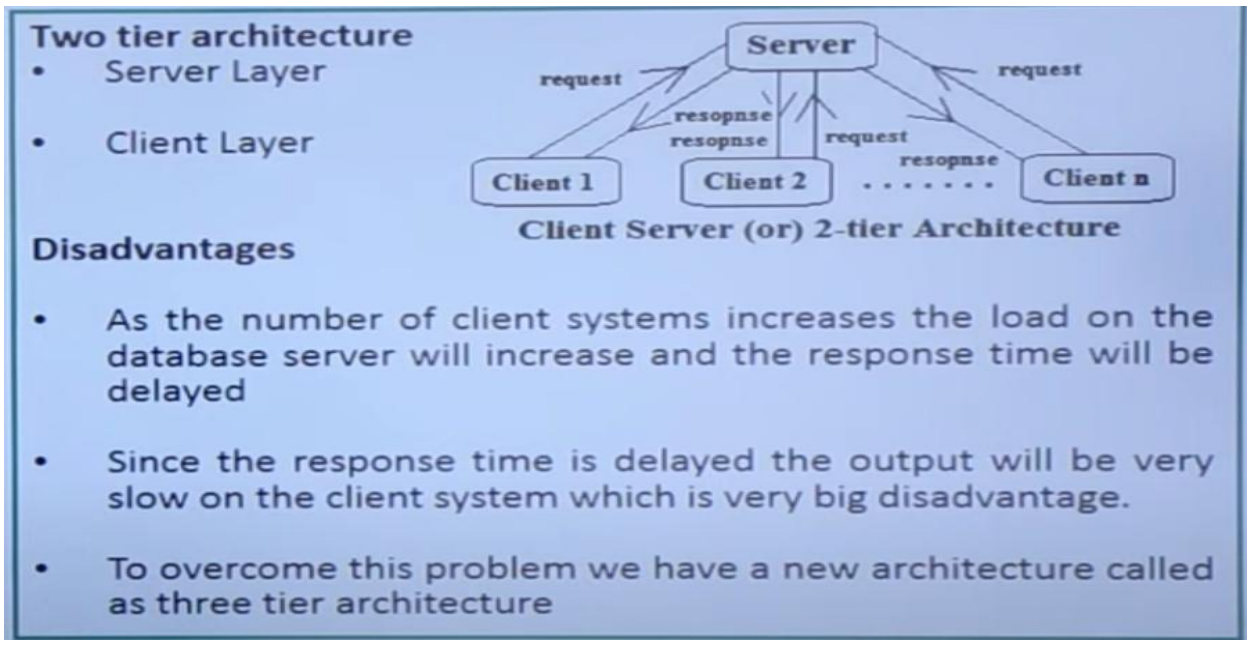
SAP: SYSTEMS APPLICATIONS AND PRODUCTS

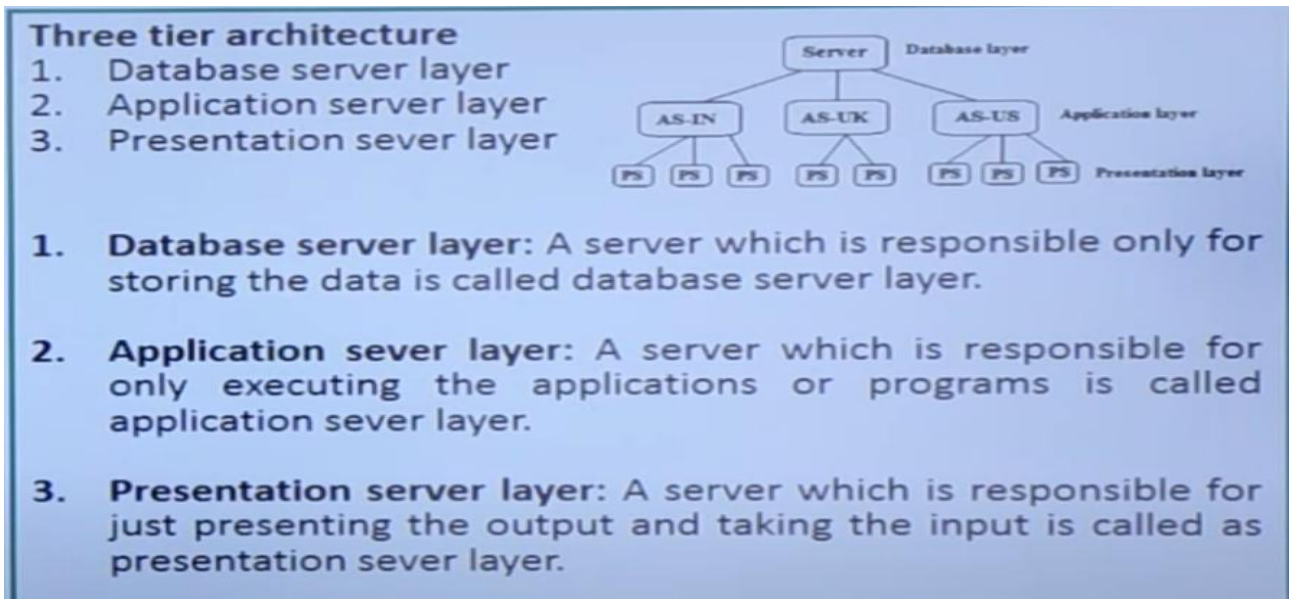
ARCHITECTURE OF SAP:

Two type of SAP Architecture

R/2 Architecture: Server layer and database layer.



1. R/3 Architecture :



ABAP: ADVANCED BUSINESS APPLICATION PROGRAMMING

It was initially used for report creation, later fully evolved as a programming language.

SAP MODULES:

There are various business modules in SAP. Some of them are :		
SD	:	sales and distribution
MM	:	material management
FI	:	finance
CO	:	controlling
HR	:	human resources
CRM	:	customer relation management
SRM	:	supply relation management
SCM	:	Supply chain management
APO	:	Advanced Planner and Optimizer

TRANSACTON CODE:

It is a unique code or shortcut code which is used to display a specific screen.

Example:-

SE11: Displays ABAP dictionary

SE38: Displays ABAP editor

SE37: Displays ABAP function builder

SE21: Displays ABAP package builder

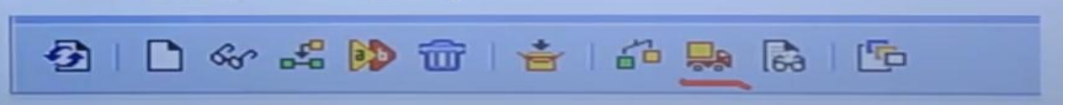
SE80: Displays ABAP development work bench .... Etc.

TRANSPORT REQUEST :(TR)

Creating a TR from Dev to Quality

Steps :

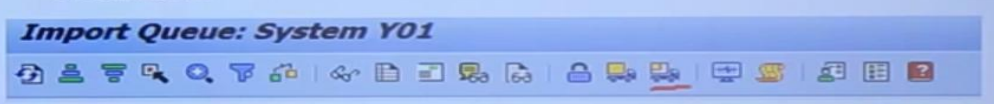
1. Create a TR in the development. e.g. while saving a Report, a prompt appears to save the report in a TR.
2. Within the Development, go to transaction se10.
3. Provide the User Name, click the Modifiable check box and click on Display.
4. You will see number of TR that are modified.
5. Open any one of the TR by pressing the + icon. You would notice a child TR.
6. Select one by one both the TR and click on the small Truck icon to move it to Quality.



Moving a TR from Dev to quality

Steps :

1. Within the Quality system, go to T-code Stms_import.
2. Here you would find all the already imported/importable transports.
3. Select the TR that you wish to import, and click on the small truck icon.

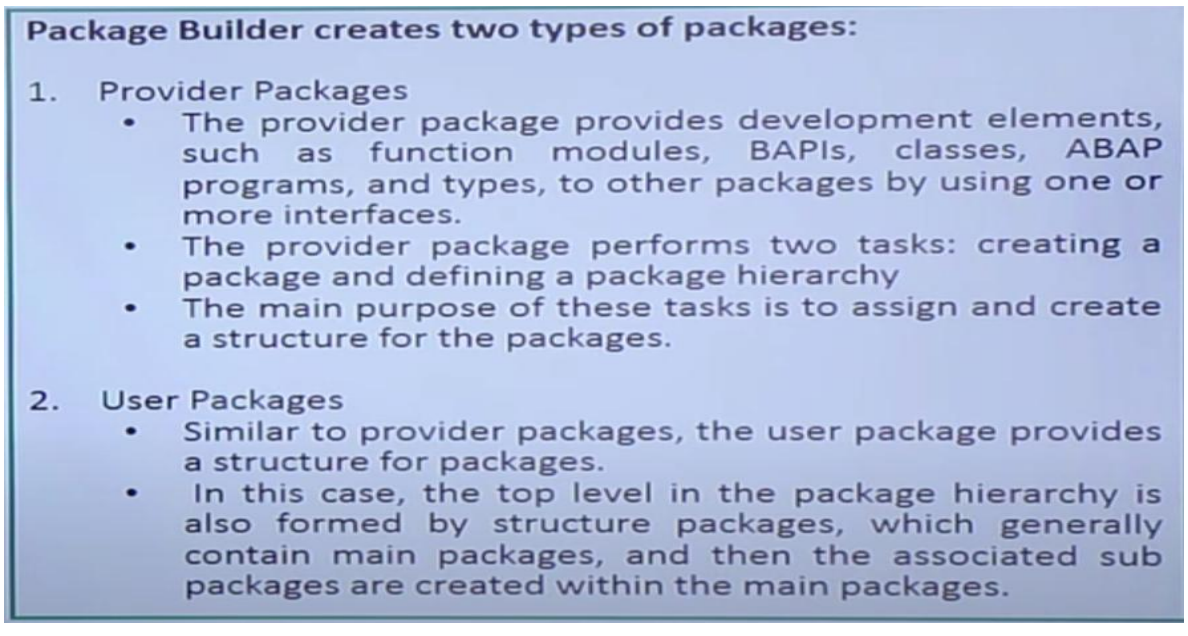


4. You get a popup, select the desired option available accordingly.
5. The TR would be imported in to the Quality system.

PACKAGE BUILDER:

Package builder is a type of development object that acts as a container to store development objects such as function modules, screens, menus and transactions.

- It is created using SE21 T- code or SPACKAGE.
- Name package starting with Z or Y.
  ex: Z_Cust_package.



Developing a ABAP Program:

- Go to SE38 T-code, provide name and click on Create.
- Provide some description or Title.
- A TR will be generated and the ABAP editor will get opened.

WRITE STATEMENT:

It is used to print the text on the output screen.

Ex program:

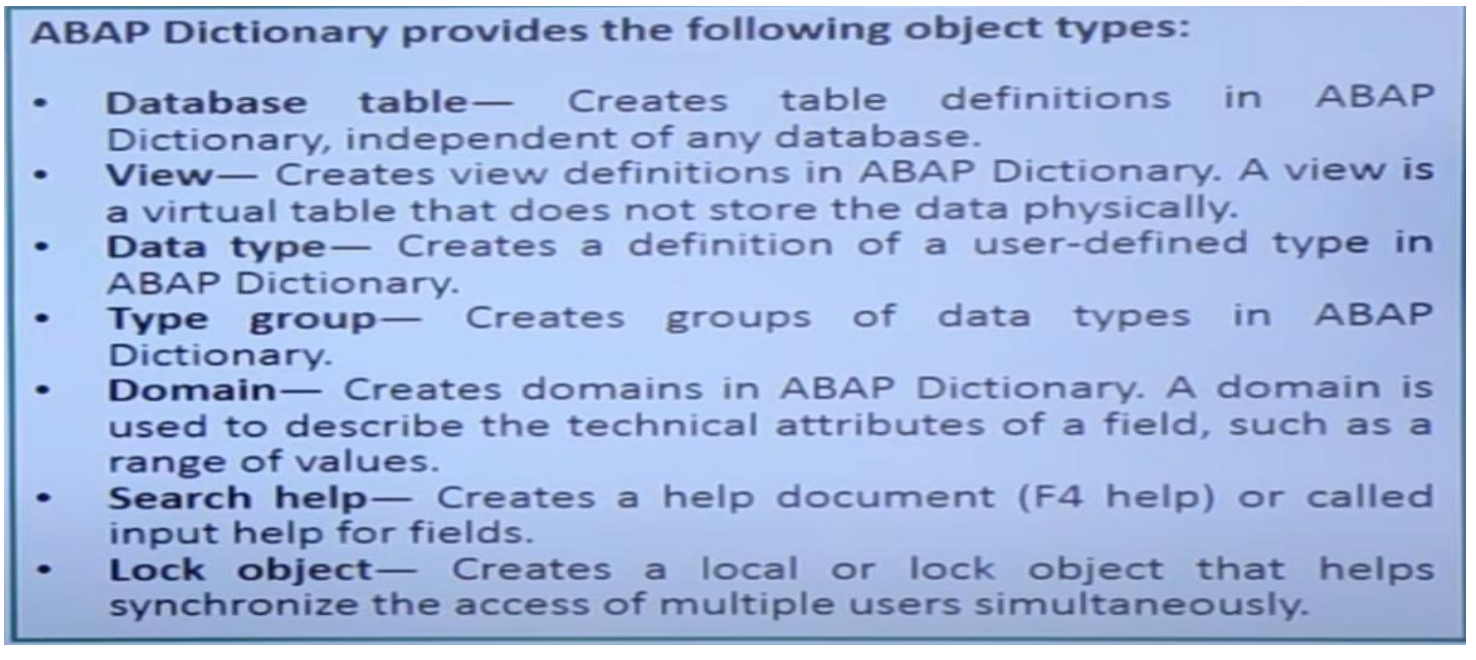
- Report ZSample.
  Data name1 (4) Type C.
  name1 = ‘abcd’.
  Write name1.

DATA DICTIONARY:

ABAP Data dictionary is the central repository where we define and maintain the objects which are related to database.

Two ways to open data dictionary:

1. SAP Menu >> Tools >> ABAP Workbench >> Development >> ABAP Dictionary or
2. Simply enter the Transaction code ‘SE11’.



- **DATABASE TABLE:**
  It is a combination of rows and columns, where rows are records and columns are known by fields.
  FIELD:
  It is a combination of Domain and Data element

Domain
- It is an object which specifies technical information such as data type and length for a field.
- It also specifies sign, lower case, conversion routine, fixed values, and value table.

