

ABAP Data Dictionary

BY

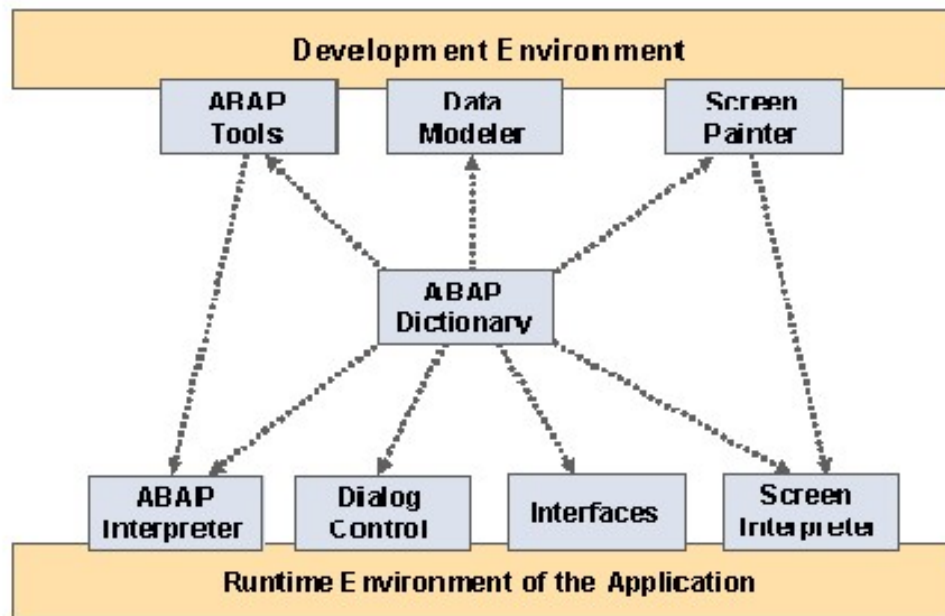
TAPAS KUMAR CHOUDHURY

AGENDA

- Introduction to ABAP Dictionary
- Domain
- Data element
- Table
- Structure
- View
- Lock objects

Introduction to ABAP Dictionary

- The ABAP dictionary (SE11) centrally describes and manages all the data definitions used in the system



Purpose

- ABAP Dictionary to create and manage data definitions (metadata). The ABAP Dictionary permits a central description of all the data used in the system without redundancies. New or modified information is automatically provided for all the system components. This ensures data integrity, data consistency and data security.
- The ABAP Dictionary supports the definition of user-defined types (data elements, structures and table types). You can create the corresponding objects (tables or views) in the underlying relational database using these data definitions.
- The ABAP Dictionary describes the logical structure of the objects used in application development and shows how they are mapped to the underlying relational database in tables or views.
- The ABAP Dictionary also provides standard functions for editing fields on the screen, for example for assigning input help to a screen field.

Integration

- The ABAP Dictionary is completely integrated in the ABAP Workbench. The SAP system works interpretatively, permitting the ABAP Dictionary to be actively integrated in the development environment. Instead of the original objects, the interpreters see only internal representations of these objects.
- These internal representations are adjusted automatically when the system finds that changes have been made in the ABAP Dictionary. This ensures that the screen and ABAP interpreters, input help, database interface, and development tools always access current data

CONT...

- When you work on development projects, objects of the ABAP Dictionary can be changed any number of times before being activated and made available to the operative components of the system. Objects can have both an active and an inactive version in the ABAP Dictionary at the same time.
- Inactive ABAP Dictionary objects have no effect on the runtime system (ABAP processor, database interface). This permits greater changes to several objects without impairing the executability of the system. The objects can be activated together only when they all have been changed.

Features

- The most important object types in the ABAP Dictionary are:
- Tables Tables are defined in the ABAP Dictionary independently of the database. From this table definition follows the creation of a table with the same structure in the underlying database.
- Views Views are logical views of more than one table. The structure of the view is defined in the ABAP Dictionary. A view of the database can then be created from this structure.
- Types The structure of a type can be defined globally in ABAP programs. Changes to a type automatically take effect in all the programs using the type.
- Lock objects These objects are used to synchronize access to the same data by more than one user. Function modules that can be used in application programs are generated from the definition of a lock object in the ABAP Dictionary.
- Domains Different fields having the same technical type can be combined in domains. A domain defines the value range of all table fields and structure components that refer to this domain.

Domain

- A domain defines a value range.
- A domain is assigned to a data element.
- All table fields or structure components that use this data element have the value range defined by the domain.
- The relationship between the field or component and the domain is defined by the data element of the field or component.
- When you change the domain, the system automatically changes fields or components that refer to this domain (with the assigned data elements).
- This ensures that the value ranges of these fields or components are consistent. Fields or components that are technically the same can be combined with a reference to the same domain.
- Only possible to define fixed values for domains of data types CHAR, NUMC, DEC, INT1, INT2 and INT4

Data type and length of fields
are always consistent

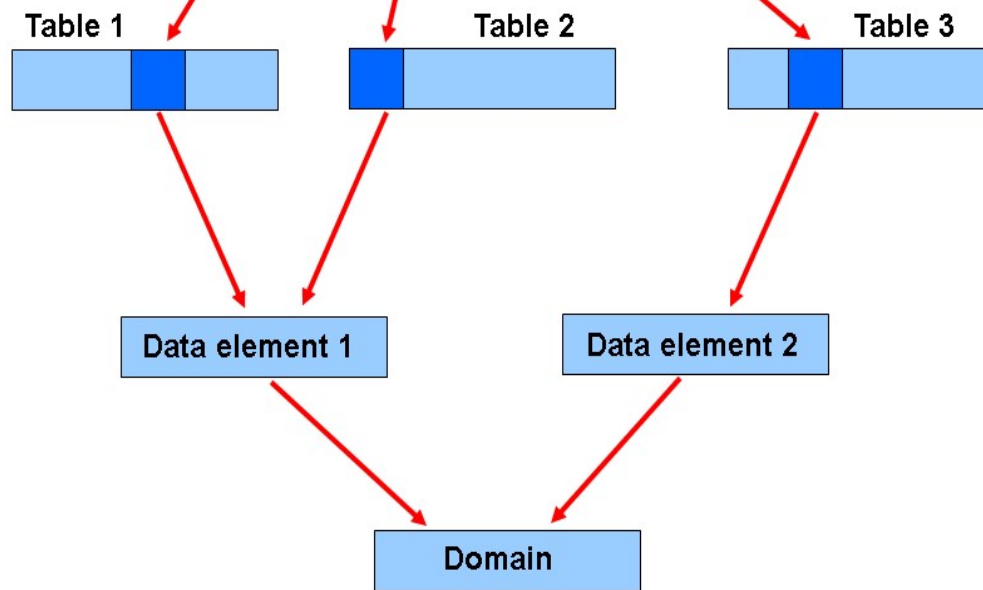


Table SBOOK

.....	FORCURAM
-------	----------	-------

Table SFLIGHT

.....	PRICE
-------	-------	-------	-------

Data Element
S_F_CUR_PR


Data Element
S_PRICE
















Domain
S_PRICE

Data Type	Description	Remarks
CLNT	Client	Client uses 3 places
DEC	Counter or amount field with decimal point, sign, and commas	Maximum length of 31 places
CURR	Currency field	Maximum length of 31 places
DATS	Date	YYYYMMDD stored as char with 8 places
TIMS	Time	HHMMSS stored as char with 6 places
CHAR	Character	Maximum length of 255. Use type LCHR for longer length
INT4	4-byte integer	Value from –2177483647 and 2177483647








Creating Domains

- Before you create a new domain, check whether any existing domains have the same technical specifications required in your table field. If so, we are supposed to use that existing domain. Let's discuss the procedure for creating the domain.
- **Step 1** – Go to Transaction SE11.
- **Step 2** – Select the radio button for Domain in the initial screen of the ABAP Dictionary, and enter the name of the domain as shown in the following screenshot. Click the CREATE button. You may create domains under the customer namespaces, and the name of the object always starts with 'Z' or 'Y'.

 **ABAP Dictionary: Initial Screen**

ABAP Dictionary: Initial Screen

☐ Database table

☐ View




☐ Data type

☐ Type Group

☒ Domain

☐ Search help

☐ Lock object

 Display  Change  Create

CONT...

- **Step 3** – Enter the description in the short text field of the maintenance screen of the domain. In this case, it is “Customer Domain”. **Note** – You cannot enter any other attribute until you have entered this attribute.
- **Step 4** – Enter the Data Type, No. of Characters, and Decimal Places in the Format block of the Definition tab. Press the key on Output Length and it proposes and displays the output length. If you overwrite the proposed output length, you may see a warning while activating the domain. You may fill in the Convers. Routine, Sign and Lower Case fields if required. But these are always optional attributes.
- **Step 5** – Select the Value Range tab. If the domain is restricted to having only fixed values then enter the fixed values or intervals. Define the value table if the system has to propose this table as a check table while defining a foreign key for the fields referring to this domain. But all these are optional attributes.

Dictionary: Change Domain



Domain New(Revised)

Short Description

Properties

Definition

Value Range

Format

Data Type Character string with only digits

No. Characters

Decimal Places

Output Characteristics

Output Length

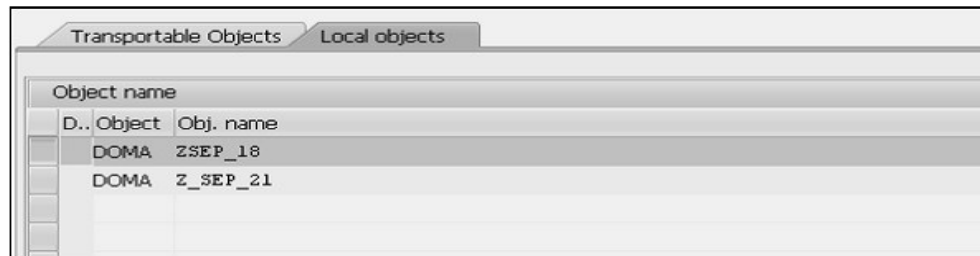
Convers. Routine

☐ Sign

☐ Lower Case

ONT...

- **Step 6** – Save your changes. The Create Object Directory Entry pop-up appears and asks for a package. You may enter the package name in which you are working. If you do not have any package then you may create it in the Object Navigator or you can save your domain using the Local Object button.
- **Step 7** – Activate your domain. Click on the Activate icon (matchstick icon) or press CTRL + F3 to activate the domain. A pop-up window appears, listing the 2 currently inactive objects as shown in the following snapshot –



The screenshot shows a window with two tabs: 'Transportable Objects' and 'Local objects'. The 'Local objects' tab is active. Below the tabs is a table with the following data:

Object name	
D.. Object	Obj. name
DOMA	ZSEP_18
DOMA	Z_SEP_21

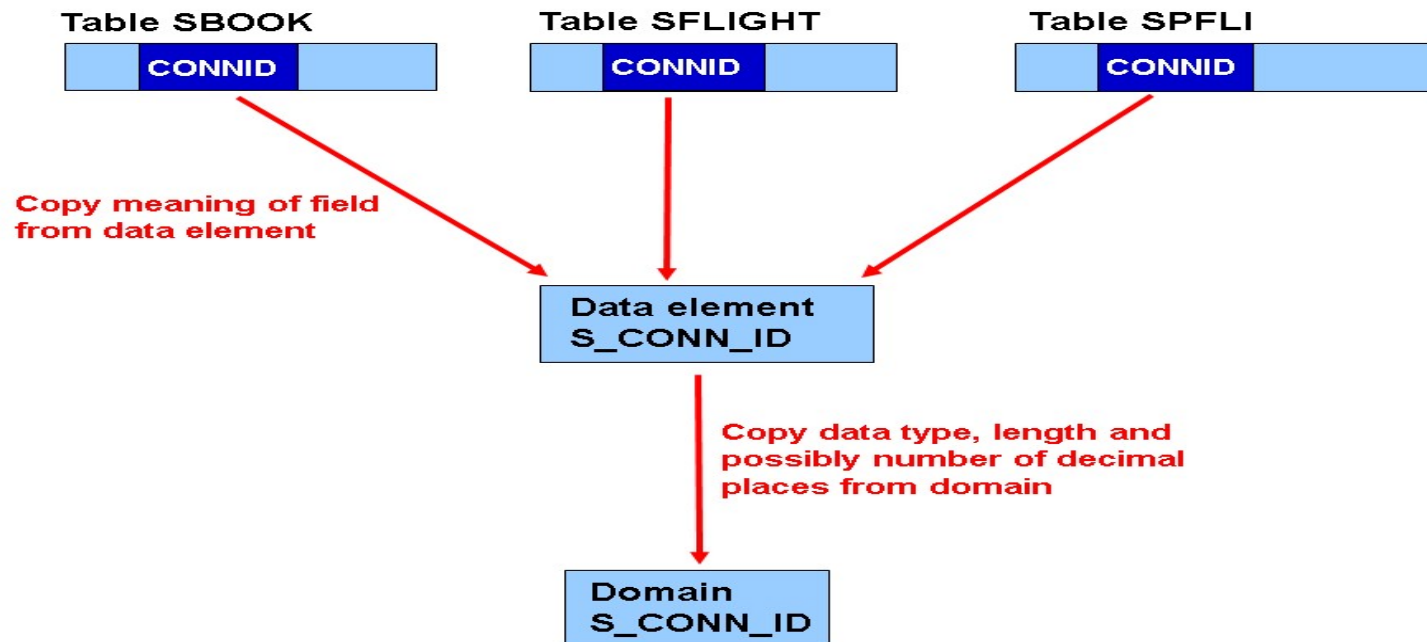
- **Step 8** – At this point, the top entry labeled 'DOMA' with the name ZSEP_18 is to be activated. As this is highlighted, click the green tick button. This window disappears and the status bar will display the message 'Object activated'.

Data Element

- Data elements describe the individual fields in the ABAP Data Dictionary. They are the smallest indivisible units of the complex types, and they are used to define the type of table field, structure component or row type of a table.
- Information about the meaning of a table field and also information about editing the corresponding screen field could be assigned to a data element. This information is automatically available to all the screen fields that refer to the data element.
- A data element consists of the **description, data type, and length**. In ABAP, there are two types of data elements:
- **Elementary type:** Elementary type refers to the data type that contains semantic attributes, such as data type, length, and the number of decimal places. These data types can be assigned in two ways to a data element:
 - 1.By assigning a predefined ABAP Dictionary type:** A data element can be assigned using the predefined types such as CHAR, NUMC, etc.
 - 2.By assigning a domain:** It specifies that a data element inherits the properties or technical attributes of the predefined domain. A domain can be used with multiple numbers of data elements.
- **Reference Type:**
The reference type describes the single field that has references for global classes and interfaces with the multiple numbers of data elements.

- The field CONNID (flight class) of table SBOOK refers to the S_CONN_ID data element. This data element gets its technical attributes (data type NUMC, field length 4) from domain S_CONN_ID. Data element S_CONN_ID describes the technical attributes and meaning (with an assigned long text and an explanatory short text) of field CONNID (and all other fields that refer to this data element).
- A variable with the type of data element S_CONN_ID can be defined in an ABAP program with the statement:

```
DATA CONNID TYPE S_CONN_ID
```



Creating Data Elements

- Before creating a new data element, you need to check whether any existing data elements have the same semantic specifications required in your table field. If so, you may use that existing data element. You can assign the data element with a predefined type, domain, or reference type.
- Following is the procedure for creating the data element –
- **Step 1** – Go to Transaction SE11.
- **Step 2** – Select the radio button for Data type in the initial screen of the ABAP Dictionary, and enter the name of the data element as shown below.
- **Step 3** – Click the CREATE button. You may create data elements under the customer namespaces, and the name of the object always starts with 'Z' or 'Y'.
- **Step 4** – Check the Data element radio button on the CREATE TYPE pop-up that appears with three radio buttons.
- **Step 5** – Click the green checkmark icon. You are directed to the maintenance screen of the data element.
- **Step 6** – Enter the description in the short text field of the maintenance screen of the data element. In this case, it is "Customer Data Element". **Note** – You cannot enter any other attribute until you have entered this attribute

CONT...

- **Step 7** – Assign the data element with the type. You can create an elementary data element by checking elementary type or a reference data element by checking Reference type. You can assign a data element to a Domain or Predefined Type within Elementary Type and with Name of Reference Type or Reference to Predefined Type within Reference Type.
- **Step 8** – Enter the fields for short text, medium text, long text, and heading in the Field Label tab. You can press Enter and the length is automatically generated for these labels.
- **Step 9** – Save your changes. The Create Object Directory Entry pop-up appears and asks for a package. You may enter the package name in which you are working. If you do not have any package then you may create it in the Object Navigator or you can save your data element using the Local Object button.
- **Step 10** – Activate your data element. Click the Activate icon (matchstick icon) or press CTRL + F3 to activate the data element. A pop-up window appears, listing the 2 currently inactive objects as shown in the following screenshot.
- **Step 11** – At this point, the top entry labeled 'DTEL' with the name Z_CUST is to be activated. As this is highlighted, click the green tick button. This window disappears and the status bar will display the message 'Object activated'.
- **Step 11** – At this point, the top entry labeled 'DTEL' with the name Z_CUST is to be activated. As this is highlighted, click the green tick button. This window disappears and the status bar will display the message 'Object activated'.

ABAP Dictionary: Initial Screen



☐ Database table

☐ View

☒ Data type

☐ Type Group

☐ Domain

☐ Search help

☐ Lock object



Display



Change



Create

Create Type Z_CUST

☒ Data element
☐ Structure
☐ Table type

☐ ☐

Dictionary: Change Data Element

← → | | Documentation Supplementary D

Data element New(Revised)

Short Description

Attributes Data Type Further Characteristics Field Label


☒ Elementary Type

☒ Domain Customer Domain

Data Type NUMC Character string with only digits

Length 8

Dictionary: Change Data Element


[Documentation](#)
[Supplementary](#)

Data element New(Revised)

Short Description

Attributes Data Type Further Characteristics **Field Label**

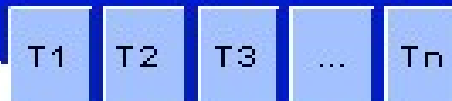
	Length	Field Label
Short	<input type="text" value="10"/>	<input type="text" value="Customer M"/>
Medium	<input type="text" value="15"/>	<input type="text" value="Customer Number"/>
Long	<input type="text" value="20"/>	<input type="text" value="Customer Number"/>
Heading	<input type="text" value="15"/>	<input type="text" value="Customer Number"/>

Transportable Objects		Local objects	
Object name			
	D.. Object	Obj. name	
	DTEL	Z_CUST	
	DOMA	Z_SEP_21	

Tables

- A database table is an object type in DDIC that can be defined independently of a database in the ABAP dictionary, which stores the actual data in the matrix form of **rows** and **columns**.
- The name of the custom table in DDIC must start with a letter z or y and can have a maximum of 16 characters. The table name can consist of any **letter, number, or underscore**.
- Each row of the table is called a **record**, and each column of the table is called the **field**.
- The field of the table is the combination of the **data elements** and **domain** and can be defined with their data type and length.
- When this table is activated in DDIC, the SAP system creates a physical copy of its field in the database also. The table is automatically translated into the format that is compatible with the actual database. We can see the relationship between the table and the actual database in the below figure:

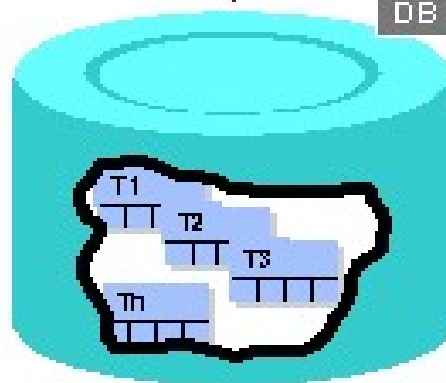
Database-independent Definition of the Tables in the ABAP Dictionary



Activation
program and
DB UTILITY



DB



Definition of the tables
in the database

CONT...

