - Block, 32*
 - Example, 327*
 - Keyword, 325, 375*
 - Orientation, 375*
 - Terminate, 295*
- Exception handling, 470

Exchange rate conversion, 227
 Exclusive lock, 247
 Executable program, 73, 324
 EXIT, 293, 294, 315
 Exponential number, 113
 EXPORT, 480
 Export parameter, 466, 470
 EXPORT TO MEMORY ID, 480
 External report
 Call, 453

F

F1, 68, 83, 331, 361
 F3, 86, 355
 F4, 58, 68, 182, 192, 331, 332, 348
 F5, 148
 F6, 148
 F7, 149
 F8, 85, 149, 376
 Facebook, 518
 Field, 51, 99, 104, 141, 197
 Asterisk, 88
 Declare, 104
 Define, 213
 Identical names, 391, 394, 396, 398, 399
 Insert, 173
 Label, 55, 344
 Length, 105, 122
 List, 88, 184, 389, 449, 480
 Mode, 147, 148
 Name, 51
 Process, 104
 Tab, 197
 Type, 388
 Field content
 Change, 151
 Check, 149
 Modify, 150
 Monitor, 155
 Shift, 126
 Find, 81
 Find next, 81
 Fixed point arithmetic, 74
 Fixed value, 110, 181
 Individual value, 182
 Interval, 182
 Maintain, 181
 Test, 182
 Floorplan Manager, 511
 Flow control, 501
 Foreign key, 60, 186, 191, 246
 Maintain, 188, 189
 Relationship, 207
 Test, 192
 Foreign key field
 Type, 191
 Foreign key table, 189
 FOR FIELD, 361
 FORM, 447, 448, 451
 Forum, 517
 Forward navigation, 54, 87, 197, 200, 202, 443, 446, 480
 FPM, 511
 Frame, 360
 Framework, 504
 Debug, 160
 FREE, 413, 414, 432
 Free text, 360
 FROM, 405
 Frontend editor (new), 75
 Frontend editor (old), 76
 Frontend mode, 75
 Function Builder, 41, 456, 459
 Test environment, 463
 Function code field, 40
 Function group, 324, 456, 458
 Function library, 456
 Function module, 32, 41, 78, 247, 441, 456
 Attribute, 459
 Call, 466, 468
 Documentation, 467
 Exception, 462
 Export, 461
 Import, 460
 Name, 466
 Pass value, 460
 Processing type, 459
 Search, 457
 Test, 457, 463, 464

G

Generate blank line, 89
 Generic data type, 122, 123
 Generic key, 191
 GET PARAMETER, 480
 GET PARAMETER ID, 479
 Global SAP memory, 479
 Global variable, 447
 Graphical Screen Painter, 323
 Graphical user interface, 23
 GUI, 23

H

Hash algorithm, 386, 409
 Hashed table, 386, 409
 Hat, 423, 432, 451
 Header line, 396, 398, 405, 451
 Initialize, 413
 Help, 78
 Helper class, 478
 Hierarchy level, 123
 HIGH, 336
 Hollywood principle, 506
 HTTP, 512
 Human resources, 36
 Hyphen, 87

I

ICF, 25
 IF, 276, 281, 283, 292, 298
 IF structure, 285, 335, 470
 Nest, 283
 IMPORT, 480
 IMPORT FROM MEMORY ID, 480
 Import parameter, 466
 INCLUDE, 395, 400, 442
 Include report, 442, 443
 Include structure, 197
 Activate, 200
 Create, 199
 Insert, 199
 Maintain, 198
 Position, 198
 Precondition, 198

INCLUDE STRUCTURE, 400
 Index, 385, 403, 409
 INDEX, 403, 407
 Index table, 386, 404, 407, 409
 Individual software, 514
 Industry minute, 238
 Information system, 37
 INITIALIZATION, 326, 327, 375
 INITIAL SIZE, 389
 Initial value, 51, 253
 Inline documentation, 82
 IN PROGRAM, 453
 Input block, 360
 Input field
 Position, 360
 Input help, 180, 186, 192
 List, 182, 192
 Input verification, 327, 329, 377
 Insert, 203
 INSERT, 249, 250, 254, 403, 407, 420
 Instance, 25, 28
 Integer, 105
 Integral division, 115
 With remainder, 115
 Interface, 388
 Internal table, 382, 450, 451, 462, 481
 Benefit, 382
 Define, 389, 390
 Fill one record at a time, 396
 Process, 402
 Structure, 382
 Internet Communication Framework,
 25
 Interval, 59, 297
 INTO, 403, 404
 INTO CORRESPONDING FIELDS, 401
 Inversion of control, 506
 IS INITIAL, 297, 334
 Item number, 123