Data Element
- It is an object which specifies semantic information such as field description, field labels (short, medium, long and heading) for a field.

Advantages
- Reusability: The same data element and domain can be reused by multiple table fields, instead of creating again and again.
- These are also used in creating foreign key relationship, search help, ale-idoc's.

Key field
- A field which is used to identify the record uniquely is called a key field. In a table there should be at least one key field.

<u>STEPS TO CREATE A TABLE:</u>

- Go to **SE11** T-code, provide table name 'Z_customTable' and click on create.
- Provide description and specify delivery class as 'A'.
- Specify display maintenance allowed and click on fields tab.
- Provide field name and data element name, then click on Save
- Click on technical settings button
- Provide the details as data class → APPL0, Size category → 0
- Click Save and click on back, then save and activate the table.

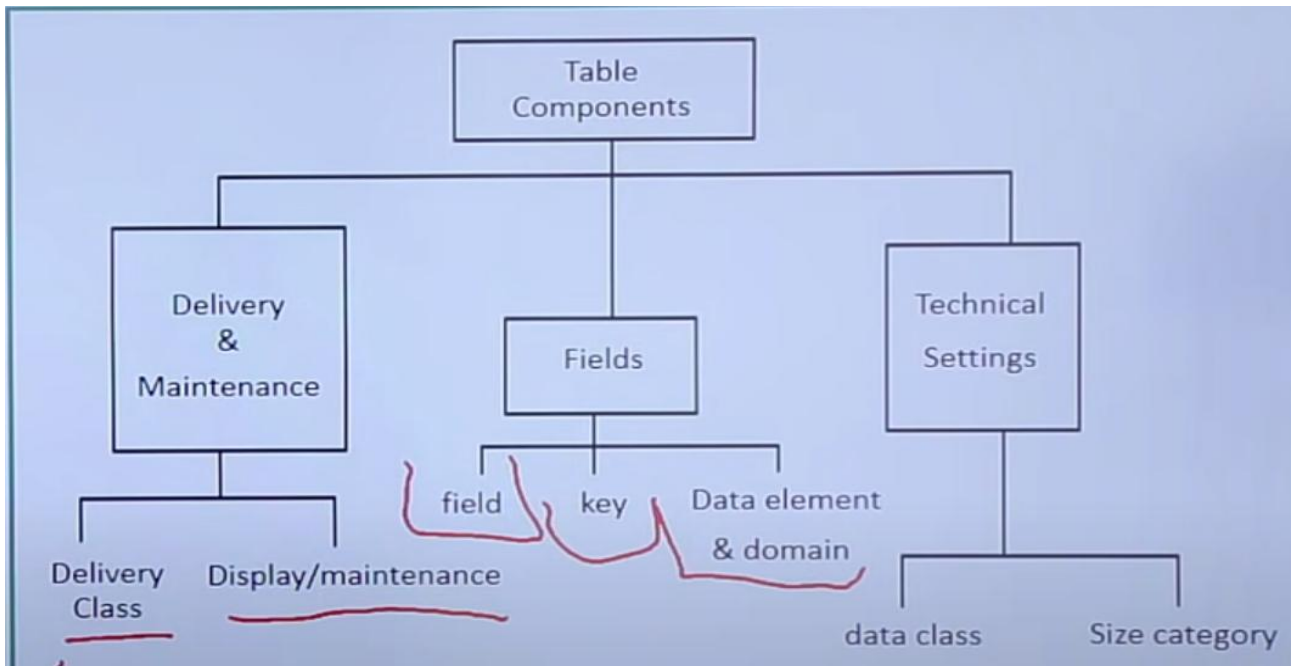
Creating records into the table:

- Go to SE11
- Give the table name and click on change
- Click on utilities -> table contents -> create
- Provide the cno, cname, city
- Repeat the same procedure and create the records

Displaying the table contents:

- Click on utilities -> table contents -> display
- Click on execute button
- The records will be displayed

COMPONENTS OF TABLE:



DELIVERY CLASS:

Delivery class:

- It specifies the type of the data that stored in a table.
- The data can be business data or system data.
- Business data means application data (master and transaction data).

The following options are available.

Delivery Class	Short Text
A	Application Table(Master& Transaction Data)
C	Customizing Table, Maintenance only Customer, not by SAP
L	
G,E,S,W	Table for storing temporary data, delivery report System Data

DISPLAY MAINTENANCE:

Display maintenance:
It specifies whether the data should only be displayed or it can also be maintained.
Data maintenance means creation, deletion, and changing.

There are 3 options available.

- Display maintenance allowed
- Display maintenance not allowed
- Display maintenance allowed with restrictions

Data class:
It specifies the physical area of a table inside the database
Depending on the table we are supposed to select the required option.
The options available are,

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

STRUCTURES IN ABAP:

- It is a Data Type in DDIC, composed of data element, table type and tables.
- Structures are similar to tables, except it doesn't have Primary key or technical characteristics.
- These are reusable components, it is used to define the same work area in multiple programs.
- A Structure can be nested within another structure.

Two options for including structures in tables:

- ❖ Include Structure
- ❖ Append Structure

Include	Append
1. This option is used only with custom Tables.	1. This option is used only by SAP Tables.
2. These Include structure are reusable by Multiple tables.	2. These are not reusable.
3. Just give Field name as .Include and Data Element as Structure Name.	3. Click on 'Append Structure' Button To create Structure for add a field.

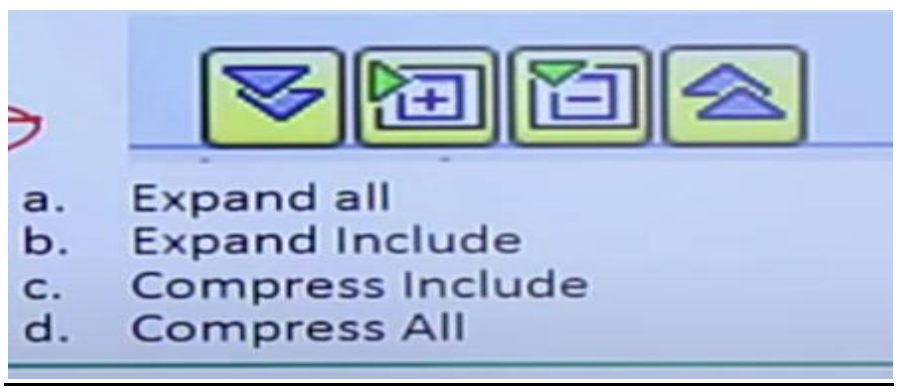
STEPS TO CREATE INCLUDE STRUCTURES:

Step-1:

- Go to SE11 T-code, Select Data Type and provide a name, Then click on Create.
- Select 'Structure' and provide description
- Specify the fields and data elements, then Save and activate

Step -2:

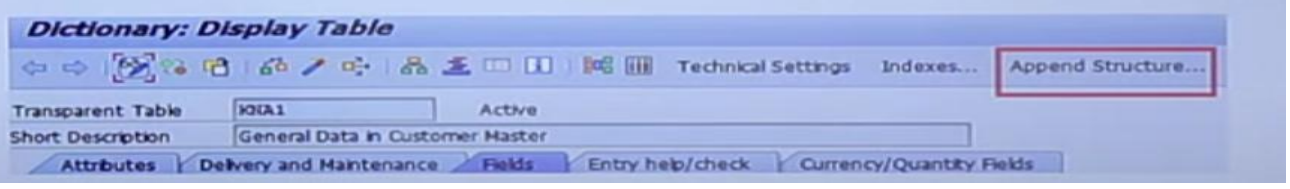
- Open any table which is already created.
- Provide the field name as '.include', data element as structure name which is create above.
- Then Enter, the fields will be automatically copied.



EXAMPLE ON APPEND STRUCTURE:

Add a custom field 'MNAME' (middle name) to the standard SAP table 'KNA1'

1. Open the table kna1 in display mode
2. Click on 'append structure' button
3. A pop-up is displayed, click on create icon
4. Provide the append name 'zapp1' and press enter
5. Provide the field name as 'mname' and data element as 'zmname'
6. Save, activate and click on back
7. Check the field in the table, it will be available at the bottom



CURRENCY AND QUANTITY FIELDS:

- Currency and quantity fields are used to store currency amounts data and quantity data respectively.
- So, for every currency and quantity field we have to specify the corresponding units like INR or USD or EURO etc. or KGS or EA or PC etc.
- **CURR:** It is the data type which is used to store the currency amount or price fields.
- **CUKY:** It is the data type which is used to store the corresponding units(INR or USD or EURO) for the currency fields.
- **QUAN:** It is the data type which is used to specify the quantity for a material or stock.
- **UNIT:** It is the data type which is used to specify the units(KGS or EA or PC) for the quantity fields.

Example:

- Open any table which is already created.
- Add the below fields
- Click on Currency/Quantity fields tab and specify Ref Tab Name and Ref Field Name.

Field	DataElement	DataType	Ref.Table	Ref.Field
Amount	Zamount	CURR(15,2)	ZCUST_TABLE	Zamount_units
Amount_Units	Zamount_units	CUKY(5,0)		
Quantity	Zquantity	QUAN(5,2)	ZCUST_TABLE	
Quantity_Units	Zquantity_units	UNIT(3)		Zquantity_units

- Save and activate the table

FOREIGN KEYS;

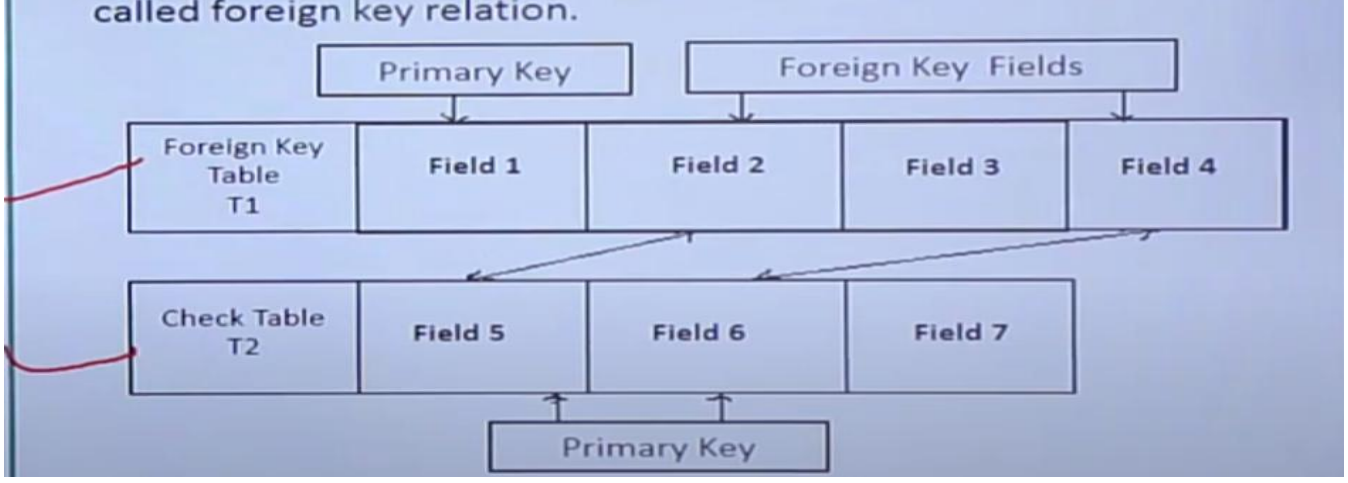
- Foreign keys are used to establish relationships between various tables in ABAP Dictionary.
- Value checks can be created for input fields with the help of foreign keys
- A foreign key links two tables by assigning the foreign key fields of one table to the primary key fields of another table.

Table validation is a concept through which we can restrict invalid entries in a table. Possible methods/concepts for table validation:

1. Field Level validations: We can validate entries at field level with the help of check table concept.
2. Domain level validations: We can restrict entries at the domain level with the help of fixed values of the domain and value table of the domain.

CHECK TABLE:

- A table which stores master data is called a check table.
- A table which is linked with check table for validating its own data is called foreign key table.
- A relation between two or more tables for validating the data is called foreign key relation.



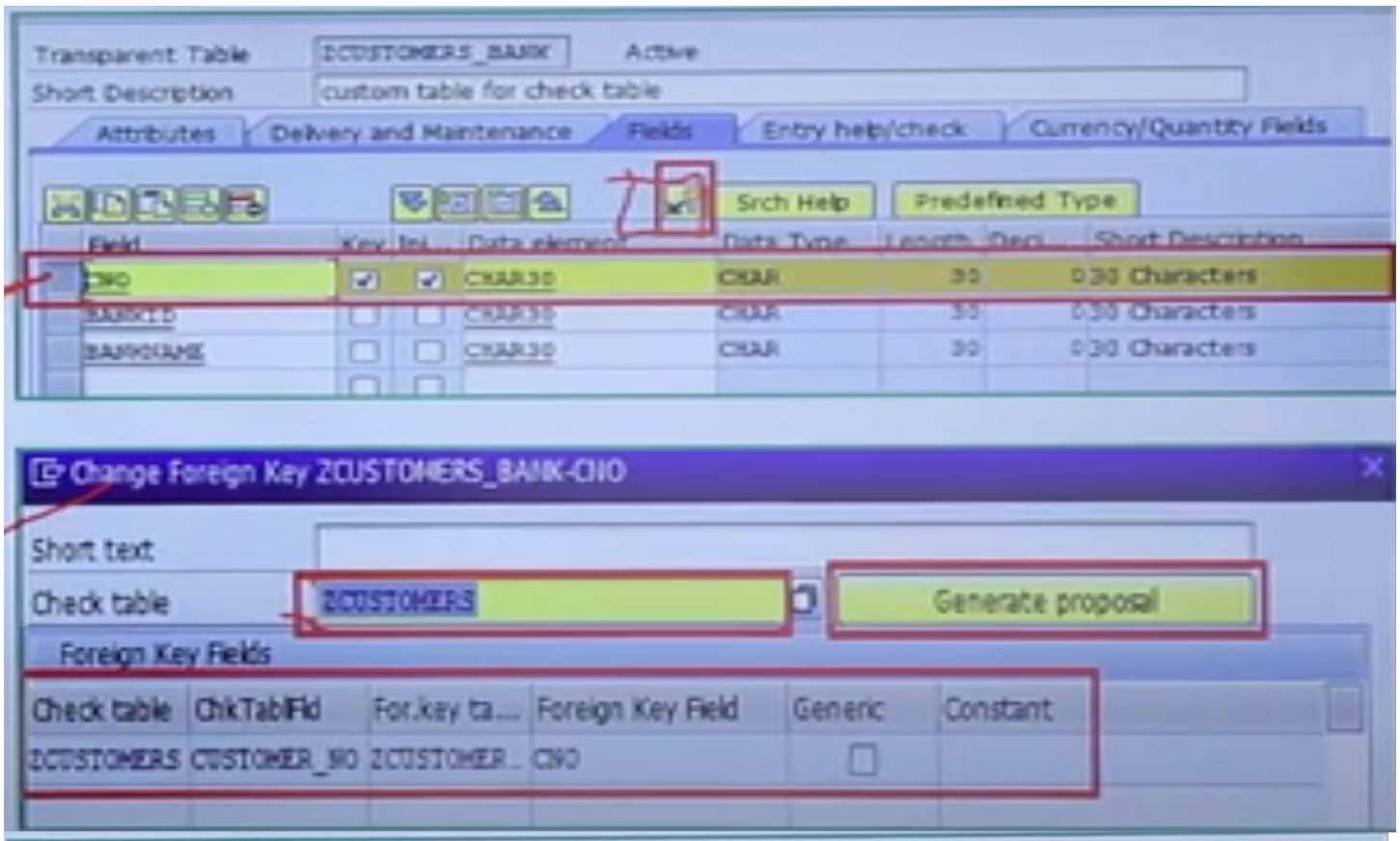
- **STEPS TO CREATE FOREIGN KEY RELATION:**

Step 1: Create a table by name 'zcustomers' with field's customer number and customer name, also create some records.