- The following is a list of a few database tables that are commonly used in programs:

Table	Description
MARA	General material data
MARC	Plant data for material
MARD	Storage location data for material
MBEW	Material valuation
MKPF	Material document header
MSEG	Material document segment
KNA1	General customer data
KNB1	Company-specific customer data
LFA1	General vendor data
LFB1	Company-specific vendor data
BKPF	Accounting document header
BSEG	Accounting document segment

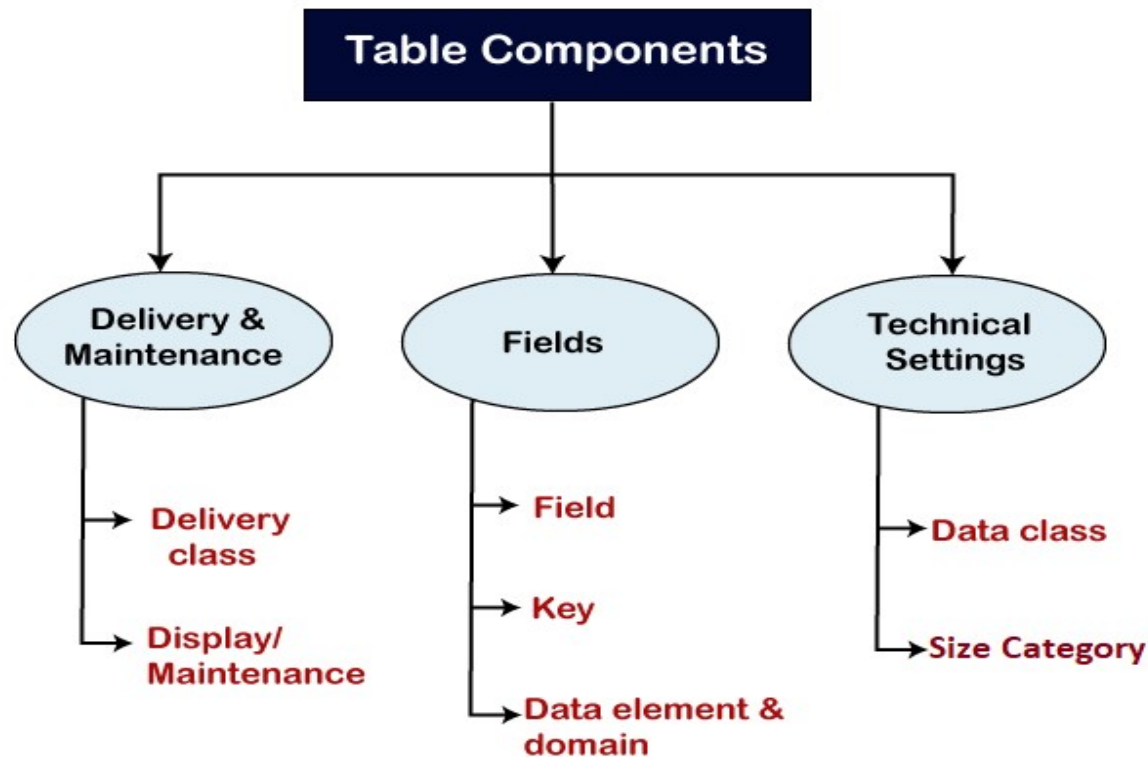
More common tables

<u>Table</u>	<u>Description</u>
--------------	--------------------

VBAK	Sales order header
VBAP	Sales order items
LIKP	Delivery order header
LIPS	Delivery order items
VBRK	Billing header
VBRP	Billing items
VBFA	SD document flow
EKKO	Purchasing document header
EKPO	Purchasing document items
MAST	Material to BOM link
STKO	BOM header
STPO	BOM items

Components of Table in ABAP DDIC

- The components of the ABAP table are given below:



CONT...

- **Delivery Class:**
- Delivery class is used to define the type of data that we want to store in our table. The data can be business data, which means Application data (Master and transaction data). We will choose the application data for our delivery class.
- **Display/Maintenance:**
- This option is used to allow us that whether we want only display the data or also want to maintain the data. Here the data maintenance means creating, deleting, or updating the data. There are three options available for this field:
 - **Display/maintenance allowed**
 - **Display/maintenance not allowed**
 - **Display/maintenance allowed with restrictions**
- **Field Name:** The name which we provide to each field in the table is known as the field name that consists of a maximum of 16 characters. Similar to the table name, it must start with a letter, and can have a letter, numbers, and underscore.
- **Key Flag:** It specifies whether that particular field belongs to the Key field or not.

CONT...

- **Field Length:** It defines the number of characters that can be entered into the field.
- **Decimal Places:** It specifies how many digits are permitted after the decimal point, and it can only be used for the numeric data.
- **Short text:** It specifies the meaning of the particular field entered within the table.
- **Data class:** It defines the actual area of the table inside the database. The options available for the data class are given below, which we can select according to our table:

Data Class	Description
APPL0	Master Data, Transparent Tables
APPL1	Transaction Data, Transparent Tables
APPL2	Organizing & Customizing

- **Domain:** The domain is an object that defines the **technical information** of the field, such as data type and length.
- **Data Element:** The data element is an object that specifies **the semantic information** of the field, such as description, labels, etc.

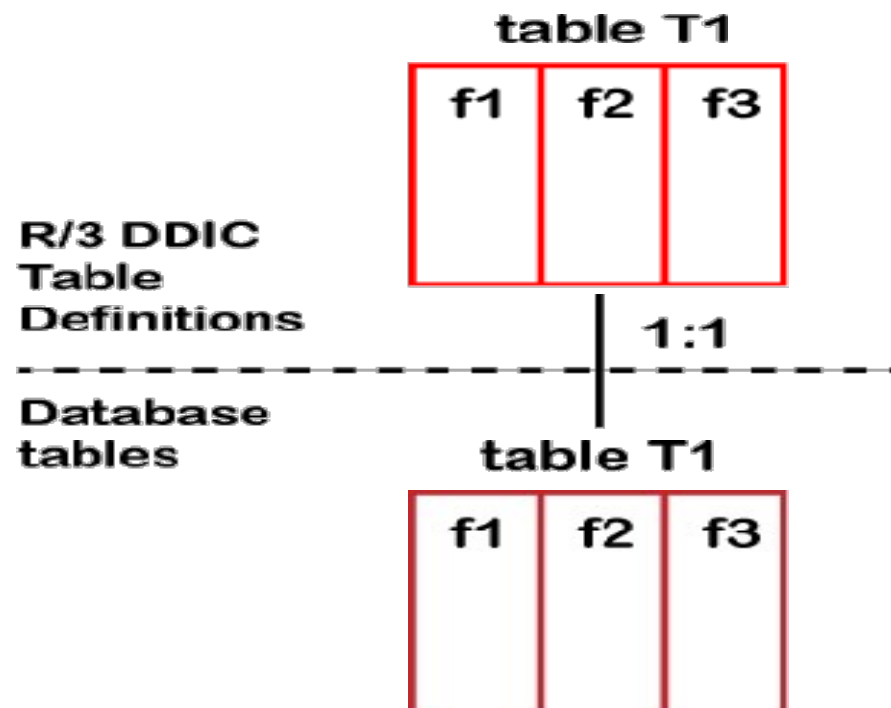
Types of ABAP Tables

- In the ABAP dictionary, we can create three types of tables:
 - **Transparent Tables**
 - **Pooled Tables**
 - **Cluster Tables**
- We can also create some special table types apart from these three tables, such as table pools and clusters. These two table types store the information of other tables such as program parameters, temporary data, documentation text, etc.

1. Transparent Table

- ABAP transparent table contains the application data that represents the **master data** or **transaction data** used by an application. A **table of vendors** or **table of customers** is an example of the master data.
- In transparent tables, the database table has the same name and the same number of fields with field names as the data dictionary table.
- The transparent table shows the **one-to-one relationship** with the table definition in the SAP database. The below image is showing the transparent table in ABAP:

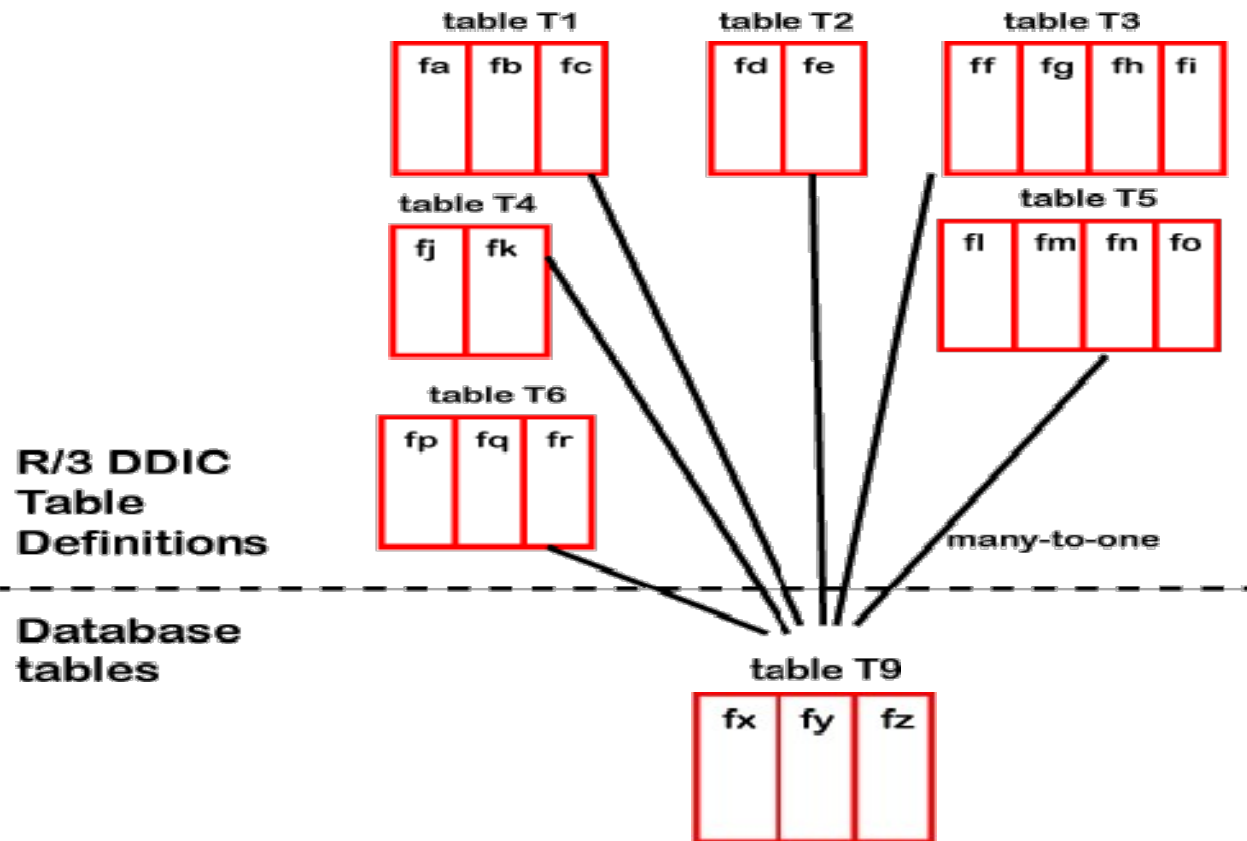
Transparent Table



2. Pooled Tables

- The pooled table in ABAP shows the **many-to-one** relationship with the table definition in the database, which means for a single table defined in the SAP database, there are various tables in the ABAP dictionary.
- The names of the tables stored in the Dictionary and the database must be different.
- The SAP database stores all the Pooled tables in a single table, which is known as **a table pool**.
- The pooled tables in the data dictionary may or may not have a common primary key field.
- Below image shows the relation between the Pooled tables and table pools:

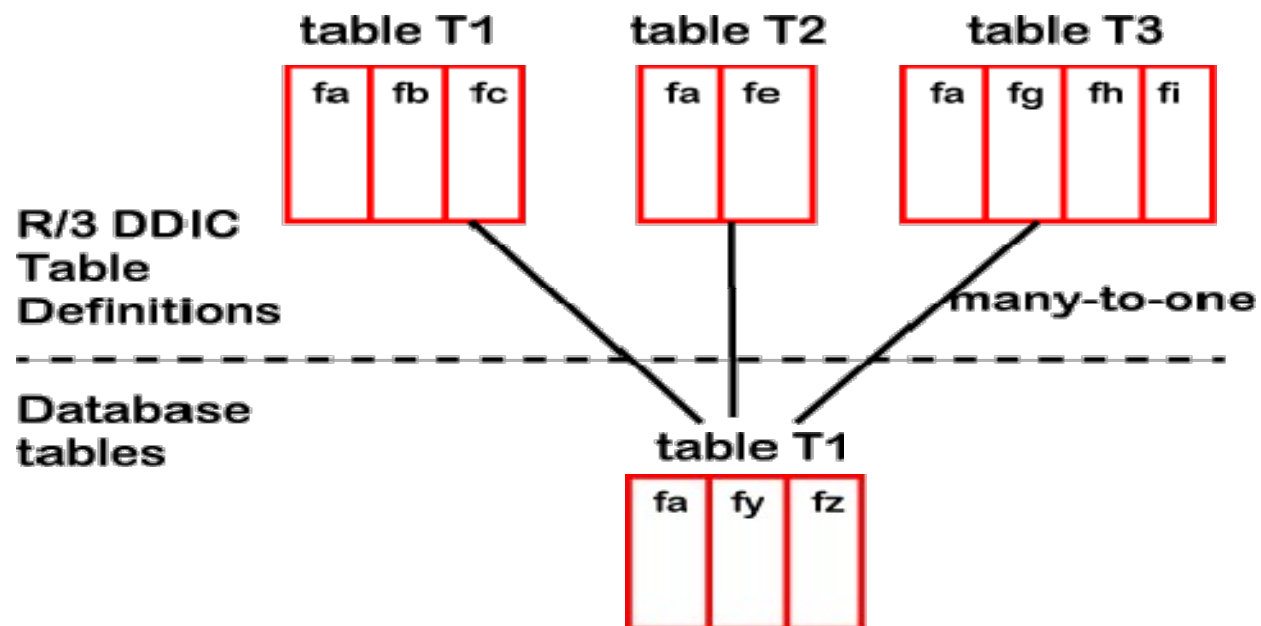
Pooled Table



3. Cluster Tables:

- Cluster tables are similar to the Pooled tables, as they also show the **many-to-one** relationship with the table definition in the SAP database. All the cluster tables are stored in a single table in the database, and that table is known as **a table cluster**.
- The tables in the data dictionary must have **at least one primary key field in common**, and the table is usually accessed simultaneously.
- The relationship between the cluster table and table clusters can be understood by using the below diagram:

Cluster Tables



CONT...

- The key difference between the Pooled, Cluster, and Transparent tables:
- Pooled and cluster tables do not usually hold the application data as the transparent table; instead they are used to store the **system data, such as system configuration information, historical data, etc.**

Creating Tables in ABAP Dictionary

- **Step 1** – Go to transaction SE11, select the 'Database table' radio button, and enter a name for the table to be created. In our case, we have entered the name ZCUSTOMERS1. Click the Create button. The Dictionary: Maintain Table screen appears. Here the 'Delivery and Maintenance' tab is selected by default.
- **Step 2** – Enter an explanatory short text in the Short Description field.
- **Step 3** – Click the Search Help icon beside the Delivery Class field. Select 'A [Application table (master and transaction data)]' option.
- **Step 4** – Select the 'Display/Maintenance Allowed' option from the 'Data Browser/Table view Maintenance' drop-down menu. The Dictionary: Maintenance Table screen appears.

Transp. Table	ZCUSTOMERS1	New(Revised)
Short Description	Customers Table	
<div>Attributes Delivery and Maintenance Fields Entry help/check Curr</div>		
Delivery Class	A Application table (master and transaction	
Data Browser/Table View Maint.	Display/Maintenance Allowed	

CONT...

- **Step 5** – Select the Fields tab. The screen containing the options related to the Fields tab appears.
- **Step 6** – 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.
- The 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.
- **Step 7** – Select the Key column if you want the field to be a part of the table key. Let's create fields such as CLIENT, CUSTOMER, NAME, TITLE and DOB.
- **Step 8** – The first field is an important one and it identifies the client which the records are associated with. Enter 'Client' as the Field and 'MANDT' as the Data Element. The system automatically fills in the Data Type, Length, Decimals and Short Description. The 'Client' field is made a key field by checking the 'Key' box.
- **Step 9** – The next field is 'Customer'. Check the box to make it a key field and enter the new Data Element 'ZCUSTNUM'. Click the Save button.
- **Step 10** – As the Data Element 'ZCUSTNUM' doesn't yet exist, it has to be created. Doubleclick the new Data Element and the 'Create Data Element' window appears. Answer 'Yes' to this and a 'Maintain Data Element' window appears.
- **Step 11** – Enter 'Customer Number' in the Short Description area. The Elementary data type called 'Domain' should be defined for the new Data element. So enter 'ZCUSTD1', double-click it and agree to save the changes made. Choose 'Yes' to create the domain and type into the 'Short Description' box a description of the domain.

Attributes	Data Type	Further Characteristics	Field Label
<input checked="" type="radio"/> Elementary Type			
<input checked="" type="radio"/> Domain			
	<input type="text" value="ZCUSTD1"/>		Number
	Data Type	NUMC	Character string with only digits
	Length	8	
<input type="radio"/> Predefined Type			
	Data Type	<input type="text"/>	
	Length	<input type="text" value="0"/>	
<input type="radio"/> Reference Type			
<input type="radio"/> Name of Ref. Type			
	<input type="text"/>		

CONT...

- The 'Definition' tab opens automatically. The first field is 'Data Type'.
- **Step 12** – Click inside the box and select 'NUMC' type from the drop-down menu. Enter the number 8 in the 'No. of characters' field (a maximum of 8 characters) and enter 0 in 'Decimal places' area. The Output length of 8 must be selected and then press Enter. The 'NUMC' field's description must re-appear, confirming that this is a valid entry.
- **Step 13** – Click Save button and Activate the object.
- **Step 14** – Press F3 to return to the 'Maintain/Change Data Element' screen. Create four Field labels as shown in the following snapshot. After this, Save and Activate the element.

Dictionary: Change Data Element

Navigation icons: back, forward, search, etc. | Documentation | Supplementary Document

Data element: ZCUSTNUM New(Revised)

Short Description: Customer Number

Attributes

Data Type

Further Characteristics

Field Label

	Length	Field Label
Short	10	Customer C
Medium	15	Customer Number
Long	20	Customer Number
Heading	15	Customer Number

CONT...

- **Step 15** – Press the back button to return to the table maintenance screen. The Customer column has the correct Data Type, Length, Decimals and Short Description. This indicates the successful creation of a Data element and also the Domain used.

Dictionary: Change Table

← → | Technical Settings Indexes... A

Transp. Table New

Short Description

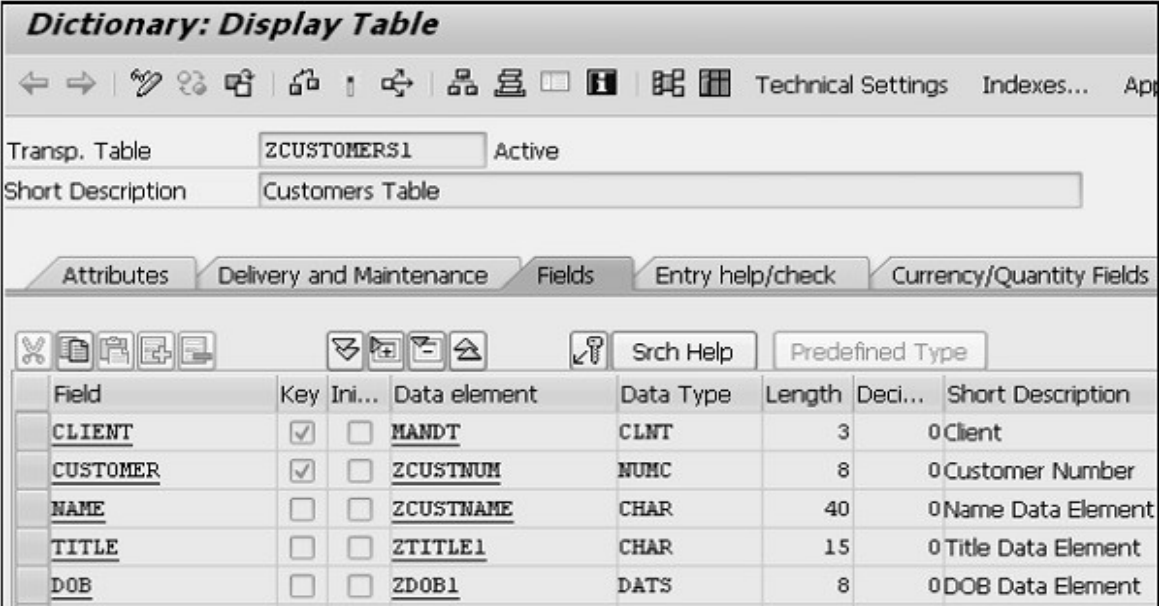
Attributes Delivery and Maintenance **Fields** Entry help/check Currency/Quantity Field:

Srch Help Predefined Type

Field	Key	Ini...	Data element	Data Type	Length	Deci...	Short Description
<u>CLIENT</u>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u>MANDT</u>	CLNT	3	0	Client
<u>CUSTOMER</u>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<u>ZCUSTNUM</u>	NUMC	8	0	Customer Number

CONT...