K

Key, 51, 389, 403, 404, 408
 Generic, 191
 Non-unique, 385, 388
 Unique, 385
 User-defined, 385

Key directory, 514
 Key field, 51, 198, 203, 404, 408
 Change, 203
 Delete, 204, 207
 Manipulation, 174
 Keyword, 78
 Documentation, 83

L

Language key, 356
 Layer-aware debugging, 148, 160, 161
 Object set, 164
 Leap year, 301
 Legibility, 33
 Length, 132
 LIKE, 107, 123
 Limit, 384
 Line
 Break, 88, 94
 Change, 403
 Concatenate, 81
 Copy, 80
 Cut, 80
 Delete, 81, 409
 Duplicate, 81
 Duplication, 386
 Free layout, 360
 Insert, 80, 407
 Move, 81
 Redundant, 404
 Line block, 362
 LINES, 407
 LINE-SIZE, 94
 Line structure, 389
 Line type, 384, 388, 391, 392
 Define, 390
 List
 Display, 149
 Format, 88
 Screen, 96
 List heading, 342
 List output, 323
 List processor, 324, 326
 List screen, 324
 Literal, 83, 108, 124, 235, 257, 272, 313, 354, 406, 434, 466, 481
 String literal, 127
 Text field literal, 127
 LOAD-OF-PROGRAM, 33, 326
 Local class, 478
 Local name, 452
 Local object, 28, 50, 101, 177
 Local SAP memory, 480
 Local variable, 448, 449, 450
 Lock
 Business lock, 246
 Database lock, 246
 Logical, 247
 Point-in-time problem, 248
 Remove, 248
 Lock concept, 244, 246
 Physical database, 246
 SAP, 246
 Lock object
 Remove, 247
 Set lock, 247
 Lock problem, 248
 Lock table, 247
 Logical AND, 299
 Logical database, 73
 Logical expression, 275, 295
 Link, 284, 298
 Logical operator, 295
 Logical OR, 298
 Logistics, 36
 Log off, 42
 /NEX, 44
 Without security question, 44
 With security question, 42
 Logon, 34
 Client, 67
 Screen, 34
 Long text, 467
 Loop, 88, 92, 93, 240, 287, 383
 Endless, 288
 Inner, 289
 Limit runs, 288
 Outer, 289
 Processing, 93
 Termination statement, 291
 With condition, 287
 Without condition, 287
 LOOP, 402, 403, 407, 410

LOW, 336
 Lowercase, 329
 LOWER CASE, 332, 333, 341
 Lower-case and upper-case letters, 257
 LT, 292

M

Maintainability, 33
 Maintenance dialog, 179
 Maintenance screen, 48, 199
 Mandatory parameter, 466
 Master data, 49
 Memory area, 391
 Memory ID, 480
 Memory space
 Initial, 388
 Menu Painter, 41
 Message, 354
 Edit, 356
 Save, 357
 MESSAGE, 356, 358, 359, 377
 Side effect, 359
 Message class, 355, 359
 Create, 356
 MESSAGE-ID, 356
 Message in the status bar, 328
 Message number, 358
 Message pool, 356
 Message type, 358, 377
 Method, 440
 Mini-SAP system, 161, 517
 MOD, 115, 222
 Modification, 514
 MODIFY, 249, 254, 403, 406, 407
 FROM, 403
 Modularization, 22, 388, 440
 Module, 448
 Module pool, 324
 MOVE-CORRESPONDING, 174, 399
 Multibyte coding, 74
 Multiple selection, 338
 Multiplication, 110
 MULTIPLY, 111

