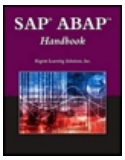


SAP ABAP Handbook

by Kogent Learning Solutions, Inc.

Chapter 5: ABAP Dictionary

Highlights

ABAP Dictionary, a tool of ABAP Workbench, is used to store the description of data definitions. Using ABAP Dictionary, you can create and maintain user- defined types, such as data elements, structures, and table types, which are used in ABAP programs to declare their respective objects. In addition, ABAP Dictionary is used to create and maintain database objects, such as tables and views. ABAP Dictionary also provides several services that support program development. For example, you can set or release locks while executing database objects, define an input help (F4 help), or attach a field help (F1 help) with a field on a screen.

ABAP Dictionary is integrated completely in the development and runtime environments of the SAP system. This means any changes made to a data definition are reflected in all the related ABAP programs, function modules, menus, and screens. In addition, ABAP Dictionary ensures data integrity and manages all the data definitions without redundancy.

In this chapter, you learn about ABAP Dictionary, which acts as a storage medium for the data of an SAP system. This chapter also discusses domains, which are used to define the characteristics of fields, such as data type and value range. Next, you learn about data elements, which are used to assign a description to a field in a database table. You also learn about different kinds of tables used in ABAP Dictionary and how to create them. This chapter also describes foreign keys used to relate tables, and different kinds of table fields and views. Next, you learn about search helps, which help view the possible set of values for a field. The chapter concludes with a discussion of the Lock Object, used to synchronize simultaneous access to the same data records by multiple users.

Overview of ABAP Dictionary

ABAP Dictionary is a central storage area for the description of various types of repository objects in an SAP system. The repository objects include database tables, views, data types, type groups, domains, search helps, and lock objects. You can access the repository objects in an SAP system from the initial screen of ABAP Dictionary. The initial screen of ABAP Dictionary can be opened from the SAP `Easy Access` screen, by entering the `SE11` transaction in the `Command` field on the standard toolbar and then pressing the `ENTER` key, or by clicking the `Enter` icon. Figure 5.1 displays the initial screen of ABAP Dictionary:

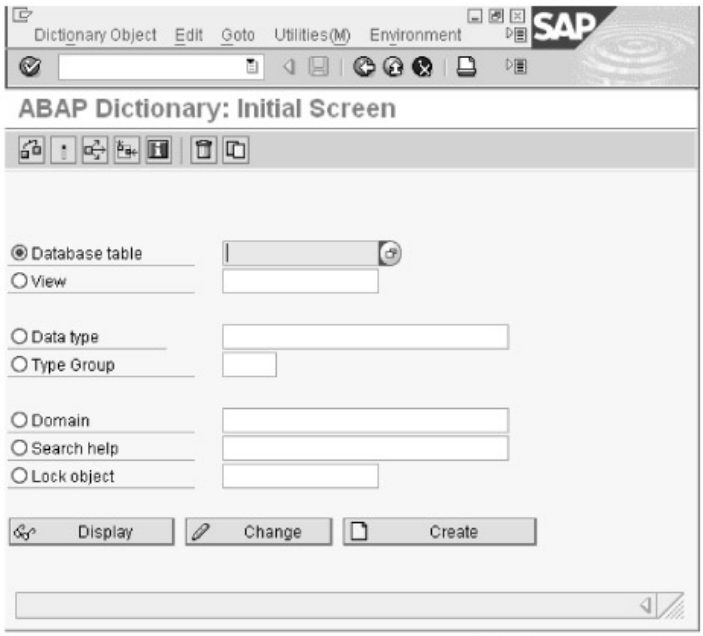


Figure 5.1: The initial screen of ABAP dictionary

Table 5.1 lists the ABAP Dictionary repository objects with their descriptions

Table 5.1: Descriptions of the ABAP dictionary repository objects

ABAP Dictionary Repository Object	Description
Domain	Describes the technical attributes of a field, such as the data type of the field or a fixed range of values accepted by the field.
Data Type	Creates user-defined data types, such as data elements, structures, and tables.
Type Group	Creates a group of data types in ABAP Dictionary.
Database Table	Helps design tables. Note that the name of the table should begin with either the letter Y or Z, and its length can be up to 16 characters, including Y and Z.
View	Retrieves data from database tables. A view acts as a virtual table because it does not store the data physically.
Search Help	Defines input help (F4 helps) for the fields of a database table. Input help provides a set of valid values that you can enter for a field.
Lock Object	Synchronizes simultaneous access of the same set of data records by multiple users.

Now, let's explore each of these objects in detail.

Exploring Domains

A domain is used to define a data type, length, and value range for a table field. The fields of a table and the fields of a structure that uses a domain automatically get the value range as defined in the domain. In other words, the relationship between the field and the domain is defined by the data element of the field. If you make any changes in a domain, the attributes of the fields related to that domain change accordingly. You can link any number of data elements with a single domain. Figure 5.2 displays the relationship between a data element and a domain:

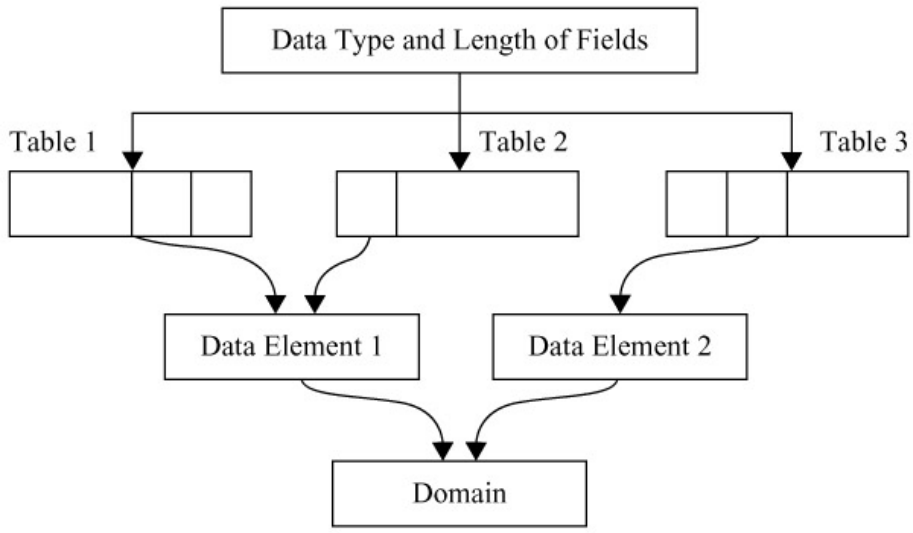


Figure 5.2: Flowchart showing the relationship between domain, data elements, and tables

You can also set a filter to check the values entered in a field. For this, you need to create a table called the Value table which checks every field value against some specified criteria mentioned in the Value table. In addition, a conversion routine can be assigned to a domain, which converts field values from display format to internal format (as discussed later in this chapter) for the fields referring to a domain.

Now, let's discuss the following topics in the context of domains:

- Describing fixed values for domains
- Exploring conversion routines for domains
- Creating a domain
- Modifying the existing domain

- Deleting a domain

Describing Fixed Values for Domains

You can assign certain fixed values specifying a value range, such as data type and length, to a domain. These fixed values are used as an input check for the fields on a screen; that is, they provide a set of data for users to select among. Fixed values also act as the source of input help (F4) provided to the fields in case no input (F1) help is maintained for the fields. Fixed values can be defined by specifying either single values or the lower and upper limits for the fields. In addition to this, you can also assign a short text to the fixed values assigned. These short descriptions can be viewed when the input help for the field is called. Some data types for which the fixed values can be defined are `CHAR`, `NUMC`, `DEC`, `INT1`, `INT2`, and `INT4`.

Sometimes you need to specify a large number of fixed values. In such cases, you can store these values in the Value table, which are used to verify the fields against the values contained in the fields of some other table.

Exploring Conversion Routines for Domains

The data entered in an SAP system is not stored in the original format in the SAP database; however, it needs to be in a format compatible with the SAP database. One example of this is the FTIME screen field, which stores the time of login in the HHH:MM (hours:minutes) format. Now, let's suppose that a user enters a login time as 2:45 in the screen field. When this value of the screen field is stored in the SAP database, it is converted to an integer number, such as 165 minutes (2 hours and 45 minutes). However, when you retrieve this data, it is again displayed in its original format, HHH:MM (hours:minutes). The conversion of values from one format to another is performed by the conversion routine assigned to a domain, which in turn is related to the screen field.

Conversion routines are defined by a five-place name and are stored as a group of two function modules, `CONVERSION_EXIT_xxxxx_INPUT` and `CONVERSION_EXIT_xxxxx_OUTPUT`.

The INPUT function module converts the contents of a screen field from the display format to the SAP internal format, and the OUTPUT function module converts the content from the SAP internal format to the display format.

Note The `CONVERSION_EXIT_xxxxx_INPUT` and `CONVERSION_EXIT_xxxxx_ OUTPUT` function modules have a fixed naming convention.

Creating a Domain

Domains are created to define the technical characteristics of a field.

Perform the following steps to create a domain in ABAP Dictionary:

1. Select the `Domain` radio button on the initial screen of ABAP Dictionary, as shown in Figure 5.3:
2. Enter the name of the domain to be created. In our case, we enter the name as `ZDomain` (see Figure 5.3).
3. Click the `Create` button (Figure 5.3).

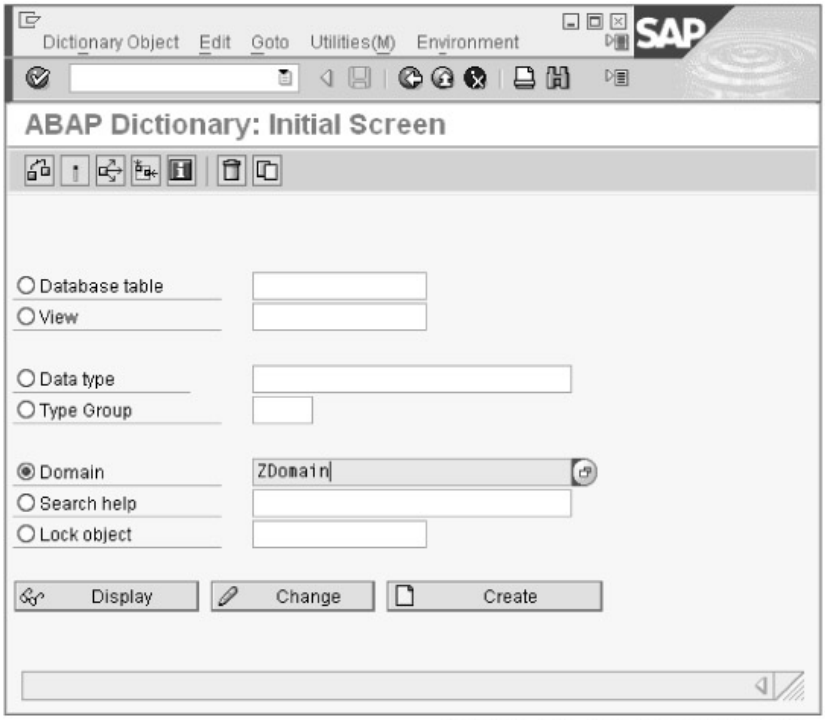


Figure 5.3: Creating a domain

The `Dictionary: Maintain Domain` screen appears, as shown in Figure 5.4.

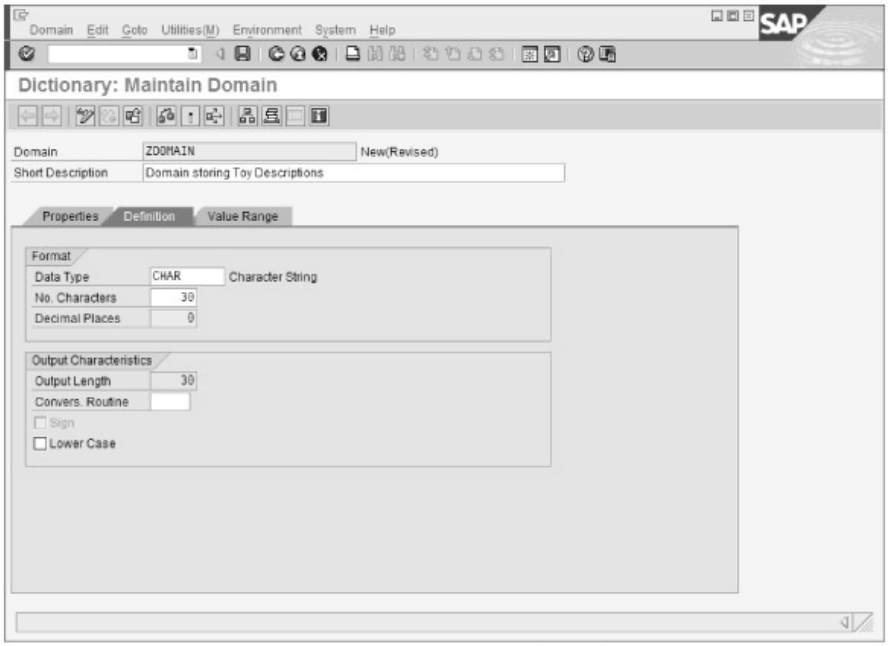


Figure 5.4: Screen for domain maintenance

4. In the `Dictionary: Maintain Domain` screen, enter a short description for the domain in the `Short Description` field. In our case, we have entered the short description as Domain storing Toy Descriptions. In the `Format` group box, enter CHAR and 30 in the `Data Type` and `No. Characters` fields, respectively. In addition, specify the value 30 in the `Output Length` field of the `Output Characteristics` group box, as shown in Figure 5.4:

5. Select the `Value Range` tab. The screen containing the options related to the `Value Range` tab appears (see

Figure 5.5). Enter the values in the `Single Vals` section, as shown in Figure 5.5:

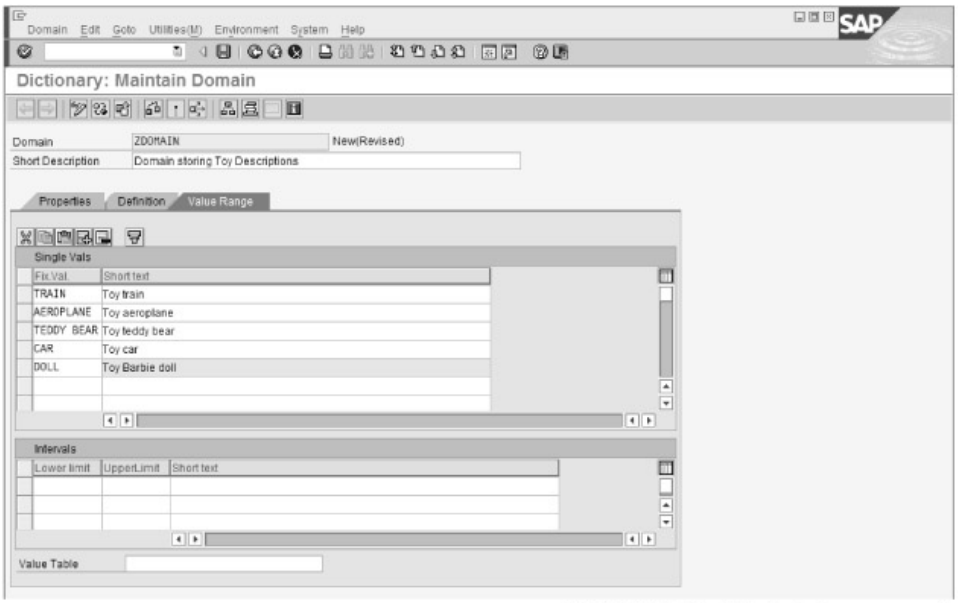


Figure 5.5: The maintenance screen for the value range tab

Note It is not mandatory to define a value range for a domain.

The screen shown in Figure 5.5 displays the fixed values along with their short description under the `Single Vals` section.

6. Click the `Save` icon to save the domain. The `Create Object Directory Entry` dialog box appears, as shown in Figure 5.6:
7. Click the `Local Object` button in the `Create Object Directory Entry` dialog box.
8. Finally, click the `Check` and `Activate` icons to check the syntax of the `ZDOMAIN` domain and activate it, respectively.

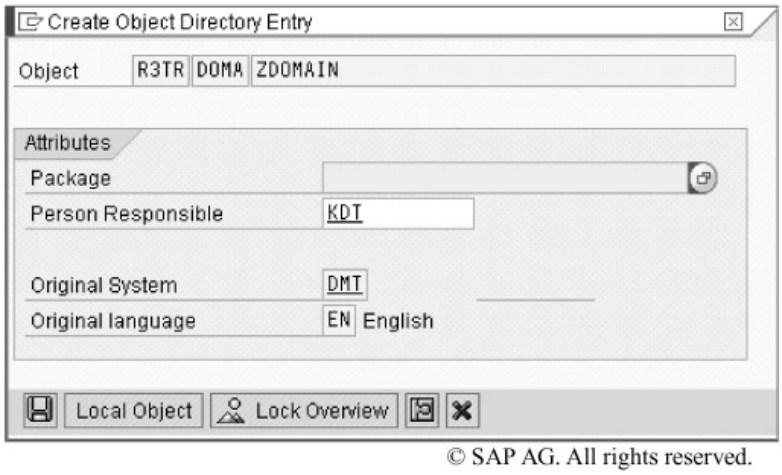


Figure 5.6: The create object directory entry dialog box

The `ZDOMAIN` domain is now activated and ready to be used with any number of fields. It must be noted that the domain is saved as a local object, which means that it cannot be shared among multiple SAP servers.

Note You can make repository objects accessible over a network by storing them in a package.

Perform the following steps to create a package:

1. Open Object Navigator by entering the SE80 transaction code on the SAP Easy Access screen.
2. Select Package as the object type and enter a name for the package to be created. In this case, we have named the package as ZKOG_PCKG, as shown in Figure 5.7:
3. Click the Display icon.

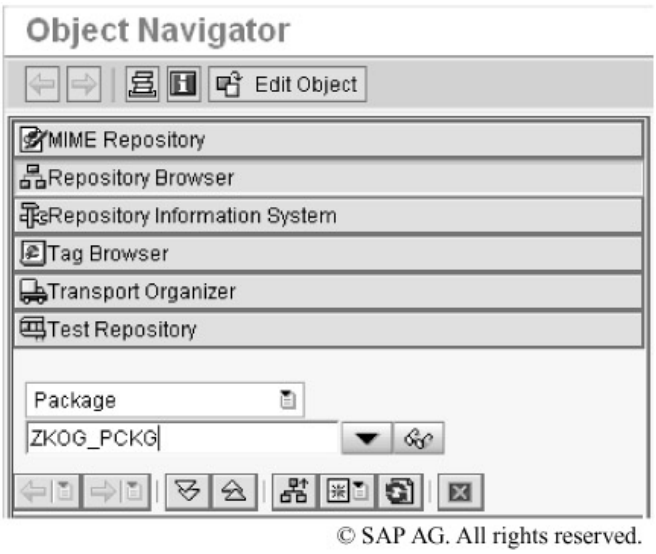


Figure 5.7: Creating a package

The Create Object dialog box appears, as shown in Figure 5.8:

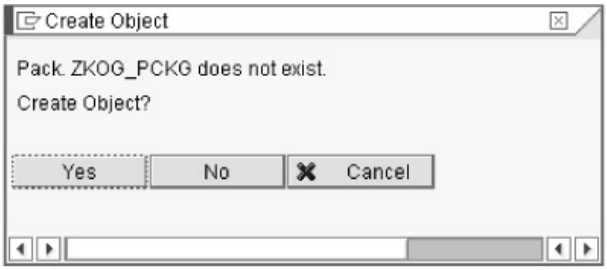


Figure 5.8: Creating an object

4. Click the Yes button to create the package.

The Package Builder: Create Package dialog box appears, as shown in Figure 5.9:

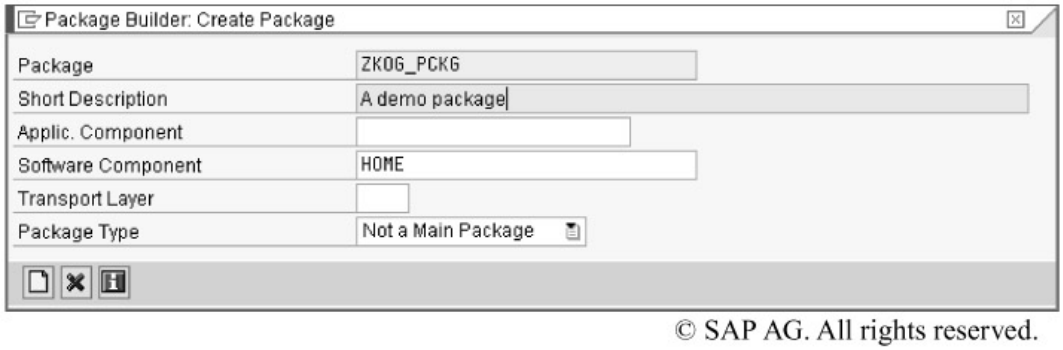


Figure 5.9: Creating a package

5. Enter the text, A demo package, in the `Short Description` field and HOME in the `Software Component` field.
6. Select "Not a Main Package" as the type of the package and then click the `Save` icon to save the package (see Figure 5.9).

The `Prompt for Local Workbench Request` dialog box appears, as shown in Figure 5.10:

The screen shown in Figure 5.10 displays the name and description of the package, along with the automatically generated workbench request number.

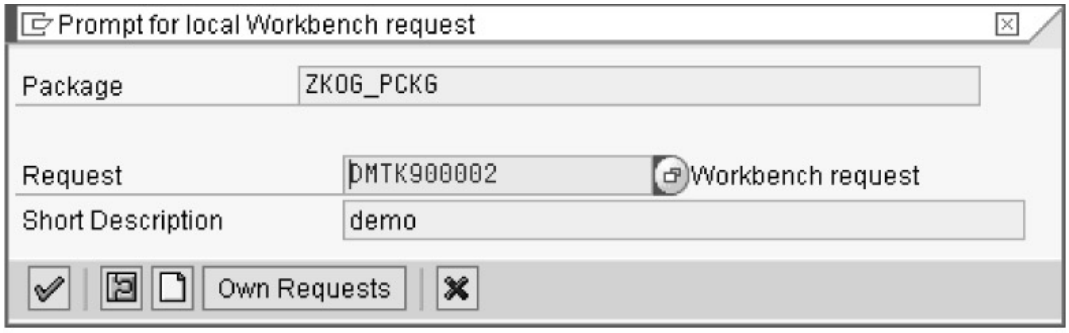


Figure 5.10: Displaying the request number

7. Click the `Continue` icon (Figure 5.10)

The `Object Navigator` screen with the package name, `ZKOG_PCKG`, appears, as shown in Figure 5.11:

The `ZKOG_PCKG` package is now created and can be used for storing repository objects.

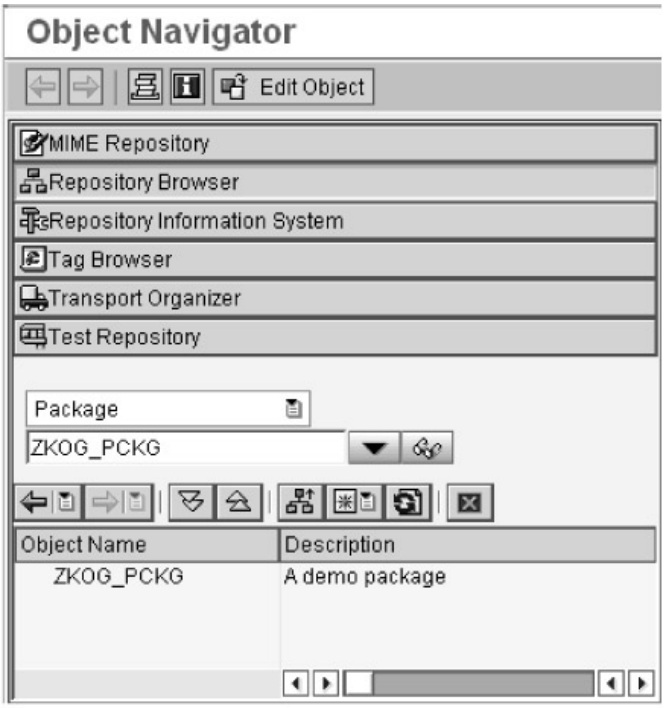


Figure 5.11: Showing the created package

Note Hereafter, we will save all the repository objects in this package. The repository objects can be saved in a package by providing the name of the package in the `Package` field (Figure 5.6) and then clicking the `Save`

icon. In addition, any repository object created in the forthcoming chapters will also be saved in this package.

Modifying the Existing Domain

You can modify the characteristics of an already created domain. The attributes, such as data type, number of characters, and number of decimal places, can be modified by making changes in the relevant fields in the `Dictionary: Maintain Domain` screen (shown previously in Figure 5.4). The changes are reflected in all the database tables that contain any field related to the modified domain.

You can modify an existing domain by performing the following steps:

1. Select the `Domain` radio button on the initial screen of ABAP Dictionary, and enter the name of the domain that has to be modified. In our case, we enter the name as `ZDomain` and click the `Change` button (Figure 5.3).

The `Dictionary: Maintain Domain` screen appears (see Figure 5.4).

2. In the `Dictionary: Maintain Domain` screen, you can modify the short description for the domain in the `Short Description` field. In our case, we have entered the short description as Creating a Domain. In the `Format` group box, change the values of the `Data Type` and `No. Characters` fields to "CHAR" and "40", respectively. In addition, specify the value 40 in the `Output Length` field of the `Output Characteristics` group box, as shown in Figure 5.12:

3. In the `Dictionary: Maintain Domain` screen, select the `Value Range` tab and modify the values in the `Single Vals` section, as shown in Figure 5.13:

4. Finally, click the `Save` icon to save the domain, and click the `Check` and `Activate` icons to check the syntax of the `ZDOMAIN` domain and activate it, respectively.

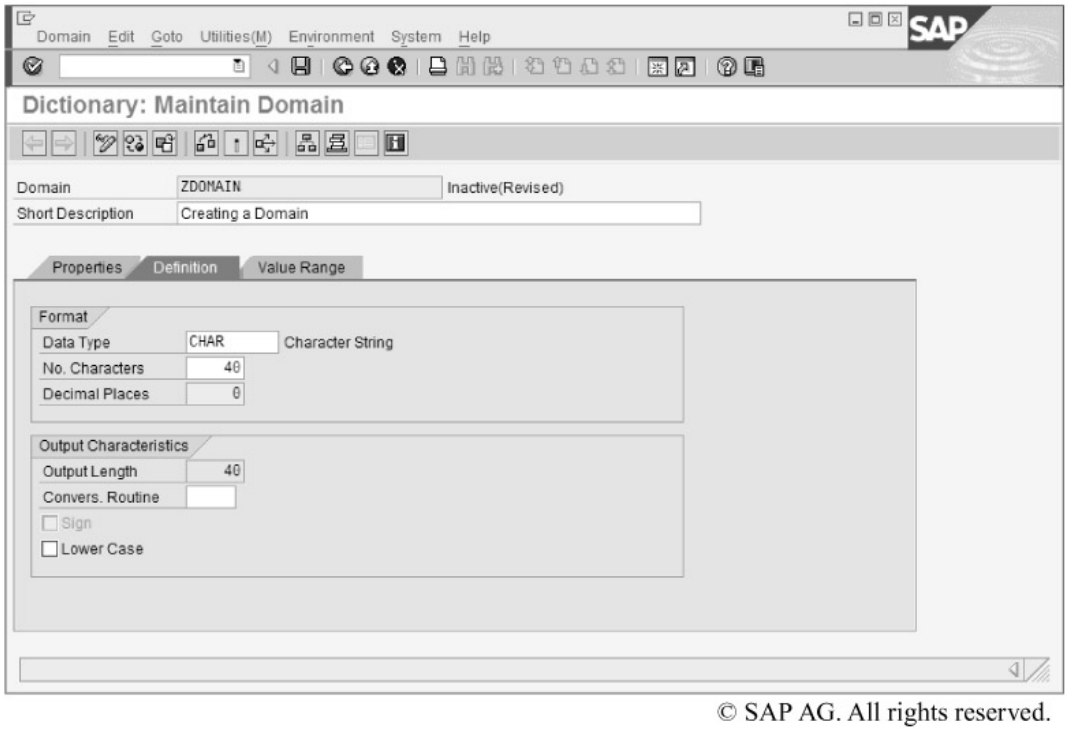


Figure 5.12: Modifying the description and other field values of a domain

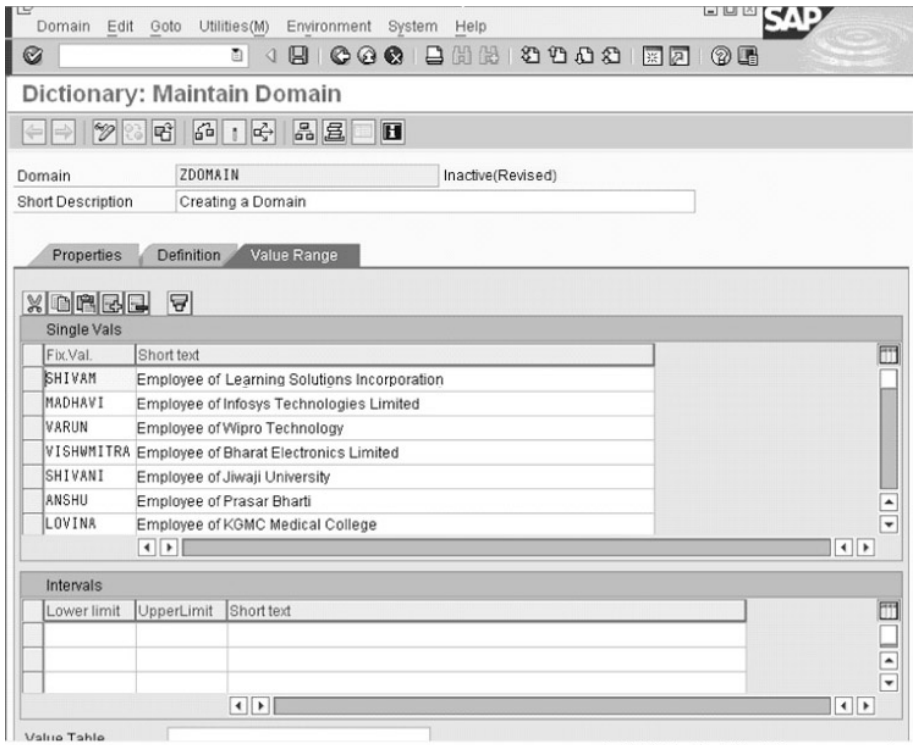


Figure 5.13: Modifying the range of values of a domain

You should note that sometimes when you modify a domain related to a database table, the modification can affect that database table. This is because the modification in a domain attribute can also affect the attributes of the database table, such as data type of fields, number of characters to be displayed in a field, output attributes, and value table.