- Similarly, we need to create three additional fields such as NAME, TITLE and DOB.
- **Step 16** – Select 'Technical settings' from the toolbar. Choose APPL0 for the 'Data class' and the first size category 0 for the 'Size' category' field. In case of buffering options, 'Buffering not allowed' has to be selected.
- **Step 17** – Click Save. Go back to the table and Activate it. The following screen appears.
- The table 'ZCUSTOMERS1' is activated.



Dictionary: Display Table

Transp. Table: ZCUSTOMERS1 Active
Short Description: Customers Table

Attributes | Delivery and Maintenance | **Fields** | Entry help/check | Currency/Quantity Fields

Field | Key | Ini... | Data element | Data Type | Length | Deci... | Short Description

CLIENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	MANDT	CLNT	3	0	Client
CUSTOMER	<input checked="" type="checkbox"/>	<input type="checkbox"/>	ZCUSTNUM	NUMC	8	0	Customer Number
NAME	<input type="checkbox"/>	<input type="checkbox"/>	ZCUSTNAME	CHAR	40	0	Name Data Element
TITLE	<input type="checkbox"/>	<input type="checkbox"/>	ZTITLE1	CHAR	15	0	Title Data Element
DOB	<input type="checkbox"/>	<input type="checkbox"/>	ZDOB1	DATS	8	0	DOB Data Element

SAP ABAP View

- What are views in DDIC?
- Views are the data dictionary repository objects in ABAP, which are used to **view the data of several tables in one place.**
- A view is similar to the database table, but it does not contain any physical data; instead it derives data from different tables and acts like a virtual table (table without any existence). Since it does not physically store any data, and the database only contains the view definition, hence it takes very little space in the database.
- A view can be created by combining the data of single or multiple tables, which are called **base tables.**
- Before creating a view, we need to specify the structure for the tables and fields that need to be stored in the view. If we make any change in the base tables, then the changes also reflect in the views automatically.
- The views in DDIC help to save time and increase efficiency, as it is a time-consuming process to extract data from each table; for such cases, we can use view in SAP ABAP Dictionary.
- A view can be used to represent a subset of data stored within a table or to join multiple tables into a single virtual table.
- Whenever a view is executed, it displays the data extracted from the multiple tables.
- In order to create a view in DDIC, we need to join the table, and for joining the table, each table must have at least one common key field.

INNER JOIN AND OUTER JOIN

INNER JOIN

- In this join only the matching records between two or multiple tables will be selected.
- The unmatched records will not be selected.

OUTER JOIN

- In this type of join all the records from left table(1st table) will be displayed.
- If there is a matching record in the 2nd or 3rd table the data will be displayed.
- If there will be no matching record, the data will be displayed as blank value.

ZCUST_DATA

C NO	C NAME	CITY
1001	AAA	HYDB
1002	BBB	BLORE
1003	CCC	CHHENAI

ZCUST_BANKDATA

C NO	BANK ID	BANK NAME
1001	CYTY01	CYTY
1001	SBI01	SBI
1001	SBH01	SBH
1003	ICICI01	ICICI
1003	HDFC01	HDFC

Joined on ZCUST_DATA-C NO
=
ZCUST_BANKDATA-C NO

INNER JOIN OUTPUT

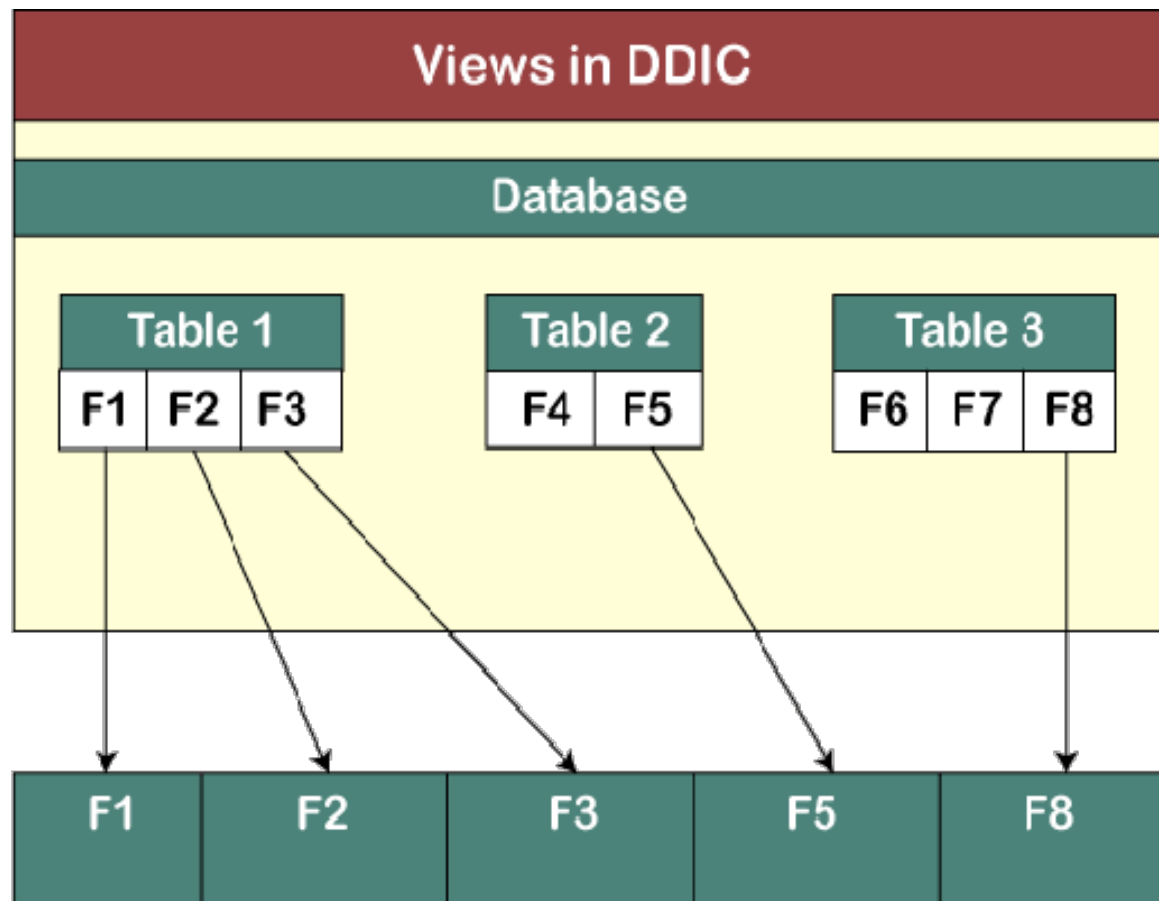
C NO	CNAME	BANK ID	BANK NAME
1001	AAA	CYTY01	CYTY
1001	AAA	SBI01	SBI
1001	AAA	SBH01	SBH
1003	CCC	ICICI01	ICICI
1003	CCC	HDFC01	HDFC

OUTER JOIN OUTPUT

C NO	CNAME	BANK ID	BANK NAME
1001	AAA	CYTY01	CYTY
1001	AAA	SBI01	SBI
1001	AAA	SBH01	SBH
1002	BBB	-	-
1003	CCC	ICICI01	ICICI
1003	CCC	HDFC01	HDFC

Key steps to define a view:

- First, we need to select the base tables to define the view.
- These base tables must be linked together using join conditions.
- Select the required fields of the base tables that need to use by the view.
- Apply some selection conditions to restrict or filter the records in the view.



Types of Views in ABAP

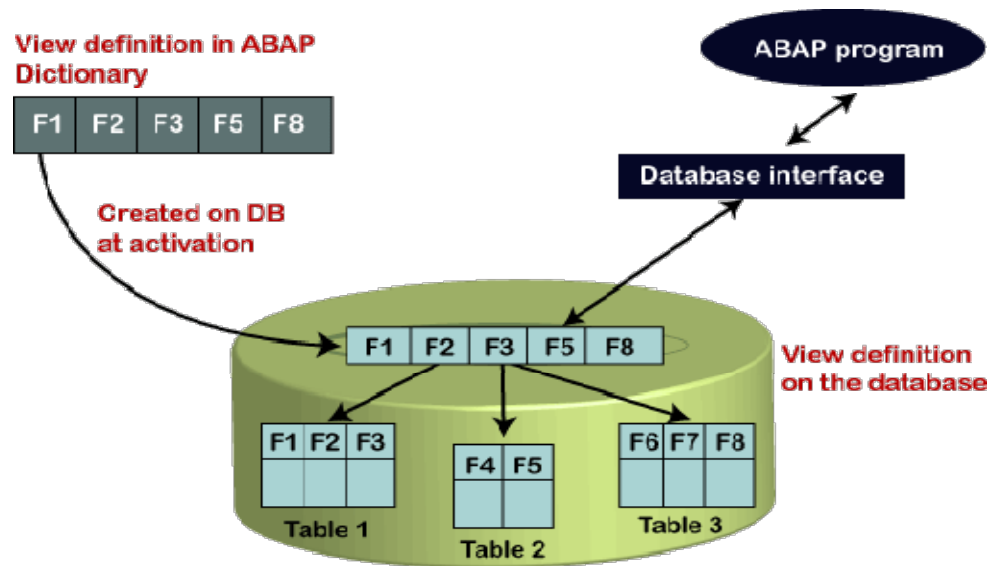
- In ABAP DDIC, there are four types of views that differ from each other in a way in which they are implemented and access the data. These are given below:
- DATABASE VIEW
- PROJECTION VIEW
- MAINTENANCE VIEW
- HELP VIEW

DATABASE VIEW

- If a view is created on one or more tables by combining the fields using **inner join**, such a view is called a Database view.
- Since this view uses the inner join, it only combines the matching records from the table.
- The database view gives an application-specific view to data of an application object, which is distributed among different tables.
- In this view, we cannot perform any maintenance operation on table data; instead we can just read the data.
- The database views are specified within the ABAP dictionary, and a database view is automatically created in the underlying database when we activate the view.
- We can also view the data of database views in an ABAP program with the help of the **database interface**. One can access the data in the program using OPEN SQL and NATIVE SQL.
- This view can include data from some fields of a table and also the entire table.

Cont...

- Note: The join conditions used in this view can be applied using the equality relationship between any base fields. But in other views type(maintenance, help, and projection view), the join condition must be achieved using the foreign key relationship.



Key features of database view

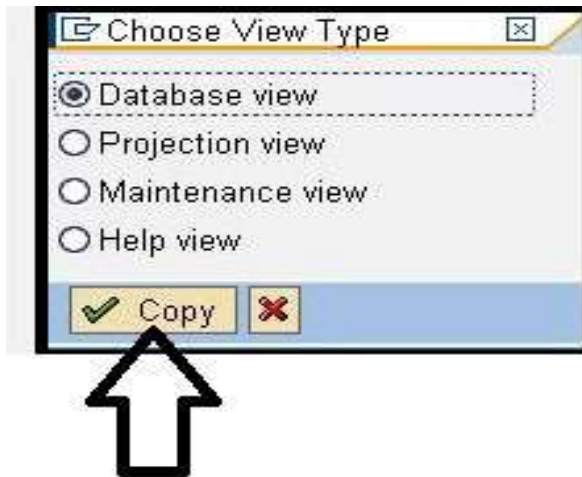
- If the view contains more than one table, then it only allows us to read the data, but if the view contains only a single table, the maintenance status can be checked to find if the data can also be inserted in the view table or not.
- If we want to select the logically connected data from different tables simultaneously, then we need to create the database views.
- This view can only combine the transparent tables, as it is implemented in the database.

Creating Database view

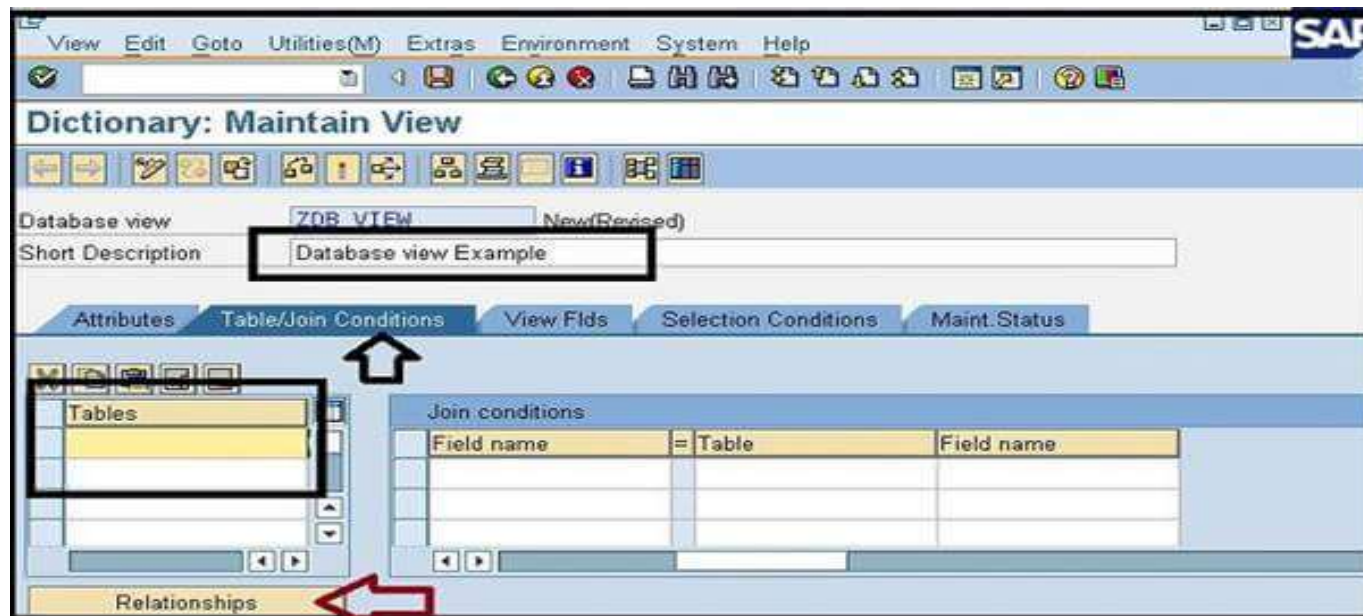
- **Step-1:** Open the data dictionary initial screen by navigating the menu path or entering the transaction code
- **Step-2:** Click the radio button in front of **View** option, and give a name to the view then click the **Create** button given on the screen. SE11 in the command field.

The screenshot shows the 'ABAP Dictionary: Initial Screen' in SAP. The menu bar includes 'Dictionary Object', 'Edit', 'Goto', 'Utilities(M)', 'Environment', 'System', and 'Help'. Below the menu is a toolbar with various icons. The main area has a section for selecting the object type, with 'Database Table' and 'View' options. The 'View' option is selected, and the name 'ZDB_VIEW' is entered in the text field next to it. Below this, there are several radio buttons for 'Data type', 'Type Group', 'Domain', 'Search help', and 'Lock object', each followed by a text field. At the bottom, there are three buttons: 'Display', 'Change', and 'Create'. A blue arrow points to the 'Create' button.

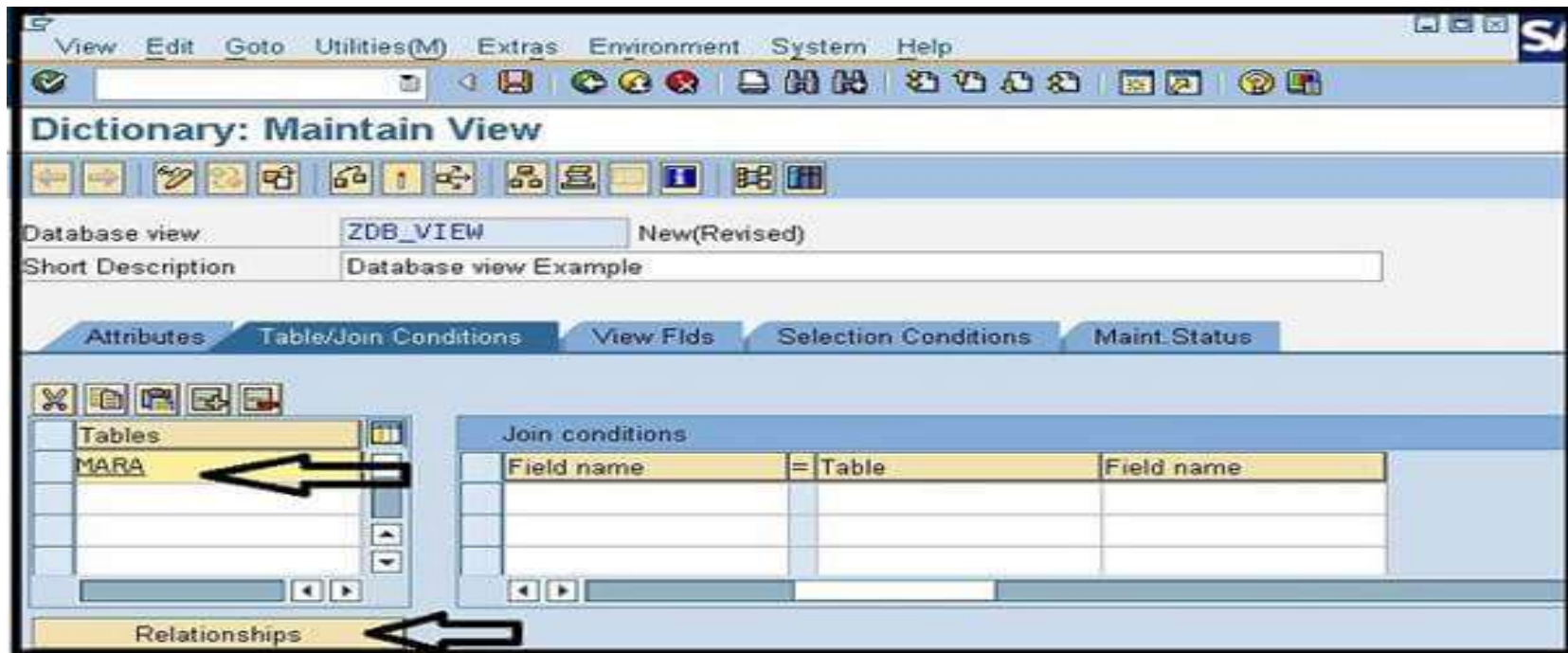
Step-4: A pop-up window will appear with all the views, from where select the "**Database view**" and click on the **Copy** button .