N

Name
 Local, 452
 Name assignment
 Dynamic, 251, 256
 Static, 255, 256
 Namespace, 47
 Namespace prefix, 356
 Naming convention, 195
 Native SQL, 30, 47
 Navigation window, 78
 Display, 78
 Negation, 299
 Nesting, 93, 286, 289, 291, 388, 445
 Level, 283
 Network drive, 464
 Network path, 464
 NEW-PAGE PRINT ON, 324
 NO-EXTENSION, 341
 NO-GAPS, 129
 Non-key field, 203
 Add, 180
 NOT, 299
 Number
 Integer, 105
 Packed, 113
 Numeric character, 123
 Numeric text field, 123

O

Object
 Activate, 180
 Class, 245
 Local, 28, 50, 101, 177
 Object key, 193
 Object list, 78
 Display, 78
 Object Navigator, 42
 Object Services, 257
 OBLIGATORY, 330, 341
 OCCURS, 391
 Office, 37
 OpenSQL, 30, 40, 47, 244, 513
 Statement, 249

OPTION, 336
 OR, 298
 Logical, 298
 Output position, 120

P

Package, 49
 Packed number, 113
 Parameter
 Mandatory, 466, 469
 Optional, 469
 Parameter ID, 479
 Create, 480
 Initialize, 494
 Read, 494
 Write, 501
 Parameter list
 Check, 453
 PARAMETERS, 329
 Parentheses, 301
 Parenthetical level, 298
 Pattern, 79, 277
 Insert, 79, 277, 279
 PERFORM, 445, 447, 450, 451, 453, 493
 IN PROGRAM, 453
 USING, 449
 Period of time for date value, 221
 Physical area, 65
 PI, 513
 Placeholder, 359
 &, 359
 Point-in-time problem, 248
 Pool table, 47
 Position, 119
 Operand, 449, 453
 Specify, 119
 POSITION, 362
 Positioning, 132
 Position operand, 129, 359
 Posting activity, 459
 Preassignment, 107
 Prefix namespace, 195, 196
 Preselection, 304
 Presentation layer, 23
 Presentation server, 23, 25
 Pretty Printer, 79, 81
 Primary key, 254, 255, 256
 Printing, 324
 Procedure, 32, 445, 456
 Processing
 Reading, 383
 Processing block, 32, 325, 440
 Processing type
 Background, 206
 Direct, 206
 Production system, 25, 27
 Profile, 245
 Program
 Activate, 78, 85
 Check, 84
 Check logic, 141
 Executable, 73, 324
 Execution, 85
 Save, 84
 Start, 85
 Status, 77
 Termination, 86
 Program flow control, 275
 Program status, 73, 77

Q

Quality assurance system, 26
 Quantity field, 183, 184
 Quantity unit, 184

R

Radio button, 334
 RADIOBUTTON GROUP, 335
 Range, 337, 339
 READ, 404, 408, 431, 434
 INTO, 403
 Readability, 105
 READ INDEX, 403
 Record
 Create, 254
 Modify, 254
 Redundancy, 204
 Reference field, 185, 208, 215
 REFRESH, 414, 421, 432

Release reserved memory area, 414
 Remote call, 511
 Remote function call, 511
 Reorganization work, 174
 Repetition, 276
 REPLACE, 126, 139
 Report, 28, 352
 Copy, 100
 Data transfer, 479
 Start with variant, 352, 454
 Variant, 352
 REPORT, 83, 101, 144, 356
 REPORT, LINE-SIZE, 94
 Repository info system, 457
 Response time, 386
 Return code, 152, 249, 253, 254, 255,
 256, 286, 309, 436, 470
 Return value, 152
 Reusability, 199, 388
 REUSE_ALV_GRID_DISPLAY, 162, 168
 RFC, 511
 Role, 245
 Row
 Create, 255
 Delete, 203, 256
 Highlight, 76
 Insert, 203
 Update, 255
 Runtime, 206
 Environment, 73, 326
 Object, 178