Perform the following steps to view the database tables related to a specific domain after modifying the domain:

1. Select the `Domain` radio button on the initial screen of ABAP Dictionary. Enter the name of the domain whose attributes need to be changed (in this case, `ZMYDOMAIN`), in the `Domain` field and click the `Display` button.

The `Dictionary: Maintain Domain` screen appears.

2. In the `Dictionary: Maintain Domain` screen, click the `Where-Used List` icon.

The `Where-Used List Domain` dialog box appears, as shown in Figure 5.14:

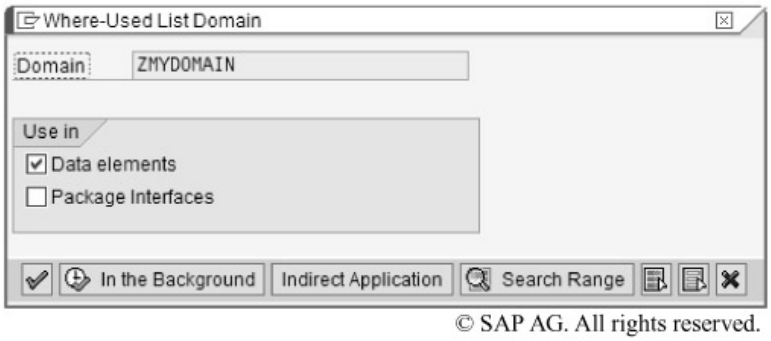


Figure 5.14: The where-used list domain dialog box

3. In the `Where-Used List Domain` dialog box, click the `Indirect Application` button (Figure 5.14).

The `Where-Used List Domain` dialog box displays the information related to the domain, as shown in Figure 5.15:

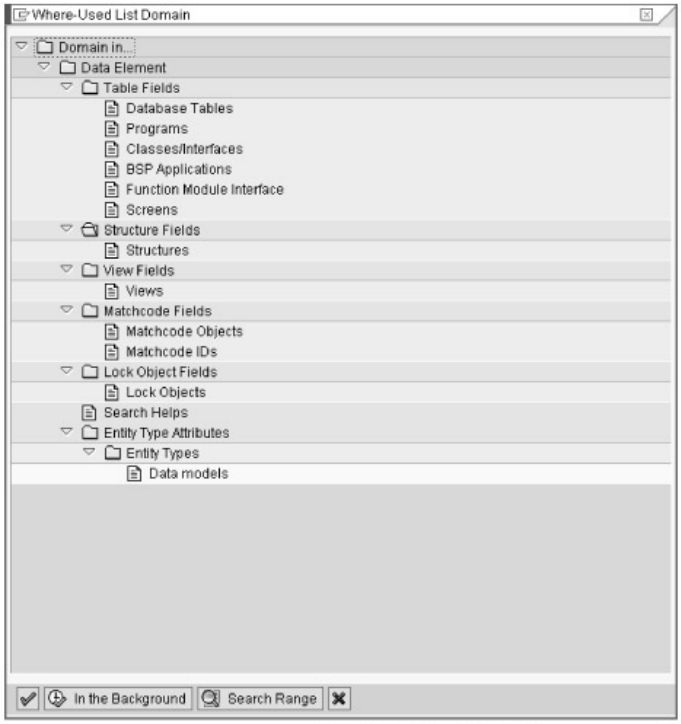


Figure 5.15: Selecting database tables

4. Select the `Database Tables` option from the `Table Fields` folder and then click the `Continue` icon (see Figure 5.15).

The `Where-used Domain ZMYDOMAIN in Database tables (2 Hits)` screen appears, displaying the names of dependent database tables using the domain.

Figure 5.16 displays a list of the database tables that are related to the `ZMYDOMAIN` domain.

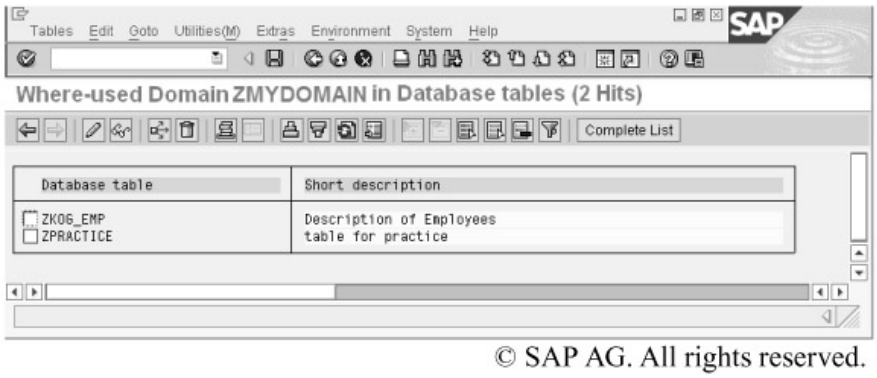


Figure 5.16: List of tables

Deleting a Domain

You can also delete an existing domain by performing the following steps:

1. Select the `Domain` radio button on the initial screen of ABAP Dictionary and then enter the name of the domain that needs to be deleted.
2. Click the `Delete` icon. The `Delete Domain` dialog box, which is followed by the name of the domain, appears. The `Delete Domain` dialog box contains three buttons, `Yes`, `No`, and `Cancel`.
3. Click the `Yes` button to delete the selected domain.

Exploring Data Types

A data type is a repository object of ABAP Dictionary used to create user- defined data types, such as data elements, structures, and tables. User-defined data types are the type definitions in ABAP Dictionary and can be used in an ABAP program with the TYPE clause. The following are the three different kinds of data types:

- Data elements
- Structures
- Table types

Data Elements

Data elements describe individual fields of a table in a database and are used to specify the types of columns in the database. The following are the two different kinds of data elements:

- **Elementary types**—Refer to data types that have semantic attributes, such as built-in data type, length, and number of decimal places. These data types can be assigned to a data element in either of the following ways:
  - **Assigning a predefined ABAP Dictionary type—** Specifies that the data element can be assigned with the predefined ABAP Dictionary types, such as CHAR and NUMC. Apart from this, you can also assign the length to the selected data type.
  - **Assigning a domain—** Specifies that the data element being created inherits the technical characteristics related to the elementary types from a predefined domain. You can use a domain with any number of data elements.
- **Reference types**—Refer to single fields that contain references to global classes and interfaces from the ABAP class library.

After creating a data element, you can also assign documentation and text information to it. The documentation of a data element is the description of the data element, which serves as an online aid to the user. The documentation entered for a data element is displayed when the user selects F1 help on fields referring to that data element. If there is no documentation for the data element, only the short text of the data element appears when the F1 key is pressed.

The documentation status is used to determine whether or not the documentation for the data element has been written. In addition to this, the documentation status also explains whether the documentation is required or not.

Table 5.2 lists the description of the different kinds of possible status entries:

Table 5.2: Possible Documentation Status

Documentation Status	Description
Object Requires Documentation	Shows whether or not documentation already exists. If it does not exist, it should be written.
Object Is not Used in Any Screens	Shows that the documentation does not exist and is not required, because this data element is not used with any of the existing fields.
Object Explained Sufficiently by Short Text	Means that the short text provided for the data element is sufficient to explain the purpose for its creation. In this case, the documentation does not exist and is not required.
Documentation Is Postponed Temporarily	Means that the data element requires documentation. In this case, the documentation has not yet been written.

Field Labels

Field labels are used to assign text information to data elements. You already know that a field is assigned to a data element. The text information related to the data element is referred to by the field and is displayed on the screen in place of the field name. If a field label is not assigned to a data element, the text entered in the short description field of the data element acts as the default field label for that field. The following are the two types of field labels:

- **Short, medium, and long fields—** Refer to keywords used to define the lengths of different fields.
- **Header fields—** Refer to information displayed above a column. The length of the header field must not exceed the length assigned to the data element.

Creating a Data Element

Perform the following steps to create a data element in ABAP Dictionary:

1. Open the initial screen of ABAP Dictionary by entering the `SE11` transaction code in the `Command` field and click the `Enter` icon or press the ENTER key.
2. Select the `Data type` radio button and enter a name for the data type, as shown in Figure 5.17:

Figure 5.17: Initial screen of ABAP dictionary

In Figure 5.17, we enter the name of the data type as Zdelement.

3. Click the `Create` button. The `Create Type ZDELEMENT` dialog box appears, as shown in Figure 5.18:
4. Select the `Data element` radio button and click the `Continue` icon.

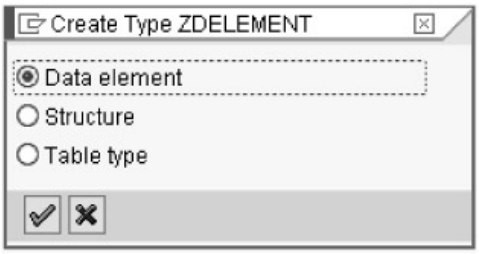


Figure 5.18: The create type dialog box

The `Dictionary: Maintain Data Element` screen appears, as shown in Figure 5.19:

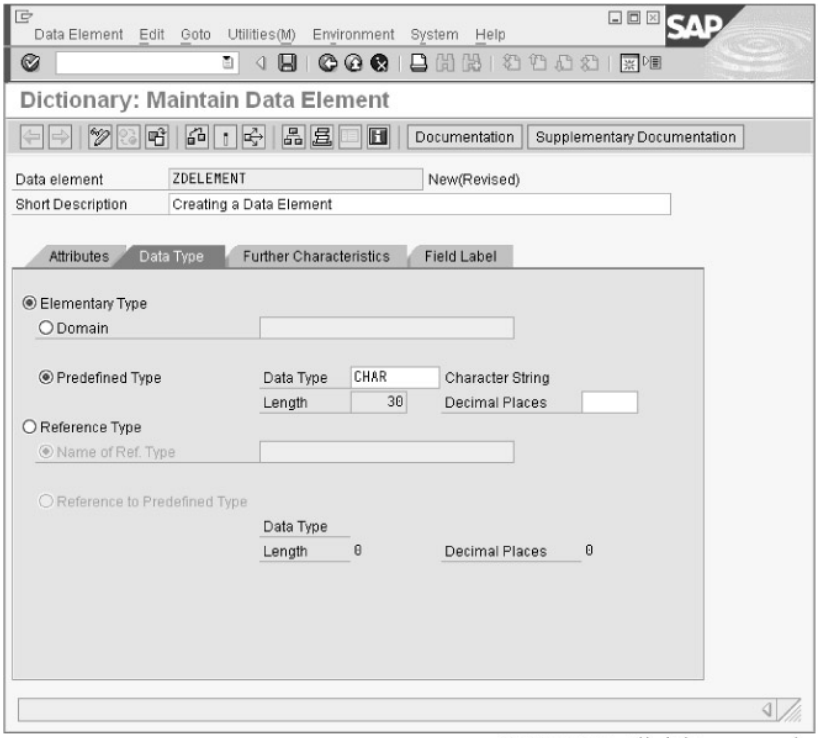


Figure 5.19: The dictionary—maintain data element screen

5. Enter a short description for the data element in the `Short Description` field. In this case, we have entered Creating a Data Element.
6. Select the `Elementary Type` and `Predefined Type` radio buttons, as shown in Figure 5.19.
7. Enter `CHAR` in the `Data Type` field and 30 in the `Length` field.

Note Sometimes the technical attributes of a data element are defined within a domain. In such cases, you only need to select the `Domain` radio button and enter the domain name in the corresponding field to access the data element.

8. Select the `Field Label` tab, which provides the options to define the text information related to the data element created, as shown in Figure 5.20:
9. Enter a name for the field and specify the desired length for it. In this case, we have entered Employee Name in the `Field Label` column and 30 in the `Length` column of the `Long` field.
10. Click the `Save` icon to save the created data element in a package. In our case, we save the created data element in the `ZKOG_PCKG` package.
11. Now, select `Go To > Documentation > Change` from the `Dictionary: Maintain Data Element` screen (see Figure 5.20) to define the documentation for the data element.

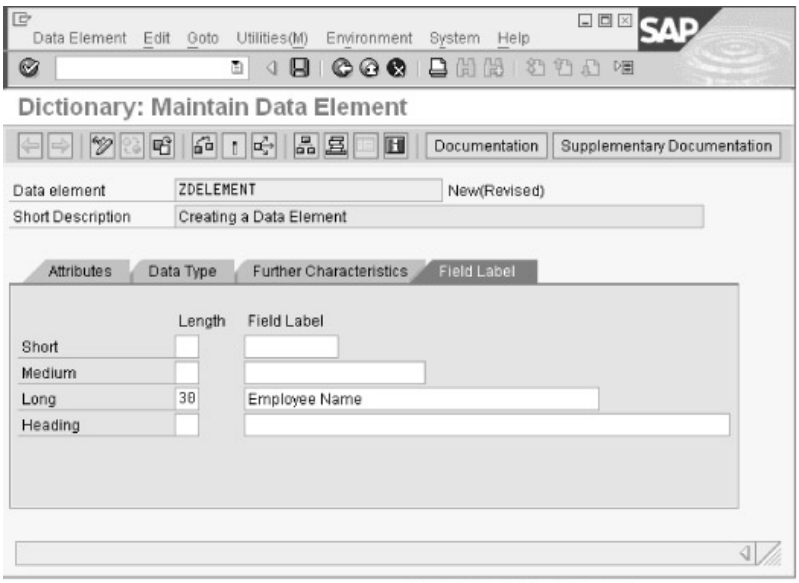


Figure 5.20: The field label tab

The `Change Data element: ZDELEMENT Language EN` screen appears, as shown in Figure 5.21:

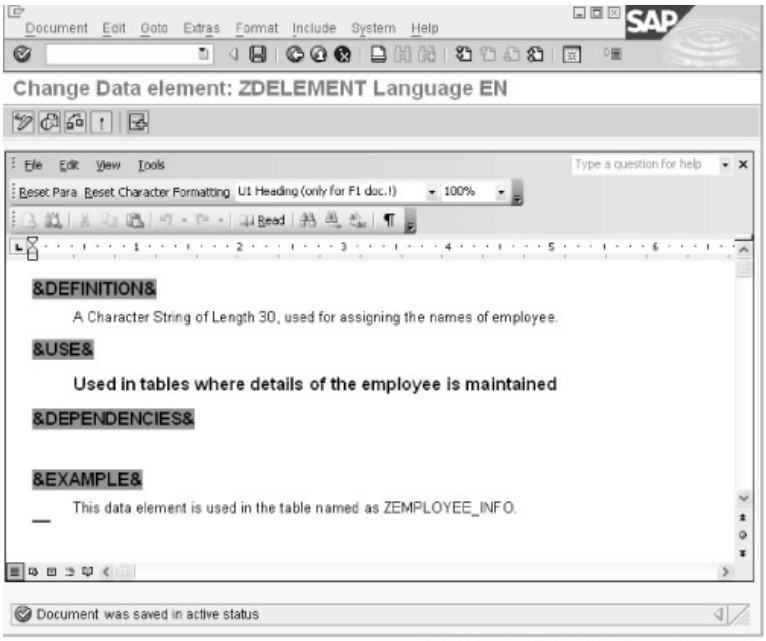


Figure 5.21: Documentation for the data element

12. Enter the documentation related to the data element under the `&DEFINITION&`, `&USE&`, `&DEPENDENCIES&`, and `&EXAMPLE&` head ings, as shown in Figure 5.21.
13. Click the `Save Active` icon to save the documentation.
14. Click the `Back` icon to navigate to the maintenance screen of the data element (Figure 5.20).
15. Click the `Check` icon and then click the `Activate` icon.

The documentation for the data element is now activated and can be displayed by pressing the F1 key.

Structures

A structure is a data type in ABAP Dictionary composed of several components, including data elements, table types, and database tables. Structures are very similar to database tables, but a structure neither has a primary key nor technical characteristics associated with it. Structures are used when you want to define the same work area in multiple programs. For example, you use structures when you want to write records to a sequential file using one program and then read the same set of records by using another program. Notice that both programs must recognize the layout of the records being read and written. For this, you define the records layout in ABAP Dictionary with the help of structures. This defined structure can be included in both the programs with the help of the TABLES statement.

A structure can be nested within another structure. Note that whenever a structure is created in ABAP Dictionary, each of its components must be assigned a name and a data type. The structures created in ABAP Dictionary can be referred to in an ABAP program with the TYPE clause.

Table Types

A table type is a data type in ABAP Dictionary that describes the structure and functional attributes of an internal table. An internal table is a temporary table that exists only during the runtime of an ABAP program. In an internal table, all the records have the same structure and key. The record of the internal table is discarded when the execution of the program ends.

Note You learn more about internal tables in Chapter 7.

A table type is defined by the following attributes:

- **Row or line type—** Defines the structure and data type attributes of a line of an internal table.
- **Options—** Defines the access mode (used to manage and access data) in an internal table
- **Key—** Defines the key definition and key category of an internal table.

Figure 5.22 shows the attributes of the TABTYPE table type in ABAP Dictionary:

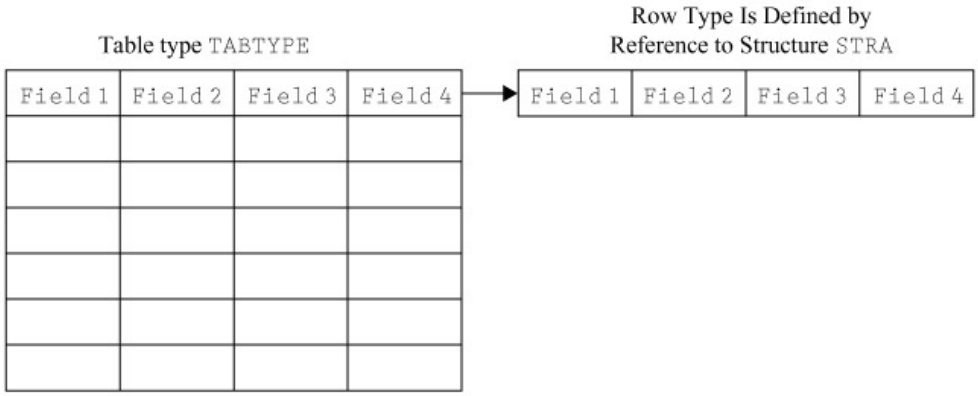


Figure 5.22: Creation of a table type

In Figure 5.22, STRA represents the row type of the TABTYPE table type. It is related to the `Field 1`, `Field 2`, `Field 3`, and `Field 4` fields of the TABTYPE table type. The TABTYPE table type can be used in an ABAP program by using the following syntax:

```
DATA <name> TYPE TABTYPE
```

In the preceding syntax, the `<name>` expression is the name of the internal table to be created in an ABAP program. The internal table is similar to the TABTYPE table type created in ABAP Dictionary.

Exploring Type Groups

Type groups, another repository object of ABAP Dictionary, allow you to define data types directly in ABAP Dictionary, instead of defining them in an ABAP program. The name of a type group in ABAP Dictionary must have a maximum length of five characters. Every data type name defined in a type group must begin with the name of the type group followed by an underscore sign. These defined data types can be used in an ABAP program with the help of the `TYPE- POOLS`

statement.

To create a type group, select the `Type Group` radio button in the initial screen of ABAP Dictionary (see Figure 5.17), enter the name of the type group, and click the `Create` button. Then, enter the following code snippet in the maintenance screen of the ZTYPE type group:

```
TYPE-POOL ZTYPE.
TYPES: ZTYPE_NAME(30) TYPE c,
        ZTYPE_AGE(3) TYPE i.
```

In this code snippet, two data types are created. The declared type group (that is, ZTYPE), can be used in an ABAP program, as shown in the following code snippet:

```
REPORT ZSHIVAM.
TYPE-POOLS ZTYPE.
DATA: NAME TYPE ZTYPE_NAME,
       AGE TYPE ZTYPE_AGE.
```

In the preceding code snippet, the ZTYPE type group is included in the ZSHIVAM program with the `TYPE-POOLS` statement. Therefore, the data types in the ZTYPE type group can be used in the ZSHIVAM program with the help of the DATA statement.

Exploring Database Tables

In ABAP Dictionary, tables can be defined independent of the database. When a table is activated in ABAP Dictionary, similar copy (that is, definition) of its fields is created in the database as well. In other words, the tables defined in ABAP Dictionary are translated automatically into the format that is compatible with the database because the definition of the table depends on the database used by the SAP system.

In ABAP Dictionary, a table definition consists of the following components:

- **Fields—** Store information.
- **Foreign keys—** Define the relationships between tables.
- **Technical settings—** Specify the settings of the table, such as data class and size category.
- **Indexes—** Helps retrieve the rows within a table.

Figure 5.23 displays the definition of the table in a database as compared to its definition in ABAP Dictionary:

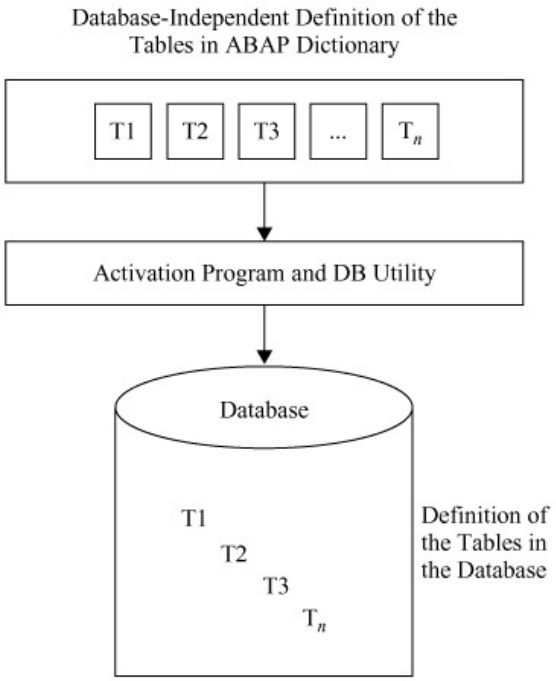


Figure 5.23: Database-independent table definition

A table can contain one or more fields, each defined with its data type and length. The large amount of data stored in a table is distributed among the several fields defined in the table.

In the next sections, we discuss the following topics in the context of database tables in ABAP Dictionary:

- Types of tables
- Types of table fields
- Technical settings of a table
- Creating tables
- Relating tables by using foreign keys

Types of Tables

In ABAP Dictionary, you can create three types of tables with different characteristics:

- Transparent tables
- Pooled tables
- Cluster tables

Apart from these, you can create some special table types, such as table pools (also called just pools) and table clusters. These two special table types are used to store information about other tables, such as internal control information, which includes screen sequences, program parameters, temporary data, and continuous texts (such as documentation). The tables assigned to a table pool or table cluster are referred to as pooled tables and cluster tables, respectively.

Transparent Tables

A transparent table forms a one-to-one relationship with the table definition in the database. Transparent tables are used to hold the application data, which represents master data or transaction data used by an application. An example of master data is the table of vendors (also called vendor master data) or the table of customers (also called customer data). An example of transaction data is an order placed by a customer or an order placed with a vendor.

Pooled Tables

A pooled table in ABAP Dictionary forms a many-to-one relationship with the table definition in the SAP database. It means that for a single table defined in the database, there are many tables in ABAP Dictionary. The names of the tables in ABAP Dictionary must differ from those of the tables stored in the database.

All pooled tables are stored in the SAP database in a single table, called a table pool, which is a database table with a special structure that can store the data of multiple pooled tables.

The table pool definition consists of two key fields (`Tabname` and `Varkey`) and a long argument field (`Vardata`). Table 5.3 lists the descriptions of fields in a table pool:

Table 5.3: Fields of a table pool and their data types

Field	Data Type	Meaning
Tabname	CHAR (10)	Holds the name of a pooled table.
Varkey	CHAR (*n*)	Stores the entries of the key fields of a pooled table as a string. The maximum length of the string is specified by the *n* variable.
Dataln	INT2(5)	Stores the length of a string in `Vardata`.
Vardata RAW	RAM(*n*)	Holds the entries from all the data fields of a pooled table as a string, where the maximum length of the string is determined by the database system being used.

The length of the name of a pooled table must not exceed 10 characters. `Varkey` is a character field, so all the key fields of a pooled table must have character data types, such as `CHAR`, `NUMC`, and `CLNT`. The total length of all the key fields or all the data fields of pooled tables must not exceed the length assigned to the `Vardata` or `Varkey` field.

Figure 5.24 displays the relationship between table pools and pooled tables:

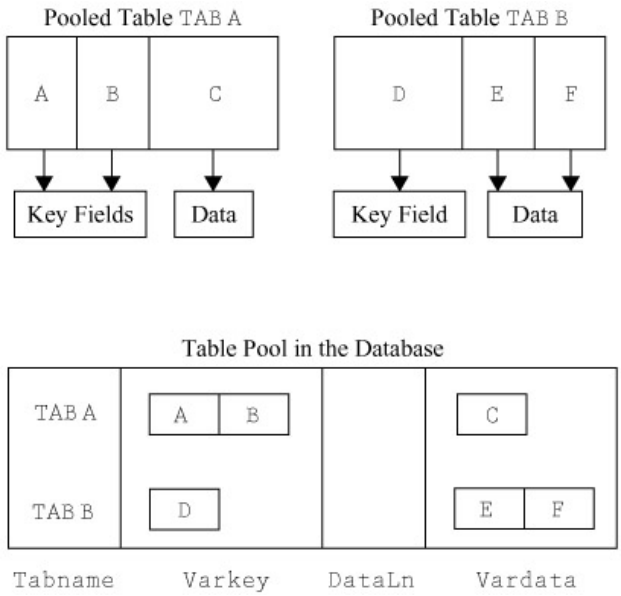


Figure 5.24: Relationship between the table pool and pooled tables

In Figure 5.24, you can see two pooled tables, `TAB A` and `TAB B`. The `TAB A` table consists of two key fields, `A` and `B`, which hold values in the fields as `C`. The `TAB B` table consists of a key field called `D` and the data are `E` and `F`. Notice that both pooled tables are stored in a table pool in the database.

Cluster Tables

Similar to a pooled table, a cluster table also forms many-to-one relationships with table definitions in the SAP database. Cluster tables are stored in a single table in the SAP database, called the table cluster. Cluster tables are used when the constituent tables have a common primary key. The data from these common primary key fields can be accessed simultaneously. The data contained in the rows that have the common primary key can be combined into a single row in the table cluster.

Table 5.4 lists the description of fields in a table cluster:

Table 5.4: Fields and data types in a table cluster

Field	Data Type	Meaning
`CLKEY1`	*	Denotes the first key field.
`CLKEY2`	*	Denotes the first key field.
`CLKEYn`	*	Denotes the *n*th key field.
`Pageno`	INT2(5)	Denotes the number of continuation records.
`Timestamp`	CHAR(14)	Describes time stamps.
`Pagelg`	INT2(5)	Denotes that the length of the associated string is stored as the `Vardata` type
`Vardata`	RAW(*n*)	Contains the entries from the data fields of the assigned cluster tables

Now, let's try to understand the concept of table clusters with the help of Figure 5.25:

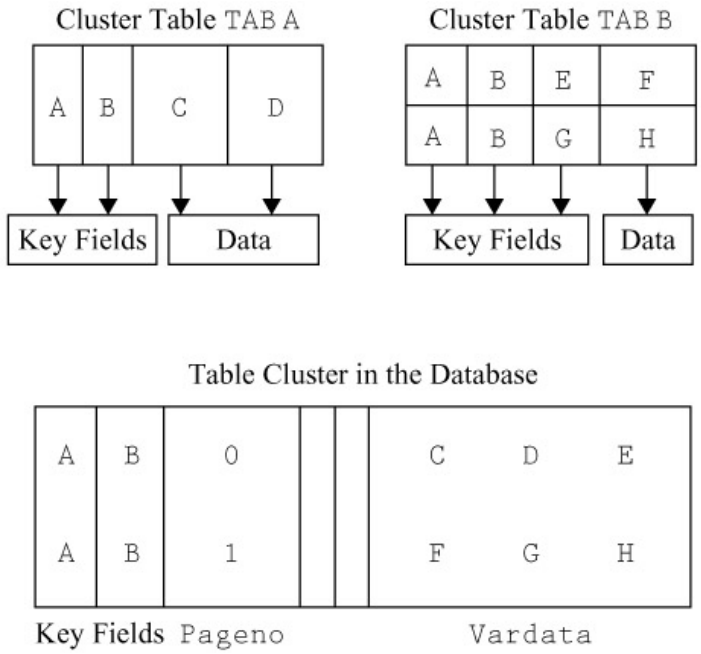


Figure 5.25: Table cluster and cluster tables

Figure 5.25 shows two cluster tables, `TAB A` and `TAB B`. The `TAB A` table consists of two key fields, `A` and `B`, which hold the data, `C` and `D`, respectively. The `TAB B` table consists of key fields `A`, `B`, `E`, and `G` with the data `F` and `H`. Now, the data from both these tables, `TAB A` and `TAB B`, is stored in the `Vardata` field of the table cluster.

