ERP: Enterprise → Company Ex: Reliance, BP, BMW
Resources planning → Man, Machine, Material → Raw materials, semi financial, Finished material
→ Effective plans with minimum utilizatio of Resources for maximum profit in a company.

✓ ERP S/W is a software which is used to maintain the Business activities of a company or Enterprise is called ERP software

- The Business activities of a company means Sales related Transactions / scenarios
  material related "
  finance related "
  HR related "

Examples of ERP S/W

1. people soft
   - Used by small organizations
   - less No of Business scenario's
   - less No of Business modules
   - very good at HR modules
   - less cost S/W

2. ORACLE
   - Used by MNC's, More No of Business Scenario Appl
   - less No of Business Modules
   - very good at FI module
   - High cost S/W

3. SAP
   - Used by MNC's
   - Very Huge No of Business App's
   - Supports High No of Business modules
   - very good at integration b/w all modules
   - High cost S/W

HISTORY OF SAP

SAP stands for SYSTEMS,
APPLICATIONS,
PRODUCTS in data processing

1972 – SAP AG

1980 – SAP R/2 –

1990 – SAP R/3

SHELL SHELL

- In the year 1972 5 IBM Employees started a Company by the Name SAP AG
- In the year 1980, They Release the first product by the Name SAP R/2 where R stands for Real time 2 stands for 2 Tier architecture
- In the year 1990, SAP Release the Enhanced version of SAP R/2 by the New Name called as SAP R/3 where 3 stands Three tier architecture
- Even to day also we are using SAP R/3

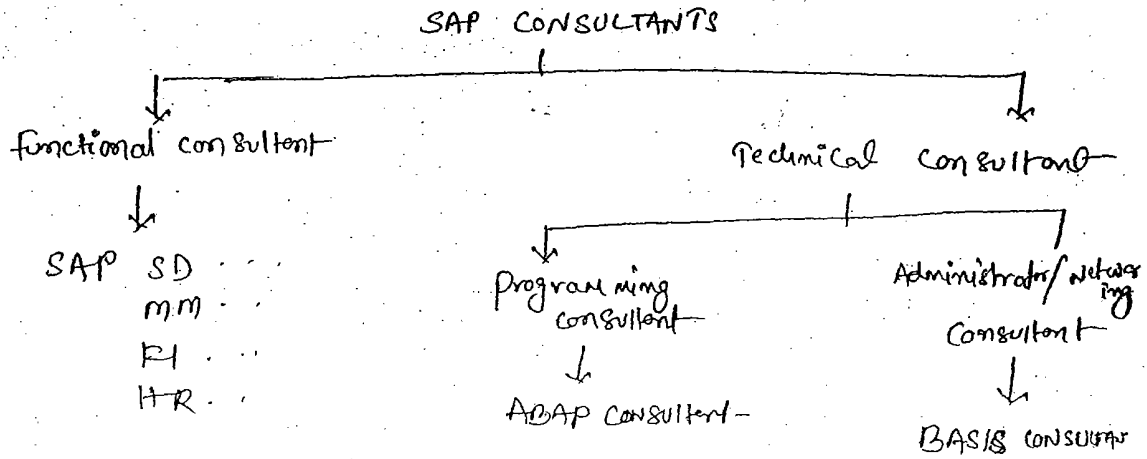
A

ABAP/4 :

- ABAP stands for Advanced Business Application programming Language where 4 stands for IV The Generation Language.
- The Entire SAP S/W is developed by using ABAP as a promming language.
- SO, If we have very good Knowledge on SAP ABAP we can work on any SAP module.

SAP MODULES

SAP SD – Sales and Distribution
SAP MM – material management
SAP FI – Finance
SAP CO – Controlling
SAP HR – Human Resources
SAP PM – Plant management
SAP CRM – Customer Relationship mgmt
SAP SRM – Supply Relation mgmt
SAP SCM – Supply chain management
SAP WM – warehouse mgmt



Project size (14 TO 20)

⊛ GO-LIVE

FUNCTIONAL CONSULTANTS

A Consultant who is responsible for configuring the business Scenario's like configuring (Sales) organizing, distribution channels, divisions and plant configuration and G/L account configuration etc are called Functional Consultant.

- These Consultants directly communicates with the business Users and they gather the requirements.
- once the requirements are gathered they prepare a document called as FUNCTIONAL SPECIFICATION

FUNCTIONAL SPECIFICATION

It is a document which containing the business requirements to be developed or configured.

PROGRAMMING CONSULTANT

→ A Consultant who is responsible for Creating the ABAP programs or reports, Creating SAP Screens, Creating database tables, Enhancing the standard SAP functionality are called as ABAP Consultant.

→ The ABAP Consultant will receive the functional Specification, and they will create the Technical Specification's documents based on functional specification document.

→ Once the Technical specification is approved, then the ABAP Consultant will develop the ABAP programs Etc.

UNIT TESTING

- A Sample Testing done by the ABAP Consultant for the develop objects is called Unit Testing
- Generally we Create 5 Test Cases or Scenario's for Each object. (3-5 Test Cases)
- All these Test Cases will be saved in a document called as Unit Test document.

BASIS CONSULTANT

- A Consultant who is responsible for installing SAP S/W, installing data base, Creating System or Networking architecture, Creating User ID's and passwords etc are called as BASIS Consultants.
- In Simple words Administrators are called as BASIS Consultants.

ROLES AND RESPONSIBILITIES OF ABAP CONSULTANTS

- Analyse the functional specification document
- Estimate the No of hours, for developing the object
- prepare the Technical specification document
- Develop the object.
- Finally, perform the Unite testing And release the object to the quality or Testing server.

E-mail:
- Lotus
- Microsoft Outlook

* Black Business is not possible in SAP

SAP ARCHITECTURE R/3

- Basically there are 3 types of architectures in S/w Engineering

1. Single Tier architecture
2. Two Tier architecture
3. Three Tier architecture

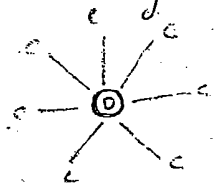
SINGLE TIER (14/24)

- A system which is responsible for loading the program into memory, compiling, ~~Exe~~ interpreting, Executing on a single system or machine are called as single tier architectures.

Ex: 'C' language programs are based on Single tier architecture

TWO TIER

- In this type of architecture we have two layers called as 1. Database layer
  2. Client layer

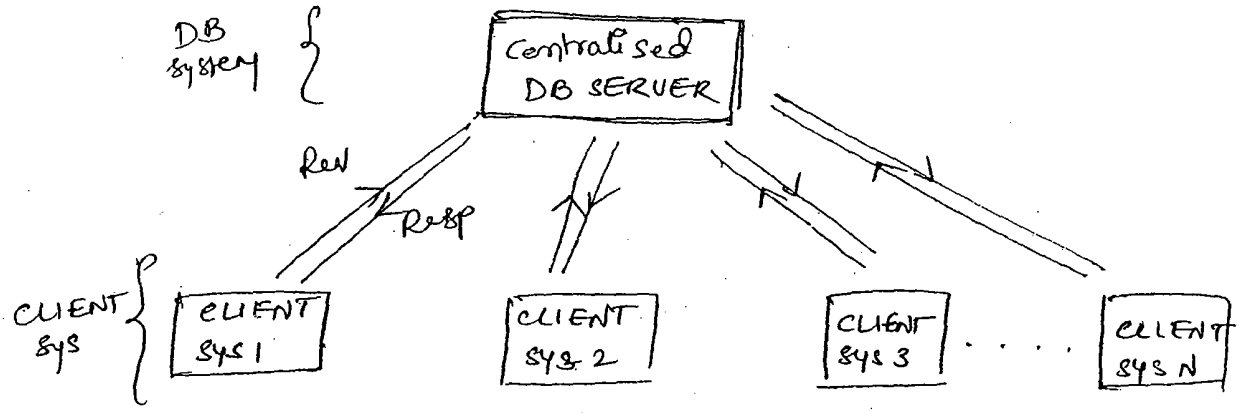


Database Layer

- In this layer we have a system with very high configuration called as Centralised database server.
- The main roles of Database server is to read the data and write the data in to data base.

CLIENT Layer

- In this layer, we have n no of system's which are connected to database Server.
- These systems are called as client systems
- Each user will operate from client system and raises a requesting, request's for some information
- The request is received by database Server and it will give a response back to the client system.
- This type of architecture is called as 2 Tier or CLIENT SERVER ARCHITECTURE



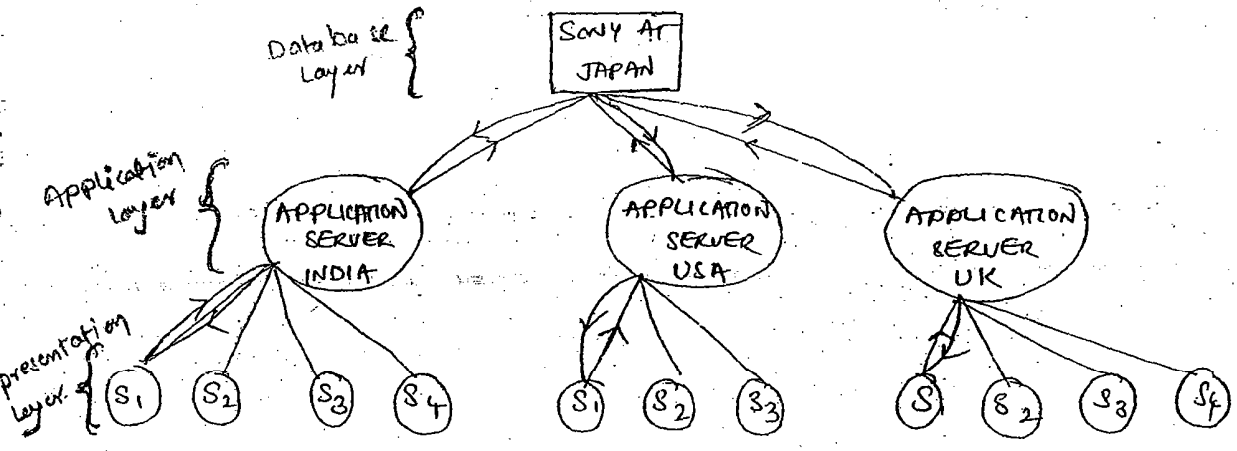
TWO TIER CLIENT SERVER ARCHITECTURE

Ex: ORACLE
SAP R/2

DISADVANTAGES

- If there are more no of ~~Req~~ client systems then there will be heavy load on the database server and all the client systems will be very slow.
- To over come this problem we have the other architecture by name three tier architecture

THREE TIER ARCHITECTURE:



- In this type of architecture we have three layers
  1. Data base Layer
  2. Application Layer
  3. Presentation Layer

Database Layer :

- A system with very high configuration to store the data is called a data base server.
- The main functionality of the data base server is to read or write the data in to data base server.

APPLICATION LAYER

- In this layer we have n no of system's which are connected to data base server.
- The main functionality of application server is to execute the statements which don't require a data base connection

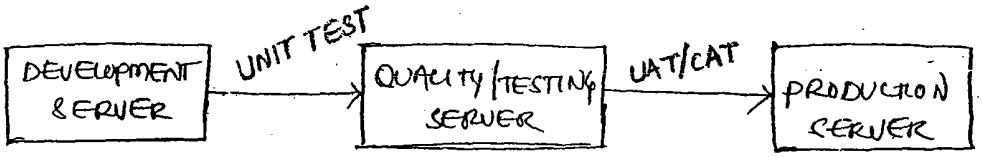
  Ex : Calculating Total's etc.
- If a statement requires a database connection, It will be sent to the data base server.

PRESENTATION LAYER

- In this layer also we have n no of systems which are connected to application server.
- These systems are called as presentation systems, all the users will be operating from this presentation system.

"ABAP"

SYSTEM LANDSCAPE



Development Server:

- A server or system where all the SAP Consultants will do the developments for SAP objects is called A development Server.
- All the ABAP consultants will be developing the objects like Creating a program, a Table, a Screen etc.
- Once the development is finished, we should do Unit Testing for the developed objects.
- Once Unit Testing is finished we have to release or move the object from development server to the quality Server.

QUALITY SERVER

- In this Server, the functional Consultants will do a regorous testing for the developed object.
- If there are any Errors they will inform Concerned ABAP Consultant.
- The quality Testing will be generally done with various Test Cases or Scenario's.
- Once the quality testing is finished it will be sent for UAT/CAT Testing

UAT/CAT TESTING

★ In this Testing, the User or business User will do the Testing for the developed object.

★ If UAT/CAT Testing is approved, it will be moved to production Server.

PRODUCTION SERVER :

- This is the Server which is used by the business Users to maintain their business
- This Server Contains the live market data.
- All the developed objects will be used for the realtime Customer, s vendors, materials Etc.
- It is called as LIVE SERVER. Because it Contains the live data.
- No ABAP Consultant will have Authorisation to the production Server.
- ABAP Consultants will be working only with development Server.

KT

LOGIN TO SAP

- Double click on SAP LOGON PAD / ICON
- Give the user Name as : sapuser
  password as : india123
  and press Enter.
- The SAP Easy Access Screen will be displayed. This is the default screen / SAP Home page screen.

TYPES OF SOFTWARES

- SAP is available in two types of softwares.
  1. Development software
  2. IDES software.

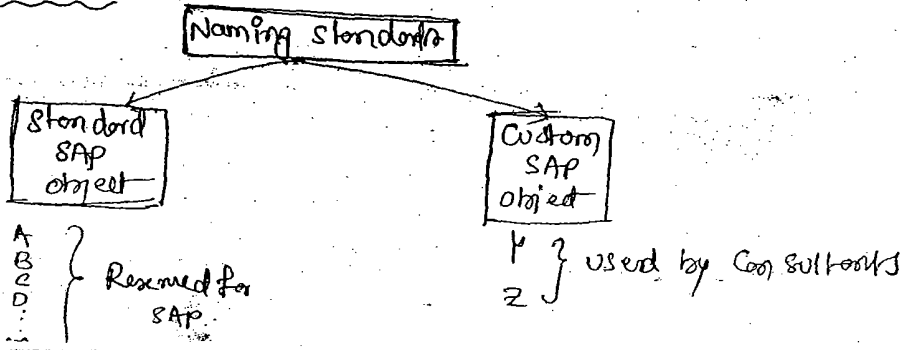
DEVELOPMENT SOFTWARE

- It is a original software which is used by the real time companies to maintain the business.
- All the project related objects must be developed in this development software.
- The client no's used in the development software are 100 (client), 120, 200, 220, 300, 320 Etc.

IDES SOFTWARE

- It is called as Internet demonstrative Evaluation System.
- It is mainly used to learn SAP
- It is called as Training software.
- The client no's used in the IDES software are 800, 810 (learning), 820, Etc.

NAMING CONVENTIONS :



Standard SAP OBJECTS

- The objects which are developed by SAP Company are called as standard SAP objects.
- All standard objects must start with either A, B.....X

CUSTOM OBJECTS

- The objects which are developed by the ABAP consultant are called Custom SAP Objects
- All Custom objects must start with either Z/Y

Ex: ZMM_Salesreport

TRANSACTION CODES (Command)

- It is a code which is used to call a program/screen/table Etc.
- A Transaction code is called as a Command in Normal language.

20,000 - 30000

- SE11 – Data/ABAP Dictionary Transaction code
- SE38 – Transaction code for ABAP program/REPORT
- SE80 – Transaction code for ABAP development workbench
- SE24 – Transaction code for package builder
- SE37 – Function Builder
- Etc

where 'S' stands for Session (window)
'E' stands for Editor.

NOTE: we can open maximum of * (six) sessions.