Step-5: A view maintains screen appears that contains multiple tabs, where first provide the description in the field "**Short description**." Consider the below image:

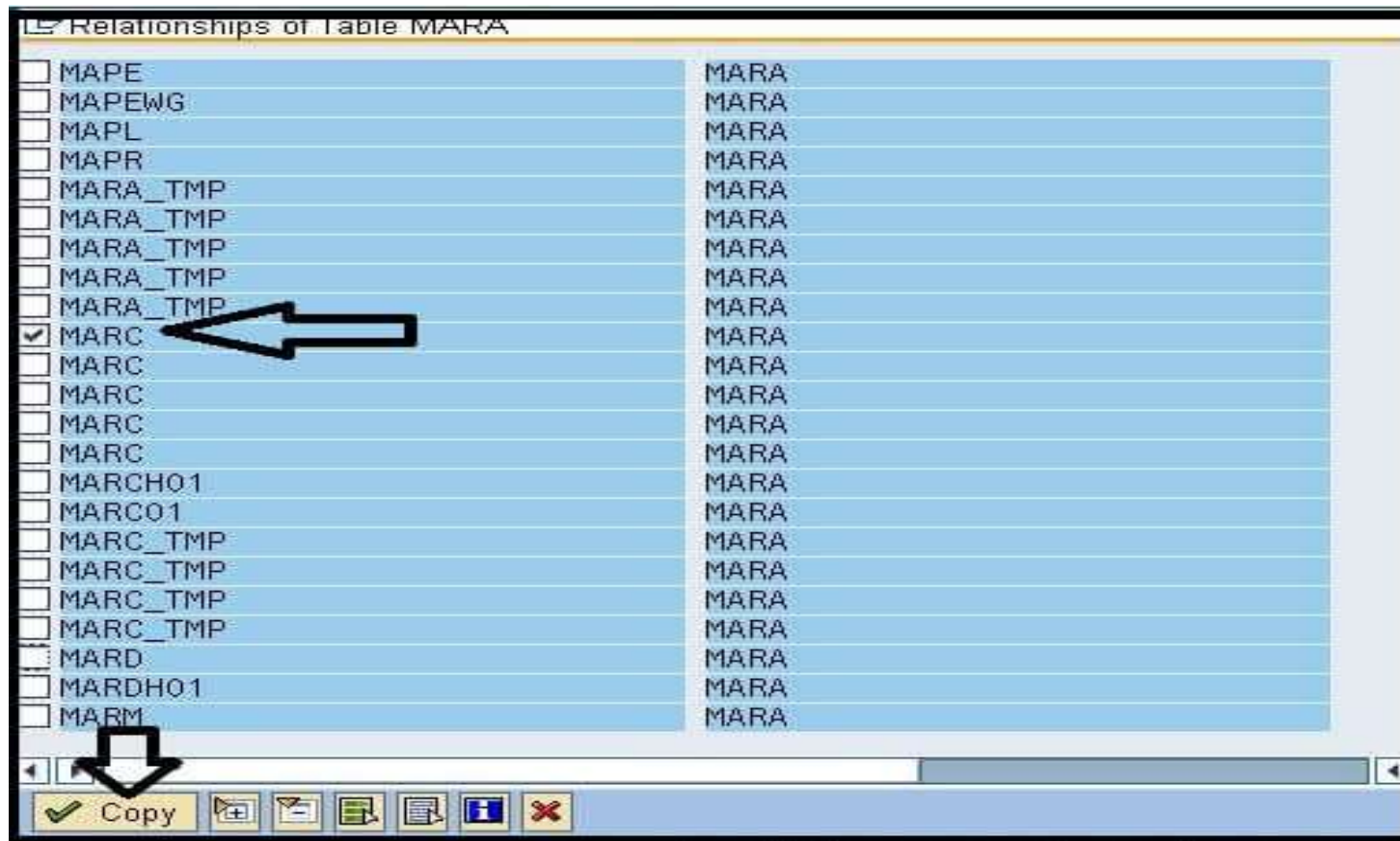


- In the above screen, first, we will select the **Table/Join Conditions** tab and will provide the table names (base tables) that we want to link. In our example, we will take two SAP standard tables that are **MARA** and **MARC**. After entering the table name, we need to click on the **Relationships** button. Consider the below image:

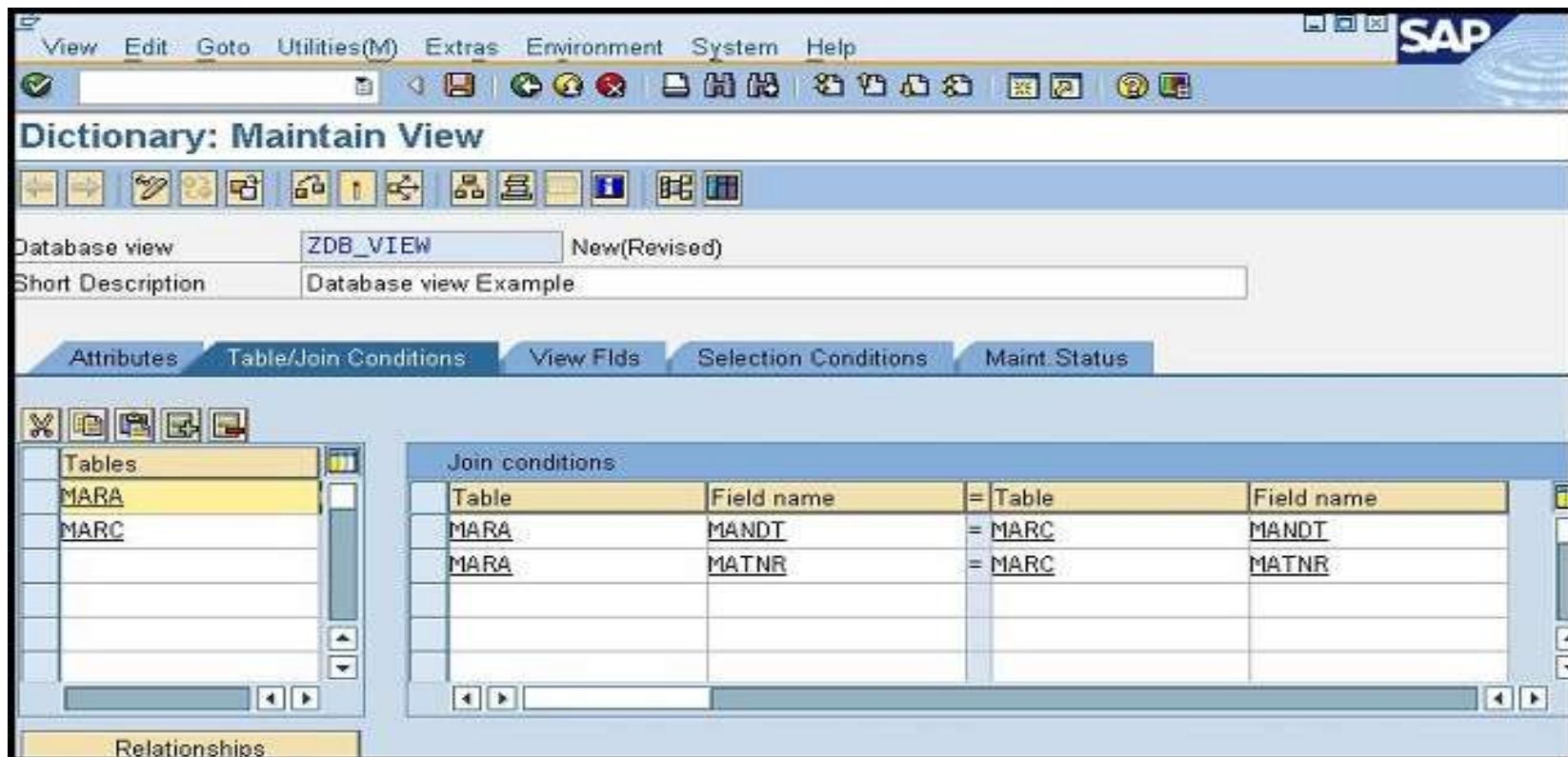


Step:7- When we click on the Relationships button, all the tables with the foreign key relationship with the base table (MARA) will be displayed. Consider the below image:

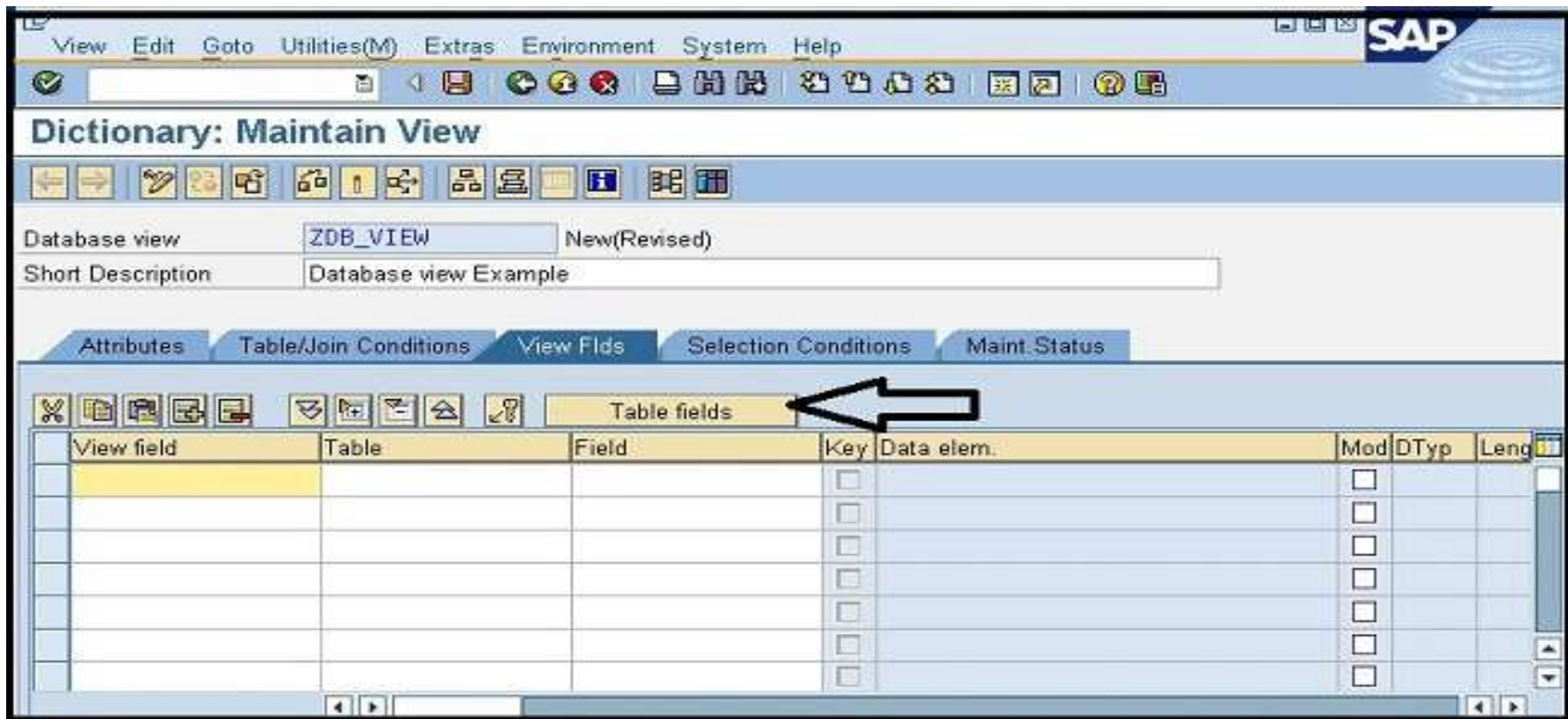
As another table, we have selected **MARC** (SAP standard table), and clicked on the Copy button, given at the bottom of the screen.



Step-8: Once we will select the MARC table, all the join conditions field names will be filled automatically. Consider the below image:



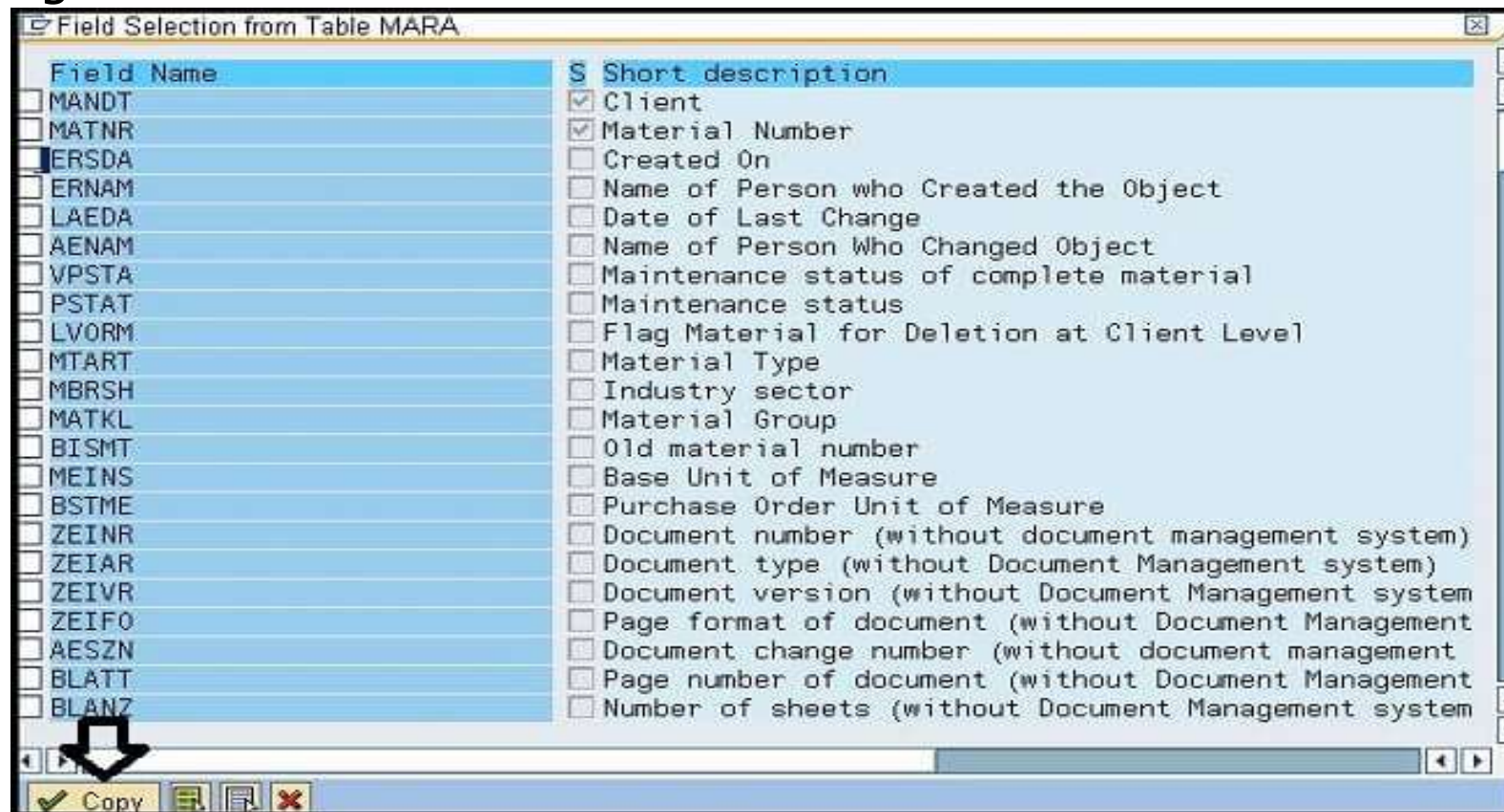
Step-9: Now, we will move to the next tab, which is **View fields**, to select the fields of both tables that we want to display. Consider the below image:



Step-10: To select the fields of each table, we need to click on the **Table fields** button given on the screen (See the above image). It will open a new pop window with both table names (MARA and MARC). We will select one table at a time and will select the required fields of each table.

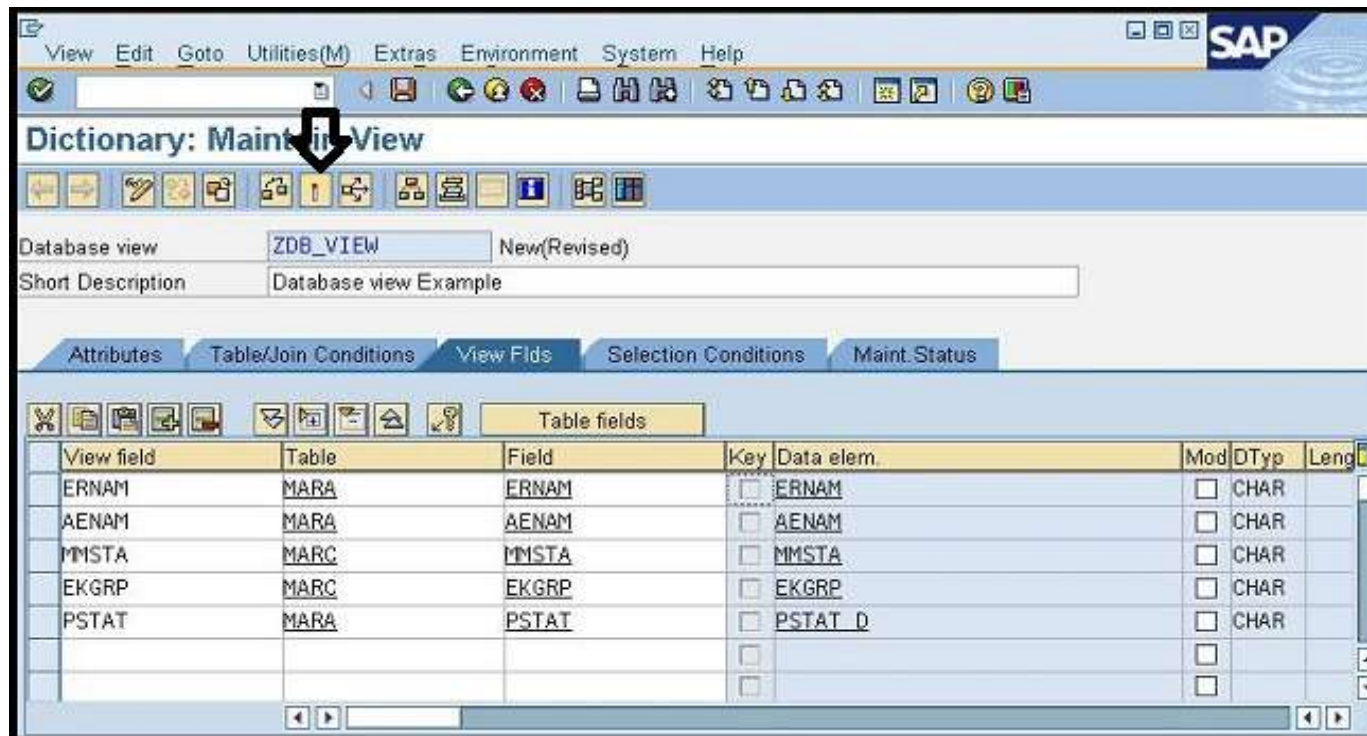


Step-11: Once we click on the **Choose** button given at the bottom of the screen, it will open all the fields of the selected table from where we can select the required fields. Consider the below image:



After selecting the fields, click on **the Copy** button. Similarly, we will choose the fields of another table (MARC).

Step-12: After selecting all the required fields of both tables, save the view as a **local object** and activate the view (**CTRL+F3**) or by clicking on **the activate** button. Consider the below image:



Step-13: The activation window will open, select the View, and click to the Green tick.



Step-14: Now, we can display the records that we have selected using the database view. For this, click on the **Content** (CTRL+SHIFT+F10) option given at the screen. Once we click on the content option, it will open the selection screen as given in the below image:

The screenshot displays the SAP Data Browser interface for Table ZDB_VIEW. The title bar reads "Data Browser: Table ZDB_VIEW: Selection Screen". Below the title bar, there is a toolbar with several icons. A black arrow points to the "Content" icon, which is the first icon in the toolbar. To the right of the toolbar is a button labeled "Number of Entries". The main area of the screen contains several selection criteria fields, each with a "to" field for range selection. The fields are: "Created by" (with a yellow background), "Changed by", "P-S matl status", "Purch. Group", and "Maint. status". Below these fields are two more fields: "Width of Output List" with the value "250" and "Maximum No. of Hits" with the value "200". On the right side of the screen, there is a vertical toolbar with five yellow buttons, each containing a right-pointing arrow.