Types of Table Fields

A table consists of many fields, and each field contains many elements. Table 5.5 lists the different elements that can be defined for a field:

Table 5.5: Different elements of table fields

Elements	Description
Field name	Represents the name given to a field, which can contain a maximum of 16 characters. The field name may be composed of digits, letters, and underscores; however, it must begin with a letter.
Key flag	Determines whether or not a field belongs to a key field.
Field type	Assigns a data type to a field.
Field length	Represents the number of characters that can be entered in a field.
Decimal places	Defines the number of digits permissible after the decimal point. This element is used only for numeric data types.
Short text	Explains the meaning of the corresponding field.

There are two types of table fields:

- Currency field
- Quantity field

Currency Field

The currency field is composed of two subfields: the currency field and the currency key field. The field that holds the numeric amount is known as the currency field, and the one that holds the currency type is known as the currency key field. For instance, if you need to store the value 1000 rupees, 1000 is stored in the currency field and rupees is stored in the currency key field. The data type of the currency field must be `CURR` and must be linked to a currency key field of the `CUKY` data type holding a currency key, such as `USD` (US dollars), `CAD` (Canadian dollars), or `ITL` (Italian lira). The currency key field acts as the reference field for the currency field.

Figure 5.26 shows the relationship between the currency field and the reference key field:

In Figure 5.26, there are two tables, T1 and T2. The T2 table acts as the reference table for the T1 table. Field 2 in the T1 table is of the type CURR (currency field) and Field 5 in the T2 table is of the type CUKY (Currency key table). The value entered in Field 2 shows the numeric amount, whereas Field 5 is used to show currency key type. In Figure 5.26, T1 - Field 2 = 4003.00 shows the amount, and T1 - Field 2 = INR shows the type of the currency.

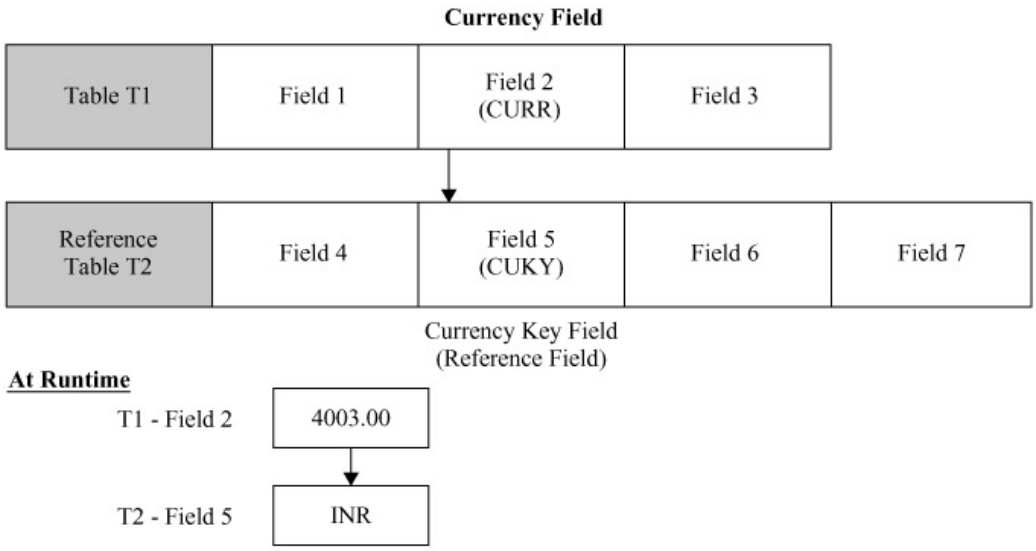


Figure 5.26: Relationship between the currency field and the reference key field

Quantity Field

The quantity field contains a numeric value for a measurement such as length, temperature, weight, or current. These numeric measurements are supported by the units of measure field, such as inches, degree Celsius, kilograms, or kilovolts. The quantity fields have QUAN as the data type and must be linked to a reference field that has a UNIT data type.

Technical Settings of a Table

You can define technical settings for a table while declaring it. Table 5.6 lists the description of the important data parameters related to the technical settings of a table:

Table 5.6: Description of the data parameters

Data Parameters	Description
Data Class	Defines the physical area (that is, tablespace) of the SAP database in which a table must be created.
Size category	Defines the size of a table based on the number of records that can be entered in the table.
Buffering permission	Defines whether or not a table can be buffered.
Buffering type	Defines the buffering type for a table. The buffering type can be full, single-record, or generic.
Logging	Defines whether the changes made to the entries of a table should be logged. If logging is switched on, any change made to a table record is recorded in a log table.

Now, we will discuss the following topics in detail with regard to the technical settings for a table:

- Data class
- Size category
- Table buffering

Data Class

The data class is used to specify the physical area where the tables are stored in a database. This physical area in the database is also known as tablespace or DBspace. Each data class corresponds to a physical area in which all the tables

assigned to this data class are stored. Table 5.7 lists the types of a data class:

Table 5.7: Types of data class

Data Class Type	Description
APPL0 (master data)	Stores master data. An example of master data is the data contained in an address file, such as the name, address, and telephone number. APPLO is seldom changed.
APPL1 (transaction data)	Stores the transaction data. An example of transaction data is the goods in a warehouse, which change after each purchase order.
APPL2 (organizational data)	Stores organizational data, such as a table with country codes. APPL2 is a data class, which represents the customized data that is defined when the SAP system is installed and seldom changed.

Figure 5.27 displays the different types of data that is stored in tables:

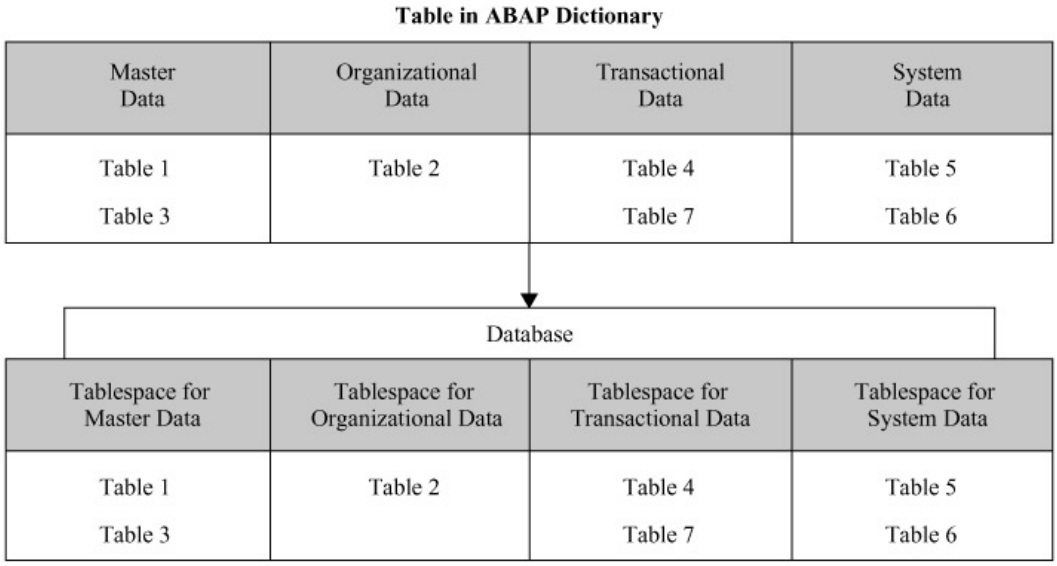


Figure 5.27: Types of data classes

Figure 5.27 shows that the tables related to a specific data are stored in their respective tablespaces. For instance, the tables for the organizational data are stored in the organizational data tablespace.

Size Category

You can assign a size for a database table by using the size category on the `ABAP Dictionary: Maintain Technical Settings` screen. The value in the size category technical setting can be a numeric value between 0 to 8. Each size category defines a possible number of records that you can maintain for a table. In addition, each size category is assigned with a certain fixed memory size in the database. The memory size allocated to the size category depends on the type of database system being used. Figure 5.28 shows the technical settings related to the size category:

Figure 5.28 shows three different tables, `TAB A`, `TAB B`, and `TAB C`, with different size categories 1, 3, and 4, respectively. Whenever a table is created, an initial space is automatically allotted to it.

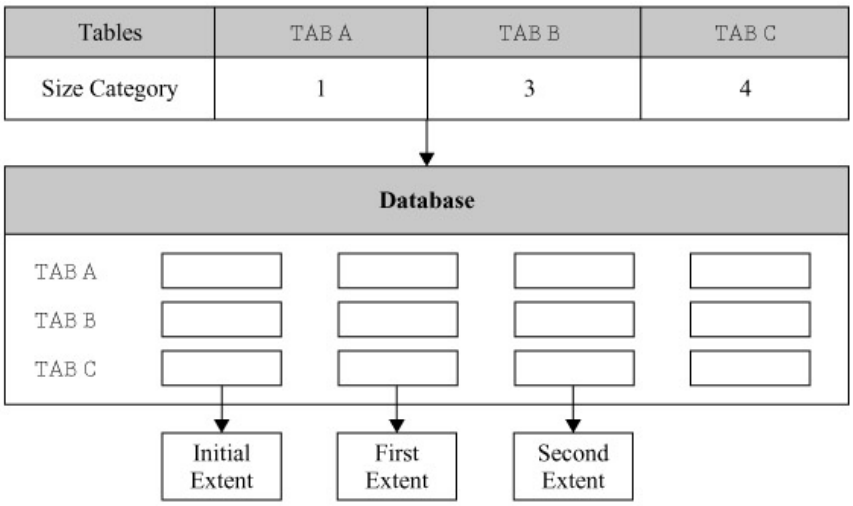


Figure 5.28: Defining size category for a table

Table Buffering

Sometimes, you need to access a particular set of records more frequently than others. Accessing records directly from a database table can be very time- consuming. To overcome this drawback, the frequently accessed records of a table can be placed in a buffer from which they can be accessed easily by a SAP system, improving performance.

You can define whether or not certain records need to be buffered from the `ABAP Dictionary: Maintain Technical Settings` screen. Note that only the transparent tables and the pooled tables can be buffered.

Table 5.8 lists various buffering permissions for a table:

Table 5.8: Buffering permissions for tables

Buffering	Description
Buffering Not Permitted	Specifies that either the application program always need the most recent data from the table or the table is changed frequently. Therefore, in this case, buffering is not required.
Buffering Permitted	Specifies that although buffering is permitted from the business and technical points of view, it is not activated.
But Not Activated	Table buffering is deactivated because it is not possible to find the type of field values that exist in a customer system. However, depending on your requirements, you can activate it on a customer system.
Buffering Activated	Specifies that the table is buffered. In this case, you must specify the buffering type.

Whenever a table record is accessed, it is loaded into the buffer of the application server. Table 5.9 lists three types of buffering:

Table 5.9: Types of buffering

Buffering Type	Description
Full buffering	Specifies that all the records of the table are loaded into the buffer when one record of the table is accessed.
Generic buffering	Specifies that when a record of the table is accessed, all the records having this record in the generic key fields (part of the table key that is left-justified, identified by specifying a number of key fields) are loaded into the buffer.
Single-record buffering	Specifies that only the accessed records of a table are loaded into the buffer.

Now, let's explain how to create a table in ABAP Dictionary.

Creating Tables

Tables in an SAP system can be created with the help of the ABAP Dictionary tool.

Perform the following broad-level steps to create a table:

1. Create and configure table fields
2. Create data elements
3. Define the technical settings of a table
4. Define the enhancement category

Creating and Configuring Table Fields

Perform the following steps to create and configure the table fields of a table:

1. Open the initial screen of ABAP Dictionary, select the `Database table` radio button, and enter a name for the table to be created. In our case, we have entered the name ZKOG_EMP, as shown in Figure 5.29:
2. Click the `Create` (Create) button.

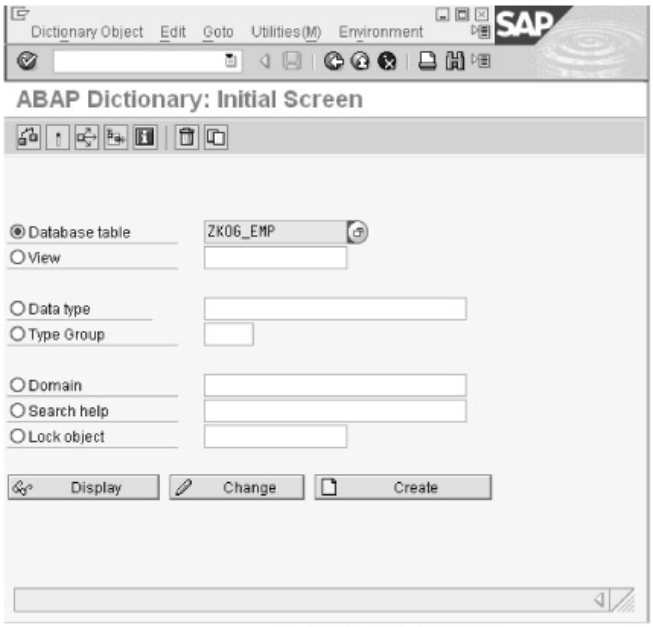


Figure 5.29: Creating a table

The `Dictionary: Maintain Table` screen appears, as shown in Figure 5.30:

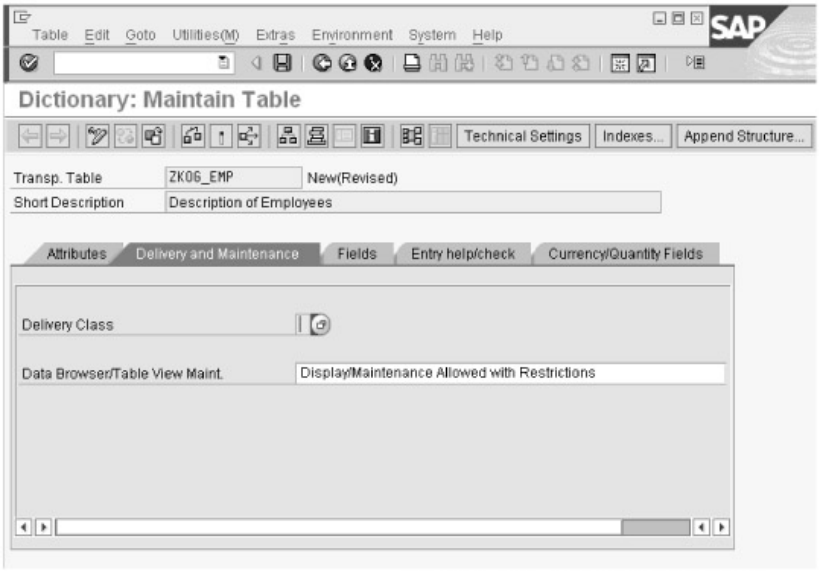


Figure 5.30: Maintenance screen for the table

Notice that in the `Dictionary: Maintain Table` screen, the `Delivery and Maintenance` tab is selected by default.

3. Enter an explanatory short text in the `Short Description` field. In Figure 5.30, the short text is entered as Description of Employees.

4. Click the `Search Help` icon beside the `Delivery Class` field.

The `Delivery class (1) 7 Entries found` dialog box appears, as shown in Figure 5.31:

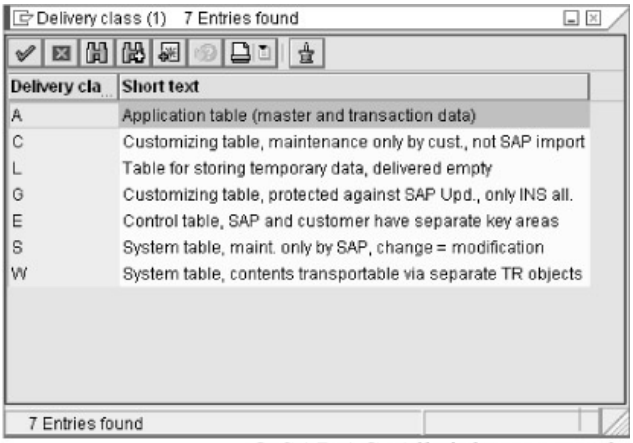


Figure 5.31: Showing the delivery class dialog box

5. Select an option from the displayed list and click the `Copy` icon (see Figure 5.31). In our case, we have selected A [Application table (master and transaction data)] option.

6. Now, select the `Display/Maintenance Allowed` option from the `Data Browser/Table View Maintenance drop-down` menu, as shown in Figure 5.32:

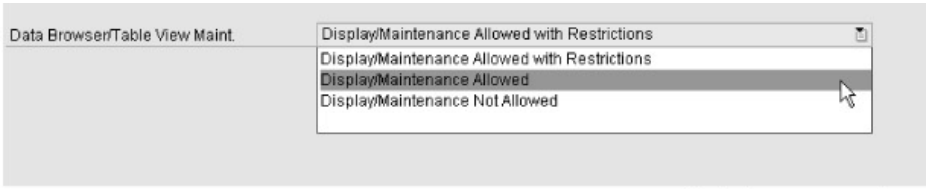


Figure 5.32: Entering value for table maintenance

The `Dictionary: Maintenance Table` screen appears, as shown in Figure 5.33:

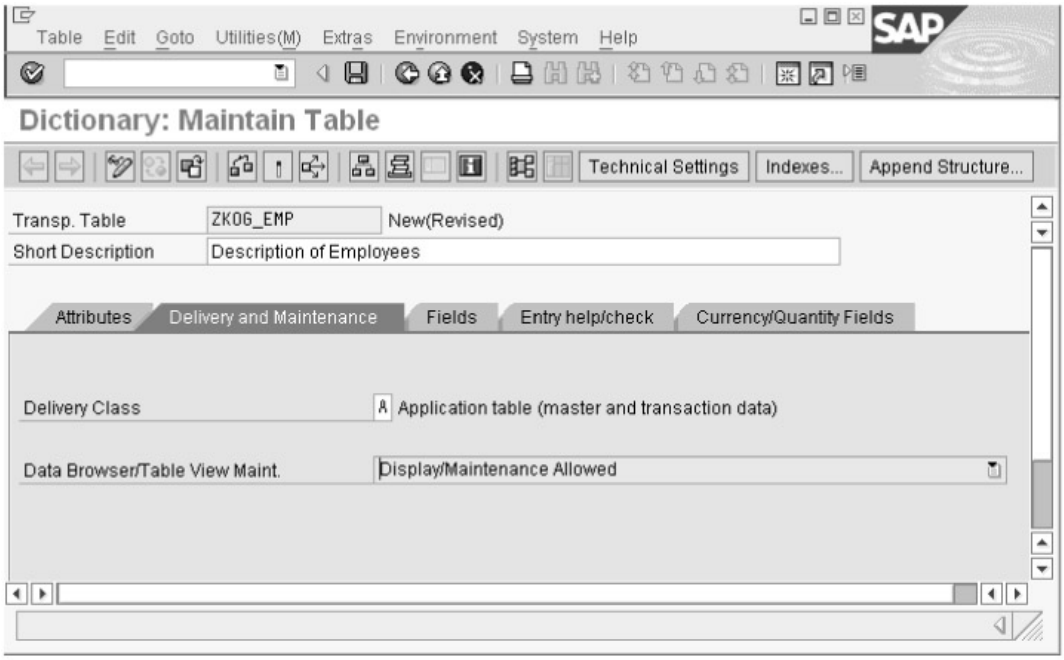


Figure 5.33: Showing the maintenance screen

7. Select the `Fields` (Fields) tab. The screen containing the options related to the `Fields` tab is shown in Figure 5.34.
8. Enter the names of table fields in the `Field` column. A field name may contain letters, digits, and underscores, but it must always begin with a letter and must not be longer than 16 characters. Select the `Key` column if you want the field to be a part of the table key. In our case, we have created two fields, `KEMPCODE` and `KEMPNAME`.

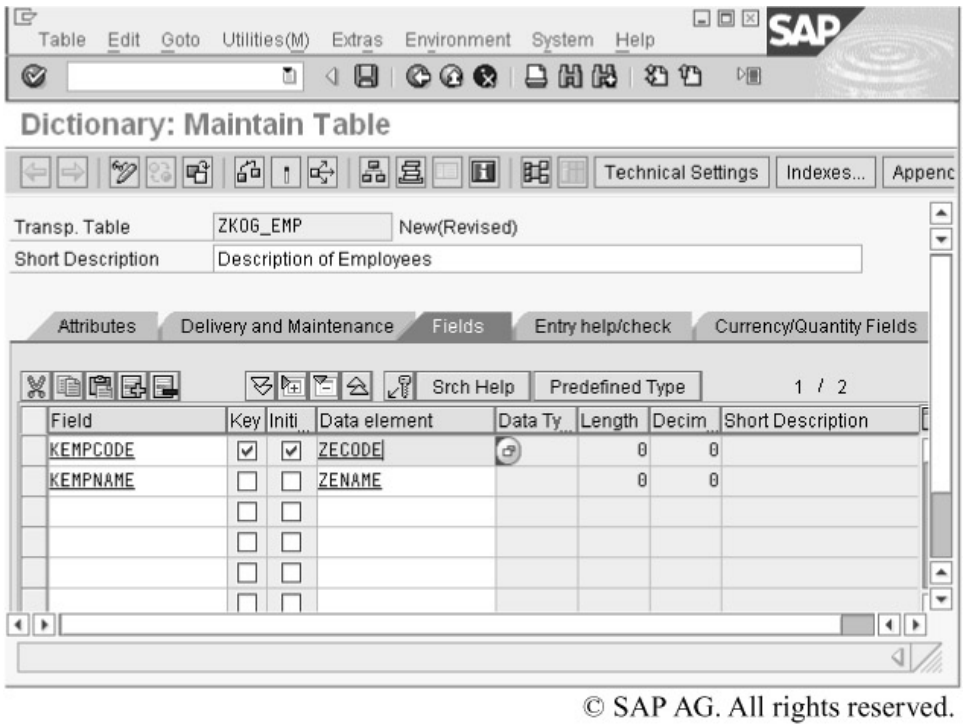


Figure 5.34: The data element for the KEMPCODE field

Creating Data Elements

T he fields that are to be created must also have data elements because they take the attributes, such as data type, length, decimal places, and short text, from the defined data element. If no suitable data type is already present, you need to create a data element.

Perform the following steps to create a data element:

1. Enter the names for the data elements in the `Data element` column. In our case, we have assigned the names, `ZECODE` and `ZENAME`, to the `KEMPCODE` and `KEMPNAME` fields, respectively, as shown in Figure 5.34:

2. Now, double-click `ZECODE` to create the data element (see Figure 5.34).

The `Exit Processing` dialog box appears (see Figure 5.35).

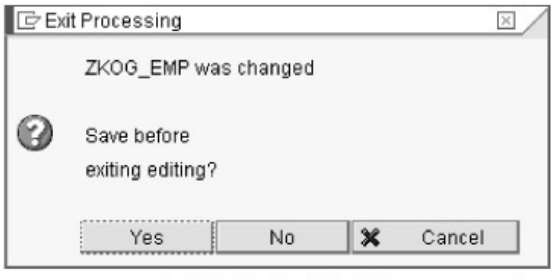


Figure 5.35: Saving the data element

3. Click the `Yes` button, as shown in Figure 5.35:

The `Information` dialog box appears, specifying that the table created should be client-specific.

4. Click the `Continue` icon, as shown in the Figure 5.36:

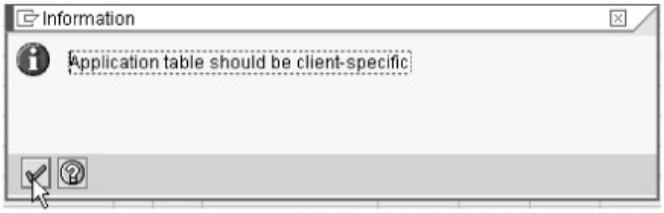


Figure 5.36: The information dialog box

The `Create Data Element dialog` box appears, as shown in Figure 5.37:

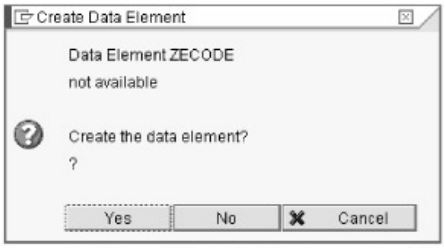


Figure 5.37: Creating data element

5. Click the `Yes` button on the `Create Data Element` dialog box.

The `Dictionary: Maintain Data Element` screen, also known as the maintenance screen for the data element, appears, as shown in Figure 5.38:

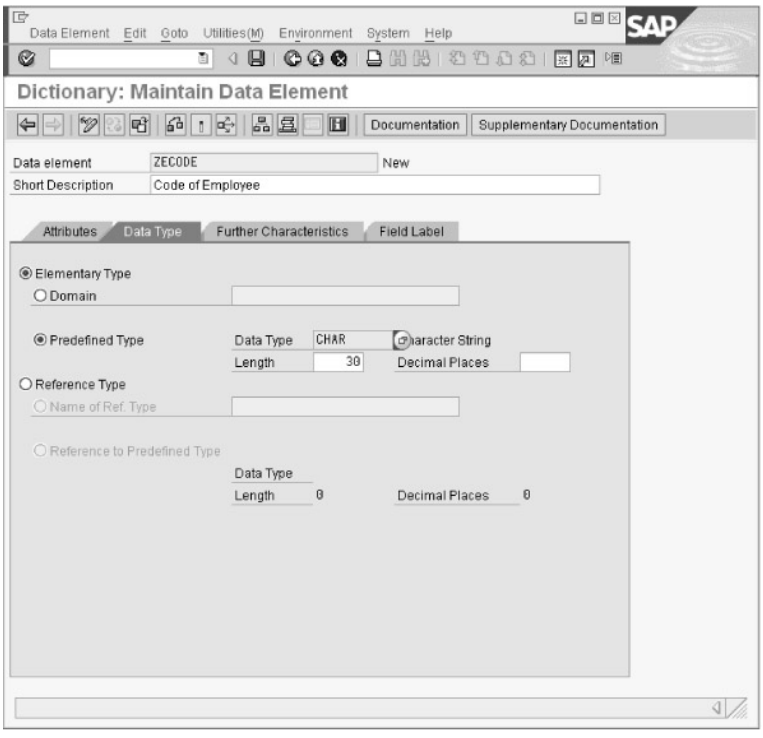


Figure 5.38: Maintaining the data element

6. Enter a short description, i.e., Code of Employee, in the `Short Description` field. If a field label for the data element is not defined, this short description acts as the description for the field when the field is displayed on the screen.

7. Select the data type for the data element as CHAR by clicking the `Search Help` icon. Enter the desired length in the `Length` field (see Figure 5.38).
8. Select the `Field Label` tab. The screen containing the options related to the field label appears, as shown in Figure 5.39:
9. Enter a name for the field label and specify the desired length for it. In Figure 5.39, we have entered Employee Code in the `Field Label` column and 20 in the `Length` column for the `Medium` field.
10. Click the `Save` icon to save the settings of the created data element.
11. Click the `Back` icon and then click the `Activate` icon. The data element is activated.

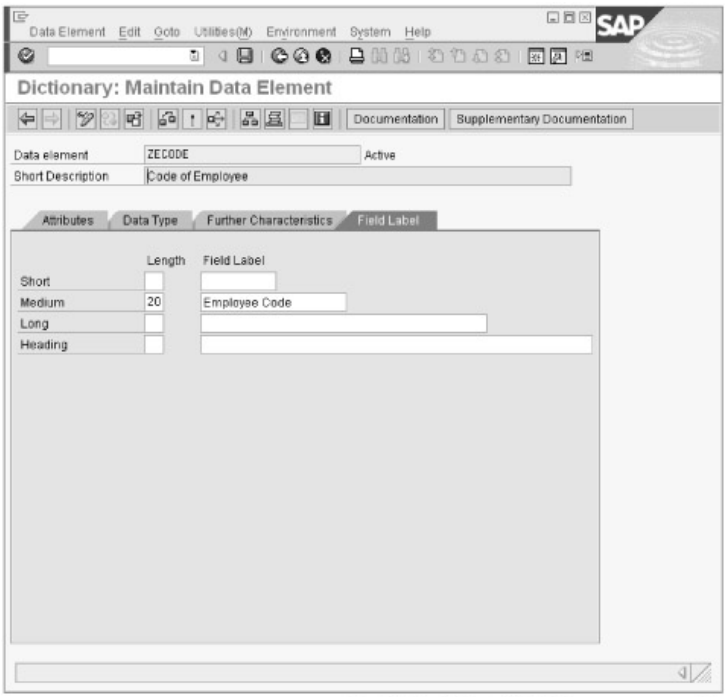


Figure 5.39: Field label

Now, let's define the attributes for the ZENAME data element of the KEMPNAME field.

12. Double-click the ZENAME data element (Figure 5.34).

The `Create Data Element` dialog box appears, as shown in Figure 5.40:

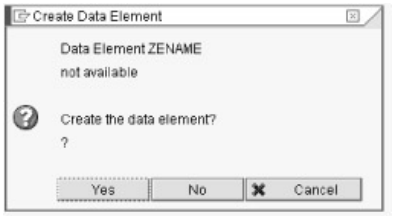


Figure 5.40: Creating data element

13. Click the `Yes` button (see Figure 5.40).

The `Dictionary: Maintain Data Element` screen appears, as shown in Figure 5.41:

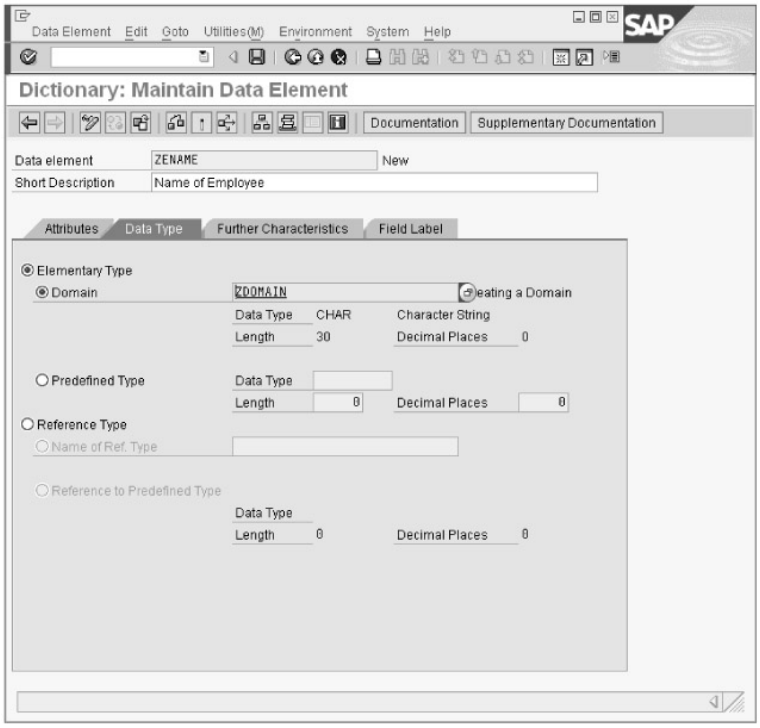


Figure 5.41: Creating data element

14. Select the `Domain` radio button under the `Elementary type` field and enter the name of the domain. In Figure 5.41, we have entered the name of the predefined domain, that is, ZDOMAIN, as shown in Figure 5.41. If the domain has not been defined earlier, double-click the name of the domain and define its attributes.