* Transport request Number:

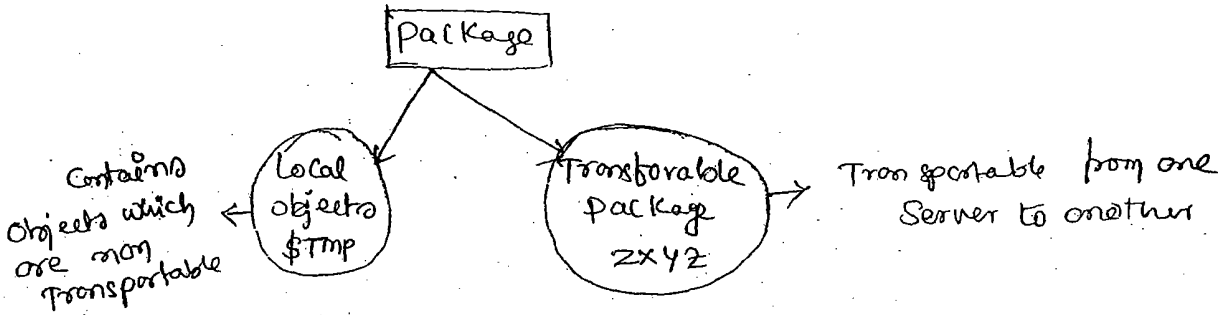
- A number which is used to move the object (prog/screen/table.. from one Server to another Server in the system landscape is called TRANSPORT REQUEST NUMBER
  - * Ex: ECCK900012
    - ↳ Application Server Name
    - Basis/TL
    - * Root Folder/Central Folder

Package :

- It is a container of SAP objects.
- All our developments like programs screens, Table's Extra must be saved under a package

  SE21 is the key code / T code to create a package
- In the Real time, package will be created by BASIS Consultant / TL / PM.

TYPES OF PACKAGES



LOCAL OBJECT :

- The package Name for the local object is $TMP it will contain the non - Transportable objects.
- Transportable package
- These package's will contain the objects which are Transportable from one server to another Server.
- When Ever we Create any SAP object, System will ask for package Name.
- If we give the Transportable package Name, it will ask for Transport request Number.
- If we give the package Name as local object, it will not ask for any Transport request Number.

STEPS TO CREATE A PACKAGE : (SE21)

→ Go to SE21
→ Give package Name as ZRAMESH
→ Click on [Create]
→ Give description as package → ZJANJANAM Created
→ Click on (📁) Create Icon
→ pop up is displayed for TR#
→ click on (📁) for creating TR #
→ A pop up is displayed.
→ Give short description as TR # for package
→ CLICK on SAVE/ press Enter
→ A TR# generated by system
→ Press Enter
→ Click on Save
→ By above steps, A Package Created

STEPS TO CREATE A SAMPLE PROGRAM : (SE38)

- Go to SE38
- Give program Name as ZABCD
- Click on [Create] ZSAMPLE_ram
- provide title as Sample prog
- Select type as "Executable program"
- click on [Save]
- provide package Name, Ex: ZLABS
- provide TR # (or) click on (📁) to Create new TR#
- Finally ABAP Editor will opened
- write the Below codes

```
WRITE 'IBM INDIA'
WRITE 'IN HYD'
WRITE / 'Intell group'
WRITE : / 'TCS'
        / 'In 'Hyd',
        / 'In Hitech city'
```

:} → CHAIN OPERA

- Save
- check
- Activate
- Execute

{ SAVE CTRL+S
CHECK CTRL+F2
ACTIVATE CTRL+F3
F8 - EXECUTE }

WRITE :

- This statement is used to print a text on the output screen

SYNTAX :
WRITE 'TEXT'

EX :
WRITE 'INDIA'

{ COMMENT LINES
CTRL + <
CTRL + > Removing }

{: chain operator}

WRITE '/':

- This statement is used to print the Text in a saparate line

SYNTAX :
WRITE / 'The colon is :'

WRITE ':'

- The : is used to print multiple Text Saparated by Commas using a single WRITE Statement.
- The : is called as chain operator

Specifying the position

WRITE <POS> <TEXT>

EX! : WRITE 45 'SALES REPORT'.

Specifying position and width :

WRITE <POS> (width) <TEXT>

EX! WRITE 45(5) 'SALES REPORT'

Specifying left Right Center Justification:

- We can print the text either from left side or from Right side or Exactly at the center position.
- By default it will print from the left side.

WRITE <TEXT> left/Right/Center JUSTIFIED

Ex:

```
WRITE /25(30)  'IBM INDIA'  Left - Justified
WRITE /25(30)  'IBM INDIA'  Centred
WRITE /25(30)  'IBM INDIA'  RIGHT JUSTIFIED
```

SPECIFYING COLOURS FOR THE TEXT

- We can apply Colours for the Font or Background Colour
- We have Colours starting from 1 to 7

WRITE <TEXT> Color <Colorno>

```
WRITE  'IBM INDIA'  Color 6.
WRITE  'IBM INDIA Hyd'  Color 6 Inverse.

WRITE: /25(30)  'SALES REPORT'  LEFT JUSTIFIED Colour 6.
WRITE: /25(30)  'SALES REPORT'  Centered colour 6.
WRITE: /25(30)  'SALES REPORT'  RIGHT JUSTIFIED Colour 6:
"      "        "               "     "  INVERSE.
```

System VARIABLES

- The variables which are declared automatically by the System are called System variables.
- All the System variables will be automatically assigned with default values.
- All the System variables will starts with 'SY'
- All the System variables are stored in a structure by Name 'SYST'

SE11 – SYST
↳ DATUM (search)

Ex:

WRITE / SY-DATUM.		CURRENT DATE APP'
WRITE / SY-UNAME.		User Name
WRITE / SY-UZEIT.		current time of Ap Se
WRITE / SY-REPID.		Report ID/Name
WRITE / SY-ULINE.		Horizontal line
WRITE / SY-VLINE.		Vertical line

Creating Variables

- A variable is a memory location use to store a single character or a multiple character

SYNTAX: `DATA : <VARIABLE NAME> TYPE <data type>`

EX : DATA V_Name TYPE C.

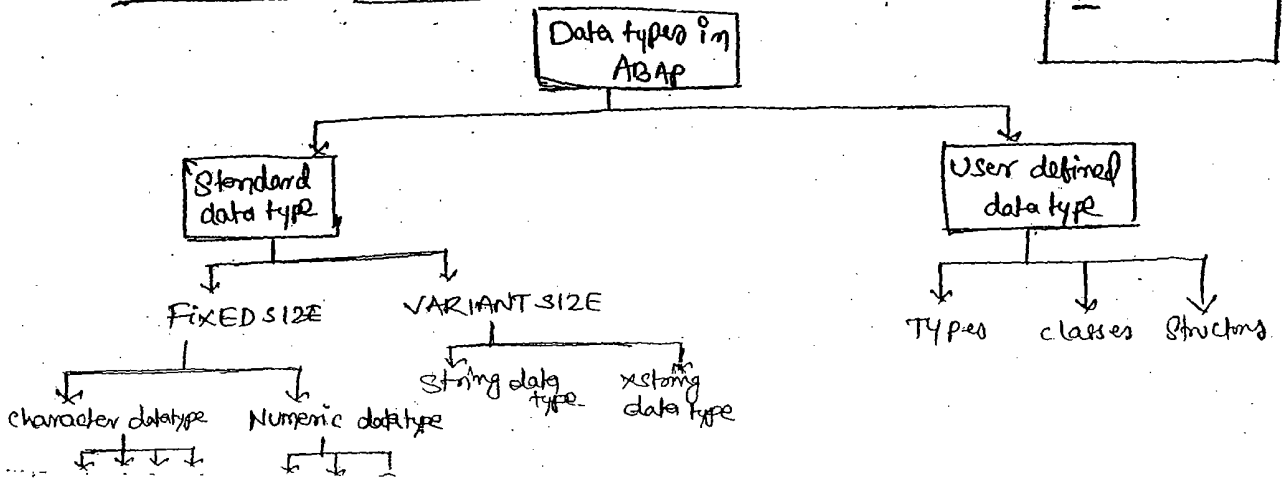
SYNTAX: `DATA : <VARIABLE NAME> (length) <TYPE C>` (data type)

EX : DATA : V_Name (25) TYPE C.

```
DATA V_NAME1 TYPE C.
DATA V_NAME2 (25) TYPE C.
V_NAME1 = 'A'.
WRITE / V_NAME1.
V_NAME2 = 'ACCENTURE'
WRITE / V_NAME2.
```

C	character
N	Numeric character
D	Date
T	Time
I	Integer
P	packed
F	Float

DATA TYPES IN ABAP:



** DATA DECLARATION

```
DATA V_NAME (15) TYPE C.
DATA V_PIN (6) TYPE N.
DATA V_DOB TYPE D.
DATA V_TIME TYPE T.
DATA V_COUNT TYPE I.
DATA V_COST TYPE P DECIMALS 2.
DATA V_TOTCOST TYPE F.
```

** ASSIGNING VALUES

```
V_NAME = 'Sales Report'
V_PIN = '500435'
V_DOB = '20101024'
V_TIME = '102030'
V_COUNT = '1'
V_COST = '1999.999'
V_TOTCOST = '1999.999'
```

** DISPLAYING

```
WRITE : /V_NAME, /V_PIN, /V_DOB, /V_TIME, /V_COUNT,
        /V_COST, /V_TOTCOST.
```

*** EXAMPLE

```
DATA V_UNAME TYPE SY-UNAME.
DATA V_DATE TYPE SY-DATUM.
DATA V_REPID TYPE SY-REPID.

V_UNAME = SY-UNAME.
V_DATE = SY-DATUM.
V_REPID = SY-REPID.
WRITE : /V_UNAME , /V_DATE, /V_REPID.
```

'SETTING'

System → User profile → Own data

/NEX ⇒ Exit

MAINTAINING USER SETTINGS

- open the program click on SYSTEM → USER PROFILE → OWN DATA → Dafaults Tab
- change the date format what you need.
- Click on Save
- Logout and Login and Execute the same program.

Difference between DATE DATATYPES

- We can use either D or DATS to store the dates.
- The difference between the above two is,
  - TYPE D does't store any separators
  - TYPE DATS stores separator

CONDITIONAL STATEMENTS IN ABAP

- If .... Endif
- If ... ELSE ... ENDIF
- If ... ELSEIF ... ELSEIF ... ELSEIF .. ELSE ..
- Switch (CASE .... End CASE)

① SYNTAX:

```
If <Condition is true>
...
...
...
End if
```

②

```
If <Condition is true>
...
...
Else
...
...
End if
```

③

```
If <Condition is true>
...
...
Else if <Cond2 is true>
...
...
Else if <Cond3 is true>
...
...
Else (...)
Endif
```

④

```
CASE <Condition>
  when 'val1'
  ...
  ...
  when 'val2'
  ...
  ...
  when 'val3'
  ...
  ...
  when others
  ...
End CASE
```

Example on IF and ENDIF

```
DATA : V_LANDI (3)   TYPE C.
PARAMETERS : P_LANDI(3)  TYPE C.
V_LANDI = P_LANDI.
IF V_LANDI = 'IN'.
    WRITE : / 'The Country Name is INDIA'.
ENDIF.
```

PARAMETERS : SYNTAX : PARAMETERS : <P_NAME> (length) TYPE <Datatype>

- This statement is used to create a input field on Input screen called as Selection Screen (Input Screen) same

DEBUGGING

- It is a SAP TOOL which is used to watch or Examing the Execution of Each statement in a program is called debugging.

BREAK POINT mag 30

- It is a point which is used to break the Execution of a program at any statement is called Break point
- There are two types of break point,
    1. static Break point (FIXED)
    2. DYNAMIC Break point (max 30)

STATIC Break point

- A Break point which is set using the statement BREAK-POINT. is called static Break point
- It is fixed in the program until and unless you delete it.

DYNAMIC Break point

- A Break point which is active for a certain period of time is called dynamic Break point
- Dynamic Break points are set by clicking on the symbol (STOP)
- we can set a maximum of 30 dynamic break points.

Single step Execution (F5)

- It is a button in the debugging tool to Execute the program line by line i.e.

EXAMPLE PROG ON If..

```
If V_LANDI = 'IND'.
   WRITE : / ' we are calculating INDIAN Customers'
ELSEIF V_LANDI = 'US'.
   WRITE : / ' we are calculating for U.S Customs'
ELSEIF V_LANDI = 'UK'
   write : / ' we are cal for UK'.
ELSE.
   WRITE : / ' INVALID CUSTO'
ENDIF.
```

——— *** EX 2 *** ———

```
CASE V_LANDI.
when 'IN'.
write : / ' we are cal for INDIAN'.
when 'US'
write : / ' we are US.'
when 'UK'
write : / ' we are UK'.
when others
   write : / ' Invalid'.   ENDCASE.
```

Loop Statements

- These statements are used to Execute the same set of statements again and again.
- In simple words all the repeatable statements are placed in a loop

TYPES OF LOOPS IN ABAP

1. while loop
2. Do loop
3. Loop .... End loop

while loop syntax

```
while <condition is true>
  ...
  ...
  ...
Endwhile
```

★ SY-INDEX
Loop Counter

Example:

```
DATA V_Count TYPE I.
while V_Count < 10.
  WRITE: / V_Count.
  V_Count = V_Count + 1.
Endwhile.
```

Do Loop syntax

```
DO <NO> TIMES
  ...
  ...
  ...
Enddo
```

Example:

```
DATA V_Count TYPE I.
  Do 10 Times.
    WRITE: / V_Count.
    V_Count = V_Count + 1.
Enddo.
```

SY-INDEX :

- It is a System Variable which holds the loop iterati Counter. ie, whether it is first iteration or Secon iteration etc..

Ex :

```
While V_Count <10.
  WRITE :\ SY-INDEX.
EndWhile. V_Count = V_Count+1.
```

EXIT STATEMENT

- This statement is used to Exit the loop

Ex :

```
DATA V_Count type I.
V_Count = 1.
DO 10 times.
  WRITE : / V_Count.
  V_Count = V_Count+1.
If SY-INDEX = 5.
  EXIT.
ENDIF.
ENDDO.
```

CONTINUE :

- This statement is used to continue the Next iteratio by bipassing all the below statement under it.

Ex :

```
DATA V_Count TYPE I.
V_Count = 1.
Do 10 times
  If SY-INDEX = 5
  V_Count = V_Count+1.
  CONTINUE.
Endif.
WRITE : / V_COUNT.
V_Count = V_Count+1.
Enddo.
```

OUTPUT

1
2
3
4
6
7
8
9
10

"DATA DICTIONARY" SE11

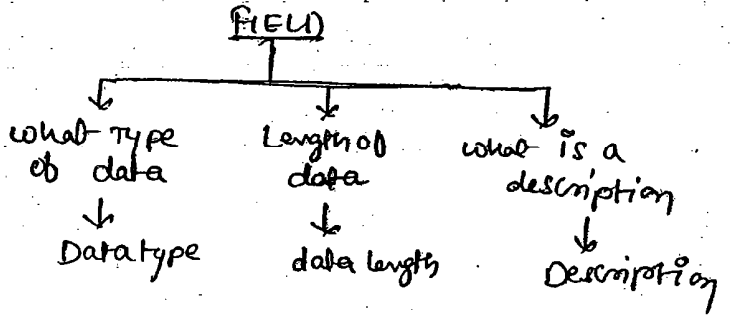
It is a central repository which contains all the objects which are related to data base.