Field	Value	to
Created by		
Changed by		
P-S matl status		
Purch. Group		
Maint. status		

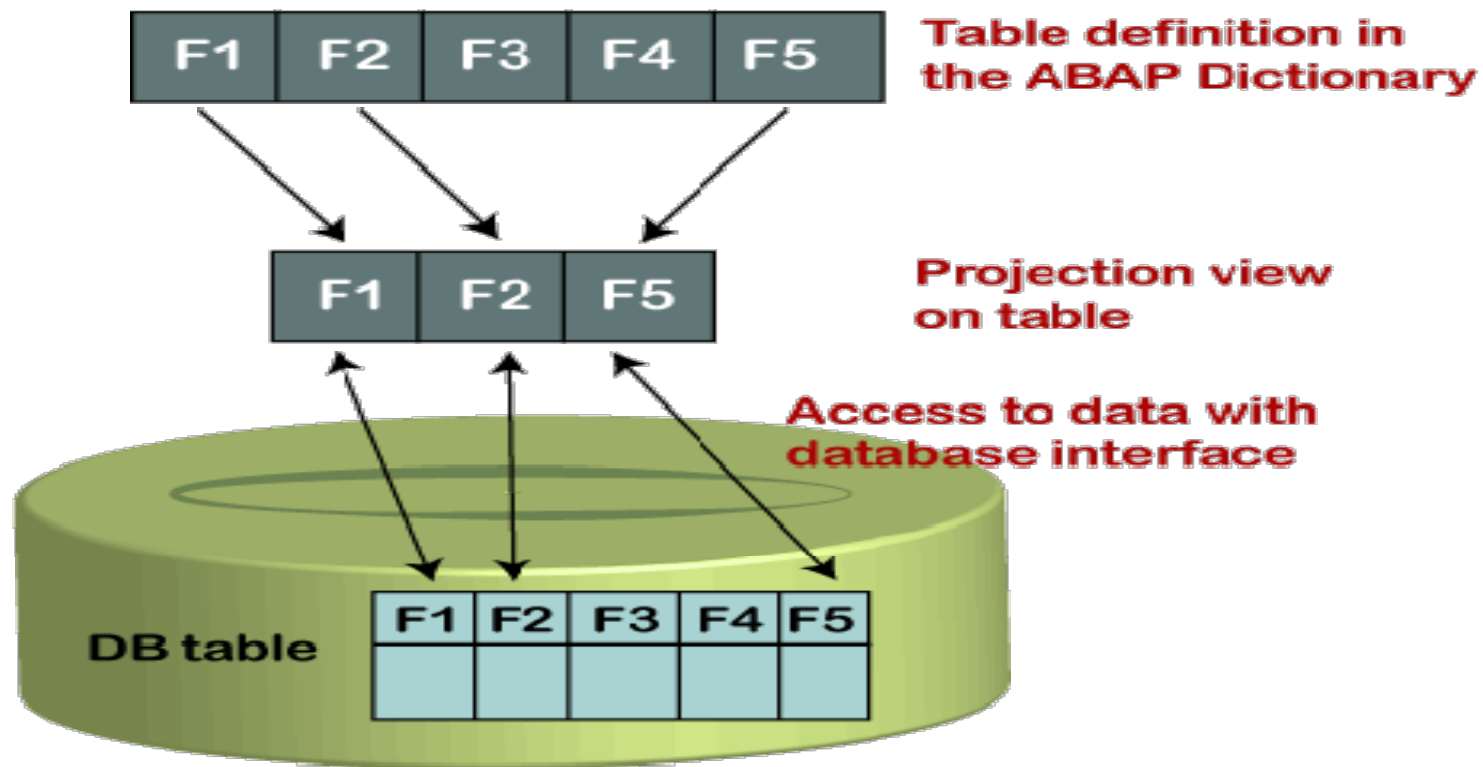
Width of Output List	250
Maximum No. of Hits	200

In the above, we can see all the fields of the table are displayed that we have selected from both the table. We can either fill the details in the field for the particular record or can view all the records using the **execute** button given at the above image. It will display all the records, consider the below image:

	Created by	Changed by	P-S matl status	Purch. Group	Maint. status
<input type="checkbox"/>	BIRNLEY			001	KVEDALBG
<input type="checkbox"/>	BIRNLEY	BIRNLEY		001	KVEDALBG0
<input type="checkbox"/>	SAHAD	SAHAD			KVEDPALSQBG
<input type="checkbox"/>	SAHAD	SAHAD			KVEDPALSQBG
<input type="checkbox"/>	SAHAD				KVEDPALSQBG
<input type="checkbox"/>	SAHAD				KVEDPALSQBG
<input type="checkbox"/>	MERTZLUFFT				LK
<input type="checkbox"/>	MICHALSKY	ID21066			S
<input type="checkbox"/>	MICHALSKY	ID21066			S
<input type="checkbox"/>	MICHALSKY				V
<input type="checkbox"/>	MICHALSKY				V
<input type="checkbox"/>	DEVENTER	LINDHOLM			VK
<input type="checkbox"/>	DEVENTER	LINDHOLM			VK
<input type="checkbox"/>	DEVENTER	LINDHOLM			VK
<input type="checkbox"/>	DEVENTER	LINDHOLM			VK

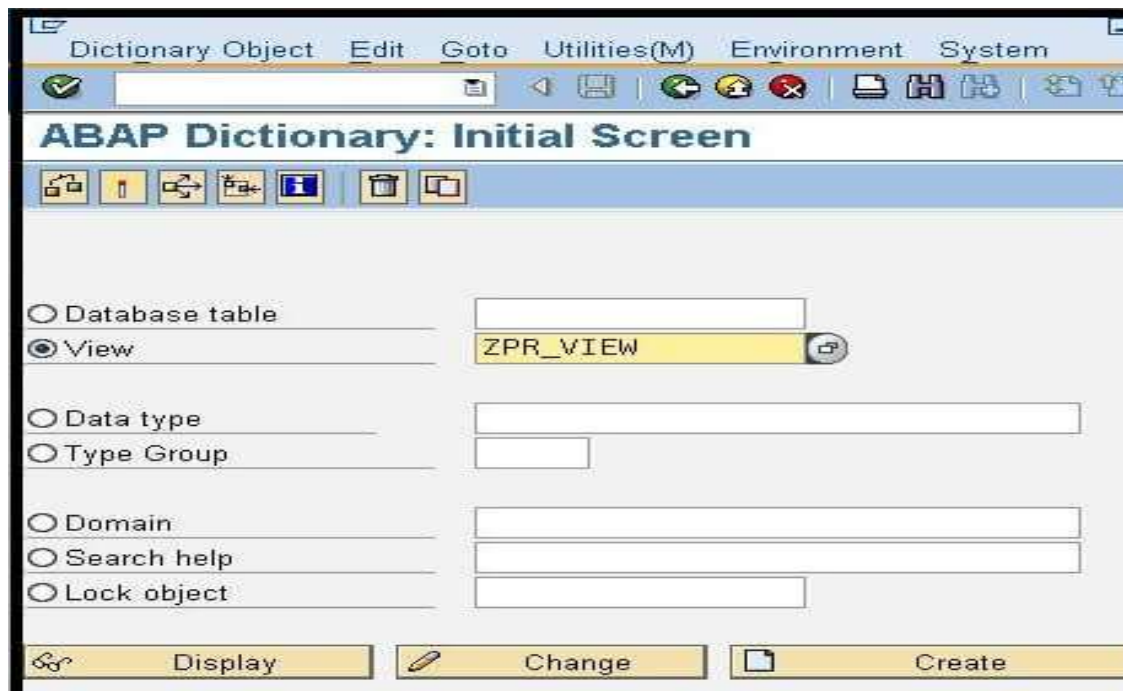
PROJECTION VIEW

- **ABAP Projection views** are special views used to ***hide some fields of a database table and display only the required or selected fields***. Hence, using this type of view, we can minimize the fields by projecting only required fields and rest fields will be filtered out.
- This view allows us to read and also maintain the needed data.
- The projection view is created on a **single table only**, and we cannot specify any selection/join conditions for this view.
- With the projection view, apart from the transparent database table, we can also access the **pooled tables** or **cluster tables**.
- Let's consider an example to understand the use of the projection view. Suppose we have Students Details table, which contains records such as **Student Name and Roll number, Student address, Student fees**, etc. We only want to share student name and roll number with some companies. In this case, we will use the projections view of the Student Details table. Hence, if there is a huge amount of data, but we only want to display the relevant data field, we can choose the projection view as the perfect option.



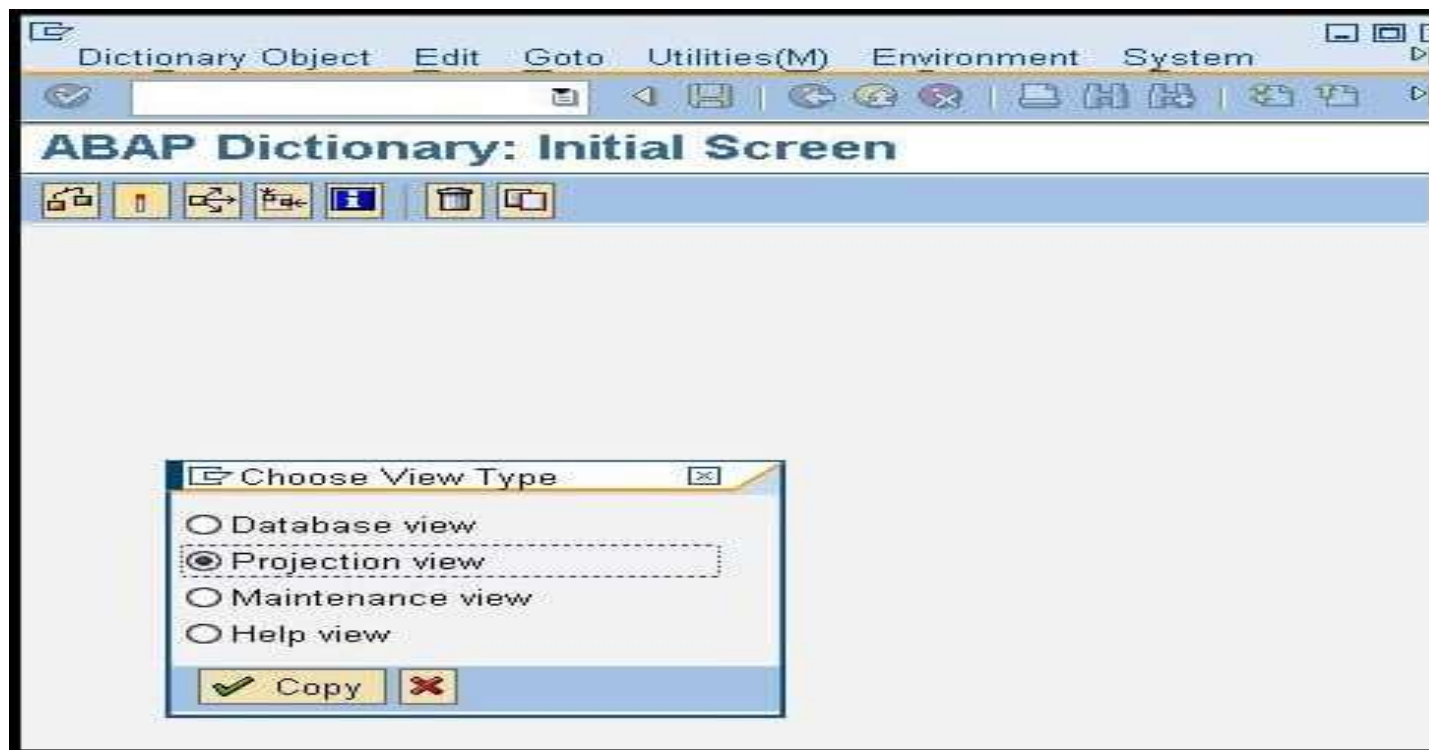
Creating Projection Views in ABAP:

- **Step-1:** Open the ABAP dictionary initial screen by navigating the menu path or entering the **transaction code SE11** in the command field.
- **Step-2:** Click the radio button in front of **the View** Give a name to the view, **and** click on the **Create** button, given on the screen. **(The name of the view should be started with z).**

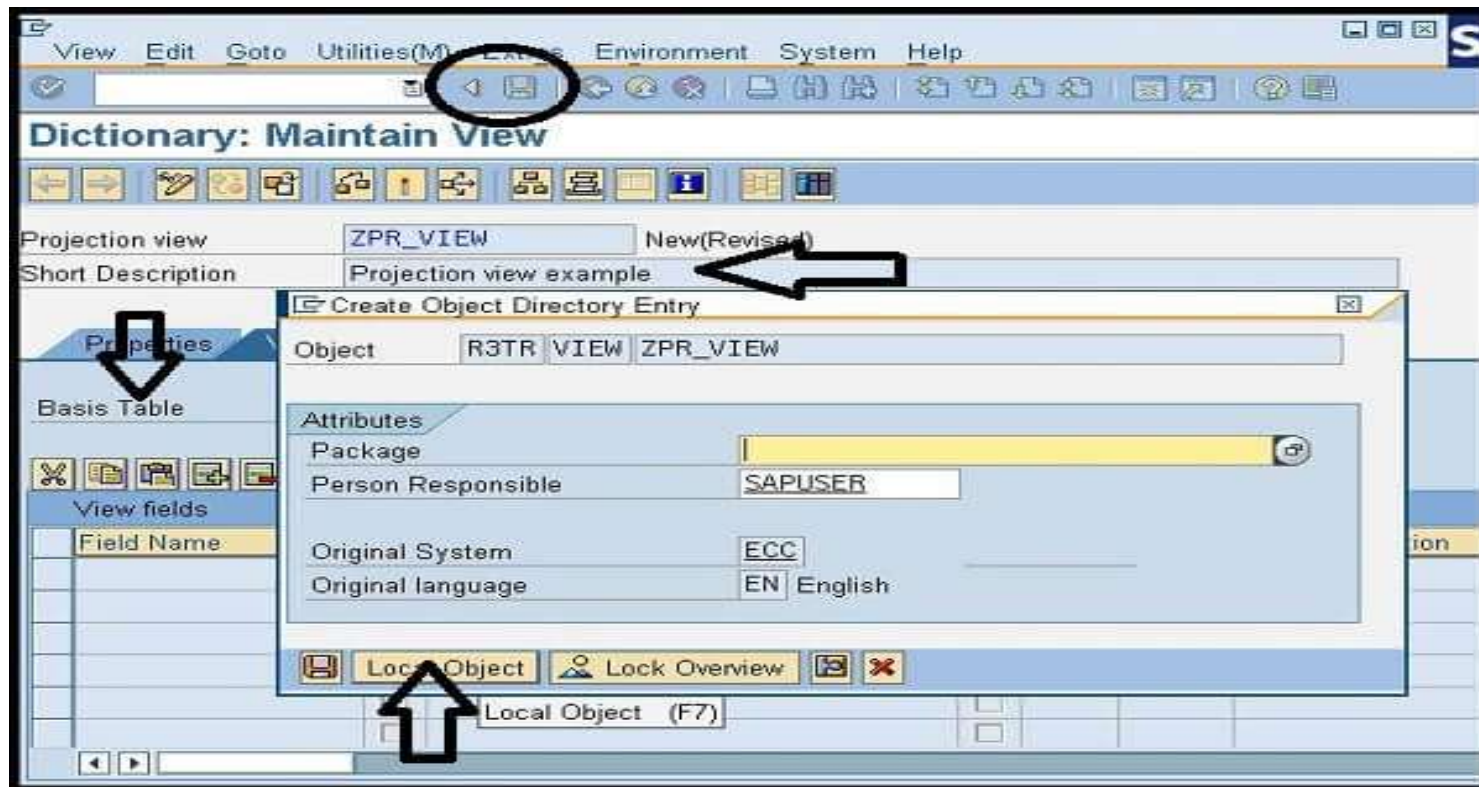


The screenshot shows the 'ABAP Dictionary: Initial Screen' window. The title bar includes 'Dictionary Object', 'Edit', 'Goto', 'Utilities(M)', 'Environment', and 'System'. Below the title bar is a toolbar with various icons. The main area contains several radio buttons for object types: 'Database table', 'View' (which is selected), 'Data type', 'Type Group', 'Domain', 'Search help', and 'Lock object'. To the right of these radio buttons are input fields. The 'View' radio button is selected, and the text 'ZPR_VIEW' is entered in the corresponding input field. At the bottom of the window, there are three buttons: 'Display', 'Change', and 'Create'.

Step-3: A pop-up window will appear with all the views, from where select the "**Projection view**," and click on the **Copy** button.

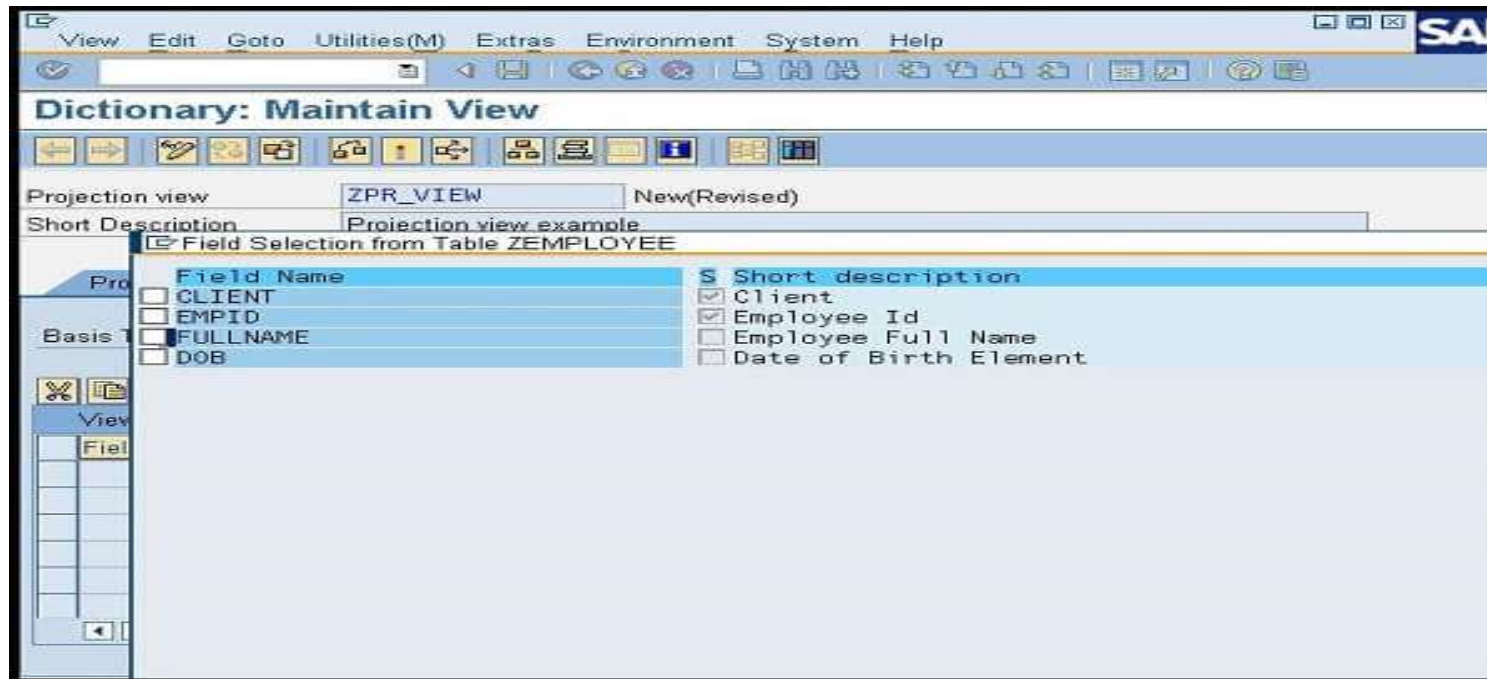


Dictionary: Maintain Screen Appears, consider the below image:



Step-4: Here, we have provided the short text for an explanation of view in the "**Short Description**" field, and entered the table name "ZEMPLOYEE"(we have created in the table creation)that we are going to use, in the **Base Table** Consider the above image. After entering the details, we need to save (CTRL+S) it as the local object or within a package. We are saving it as a local object.