15. Select the `Field Label` tab. The maintenance screen containing the fields corresponding to the `Field Label` tab appears, as shown in Figure 5.42:

16. Click the `Save` icon to save the created data element in a package. In our case, the data element is saved in the `ZKOG_PCKG` package.

17. Click the `Back` icon and then click the `Activate` icon. The data element, `ZENAME`, is activated.

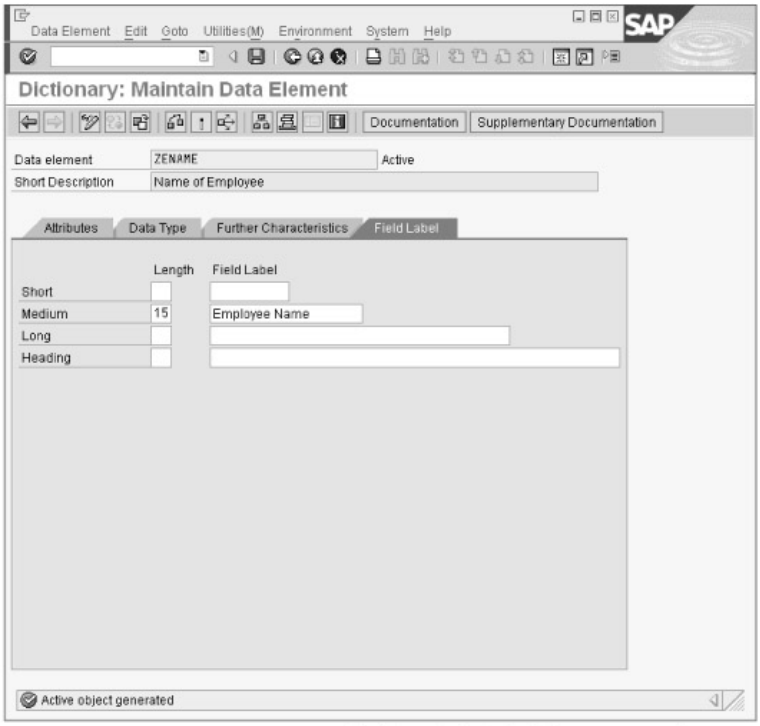


Figure 5.42: Defining a field label

Defining the Technical Settings of a Table

Now, let's learn how to define the technical settings related to the ZKOG_EMP table.

1. Click the `Technical Settings` (Technical Settings) button (previously shown in Figure 5.34).

The `Dictionary: Maintain Technical Settings` screen appears, as shown in Figure 5.43:

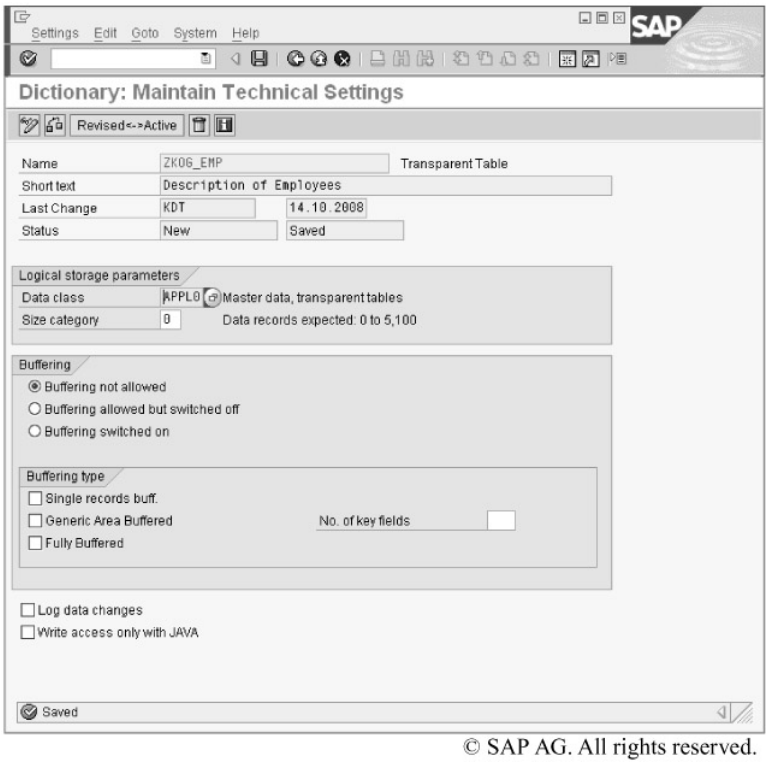


Figure 5.43: Maintaining technical settings

2. Set the `Data` class as `APPLO` and `Size category` as 0, as shown in Figure 5.43.
3. Click the `Save` icon to save the technical settings of the table.
4. Click the `Back` icon to go back to the `Dictionary: Maintain Table` screen (Figure 5.34).

Defining the Enhancement Category

Next, we define an enhancement category for the table to enhance the structures and tables created in ABAP Dictionary:

1. In the `Dictionary: Maintain Table` screen, select Extras > Enhancement Category to define the enhancement category, as shown in Figure 5.44:

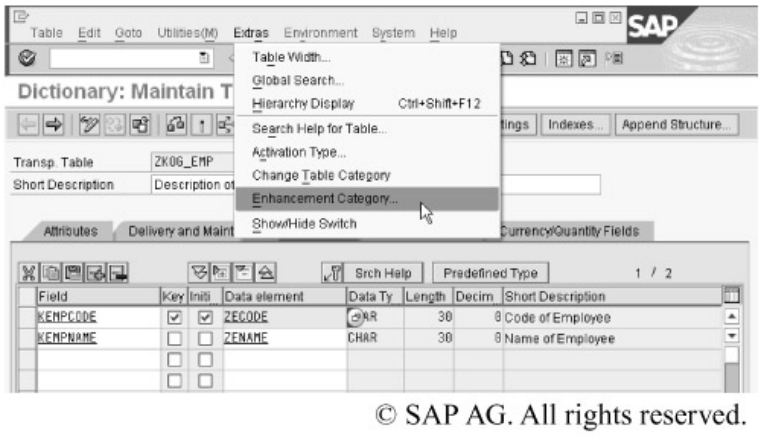


Figure 5.44: Defining an enhancement category

The `Information` dialog box appears, as shown in Figure 5.45:

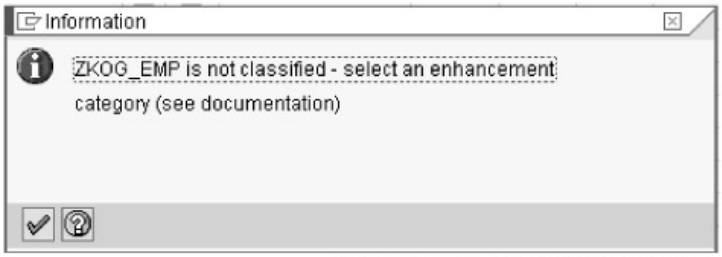


Figure 5.45: Displaying the information dialog box

2. Click the `Continue` icon.

The `Maintain Enhancement Category for ZKOG_EMP` dialog box appears, as shown in Figure 5.46:

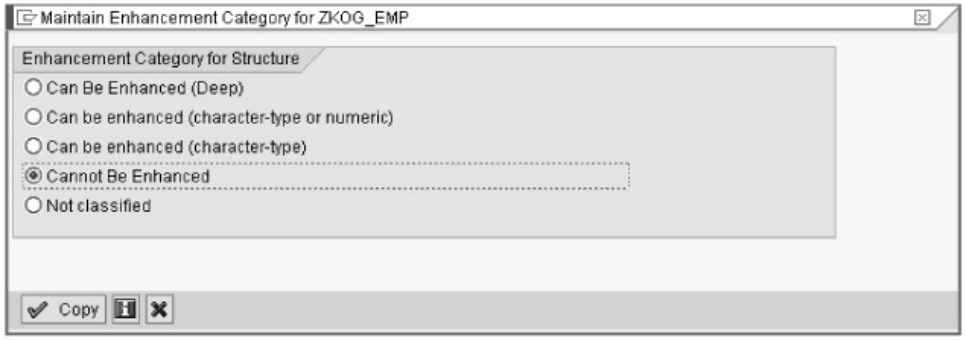


Figure 5.46: Selecting an enhancement category

3. Select the `Cannot be Enhanced` radio button for the enhancement category.

Table 5.10 lists the description of the different possible enhancement categories:

Table 5.10: Enhancement categories for a table

Enhancement Category	Description
Can Be Enhanced (Deep)	The table can be enhanced to contain components with any data type.
Can Be Enhanced (Character-Type or Numeric-Type)	Specifies that a table can contain only the character or numeric data types.
Can Be Enhanced (Character-Type)	Specifies that a table can contain only the character data types.
Cannot Be Enhanced	Specifies that a table cannot be enhanced.
Not Classified	Specifies that the table does not have any enhancement category.

4. Click the `Copy` (Copy) button (see Figure 5.46).

5. Click the `Save` icon, the `Check` icon, and then the `Activate` icon. The maintenance screen for the table appears, displaying the status as Active, as shown in Figure 5.47:

6. Select Utilities > Table Contents > Create Entries, as shown in Figure 5.48:

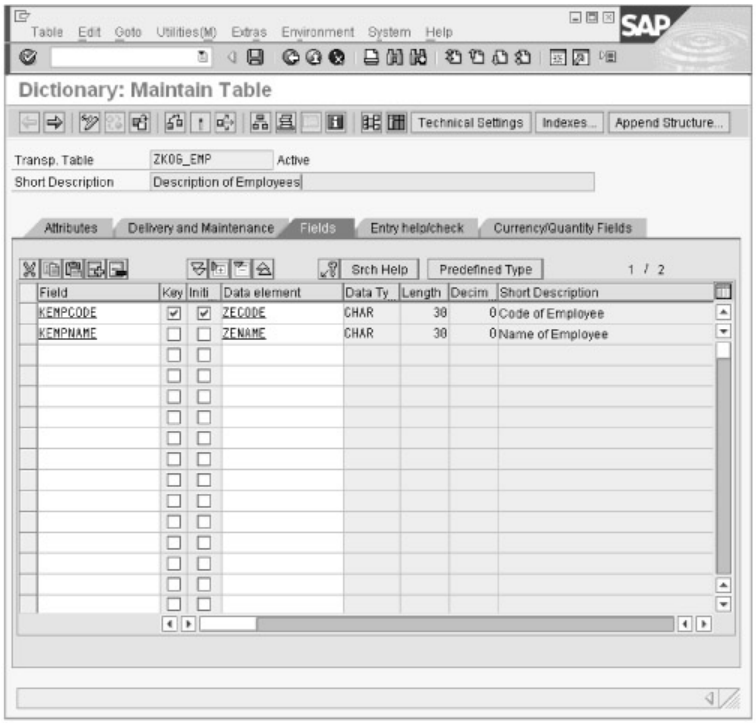


Figure 5.47: Active version of the table

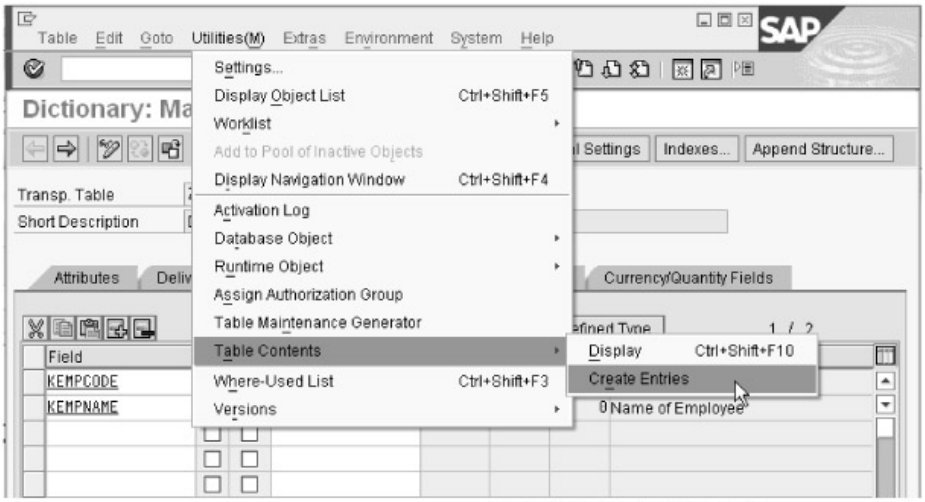


Figure 5.48: Creating entries in table

The `Table ZKOG_EMP Insert` screen appears, as shown in Figure 5.49:

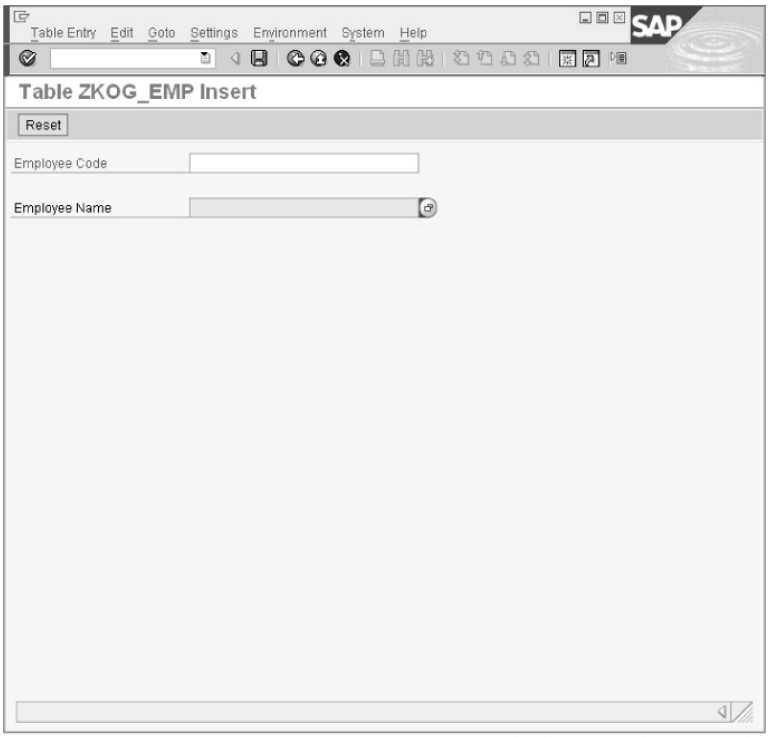


Figure 5.49: Entering values into a table

7. Click the `Search Help` icon beside the `Employee Name` field. The screen containing the predefined value range for the `ZDOMAIN` domain appears, as shown in Figure 5.50:

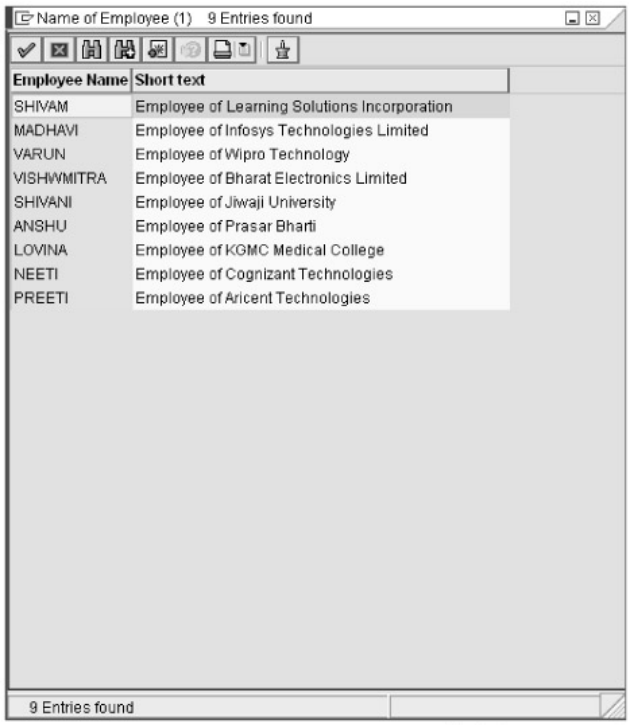


Figure 5.50: Valid set of values

Note You cannot enter any values other than those defined in the value range for the `ZDOMAIN` domain. The `KEMPNAME` field is assigned with the `ZENAME` data element, which is further assigned with an elementary data type called `ZDOMAIN`.

8. Select `SHIVAM` and click the `Copy` icon.

9. Assign an employee code, say, `KSI-0079`, to the selected name, that is, `SHIVAM`, and click the `Save` icon.

Figure 5.51 shows the entry of the employee code for the employee name `SHIVAM`:

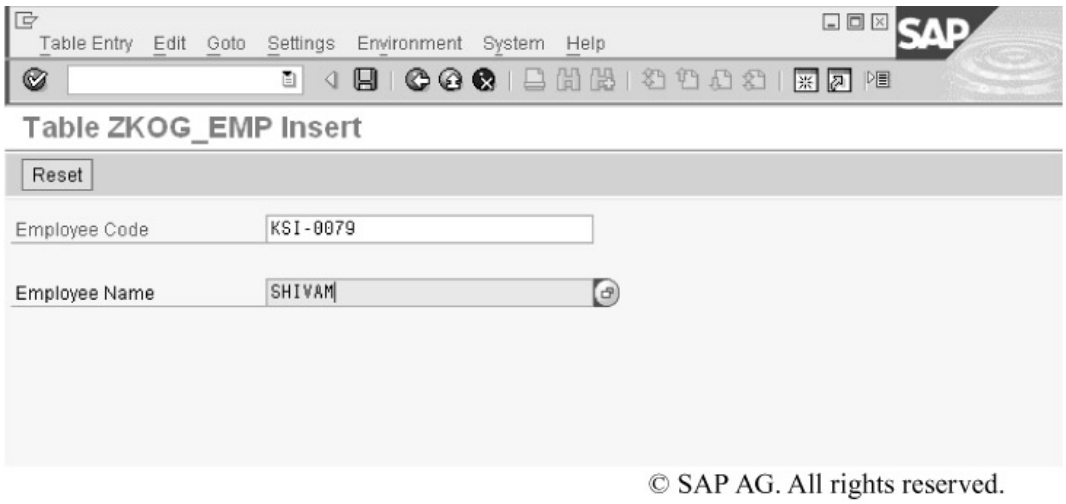


Figure 5.51: First entry of the table

10. Click the `Reset` button (see Figure 5.51).

Similarly, maintain the employee codes for the remaining employees.

11. Click the `Back` icon to go back to the `Dictionary: Maintain Table` screen.

12. Select `Utilities (M) > Table Contents > Display` to display all the entries, as shown in Figure 5.52:

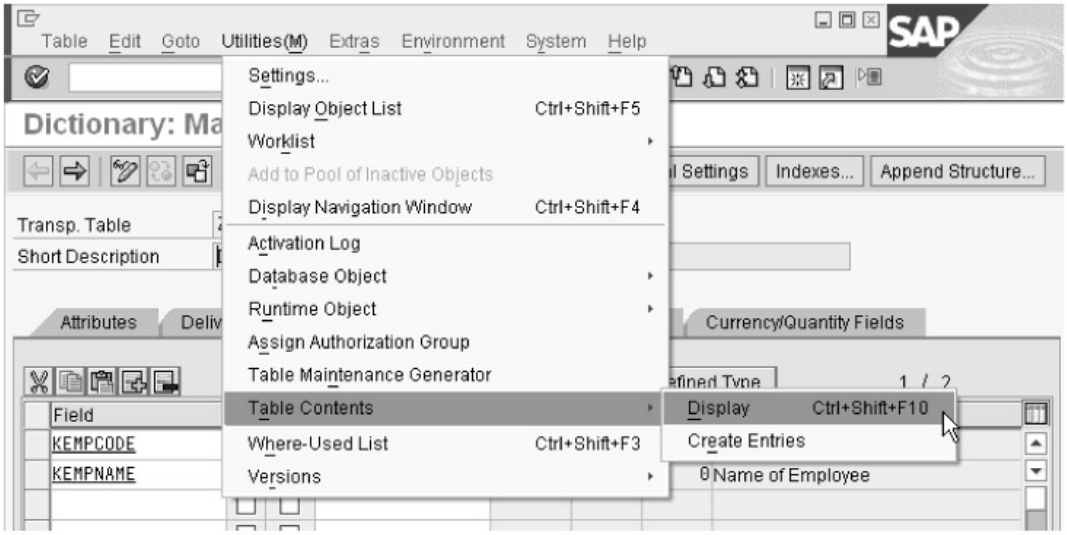


Figure 5.52: Displaying the contents of the table

The selection screen for `Data Browser: Table ZKOG_EMP` appears, as shown in Figure 5.53:

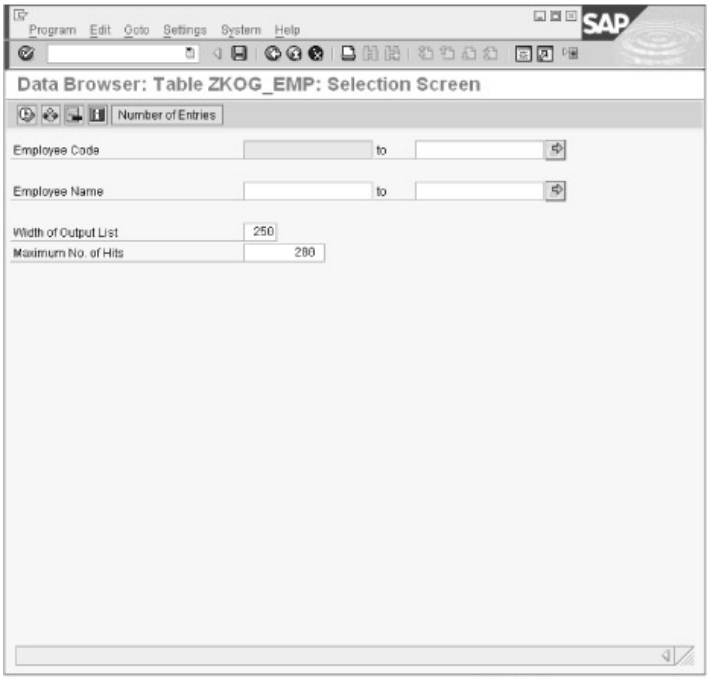


Figure 5.53: Showing the selection screen for the table

13. Click the `Execute` icon to display all the entries made in the table, as shown in Figure 5.53.

The `Data Browser: Table ZKOG_EMP Select Entries` screen appears, as shown in Figure 5.54:

Figure 5.54: Showing the contents maintained in the table

Figure 5.54 displays all the records of the ZKOG_EMP table.

Relating Tables by Using Foreign Keys

Foreign keys are used to establish relationships between various tables in ABAP Dictionary. You can create value checks for input fields with the help of foreign keys. A foreign key links two tables by assigning the foreign key fields of one table to the primary key fields of another table. Figure 5.55 shows the process of field assignment by using a foreign key:

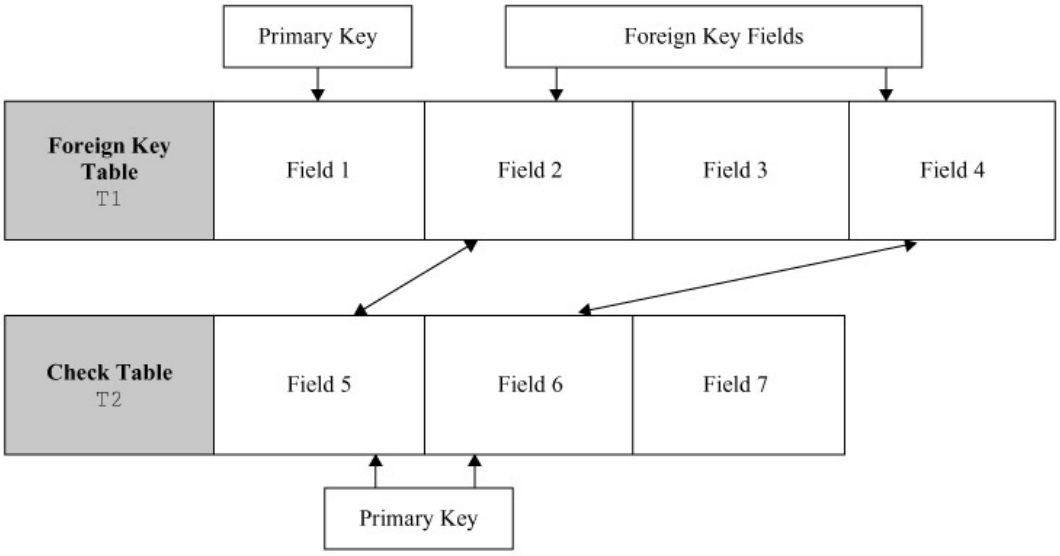


Figure 5.55: Field assignment by using a foreign key

In Figure 5.55, the T1 table is the foreign key table, whereas the T2 table refers to a check table, also known as a dependent table. The pair of fields for the two tables must have the same data type and length so that a relationship can be created between them. You can notice that fields of the foreign key table relate to the primary key fields of the check table. In our case, Field 2 and Field 4 of the foreign key table are known as foreign key fields.

A field from the foreign key table is marked as the check field. This check field is required to perform the value checks for the data entered in the input fields, which means that whenever an entry is made in the check field of the foreign key table, a check is always performed to validate whether the check table contains a record with the specified key. If the entered

value matches with the value specified in the check table, the SAP system accepts the value; otherwise, the system rejects the value. Figure 5.56 shows how the check is performed between the foreign key table (T1) and the check table (T2):

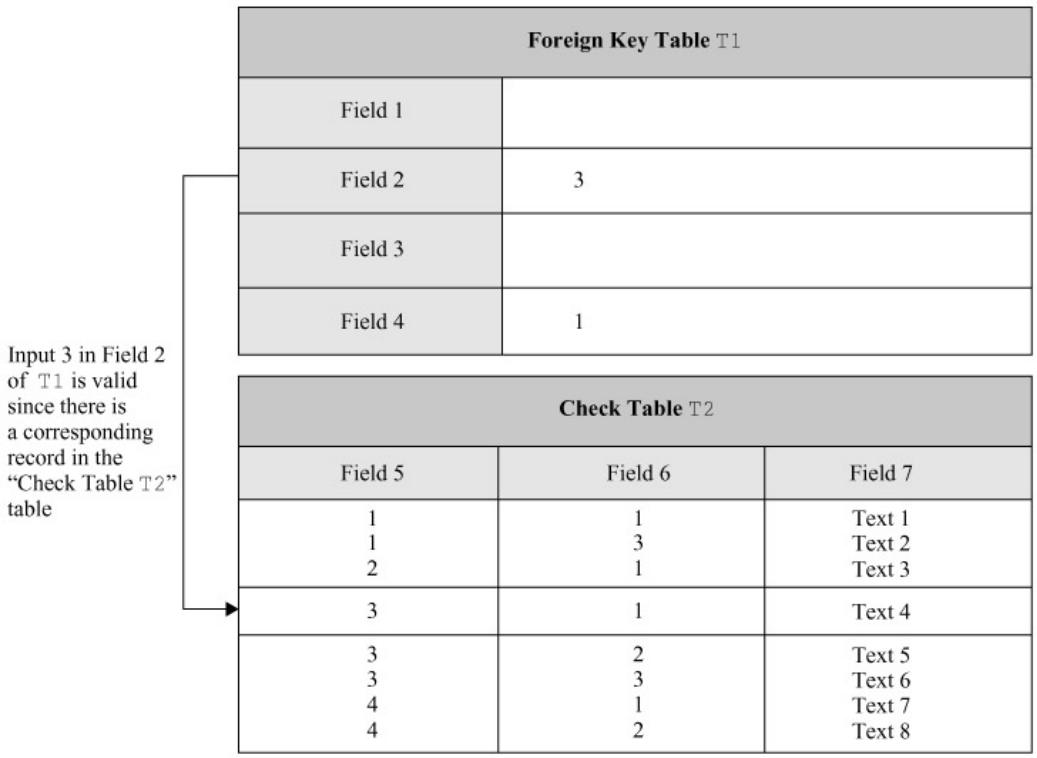


Figure 5.56: The check field and the value check

In Figure 5.56, Field 2 = 3 and Field 4 = 1 are accepted because the T2 check table contains a similar record in the key fields Field 5 = 3 and Field 6 = 1. The SAP system rejects other values apart from those specified in the check table.

Now, let's discuss the following topics in the context of foreign keys:

- Triggering foreign keys
- Exploring compound foreign keys
- Describing generic and constant foreign keys
- Exploring the cardinality of a foreign key
- Describing the foreign key field type

Triggering Foreign Keys

A foreign key is triggered as soon as you enter a value in a foreign key table. A SELECT statement is used to initiate this triggering. The SELECT statement checks for the matching rows in the check table; if the specified rows are not found in the check table, a standard message indicating that the value entered is invalid is displayed.

> **Note** The foreign key field and the check table field must have the same domain name. This ensures that the fields being compared are compatible in data type and length.

Apart from the SELECT statement, a foreign key can also be triggered with the help of a function key, button, or a menu item.