Step 2: Create another table by name zcustomers_bank with fields cno, bankid and bankname.

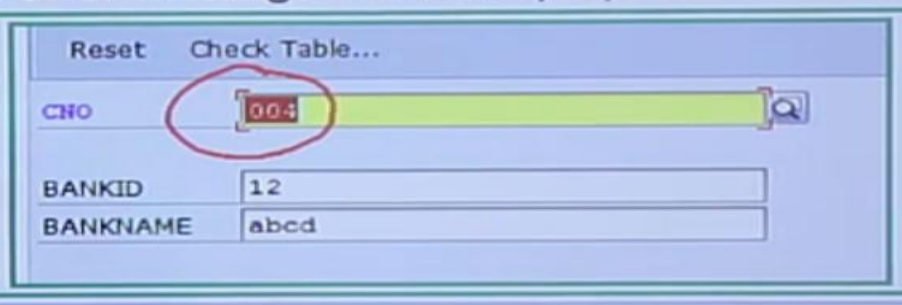
Step 3: Create the foreign key relation as below

1. Open the table zcustomer_bank
2. Select cno field
3. Click on foreign keys button or Icon
4. Provide the check table name as zcustomer(master data table)
5. Click on generate proposal button
6. Click on copy button
7. Save and activate



Step 4: Click Unit testing

- on utilities -> table contents -> create
- Enter the invalid customer number and click on save
- The error message will be displayed



Entry 004 does not exist in ZCUSTOMERS (check entry)

VALUE TABLE:

- A table name defined at the domain level so that all the table fields will be referring to the domain will be checked or validated with a single table called as value table.
- The main advantage of value table is to automate the system for foreign key relation i.e. the system will automatically display check table name to generate foreign key proposal.
- If you try to define a foreign key for a field that points to this domain, the value table of the domain is proposed as the check table for the foreign key.

STEPS TO CREATE VALUE TABLE:

STEP1 : Create a table zzcustomer with field cno and cname, also create some records

STEP2 : Define the check table name at domain level as below.

Now check table is called as value table

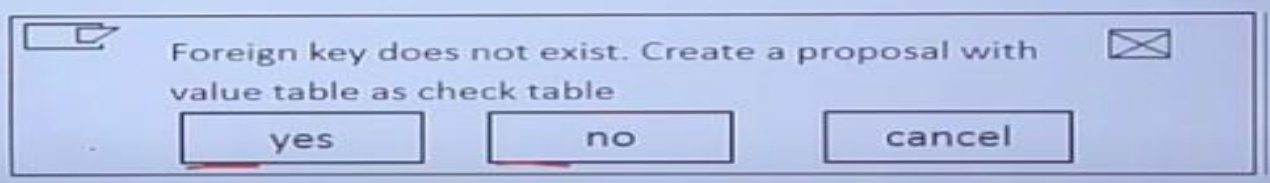
1. Go to the domain zcno
2. Click on value range tab
3. Provide the value table as zzcustomer
4. Save and activate, domain and table

STEP3 : Create another table by name zzcustomer_bank with fields cno, bankid, bankname

---

STEP4 : Maintain the foreign key relation as below

- Open zzcustomer_bank table
- Select cno field and click on foreign key button
- A pop-up displayed as below with value table to be proposed
- for check table



- If we click on 'yes', foreign key relation will be automatically displayed
- If we click on 'no', it will not be displayed

Difference between check table and value table:

Check Table	Value Table
1. It is defined at field level.	1. It is maintained at Domain level.
2. FK relation is not automatic, that means we must enter check table name.	2. FK relation is automatic, that means once we maintain value table , FK relation automatically generated.

TABLE MAINTENANCE GENERATOR:

- It is a standard SAP program created in the form of function modules to maintain mass or bulk amount of data instead of maintaining each and every record.
- Maintenance means creation, deletion and modification.
- It is also used for validating the table data using the concept of events.

STEPS FOR TABLE MAINTENANCE GENERATOR:

Steps :

1. Create any table by name 'zcust' with mandt, cno, cname and land1
2. Save and activate the table
3. Click on utilities -> select table maintenance generator
4. Provide the details as below
5. Authorization group = &NC&
6. Function group = zcust (table name)
7. Maintenance type as one step
8. Click on find screen number button
9. A pop-up is displayed, just press enter
10. The screen number will be automatically proposed
11. Click on create icon
12. The TMG will be created in the form of function modules

ACTIVATING TMG AND UNIT TESTING:

Activating TMG:

- Go to SE80
- Select function group from the list
- Provide function group name (table name) and press enter
- Right click on function group name and select activate

Unit testing:

- Go to SM30
- Provide the table name
- Click on 'maintain' button
- Enter customer number, name and country for a single record or bulk amount of records.
- Click on save, the data will be saved

VIEWS IN DDIC:

- A view is a collection of fields from multiple tables, It doesn't store any data.
- Whenever a view is executed, it displays the data by selecting from multiple tables.
- To create a view, we must join tables.
- To join the tables , there must be at least one common key field.

JOINS:

- It is a concept to link two or multiple tables for displaying the data.

There are two joins

- Inner join
- Outer join

INNER JOIN:

- In this type of join, only the matching record b/w 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 is no matching record, the data will be displayed as blank values.

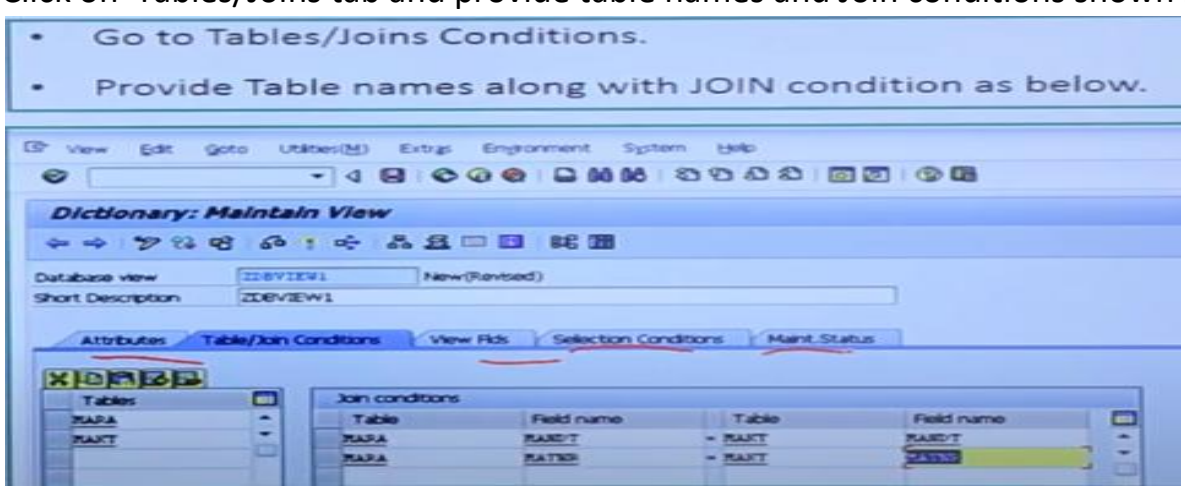
TYPES OF VIEWS:

There are four types of views

1. **Database view:**
    - A view created on two or more tables by clubbing the fields using inner join.
    - In this view, we can only read the data and can not do any maintenance operation.
    - Database uses only inner join, so that it selects only the matching records b/w tables.

    Steps to create Database view

- Go to SE11 and select the view and name it.
- Click on create and select Database view and click copy button.
- Click on Tables/Joins tab and provide table names and Join conditions shown in picture below



- Now click on view Fields tab and then click on Table fields button.
- Double click on 1st table and select fields and then click on copy button
- Similarly double click on 2nd table and select the fields.
- Now go to selection condition tab, provide the condition.
- Save, check and activate.
- Now click on content icon and then click on Execute button.

2. **Projection view:**
    - A view is created on single table is called “Projection View”.
    - Main advantage of using this project view is, we can project only required or limited fields by filtering the unwanted fields.
    - We can read and maintain the data, since the view is created on single table.
    - From database point of view, projection views improve the system performance.

    Steps to create projection view

    - Go to SE11 and select view and name it.
    - Click on create and select the **projection view** and click on copy button.
    - Provide short description and give table names.
    - Click on table fields button and select the fields.
    - Click on Maintenance status tab.
    - Select ‘Read and change’ , then save , check and Activate.
    - Click on contents icon, click on execute and the data will be displayed.

3. **Help view:**
    - A view created on two or more tables for ‘search-helps’ is called Help view.
    - It uses outer join concept
    - Here we can read the data, but cant maintain the data.

- we can not execute help views directly, because they are designed specially for ‘search-helps’.

Steps to create help view:

- Go to SE11 and select view and name it and click on create button.
- Select the Help view and click on copy button.
- Provide short description and give the table and press enter
- Now click on relationship button, select and click on copy button.
- Join condition automatically proposed by system.
- Go to view fields tab.
- Now click on table field’s button and select table1 and click on choose button and select the fields and click on copy button.
- Similarly, select table2 and select required fields.
- Go to selection conditions tab and give the condition.
- Click on save, check and activate.

4. **Maintenance view:**
    - A view created on two or more tables using inner join is called Maintenance view.
    - In this view, we can read and maintain the data.
    - Maintenance view should not be created on standard tables. Because the data inconsistency problem will occur when we change the records.
    - It can be created on custom tables (Z/Y tables).

    Steps to create Maintenance view:

    - Go to SE11 and select view and name it and click on create button.
    - Select the Maintenance view and click on copy button.
    - Provide short description and give the table and press enter
    - Now click on relationship button, select and click on copy button.
    - Join condition automatically proposed by system.
    - Go to view fields tab.
    - Now click on table field’s button and select table1 and click on choose button and select the fields and click on copy button.
    - Similarly, select table2 and select required fields.
    - Go to selection conditions tab and give the condition.
    - Go to maintenance status tab, select **Read, change, delete and insert** button.
    - Click on save, check and activate.
    - Click on **Utilities**, →**Table Maintenance generator**.
      Provide details of Authorization group, function group and Maintenance type as ‘one step’ and Overview screen as ‘1’.
    - Click on create button and then click on Save.
    - Go to SM30
    - Provide the view name and click on maintain button.

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.

- two different kinds of data elements:
  1. Elementary types
  2. Reference types

Field Labels: Field labels are used to assign text information to data elements

- 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.
- Two types of Field labels:
  1. Short, medium, and long fields
  2. Header fields

STEPS TO CREATE DATA ELEMENTS:

- Go to SE11 and select data type and name it
- Click on create and select data element
- Provide short description and select the type of data element.
- If Elementary type, then either provide a domain name or predefined type.
- Click on fields label tab, specify short, medium, long and heading labels.
- Save and activate it.
- Then we can use those data elements.

TYPE GROUP:

- Type group is another repository object of ABAP Dictionary
- It allows 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.

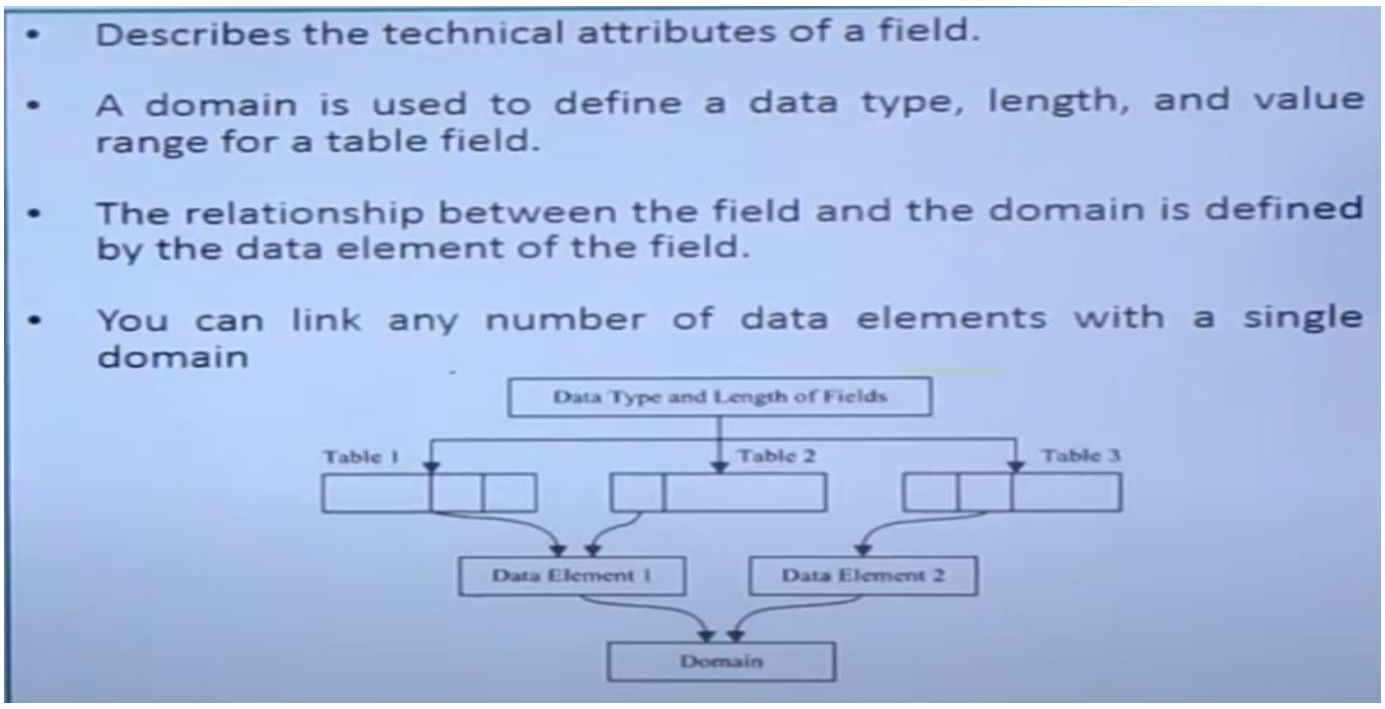
STEPS TO CREATE A TYPE GROUP:

- Go to SE11 and select **Type Group.**
- Enter the type group name and click on create.
- 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 i.e; ZTYPE, Can be used in an ABAP program.
  Ex:
  REPORT ZTEST
  TYPE-POOLS ZTYPE
  DATA: NAME TYPE ZTYPE_NAME,
  AGE TYPE ZTYPE_AGE.