S

Sandbox, 517
 Sandbox system, 26, 517
 SAP Community Network, 517
 SAP ERP, 22
 SAP GUI, 23, 34
 SAP Inside Track, 518
 SAP List Viewer, 162
 SAP Logon, 34
 SAP memory, 494
 Read, 479
 Write, 479
 SAP NetWeaver Business Process
 Management, 510
 SAP NetWeaver Process Integration,
 513
 SAPPHIRE NOW, 518
 SAP release, 515
 SAPscript Editor, 56
 SAP system architecture, 23
 SAP TechEd, 518
 SBAL_DOCUMENTATION, 359
 SCN, 517
 Screen
 Check, 191
 Classic, 323
 Flow logic, 323
 Selection screen, 323
 Screen Painter, 42
 Search
 Generic, 457
 Search term, 126
 SELECT, 88, 240, 249, 286, 287, 397,
 401
 Selection, 328, 336
 Selection button, 334
 Selection mode, 80
 Selection screen, 74, 324, 332, 346,
 351, 454
 Avoid, 455
 Process, 358
 Save, 346
 SELECTION-SCREEN, 360
 BEGIN OF LINE, 360
 COMMENT, 361
 END OF BLOCK, 363
 END OF LINE, 360
 FOR FIELD, 361
 POSITION, 362
 SELECTION-SCREEN TITLE, 363
 SELECTION-SCREEN WITH FRAME,
 363
 Selection table, 336
 Selection text, 342
 Activate, 344
 Create, 342
 Edit, 344
 Use, 342
 Selection variant
 Create, 347
 SELECT-OPTIONS, 336, 337, 342

- Semantic attribute, 191
- SEPARATED BY, 130
- Separator string, 131
- Session, 85
- Session breakpoint, 79
 - Set*, 79
- SET PARAMETER, 480
- SET PARAMETER ID, 479
- Shared lock, 247
- Shared object, 481
 - Area*, 482
 - Area root class*, 482
- SHIFT, 125
- Shift field contents, 126
- SHMA, 482
- Short description, 55
- Short documentation, 91
- SIGN, 336
- Signature
 - Digital*, 386
- Single quotation mark
 - Reversed*, 126
- Single quotation marks, 92
- Single value, 59, 337
- Size category, 65
- Sizing, 26
- SKIP, 89, 306
- Slash, 88
- Social media, 518
- Software-as-a-service, 517
- SORT, 411, 436
- SORT BY, 411
- Sorted table, 386, 407, 409
- SORTED TABLE, 389
- Sorting
 - Binary*, 411
 - Language-specific*, 412
- Sorting criterion, 412
- Sort sequence, 411
- Source code
 - Edit*, 79, 90
 - Modularize*, 440
 - Switch*, 77
 - Write*, 90
- Source field, 112, 129
- Source table, 175
- SPACE, 137
- SPLIT, 131, 139
- SPLIT AT, 131
- Standard key, 385
- Standard selection screen, 325
- Standard table, 196, 202, 385, 388, 407
 - Enhance*, 194
- STANDARD TABLE OF, 389
- START-OF-SELECTION, 33, 326, 377, 446
- Start position, 132
- Start using variant, 74
- Statement, 83
- Statement block, 276, 282, 285, 360
 - Terminate*, 291
- Static name assignment, 255, 256
- Step into, 148, 422
- Stop sign, 421
- Stopwatch, 290
- String operation, 124, 133, 223, 235
- Structure, 86, 196, 383
 - Component*, 196
 - End*, 276, 325
 - Include*, 400
 - Internal table*, 382
 - Maintenance screen*, 199
 - Start*, 276
- SUBMIT, 454, 455, 480
 - AND RETURN*, 455
 - VIA SELECTION-SCREEN*, 454
 - WITH*, 454
- Subprogram, 441, 445, 449
 - Call*, 445, 452
 - External*, 452
 - Field*, 449
 - Internal*, 452
- Substring, 132
- Substructure, 198
- SUBTRACT, 111
- Subtraction, 110
- Suffix, 464
- Switch Framework, 515
- SY-DATUM, 214
- SY-INDEX, 291, 294
- Syntax, 386, 387

- Check*, 78, 445
- Error*, 78
- System
 - Table*, 152
- System date, 214
- System table, 356
- SY-SUBRC, 152, 249, 253, 286, 462
- SY-TABIX, 410