Exploring Compound Foreign Keys

A compound foreign key is a foreign key consisting of two or more fields. In this case, a check is made to compare the two fields in the foreign key table against the two fields in the check table. An entry made in the foreign key fields is valid only when a corresponding value for that entry exists in the check table fields.

The field on which the compound foreign key is defined is known as the check field. A check field must not be blank. If a check field is blank, the compound foreign key is not triggered.

Describing Generic and Constant Foreign Keys

As already discussed, whenever a foreign key is created, all the primary key fields of the check table must be included in the foreign key relationship. However, in some cases, you may not want to perform a check against all these fields. In such situations, you can use generic foreign keys. In a generic foreign key, some fields can be excluded from being assigned to the key fields of the check table. The check can be performed only against the remaining key fields. Apart from this, you can assign a constant value to a key field of the check table. This allows you to check the entries against a constant value in the key field of the check table.

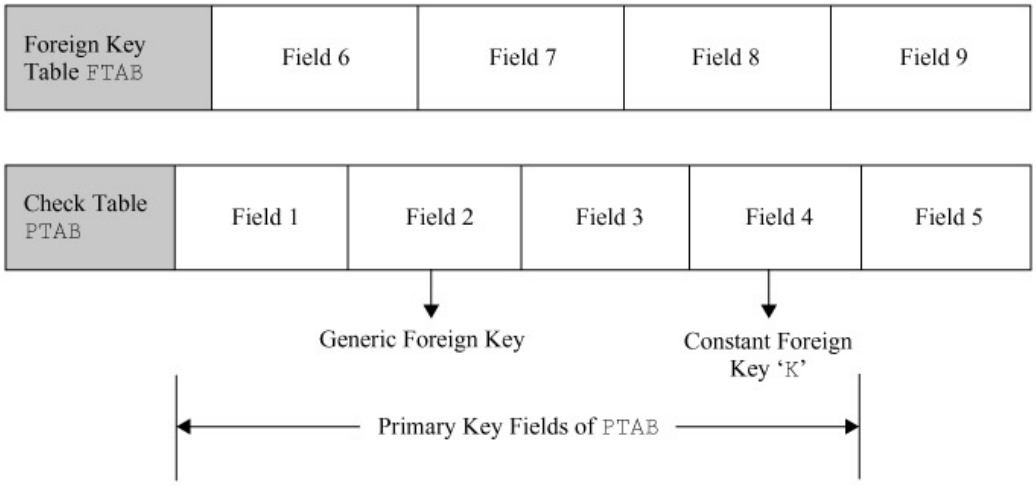


Figure 5.57: Generic and constant foreign keys

Figure 5.57 shows the Generic and Constant foreign keys:

As shown in Figure 5.57, the Generic and Constant foreign keys are excluded from the foreign key relationship. The two tables used are FTAB (foreign key table) and PTAB (check table). In Figure 5.57, a foreign key relationship is established between Field 6 and Field 1 and between Field 8 and Field 3. Apart from this, Field 2 acts as the Generic key and Field 4 acts as the Constant key; therefore, both these fields are excluded from the foreign key relationship.

Let's look at how to perform a check of valid values entered by a user in different fields. Figure 5.58 shows the process of performing the check:

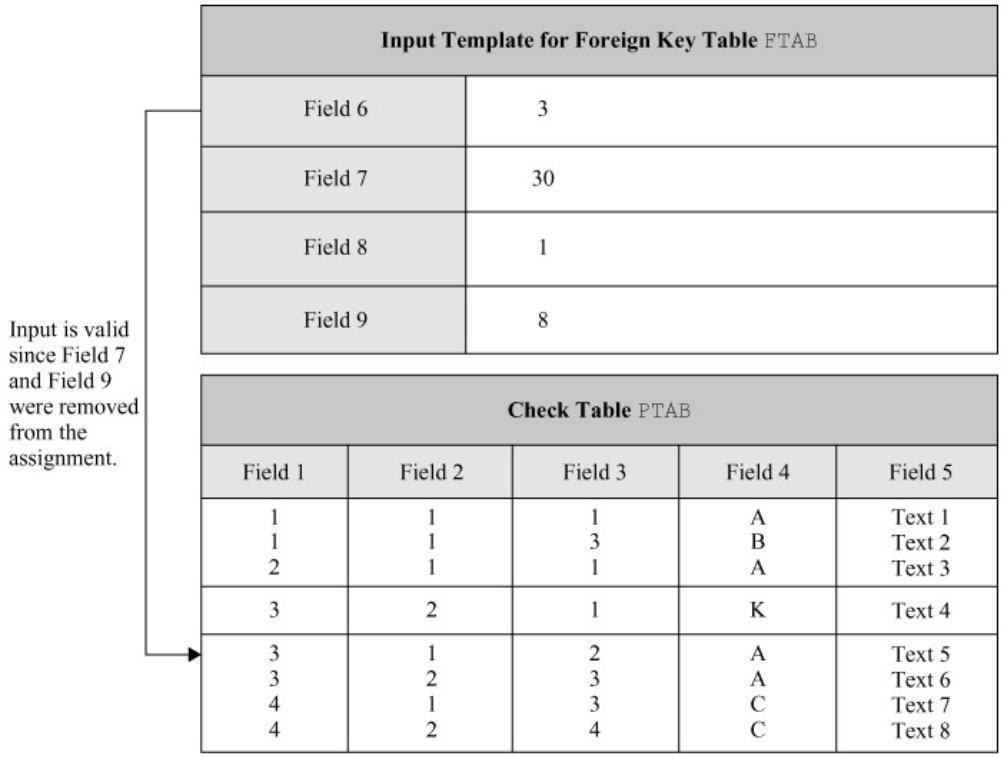


Figure 5.58: Performing a check on the values in the fields of the check table

Figure 5.58 shows that the values entered for Field 7 and Field 9 (see Figure 5.57) are meaningless when checked against the check table. If an entry, say 3, is made in Field 6 and another entry, say 1, is made in Field 8, the SAP system accepts these entries. Notice that Field 2 and Field 4 are not included in the check.

Exploring the Cardinality of a Foreign Key

Cardinality describes the foreign key relationship with regard to the number of possible dependent records (records of the foreign key table) or referenced records (records of the check table). Cardinality is expressed as the *n*:*m* relationship. Table 5.11 lists the possible values for the *n* variable:

Table 5.11: Possible values of the *n* variable

Value	Description
1	Specifies that a check table has only one record for each record of the foreign key table.
C	Specifies that the foreign key table may contain records that do not correspond to any record of the check table.

Table 5.12 lists the description of the possible values of the *m* variable:

Table 5.12: Possible values of the *m* variable

Value	Description
1	Specifies that only one dependent record exists for each record in the check table.
C	Specifies that at the most, only one dependent record exists for each record of the check table.
N	Specifies that at least one dependent record exists for each record of the check table.
CN	Specifies that any number of dependent records may exist for each record of the check table.

Describing the Foreign Key Field Type

The foreign key field type is used to describe the meaning of the foreign key field in the foreign key table. Table 5.13 describes different types of foreign key fields:

Table 5.13: Types of foreign key fields

Foreign Key Field	Description

Type	
No key fields/candidates	Specifies that the foreign key fields are not part of the primary key fields of the foreign table. They also do not uniquely identify a record of the foreign key table (key candidates).
Key fields/candidates	Specifies that the foreign key fields are either a part of the primary key fields of the foreign key table or uniquely identify a record of the foreign key table (key candidates).
Key fields of a text table	Specifies that the foreign key table acts as the text table for the check table.

Exploring Views

ABAP Dictionary provides a repository object, known as a view, to facilitate viewing data stored in multiple database tables. A view acts similarly to a virtual table that is, a table that does not have any physical existence. A view is created by combining the data of one or more tables (called base tables) containing information about an application object. Before creating a view, you need to define a structure for the tables and fields that are to be contained in the view. Note that if you change the data in the base tables of a view, the view itself also reflects the change. Using views, you can represent a subset of the data contained in a table or you can join multiple tables into a single virtual table. Moreover, views take up very little space in a database. This is because the database contains only the definition of the view, not a copy of the data displayed by the view.

Figure 5.59 shows that the viewed data is distributed among multiple tables:

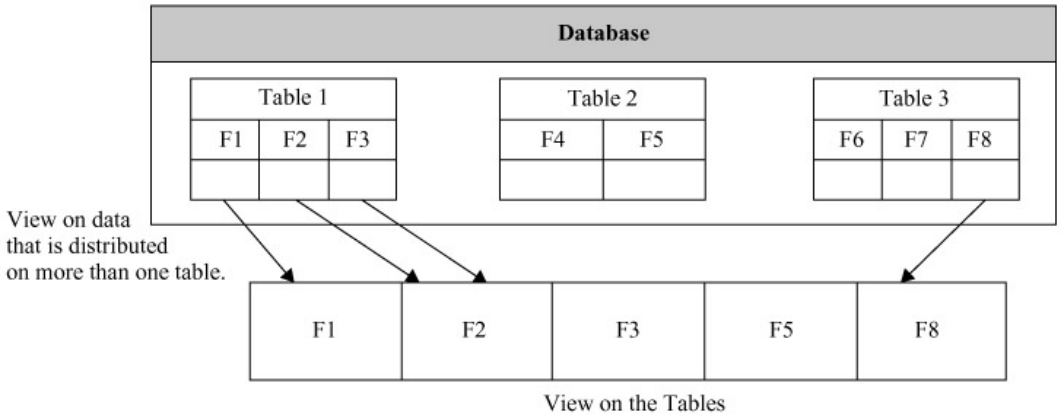


Figure 5.59: Distribution of a view

Figure 5.59 shows that the selected data from Tables 1, 2, and 3 are displayed with the help of the view object. To extract data from a view, you can either mask out one or more fields from the table (projection) or include only certain entries of a table in the view (selection). In more complicated views, data is extracted from multiple tables, where each individual table is linked to another table with a relational join operation.

Now, let's explain the following topics in the context of views:

- Relating database tables using relational operators
- Creating different types of views
- Deleting views

Relating Database Tables Using Relational Operators

Join, projection, and selection operators are the relational operators provided by SAP. These operators are used to establish relationships between different database tables. Suppose that there are two tables, TAB A and TAB B. TAB A has two entries and TAB B has four entries. Figure 5.60 shows the cross-product of both these tables:

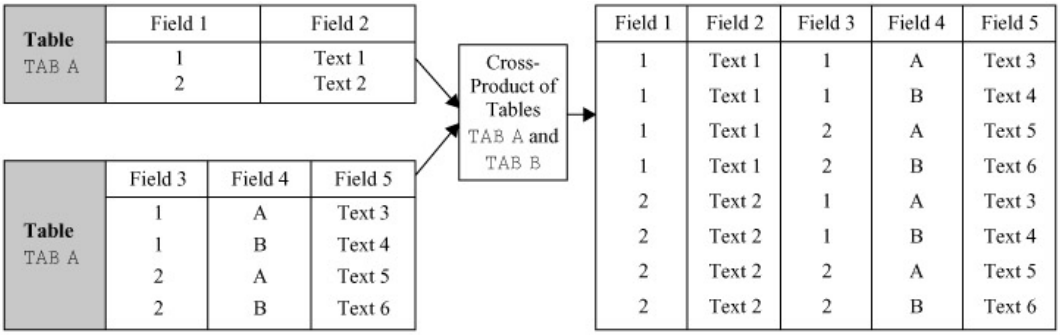


Figure 5.60: The tables and the table entries

Figure 5.60 shows the output of cross-multiplication of the records of the TAB A and TAB B tables. Now, we will see how this output varies when we apply different relational operators on the tables.

Now, let's learn how to implement the following relational operators on two or more tables:

- Join condition
- Projection condition
- Selection condition

Join Condition

The join condition is applied on tables to retrieve only the desired records in their cross-product. Figure 5.61 shows the result of applying the join condition between the TAB A and TAB B tables:

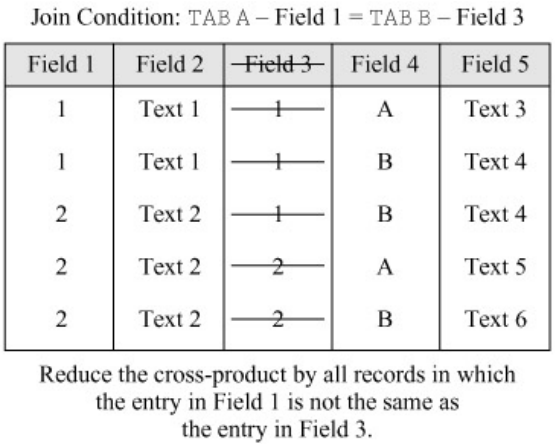


Figure 5.61: Join condition

In Figure 5.61, you can see that a join condition, TAB A - Field 1 = TAB B - Field 3, is applied to the TAB A and TAB B tables. The join condition removes all the records for which the entry in Field 1 is not the same as that in Field 3 (scored through in the figure).

Projection Condition

You can select the fields that you want to display in a view object. The condition to select only a specific set of fields is known as projection. Figure 5.62 shows how the projection condition can be applied to the output obtained after applying the join condition:

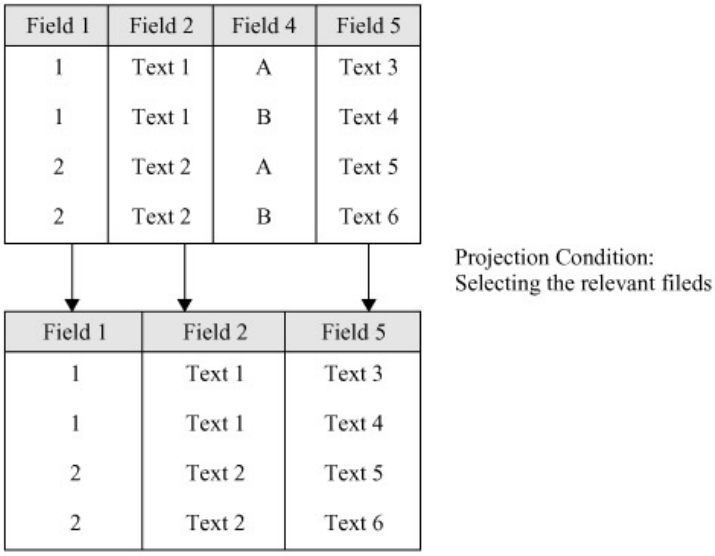


Figure 5.62: Projection relational operator

Selection Condition

A selection condition is used to filter the desired data from database tables by applying certain conditions. The selection condition uses logical operators, such as `AND` and `OR`, to filter the data. Figure 5.63 shows how to apply the selection condition on the output obtained after applying the join condition on the database tables:

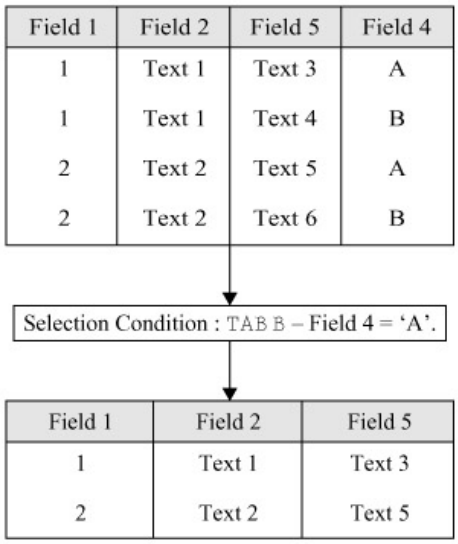


Figure 5.63: Selection condition

In Figure 5.63, notice that a selection condition is defined as `TAB B` - Field 4 = 'A'. This selection condition shows that only the records corresponding to Field 4 = 'A' are selected and displayed.

Creating Different Types of Views

As learned earlier, views are virtual tables that do not store data physically but store the data in database tables. In this section, you learn how to create the following types of views:

- Database views
- Projection views
- Maintenance views
- Help views

Let's explain these views in detail, one by one.

Database Views

As you know, data related to an application object is distributed among multiple tables by using database views. Database views use the inner join condition to join the data of different tables. You can create a database view when you want to view logically connected data from multiple tables simultaneously. You can also view the data of the database view with an ABAP program. The ABAP program retrieves the data from the database view with a database interface. Figure 5.64 shows the example of a database view:

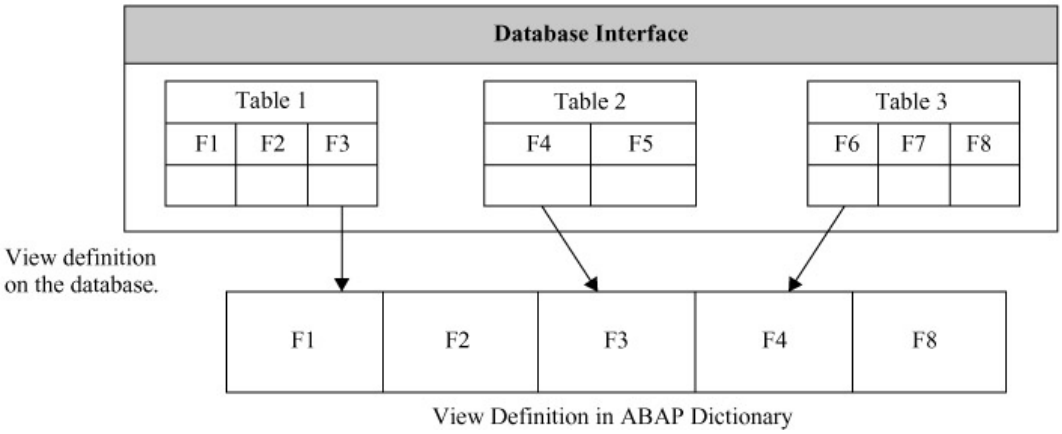


Figure 5.64: Database view

Figure 5.64 shows that the data is read from different fields distributed among multiple tables.

A database view can include an entire table. Figure 5.65 displays all the fields of a table that are included in the view:

In Figure 5.65, all the fields of the `TAB B` table are included in the database view. Apart from this, the fields of the `TAB A` and `TAB C` tables are also used in the view. The database view is updated automatically if any change, such as addition or deletion, is made in the existing fields.

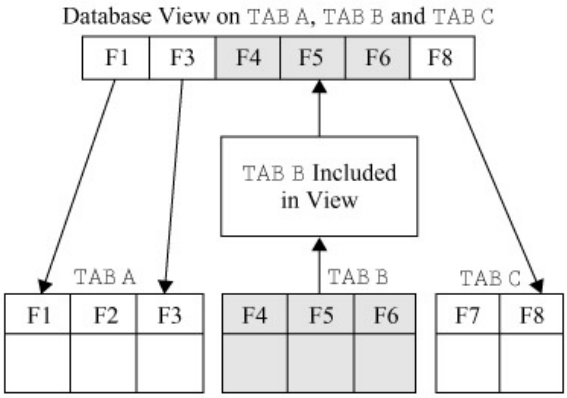


Figure 5.65: Table inclusion

Perform the following steps to create a database view:

1. Select the `View` radio button from the initial screen of ABAP Dictionary. Enter the name of the view to be created and click the `Create` button. In our case, we enter the name as `ZDVIEW`, as shown in Figure 5.66:

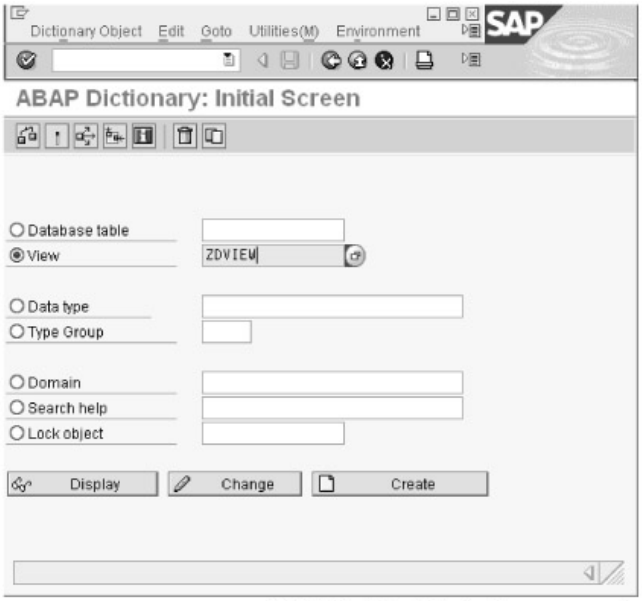


Figure 5.66: Entering the name of the view

The `Choose View Type` dialog box appears, as shown in Figure 5.67.

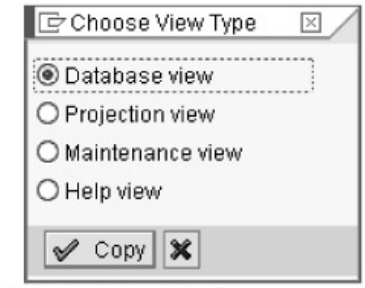


Figure 5.67: Selecting a view

2. Select a view in the `Choose View Type` dialog box. In our case, we have selected `Database view`, as shown in Figure 5.67:
3. Click the `Copy` (Copy) button. The `Dictionary: Maintain View` screen for the selected view type appears, as shown in Figure 5.68.
4. Enter a short description of the view in the `Short Description` field, such as Creating Database View, as shown in Figure 5.68:
5. Enter the names of the tables, to be included in the view, in the `Tables` field of the `Table/Join Conditions` tab page.

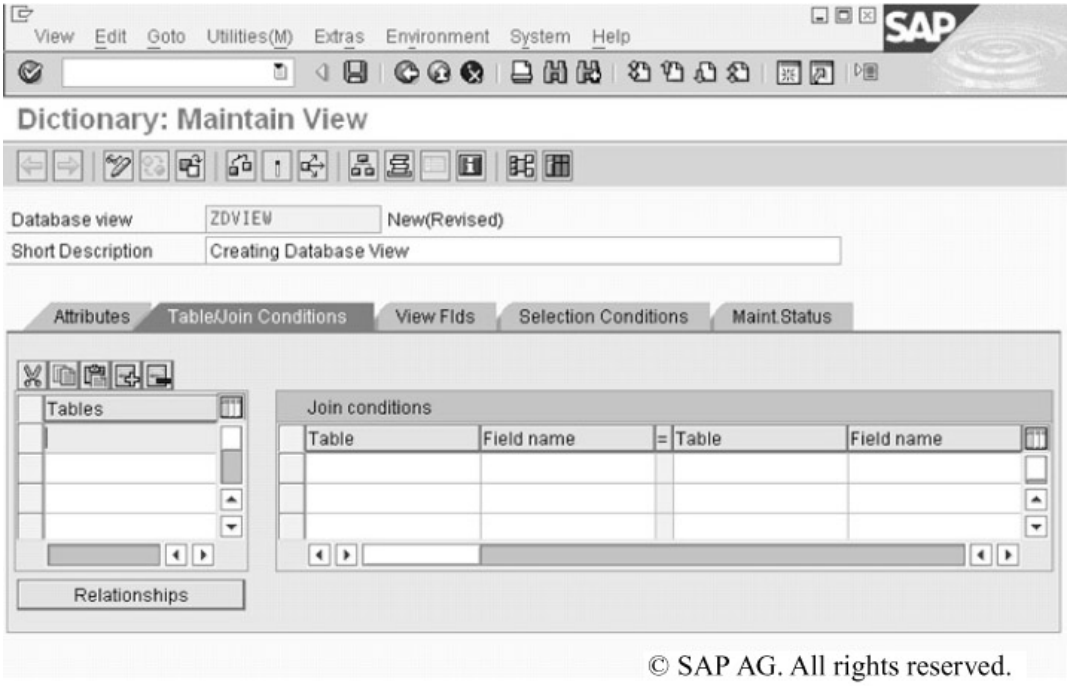


Figure 5.68: Showing the maintenance screen for the selected view

In this case, we use the ZSTUDENT_DATA and ZSUBJECT_DATA tables, which are user-defined database tables. These two tables are linked with a foreign key. The ZSTUDENT_DATA table contains nine fields: STUDENTID, STUDENTNAME, SUBJECTCODE, MARKS, ADDRESS, CITY, CLASS, MONTHLYFEES, and YEARLYFEES. The ZSUBJECT_DATA table contains two fields, SUBJECTCODE and SUBJECTNAME. The ZSTUDENT_DATA table acts as the foreign key table, and the SUBJECTCODE field acts as the foreign key field. The ZSUBJECT_DATA table acts as the check table and the SUBJECTCODE field of this table is the primary key of the table used to define the foreign key relationship between the two tables.

Note You can include only transparent tables in a database view.

6. Enter the name of the table containing the foreign key, that is, ZSTUDENT_DATA, in the `Tables` column and click the `Relationships` (Relationships) button to define a join condition, as shown in Figure 5.69:

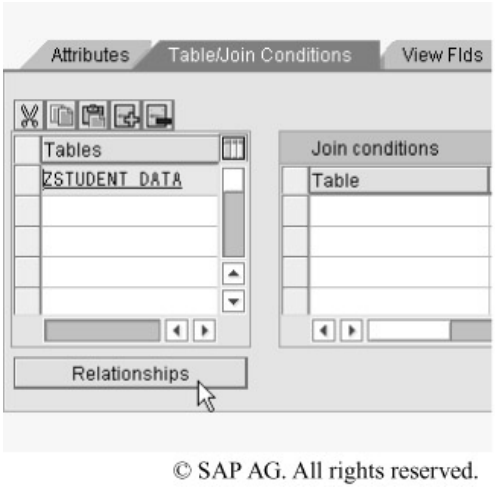


Figure 5.69: An entry in the tables column

The `Relationships of Table` ZSTUDENT_DATA dialog box appears (see Figure 5.70).

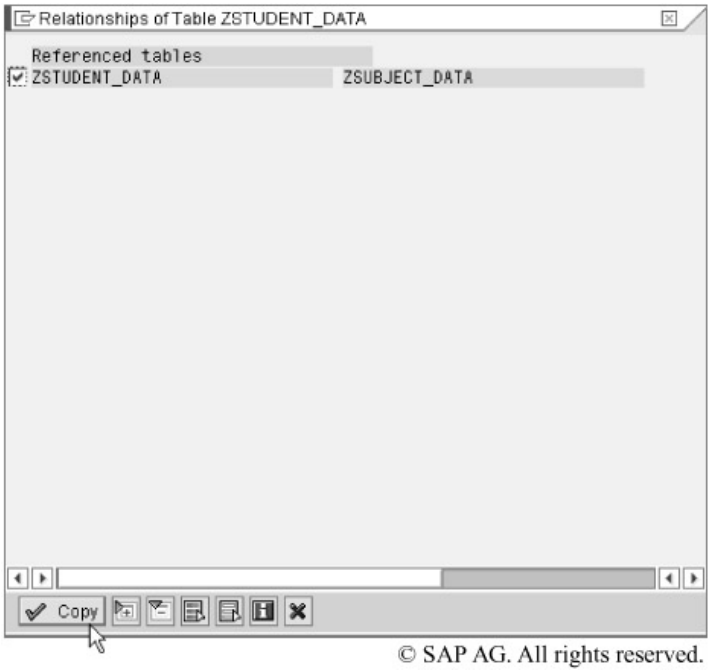


Figure 5.70: Copying the join conditions

7. In the `Relationships of Table ZSTUDENT_DATA` dialog box, select the given check box that is displayed in respect to the `ZSTUDENT_DATA` table. The `ZSTUDENT_DATA` table has a field, which acts as a foreign key field; and the `ZSUBJECT_DATA` table has a field, which acts as a primary key field. Click the `Copy` (Copy) button, as shown in Figure 5.70:

The fields of tables that are related in a join condition are displayed, as shown in Figure 5.71:

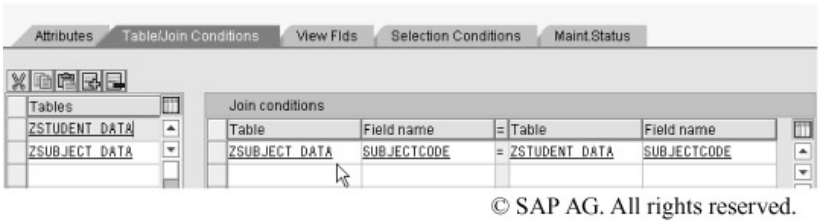


Figure 5.71: Join conditions

8. Now, select the `View Flds` (View Flds) tab to select the fields that you want to copy to the view.

The screen showing the fields related to the `View Flds` tab appears.

9. Click the `Table fields` (Table fields) button, as shown in Figure 5.72:

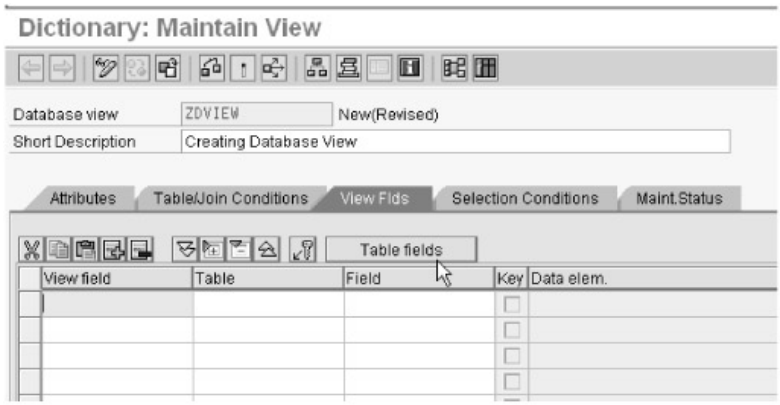


Figure 5.72: Showing the view flds tab page

The `Base Tables` dialog box, which displays all the tables contained in the view, appears, as shown in Figure 5.73.

Figure 5.73: Showing all the tables present in the view

10. Select all the tables, one at a time, and click the `Choose` (Choose) button (see Figure 5.73):

The `Field Selection from Table ZSTUDENT_DATA` dialog box appears, displaying all the fields used in the table (see Figure 5.74).