DOMAINS:



Steps to create a domain:

- Go to SE11
- Select domain and give it a name, then click on create
- Provide short description
- Provide data type as CHAR and length as 10
- Press enter, save, check and activate.

SEARCH HELPS:

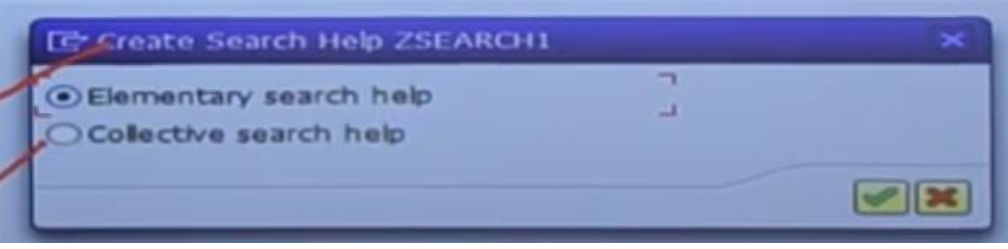
- Search helps is an another repository object of ABAP Dictionary.
- They are used to display all the possible values for a field in the form of a list.
- It provides input helps F4 for different fields.
- You can select the values which are to be entered in the fields from the list.

TYPES OF SEARCH HELPS:

1. ELEMENTARY SEARCH HELP
2. COLLECTIVE SEARCH HELP
3. APPEND SEARCH HELP

STEPS TO CREATE SEARCH HELP:

- **Step 1** – Go to transaction SE11. Select the radio button for Search help. Enter the name of the search help to be created. Let's enter the name ZSRCH1. Click on the Create button.
- **Step 2** – The system will prompt for the search help type to be created. Select the Elementary search help, which is default. The screen to create elementary search help as shown in the following screenshot appears.
- **Step 3** – In the selection method, we need to indicate whether our source of data is a table or a view. In our case it happens to be a table. The table is ZCUSTOMERS1. It is selected from a selection list.
- **Step 4** – After the selection method is entered, the next field is the Dialog type. This controls the appearance of the restrictive dialog box. There is a drop-down list with three options. Let's select the option 'Display values immediately'.



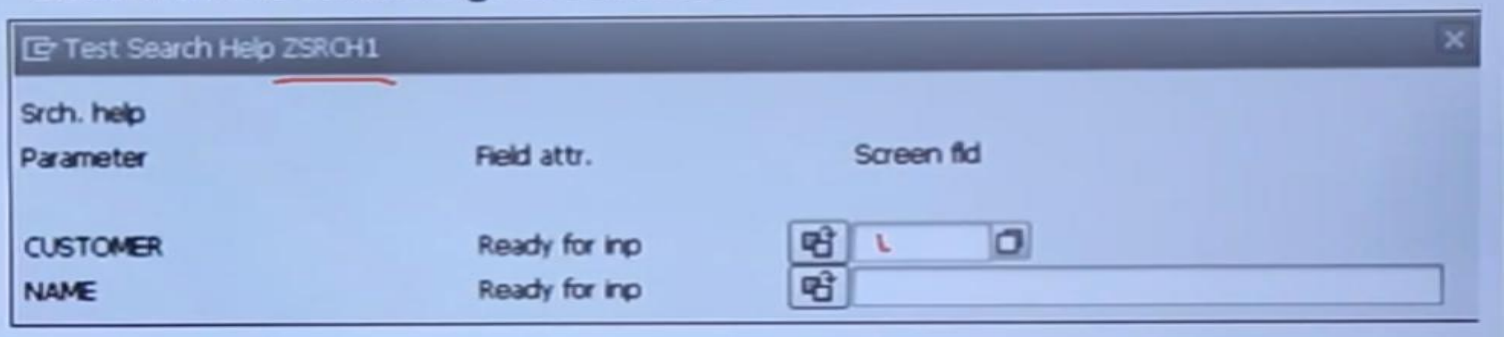
- **Step 5** – Next is the parameter area. For each Search help parameter or field, these column fields have to be entered as per the requirements.

- **Search help parameter** – This is a field from the source of data. The fields from the table are listed in the selection list. The fields participating in the search help would be entered, one field in each row. Let's include the two fields CUSTOMER and NAME. How these two fields participate is indicated in the rest of the columns.

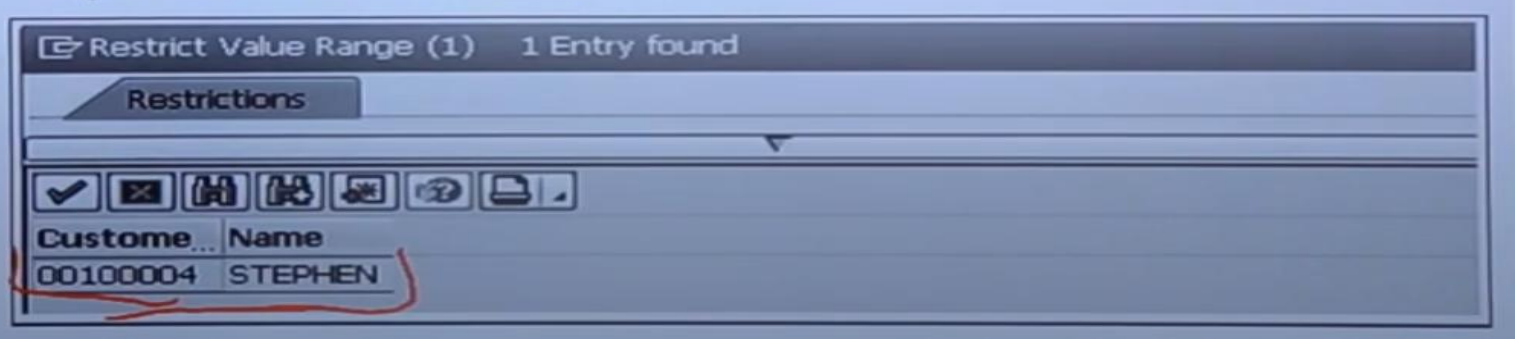
Parameter

Search help parameter	IMP	EXP	LPos	SPos	SDis	Data element
CUSTOMER	☑	☑	1	1	☐	ZCUSTNUM
NAME	☑	☐	2	2	☐	ZCUSTNAME
	☐	☐			☐	
	☐	☐			☐	

- **Step 6** – Perform a consistency check and activate the search help. Press F8 to execute. The 'Test Search Help ZSRCH1' screen appears as shown in the following screenshot.



- **Step 7** – Let's enter the number 100004 in the CUSTOMER's 'Ready for inp' screen field. Press Enter.

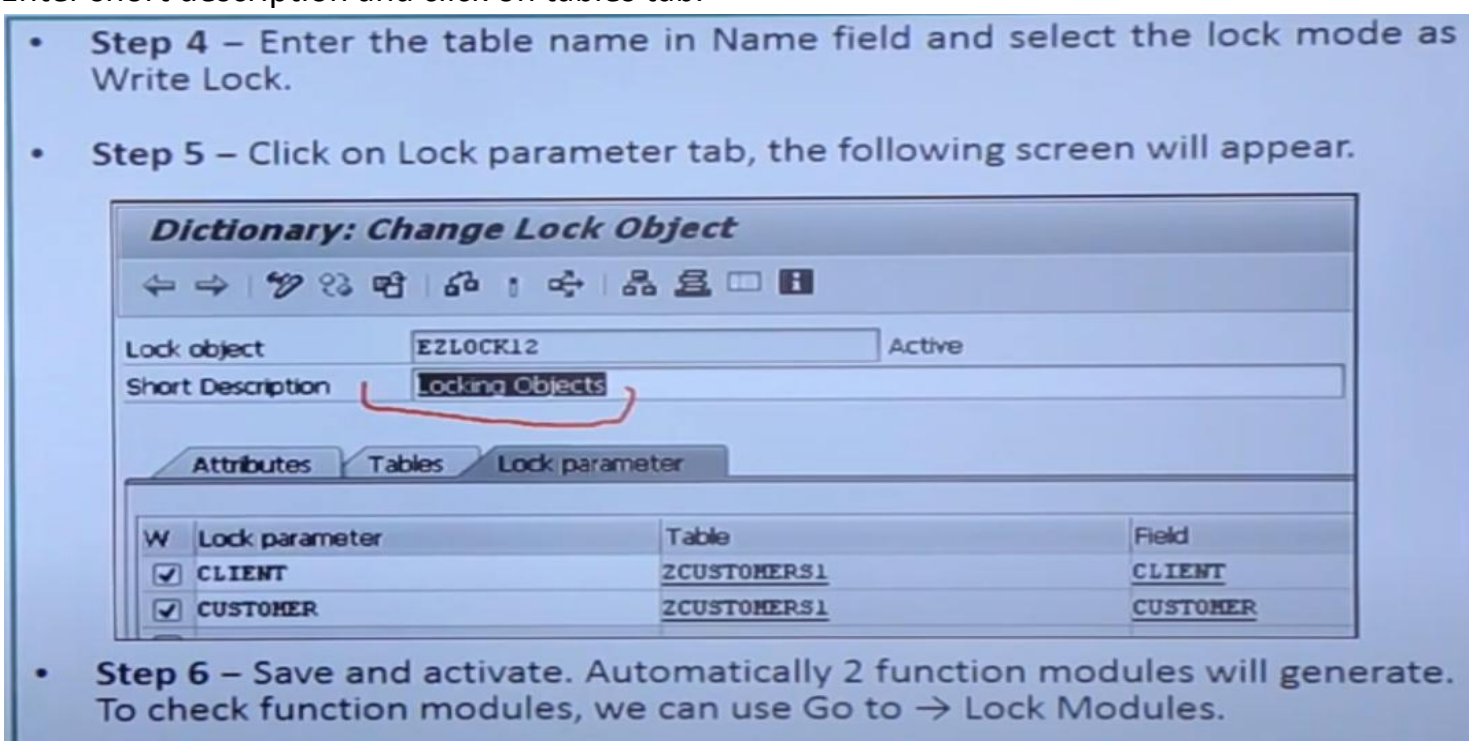


LOCK OBJECTS:

- Lock objects is a feature offered by ABAP dictionary.
- It is used to synchronize access to the same data by more than one program
- Lock objects are used in SAP to avoid the inconsistency when data is inserted into or changed in the database.
- Tables whose data records are to be locked must be defined in a lock object, along with their key fields.

STEPS TO CREATE A LOCK OBJECT:

- Go to SE11 and click on LOCK OBJECT
- Name object starting with ‘E’ and click on create button.
- Enter short description and click on tables tab.



- **Step 7** – Click Lock Modules and the following screen will open.

Repository Info System: Function Modules Find (2 Hits)

Function group Function Module Name	Function group short text Short text for function module
/1BCDWBEN/TEN0000	xRPM 4.0 Demand Planning
DEQUEUE_EZLOCK12	Release lock on object EZLOCK12
ENQUEUE_EZLOCK12	Request lock for object EZLOCK12

- The lock object is created successfully.
- The key fields of a table included in a Lock Object are called lock arguments and they are used as input parameters in function modules.
- These arguments are used to set and remove the locks generated by the Lock Object definition.

INTERNAL TABLES AND WORKAREA:

INTERNAL TABLE:

- An internal table is a temporary table that contains the records of an ABAP program while it is being executed.
- An internal table exists only during the runtime of an SAP program.
- An IT is declared in an ABAP program when you need to retrieve data from database tables.
- Internal table is stored in rows and columns.
- The individual records of an internal table are accessed with an index or a key.
- The records of the internal table are discarded when the execution of the program is terminated.

SYNTAX FOR INTERNAL TABLE:

Syntax to Create an Internal Table :

- DATA<ITABNAME> TYPE TABLE OF <DB TABLE NAME>
- DATA<ITABNAME> TYPE TABLE OF <USER-DEFINED TABLE>
- DATA<ITABNAME> TYPE TABLE OF <DB.TABLE> OCCURS 0 WITH HEADER LINE.

```
Report ZEX_PRG.
* Data Declarations
DATA: I_ZCUST_TABLE TYPE TABLE OF ZCUST_TABLE.
DATA: WA_ZCUST_TABLE TYPE ZCUST_TABLE.
Select * from ZCUST_TABLE into Table I_ZCUST_TABLE.
Loop At I_ZCUST_TABLE into WA_ZCUST_TABLE.
   Write : / WA_ZCUST_TABLE-- CUSTNO,
             WA_ZCUST_TABLE-- CNAME,
             WA_ZCUST_TABLE-- CITY,
             WA_ZCUST_TABLE-- GENDER.
Endloop.
```

EXAMPLE OF USING INTERNAL TABLE:

Step 1 – Open the ABAP Editor by executing the SE38 transaction code. The initial screen of ABAP Editor appears.

Step 2 – In the initial screen, enter a name for the program, select the Source code radio button and click the Create button to create a new program.