Step-6: Click on **Table fields** button, and select the required fields that you want to display in the output. Click the **Copy** button, and all the selected fields will be displayed on the maintenance screen, as in the given image:

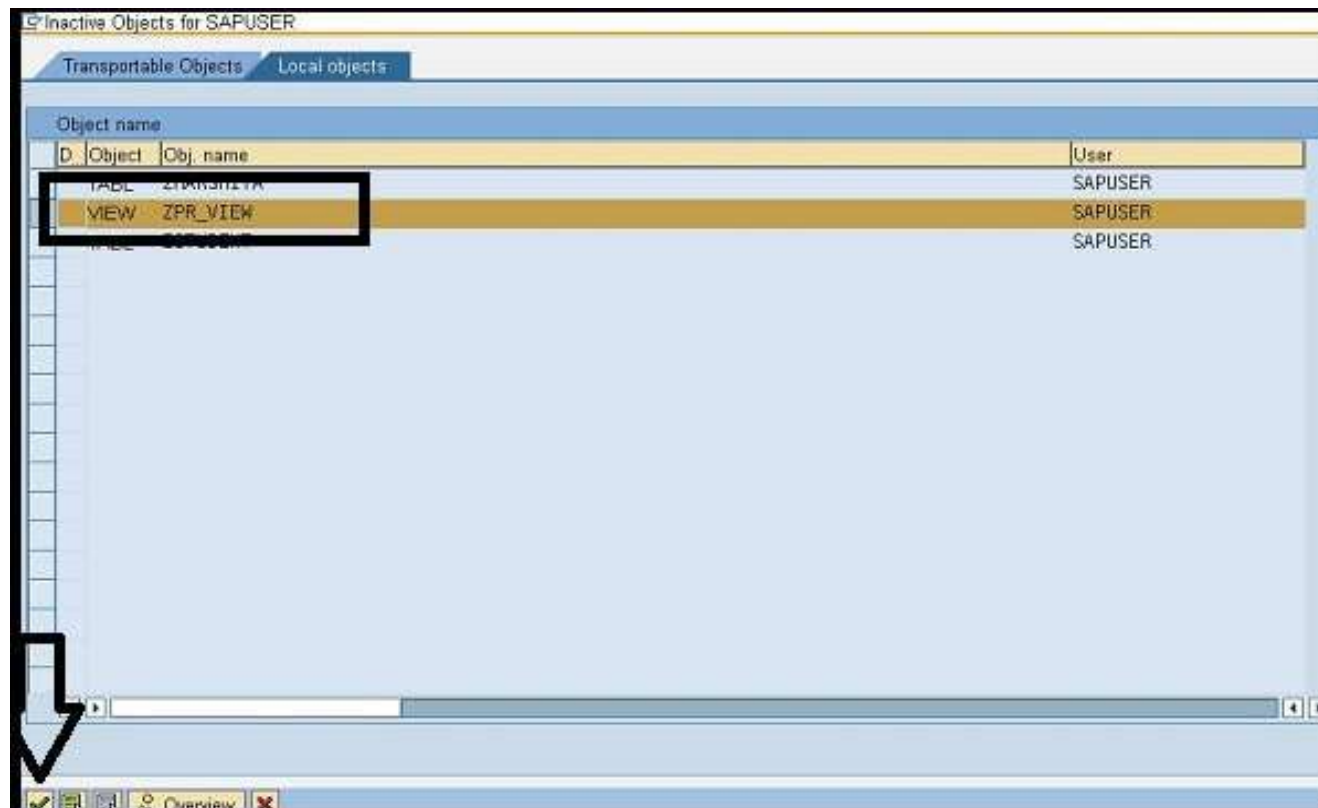


Here, we are choosing the EMPID and FULLNAME fields from the table. These fields will appear under the **Field Name** option.

The screenshot shows the SAP Dictionary: Maintain View interface. The 'Projection view' is set to 'ZPR_VIEW' and the 'Short Description' is 'Projection view example'. The 'Basis Table' is 'ZEMPLOYEE'. The 'Table fields' tab is active, showing a list of fields. The 'View fields' section is expanded, and the 'EMPID' and 'FULLNAME' fields are selected. The 'EMPID' field is highlighted in yellow, and the 'FULLNAME' field is highlighted in blue. The 'Key' column shows a key icon for 'EMPID' and a checkbox for 'FULLNAME'. The 'Data element' column shows 'ZEMPID' for 'EMPID' and 'ZFNAME' for 'FULLNAME'. The 'Mod' column shows checkboxes for both fields. The 'DType' column shows 'NUMC' for 'EMPID' and 'CHAR' for 'FULLNAME'. The 'Length' column shows '8' for 'EMPID' and '40' for 'FULLNAME'. The 'Short description' column shows 'Employee Id' for 'EMPID' and 'Employee Full Name' for 'FULLNAME'.

Field Name	Key	Data element	Mod	DType	Length	Short description
EMPID		ZEMPID	<input type="checkbox"/>	NUMC	8	Employee Id
FULLNAME	<input type="checkbox"/>	ZFNAME	<input type="checkbox"/>	CHAR	40	Employee Full Name
	<input type="checkbox"/>		<input type="checkbox"/>			
	<input type="checkbox"/>		<input type="checkbox"/>			
	<input type="checkbox"/>		<input type="checkbox"/>			
	<input type="checkbox"/>		<input type="checkbox"/>			

Step- 8: Now, save the view, check for any inconsistency, and Activate the view using ABAP Projection View the symbol.



Step-9: Click on the Utilities -> Contents, and execute, and you will get the Selection screen. The output data will be displayed at your screen. Consider the below output image

The screenshot displays the SAP Data Browser interface for Table ZPR_VIEW. The title bar includes the SAP logo and standard window controls. The menu bar contains Program, Edit, Goto, Settings, System, and Help. The toolbar features various icons for file operations and navigation. The main area is titled "Data Browser: Table ZPR_VIEW: Selection Screen". Below the title, there is a toolbar with icons for adding, deleting, and other functions, followed by a button labeled "Number of Entries". The selection criteria are defined by the following fields:

Employee Id	<input type="text"/>	to	<input type="text"/>	<input type="button" value="↗"/>
FullName	<input type="text"/>	to	<input type="text"/>	<input type="button" value="↗"/>
Width of Output List	<input type="text" value="250"/>			
Maximum No. of Hits	<input type="text" value="200"/>			

As we can see in the above output image, only two fields are displaying here that we have selected in the table.

We can either enter the details or can click on the execute or F8 given on the above screen for displaying all the records of the selected field. Consider the below image:

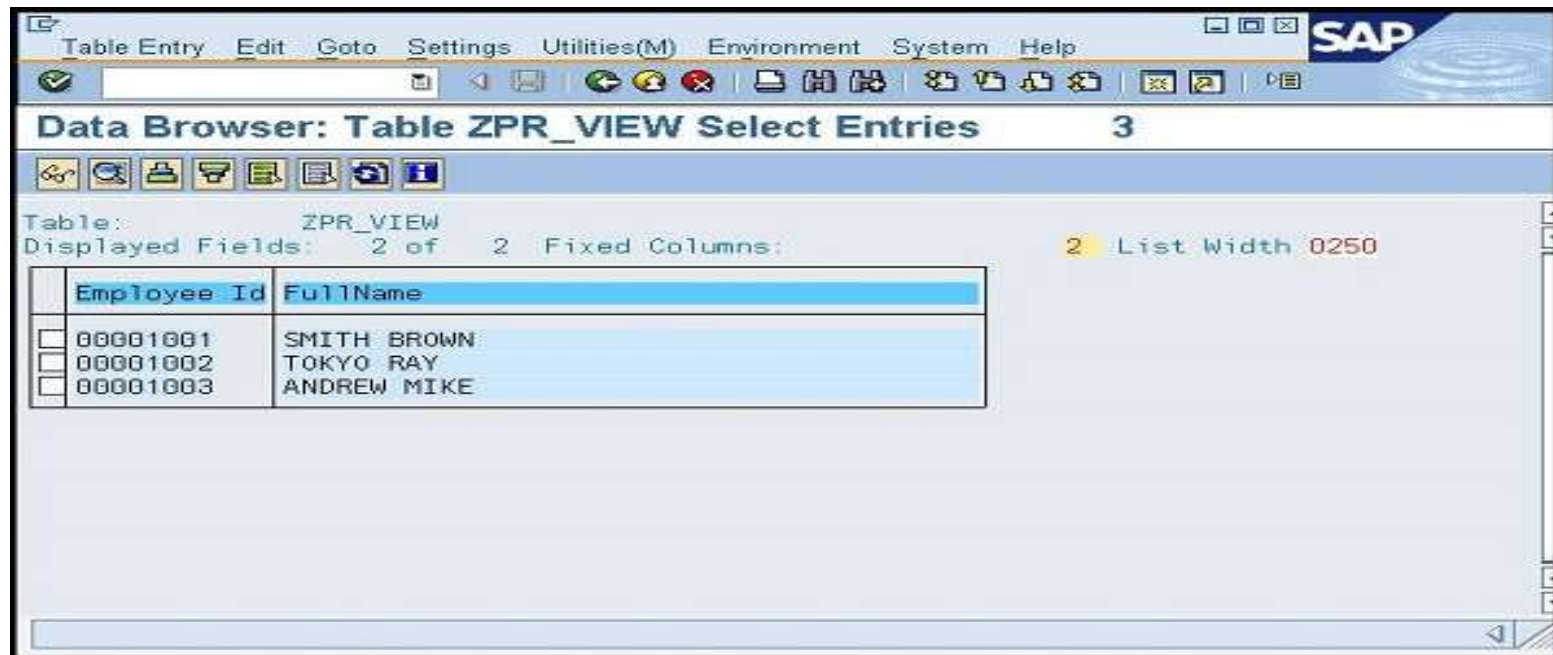


Table: ZPR_VIEW
Displayed Fields: 2 of 2 Fixed Columns: 2 List Width 0250

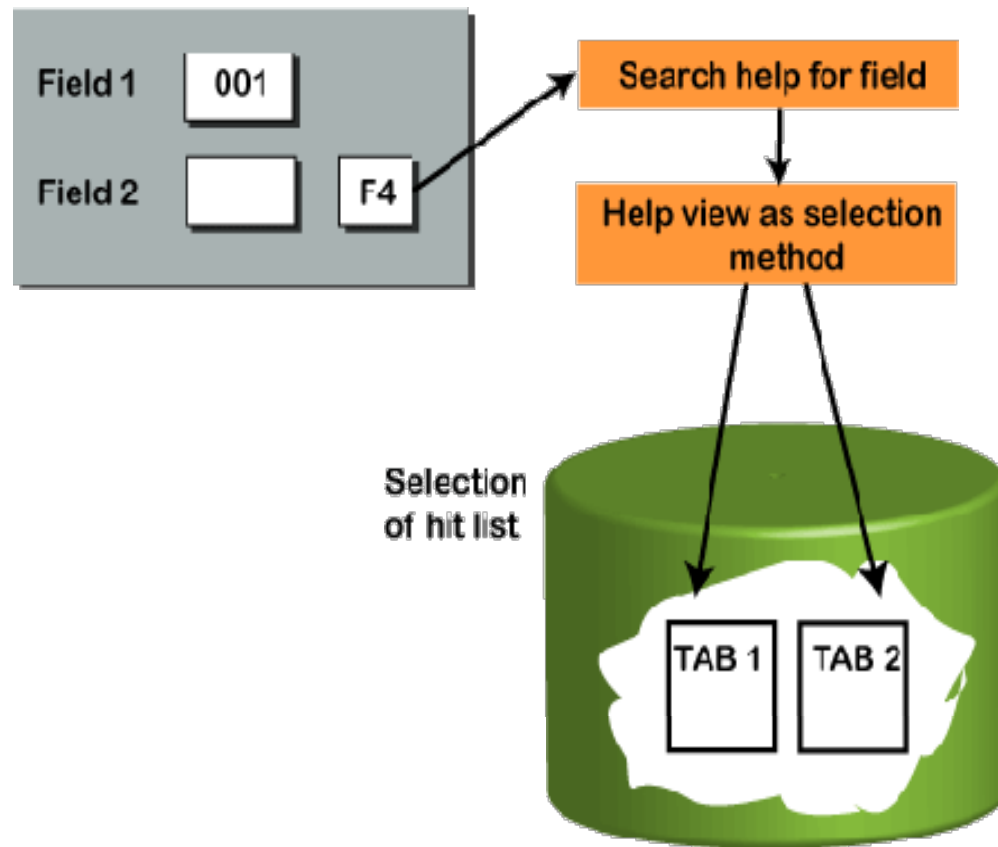
	Employee Id	FullName
<input type="checkbox"/>	00001001	SMITH BROWN
<input type="checkbox"/>	00001002	TOKYO RAY
<input type="checkbox"/>	00001003	ANDREW MIKE

Advantages of ABAP Projections View

- Using the projection view, we can display only the required fields and mask other fields.
- It helps to limit the complete data access of a table from an unauthorized user.
- Data access and the view from the projection view is faster than other views.
- It also improves the system's performance, as we minimize the fields by masking in this view.

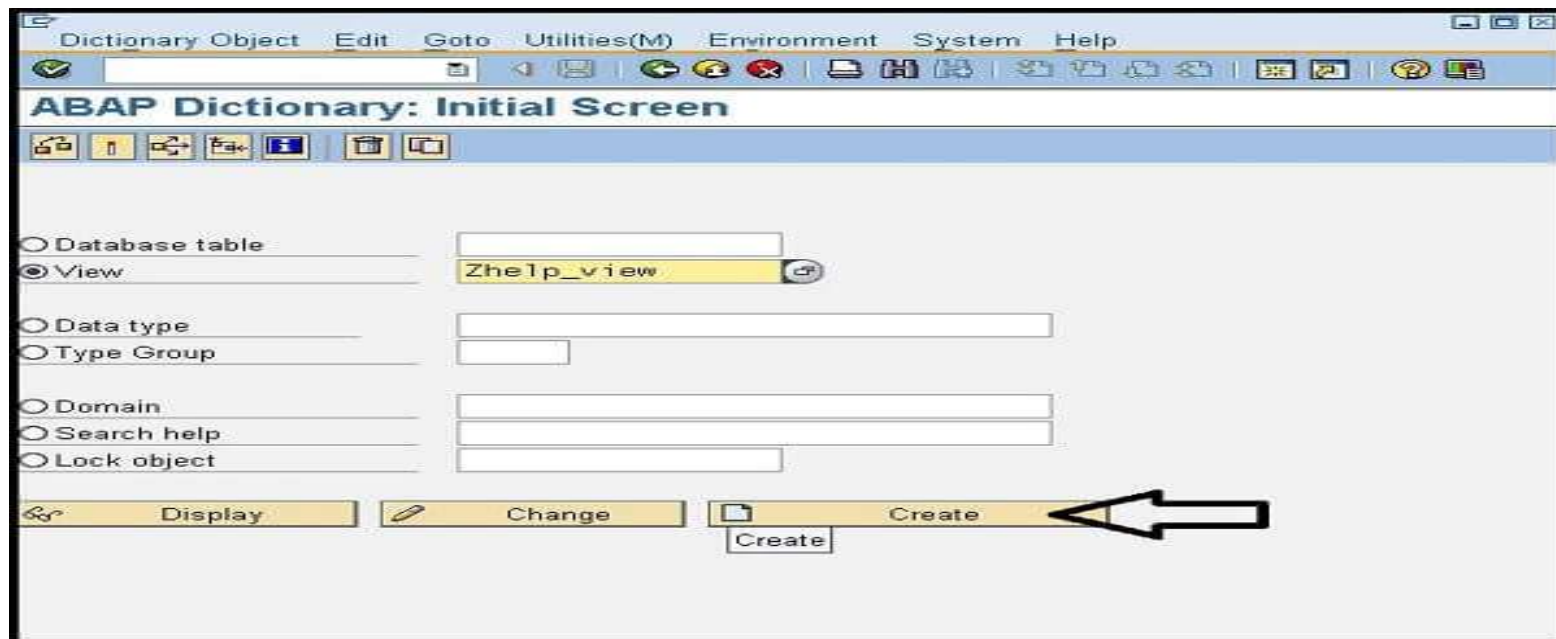
ABAP Help Views

- The help views are created on two or more tables, specifically for the "Search-helps" object in DDIC. It means they are used to provide the input helps(F4) option for different fields in ABAP.
- When going to search help, views with outer join act as the selection methods and these methods can be either a table or a view, which specifies that data can be selected from a view or table. For such purpose, we need to create the help views in ABAP.
- It combines the data by using **an outer join concept**.
- Since it is specially designed for the search help option, we cannot directly execute the data.
- It allows us to only read the data; we cannot maintain the data in this view.
- Note: A table can only be used as a selection method using the help view if we use the outer join to retrieve the data from the database.

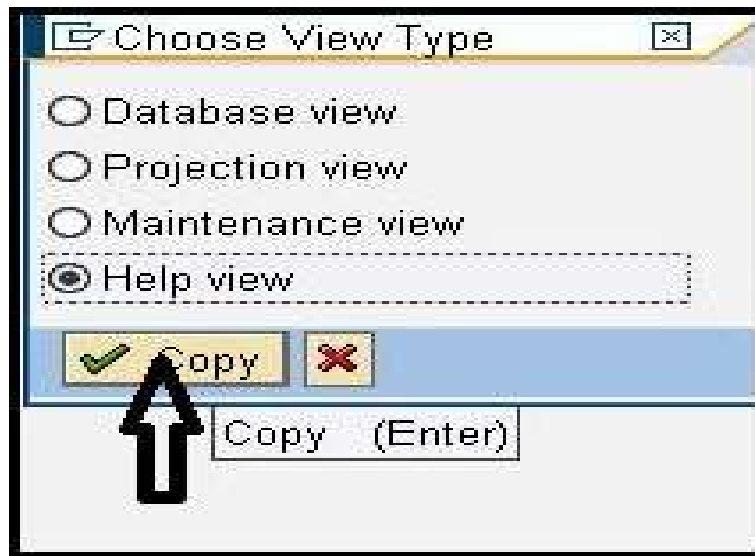


Creating the Help view in ABAP

- To create the help view, follow the below steps:
- **Step-1:** Open the data dictionary initial screen by navigating the menu path or entering the **transaction code SE11** in the command field.
- **Step-2:** Click the radio button in front of the **View** Give a name to the view, **and** click on the **Create** button, given on the screen.