LIST OF THE OBJECTS

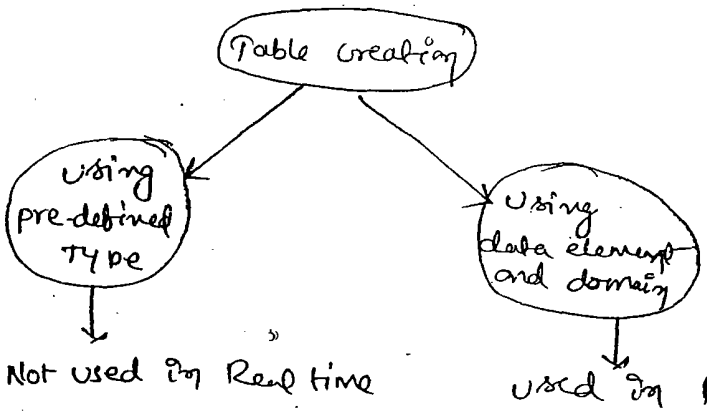
- Data base Table
- View
- Data type
  - Data Element
  - Structure
  - Table Type
- Type group
- Domain
- Search help
- Lock object

Data base Table

- A Table is a collection of Row's and Column's
- Each row is called a record in a Table
- Each Column is called a field in a Table.
- In simple words a Table is collection of field and records
- To define a Table, we must define the fields.
- To define a field, we must provide the Below property's



- A Table can be Created in two way's
  1. Using pre-defined type
  2. Using data Elements and domains.



Advantages of Table Creation Using DE and Domains:

- Re usability, The same data elements and domains can be used in multiple Tables their by using the concept called Re-Usability
- Data Elements and domains are used in Creating Foreign Key Relationship
- They are used in Creating Search, Help's
- They can also be used in Creating IDOC's in CA's.

KEY FIELD:

- A field which does not accept any duplicated data is called a Key field.
- Each Table should have at least one Key field.
- We can Create a maximum no of — ? 16

opens a New Session
/O → SE11
/N → SE11
Kill current Session

STEPS TO CREATE A TABLE:

- Go TO SE11
- Give The Table Name ZSTD_TAB
- click on Create Button
- provide description
- Give the delivery class as A
- Select display maintainence allowed.
- click on field's Tab.
- click on The Button pre-defined type

- Enter the Field's as Below.

Field Name	KEY	Data type	Length	description
STD_NO	✓	CHAR	10	student NO
STD_NAME		CHAR	10	student Name
city		HYB	20	city

- click on the Button [Technical settings]
- Give the data class as [APPL0]
- Give the size category as [0]
- click on Save
- click on Back Button
- Save And Activate the Table

Creating Table Records

- click on Utilities
  - ↳ Table Contents
    - ↳ Create
- Enter the Student No, Name, city
- click on Save
- A record will be saved.

Displaying the Table records

- click on UTILITIES
  - ↳ Table Contes
    - ↳ Display (or) click on [icon]

Assignment

1. Create a Table with Below Field.

   CustNo, Cust Name, city, country

   CustNo (KUNNR)
   Cust Name (NAME1)
   city (ORT01)
   country (LAND1)

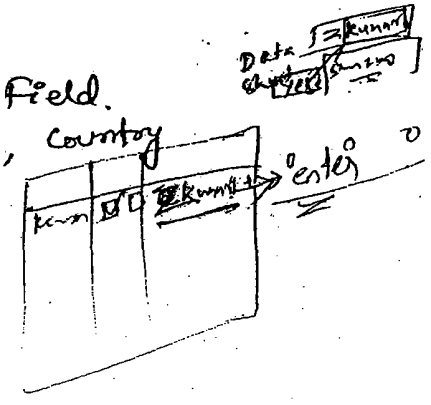


Table creation using data Elements and domains

Domain: It is an object which specifies the Technical attributes such as the data type and length is called a domain.

Eg: KUNNR is a domain for Customer no

DATA ELEMENT:

It is an object which specifys the sceryantic info such as the field description, field labels Extra for a field is called data Eleryent.

Eg: KUNNR is a data Eleryent for Customer No.

NOTE: To define a field we need to specify the domain Name, data Eleryent Name.

- The domain and data eleryent are interlinked with Each other.

Steps to Create a domain:

- Go to SE11 Give the domain Name as ZNO
- click on Create Button, Give the description
- Give the Technical attributes of below

Datatype : char
No of char : 10
Save and activate it.

Steps to Create a data Element

- Go to SE11, Select data type Give the Name of ZNO click on Create.
- Select data Element press Enter.
- provide the description
- Give the domain Name as ZNO and press Enter.
- provide the Field labels.
- Save and activate.

BUSINESS REQUIREMENT

Create a Table with below Fields

Custo NO — char, 10, CustomerNO
Customname — char, 35, Name
City — char, 15, city

Field	Key	~~Field~~ data element
CUST_NO	✓	ZNO (Create it)
CUST_NAME		ZNAME1 (Create it)
CITY		ZCITY (Create it)

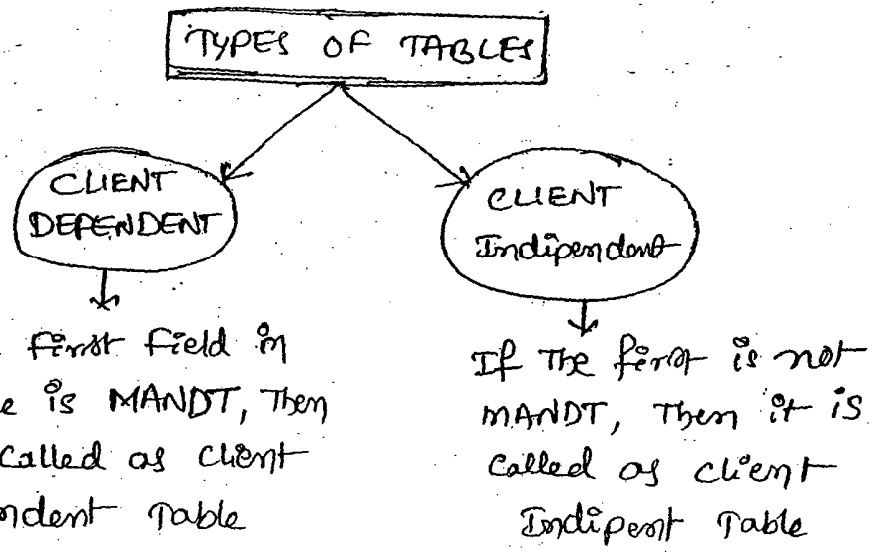
Creating data Elements and domains

- There are Two ways to Create data Elements and domai
  1. Bottom to top approach
     - Create domain
     - Create data Element
     - Create field Using Data Element and domain
  2. Top - To - Bottom approach
     - Create field with any Data element and Name
     - Double click and Create data Element
     - Give any Name for domain and Create it
     - Finally activate domain and dataelem

Initial field

- If This field is selected (☑) The field will store the initial value or Empty Space as a record.
- All the Key fields have this field selected by default
- We Can not change it.

TYPES OF TABLES



CLIENT_NO
- 100 — development
- 120 — Testing client
- 800 — Learning SAP

CLIENT NO :

- It is a Number which is used to provide the security to the data at the data base level.
- The field Name for the client NO is MANDT
- The Data Element and domain for the client NO is MANDT

" PRIMARY KEY "

- Grap of all the Key fields is called a primary Key.

  Ex: MANDT & VEND_NO is a primary key

- The duplicates will be checked based on primary Key but not on the Key fields.

EXAMPLES OF STANDARD SAP TABLE

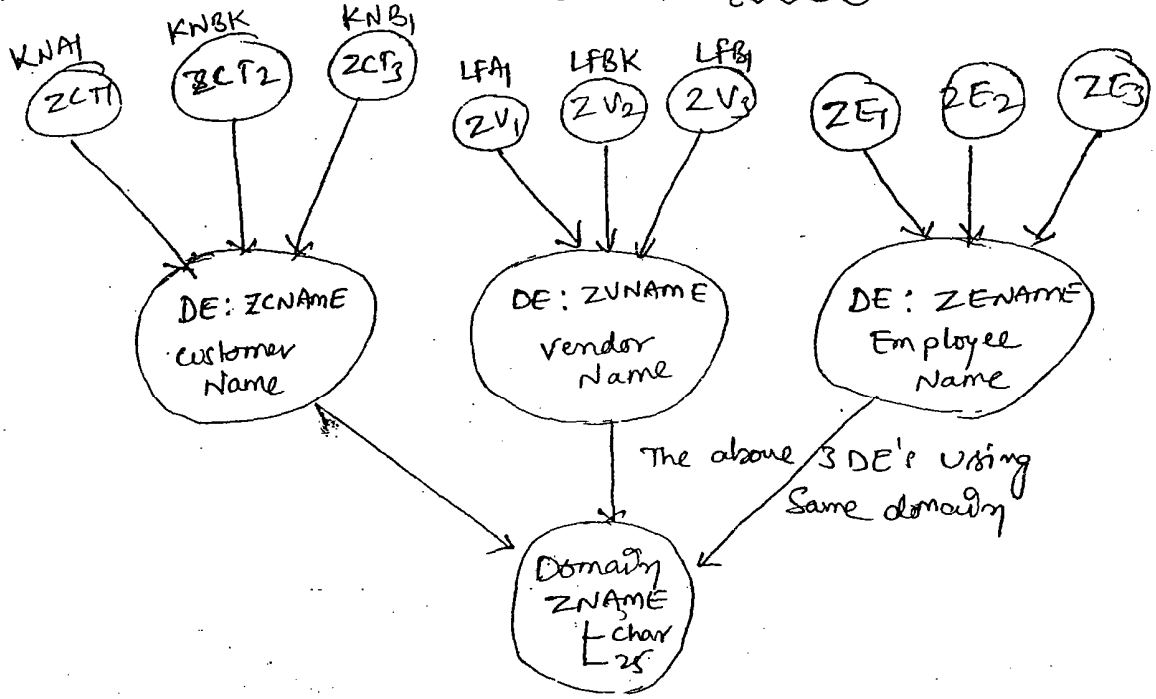
{ 80000 Standard Tables in SAP }

- KNA1 - Customer Master data Table
  LFA1 - Vendor master data Table / LIFNR
  MARA - material master data Table
  MAKT - material description data
  KNBK - Customer Bank data
  KNB1 - Customer Company code data
  LFBK - Vendor Bank ~~description~~ data
  LFB1 - Vendor Company code data

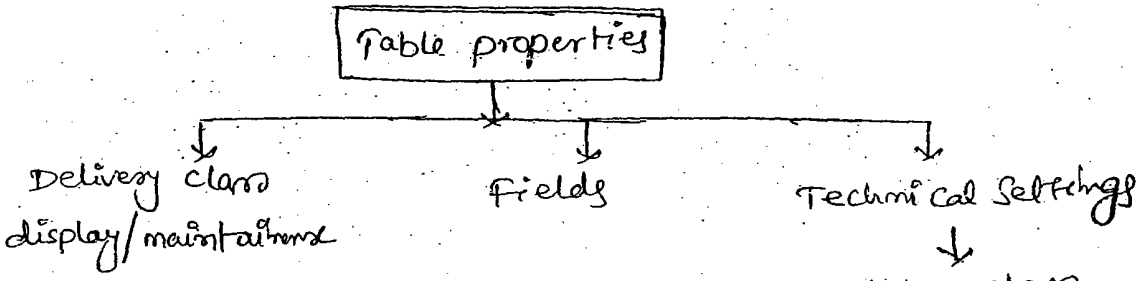
EXAMPLES OF COMMONLY USED FIELDS

KNA1 - KUNNR → Customer Number
KNA1 - NAME1 → First Name
KNA1 - NAME2 → Second Name
KNA1 - LAND1 → Country Code
KNA1 - ORT01 → City

REusability of data Elements and domains:



PROPERTIES OF THE TABLE:



Delivery class

- Delivery class specifies the type of the data that is stored in a Table.
- Below are the options available

Delivery class	Description
A	(Master and Transaction data) Application Table
C	Customising Table (maintaining only by customer NOT SAP)
L	Temporary Table
S, E, W	System Table for store System data

DISPLAY / MAINTAINENCE :

- It will specify whether the table data can only be display or maintain
- The maintainence operation means, Creation, changing deleting the table data.

: Below are the three options available.

1. Display/maintainence allowed.

   The table data can be displayed as well as it can be maintained
   ie, The data can be created, changed, deleted

2. Display/maintainence not allowed

   The table data can't be displayed and can not be maintained.

3. Display/maintainence allowed with Restrictions

   The table data can only be displayed. The maintainence is given with restrictions.
   ie, ONLY certain group of people can maintain the data based on authorisation group.

- The authorization group is created by BASIS Consultant.

DATA CLASS

- It specifys the physical area of the table in the database server.
- Below the options are available.

  APPL0 – master data
  APPL1 – Transaction data.

  } These two are used by ABAP Consult

SIZE CATEGORY :

- It specifys the maximum no of records that can be stored in a Table.
- Below are the options available,

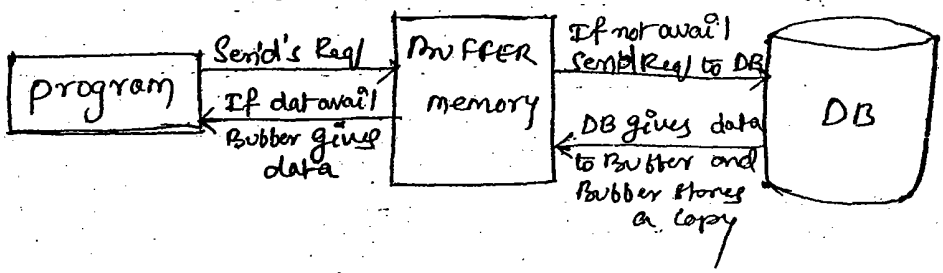
SIZE CATEGORY	maximum no of Records
0	0 to 3,600
1	3600 to 7,200
2	7,200 to 10,000
:	:

- Always select the size category is Zero (0). Because the memory will be created in small block.

  i.e, 0 – 3600 Record space.

BUFFER :

- It is a temporary memory which is used to store certain amount of data
- It is mainly used for Reducing the load on the database Server and improving the performance of the program.



- when Ever a program wont's to read the data, the program sends the requests to the bubber.
- If the data is available in the bubber it is given to the program.
- If the data is not available in Bubber the Req is fenworded to database Server.
- The DB gives the data to Bubber memory.
- [illegible] Copy and Then gives data to Prog.

SE14
↳ Table Name
↳ Edit
↳ Activate and Adjust database

* modifying fields

* BUFFERING OPTIONS :

⊙ Buffering NOT allowed → The Table data is not stored in Buffer
○ Buffering allowed but switched off → Used by SAP (NEVER WORKED ON IT)
○ Buffering switched on → The Table data is stored in Buffer

(NOTE: Select this option only, If the Table is accessed Every day with huge amt of data)

BUFFERING TYPE :

we have three options under this

☐ Single record buffering — only single/first record in stored in buffer
☐ fully buffered — All records are stored in fully Buffer
☐ Generic Area Buffered — ONLY few fields data is stored in the buffer.

• These few fields are called as Generic area

GENERIC AREA

• It is an area which contains the key fields data
• Generic Area is specified by No of Key Fields
• If the No of Key fields is 3, all the three fields is called Generic area which will be stored in the Buffer.

☐ Log data changes — If this option is selected, all the changes done to the data,
ie, Create, change, Delete will be recorded.

USING CURRENCY/QUANTITY FIELDS:

CURR: It is the data type which is used for defining currency or amount fields.

CUKY: It is the data type used for defining currency key for the above currency or amount fields.

QUAN: It is the data type used for defining quantity fields.

UNIT: It is the data type used for defining the units for the quantity fields.

CURR
CUKY
QUAN
UNIT

STEPS for defining currency/quantity fields"

1. Define a field with datatype 'CURR/QUAN'
2. Define another field with data type 'CUKY/UNIT'
3. Create a link between the above two by providing reference Table Name and Field Name.