Step 3 – In the 'ABAP: Program Attributes' dialog box, enter a short description for the program in the Title field, select the 'Executable program' option from the Type drop-down menu in the Attributes group box. Click the Save button.

Step 4 – Write the following code in ABAP editor.

Step 5 – Save, activate and execute the program as usual.

```
REPORT ZINTERNAL_DEMO.
TYPES: BEGIN OF CustomerLine,
Cust_ID TYPE C,
Cust_Name(20) TYPE C,
END OF CustomerLine.

TYPES mytable TYPE SORTED TABLE OF CustomerLine
WITH UNIQUE KEY Cust_ID.
WRITE:/'The mytable is an Internal Table'.
```

WORKAREA:

- An internal table is accessed by using the concept of a work area.
- A work area is a temporary memory space that helps in reading and modifying the data of an internal table, line by line.
- The work area must have the same structure as that of the associated internal table.
- By default no operation can be done on internal tables. And for that purpose, we need Work Area.
- Once the operation is finished on work area, then we modify the internal Tables.

OPERATIONS ON INTERNAL TABLE:

There are multiple operation that can be performed on internal tables

1. **APPEND:**
   It is used to add a single record from WORKAREA to INTERNAL TABLE.
   This record is always added at the bottom.
   SYNTAX:
   APPEND <WA> TO <ITAB>.

2. **INSERT:**
   It is used to insert a record from WORKAREA to INTERNAL TABLE at a specified location.
   SYNTAX:
   INSERT <WA> INTO <ITAB> INDEX <INDEX NO>.
3. **SORT:**
   It is used to sort the data of internal table in ascending or descending order.
   By default it is ascending order.
   SYNTAX:
   SORT <ITAB> BY F1 F2 F3 …. <ASCENDING/DESCENDING>
4. **DESCRIBE TABLE:**
   It is used to find the total no of records in an internal table.
   SYNTAX:
   DESCRIBE TABLE <ITAB1> LINES <VARIABLE>
5. **READ TABLE WITH INDEX:**
   It is used to read a single record from ITAB into WA specified by index no.
   SYNTAX:
   READ TABLE <ITAB> INTO <WA> INDEX <INDEX NO>
6. **MODIFY:**
   It is used to modify single or multiple records based on condition.
   SYNTAX:
   MODIFY <ITAB> FROM <WA> TRANSPORTING F1 F2 F3 …..
   Where F1 = value and F2 = value
7. **DELETE:**
   This statement is used to delete single or multiple records based on condition.
   SYNTAX:
   DELETE <ITAB> INDEX <INDEX NO>
8. **DELETE ADJACENT DUPLICATES:**
   This statement is used to delete adjacent records from the ITAB.
   SYNTAX:
   DELETE ADJACENT DUPLICATES FROM <ITAB>
   COMPARING F1 F2 F3 ALL FIELDS.
9. **CLEAR:**
   This statement is used to delete the data from WA
   SYNTAX:
   CLEAR <WA>
10. **REFRESH:**
    This statement is used to delete the data from ITAB.
    SYNTAX:
    REFRESH <ITAB>
11. **FREE:**
    This statement is used to delete the data from WA and ITAB
    SYNTAX:
    FREE <WA/ITAB>
12. **APPEND LINES OF:**
    This statement is used to append to data from ITAB into another ITAB based on selection.
    The data will append at bottom in the 2nd ITAB
    SYNTAX:
    APPEND LINES OF <ITAB1> FROM <INDEX NO1> TO <INDEX NO2> INTO <ITAB2>.

13. **INSERT LINES OF:**
    This statement is used to insert the data from one ITAB into another ITAB at specified location based on selection.
    SYNTAX:
    INSERTLINES OF <ITAB> FROM <INDEX NO1> TO <INDEX NO2> INTO <ITAB2> INDEX <INDEX NO>
14. **MOVE ITAB1[] TO ITAB2[]:**
    This statement is used to move the entire data from one ITAB1 to ITAB2
    SYNTAX:
    ITAB1[] = ITAB2[]
15. **COLLECT:**
    This statement checks whether the WA record already exists with the
    SYNTAX:
    COLLECT <WA> INTO <ITAB>
    Checks whether the WA already exists with same key
    If yes, ADD NUMERICAL FIELDS
    If no, APPEND a new record

TYPES OF INTERNAL TABLES:

STANDARD	SORTED	HASHED
• These are the default internal tables which are created by us. • We use either key operation (or) index operation to read a record. • We use either linear search (or) Binary search for reading record. • If use Binary Search, the response time will be • Resp.Time = Log(N). • We can append , insert the records whenever we want. • We can sort the data based on our own conditions.	• These are special ITAB's where the data is automatically sorted whenever a new record is added. • We use either key(or) index operation to read a record. • We use only Binary search for reading a record, because the data is automatically sorted. • The Response Time will be same. • The main disadvantage is , we can not sort ITABS based on our conditions, because the data is already sorted.	• These are also special type of ITAB which should be used when working with large data sets(Bulk amount of data). • Here, we use only Key operation, but not the index operation. • It uses Hashed algorithm for reading a record. • The Resp.Time is always fixed regardless of the total no of records. • In Real-Time , we use hashed ITAB's only whenever we work with server to server communication like transferring the data from ABAP to BI server.

DATABASE:

SAP R/3 is independent of database being used. i. e; SAP can use any database to store the data.

We use databases to store mass amount of data

Whenever we want to store the data into database, we need to use the respective database language statements.

R/3 INTERFACE LAYER:

It is an interface which is given by standard SAP software which is used to convert the ABAP language statements into respective database statements and vice versa.

There are two types of statements:

1. OPEN SQL Statements

SYNTAX OF OPEN SQL STATEMENTS

INSERT : INSERT <DB TABLE> FROM <WA>.
INSERT <DB TABLE> FROM TABLE <ITAB>.
MODIFY : MODIFY <DB TABLE> FROM <WA>.
MODIFY <DB TABLE> FROM TABLE <ITAB>.
UPDATE: UPDATE <DB TABLE> FROM <WA>.
UPDATE <DB TABLE> FROM TABLE <ITAB>.
DELETE: DELETE <DB TABLE> FROM <WA>.
DELETE <DB TABLE> FROM TABLE <ITAB>.

PRE-REQUISITIES FOR OPEN SQL STATEMENTS

1. The structure of internal table/work area must be same as database table.
   Ex: Data: i_kna1 type table of kna1.(correct)
   Data: i_kna1 type table of ty_kna1.(incorrect)

2. Always work with internal tables rather than work areas to insert /update/modify data in database table

2. NATIVE SQL Statements

SELECT STATEMENT:

In Open SQL, the SELECT statement and its various clauses are used to read data from database tables. The syntax to use the SELECT statement to read the data from database tables is:

```
SELECT      <selected_result>
INTO        <target_area>
FROM        <source_database_tab>
[WHERE      <where_condition>]
[GROUP BY   <fields>]
[HAVING     <having_condition>]
[ORDER BY   <fields>].
```

SYSTEM VARIABLES:

SYSTEM VARIABLES: There are two system variables which are updated automatically for every open sql operation.

1. **SY-SUBRC**

   It is a system variable which stores the return code of an open sql operation.
   If SY-SUBRC = 0.
   It means the operation is successfully executed.
   If SY-SUBRC NE 0.
   It means the operation failed.

2. **SY-DBCNT**

   It is a system variable which stores the no. of records successfully processed.

EXAMPLE:

Using Select Clause to read a single line :

Syntax:

```
SELECT SINGLE <database_tab_columns> ... WHERE ...
```

Example

```
REPORT ZDATA_ACCESS.
*/Reading the data using SELECT statement
DATA LINE TYPE KNA1.
SELECT SINGLE KUNNR LAND1 NAME1 ORT01 ADRNR
INTO CORRESPONDING FIELDS OF LINE
FROM KNA1 WHERE KUNNR EQ '0000000515'.

 IF SY-SUBRC EQ 0.
 WRITE: / LINE-KUNNR, LINE-LAND1, LINE-NAME1, LINE-ORT01,
 LINE-ADRNR.
 ENDIF.
```

SELECT WITH JOINS:

This statement is used to fetch the data simultaneously from multiple tables.

That means the multiple tables must be joined using one or two fields.

There should be atleast one common field between tables.

There are two types of joins:

1. **Inner join:**
   In this join, only matching records between the tables will be fetched.
   The unmatched records are not selected.

2. **Outer join:**
   In this join, all the records from the left table or first table are selected first.
   If any matching records from the second or third tables, then data will be displayed.
   If there are no matching records, the data will be displayed as blank from second & third tables.

SELECT WITH ALL ENTRIES:

1. This statement is used to replace select with joins, as JOINS statement cannot be used for more than three tables.
2. If we use more than three tables it puts heavy load on the Database, because the data has to be selected by comparing each table in the database server.
3. So it takes the long time for execution.
4. In such cases we go for SELECT FOR ALL ENTRIES.
5. This statement will never put load on the database. Because only two tables (Internal table and database tables) are compared.

```
Select  F1  F2  F3.....
        From <DB. Table1>
        Into table <ITAB1> Where <conditions>.
If ITAB1[] is not initial.
Select  F1  F2  F3.....
        From <DB.Table2>
        Into table <ITAB2>
        For all entries in <ITAB1>
        Where F1 = <ITAB1-F1> AND F2 = <ITAB1-F2>.
Endif.
```

SELECTION SCREENS:

- A selection screen is one of the four types of user dialogs.
- The selection screens are designed whenever a user wants to design a screen meant only for accepting data input.
- Both Single value input or Complex value input can be provided using Selection screen.
- The single value entered in a field is used primarily to control the flow of a program.
- The complex criteria, on the other hand, are used to restrict the amount of data read from the SAP database.(filter)
- The selection screen can be defined with ABAP statements.

For Designing the Selection Screen , we have the following Statements :

1. Parameters .
2. Select – Options.
3. Selection – Screen Command

1. **PARAMETERS:**

   The parameters statement is used to define a single field on the selection screen.
   The variables defined using the parameters statement accepts only single values.
   Parameters are used to create input fields, checkboxes, and radio buttons on the selection screen.

EX ON PARAMETERS.

```
DATA: WA_KNA1 TYPE KNA1.
PARAMETERS P_KUNNR TYPE KNA1-KUNNR.
 SELECT SINGLE * FROM KNA1 INTO WA_KNA1 WHERE KUNNR =
P_KUNNR.
WRITE:/ WA_KNA1 – KUNNR,
              WA_KNA1 – NAME1,
              WA_KNA1 – LAND1.
```

EX ON CHECKBOXES.

```
PARAMETERS P_CHECK AS CHECKBOX.
If P_CHECK = 'X'.
Write:/ 'CheckBox is Selected'.
Else.
 write:/ 'Checkbox is not selected'.
Endif.
```

2. SELECT OPTION:

- Using SELECT – OPTION statement, user can enter a range of values.
- It determines how a selection screen is displayed by entering the values of the fields of the selection table.

Syntax :

- SELECT-OPTIONS <selection_tab> FOR <fld>.
- When the SELECT-OPTIONS statement is executed, an internal table (or a selection table in this context) containing four components, SIGN, OPTION, LOW, and HIGH, is created. These components correspond to the fields of a database table or an internal field in the corresponding program

Example:

EX ON SELECT – OPTIONS :

```
Tables KNA1.
 Select – Options  S_KUNNR for KNA1-KUNNR.
Parameters Download as CheckBox.
Data: I_KNA1 type table of KNA1.
Data: WA_KNA1 type KNA1.
Select   *   from KNA1 into table I_KNA1 where KUNNR in
S_KUNNR.
If  Download = 'X'.
Call Function 'GUI_DownLoad'
Exporting
     FILENAME  = 'C:/ KNA1.txt'
     FILETYPE   = 'ASC'
     WRITE_FIELD_SEPERATOR = 'X'
Tables
      DATA_TAB = I_KNA1.
If SY-SUBRC = 0.
      Message  'DATA SUCCESSFULLY DOWNLOADED' Type 'I'.
Endif.
```

3. **SELECTION SCREEN STATEMENT:**

- Using SELECTION – SCREEN statement, we can customize the layout.
- It can be done with the help of various formatting options provided by the SELECTION-SCREEN statement.
- Various formatting options include setting the layout of parameters, setting the selection criteria for parameters, setting the display of comments, and setting underlines on the selection screen.

The layout designed by a user can be viewed only if the selection screen is called.

MODULARIZATION:

- Modularization techniques enhance the readability and understandability of large ABAP programs.
- It helps to remove redundancy, by reducing same lines of code and function.
- It helps improve structure of an ABAP program.
- Basically, it is a technique of dividing a main program into smaller programs or modules for better reusability and readability.

There are various modularization techniques, such as:

1. Include Programs
2. Function Modules
3. Subroutines

1. **INCLUDE PROGRAMS:**
   - Include programs are global repository objects used to modularize the source code.
   - They allows you to use the same source code in different programs.
   - They also allows you to manage complex programs in an orderly way.
   - To use an include program in another program, use the following syntax of include statement. i. e; INCLUDE <INCL>

The following restrictions must be considered while writing the source code for include programs.

- ❖ Include programs cannot call themselves.
- ❖ Include programs must contain complete statements.

- **SE38 is the T- code to open ABAP Editor.**

Steps :

1. Create the program to be included in ABAP Editor. Remember to set the Type of the program to INCLUDE program.
2. Click the Save button and save the program in a package named ZKOG_PCKG.

```
PROGRAM ZINCLUDED.
WRITE:    / 'Program started by::', SY-UNAME,
          / 'On host:', SY-HOST,
          / 'Date:', SY-DATUM,
          / 'Time:' SY-UZEIT.
```

3. Create another program where the program ZINCLUDED has to be used. In this case, we have created another program named ZINCLUDING and assigned a type for the program, such as Executable program

Now, the coding for the ZINCLUDING program includes the ZINCLUDED program with the help of the INCLUDE statement.

```
INCLUDE ZINCLUDED.
```

2. **FUNCTION MODULES:**

- Function modules are sub-programs that contain a set of reusable statements with importing and exporting parameters.
- Unlike Include programs, function modules can be executed independently.
- SAP system contains several predefined function modules that can be called from any ABAP program.
- The function group acts as a kind of container for a number of function modules that would logically belong together.