Figure 5.74: Showing the selection of fields

11. Select the fields that you want to include in the view and click the `Copy` button, as shown in Figure 5.74:

All the selected fields are displayed. Repeat step 5 for the second table, `ZSUBJECT_DATA`; however, do not select the fields that are already selected for the `ZSTUDENT_DATA` table. Finally, the maintenance screen displaying all the selected fields appears, as shown in Figure 5.75:

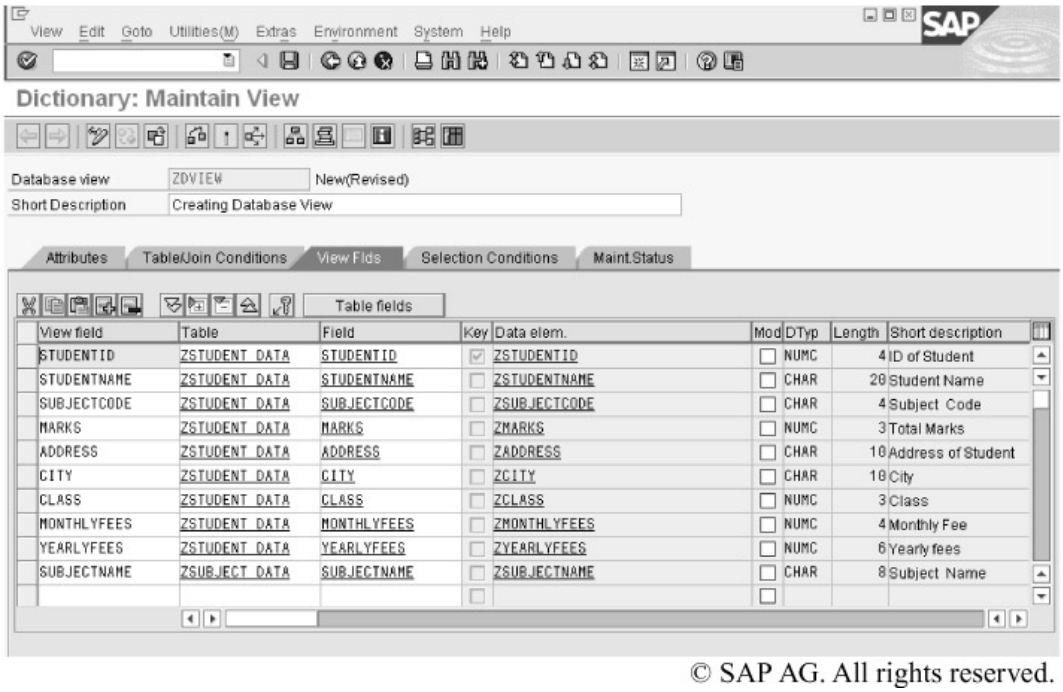


Figure 5.75: Fields copied in the view

Note If you want to insert restrictions for the data records to be displayed with the view, select the `Selection Conditions` (Selection Conditions) tab. In this case, we do not define any selection condition.

12. Select the `Maint. Status` (Maint.Status) tab to define the value for the `Data Browser/Table View Maint.` field. In this case, we have selected the value as the `Display/Maintenance Allowed with Restrictions` option,

as shown in Figure 5.76:

13. Click the `Save` icon to save all the settings related to the database view in the ZKOG_PCKG package.

    Note You can maintain technical settings, such as Buffering and Buffering Type, by selecting Goto > Technical Settings.

14. Click the `Back` icon and then the `Activate` icon to activate the database view.

    Note When the Activate icon is clicked, a log, known as the activation log, is written. The activation log can be displayed by selecting Utilities > Activation log from the maintenance screen of the table.

15. Now, select `Utilities(M) > Contents(H)` to view the content regarding the tables used in the view, as shown in Figure 5.77:

Figure 5.76: Showing the maintenance status for database view

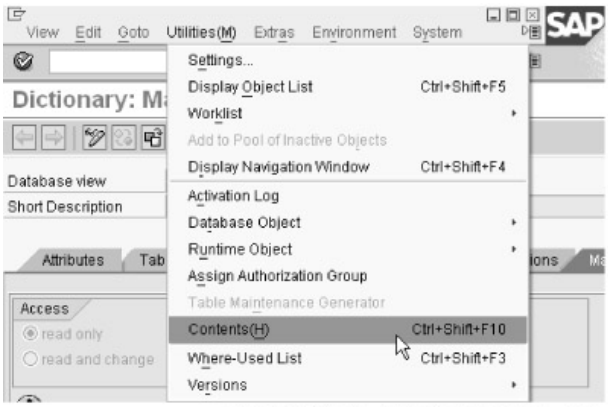


Figure 5.77: Displaying the content

The selection screen for `Data Browser: Table ZDVIEW` appears, as shown in Figure 5.78:

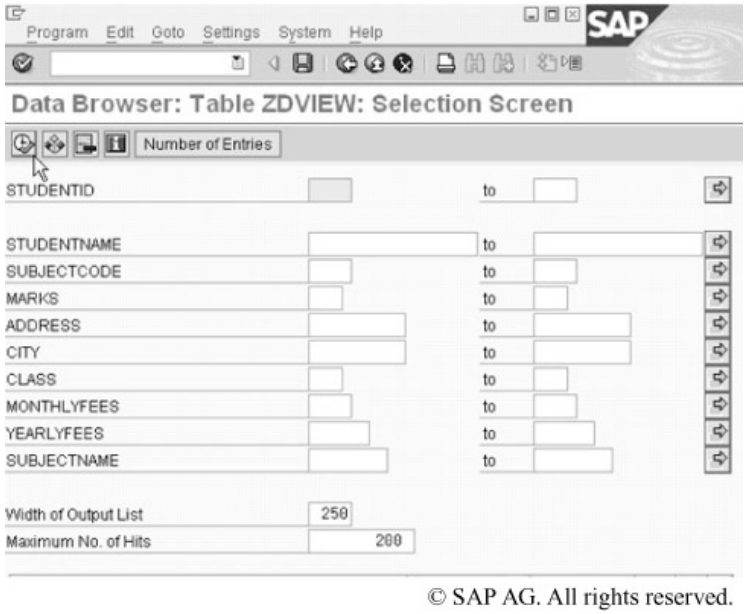


Figure 5.78: The selection screen for database view

16. Click the `Execute` icon on the application toolbar.

The `Data Browser: Table ZDVIEW Select Entries` screen appears, displaying the output of the database view, as shown in Figure 5.79:

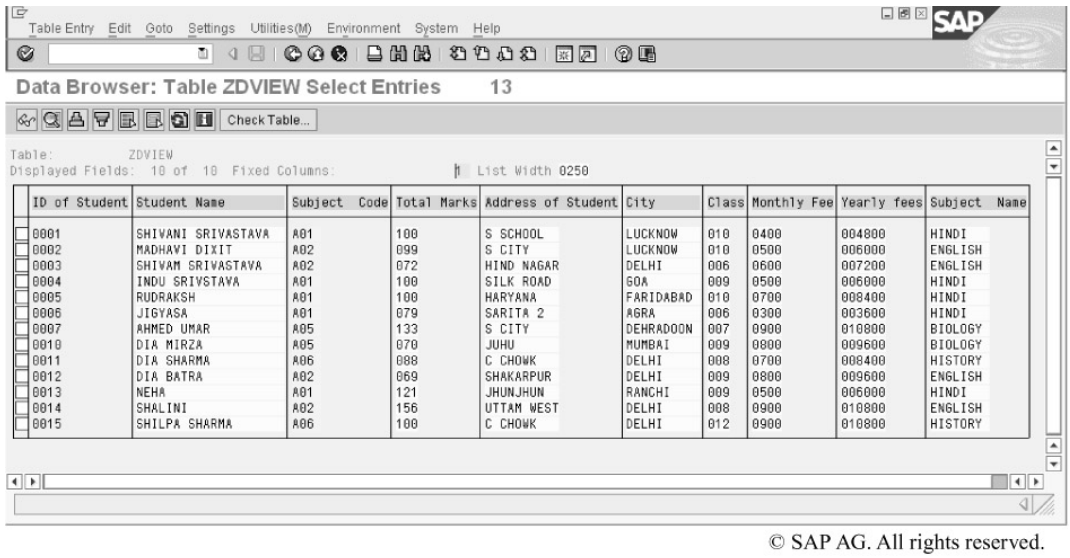


Figure 5.79: Output of the database view

Now, let's learn how to create a projection view for the database table.

Projection Views

Projection views are special views used to mask certain fields in a table, displaying only the selected fields. You cannot define any selection conditions in a projection view. Apart from transparent database tables, pooled and cluster tables can be accessed with the help of a projection view. Figure 5.80 shows how to apply a projection view on a database table:

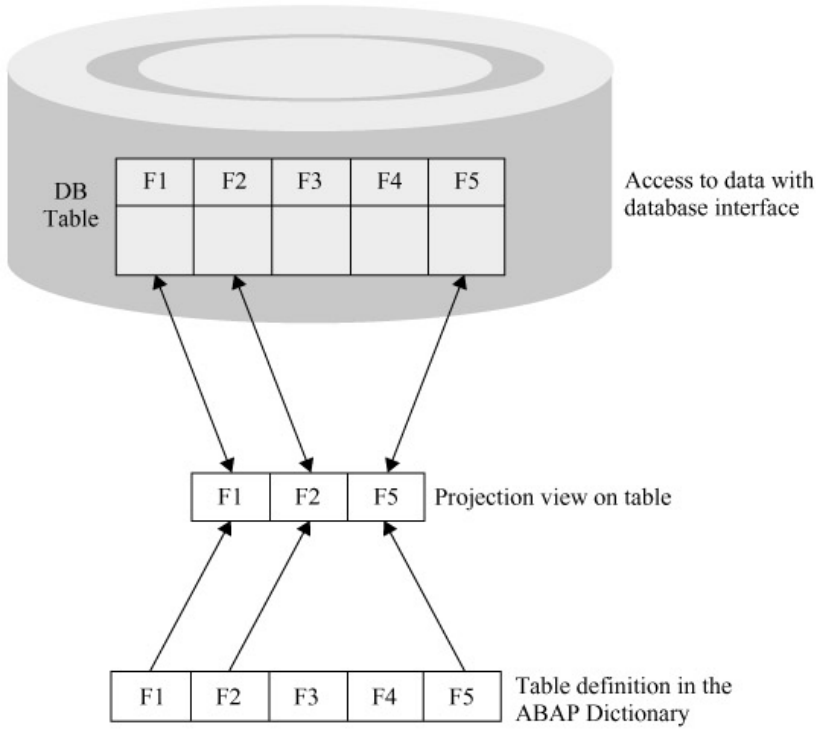


Figure 5.80: Projection view

Figure 5.80 shows that only the selected fields, such as F1, F2, and F5, of a database table are displayed in the projection view. Note that only one table can be viewed in a projection view.

Perform the following steps to create a projection view:

1. Select the `View` radio button on the initial screen of ABAP Dictionary. Enter the name of the view to be created and then click the `Create` button. In our case, we have entered the name of the view as `ZPVIEW`.

The `Choose View Type` dialog box appears.

2. Select the appropriate view to be created. In our case, we have selected the `Projection view` radio button, as shown in Figure 5.81:

3. Click the `Copy`  button.

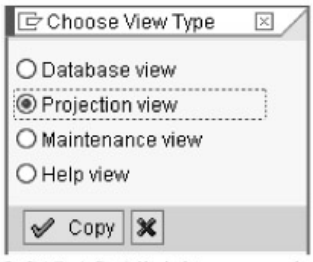


Figure 5.81: Selecting the view type

The `Dictionary: Maintain View` screen appears (see Figure 5.82).

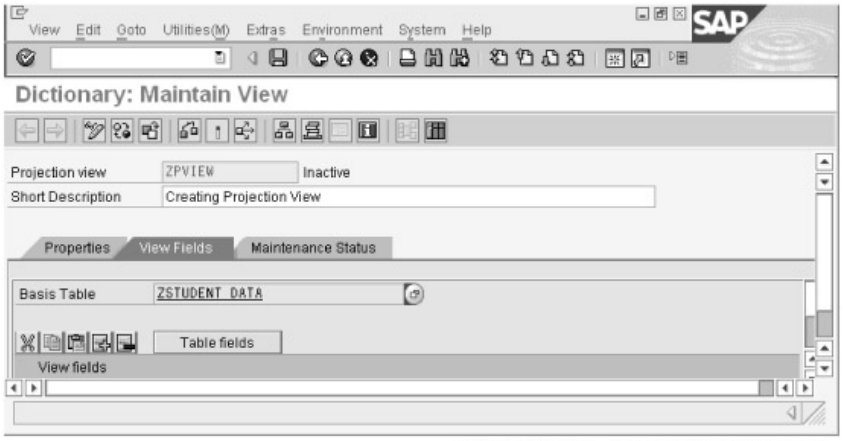


Figure 5.82: Showing the maintenance screen for the projection view

4. Enter a short description in the `Short Description` field, such as Creating Projection View, as shown in Figure 5.82:

5. Enter the name of the table to be used in the projection view in the `Basis Table` field. In this case, we have entered the name of the basis table as `ZSTUDENT_DATA`.

6. Click the `Table fields` button to include the fields of the `ZSTUDENT_DATA` table in the projection view.

The `Field Selection from Table ZSTUDENT_DATA` screen appears (Figure 5.83).

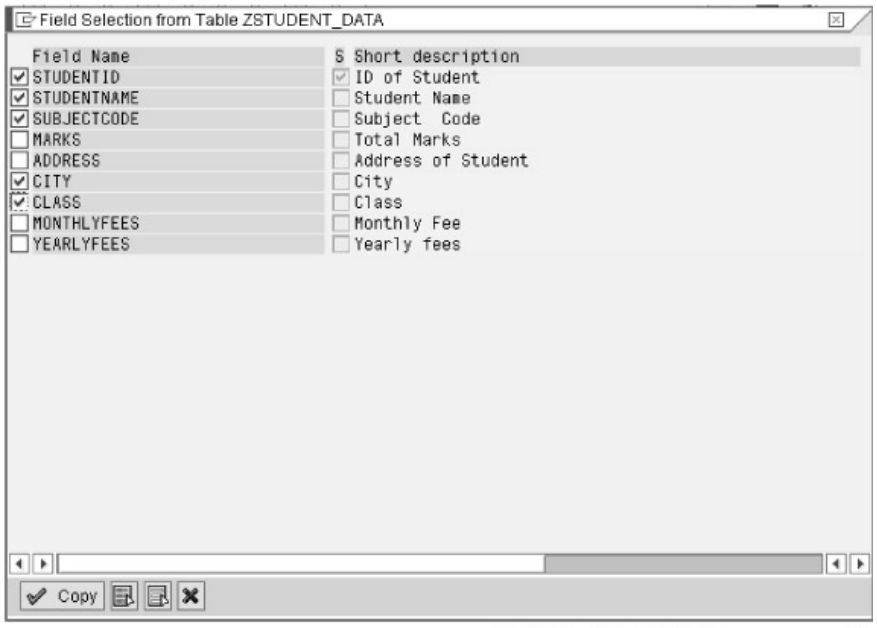


Figure 5.83: Selecting fields for projection view

7. Select the fields that you want to include in the projection view, as shown in Figure 5.83:

In this case, we have selected the `STUDENTID`, `STUDENTNAME`, `SUBJECTCODE`, `CITY`, and `CLASS` fields.

8. Click the `Copy` (Copy) button. All the selected fields for the projection view are displayed on the maintenance screen, as shown in Figure 5.84:

9. Select the `Maintenance Status` (Maintenance Status) tab to define an access method. Select the `read only` radio button, as shown in Figure 5.85:

10. Select the `Display/Maintenance Allowed with Restrictions` option from the `Data Browser/Table View Maint.` drop-down menu (see Figure 5.85).

11. Finally, click the `Save` icon and then click the `Activate` icon.

12. In the `Dictionary: Maintain View` screen, select `Utilities (M) > Contents` (see Figure 5.85).

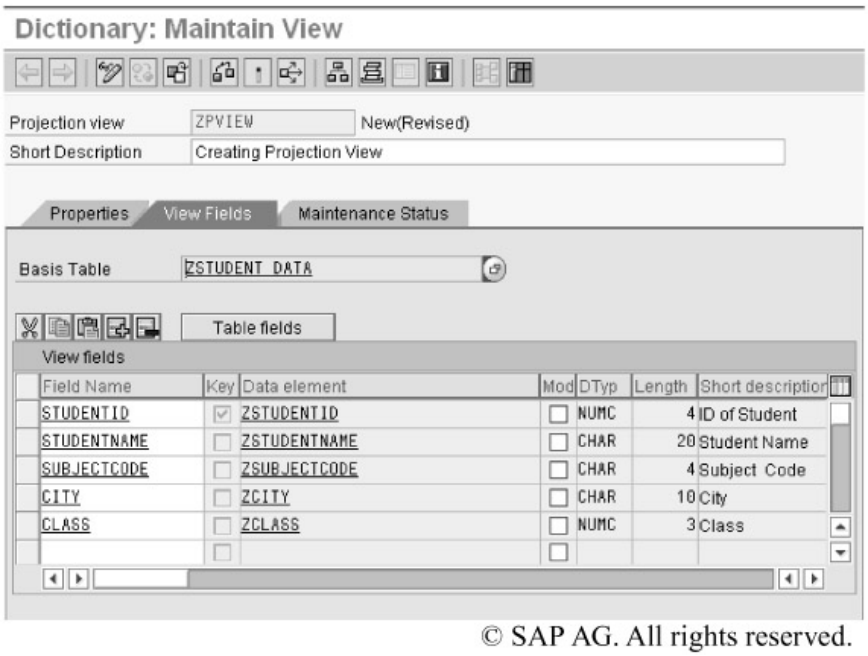


Figure 5.84: Fields used for projection views

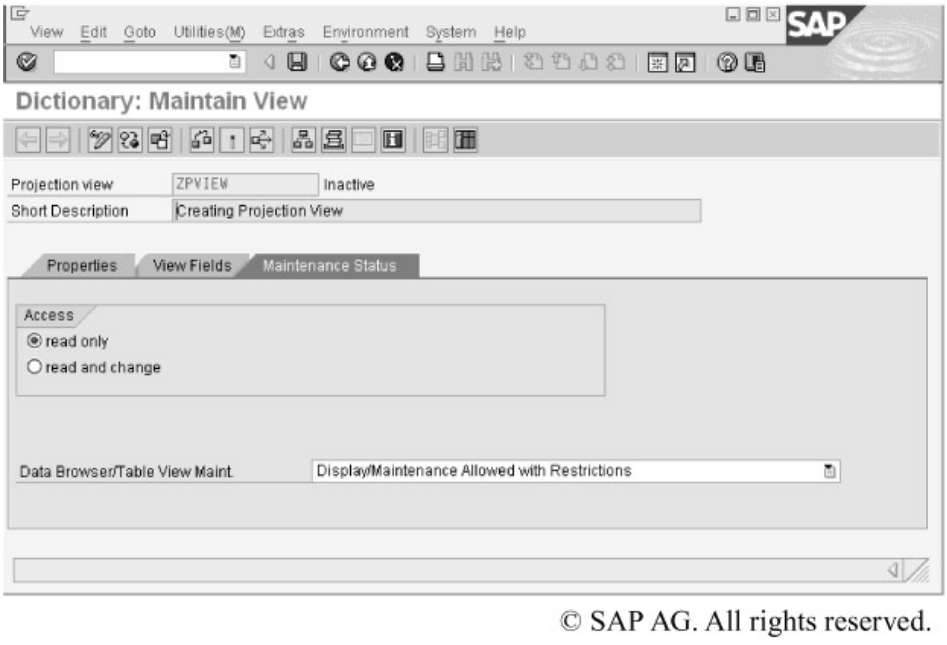


Figure 5.85: Defining the maintenance status for projection view

This action displays the selection screen for `Data Browser: Table ZPVIEW`, as shown in Figure 5.86:

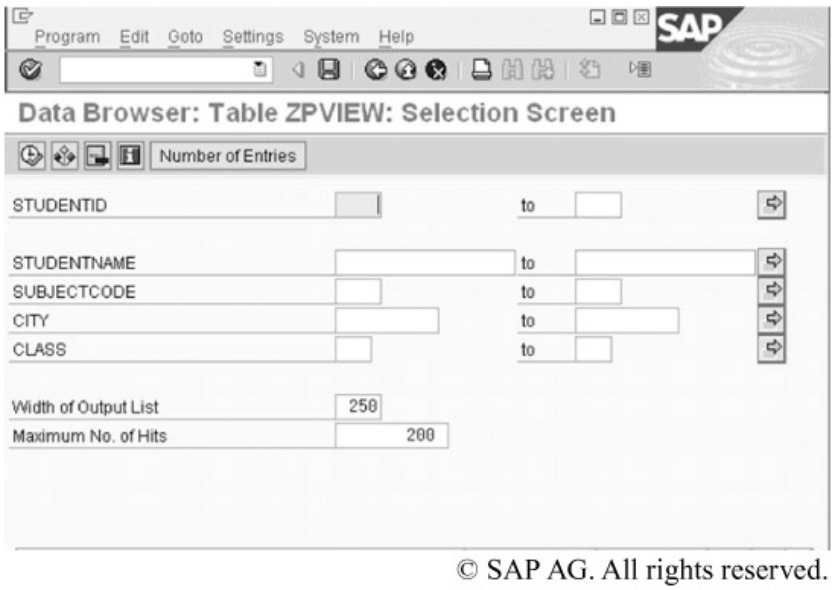


Figure 5.86: The selection screen

13. Click the `Execute` icon. The output of the projection view appears, as shown in Figure 5.87:

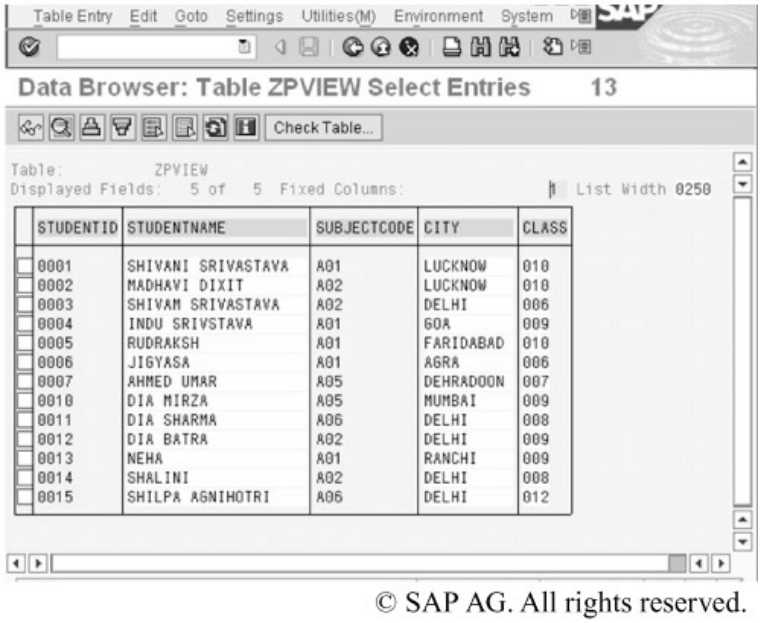


Figure 5.87: Output of the projection view

Now, let's learn how to create a maintenance view.

Maintenance Views

Multiple tables of a database can be combined to form a logical unit, but only if they are connected to each other by foreign keys. This logical unit can act as an application object. You can use a maintenance view to display and modify the data stored in an application object. Every maintenance view has a maintenance status associated with it, which determines the operations that have to be performed on the data of the associated tables.

Note A foreign key relationship must exist between the tables used in a maintenance view. The foreign key relationship is used to define the join conditions for the maintenance view.

Figure 5.88 displays the maintenance view defined for three tables:

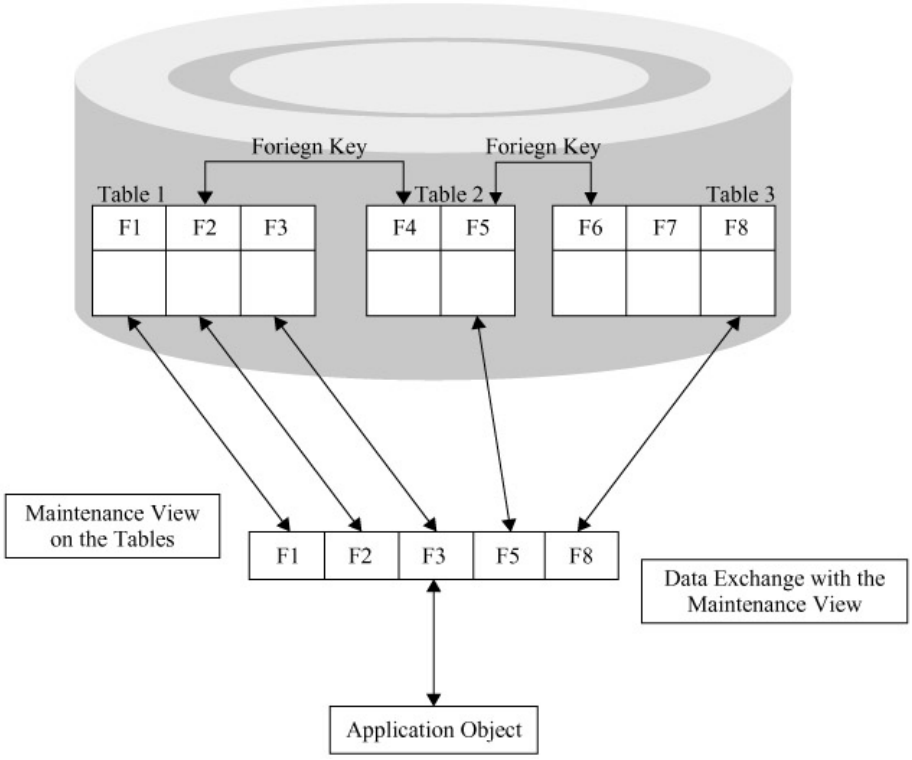


Figure 5.88: Showing the maintenance view

Figure 5.88 shows three tables: Table 1, Table 2, and Table 3. The F2 field of Table 1 is connected to the F4 field of Table 2 with a foreign key. Similarly, the F5 field of Table 2 is connected to the F6 field of Table 3 with a foreign key. The maintenance view is implemented on the three tables and is used to extract data from the F1, F2, F3, F5, and F8 fields.

Perform the following steps to create a maintenance view:

1. Select the `View` radio button from the initial screen of ABAP Dictionary. Enter the name of the view to be created and click the `Create` button. In this case, we have entered the name of the view as ZMVIEW.

The `Choose View Type` dialog box appears (see Figure 5.89).

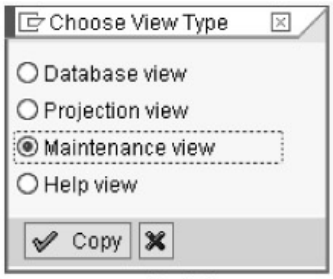


Figure 5.89: Selecting type of the view

2. Select the `Maintenance view` radio button, in the `Choose View Type` dialog box, as shown in Figure 5.89:
3. Click the `Copy` (Copy) button.

The `Dictionary: Maintain View` screen appears.

4. Enter a short description in the `Short Description` field, such as Creating Maintenance View, as shown in Figure 5.90:
5. Enter the name of the primary table of the view under the `Tables` column in the `Tables/Join Conditions` tab page. In this case, we have entered the name of the primary table as `ZSTUDENT_DATA`, as shown in Figure 5.91:

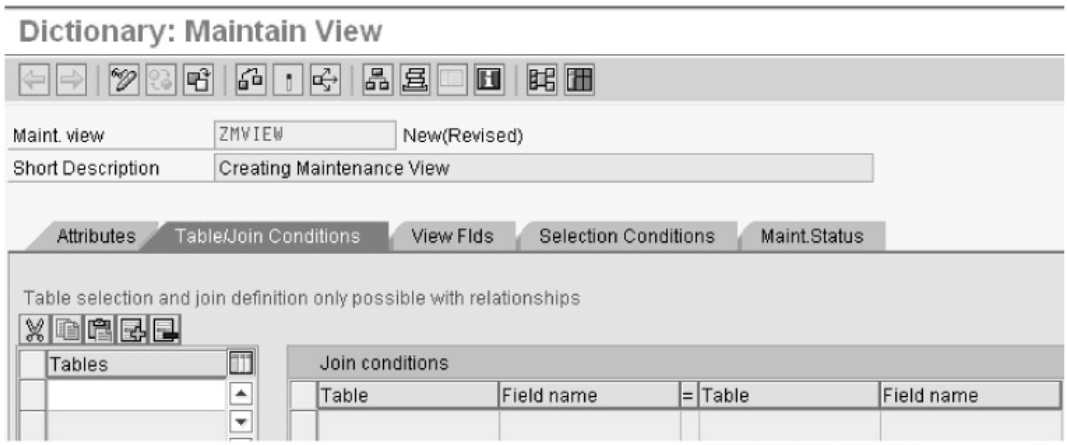


Figure 5.90: Maintenance screen

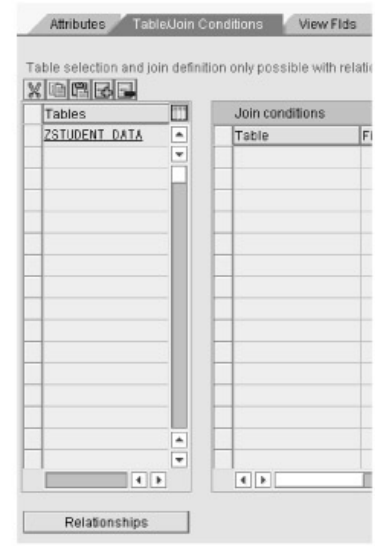


Figure 5.91: Primary table

Note that in a maintenance view, only those tables that are linked to the primary table (indirectly) with a foreign key can be viewed.