Define a Table with the Below fields: TNAME: ZCUST_TABR

Field Name	Data Element Name	Reference Table Name	Reference Field Name
CNOR	ZENOR	-	-
CNAMER	ZENAMER	-	-
CITYR	ZCITYR	-	-
AmountR	ZAmountR	ZCUST_TABR	CURRENCYR
CurrencyR	ZCKey	-	-
QuantityR	ZQKey	ZCUST_TABR	UNITS
UNITYR	ZUnits	-	-

STRUCTURE:"

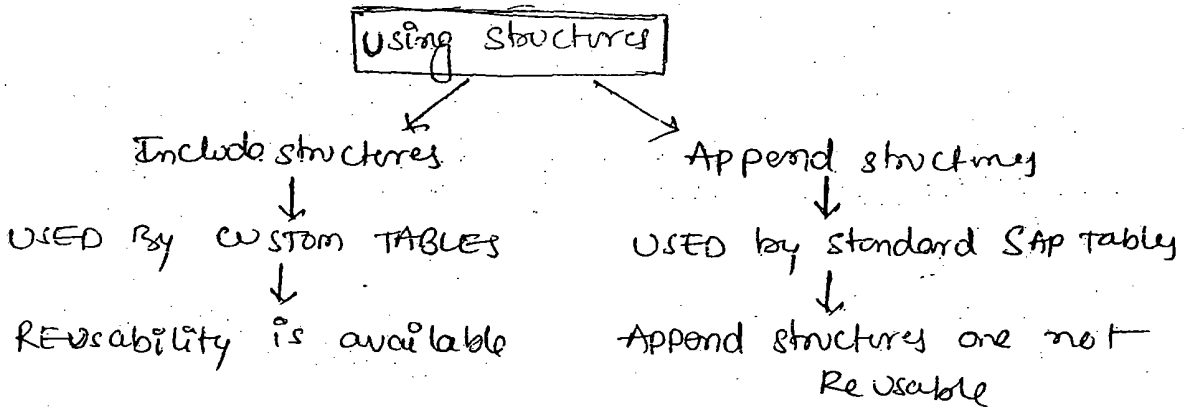
It is an object which Contains a set of re-usable fields.

- Basically a structure is group of fields
- Structures are used to store a single record.
- The main advantage of structure is re-using the same structure in more than one Table.
- Structures are defined in SE11 transaction Code.

RE-usability of Structures

There are two options available for using the structure

1. Include structure
2. Append structure.



Include Structures

- This option is used by Custom tables.
- Include structures are re-usable

SYNTAX:

```
• Include <Structure Name>
```

Ex: Give Field Name as . Include
and data Element as <Structure Name>

Steps for using Include Structure

* **Business Requirements**
  - Create a structure which holds address fields and re-use them in customer, vendor, Employee Tables.

1. Go to SE11
2. Give the data type as ZADDR
3. Click on [Create]
4. Select Structure and press Enter
5. provide the description as a structure to hold address fields.
6. Define the following fields.

Component	Component type	Datatype	Length	dec	descript
CITY	ZCITY	CHAR			
LAND1	ZLAND1	CHAR			
POST CODE	ZSTLZ	CHAR			

7. Save and activate it

RE-Using structure in Table

- Create a Table with MANDT, Customer Number fields.
- Include the structure by giving the field Name as
  - Include, data element name as Structure Name.

Field	Data Element
MANDT	MANDT
CNO	KUNNR
CNAME	VCANAME
• Include	ZADDR

- Save and activate
- Similarly Create another Table and Re-use the same Structure

Using APPEND Structure (In Real time we will not go for this, If need permissions needed)

- Append structures are used for adding Extra fields to the standard SAP Table
- The append structures are not re-usable

Steps for append structure

- Go to SE11
- Give the Table Name as KNA1
- click on display
- Click on the button Append structure
- click on the Icon create append
- Give the append Name as ZSTR1 and press Enter
- Give the field Name as MiddleName, Give the data Element as Name1, press Enter
- Save and activate it.

* In the Real time we generally add Maximum 2/3 fields to the standard SAP Table

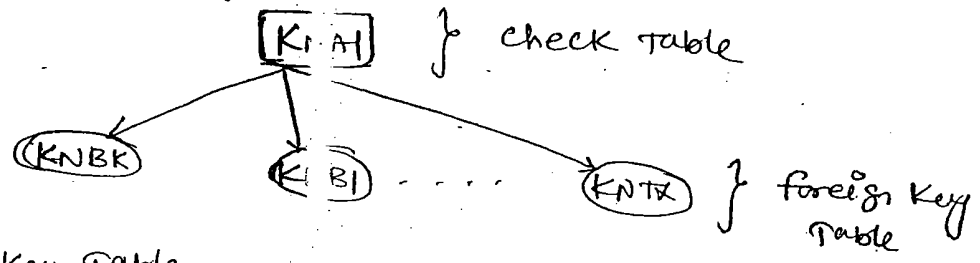
FOREIGN KEY RELATIONSHIPS:

- The Relation between two or more Tables for Table field validation is called foreign key Relationship
- The Table field validation mean's, checking whether the Table field value is correct or not is called validation
- Check Table
  - It is a Table which stores the original list of the Values or main data is called CKECK Table.
  - It is also called as Master Table, parent Table or primary Table.

Eg: KNA1, LFA1, MARA, T001W, T005

T001W - plants master Table

T005 - Country code Table



Foreign Key Table

- A Table which is linked with check Table for field validation is called foreign Key Table

Eg: KNBK, KNBI, KNTX.

Foreign Key Field

- A field in the check Table which is linked with foreign Key Table field is called FOREIGN KEY FIELD.

Eg: KNA1 - KUNNR is called foreign Key field.

PRE REQUISITES FOR FOREIGN KEY Relation

- The domain Names must be same for CHECK Table, foreign Key Table
- The foreign Key field must be a primary Key in the check table.

STEPS TO Create foreign Key Relationships

- Create a Table by Name: YYKNA1R with Below fields

mandt	☑	mandt
Kunnr	☐	YYKUNNR
Name1		Name
Land1		Land

Create 5 Records in this Table

— Create a table by the Name YYKNBKR with Below fields

mandt	☐	mandt
KUNNR	☐	YYKUNNR
BankName	☐	YBankName
ORT01	☐	ORT01

— Create a foreign key relation as Below

- open the Table YYKNBKR
- Select KUNNR field
- click on foreign key [F4] Button
- Enter check Table Name YYKNAIR
- click on [Generate proposal]
- Click on [Copy] Button
- Save and activate

— TESTING

open the Table YYKNBK
↳ Utilities
↳ Table Contents
↳ Create

- Enter wrong Customer No which is not available in YYKNA1
- The Error will be displayed.

ASSIGNMENT

- Create a check Table by Name YYCURR39 with Below fields, mandt ☑ mandt — ZCURR39
  Currency ☑ YCurrency
  Text ☐ YText
- Create a foreign Key Table by Name YYCUST with Below fields, mandt ☑ mandt — ZCURR40
  CUSTNO ☐ YYKUNNR
  Amount ☐ YYAmount

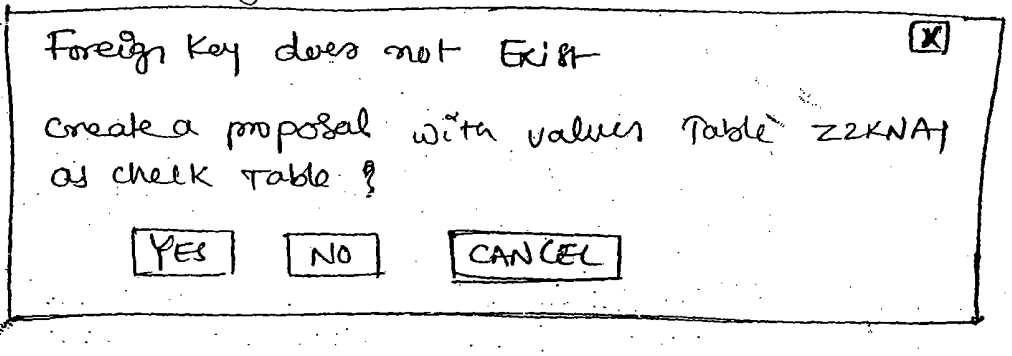
* Create a foreign key Relation Between

YYCURR – CURRENCY

YYCUST – CURRENCY

VALUE TABLE

- A Table Name defines at the domain level, So that a the Table fields will be validated against a single ~~check~~ Table which is called as value Table.
- In simple words, when you click on foreign key button System will display a message as below.



- If you click on YES the value Table will be Converted as check Table.
- If you click on NO we can Enter our own check Table Name.

STEPS To Create a value Table

1. Create a Table by Name Z2KNA1R with Below fields

MANDT	☑	MANDT	ZRAM01
KUNNR	☐	Z2KUNNR	
NAME1	☐	NAME1	
LAND1	☐	LAND1	

2. Double click on data Element Z2KUNNR
   - Double click on Z2KUNNR click on value range Tab
   - Specify value Table as Z2KNA1
   - Save and activate it.

3. Create a Table by Name Z2KNBK with fields Same as YYKNBK

4. Create a foreign Key relation between the above two Tables.

5. The check Table will be automatically proposed.

6. Finally Test it, by Entering Wrong Entry which is not available in check Table.

** TABLE MAINTAINENCE GENERATOR (SM30)

- It is used to maintain the BULK amount of data
- Maintainence means Creation, deletion, modification of the records.
- It is also used for validating the Table records by using the Events.
- SM30 is the Transaction Code for Table maintainence generator

STEPS TO Create TMG

- Go to SE11
- Create a table by Name ZCUST_TABLE with the Below Fields,
  MANDT
  KUNNR
  NAME1
  LAND1
  ORT01
- Save and activate
- click on Utilities,
  ↳ Table maintainence generator
- provide the Below properties
  - Give the authorization group as &NC& (without authorization group)
  - Give the function group ZCUST_TABLE (Give Table Name)
  - Select the maintainence as ONE step ⊙
  - click on the Button find screen Number

- System will propose the Screen Number.
- Click on the Icon (Create)
- The TMG will be created (standard function modules will be created)
- Click on Save.

- Go to SE80
- Select the function group from the list
- Give the function group as ZCOST_Table press Enter.
- Right click on function group Name → activate

By the above steps the TMG will be created and activated.

MAINTAINING THE TABLE DATA

We have two steps.

①
- Go to SM30
  → Give the Table Name ZCOST_TABLE
- Click on the maintain Button
- Click on the New records Button
- Enter the data and Save it.

② open the Table → Click on Utility
→ Table Contents
→ Create

***

NOTE

- Once the TMG is generated for a Table, Then this TMG will not work if you make any changes to the Table again (Like adding Extra Field)
- Suppose, If Extra field is added or Table is changed Then the TMG must be deleted and Regenerated.

Assignment

✓ Add Extra field by Name PSTLZ, Save and activate.
✓ Now Re-generate the TMG

★★★ VALIDATIONS WITH TMG

- By using the Events available in TMG we can write the validations
- Totally there are around 35 to 40 Events available.
- To check the Events fallow Below steps.

  - Go to SE11
  - Give the Table Name click on change
  - click on Utilities
    ↳ TMG
  - click on Environment
    ↳ modification
    ↳ Events
  - click on the Button New Entries
  - In the first column of the table, click on Search help Button.
  - All the Events will be displayed.

Event Name	TEXT
01	Before saving the data in the database
02	After saving the data in the database
03	Before deleting the data displayed
04	After deleting the data displayed
05	Creating a New Entry

BUSINESS REQUIREMENT

★ Raise an Error message when Ever the Customer Number is Entered and Customer Name is blank

STEPS

- Go to Utilities
  ↳ Table maintainence generator
- click on Environment
  ↳ modification
  ↳ Events
- Click on the [New Entries]
- Select the Event, 05, ie, Create a New Entry
- Give the form routine Name as CREATE_NEW_REC press Enter.
- Click on the Icon to display ABAP Editor write the Below code,

Name of the form routine

```
FORM CREATE_NEW_REC.
  IF ZCUST_TABLE_NAME1 = ' '.
Table name
    MESSAGE 'please Enter Customer Name' TYPE 'E'.
  ENDIF.
  ENDFORM
```

- Save and activate
- Click on Back, Click on Save
- Click on Back, Click on Save
- Click on Back
- A message will be displayed as function group can't be processed.
- Now go to SE80 and activate the function group.
- Now Test the Table by Entering Customer Number and Leaving Customer Name as Blank.

PROPERTIES OF TMG

Authorisation Group

- It specifys the group of User's who have the accen to maintain the Table data
- Authorisation group is given by BASIS Consultants

Function Group

- It is a Container of function modules which one generated by SAP when Creating TMG
- These function modules will Contain the ABAP Code for TMG

Maintainence type

- It Specifys whether ONE Screen / TWO SCREENS to be displayed for maintaining the data

ONE STEP:

- If this option is selected only a single Screen is displayed for maintaining the Table data

TWO STEP:

- If this option is Selected Two Screens one displayed for maintaining the Table data

Recording Routine

- It is a routine which is used to record the Table changes
- If standard recorded routine is selected, all the changes are recorded by standard routine
- If No or User recording routine is selected No changes are recorded.
- If User recording routine is available Then Those Changes are recorded by User routine

INDEXES

- These are used to read the data from a database Table with less span of time

There are two types of Indexes

1. Primary Index
2. Secondary Index

Primary Index

- An Index which is created on the Key fields automatically By SAP is called primary Index

Secondary Index

- An Index which is created on the Non Key fields by the ABAP Consultant is called Secondary Index.

★ NOTE - We can Create a Maximum of ★ 9 INDEXES (Secondary Indexes)

- As per the Best programming Technique, Every Select Statement must Contain all the Key fields in the where Condition.
- If Non Key fields are used in the where Condition Then it will Take a long time for Execution.
- So to Improve the performance of the above statement, Create an Index on the above non-Key field called as Secondary Index.

Steps for Creating Secondary Index

- click on the button [INDEXES]
- click on the [Create] Icon
- Give the description
- provide the field name, Save and activate

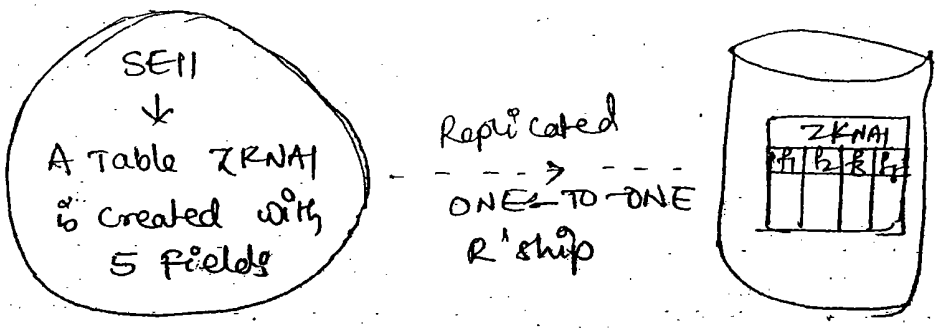
TYPES OF DATABASE TABLES

- There are Three types of data base Tables.