Components of FM are:

1. IMPORT: **Input to Function Module**
2. EXPORT: **Output to Function Module**
3. EXCEPTIONS: **Certain type of Error in Function Module**
4. TABLES: **ITAB's which acts as Importing and Exporting**
5. CHANGING:**A variable or WA which acts as Importing & Exporting**

STEPS TO CREATE A FUNCTION MODULE:

- To create function Module, go to Tcode: se37, i.e. function builder. Here you can, display, edit or create a function module.

Steps :

1. Go to transaction SE38 and create a new program called Z_SPELLAMOUNT.
2. Enter some code so that a parameter can be set up where a value could be entered and passed on to the function module. The text element text-001 here reads 'Enter a Value'.
3. To write the code for this, use CTRL+F6. After this, a window appears where 'CALL FUNCTION' is the first option in a list. Enter 'spell_amount' in the text box and click the continue button.

4. Spelling the Amount Amount in words is: FIVE THOUSAND SIX HUNDRED EIGHTY

```
REPORT Z_SPELLAMOUNT.
data result like SPELL.

selection-screen begin of line.
selection-screen comment 1(15) text-001.

parameter num_1 Type I
selection-screen end of line.
CALL FUNCTION 'SPELL_AMOUNT'
EXPORTING
AMOUNT = num_1
IMPORTING
IN_WORDS = result.

IF SY-SUBRC <> 0.
   Write: 'Value returned is:', SY-SUBRC.
else.
   Write: 'Amount in words is:', result-word.
ENDIF.
```

3. SUBROUTINES:-

- Subroutines are a mini program that consists of sequence of statements.
- It is used to prevent redundancy of the statements of ABAP Program.
- It starts with FORM, ends with ENDFORM and in-between can execute variables statements of ABAP Program.

There are two types of Subroutines.

1. Local Subroutines:-
   Definition and implementation is in same program.
2. Global Subroutines:-
   Definition is in one program and implementation is in another program.

Messages in ABAP:

- All the messages is stored in message class.
- A single message class is used for the entire Project.
- Message class is created used SE91 Transaction Code.
- Message ID is assigned to message class to our report.
- Our report can access all the messages in the message class.
- Syntax:- **MESSAGE<message> TYPE <MESSAGE TYPE>**

Type		Description
A	Termination	The message appears in a dialog box, and the program terminates. When the user has confirmed the message, control returns to the next-highest area menu.
E	Error	Depending on the program context, an error dialog appears or the program terminates.
I	Status	The message appears in a dialog box. Once the user has confirmed the message, the program continues immediately after the MESSAGE statement.
S	Error	The program continues normally after the MESSAGE statement, and the message is displayed in the status bar of the next screen.
W	Warning	Depending on the program context, an error dialog appears or the program terminates.
X	Exit	No message is displayed, and the program terminates with a short dump. Program terminations with a short dump normally only occur when a runtime error occurs.

- Status: (S):
  Syntax:- **MESSAGE** 'message' **TYPE 'S'**
- Information: (I):
  Syntax:- **MESSAGE** 'This is an Information message' **TYPE 'I'**
- Error: (E):
  Syntax:- **MESSAGE** 'This is an Error message' **TYPE 'E'**
- Warning : (W): same as error message
  Syntax:- **MESSAGE** 'message' **TYPE 'W'**
- Exit: (X):
  No message is displayed and the program terminates with the short dump.
  Short dumps can be viewed in T-Code ST22
  Syntax:- **MESSAGE** 'This produces short dump' **TYPE 'X'**
- Termination: (A):
  Syntax:- **MESSAGE** 'Termination message' **TYPE 'A'**
  Creating a message:
  Menu GOTO >>> TEST Elements >>> Test Symbols.

REPORTS IN ABAP:

A report is a presentation of data in a specified format and organized structure.

Types of Reports:

1. **Classical Report:**

- The output is displayed in a single list.
- It is created using the output data, in WRITE statement inside a loop.
- We can use various events such as INITIALIS ATION, TOP-OF-PAGE to create a classical report.

- **AT Selection** : screen on value request: - This event is used to provide a search help for an input field. This event is triggered whenever the user click on search help button.

- **AT Selection** : screen on help request: - This event is used to provide help information for an input field. This event is triggered whenever the user click on Help button i.e. F1 button.

- **AT Selection-screen** : This event is used to validate multiple fields of the selection-screen, whereas AT Selection-screen on field is used to validate a single input field.

- **Start of selection** : This event is used to write the original business logic statements i.e all select statements. It is also used to indicate the starting point of a program. It is the default event.

- **Load of Program:** This event is used to load the program into memory for execution. This is internal event, which we cannot see.

- **Initialization:** This event is used to initialize the default values to program variables and selection-screen variables.
  Examples: v_land1 = 'IN'
  v_bukrs = '1001'
  p_werks = 'US'
  p_mtart = 'FERT"

- **AT Selection-screen output:** This event is used to modify the selection screen dynamically based on user actions on the selection screen. This event is triggered (or) executed for every action on the selection screen.

- **AT Selection-screen on field:** This event is used to validate a single input field on the selection-screen.

- **End of selection:** It is used to indicate the start of selection has been ended. It is mainly used with logical database, which are obsolute in ABAP (LDB's are used in old version of ABAP). But LDB's are stilled used in HR-ABAP, so end of selection is mainly used in HR-ABAP. It doesn't have much importance in ABAP, it is used to indicate the start of selection has been ended .

- **Top of page:** This event is use to display constant page heading for all the pages in the output screen. This event is triggered after the first WRITE statement.

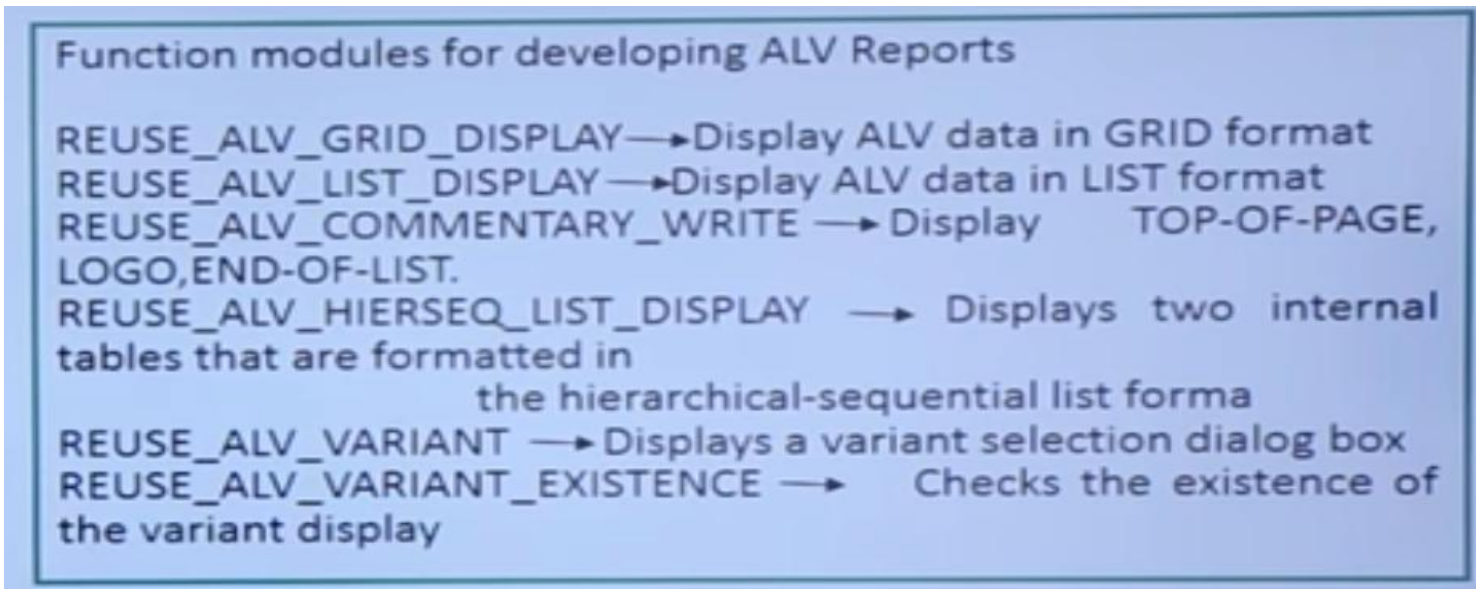
- **End of page:** This event is used to provide constant footer information across the all pages of the list-screen. To display end of page (or) footer information we have to reserve some lines as footer lines using the statement.
  Report <Report Name> Line Count Total Lines(Footer Lines)

2. **Interactive Report**:

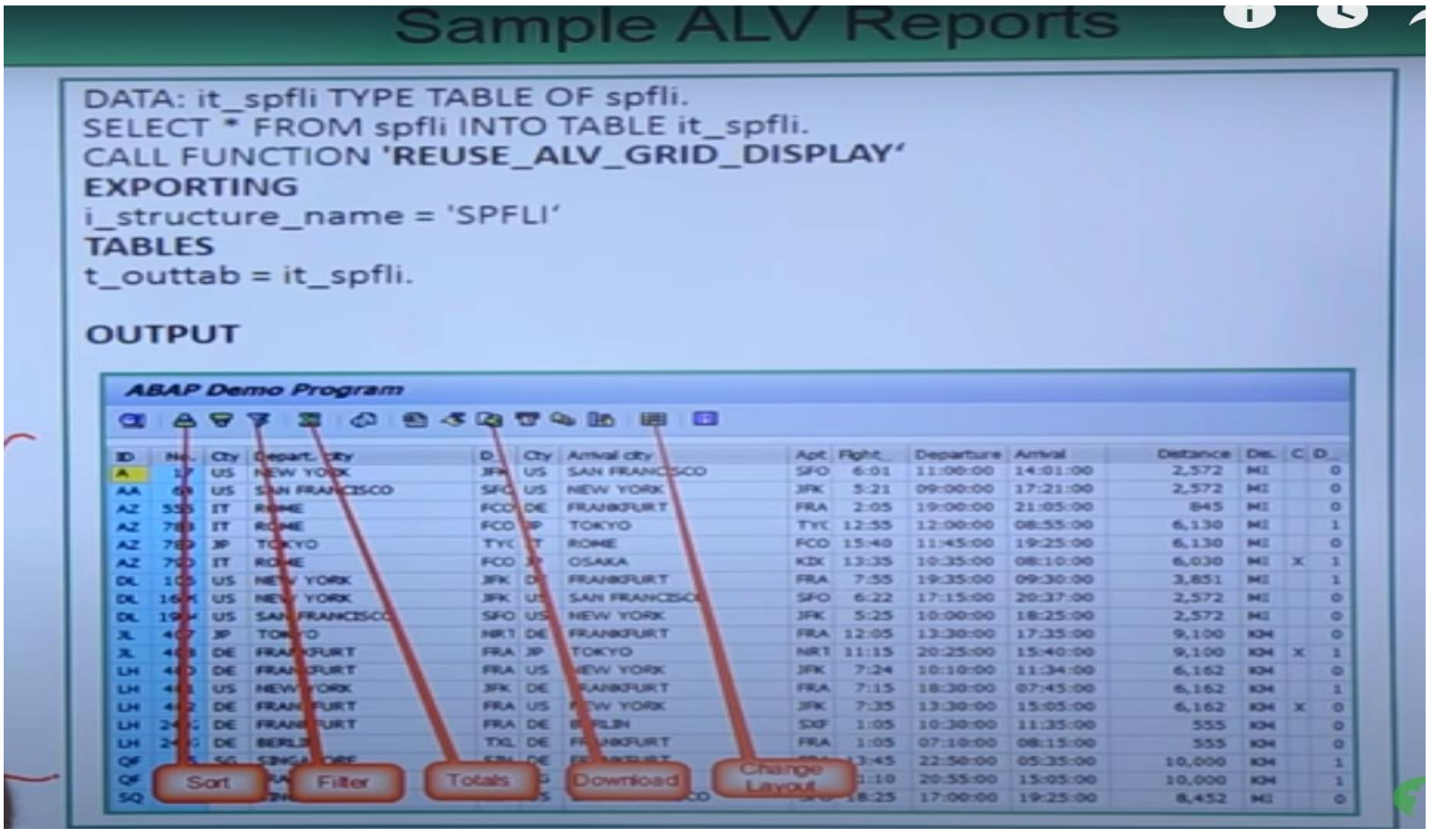
- It allows you to control the retrieval of data.
- It can view multiple lists simultaneously, firstly it displays basic list which starts with number ‘0’.
- The other lists are secondary lists which provide detailed description of data which starts from ‘1 to 20’.
- The list numbers are stored in system variable SY-LSIND
- And if you select any line, selected line data is stored in SY-LISEL.

3. **ABAP List Viewer (ALV) Reports**:

- It is a pre-defined report format in SAP.
- It allows you to perform various functions such as sorting, filtering, arranging and retrieving data.
- Can use various ALV Function Modules to perform functions on the data of report.
- Also can view the output of ALV in a list view or grid view by using **REUSE_ALV_LIST_DISPLAY** and **REUSE_ALV_GRID_DISPLAY** function modules respectively.



SAMPLE OF ALV REPORT:



BUSINESS APPLICATION PROGRAMMING INTERFACE (BAPI):

- It is also a remote function module which is used to communicate between SAP-SAP or SAP-NONSAP servers.
- BAPI uses RFC technology, which is initially a RFC function module.
- BAPI's are defined in a **BUSINESS OBJECT REPOSITORY**(**BOS**) as methods of SAP business object types that carry out specific business functions.

BUSINESS OBJECT REPOSITORY:

- It is a central access point for SAP business object types and their BAPI's.
- It is developed for SAP business workflow.
- Today, we also used it for Archive link, output control and other generic object services.

BUSINESS OBJECTS:

- It is object which is similar to class, which is a group of attributes methods, interfaces, key fields, events for a particular business application or scenario.
- **SWO1** is T-Code for Business object repository.

Uses of BAPI

BAPI'S are used in 3 scenarios.

1. Report generation using BAPI'S
2. Using standard or custom BAPI to communicate between SAP & NONSAP
3. Using BAPI's for uploading data into SAP instead of BDC.

Rules for using BAPI

- Every BAPI should start with BAPI or ZBAPI.
- All the Importing and Exporting parameters should be of Type structures, not the direct Data Types.
- All the structures must start with BAPI or ZBAPI.
- All the Parameters must be pass by Value because BAPI'S doesn't support pass by reference.
- Every BAPI should have a returning Parameter by Name RETURN or BAPIRET2, to display success or error messages.