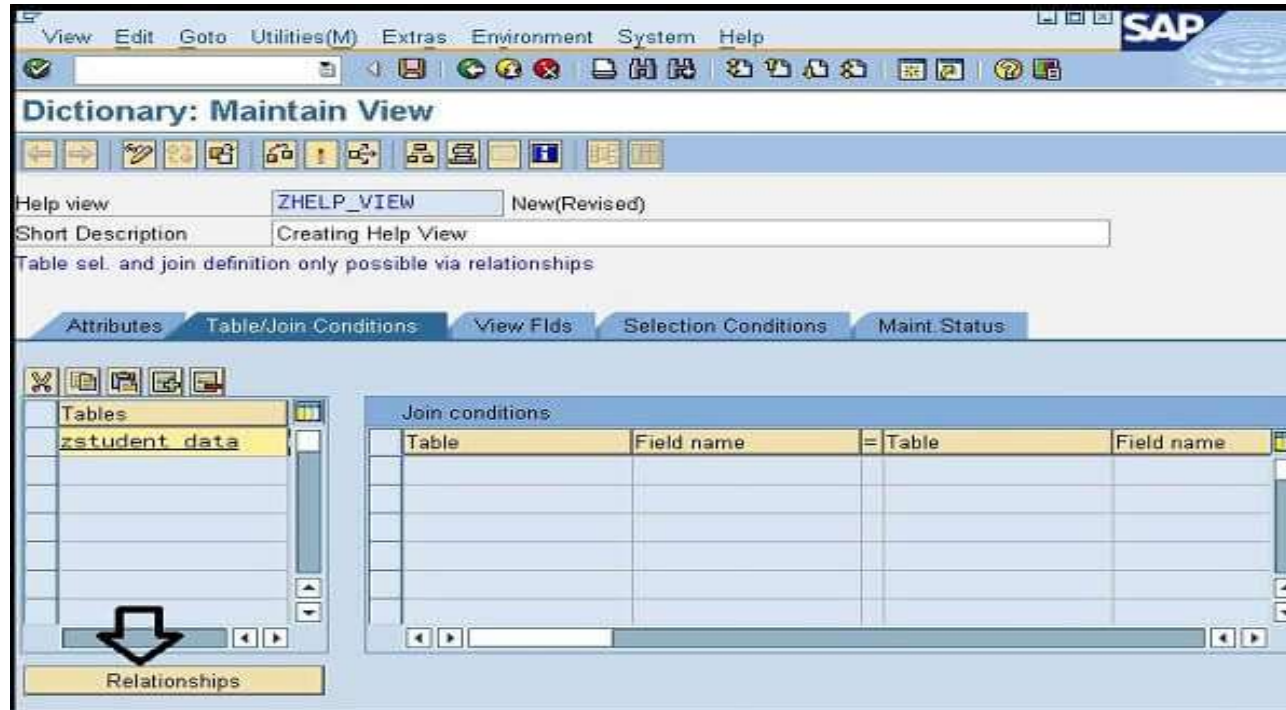


Step-3: A pop-up window will appear with all the views, from where select the "**HELP view**," and click on the **Copy**

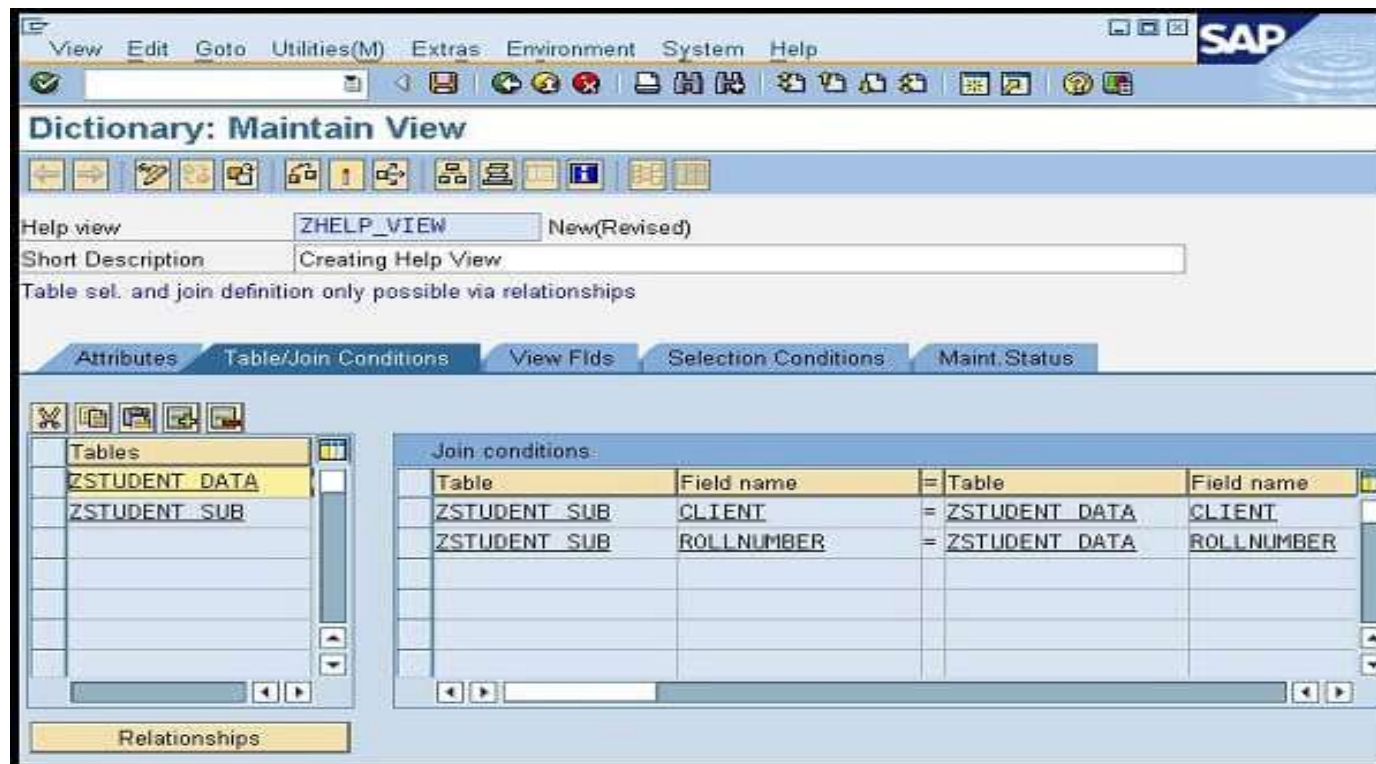


Step-4: Provide the short text for an explanation of view in the "Short Description" field. E.g., *Help view for a test or Creating a Help view.*

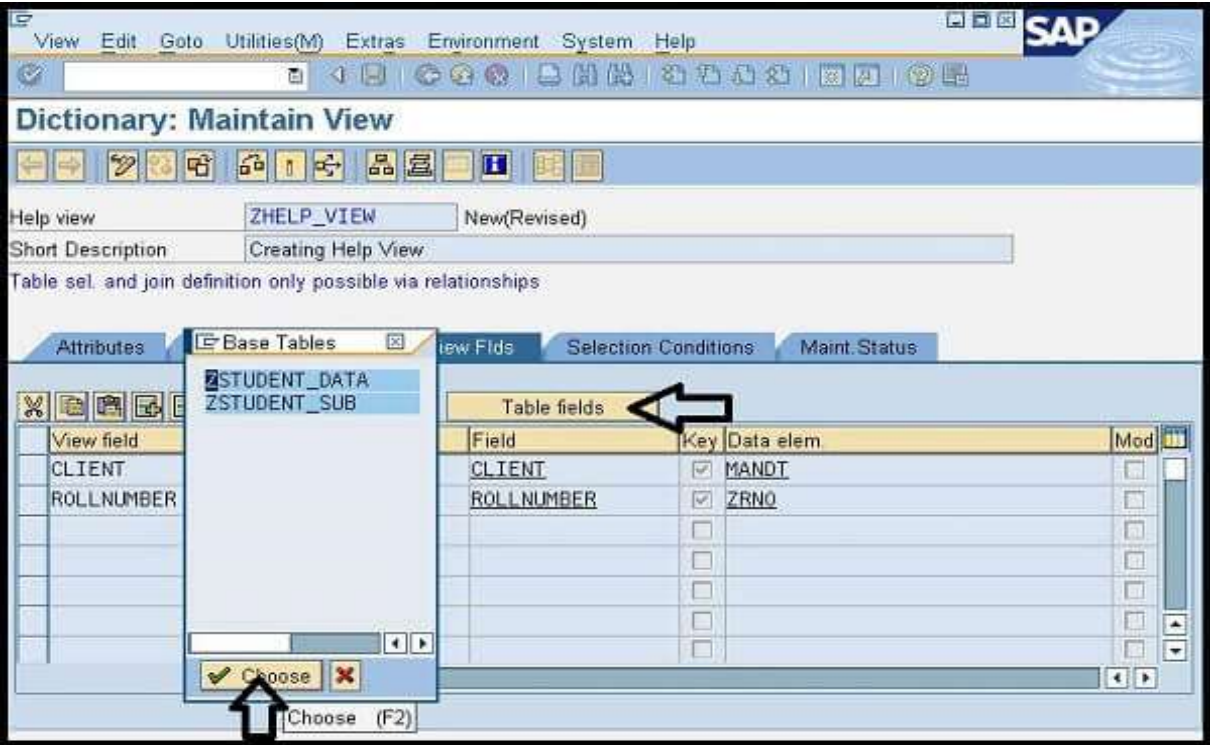
Step: 5: Provide the name of the primary table. We are giving the table name as `zstudent_data` in the primary table for view under the *Table* column in the **Table/Join Condition**.



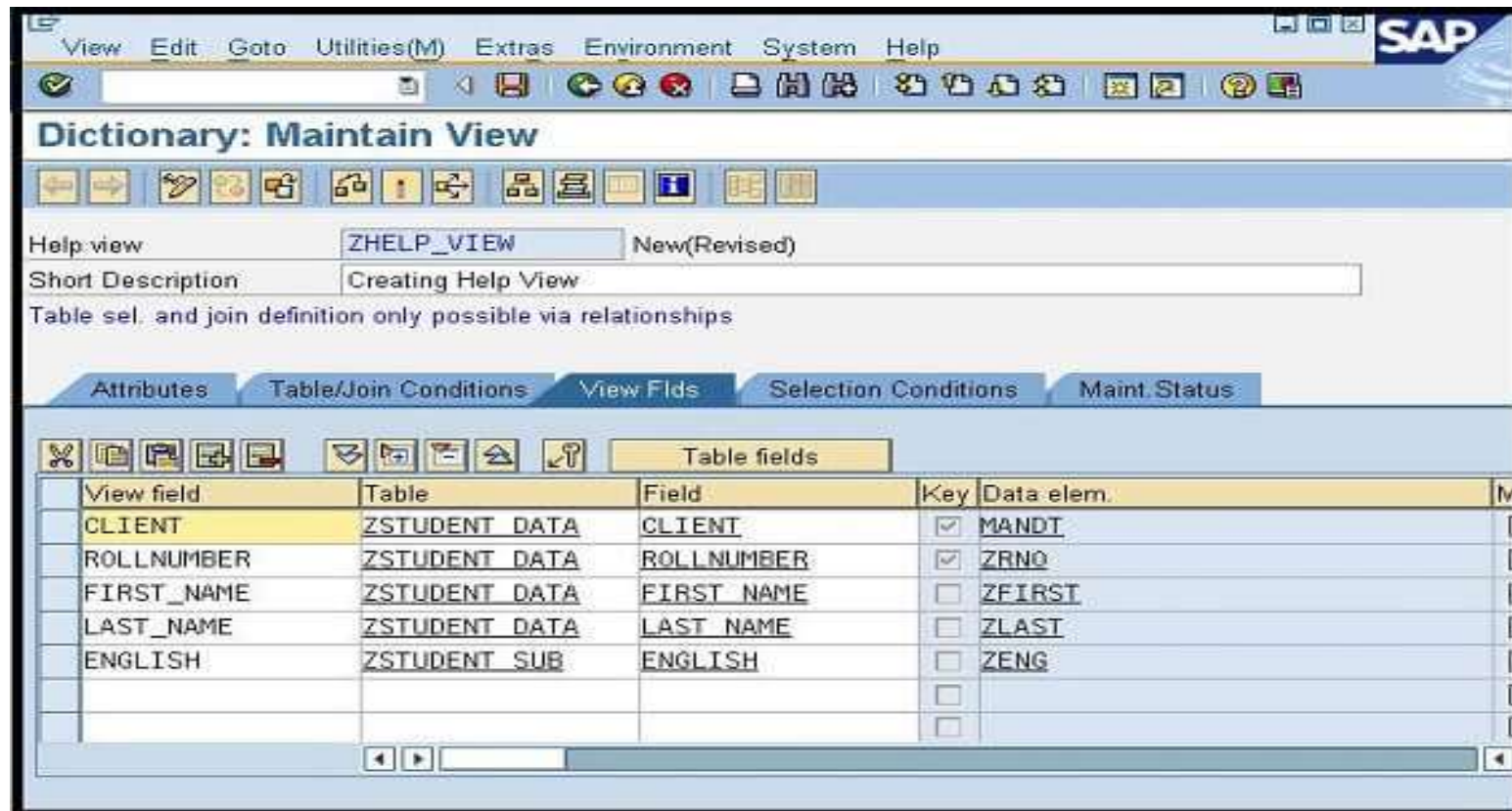
Step: 6: Place the cursor on the primary table name and press the **Relationship** A screen will appear, select the checkbox in front of the primary table's name, and click on the **copy** button. In our case, the secondary table is zstudent_sub, so it is inserted here.



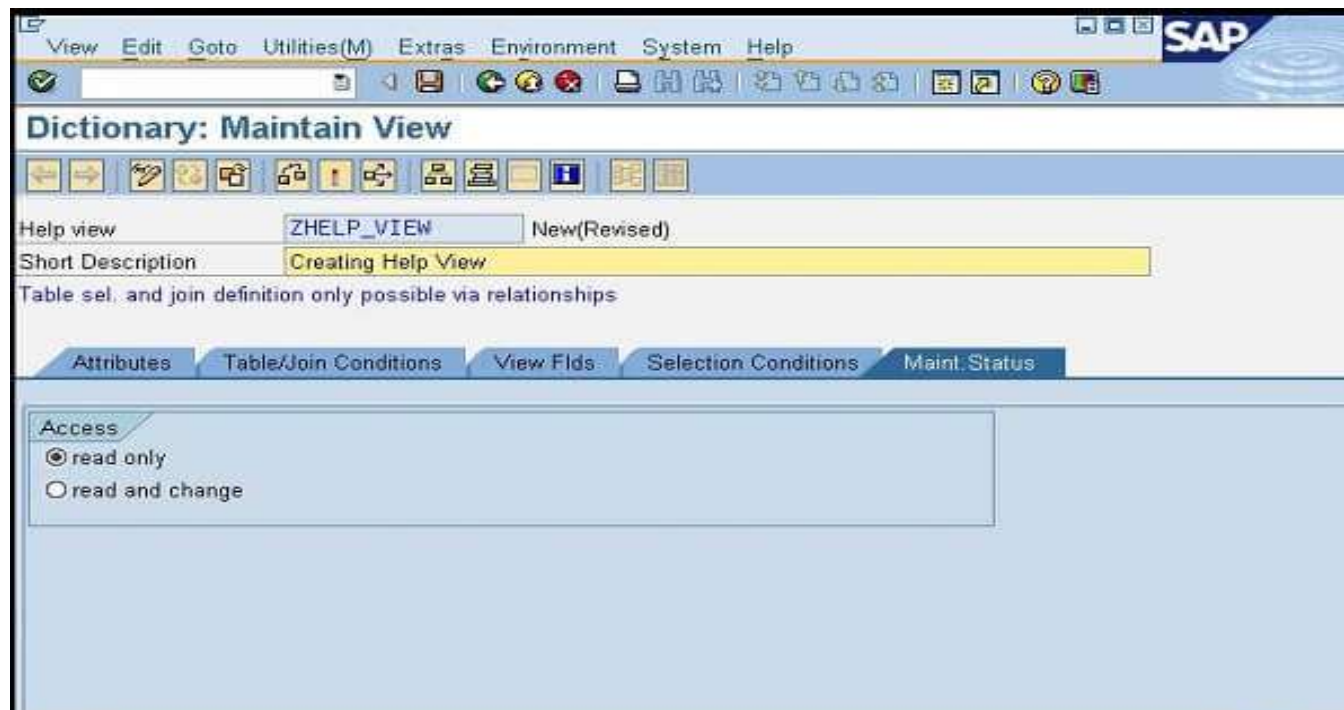
- **Step: 6** Now, we will select the required fields by choosing the **View** field tab.



Step: 8 Select the **Table Fields** button, and a list of tables contained in this view is displayed, as shown in the above diagram. From here, we can select any table, and then need to click on the Choose. All the selected fields will be displayed here. Consider the below image:



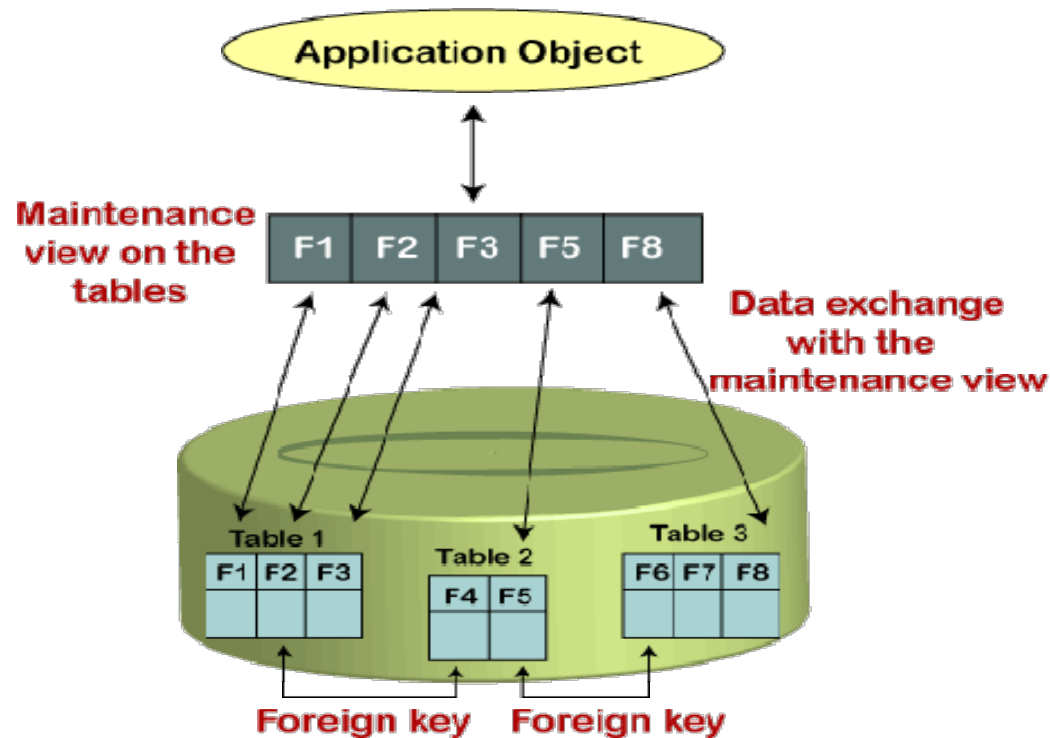
Step: 9 Select the Maint. Status tab and set the *read-only* radio button under **the Access group**



•Click on the **Save(CTRL+S)** and **Activate** it by clicking on . Once it is activated, we can use the help view as the selection method for Search Help.

ABAP Maintenance View

- The maintenance view is created on two or more tables, which is used to maintain the data of several tables altogether. It helps us to maintain the complex application objects easily.
- It can combine several tables in a single unit, *but the tables must have a foreign key relationship*.
- It allows us to **maintain and read the data** of the table.
- The data present on several tables create a logical unit that acts as an application object for the user. With the help of a maintenance view, we can **display, modify, and maintain the data** of such application objects at once.
- In the maintenance view, all tables must be linked together with the foreign key, which means the join condition must be derived from the foreign key always.
- There is always a **maintenance status** associated with each maintenance view, which specifies the operations that can be performed on the associated table.
- The below diagram explains the relationship between an application object, maintenance view, and the underlying database:



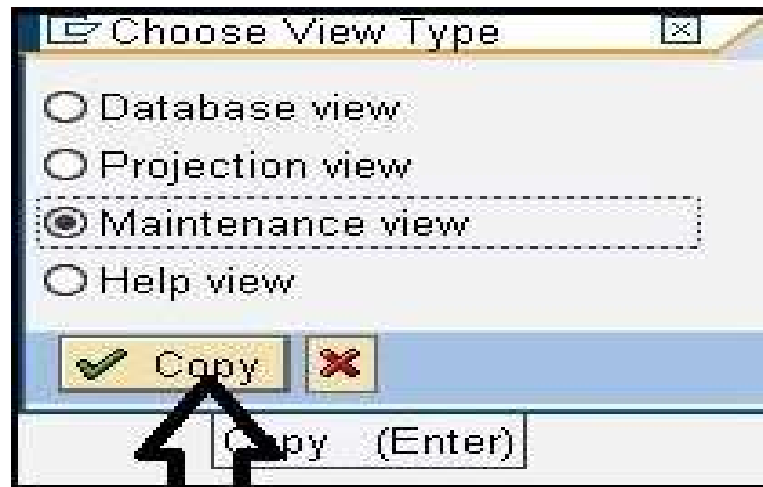
As we can see in the above diagram, the fields F2 and F4 of table 1 and table 2 are linked with the foreign key. Similarly, the fields F5 and F6 of table 2 and table 3 are connected with the foreign key. The maintenance view is implemented on these three tables and extracting data from F1, F2, F3, F5, and F8 fields.