1. Transparent Table
2. pooled Table
3. Cluster Table.

Transparent Table

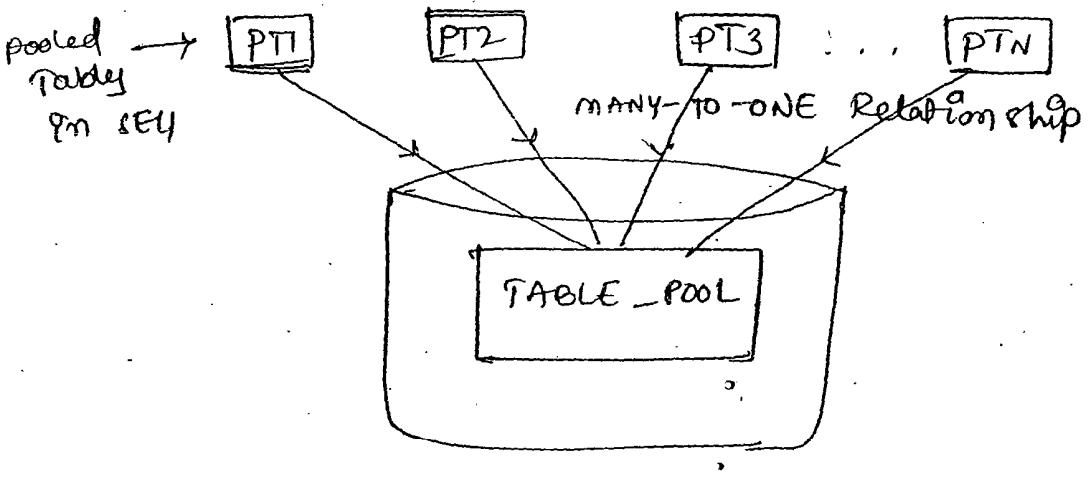
- These tables are used to store the business data i.e, Master and Transaction data.
- Each Transparent Table will have a ONE TO ONE Relationship b/w a Transparent Table in data dictionary and a Transparent Table in data base.
- That is, when ever we Create a Table in SE11 with 5 or six fields, The Same Table will be replecated in the data base with The Same structure. i.e, 5 or 6 Fieleds.



- By default all the Tables Created by ABAP Consultants are Called Transparent Tables.

POOLED TABLES

- These Tables are used to store the System data such as Screen Sequence, Statistical data, ~~Historical~~ data Etc..
- These Tables will have many-to-one Relationship between a Table Created in data dictionary and a Table in data base

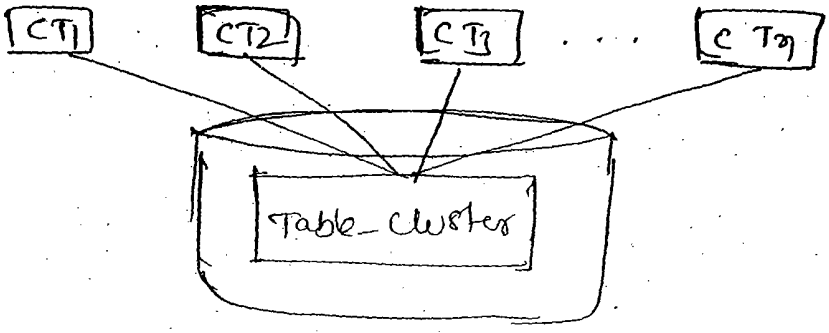


- The Name of the Table which is created in data base is called Table pool.
- The structure of Table pool is as below

1. TABNAME – This field stores the name of pooled Table, Ex: PT1
2. VARKEY – This field stores Key fields of pooled table. Ex: KF1..
3. DATALN – This field stores the lengths of pooled Table Ex: 650 KB
4. VARDATA – This field stores the data of pooled table, Ex: ZABCD..

CLUSTER TABLE

- These Tables are also used to store the system data
- These Tables are same as pooled Tables, but the structure of cluster Table is different from pooled tables.
- These Tables also have many-to-one Relationship



- The structure of Table cluster will contain the below Fields.

1. TABNAME
2. VARKEY
3. DATALN
4. VARDATA
5. PAGENO – This fields stores the page No of data
6. TIMESTAMP – This field stores the date, time, UserName of person Created the data

Example

Transparent Table : KNA1, LFA1, MARA

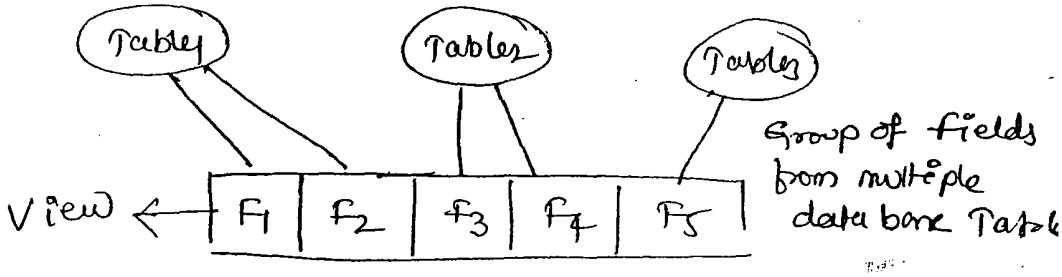
Pooled Table : m_ MTVMA/ m_MTVMB/ m_MVERB/ m_MVERC

** Cluster Table : BSEG / BSES/ BSEC / AUAA / AUAT

www.3dm.sap.com

"VIEW"

- A view is an object which holds or stores fiele from multiple data base Tables



★★

★. View doe't store any data

- View is an imaginary Table
- when Ever a view is Executed, internally SQL statements will be Executed and The data is displayed in view
- View physically doesn't Exists.

TYPES OF VIEWS:

1. Data base view (Uses Inner JOIN / Read only)
2. projection view (SINGLE Table / Read and write)
3. maintainence view (INNER JOIN / Read and write)
4. Help view (OUTER JOIN / used only in Search Helps)

DATA BASE VIEW

- A view Created on Two or more data base Tables Using inner JOIN is Called data base view.
- Using Data base view we can only Read The data
- we can't do any maintainance.
- These views are Used to generate Simple Reports.

JOINS

- It is a Concept used to link two or more data base tables.
- There are two types of JOINS in ABAP
  1. Inner Join
  2. Outer Join (Left outer Join)

*
** **Inner Join**

- 90% we use this in Real time.
- With this Join ONLY the matching Records between two or three Tables will be displayed.
- The Unmatched records are Not displayed.

OUTER JOIN

- With this Join, all the records from first table are displayed.
- If there is a matching record in Table2, the Table2 data is displayed.
- If there is no matching record in Table 2, the Table2 data is not displayed.
  i.e, It left as Blank

ZKNA1

KUNNR	NAME1
1001	SONY
1003	GE
1004	MAN

ZBANK

KUNNR	BANKNAME	BLODE
1001	SBI	SBI01
1001	AB	AB01
1003	SBH	SBH01

JOINED USING
ZKNA1 ~ KUNNR = ZBANK ~ KUNNR

Using Inner JOIN

KUNNR	BANKNAME	BLODE
1001	SBI	SBI01
1001	AB	AB01
1003	SBH	SBH01

Using outer JOIN

KUNNR	BANKNAME	BLODE
1001	SONY	SBI
1001	SONY	AB
1003	GE	SBH
1004	—	—

An Example on Database view

Business Requirement

⊙ Develop a simple report or view which displays materi details and description details

Tables used

MARA → material master
MAKT → material customer

List of fields in o/p

MATNR → material NO
MTART → material type
MBRSH → Industry sector
SPRAS → Language
MAKTX → Description

STEPS

- Go TO SE11
- Give the view Name as ZDB_VIEW
- click on Create
- Select data base view
- Give description
- Give Table Name as MARA, MAKT click on Relationships
- Select the Relation
  ☑ MAKT — MARA
- Click on copy Button
- The Relation is proposed automatically
- click on view fields TAB.
- click on Table fields Button, Select MARA, click on √ choose button
- Select MATNR, MTART, MBRSH
- click on copy Button
- Again click on Table Table fields button, Select MAKT click on [illegible] Button

- Select SPROS, MAKTS
- Click on Copy
- Click on Selection Condition Table
- Specify the table Name as MAKT, Field Name as SPRAS, operator as EQ, Comparision value as 'E'
- Save, Activate, Test it

Checking the definition of view

- Click on Utilities
  - ↳ Data base object
    - ↳ Display
- It will display the SQL statements.

"PROJECTION VIEW"

- A view Created on a single Table is called projection view
- We can Read the data as well as maintaing the data.
- ★. It is mainly used to minimise the data base Interface. i.e, Reduce the Fields of a Table

STEPS

- Go to SE11
- Give the view Name as ZPRJ_VIEW
- click on Create
- Select projection view
- Give description give the Table Name as MARA
- click on Table fields
- Select the two fields MATNR, MTART, MBRSH, MEINS
- click on Copy Button
- Save activate and Test it.

"MAINTAINENCE VIEW" ✱ (I did't use maintainence view in Real

- It is a view which is created on multiple database Tables using Inner Join Concept
- we can Read the data and we can maintain the data
- maintainence views are specially designed for SAP
- ✱✱ we don't use maintainence view's in the Realtime Because, maintainence should be done through the SAP Screens or transaction Codes,

  To maintain data integrity or data consistency.
- As per the Best programming standards maintainence should't be done through views, As it involves updating multiple data base Tables, But instead use SAP TransSaction Codes.

"HELP VIEWS"

- It is a special type of view which is designed only for Search helps.
- It is a view Created on two or more Tables using outer Join concept
- We can't Execute this view directly because it is only Used with Search Helps.
- We can only Read the data.

STEPS to Create Help views

- Go TO SE11
- Give the view Name as ZHelp_VIEW
- Click on Create, provide description, Give the Table Name as MAKT, Click on Relationships
- Select ☑ MAKT - MARA
- click on Copy
- Click on View fields Tab

- Select the Below fields

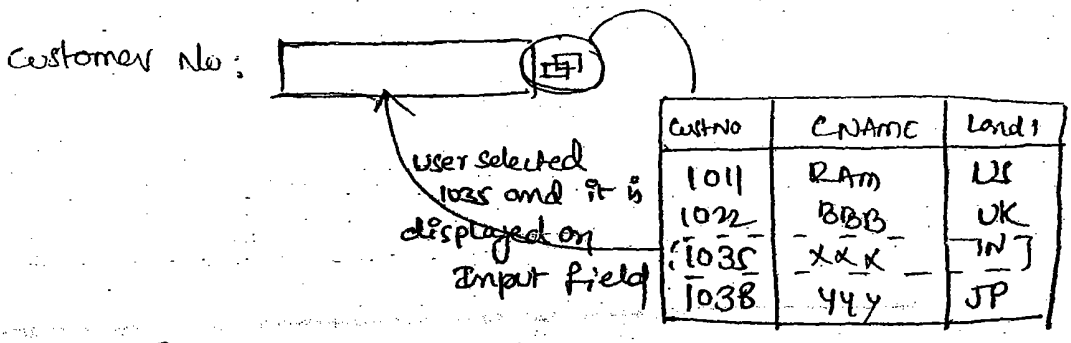
#HelpView: ZHELP_VIEW1

Viewfield	Table
MATNR	MARA
MTART	MARA
MBRSH	MARA
SPRAS	MAKT
MAKTX	MAKT

- Provide the Selection Condition, if any.
- Save and activate
- ★ we can not Execute this View directly As it is used only with Search helps.

" SEARCH HELP "

- providing a Help for the list of the value to be Selected on an input field is called Search help. (or)
- providing F4 functionality for an input field is called a Search Help.



TYPES OF SEARCH HELPS

There are two types of Search Helps,

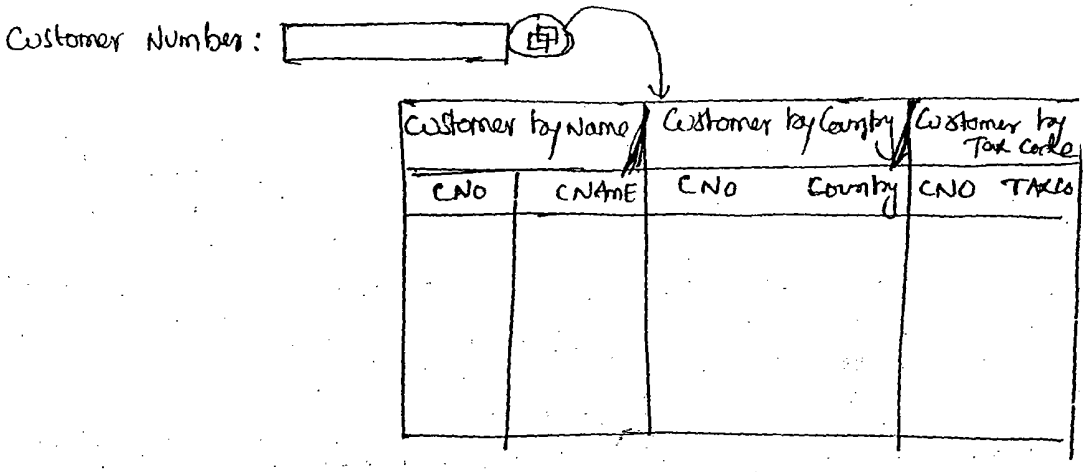
1. Elementary Search Help
2. Collective Search Help

Elementary Search help

- A single search help for an Input field is called Elemy Search help

Collective Search help

- Group of Elementary search helps for a single input field is called collective search help



Steps for Elementary Search Help

- Go to SE 11 ZS_HELP1
- Select Search help, Give a Name Ex, ZEL-MARA.
- click on [Create]
- Select Elementary Search help, Give description
- Give Selection method as MARA
- provide the Search help parameters As Below

Search help parameter	Imp	Exp	LPos	SPos
MATNR	☑	☑	1	1
MTART	☑	☑	2	2

- Save, active, Test it.

Assigning Search helps

- we can Assign the Search help at the Table level and program level

Assigning Search helps at program level

Ex.

```
REPORT ZSH_HELP_PR4.
PARAMETERS: P_NAME(15) TYPE C.
WRITE / P_NAME.
```

- parameters can also be define using Below syntax

PARAMETERS : <P_NAME> TYPE <Table field name> → 99%. we use this only in Real time

Ex: P_MATNR TYPE MARA-MATNR.

"Match Code object"

- It is the statement use to assign custom Search help by replacing the standard search help if it is available.
- If standard Search help is not available, it will display the custom search help directly

Parameters : <P_NAME> TYPE <TABLE-FIELDNAME> matchcode object <Custom Search help Name>

Example

```
REPORT ZSH_HELP_PR4.
PARAMETERS : P_MATNR TYPE MARA-MATNR
             matchcode object ZE_MARA.
```

↓ Displays Custom Search helps

```
Parameters : P_MATNR2 TYPE mara-matnr
```

↓ Displays Standard Search helps.

Assigning Search helps at Table Level

- There are 3 steps to be followed

STEP1: Create a Table

STEP2: Create a Search help

STEP3: Assign the Search help to the Table-Field

Step1:

Create a Table as below fields.

MATNR – ZMATNR (DE)
MTART – ZMTART (DE)
MBRSH – ZMBRSH (DE)
MEINS – ZMEINS (DE)

Create some few records

1. ZRAM_TAB1
2. SH: ZRAM_DBVIEW
   SM: ZRAM_TAB1

STEP②: Create a Search help as below,

Search help Name: ZE_ZMARA ZRAM_SHELP
Selection method: ZMARA ZRAM_HELPVIEW
SH parameters are: MATNR, MTART CNO

Assign the SH to the table fields as below

STEP③:

open the Table ZMARA
Select MATNR field
click on [Search help] button
give SH Name as ZE_ZMARA
Click on [General proposal]
Click on [Copy]
Save and activate Test it.

Collective Search helps

- Group of Elementary Search helps is called Collective Search helps.

STEP① : Create an Elementary Search help by Name ZE_ZMARA with Search parameters: MATNR, MTART

STEP② : Similarly Create another Search help by Name ZE2_ZMARA with SH parameters: MATNR, MEINS

STEP③ : Finally, Group the above two Search helps and Create a Collection Search help as below.