- There are two types of BAPI:

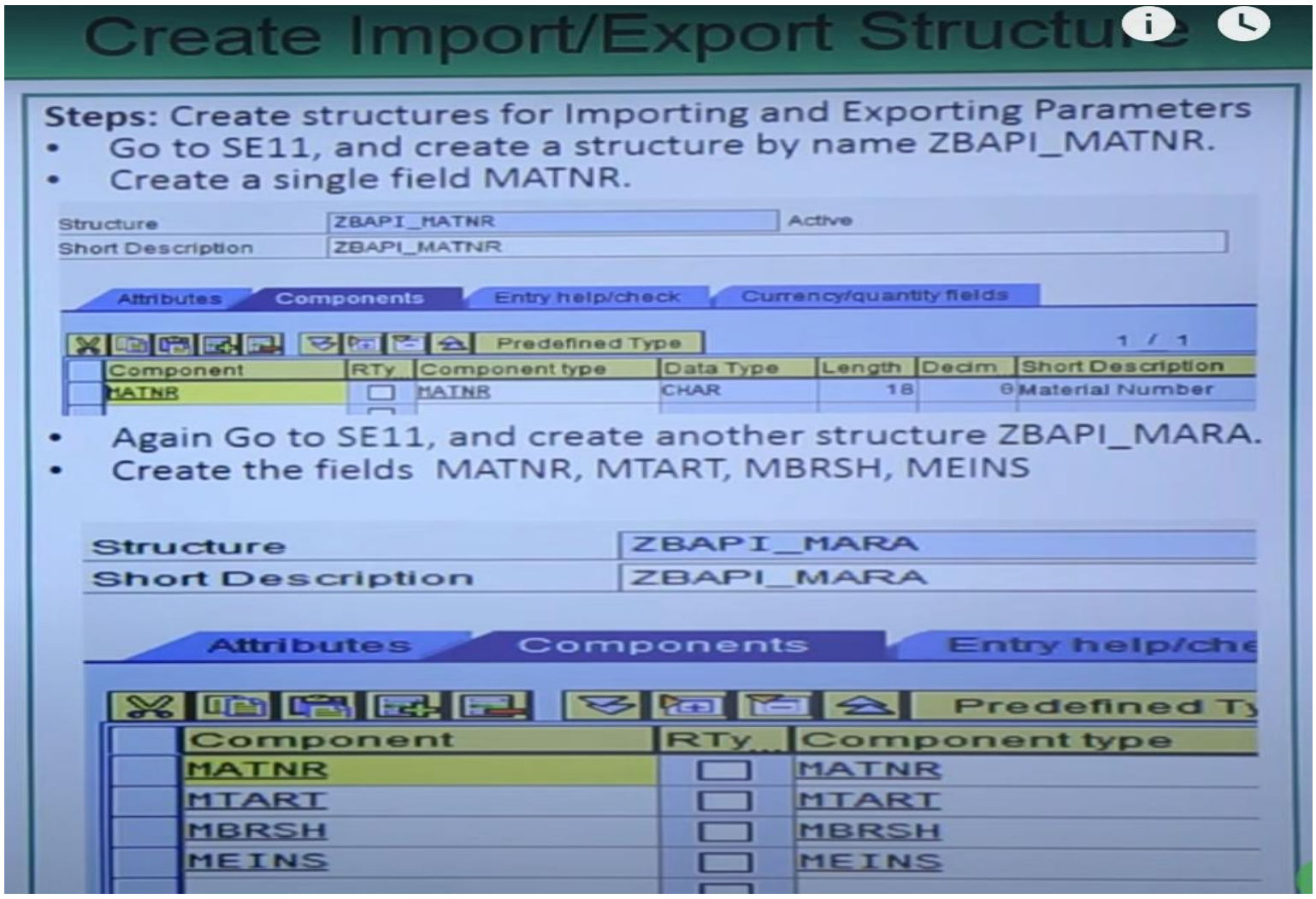
  1. Standard BAPI
  2. Custom BAPI

- Some BAPIs and methods provide basic functions and can be used for most SAP Business Objects. These are called **STANDARDIZED BAPI's.**

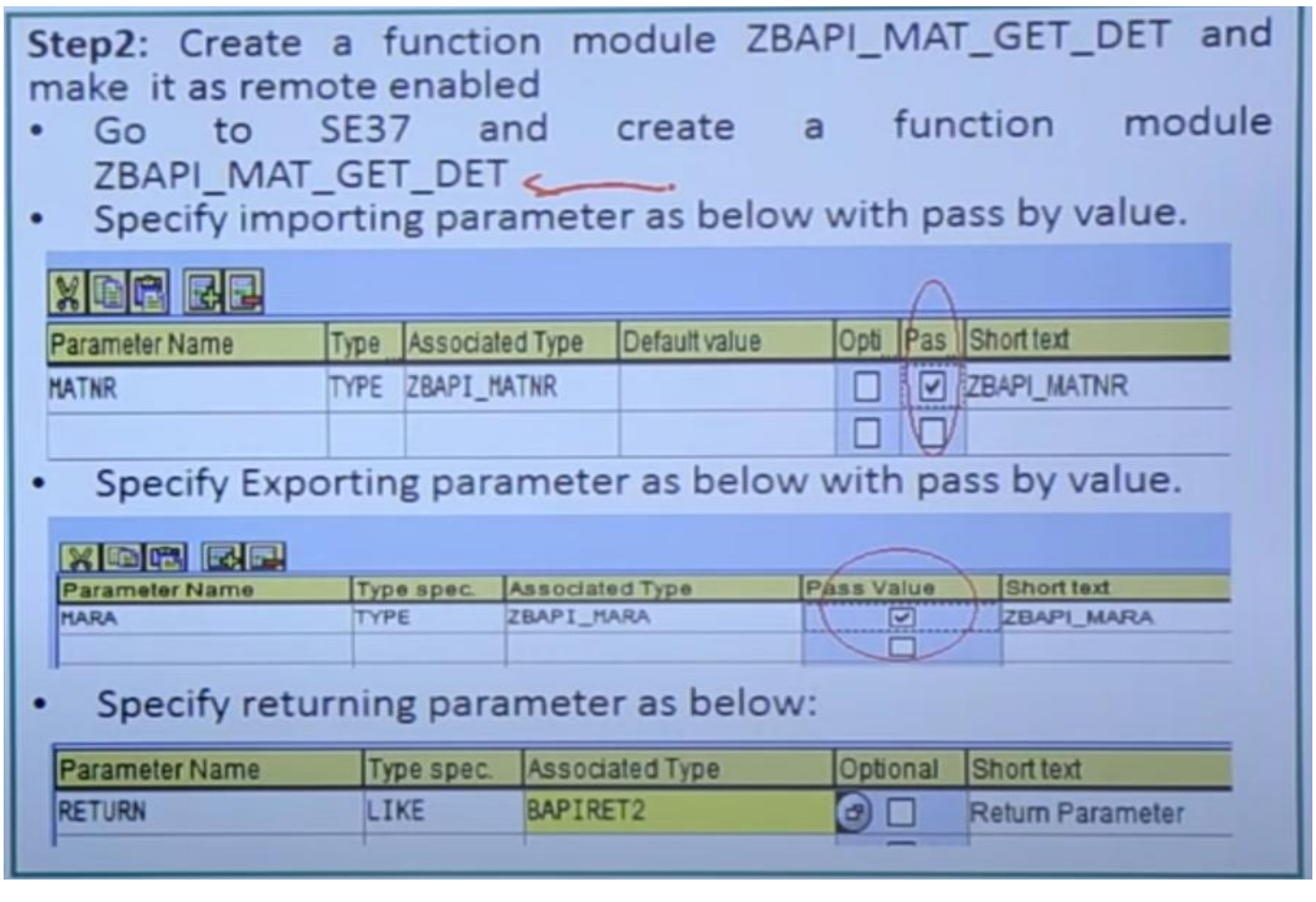
List of Standardized BAPIs:

- BAPIs for Reading Data - GetList() , GetDetail() , GetStatus() , ExistenceCheck()
- BAPIs for Creating or Changing Data- Create() ,Change(),Delete() and Undelete()
- BAPIs for Mass Processing -ChangeMultiple(), CreateMultiple(), DeleteMultiple().

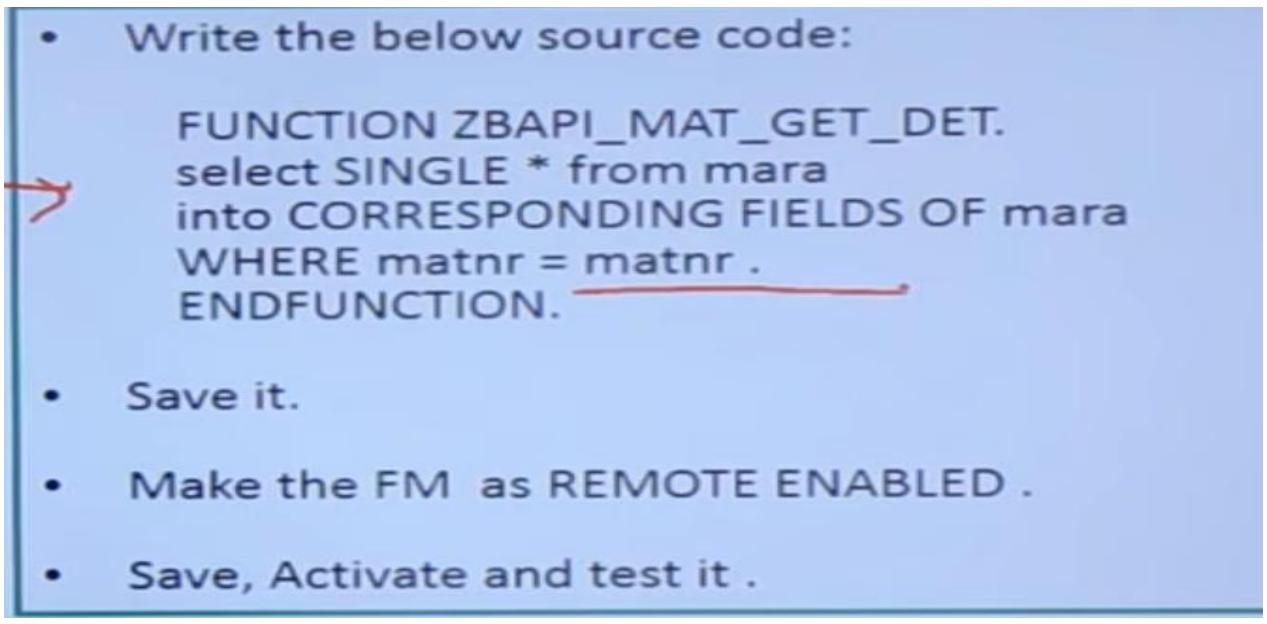
1. Standard BAPI: read above picture
2. Custom BAPI:
    - Import / Export structure



- Create a function module



- Click on Source code and enter the code and remote enabled from ATTRIBUTES tab.



- Create a Business object:
  - (a) Go to **SWO1** T-code and name it
  - (b) Click Create and specify details and at last Application give it * cross.
  - (c) Then click the cursor on methods and click on Utilities from MENU UTILITIES>>API-METHODS>>ADD METHOD
  - (d) Enter function module name and click on NEXT(>).
  - (e) Click again on Next and then click on Yes.
  - (f) Now our RFC is converted into BAPI
  - (g) Click on SAVE and click on BACK
- **Insert RFC into BO,** Go to OBJECT TYPE FROM MENU

OBJECT TYPE >> CHANGE RELEASE STATUS TO >> IMPLEMENTED.

OBJECT TYPE >> CHANGE RELEASE STATUS TO >> RELEASED.

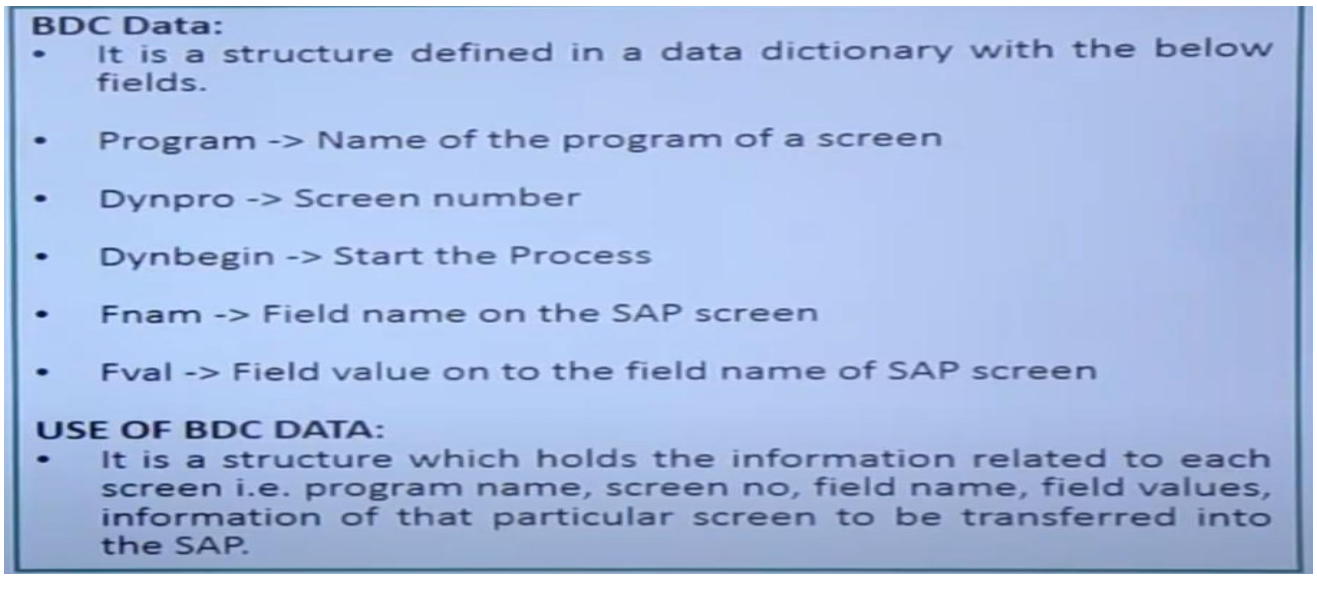
BATCH DATA COMMUNICATION (BDC):

It is technique of transferring the data from legacy system (Non-SAP) to SAP.

It is two types.