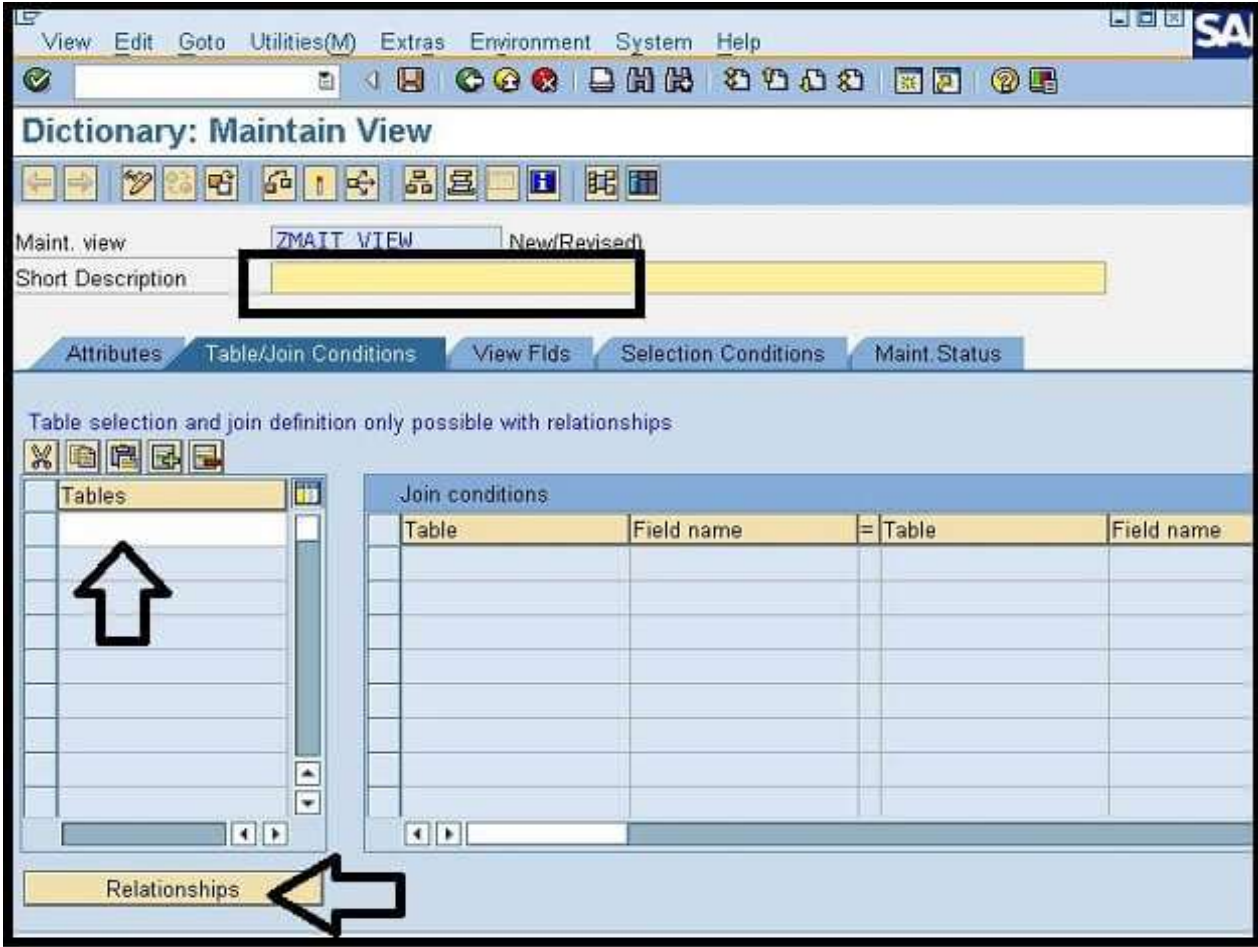
Creating Maintenance View in ABAP:

- Step-1:** Open the data dictionary initial screen by navigating the menu path or entering the **transaction code SE11** in the command field.
- Step-2:** Click the radio button in front of the **View** Give a name to the view, and click on the **Create** button, given on the screen.

The screenshot shows the 'ABAP Dictionary: Initial Screen' in SAP. The interface includes a menu bar (Dictionary Object, Edit, Goto, Utilities(M), Environment, System, Help) and a toolbar. The main area has several radio buttons for object types: Database table, View (selected), Data type, Type Group, Domain, Search help, and Lock object. The 'View' option is selected, and the text field next to it contains 'zmail_view'. Below these fields are buttons for 'Display', 'Change', and 'Create'. The 'Create' button is highlighted, and a 'Create' button is also visible in the bottom right corner.

Step-3: A pop-up window will appear with all the views, from where select the "**Maintenance view**," and click on the **Copy**

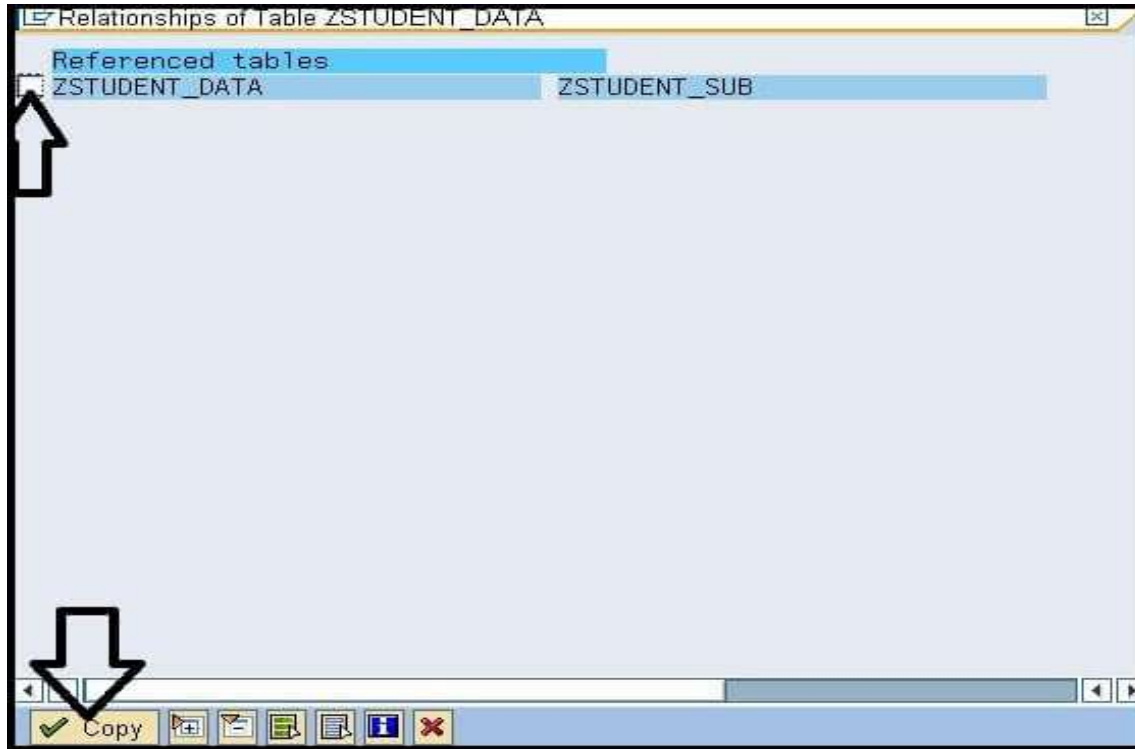




- Step-4:** Provide the short text for an explanation of view in the "**Short Description**" field. E.g., **Creating a maintenance view for test** or **Maintenance view example**.
- Step- 5:** Click on the **Table/Join Conditions** tab, provide the name of the primary table for view under the Table column.

[illegible]

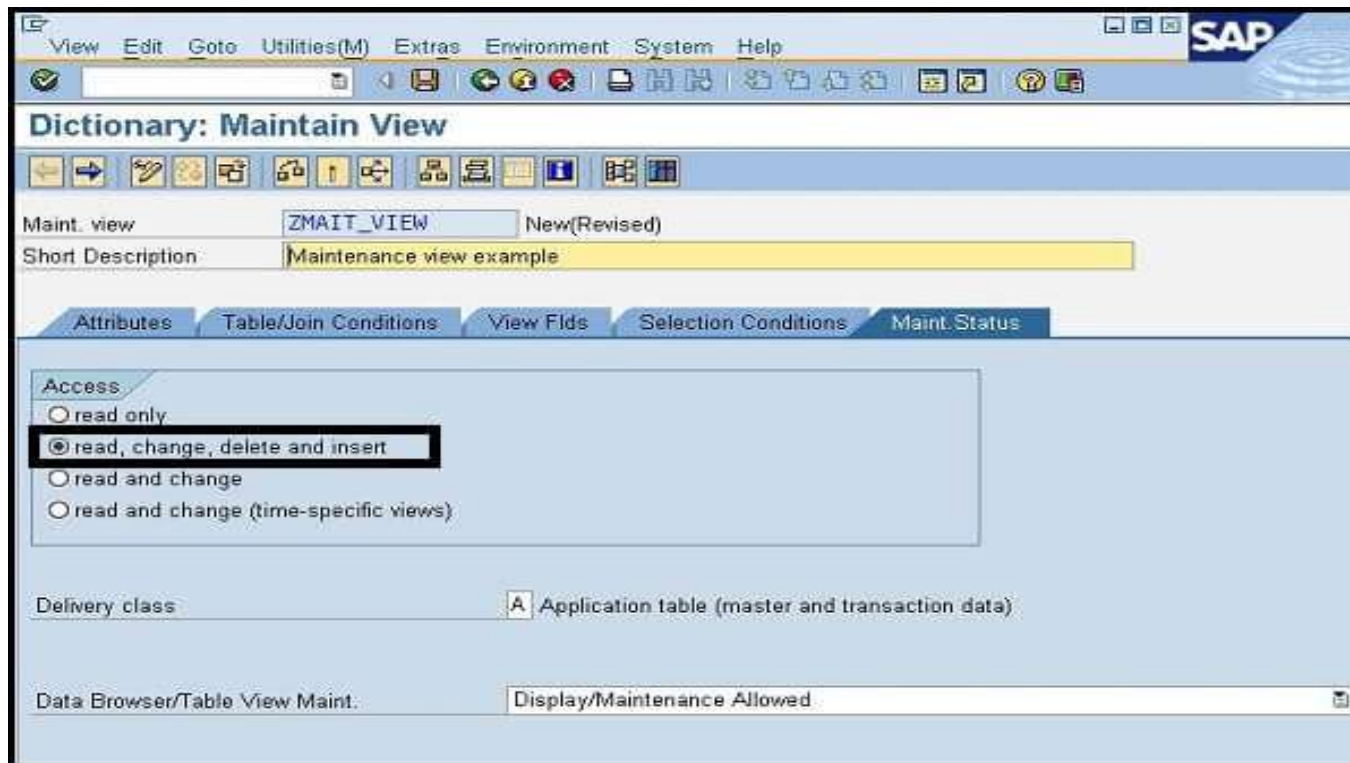
- Step-6:** Place the cursor on the name of the primary table name and click on the **Relationships**
- Step- 7:** Tick on the checkbox in front of the table name.



- Step-8:** Click on the copy button, and the secondary table will be selected.
- Step-9:** Go to the **selection conditions tab** and provide the conditions. (If required). Here we are not providing any selection condition.
- Step-10:** Go to the **View fields** tab, click on the **Table Fields** button to select the field from each table(primary and secondary).

Repeat the process for the secondary table (ZSTUDENT) also, and all the selected fields will be displayed under the view field column.

•**Step-11:** Go to **the Maint. Status tab** and select the appropriate radio button. Select the other fields, as shown in the below image:



Step-12: Go to the **Utilities> Table Maintenance Generator**, as shown in the below diagram.



Fill the details as shown by the arrow, and save it.

Generated Objects Edit Goto Environment Utilities(M) System Help

Find Scr. Number(s)

Table/View ZMAIT_VIEW

Technical Dialog Details

Authorization Group &NC&

Authorization object S_TABU_DIS

Function group zmaint_view

Package

Maintenance Screens

Maintenance type ☒ one step ☐ two step

Maint. Screen No. Overview screen 100

Single screen

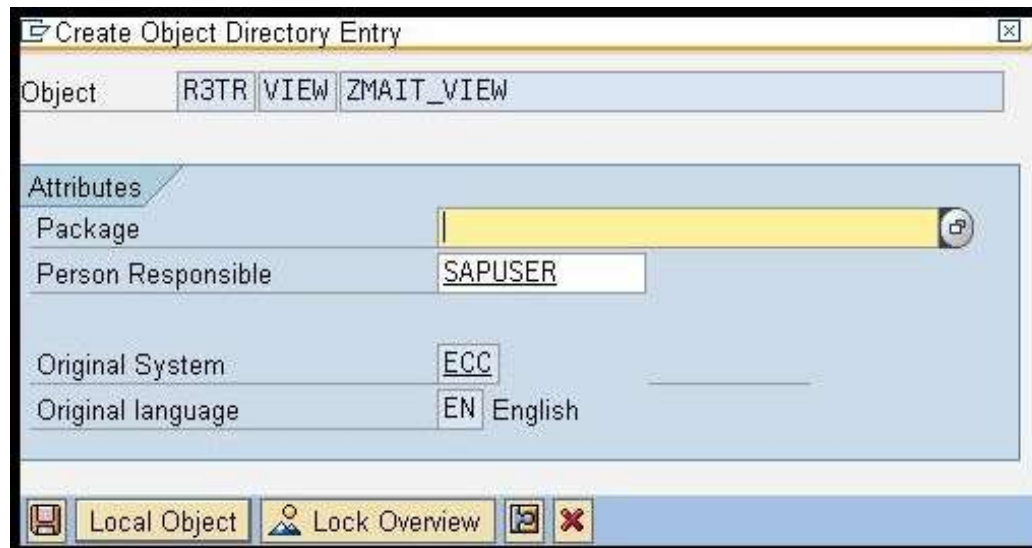
Dialog Data Transport Details

Recording routine ☐ Standard recording routine ☒ no, or user, recording routine

Compare Flag Automatically Adjustable

Note

Step-13: Click on **Save**, Save the view as the local object or under a package. Here we will save it as a local object. Check for any inconsistency and Activate the view.

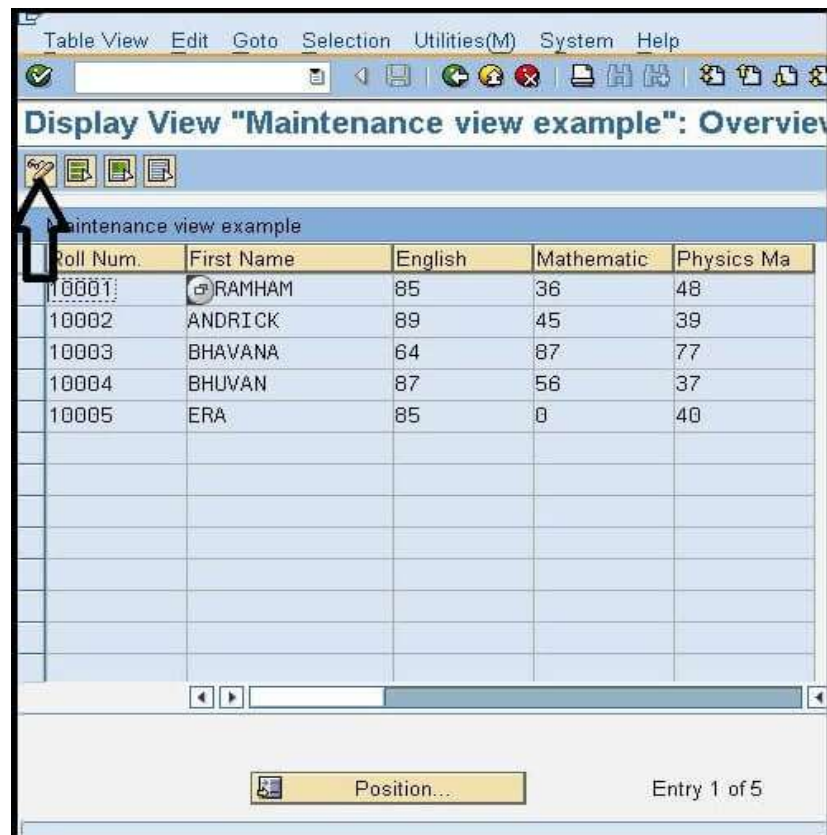


The screenshot shows the 'Create Object Directory Entry' dialog box. The 'Object' field is set to 'R3TR VIEW ZMAIT_VIEW'. The 'Attributes' section is expanded, showing the following fields:

- Package:** A yellow dropdown menu with a lock icon.
- Person Responsible:** A text field containing 'SAPUSER'.
- Original System:** A text field containing 'ECC'.
- Original language:** A text field containing 'EN' and the label 'English'.

The bottom of the dialog features a toolbar with the following buttons: 'Local Object' (highlighted), 'Lock Overview', and a red 'X' button.

Step-14: Now select **utilities> Contents**. A new screen will appear, which allows you to read, maintain, and write the data.

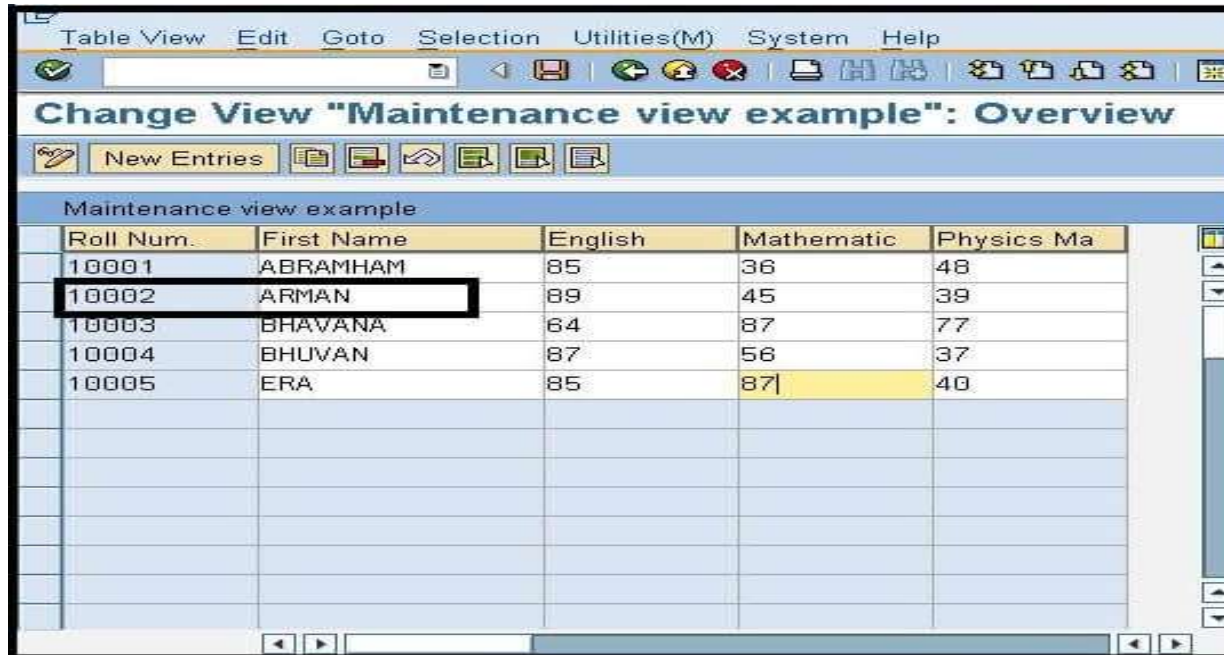


The screenshot shows a software window titled "Display View 'Maintenance view example': Overview". The window has a menu bar with "Table View", "Edit", "Goto", "Selection", "Utilities(M)", "System", and "Help". Below the menu bar is a toolbar with various icons. A black arrow points to the "Utilities(M)" menu item. Below the toolbar is a table with the following data:

Roll Num.	First Name	English	Mathematic	Physics Ma
10001	RAMHAM	85	36	48
10002	ANDRICK	89	45	39
10003	BHAVANA	64	87	77
10004	BHUVAN	87	56	37
10005	ERA	85	0	40

At the bottom of the window, there is a status bar with a "Position..." label and "Entry 1 of 5".

Here we are changing the Name **ANDRICK to ARMAN**, which has changed successfully. Consider the below image:



Roll Num.	First Name	English	Mathematic	Physics Ma
10001	ABRAMHAM	85	36	48
10002	ARMAN	89	45	39
10003	BHAVANA	64	87	77
10004	BHUVAN	87	56	37
10005	ERA	85	87	40

Advantages of Maintenance view:

- With the help of a maintenance view, we can extract the required data from several tables, which is faster than selecting data from an individual table.
- It saves time by maintaining several tables at once, as maintaining each table separately is time taking process.