- GO TO SE11
- Give Name as ZCOLL_ZMARA — ZC_SHELP
- Give description
- Give Search help parameters as below

Field Name	Imp	Exp	Data Elemnt
MATNR	☑	☑	ZMATNR
MTART	☑	☑	ZMTART
MEINS	☑	☑	ZMEINS

- Check on Included Search help's Table
- Give first Name as ZE_ZMARA and click on PARAMETER ASSIGNMENT Button
- System will generate a proposal → click on YES
- Similarly Give Second SH Name as ZE2_ZMARA and click on parameter assignment
- System will generate a proposal → click on YES
- Save — activate → Test it

Assigning Collective Search help at program level:

```
PARAMETERS: P_MATNR TYPE MARA-MTNR matchcode object
                    ZColl_ZMARA.
```

★ "NOTE"

Similarly assign the same Search help at Table level also by clicking on Search help button.

Input Screen → SelectionScreen
Logic → ABAP Code
output Screen →

SEARCH HELP PROPERTIES:

Selection method

- It specifies either the Table Name or view Name so that the value's can be selected for Search help. */ Create a view give the view name here

LPOS

- It specifies the list position on the Screen

SPOS

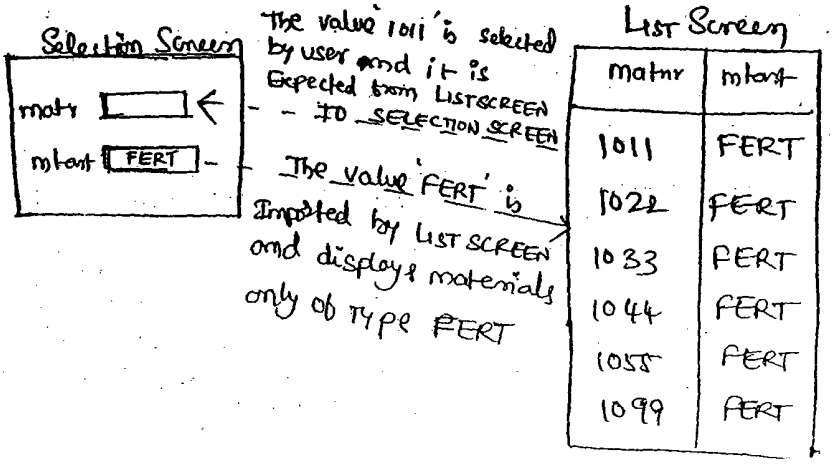
- It specifies the Selection Screen position of the field.

IMPORTING

. If this check box is selected, the value is imported by list-Screen from the Selection Screen.

EXPORTING:

. If this check box is Selected, the selected value by the user will be Exported from list Screen to the Selection Screen.



"DIALOG TYPES"

- we have 3 options available Here,

*NOTE :
Dialog means a Screen in SAP

1. Display values immediately
   - The values are immediately displayed as Soon as we click on The Search help button
2. Dialog with value restriction
   - The values are not displayed immediately but instead a Screen is displayed where you can put a filter value.
3. Dialog depends on Set of values
   - It is a combination of above Two options,
     ie, If The Set of values are less than 500, The values are immediately displayed.
   - If The Set of values are more Than 500 a dialog Screen is displayed.

USING HELP VIEW IN SEARCH HELPS

1) Step①: Create a help view with the fields MATNR, MTART, SPRAS, MAKTX Save and activate

   STEP②: Create an Elementary Search help by The Name ZE_MATRA_hv

   STEP③: provide The Selection method as ZHELP_VIEW (ViewName)

   Step④: provide The Search help parameters as Below,
   MATNR, MTART, SPRAS, MAKTS

   Step⑤: Save, activate and Test it

2) step⑥: Assign The above Search help at table level or Field Level

"TYPE GROUP" (* In the Real time we won't create it)

It is an object which contains the Re-Usable User defined data types.

Eg: SLIS is a type group which contains all the Re-Usable User defined data types related to ALV Reports.

NOTE

- we don't create Any type groups in the Real time

Steps to Create TG

- Go to SE11
- Select type Group give a Name as ZTYP
- click on Create
- Write the Below Code

```
TYPE-POOL ZTYP.
  TYPES: ZTYP_NAME(25) TYPE C.
  TYPES: ZTYP_LAND1(3) TYPE C.
```

- Save and activate it.

Using the Type Group in a program (SE38)

```
TYPE-POOLS : ZTYP
DATA: V_NAME TYPE ZTYP_NAME.
DATA: V_LAND1 TYPE ZTYP_LAND1.
```

"Note:

Type pools is a statement which is used to access the User defined types available in a type group

Syn: TYPE-POOLS <TYPE GROUP NAME>

Properties of domain

SIGN : If this option is selected we can store a sign of -ve Number

- If this option is Not Selected the sign is not Stored and it is considered as +ve Number

LOWERCASE

- If this option is selected the characters are stored as it is Entered on the SAP Screen field
- If it is not selected Every character is stored in the form of CAPITAL letters

CONVERSION ROUTINE

- It is a function module which is used to Convert a value from internal format to External format and External to internal format.

- Below are the common function modules which are used for Conversion routines

1. CONVERSION_EXIT_ALPHA_INPUT

It is a function module or Conversion routine which is Used to Convert External to internal format.

2. CONVERSION_EXIT_ALPHA_OUTPUT

It is a function module or Conversion routine which is used to Convert a value from internal to External format

FIXED VALUES / (Value Range)

- we can define a list of fixed values for a domain. So that it will accepts only those values.
- Any other values is treated as Error.
- The Use of fixed values is VALIDATION
- Validation for a field can be done through foreign key relation or fixed values in the domain.

Ex: GENDER, ~~VSARCH~~ VBTYP

which consists of FIXED values

ASSIGNMENT

Create a Table ZSTUDENT with the Below fields

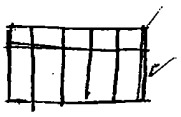
SNO, SNAME, GENDER

NOTE:

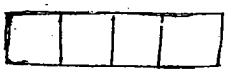
1. The SNAME fields should allow the values in the form of CAPITAL and small letters
2. The field GENDER should allow only M, F values

"INTERNAL TABLES" (Heart of ABAP)

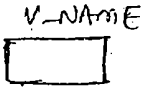
- It is a Temporary memory location which is Created during runtime of a progray to store the data of data base Tables. (RUNTIME INSTANCE OF DATA BASE TABLE) / XEROX COPY OF DB Table
- It is a run-time instance of data base Table.
- It Can store 'n' number of records

The Symbol for Internal Table is, 

WORK AREA

- It is also a temporary memory location which Can store a single Record.
- Symbol for work area is, 

VARIABLE

- It is also a Temporary memory location which stores a Single Value.
- The Symbol for variable is, 

Why Internal Tables, work Areas

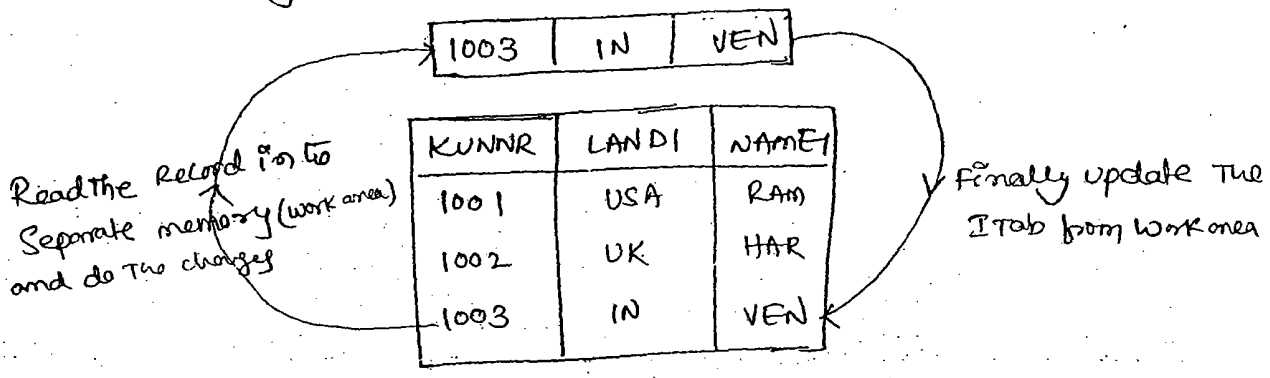
- Instead of working directly on data base Tables for modifications we Take runtime instences of data base Tables Called as internal Tables and do all our modifications in to the internal Tables Using work areas and finally update the database Tables or we Can generate a report.
- By using the internal Tables, we are not directly working on data base Tables. There by Reducing the Load on the data base.

Why work Areas

- In order to do any operations (Modifications) on the internal Table we need to use work area.

  i, Read the Record from the internal Table in to a Saparate memory called as work area.

- Do the modifications on the work area.
- Finally update the internal Table from work area or display the data from work area.



SYNTAX FOR INTERNAL TABLE (ITAB)

```
DATA : <ITAB NAME>  TYPE Table of <DB TABLE NAME>
                    (or)
DATA : <ITAB NAME>  TYPE Table of <USER DEFINED TYPE>
```

Eg: DATA : I_KNA1 TYPE Table of KNA1

DATA : I_KNA1 TYPE Table of TY_KNA1

NOTE:

- when Ever an internal Table is Created with the above Syntax, 8KB size is allocated for internal Table.
- It Expands dynamically depending up on the data

SYNTAX FOR WORK AREA

```
DATA : <Work Area Name>  TYPE <Data base Table>
                (or)
DATA : <Work Area Name>  TYPE <User defined type>
```

Ex: Data: WA_KNA1 TYPE KNA1

DATA: WA_KNA1 TYPE TY_KNA1

ACCESSING work Area fields

```
<WA NAME> -<Field Name> = 'Value'
```

Ex: wa_KNA1-KUNNR = '1011'

wa_KNA1-NAME1 = 'ABCD'

APPEND WA_KNA1 TO I_KNA1.

Reading data from data base Table

- SELECT is the statement which is used to read the data from data base Table in to internal Table.

SYNTAX:

```
Select  F1
        F2
        F3
        .
        .
        (or)
        *
        FROM <DB-TABLE NAME>
        INTO TABLE <INT_TABLE>
        WHERE <CONDITION>.
```

Ex: Select * From KNA1

in to Table I_KNA1

where LAND1 = 'US'.

Displaying Internal Table data

- To display Each record from internal Table, we need to re Each record in to work area and Then Use The WRITE statement and display Each Field.

- Below is the syntax to read Each record from internal Table in to work area.

```
Loop at <ITAB NAME> in to <WA NAME>
. . . . .
. . . . .
. . . . .
ENDLOOP.
```

Ex: Loop at I_KNA1 in to WA_KNA1.
. . . . .
. . . . .
. . . . .
Endloop.

BUSINESS REQUIREMENT

- Develop a report which displays Customer details
- The details are Customer Number, Country, Name, City

Tables Used : KNA1
Fields Used : KUNNR, LAND1, NAME1, ORT01

```
DATA : I_KNA1 TYPE TABLE OF KNA1.
DATA : WA_KNA1 TYPE KNA1.

BREAK POINT.

SELECT * FROM KNA1
    INTO TABLE I_KNA1.
Loop AT I_KNA1 INTO WA_KNA1.
WRITE : / WA_KNA1-KUNNR,
          WA_KNA1-LAND1,
          WA_KNA1-NAME1,
          WA_KNA1-ORT01,
ENDLOOP.
```

Assignment

- Develop a report to display material details.
  The details are, MAterial No, material type, Industry Sector, UNITS.
- Develop a report which displays Vendor details like
  Vendor Num, Vendor Name, Telephone Numbe, FAX Number, CITY

TYPES :

. It is a statement used to declare our own Custom data types. Then we can declare our variables reffering to the above User defined data types.

SYNTAX

```
TYPES : BEGIN OF  < User defined TYPENAME >
          F1 ..... ,
          F2 ..... ,
          F3 ..... ,
        End of  < User defined TYPENAME >
```

Ex :

```
TYPES : Begin of TY_KNA1,
          KUNNR Type KNA1-KUNNR,
          Land1 type KNA1-land1,
          Name1 TYPE KNA1-Name1,
        End of TY_Kna1.

Data : I_Kna1 TYPE Table of TY_KNA1.
       wa_ KNA1 TYPE TY_KNA1.
```

Example program on Customer details

Z CUST_REP

```
TYPES : BEGIN OF TY_KNA1,
          KUNNR TYPE KNA1-KUNNR,
          LAND1 TYPE KNA1-LAND1,
          NAME1 TYPE KNA1-NAME1,
        END OF TY_KNA1.

DATA : I_KNA1 TYPE TABLE OF TY_KNA1.

DATA : WA_KNA1 TYPE TY_KNA1.

SELECT KUNNR
       Land1
       name1
       From KNA1
       Into Table I_KNA1.

Loop at I_KNA1 into WA_KNA1
 write : / WA_KNA1-KUNNR,
           WA_KNA1-LAND1,
           WA_KNA1-name1.

End loop.
```

Into Corresponding fields

- This statement is used to compare each field from the source to the target internal table which puts extra burden on the data base.
- So, we should never use this statement in Real time.

Syntax :

```
Select * from <Database Table Name>
into Corresponding fields of
  Table <I_TAB>
```

Best programming Standards:

1. Always declare the Internal Tables Using User defined TYPES statement
2. Never Use SELECT *, Instead use Select list of the Fields.
3. Never Use Into Corresponding fields.
4. The Source list of the fields and the target internal Table field list must be Same

USING PARAMETERS

```
DATA : i_KNA1 TYPE TABLE of TY_KNA1.
DATA : wA_KNA1 TYPE TY_KNA1
Parameters : P_LAND1 TYPE KNA1-Land1.

Select KUNNR
       LAND1
       Name1
       from KNA1
       into Table i_KNA1
       where Land1 = P_land1.
```

List of of the Internal Table operations

1. APPEND
2. INSERT
3. SORT
4. DESCRIBE TABLE
5. READ TABLE
6. LOOP AT < ITAB>
7. MODIFY
8. DELETE
9. DELETE ADJACENT DUPLICATES
10. CLEAR
11. REFRESH
12. FREE
13. COLLECT
14. MOVE - CORRESPONDING
15. APPEND LINES OF
16. INSERT LINES OF

APPEND

- This statement is used to add a single record from workarea to internal Table.

Syn | Append <workArea> to <Internal Table> |

Ex:
```
wa_KNA1-KUNNR = 'SONY01'
wa_KNA1-LAND1 = 'US'
wa_KNA1-NAME1 = 'Ramesh'

Append wa_KNA1 TO I_KNA1.
```

- The record is always added at the bottom of internal Table

INSERT

- This statement is used to insert the record from work area in to internal Table at a specified location.

Syn: | Insert <wa> in to <ITAB> Index <no> |

Ex:
```
wa_KNA1-KUNNR = 'SONY02'
wa-KNA1-LAND1 = 'UK'
wa_KNA1-NAME1 = 'Ravi'

Insert wa-KNA1 into I_KNA1 Index 3.
```

SORT:

- This statement is used to Sort the internal Table data either in ascending order or descending order.

Syn:
```
SORT <ITAB> By <f1>
               <f2>
               <f3>
               :
               <ASC/DESC>
```

Ex:
```
SORT I_KNA1
SORT I_KNA1 By NAME1
SORT I_KNA1 By LAND1 DESCENDING
```

* NOTE : • The Sort Statement Sorts the data by Ascending order by default
• It will Sort based on order of the fields.

ie, $F_1, F_2, F_3 \ldots$

DESCRIBE TABLE

- This statement is used to Find the total No of Records in an internal Table.

Syn: `Describe Table <ITAB> Lines <VAR-NAME>`

Ex: Data : V_Lines TYPE I
Describe Table I_KNA1 Lines V_Lines
Write : / 'Total No of customers are', V_Line.

READ TABLE :

- This Statement is used to read a single record from internal Table in to work area.
- There are Two forms of the read Statement.

Read Table
- with Index
- with Key

Read Table with Index

- This statement will read the record specified by the Index no in to work area

Syn: `Read Table <ITAB> in to <work area> Index <No>`

Ex: Read Table I_KNA1 in to WA_KNA1 Index 3.