T

- Table, 78, 388
 - Activate*, 65
 - Body*, 384, 396, 405, 406
 - Check*, 64
 - Complete*, 63
 - Conversion*, 207
 - Copy*, 175
 - Create*, 47
 - Declare*, 91
 - Definition*, 386
 - Delete*, 207, 209, 211
 - Entry*, 179, 383
 - Error message*, 184
 - Foreign key table*, 189
 - Header*, 384
 - Index*, 409
 - Internal*, 382, 450, 462
 - Maintain*, 47, 179
 - Maintenance dialog*, 66
 - Mode*, 147
 - Modify*, 173
 - Physical conversion*, 205
 - Row*, 178
 - Save*, 49
 - Sorted*, 386, 409
 - Status*, 177
 - T100*, 355, 377
 - TCURC*, 187, 189
 - Temporary*, 382
 - Transparent*, 47
 - Type*, 388, 393
- TABLE, 405
- Table content
 - Display*, 68
 - Modify*, 249
- Table field
 - Maintain*, 51
- Table processing, 386
 - Multidimensional*, 383
 - n-dimensional*, 383
- TABLES, 86, 87
- Tablespace, 65
 - Check*, 152
- Table structure, 94
 - Physical*, 203
- Table work area
 - Buffer*, 253
 - Delete*, 253
- Target field, 112, 129
- Target table, 176
- Technical setting, 64
- Temporary data, 336
- Termination condition, 288, 295, 315
- Termination statement, 292
- Test data record
 - Enter*, 179
- Test mode, 78
- TEST.PDF, 464
- Test system, 25, 26
- Text
 - Country-specific*, 72
 - Free*, 360
- Text element, 235, 342
 - Editing dialog*, 354
- Text field, numeric, 123
- Text object
 - Completing*, 354
- Text pool, 342
 - Activate*, 355
- Text symbol, 342, 354, 360
 - Activate*, 355
 - Edit*, 354
 - Number*, 355
- Time field, 214
 - Define*, 214
 - Fill*, 215
- TIMES, 288
- TITLE, 363
- Toolbar, 77
- Totals
 - Calculate*, 305

Output, 305
 Totals field
 Initialize, 305
 Transaction
 SE11, 46, 175
 SE38, 71
 SE91, 356
 Transaction code, 39, 245
 Transaction data, 49
 Transaction runtime, 480
 TRANSLATE, 333
 Translation, 354
 Transparency, 33
 Transparent table, 47
 Transport, 49
 Transport system, 27
 Trial system, 161, 517
 Trick 35, 218, 220
 Troubleshooting, 142
 Twitter, 518
 TYPE, 105, 123, 331
 Type parameter, 331
 Type conversion, 138, 213
 Automatic, 296, 399
 Type declaration, 87
 Type definition, 388, 389, 395
 TYPES, 388

U

UDDI, 513
 ULINE, 89
 UNDER, 120
 Unicode check, 74
 Unit of currency, 269
 UPDATE, 249, 254
 Upgrade, 194
 Upper and lower case, 329
 Uppercase, 329
 Uppercase and lowercase, 406
 Uppercase and lowercase letter, 124
 User exit, 515
 User master record, 245
 USING, 449

V

Validation, 183, 192
 Value
 Negative, 107
 Transfer, 454
 Valid, 186
 VALUE, 123, 214, 432
 VALUE CHECK, 332
 Value range, 58
 Value table, 60, 187, 188, 189
 Variable, 104, 481
 Name, 104
 Variant, 74, 347
 Create, 347
 Edit, 349
 Property, 350
 Protect, 351, 352
 Start using, 74
 VIA SELECTION-SCREEN, 454, 455

W

Watchpoint, 142, 154
 Create, 154
 Mode, 154
 Save, 158
 Web Dynpro, 160
 ABAP, 509
 Java, 509
 Web service, 511
 WHEN OTHERS, 285
 WHERE, 257, 287, 410, 421, 431
 Where-used list, 78, 209, 210
 WHILE, 290
 Wildcard, 457
 WITH, 359, 454
 WITH FRAME, 363
 WITH KEY, 404, 409
 WITH UNIQUE KEY, 389
 Work area, 86, 250, 254, 388, 391, 398, 404, 406, 420
 Define, 250, 389
 Initialize, 262, 413
 Working memory area, 456, 479

Work process, 29, 247
WRITE, 83, 88, 89, 119
WRITE UNDER, 120

X

X buffer, 81

Y

Y buffer, 81

Z

Z buffer, 81
ZIP code, 108