6. Place the cursor on the name of the primary table and click the `Relationships` button.

The `Relationships of Table ZSTUDENT_DATA` dialog box appears, as shown in Figure 5.92:

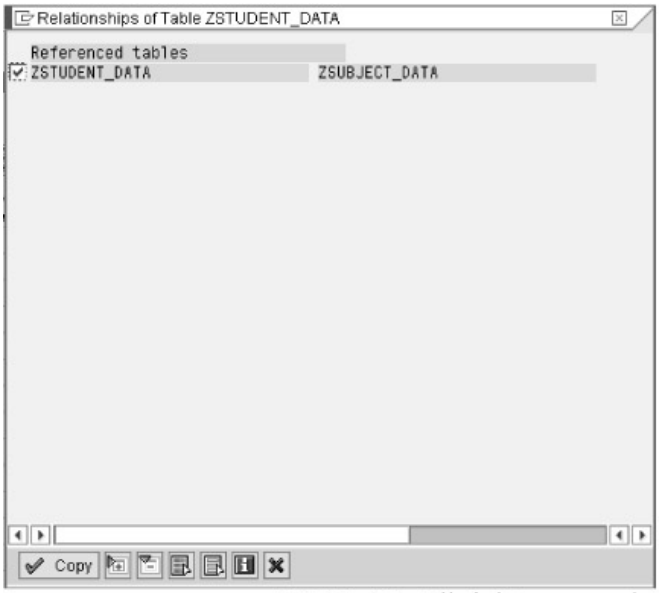


Figure 5.92: Relationships between tables

7. Select the ZSTUDENT_DATA check box (see Figure 5.92).

8. Click the Copy (Copy) button. The secondary table, that is, ZSUBJECT_DATA, is now included in the mainten ance view.

Note You can also include more secondary tables associated with the mentioned secondary table, that is, ZSUBJECT_DATA. This can be done by first placing the cursor on the name of that secondary table and then clicking the Relationships button.

Figure 5.93 displays the name of the primary and secondary tables in the maintenance screen:

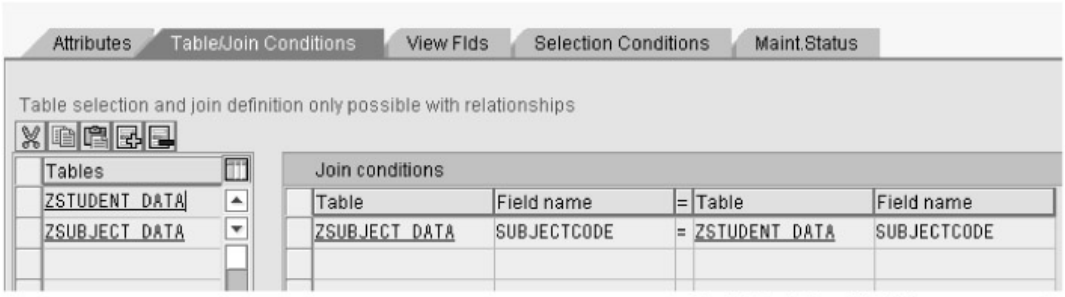


Figure 5.93: Primary table and secondary table

9. Select the View Flds (View Flds) tab to copy the fields used in the view.

10. Click the Table fields (Table fields) button. The names of the tables appear in a dialog box, as shown in Figure 5.94:

11. Click the name of the tables, one at a time, and then click the Choose (Copy) button.

Figure 5.94: Name of the tables

Note All the key fields related to the primary table (ZSTUDENT_DATA, in our case) must be included in the maintenance view. In addition to this, all the key fields of the secondary tables that are not included in the foreign key relationship must also be included in the view.

The `Field Selection from Table ZSTUDENT_DATA` dialog box appears.

12. Select the check boxes associated with the fields that are to be included in the view. Finally, click the `Copy` (Copy) pushbutton, as shown in Figure 5.95:

13. Repeat the same procedure for the secondary table, that is, ZSUBJECT_DATA. Select all the fields except those that are already selected and then click the `Copy` (Copy) button, as shown in Figure 5.96:

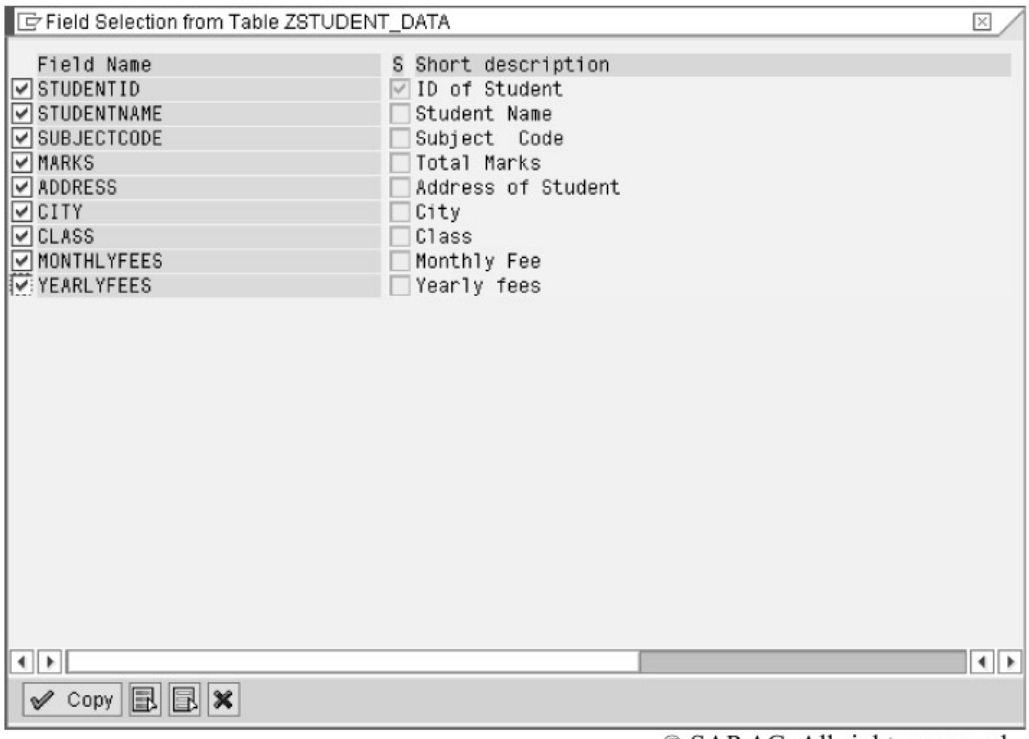


Figure 5.95: Selecting fields for a view

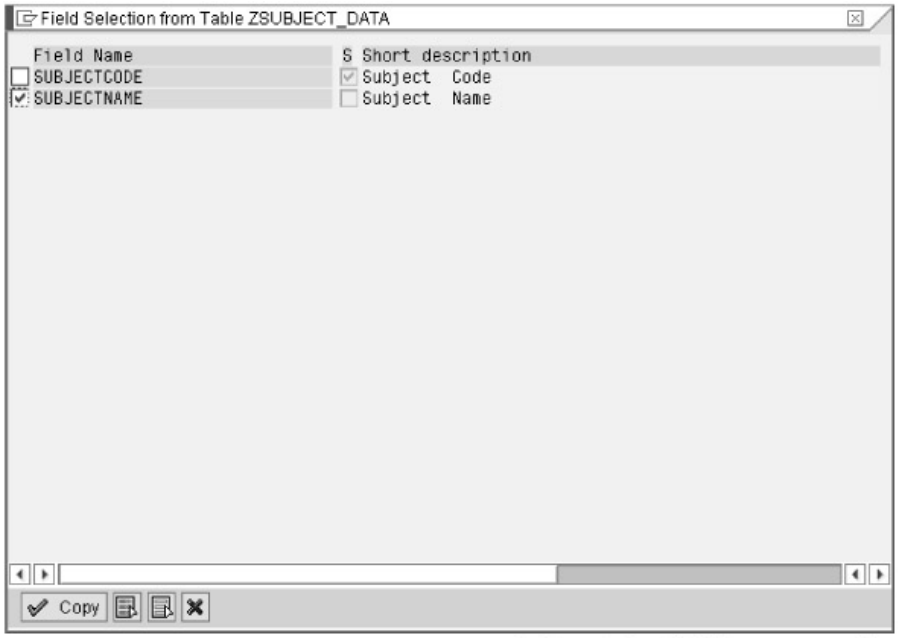


Figure 5.96: Selecting fields from the secondary table

Figure 5.96 shows the selection of the SUBJECTNAME field from the secondary table.

Note If you want to insert restrictions for the data records to be displayed with the view, select the `Selection Conditions` (Selection Conditions) tab. In this case, we do not define any selection condition.

14. Select the `Maintenance Status` (Maint.Status) tab to define the maintenance status, as shown in Figure 5.97:
15. Click the `Save` icon and save the view in the ZKOG_PCKG package. Now, click the `Activate` icon.
16. Select `Utilities(M) > Table Maintenance Generator` to generate a maintenance dialog box, as shown in Figure 5.98:

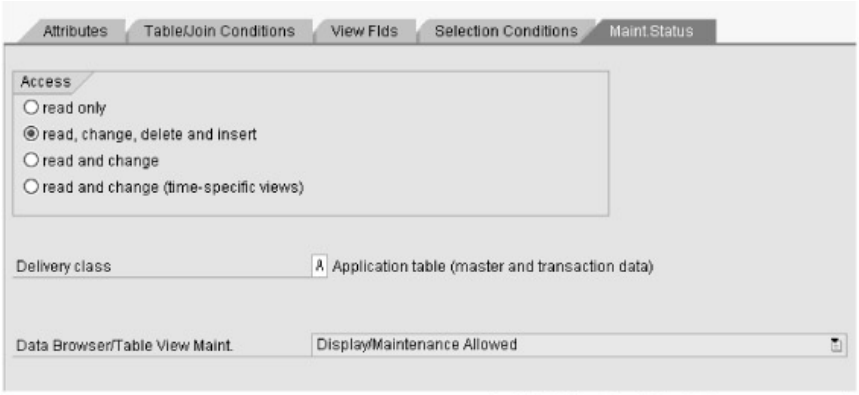


Figure 5.97: Defining maintenance status

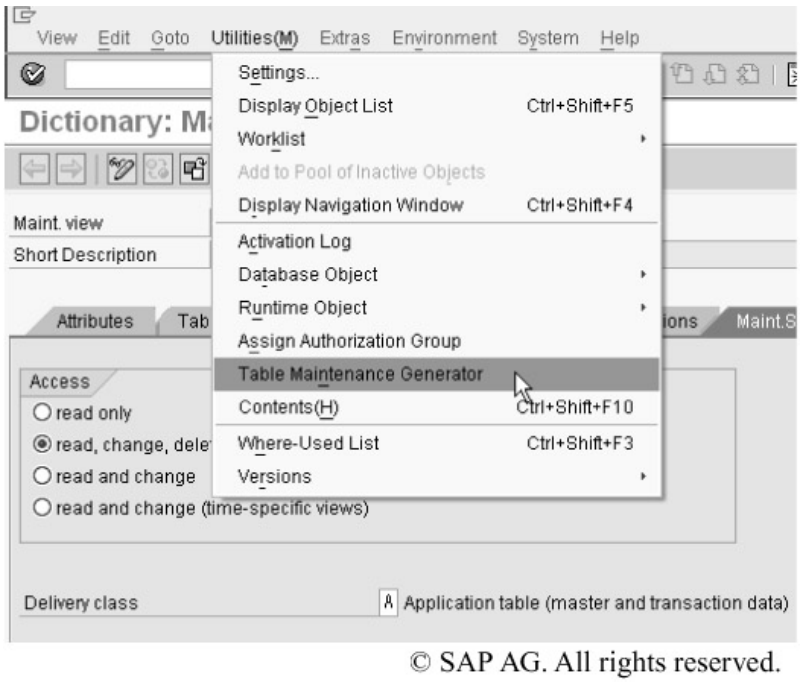


Figure 5.98: Generating a maintenance dialog box

The `Generate Table Maintenance Dialog: Generation Environment` screen appears, as shown in Figure 5.99:

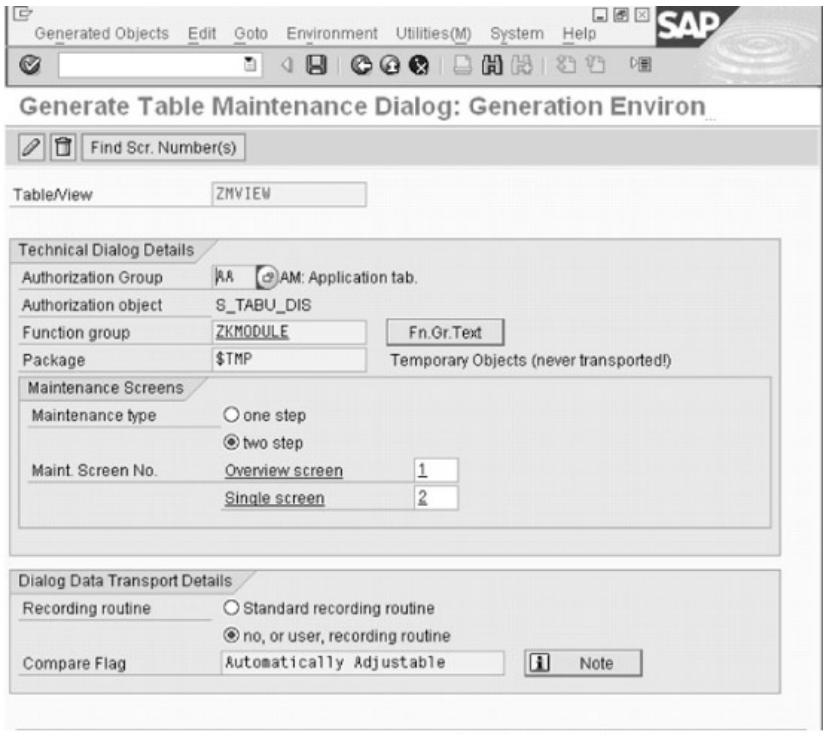


Figure 5.99: The generate table maintenance dialog—generation environment screen

Table 5.14 describes the functions of different maintenance dialog fields that appear in Figure 5.99:

Table 5.14: Maintenance dialog fields

Field	Description
Authorization Group	Shows that the user is responsible for maintaining the content of the table/view.

Function Group	Shows the function group in which the tables/view-specific maintenance dialog components are generated.
Maintenance Type	Contains two radio buttons, one step and two step, to maintain the settings of the screen (see Figure 5.99).
Maint. Screen No.	Shows the internal number of each maintenance screen.
Recording Routine	Shows the table/view contents that can be transported in the SAP system.

17. Enter AA as the authorization group, ZKMODULE as the function module, and select the `two step` radio button as the maintenance type (see Figure 5.99).

18. Finally, click the `Save` icon to save the settings for the maintenance dialog. Now, click the `Back` button to return to the screen shown in Figure 5.98.

19. Select `Utilities > Contents(H).` The `Display View "Creating Maintenance View": Overview` screen appears, as shown in Figure 5.100:

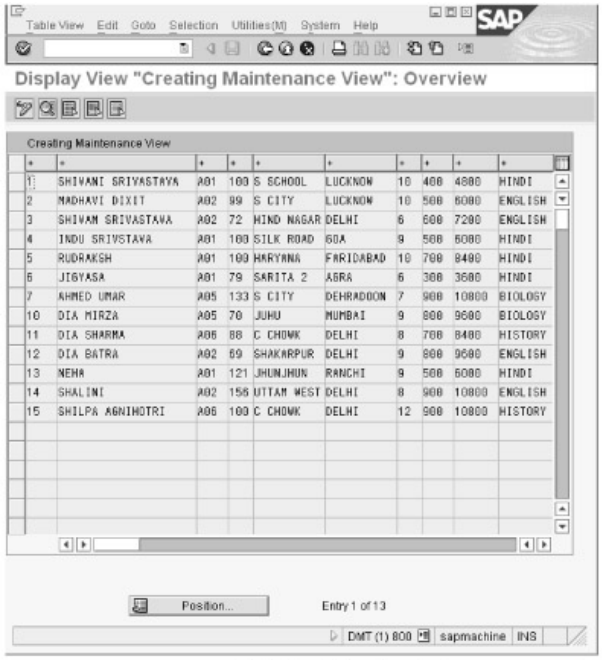


Figure 5.100: Showing the display overview screen

The screen in Figure 5.100 is shown in the display mode. Now, let's look how to change an existing entry in a field.

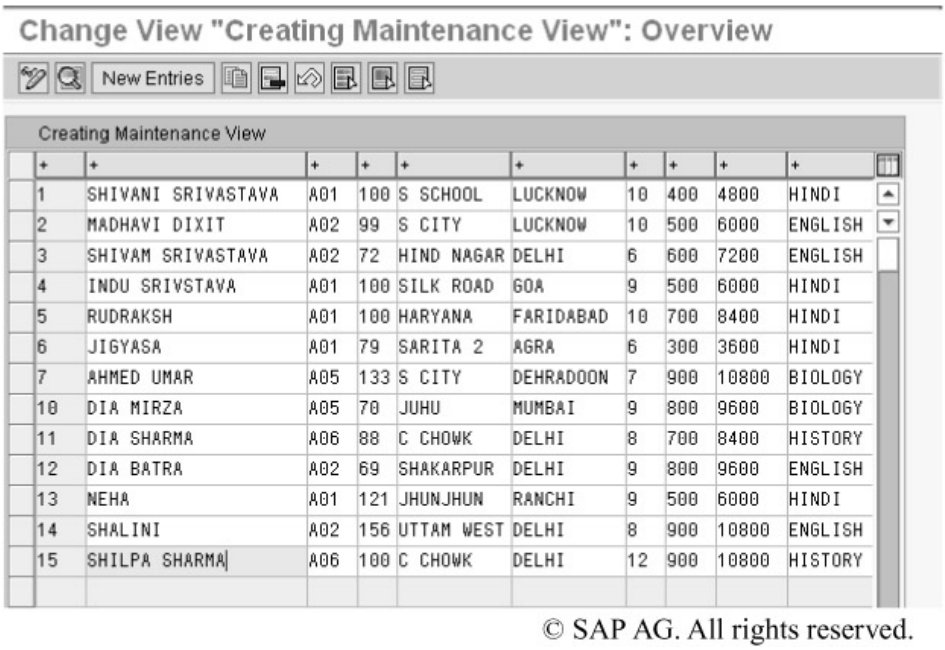
20. Select the entry for a field that you want to change and select the `Display > Change` icon.

The `Information` dialog box appears, as shown in Figure 5.101:

Figure 5.101: Showing the information dialog box

21. Click the `Continue` icon to continue.

The `Change View "Creating Maintenance View" Overview` screen appears, as shown in Figure 5.102:



1	SHIVANI SRIVASTAVA	A01	100	S SCHOOL	LUCKNOW	10	400	4800	HINDI
2	MADHAVI DIXIT	A02	99	S CITY	LUCKNOW	10	500	6000	ENGLISH
3	SHIVAM SRIVASTAVA	A02	72	HIND NAGAR	DELHI	6	600	7200	ENGLISH
4	INDU SRIVSTAVA	A01	100	SILK ROAD	GOA	9	500	6000	HINDI
5	RUDRAKSH	A01	100	HARYANA	FARIDABAD	10	700	8400	HINDI
6	JIGYASA	A01	79	SARITA 2	AGRA	6	300	3600	HINDI
7	AHMED UMAR	A05	133	S CITY	DEHRADOON	7	900	10800	BIOLOGY
10	DIA MIRZA	A05	70	JUHU	MUMBAI	9	800	9600	BIOLOGY
11	DIA SHARMA	A06	88	C CHOWK	DELHI	8	700	8400	HISTORY
12	DIA BATRA	A02	69	SHAKARPUR	DELHI	9	800	9600	ENGLISH
13	NEHA	A01	121	JHUNJHUN	RANCHI	9	500	6000	HINDI
14	SHALINI	A02	156	UTTAM WEST	DELHI	8	900	10800	ENGLISH
15	SHILPA SHARMA	A06	100	C CHOWK	DELHI	12	900	10800	HISTORY

Figure 5.102: Showing the change view overview screen

22. Change the name "SHILPA AGNIHOTRI" to "SHILPA SHARMA", as shown in Figure 5.102. Finally, click the `Save` icon to save the changes.

Help Views

Help views are created when you want to use an outer join to obtain data from database tables. This view is generally used to create a Search Help object; that is, to provide input helps (F4) for different fields. In Search Help, these views act as selection methods. These selection methods can be either a table or a view, which means that data can be selected from either a table or a view. Note that a table is used as a selection method in a help view only if you need to use an outer join to retrieve the data. Figure 5.103 shows how to apply a Help view on a database table:

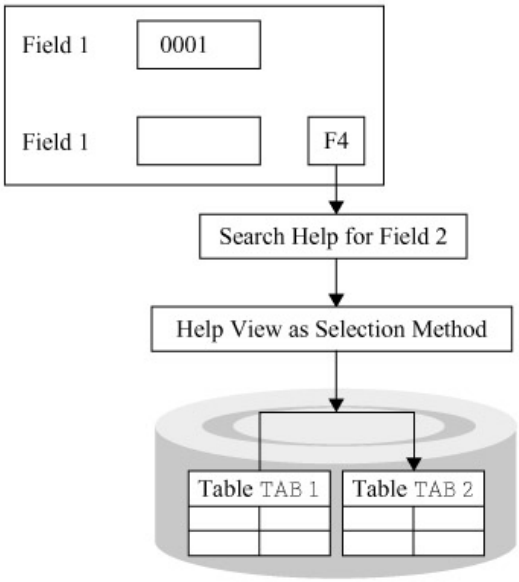


Figure 5.103: Implementing the help view on database tables

Figure 5.103 shows the method of implementing the Help view. In Figure 5.103, we have selected the Help view as the selection method for Field 2.

Perform the following steps to create a Help view:

1. Select the `View` radio button from the initial screen of ABAP Dictionary. Enter the name of the view to be created (in this case, the `ZHVIEW`) and click the `Create` button.

The `Choose View Type` dialog box appears.

2. Select the `Help view` radio button on the `Choose View Type` dialog box, as shown in Figure 5.104:
3. Click the `Copy` (Copy) button.

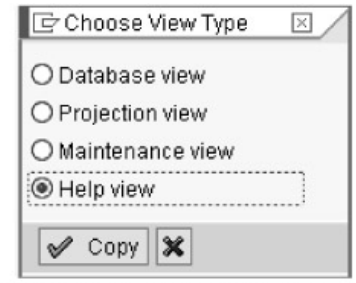


Figure 5.104: Selecting the type of the view

The `Dictionary: Maintain View` screen appears, as shown in Figure 5.105:

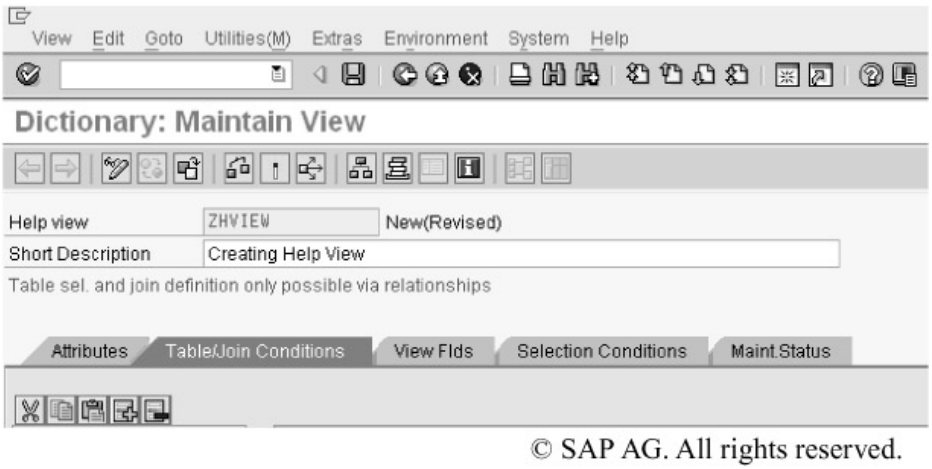


Figure 5.105: Description for the help view

4. Enter a short description in the `Short Description` field, such as Creating Help View.
5. Click the `Save` (Save) icon to save the view in the `ZKOG_PCKG` package.
6. Now, enter the name of the primary table to be used in the view under the `Tables` column in the `Tables/Join Conditions` tab page. The table that is linked with this primary table with the help of a foreign key, that is, `ZSTUDENT_DATA`, is included in the view created.
7. Position the cursor on the name of the primary table and click the `Relationship` button. When you click this button, you see the screen, that was previously shown in Figure 5.92. In Figure 5.92, we selected the check box showing the name of the primary table. Now, click the `Copy` (Copy) button. The respective join condition appears on the screen for the `Table/Join Conditions` tab page, as shown in Figure 5.106:
8. Select the `View Flds` tab to select the fields that you want to include in the view. Note that the key fields of the primary table are copied automatically to the view as proposals.
9. Select the `Table Fields` (Table fields) button. When you click this button, a list of the tables contained in this view

is displayed.

10. Select any table and click the `Copy` (Copy) button. Repeat the same for all the tables. Figure 5.107 shows the screen containing the fields included in the view:

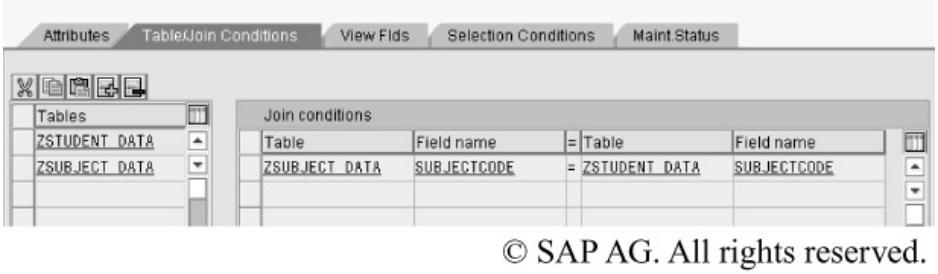


Figure 5.106: The screen showing the join conditions

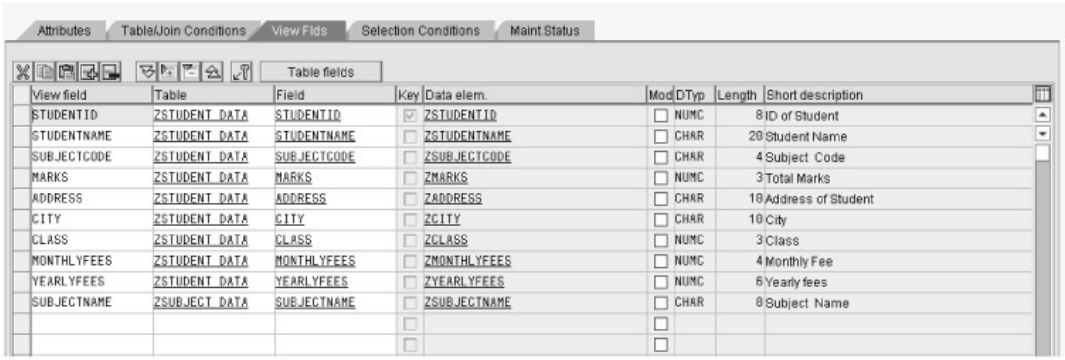


Figure 5.107: Fields included in the view

In Figure 5.107, you see all the fields that are selected for the Help view.

11. Select the `Maint. Status` (Maint Status) tab to set the access method, as shown in Figure 5.108:
12. In the `Access` group box of the `Maint. Status` tab, select the `read only` radio button.
13. Now, click the `Activate` (↑) icon from the application toolbar. The created Help view is now activated and can be used as a selection method for Search Help.

Figure 5.108: The access method

You can restrict the set of data records that can be selected in a view by defining a selection condition. These conditions can be placed on different entries under the `Selection Condition` tab. Table 5.15 shows the description of the entries on the `Selection Conditions` tab:

Table 5.15: Description of the entries on the selection condition tab

Entry	Description
Table	Shows the name of the base table from which the field is taken.

Field name	Shows the name of the field for which the selection condition is formulated.
Operator	Defines the operators used to compare the field contents, and the comparison value. You can find the set of valid operators in the F4 help.
Comparison value	Shows the constant value by which the field value can be compared.
AND/OR	Shows the link between two lines of the selection condition.

Deleting Views

A view can be deleted if it is no longer in use.

Perform the following steps to delete a view:

1. Select the `View` radio button and enter the name of the view in the initial screen of ABAP Dictionary. Select the `Where-used list`  icon to find the database tables associated with this view.
2. Select the `Delete`  icon to delete the selected view. A dialog box appears, prompting you to confirm the delete request.
3. Confirm the delete request by clicking the `Yes` button. Consequently, the view is deleted from ABAP Dictionary and from the database being used by the SAP system.

Exploring Search Helps

Search Helps, another repository object of ABAP Dictionary, are used to display all the possible values for a field in the form of a list. This list is also known as a hit list. You can select the values that are to be entered in the fields from this hit list. The following are the three types of Search Help objects provided by ABAP Dictionary:

- Elementary Search Help
- Collective Search Help
- Append Search Help

Elementary Search Help

The elementary Search Help is used to define an input help. This input help allows you to perform the following functions:

- Defining the source from where the data displayed in the hit list is retrieved (the selection method)
- Defining the Search Help parameters; that is, information related to the value selection
- Defining the dialog steps to be executed in the input help

The data in a hit list can be retrieved from a single database table, multiple tables, or a client-specific table.

The interface of Search Help defines the context data that can be used in the input help and the data that can be returned to the input template. This interface consists of parameters that define the fields of the selection method. Table 5.16 lists the descriptions of different kinds of interface parameters:

Table 5.16: Types of parameters

Search Help Parameter	Description
Import parameter	Used when the context information from the processed input template is copied to the Search Help process.
Export parameter	Used when the values from a hit list have to be returned to the input template.

Note that the parameters of Search Help must be assigned to a data element.