Read Table with Key

- This Statement is used to read a single record from Internal Table in to work are which is specified by the field values.

Syn:

```
Read Table <ITAB> into <WA>
    with Key <FNAME1> = <VALUE>
             <FNAME2> = <value>
             <FNAME3> = <value>
             .
             .
             .
    BINARY SEARCH.
```

NOTE:

The pre-requisite for the above statement is, the internal table should be sorted in Ascending Order.

Ex:

```
I_KNA1 into WA_KNA1
    with Key KUNNR = 'SONY01'
    BINARY SEARCH.
```

BINARY SEARCH

- It is one type of Search algorithm which is used to Improve the time taken for reading a record from a Internal Table.

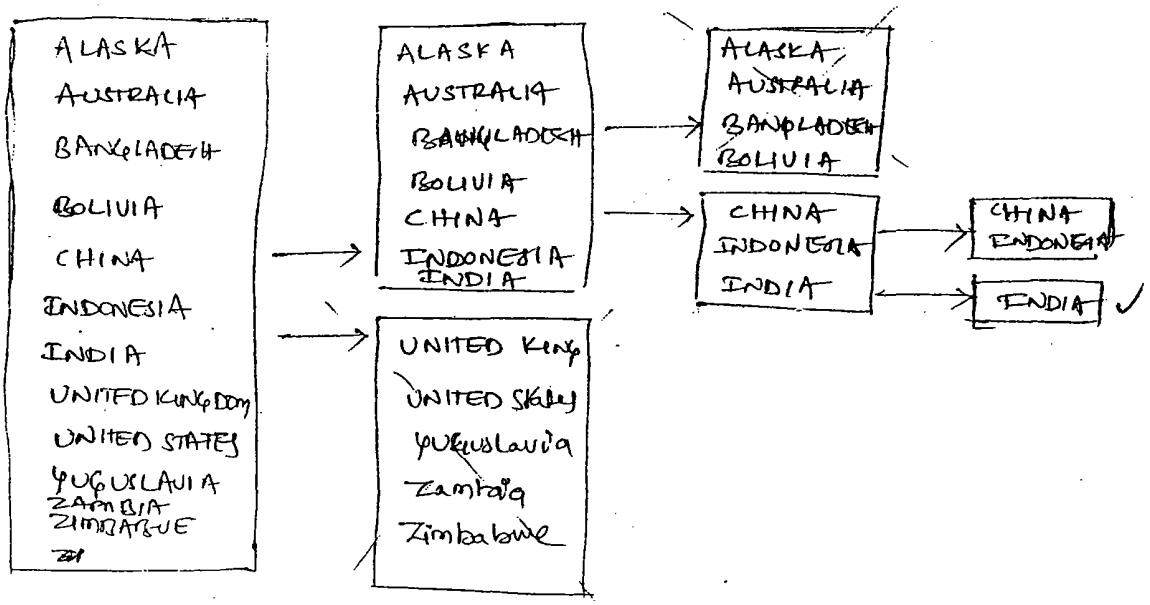
- In the Real Time, Every Read statement will have Binary Search Key word.

↙ Performance Considerations

1.
2.
3.
4. BINARY Search

FUNCTIONALITY OF BINARY Search

- The Binary Search will devide the Entire internal table in to two halfs.



- Then it will Search for the record roughly Either in first half or in the Second half.
- Suppose if the record is available in the first half it will again devide the first half in to two more half's by deleting the remaining second half.
- Again it will Search roughly either in first half or in Second half
- Suppose if the record is available in the first half it will again devide the first half in to two more half's and the procedure repeats contineously until it finds the Exact record.

LOOP ..... END LOOP:

- This statement is Used to read multiple records from internal Table in to work area one by one

Syntax:

```
Loop AT <ITAB> into <WA>
...
...
ENDLoop.
```

Syntax:

```
Loop AT <ITAB> into <wa> from <N1>
                             TO <N2>
. . .
. . .
ENDLoop
```

Syntax:

```
Loop AT <ITAB> into <wa> where <fname1 = val
                               <fname2 = val
. . .
. . .
. . .
ENDLoop
```

Ex:① Loop at I_KNA1 into wa_KNA1
write:/ wa_KNA1-KUNNR, wa_KNA1-NAME1.
ENDLoop.

Ex:② Loop at I_KNA1 into wa_KNA1 from 1 to 10
write:/
ENDLoop.

Ex③: Loop at I_KNA1 into wa_KNA1 where Land1 = 'US'
write:/
ENDLoop.

**Ex:④

```
Loop at I_KNA1 into wa_KNA1
    Loop at I_KNBK into WA_KNBK where
                                KUNNR = wa_KNA1-KUNNR.
    write:/ wa-KNA1-KUNNR, wa_KNA1-NAME1, ..
                                wa-KNBK-BANKS.
    ENDLoop.
ENDLoop.
```

MODIFY:

- This statement is used to modify a single record or multiple records based on the condition

Syn:

```
MODIFY <ITAB> FROM <WA>
              Index <NO>
              Transporting <FNAME1>
                           <FNAME2>
                           <FNAME3>.
```

★★ SY-TABIX (Internal Table Index) ★★

- It is a System variable which holds the Index number of the internal table which is currently process

```
SORT I_KNA1 BY KUNNR ASCENDING.
Read TABLE i_KNA1 into wa_KNA1 with key KUNNR = '0000000002'
                                        BINARY SEARCH
WA_KNA1-NAME1 = 'SONY Electronics - DE'.
WA_KNA1-LAND1 = 'DE'
MODIFY I_KNA1 FROM WA_KNA1
              INDEX SY-TABIX.
              TRANSPORTING NAME1 LAND1.
```

DELETE:

- This statement is used to delete the data from internal table

Syntax:

```
Delete <ITAB> from <WA>
Delete <ITAB> Index <NO>
Delete <ITAB> where <F1> = <value>
                    <F2> = <value>
```

Delete adjacent duplicates

- This statement is used to delete the duplicate records which are adjacent ie, side by side to each other.

- The pre-requisite to this statement is, The internal Table should be Sorted in ascending order.

Syntax:

```
Delete adjacent duplicates from <ITAB> Comparing <F1>
                                                  <F2>
                                                  <F3>
                                                   :
                                                   :
                                                  (or)
                                                  ALL FIELDS.
```

Example (Delete)

```
DELETE I_KNA1 INDEX 3.
DELETE I_KNA1 where KUNNR = 'SONY01'.
```

Example (Delete Adjacent duplicates)

```
WA_KNA1-KUNNR = 'SONY01'.
WA_KNA1-LAND1 = 'IN'.
WA_KNA1-NAME1 = 'SONY ELECTRONICS'.
APPEND WA_KNA1 TO I_KNA1.
APPEND WA_KNA1 TO I_KNA1.
INSERT WA_KNA1 INTO I_KNA1 INDEX 3.
SORT I_KNA1 BY KUNNR.
DELETE Adjacent DUPLICATES from I_KNA1 Comparing
                                  ALL Fields.
```

CLEAR:

- This statement is used to delete the data from a variable or work area or Internal Table

Syn:

```
CLear <VAR_NAME>.
CLEAR <WA>.
CLEAR <ITAB>[].
              BODY
```

REFRESH :

- This statement is used to delete the data from internal Table only.

```
REFRESH < ITABNAME>
```

FREE

- This statement is similar to clear and Refresh
  i, It will also delete the data from a variable, WA, ITAB.
- The difference is, with the clear and refresh statement The data will be deleted but not the memory
- With the Free statement the data will be deleted and memory also be deleted.

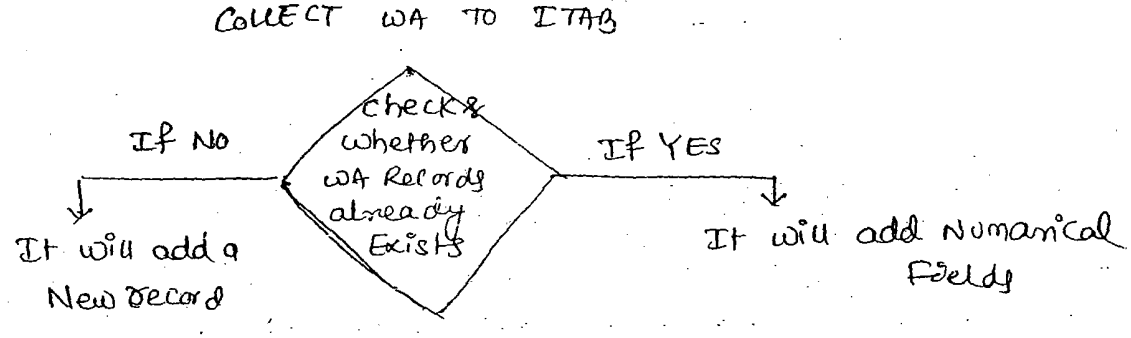
```
FREE <VAR_NAME>
FREE <WA>
FREE <ITAB>
```

Examples on clear, Refresh, Free

1. Append wa_KNA1 TO I_KNA1.
   CLEAR wa_KNA1 (Best programming Technique)
2. CLEAR V_NAME.
3. CLEAR I_KNA1[].
4. REFRESH I_KNA1.
5. FREE wa_KNA1.
6. FREE I_KNA1
7. FREE V_NAME.

COLLECT:

- It is similar to append statement
- The difference between append and collect is, The Collect statement check's whether the work area record already Exists in the internal Table, or not.
- If it Exists it will add the Numarical field.
- If it does't Exists it will add New record to the internal Table.
- Treat is similar to append.



Syn

```
COLLECT <WA>    INTO   <I_TAB>
```

Ex:

```
wa_cust-kunnr = 'SONY01'.
wa_cust-name1 = 'SONY ELE'.
wa_cust-amount = '100'.
COLLECT wa_cust into i_cust.
```

} It will work as Append

```
wa_cust-kunnr = 'SONY02'.
wa_cust-name1 = 'SONY ELE'.
wa_cust-amount = '100'.
COLLECT wa_cust into i_cust.
```

} It will work as Append

```
wa_cust-kunnr = 'SONY01'.
wa_cust-name1 = 'SONY ELE'.
wa_cust-amount = '150'.
COLLECT wa_cust into i_cust.
```

} It will work as Collect ie, It add's a Numarical fields.

APPEND LINES OF:

- This statement is used to add multiple records from one internal Table to another internal Table.

Syn:

```
APPEND LINES OF <ITAB1>  FROM <N1> TO <N2>
                         INTO <ITAB2>
```

Ex:

```
Append lines of I_KNA1  From 3 To 7
                        To I_KNA2
```

Ex:

```
TYPES : Begin of ... End.
Data : I TAB1
Data : wa1
Data : ITAB2
Data : wa2
Select ...
        up to 10 Rows.
Append Lines of ITAB1
                From 3 to 5
                TO ITAB2
SORT
Loop at ITAB2
  .
  .
ENDLOOP
```

INSERT LINES OF

- This statement is used to add the data from one internal Table to another internal Table at a specified INDEX Number.

Syn:

```
INSERT LINES OF <ITAB1>
       FROM <N1> TO <N2>
       INTO <ITAB2> INDEX <NO>
```

EXAMPLE

APPEND LINES OF i_KNA1 FROM 3 TO 5 to i_KNA2

- INSERT LINES OF i_KNA1 FROM 6 TO 8 into i_KNA. INDEX 3.

MOVING Data from one ITAB to another ITAB:

① ITAB2[] = ITAB1[] } BEST PROGRAMMING

② Loop at I_KNA1 into wa_KNA1
wa_KNA2-KUNNR = wa_KNA1-KUNNR.
wa_KNA2-LAND1 = wa_KNA1-Land1
wa_KNA2-Name1 = wa_KNA1-NAme1.
Append wa_KNA2 TO I_KNA2.
End Loop. } BAD PROG TECHNIQUE

MOVE CORRESPONDING:

- This statement is used to move the data for Comerponding fields from one work area to another work area.

Syn: MOVE CORRESPONDING <WA1> TO <WA2>

Ex: Loop at I_KNA1 in to wa_KNA1
MOVE CORRESPONDING wa_KNA1 to wa_KNA2
APPEND wa_KNA2 to I_KNA2
ENDLoop

***NOTE: ↑

This is a very daggerous statement as it has to compare Each field from Source work area to Target work area instead of use Below Statement.

Loop at I_KNA1 in to wa_KNA1
wa_KNA2-KUNNR = wa_KNA1-KUNNR,
Append WA TO ITAB.

OBSOLETE WAYS OF DECLARING ITAB's AND WA's :

– OLD TECHNIQUES

- Below are the syntaxes which are used in the oldest versions.

OCCURS 0

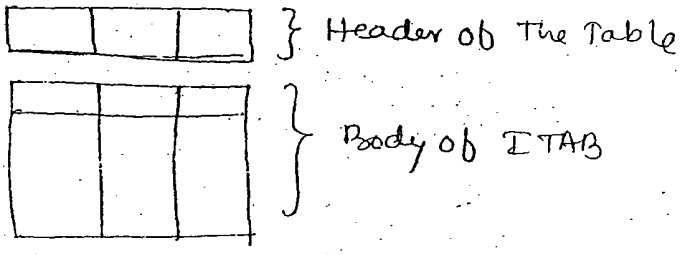
- This statement will create the body of 8 KB size for an internal table

OCCURS 0 with Header line

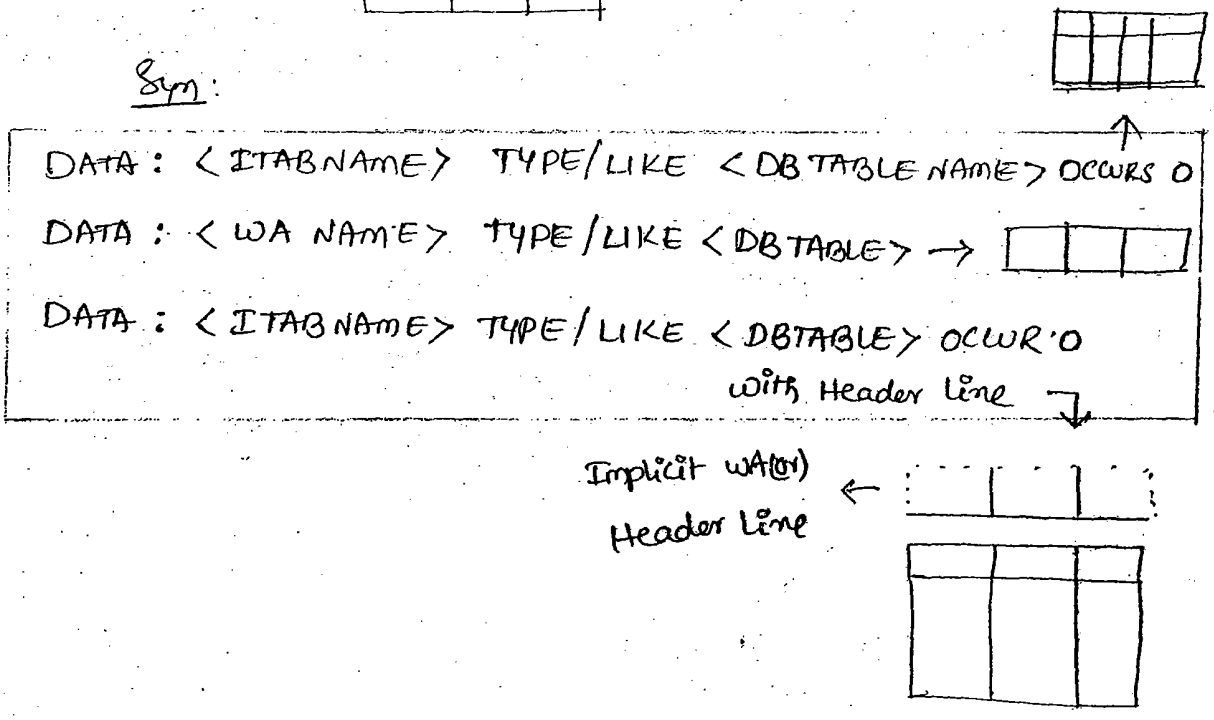
- This statement will create the body as well as an impliscit work area called as Header line.