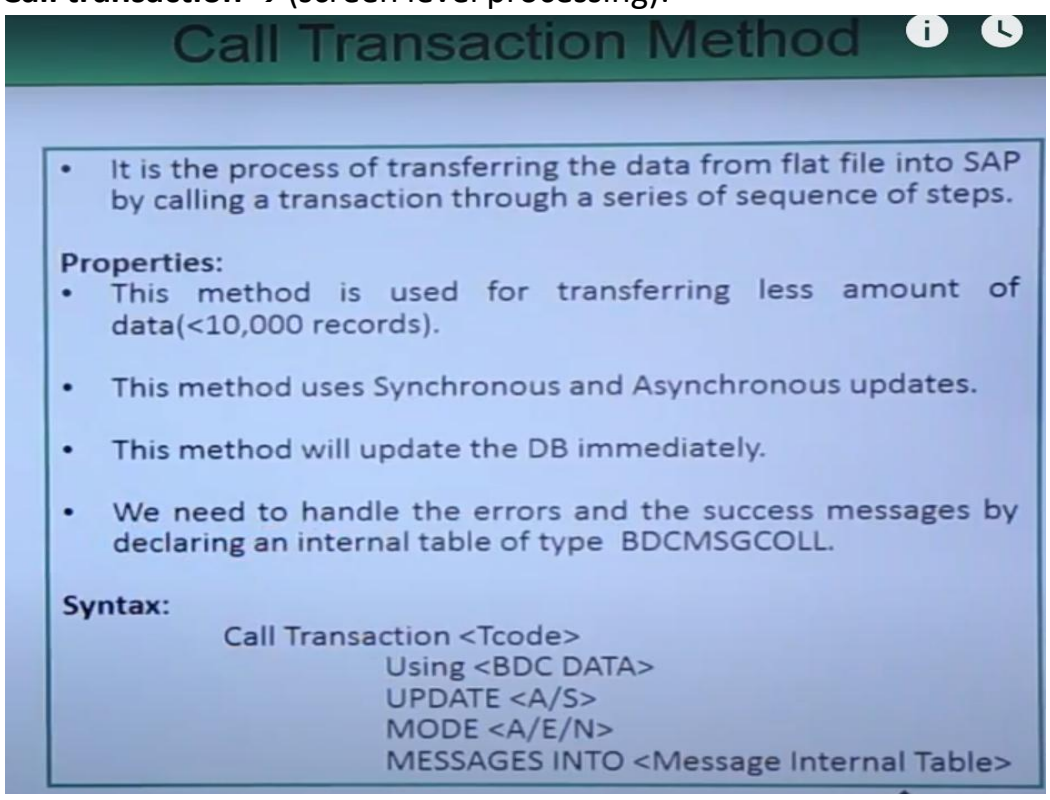
1. Outbound process:
   The process of transferring the data from SAP to Non-SAP or another SAP server.
2. Inbound process:
   The process of receiving the data from Non-SAP server to SAP server.

BDC Data:



Different methods of BDC:

a) **Call transaction** →(screen level processing):



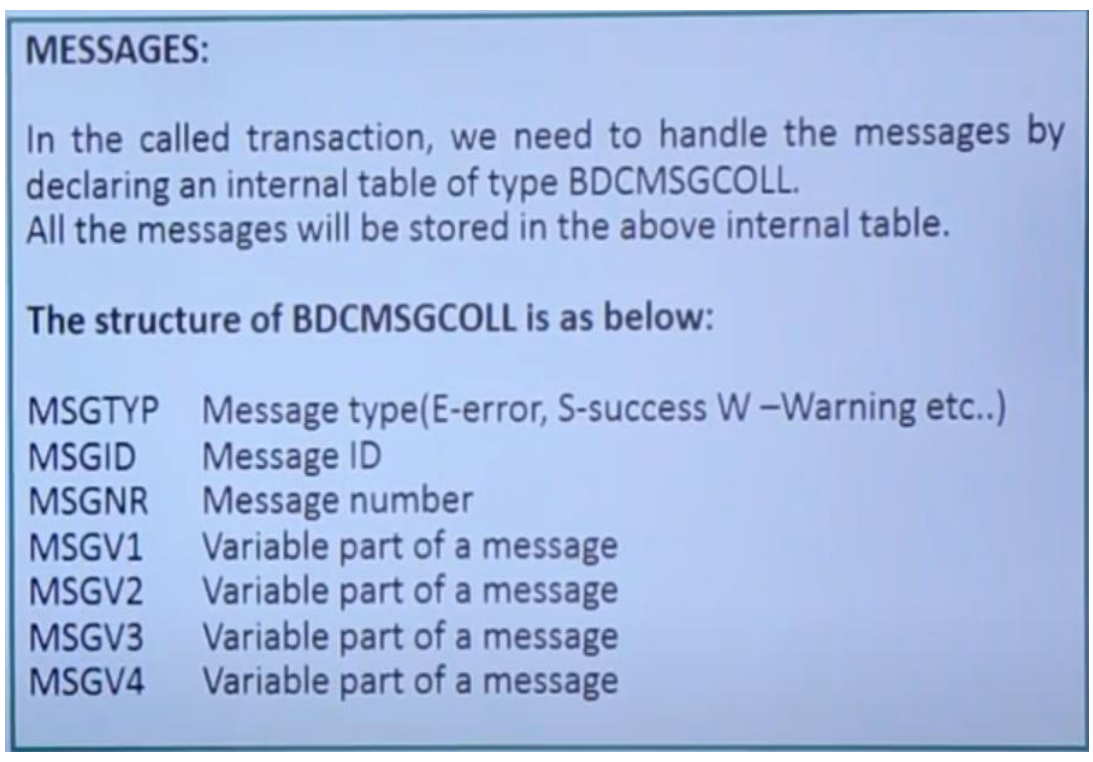
Update:

- Synchronous Update:
  It will wait for the update to be finished, then it continues to process the next record.
- Asynchronous Update
  It starts to process the next record without waiting the update to be finished.

Mode:

- A ----- All Screen mode
- E ----- Error Screen mode
- N No screen mode

MESSAGES:



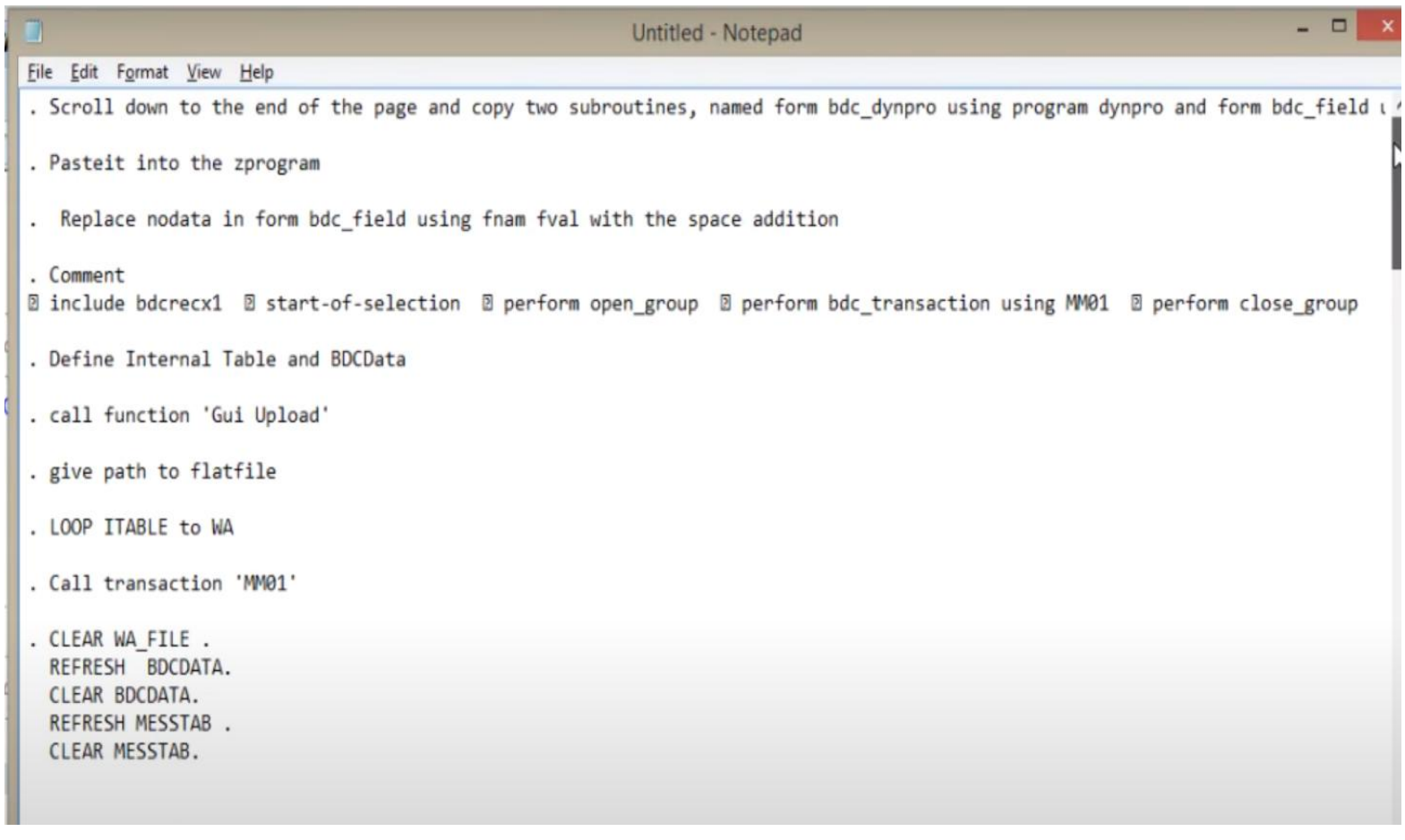
RECORDING METHOD:

It is very difficult to find technical information of each field on the screen, we go for method 'Recording method'.

This recording methods records all the fields in the transaction and generates technical information such as program name, screen no, field name, field value for each field on SAP screen.

Steps to recording a transaction:

- T-code to record a transaction is **SHDB.**
- Go to **SHDB** and click on New recording.
- Enter the recording name and enter the Transaction code as **MMO1 to create recording** and click on start recording.
- MMO1 screen is displayed, then enter **Industry sector** and **Material type**.
- Then click on **Select Views** and select **Basic Data 1** and click on continue button.
- Then enter the **Description** and **basic unit of measure** as kg's.
- Click on Save, then the recording will display all the technical information.
- Click on save and go back, then
- Select the recording and click on **program** to create a program and name it.
- **Select transfer from recording** and press enter
- Give the **title** and click on **source code**, then code will generate.
- Remove unwanted lines and continue first 4 steps written in below image.
- Create an internal table of flat file type structure.
- Upload the data from flat file into the internal table using call function GUI_upload.
- Give path to flat file and loop internal table to workArea.
- Copy the code generated from recording and paste it here.
- Perform BDC-Dynpro using <prog name > and <screen no>
- Perform BDC-Field using <fname> <fvalue>



- Call transaction code ‘MMO1’ call transaction
  Using BDC DATA
  Update A/S
  Mode A/E/N
  Message into MESSTAB
- If SY-SUBRC=0.
  Perform display_success_rec. (subroutine for success rec’s)
  Else
  Perform display_error_rec. (subroutine for error rec’s)
  Endif.

```
Form display_success_rec.
            Read table MESSTAB
                        With key MSGTYP = 'S'.
                        .....
Endform.
Form display_error_rec.
            Loop at MESSTAB where MSGTYP = 'E'.
            Call function 'FORMAT_MESSAGE'.
            ........
            Endloop.
Endform.
```

b) **Session method→**(screen level processing)

- This method is used to transfer large amount of data (eg: >10,000 to 10 lakhs)
- It is basically a 2 step process
  1. Create a session
  2. Process a session
- **Session:** a session contains a group of records, which have to be transferred into SAP.
- To create a session, we use the below function modules.
  1. BDC_Opengroup
  2. BDC_Insert
  3. BDC_Closegroup

- **BDC_OPENGROUP:** This function module is used to create a session with the below parameters.

client no	=	client no
user name	=	user name
group name	=	Session Name
Keep	=	'X'
hold Date	=	session is locked until the specified date

Keep = ‘X’ , specifies the session to be available in the session queue for processing at a later point of time.
Just call the function module and provide the above parameters.
The session will be created

BDC_INSERT:

- This function module is used to transfer the data from BDCDATA structure into the session.
- Just call the function module and provide the transaction code name and BDCDATA internal table name.

BDC_CLOSEGROUP:

- This function module is used to close the session which is created by function module BDC_OPENGROUP.
- Once the session is created, we need to process the session.

SM35 is the T-code to **process the session**.

- Go to SM35 and select the session name
- Click on process button
- Select foreground or background or error mode , click on process
- The session is processed

  Status of session
  - Select the session name
  - Click on button analysis
  - Click on tap log

c) **Direct input method→**(Standard SAP program)

- These are standard SAP programs which are used to update / upload the data from flat file into SAP.
- The direct input method generally is used when you need to transfer a large amount of data directly into an SAP system.
- In this method, a number of function modules are called to transfer data directly to the database of an SAP system.
- These function modules also make relevant checks to avoid any kind of errors during data transfer.
- If an error occurs, the error is fixed with the help of a restart mechanism.
- However, to activate this mechanism, direct input programs must be executed in the background

- We rarely use direct input programs in the real time, because of below limitations.
- The file structure should be created as per the structure of a standard internal table declared in the standard program.
- Here, we can't catch all types of errors because direct input programs works directly on open SQL statements, where as call transaction and session method uses 'Screen level processing' (data is transferred through the screens) where we can catch all type of errors.

d) **LSMW** →(A tool)

LEGACY SYSTEM MIGRATION WORKBENCH

- It is a SAP tool which is used to transfer the data from legacy system to SAP.
- The data can be transferred either all at once or periodically at regular intervals.
- The LSMW tool, a cross-application component of the SAP system, first reads the legacy data from the spreadsheet tables or sequential files and then converts it to a format supported by the target system
- Finally, the converted data is imported in the database used by the SAP system.
- It is mainly used to transfer very less amount of data. (<5000)
- LSMW contains very little coding. It is mainly designed for consultants who doesn't have any programming language. i.e. functional consultants.
- There are 4 methods available in LSMW.
    1. Direct input method
    2. Recording method
    3. BAPI method
    4. ALE/IDOC method.