Collective Search Help

The collective Search Help is a combination of several elementary Search Helps. This Search Help is used to define alternative Search Help paths. A collective Search Help is a collection of several elementary Search Helps; therefore, you

can transfer the values between various elementary Search Helps and the input templates by using the collective Search Help in the input help. Consequently, a collective Search Help also has an interface to transfer the values of the associated fields.

Now, let's explain the following topics in the context of the structure of collective Search Helps:

- Interface of the collective Search Help
- Assigned Search Helps

Interface of the Collective Search Help

Similar to the elementary Search Help, the collective Search Help uses an interface for both the import and the export parameters. Data is exchanged between the screen template and the parameters of the assigned elementary Search Helps by using this interface. Figure 5.109 shows the exchange of data in a collective Search Help:

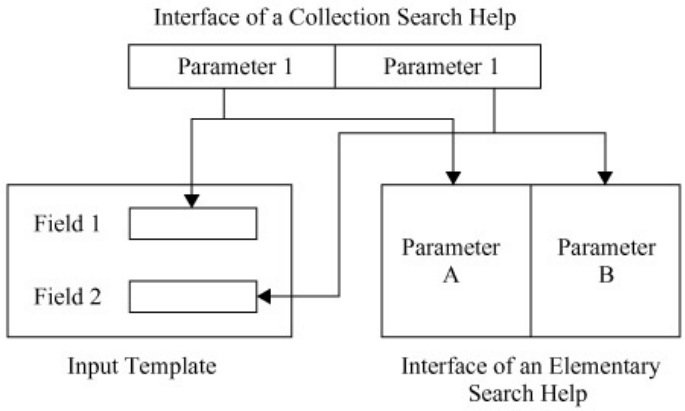


Figure 5.109: The exchange of data in a collective search help

Figure 5.109 shows how data is exchanged between a screen template and the parameters of the elementary Search Helps.

Assigned Search Helps

The assigned Search Help is a type of collective Search Help that does not include all the parameters of an elementary Search Help. Figure 5.110 shows the interface of the collective Search Help:

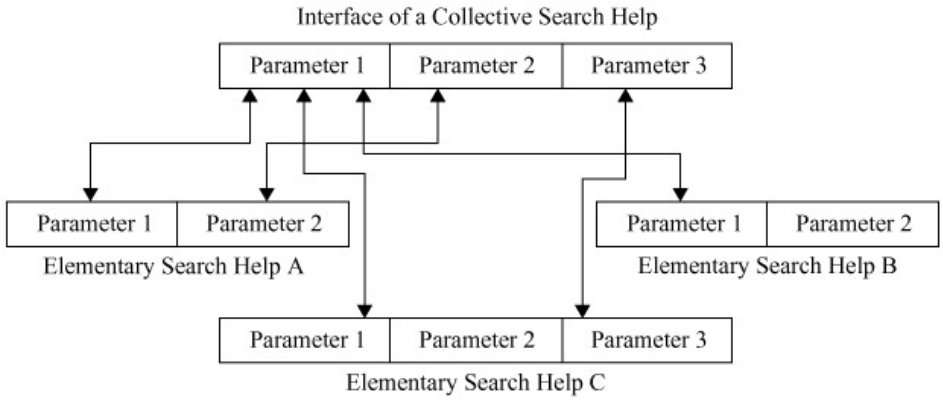


Figure 5.110: Interface of a collective search help

In Figure 5.110, you can see that certain parameters, such as Parameter 2 of the elementary Search Help B and Parameter 2 of the elementary Search Help C, are not included in the interface of the collective Search Help.

Append Search Help

You can enhance the features of the collective Search Help by appending further elementary Search Helps (also known as

search parts) to it. This kind of search help is known as an append search help. The appended elementary Search Helps can also be enhanced by appending further Search Helps to them. Figure 5.111 shows the structure of an append Search Help:

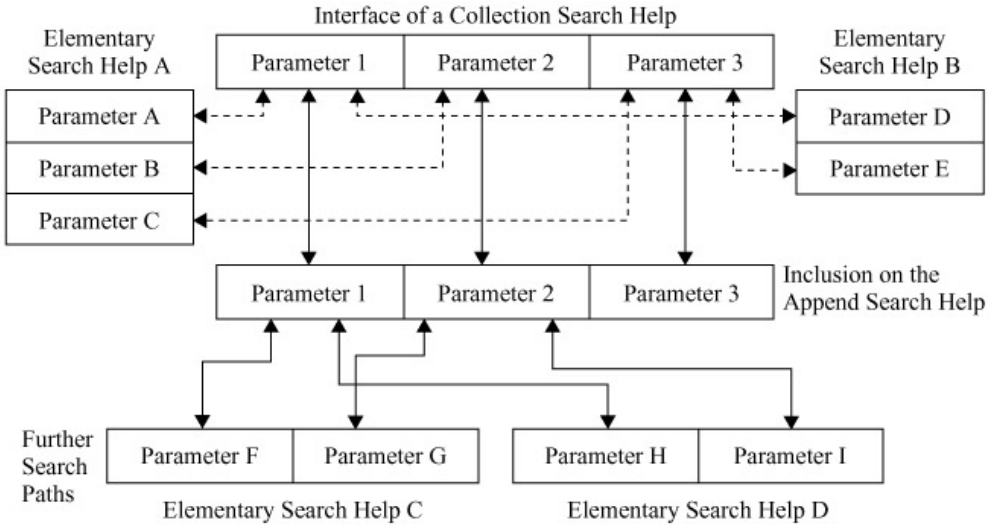


Figure 5.111: The append search help structure

Figure 5.111 shows the assignment of further search paths that is, the addition of elementary Search Helps, to an existing collective Search Help. You can see that the `Append search help` is appended to the collective Search Help. This appended Search Help is enhanced with the addition of two more Search Helps, C and D.

Exploring Lock Objects

Lock Object is a feature offered by ABAP Dictionary that helps multiple users to access the same set of data records synchronously. Data records are accessed with the help of specific programs. Consequently, any inconsistency at the time that data is inserted or changed in a table can be avoided by using lock objects. Figure 5.112 shows the simultaneous access to stored data in a database with the help of programs:

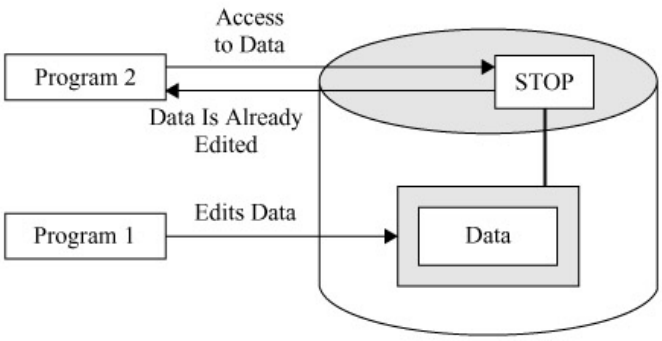


Figure 5.112: Showing simultaneous access to data

Simultaneous access is implemented with the help of a lock mechanism containing the function modules. These function modules are generated automatically from the definition of the Lock Objects in ABAP Dictionary, and they are used to set and release the Lock Objects when interactive transactions are performed.

The tables whose data records you want to lock must be defined in a Lock Object, along with their key fields. First, a primary table is selected, and then the foreign key relationships are used to add secondary tables to a Lock Object, as shown in Figure 5.113:

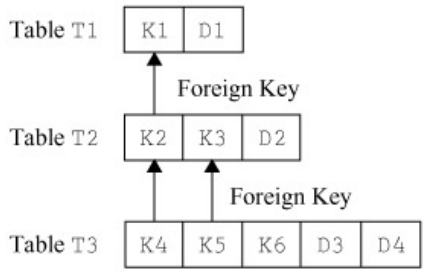


Figure 5.113: Structure of lock objects

Figure 5.113 shows the implementation of Lock Objects in tables T1, T2, and T3, which are related to each other with a foreign key.

Now, let's explain the following topics in the context of Lock Objects:

- Describing lock arguments
- Exploring the Lock mode and the Lock table
- Describing the Lock mechanism
- Creating Lock Objects

Describing Lock Arguments

The key fields of a table included in a Lock Object are called lock arguments. The function modules, generated after a Lock Object is created, use these lock arguments as their input parameters. These lock arguments are used to set and remove the locks generated by the Lock Object definition. Figure 5.114 shows the lock arguments containing the key fields of different tables:

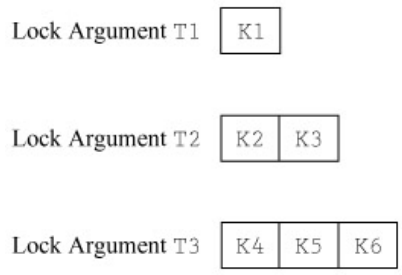


Figure 5.114: Lock arguments

Figure 5.114 shows that the lock arguments of the T1, T2, and T3 tables consist of the key fields.

A Lock Object can be placed simultaneously on the records of multiple tables. However, in case only one table is involved in the Lock Object, the primary key field of this table acts as its lock argument.

Figure 5.115 shows the entries locked in a table:

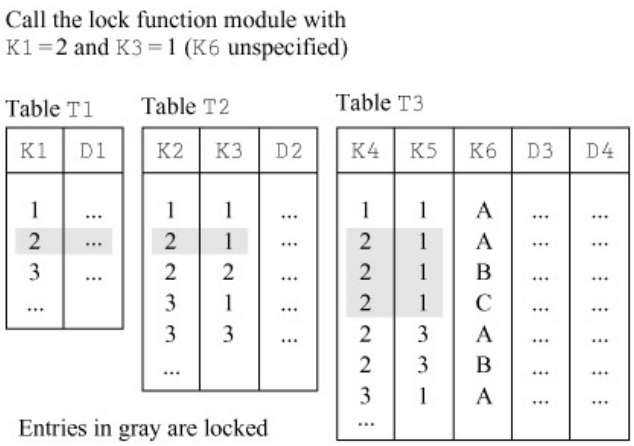


Figure 5.115: Entries locked in the table

In Figure 5.115, you see that the key fields of the T1, T2, and T3 tables are used as lock arguments. These lock arguments, that is, the key fields, are used as values for some of the fields in the function module called to set and release the lock. The function module used for the Lock Object is called whenever the value of the K1 field is 2 and the value of the K3 field is 1.

Exploring the Lock Mode and the Lock Table

A lock mode controls whether several users can access data records simultaneously. A lock mode can be assigned separately for each table in a Lock Object. When the lock is set, the corresponding lock entry is stored in the lock table associated with the table. A lock table is stored in the main memory of the enqueue server, which records the current locks in the SAP system. The lock table is used to manage locks and stores the record of the owner, type of lock mode used, name of the lock, and key fields of the table. Every time an enqueue server receives a request regarding a lock, the SAP system checks the table to determine whether the request is clashing with an existing lock. If it is clashing, then the request is rejected; otherwise, the new lock request is added to the lock table. Note that locks created in the lock table are not set at the database level. The structures for the entries in the lock table are shown in Figure 5.116:

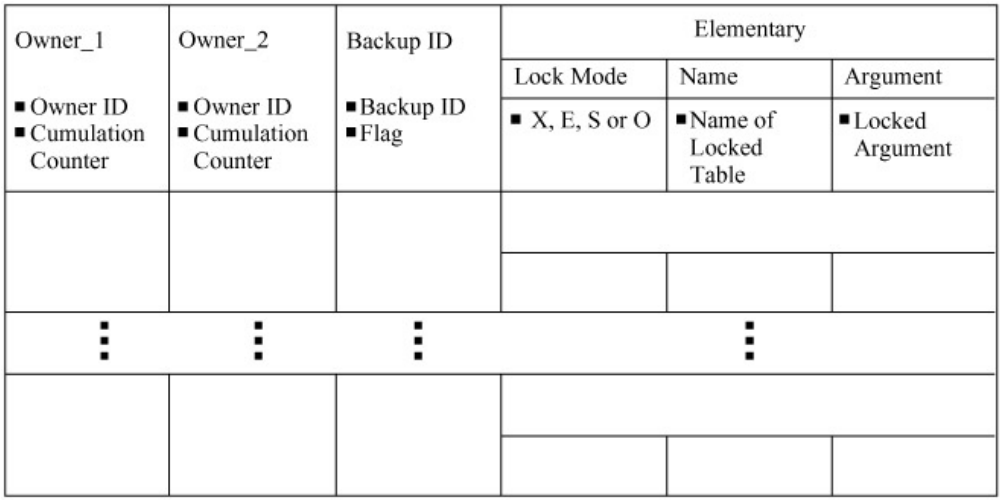


Figure 5.116: Structure of the lock table entries

Figure 5.116 shows that the structure of the lock entry contains various fields. Table 5.17 lists the descriptions of these fields.

Table 5.17: Description of the lock entry fields

Field		Content and Meaning
Owner_1		Contains the Owner ID and the cumulation counter. The Owner ID contains the computer name, the work process, and a timestamp. It is also used to identify the SAP Logical Unit of Work (LUW), whereas the cumulation counter specifies the frequency of setting the associated elementary lock.

Owner_2		Contains the Owner ID and the cumulation counter. The Owner ID contains the computer name, the work process, and a timestamp. It is also used to identify the SAP LUW, whereas the cumulation counter specifies the frequency of setting the associated elementary lock.
Backup ID		Contains the Backup ID (an index indicating where the lock entry is stored in the backup file) and Flag. The value 0 of Flag indicates that no backup is present for the lock, whereas the value 1 indicates that backup is present for the lock.
Elementary Lock	Lock Mode	Specifies the following lock modes: ▪ S (Shared lock) ▪ O (Optimistic lock) ▪ E (Exclusive lock) ▪ X (Exclusive lock, extended exclusive lock, cannot be cumulated)
	Name	Contains the name of the database table in which fields are to be locked.
	Argument	Contains the locked fields in a database table.

As mentioned previously, every Lock Object has a lock mode associated with it. Further, the lock mode describes the type of lock. Table 5.18 lists the descriptions of different kinds of lock modes:

Table 5.18: Descriptions of lock modes

Type of Lock	Lock Mode	Description
Shared lock	S (Shared)	Allows several users (transactions) to access the locked data at the same time in the display mode. A request for another shared lock is accepted, even if it comes from another user.
Exclusive lockz	E (Exclusive)	Protects the locked object against all types of locks from other transactions. Only the same lock owner can reset the lock (accumulate).
Exclusive but not cumulative lock	X (eXclusive non-cumulative)	Specifies that locks can be requested several times from the same transaction and are processed successively. In contrast, exclusive but not cumulative locks can be called only once from the same transaction.
Optimistic lock	O (Optimistic)	Behaves like shared locks initially and can be converted into exclusive locks.

Describing the Lock Mechanism

A lock mechanism is used to synchronize the access of several programs to the same set of data. The following are the two main functions accomplished with the lock mechanism:

- A program can communicate to other programs about data records that it is just reading or changing.
- A program can prevent itself from reading data that has just been changed by another program.

A logical condition is defined for the data records to be locked. When a lock is set, this logical condition is entered in a lock table. The condition remains effective until the program deletes it or the execution of the program is complete. All the locks set by the program are removed at the end of the program execution. When accessing the data records, the records just edited by other programs can be identified by an entry in the lock table. Figure 5.117 shows the lock mechanism:

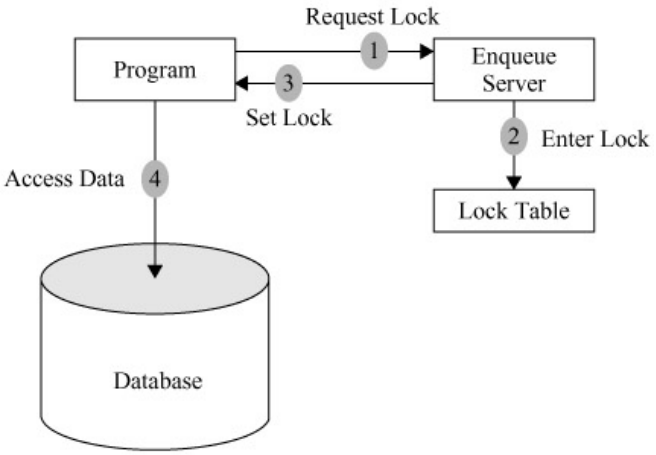


Figure 5.117: Lock mechanism

In Figure 5.117, a lock request is first generated by a program. Then this lock request goes to the enqueue server; and finally, the lock is created in the lock table. The enqueue server sets the lock, and the program is ready to access the data, with the Lock Object activated for the table.

A Lock Object must be defined in ABAP Dictionary. Whenever the Lock Object is activated, two function modules, `ENQUEUE_<lockobjectname>` and `DEQUEUE_<lockobjectname>`, are generated. The `ENQUEUE_<lockobjectname>` function module is called when data records are to be locked. When this function module is called, the values of the key fields that specify the records to be locked are passed to all the tables in the Lock Object. A generic lock is generated if a value is not passed to all the key fields. Depending on the lock mode, the SAP system decides whether the request for a lock made by another program is accepted or rejected.

Figure 5.118 shows the function module used to lock a table:

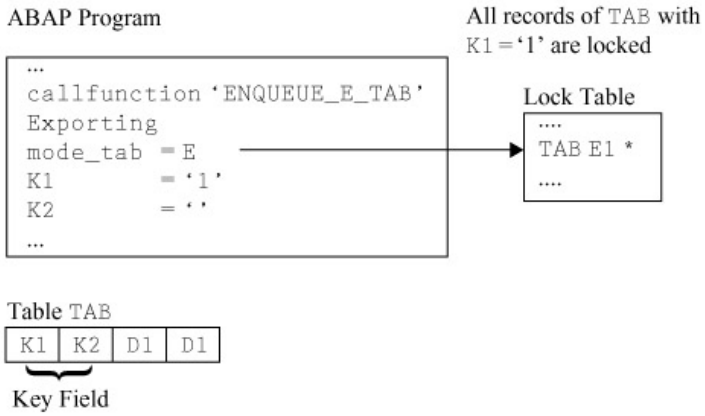


Figure 5.118: The function module used for locking

You can see that `ENQUEUE_E_TAB` is used to lock the table containing the field value of K1 as 1. The lock mode is assigned as E (exclusive lock). The locked records can be unlocked by calling the `DEQUEUE_<lockobjectname>` function module. The key values and the lock mode used to set the lock must be passed to the `DEQUEUE_<lockobjectname>` function module.

Creating Lock Objects

Perform the following steps to create a Lock Object:

1. Select the `Lock Object` radio button from the initial screen of ABAP Dictionary, as shown in Figure 5.119:
2. Enter the name of the Lock Object to be created, such as `EZLOCKOBJ` (see Figure 5.119).

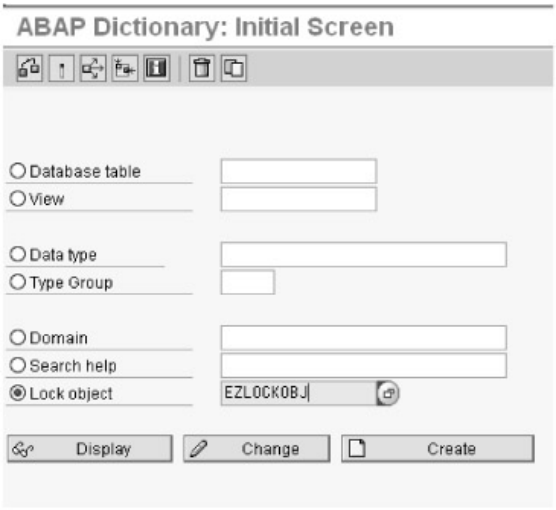


Figure 5.119: The initial screen of ABAP dictionary

Note The name of the Lock Object always starts with the letters EZ.

3. Now, click the `Create` button.

The `Dictionary: Maintain Lock Object` screen appears, as shown in Figure 5.120:

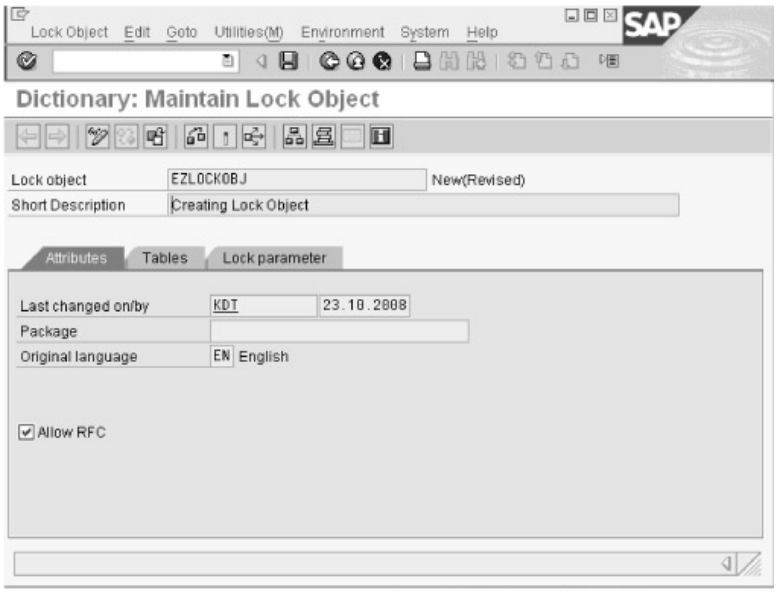


Figure 5.120: The maintenance screen for the lock object

4. Enter Creating Lock Object in the `Short Description` field and check the `Allow RFC` check box (see Figure 5.120).

Note RFC stands for Remote Function Call. When this check box is selected, the generated function modules can be called from a remote client.

5. Select the `Tables` (Tables) tab. The screen containing the options related to the `Tables` tab appears (see Figure 5.121).

6. Enter the name of the table to be locked in the `Name` field. Now, enter `ZSHIVAM` in the `Name` field. Select the `Exclusive, not cumulative` option in the `Lock Mode` field, as shown in Figure 5.121:

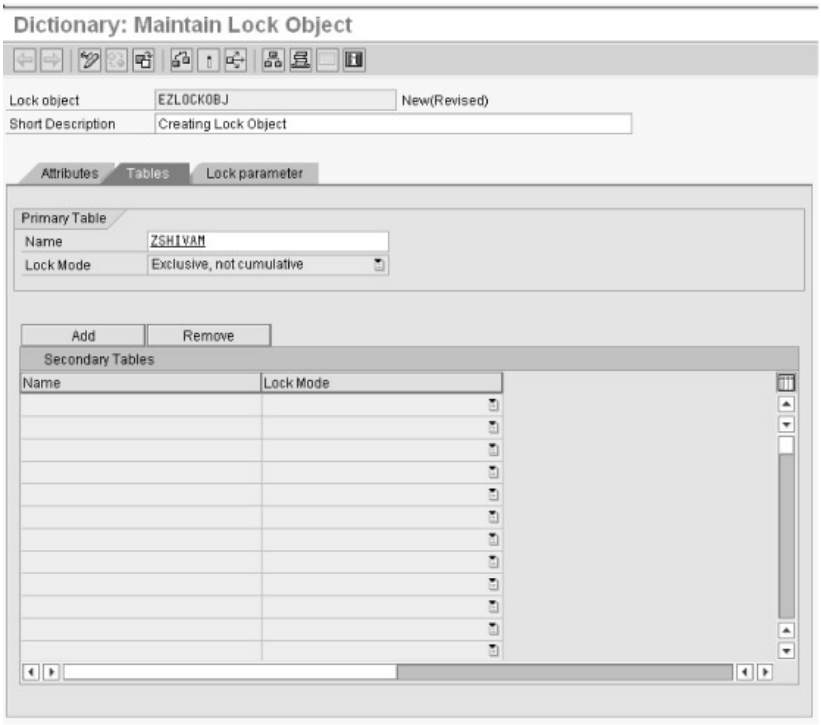


Figure 5.121: Fields for the tables tab

Note You can also lock records in more than one table. For this, click the `Add` (Add) button. A list of all the tables that are linked with the primary table specified in the `Name` field is displayed. Select the table whose records you want to lock. The primary table lock mode is copied as a lock mode. You can change this setting as required; for example, you can assign the lock mode separately for each table. Similarly, you can add a table linked with an added secondary table. To do this, place the cursor on the names of the secondary tables and click the `Add` button.

7. Click the `Save` icon to save the Lock Object in the `ZKOG_PCKG package`.

8. Select the `Lock parameter` (Lock parameter) tab. The screen containing the options related to the `Lock parameter` tab appears (see Figure 5.122). On this screen, you can set whether or not you want to include the lock parameters in the generated function module. This can be done by selecting the `W (flag)` check box. If this check box is selected, two parameters, `S` and `X_S`, are inserted in the lock parameter. These two parameters are used to copy the keys to be locked or to control the lock behavior when the initial value of the key field is passed. If you do not want the lock parameter in the generated function module, you can deselect the `W (flag)` check box. Figure 5.122 displays the `Lock paramater tab` page:

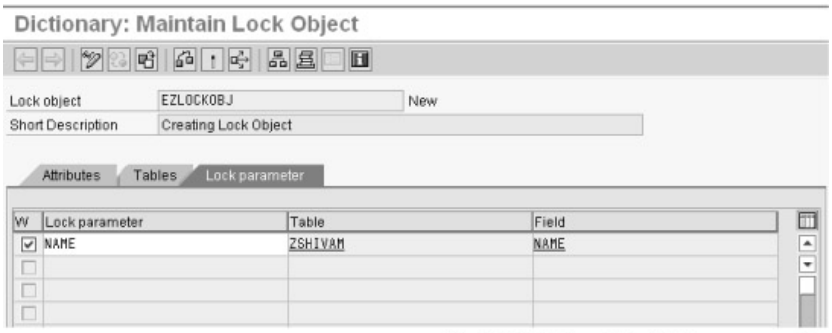


Figure 5.122: Lock parameter

In Figure 5.122, note that the lock paramater appears by default. The lock parameter is the key field of the `ZSHIVAM` table,

that is, NAME. In addition to this, the W (flag) check box is already checked by default. This flag enables the lock parameters to be specified in the generated function modules. In this case, it is left unchanged.

9. Click the Activate icon to activate the lock. When the activation process is complete, two function modules, ENQUEUE_<lockobjectname> and DEQUEUE_<lockobject>, are generated from the definition of the Lock Object.

10. Call the generated function modules to set and release the Lock Objects. In this case, we call the function module 'ENQUEUE_EZLOCKOBJ' by either clicking the Pattern button on the application toolbar of the ABAP Editor screen or pressing the Ctrl+F6 key. Figure 5.123 displays the function module together with different parameters:

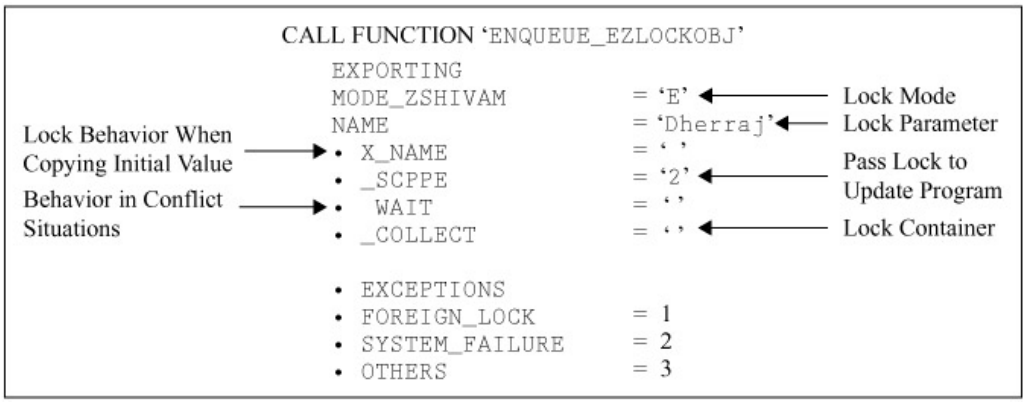


Figure 5.123: Function module for setting the lock

When the ENQUEUE_EZLOCKOBJ function module is called, the name Dherraj is locked exclusively in the ZSHIVAM table. This lock is sent to the update program (_SCOPE = '2'). If there is a lock conflict, another attempt is made to set the lock after a certain period of time (specified by _WAIT = 'X').

You can remove a lock by calling the DEQUEUE_EZLOCKOBJ function module. Figure 5.124 displays the DEQUEUE_EZLOCKOBJ function module with different parameters:

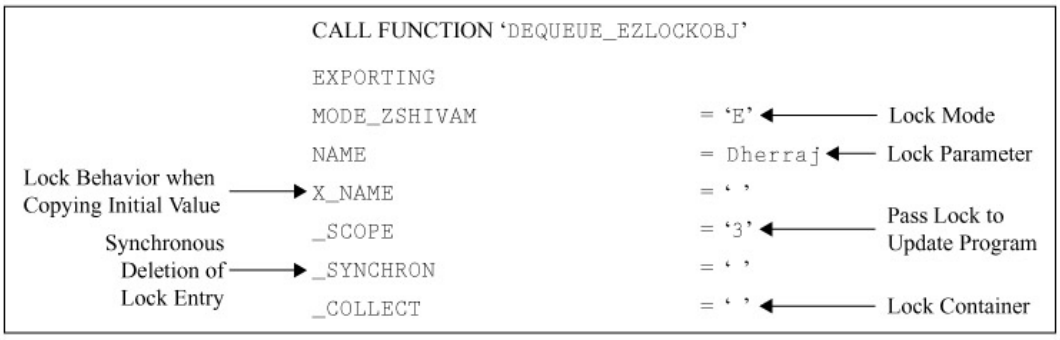


Figure 5.124: Releasing a function module

The existing exclusive lock entry for the Name field, that is, Dherraj, is deleted from the ZSHIVAM table. The request to delete the lock entries is passed to the update program (_SCOPE = '3').

Summary

In this chapter, you have learned about various repository objects of ABAP Dictionary, such as Domain, Data Type, Type Group, Database Table, Views, Search Help, and Lock Objects. Apart from creating and maintaining these objects, you have learned how to perform various types of functions on the data stored in an SAP system with these objects. This chapter has also discussed how to store these objects in a package to make them transferrable among multiple SAP servers.