★ NOTE

- Using the above statement the Name of the internal table and work area will be the same.



Syn:



USING OUR OWN FIELDS

Syn:

```
Data : Begin of <ITABNAME> OCCURS 0,
       <F1> TYPE <DBTABLE-FIELDNAME>
       <F2> TYPE <DBTABLE-FIELDNAME>
       .
       .
       .
       END OF <ITABNAME>
```

with declare ITAB with Header line

- The above syntax is used to declare an internal Table with our own fields.
- It will also create Implicit work area.

★ NOTE

- All the above syntaxes are OBSOLETE, Because these are NOT supported in OO-ABAP.
- In OO-ABAP Every statement should be declare which refers to TYPES statement.

Example prog

```
* DATA : i_KNA1 TYPE KNA1 OCCURS 0.
* DATA : wa_KNA1 TYPE KNA1.

DATA: I_KNA1 TYPE KNA1 OCCURS 0 WITH HEADERLINE.
 Select *
      FROM KNA1
       into Table i_KNA1
          UPTO 10 ROWS.
 SORT i_KNA1 BY KUNNR.
Loop AT I_KNA1.
   WRITE: / I_KNA1-KUNNR, I_KNA1-LAND1.
END Loop.
```

DIFFERENCES

ITAB OPERATION	WITH WORK AREA	WITH OUT WORK AREA
APPEND	APPEND <WA> TO <ITAB>	APPEND <ITAB>
COLLECT	COLLECT <WA> into <ITAB>	COLLECT <ITAB>
READ	READ TABLE <ITAB> into <WA> with ... Key	READ Table <ITAB> ... Key
Loop	LOOP AT <ITAB> Into <WA> ENDLOOP	Loop at <ITAB> ENDLoop

TYPES OF INTERNAL TABLE

- There are 3 Types of internal Tables.
  1. Standard
  2. Sorted (Indexed)
  3. Hashed Internal Tables

*STANDARD

- These are default internal Tables
- We use either Key operation or INDEX operation to read a record.
- We can use either binary Search or Linear Search for reading a record.
- The response time depends on the no of records on internal table
- If we are using BINARYSEARCH Then Response Time = $\log_2$ (No of Rec)
- These internal tables can be Sorted.

SORTED (INDEXED)

- Not Default internal Tables.
- *These are special type of ITAB's where the data is automatically Sorted when ever we insert a record.
- * We can not use APPEND Statement Here
- We can use either KEY operation or INDEX to read a Record
- We can use either Binary Search or Linear Search
- The Response Time depends on No of Records.
- RT = $\log_2$ (No of Records)
- * These Tables ... NO KEY operation ...

HASHED

- These are used by only SAP
- Not default internal Tables.
- These internal Tables are used only when we work with all fields of the DB Tables.
- *ie, when ITAB resembles Data base Tables
- These are used only when we work with Large data sets.
  ie, More No of Records (10,00,000)
- * The Response Time is always Fixed irrespective of no of records, Because it uses Hashed algorithm.
- * It uses only INDEX operation

SYNTAX

- DATA : <ITAB> TYPE Standard Table of <DBT/UDTYPE>
- DATA : <ITAB> TYPE Sorted Table of <DBT/UDTYPE> WITH UNIQUE KEY MATNR
- DATA : <ITAB> TYPE Hashed Tables of <DBTable>

1. Develop a Customer master report which display's CNO, NAME, Country, CITY, Telephone No, fax No 2113

The Condition is,

DISPLAY only customer's of Country 'DE'.

o/p

CNO	NAME	COUNTRY	CITY	TPHONE	FAX

"OPEN SQL"

- SAP R3 Software is independent of the Data base being used, i.e., SAP can use any data base to store the data.

SQL (Structured Query Language)

- We use data bases to store Mass Amount of data
- Whenever we want to store data in to data base, we need to use the respective data base Language statements.

  i.e., Each data base can understand it's own data base Language statements.

  1. Oracle data base understand's ORACLE statements
  2. SQL data base understands SQL SERVER statements
  3. DB2 data base understands DB2 SERVER statements

R/3 INTERFACE LAYER

- It is an interface which is given by standard SAP S/w which is used to convert the abap Language statements in to Respective data base statements and vice-versa.

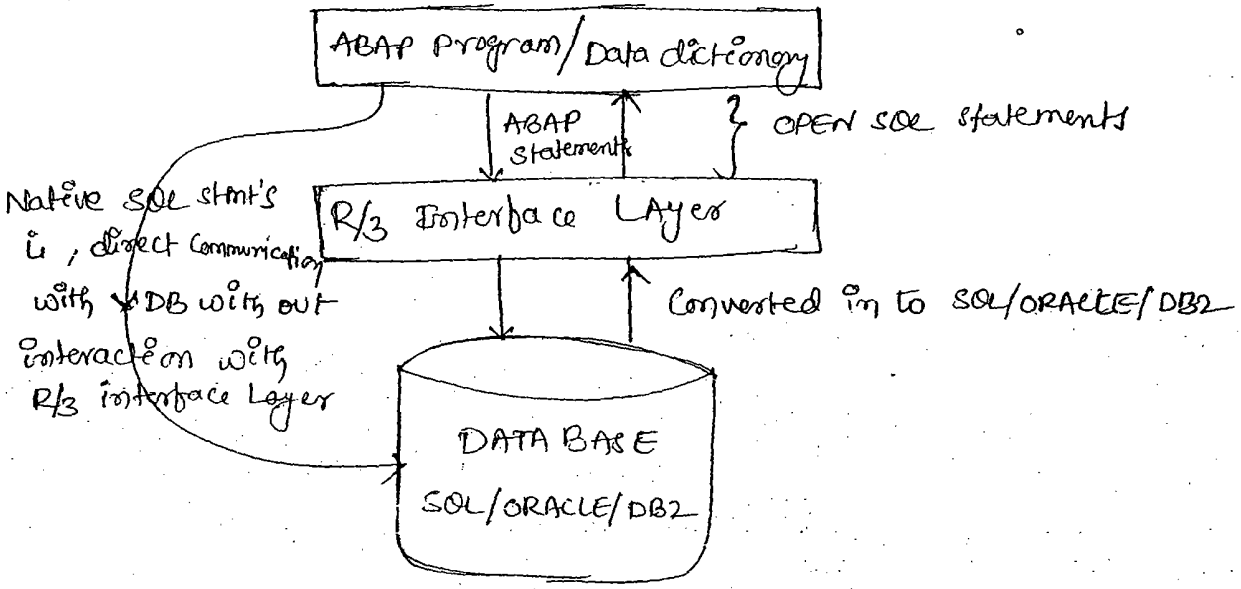
- There are two types of statements
  1. Open SQL statements
  2. Native SQL statements

OPEN SQL STATEMENTS

- The statements which is converted by R3 interface LAYER from ABAP to Respective DB and vice versa are called as OPEN SQL statements.
- In the Real time we will only work OPEN SQL statements

NATIVE SQL STATEMENTS

- These statements directly communicates with the database with out interacting with R3 interface Layer are called Native SQL statements.
- ★ We Never worked in Real time on this



Examples on OPEN SQL Statements

INSERT :
INSERT <DB TABLE> from <WA>
INSERT <DB TABLE> From Table <ITAB>

MODIFY :
MODIFY <DBTABLE> from <WA>
MODIFY <DBTABLE> From Table <ITAB>

UPDATE :
UPDATE <DBTABLE> from <WA>
UPDATE <DBTABLE> from Table <ITAB>

DELETE :
DELETE <DBTABLE> from <WA>
DELETE <DBTABLE> From Table <ITAB>

PRE REQUISITE FOR OPEN SQL STATEMENTS

- Always work with Internal Tables Rather than work area's to insert / update / modify data in to database table
- The structure of internal table / work area must be same as Data base Table

Ex: DATA : I_KNA1 TYPE Table of KNA1 ✓
DATA : I_KNA1 TYPE Table of M_KNA ✗

SYSTEM VARIABLES

- There are two system variables which are updated automatic for each open SQL operation.

(1) SY - SUBRC → Return code
System Sub operation

- It is a system variable which stores the Return code of an OPEN SQL operation
- If SY-SUBRC = 0,
  It means the operation is Succesfully Executed
- If SY - SUBRC ≠ 0,
  It means the operation is failed.

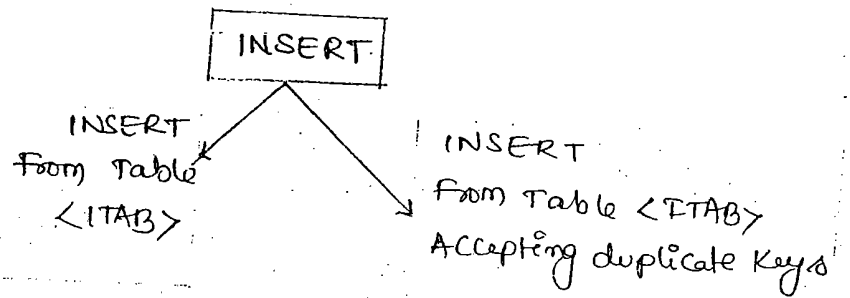
(2) SY - DBCNT System Database Control

- It is a system variable which stores the No of records Succesfully processed.

* SY - DYNNR - Screen No

INSERT

- It is used to insert a single or multiple records from internal table to database table
- SY-SUBRC = 0, means all records were succesfully inserted.
- SY-SUBRC ≠ 0, means only few records are inserted or No record is inserted.
- There are two forms of insert statement.



INSERT FROM TABLE

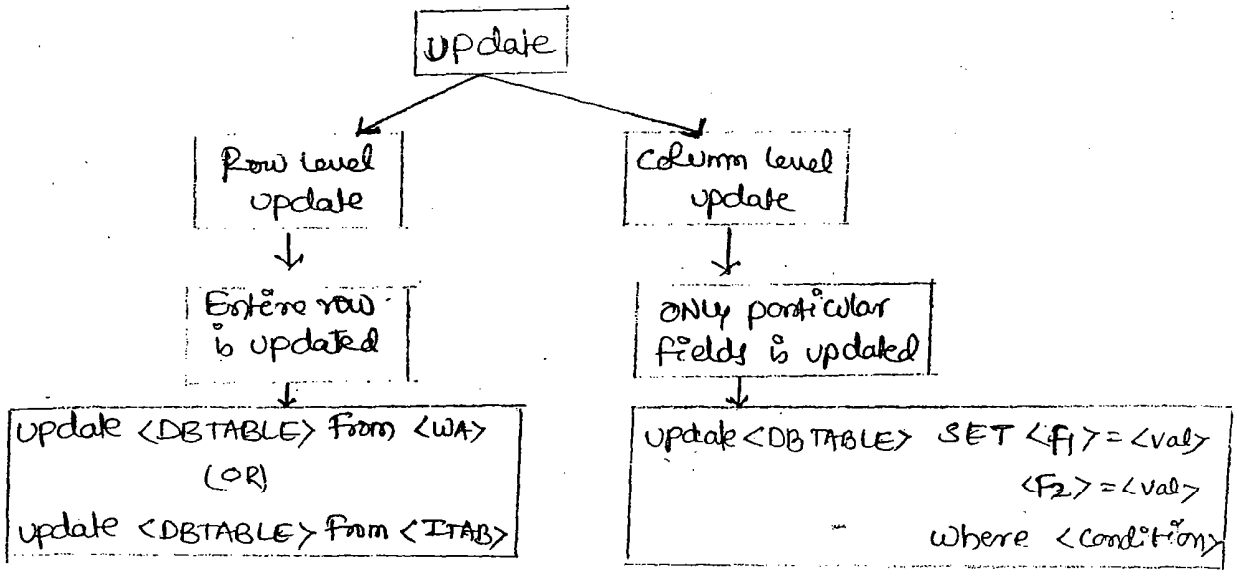
- This statement will insert the records in to database tables.
- Suppose If there are any duplicate records in the internal table it will throw an RUNTIME Error.

INSERT FROM TABLE ACCEPTING DUPLICATE KEYS

- If we use this statement it will not throw any Runtime Error Even though it has duplicate records.
- It is just use to avoid Runtime Error.

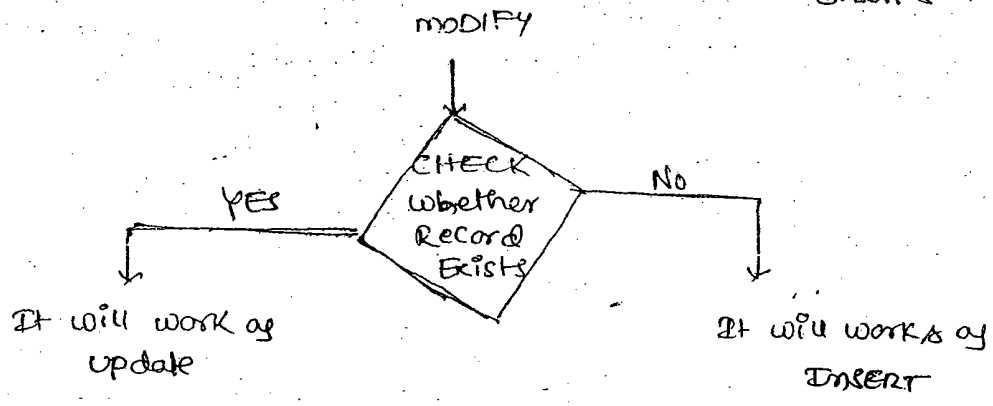
UPDATE:

- This statement will update the record which is available in the data base table.
- SY-SUBRC = 0, mean's If at-least one record is updated
  SY-SUBRC ≠ 0, mean's Non of the record's are updated.
- There are two form's for updat statement.



MODIFY

- This statement is used either to modify a record i.e, UPDATE or INSERT a new record, i.e, INSERT
- SY-SUBRC is always be '0', Because it may modify or Insert's a new record.



DELETE

- This statement is used to delete either a single or multiple records from data base Table.
- SY-SUBRC = 0 means all Records were Successfully deleted.
  SY-SUBRC ≠ 0 means No Records were deleted.

Examples ON INSERT, MODIFY, UPDATE, DELETE

- Business Requirement

To Store the following PLANT Data in to SAP data base Table.

Table Name : T001W (PLANTS Master Table / Check Table)

Field Name : WERKS - Plant Code

NAME1 - Plant Name

INSERT

```
wa_T001W-WERKS = 'HYD1'.
wa_T001W-NAME1 = 'HYD PLANT1'.
Append WA_T001W TO I_T001W.
wa_T001W-WERKS = 'HYD2'
wa_T001W-NAME1 = 'HYD PLANT2'.
Append WA_T001W TO I_T001W.
INSERT T001W FROM TABLE I_T001W Accepting duplicate Keys.
If SY-SUBRC = 0.
  WRITE: / 'Plants are Succesfully INSERTED', SY-DBCNT.
If SY-SUBRC ≠ 0.
  write: / 'few plants are inserted', SY-DBCNT.
ENDIF.
```

UPDATE * Column Level update

```
wa_T001W-WERKS = 'HYD1'.
wa_T001W-NAME1 = 'HYDERABAD PLANT1'.
Append wa_T001W TO I_T001W.
wa_T001W-WERKS = 'HYD2'.
wa_T001W-NAME1 = 'HYDERABAD PLANT2'.
Append wa_T001W TO I_T001W.
Update T001W From Table I_T001W
```

```
update T001W SET BWKEY = '01'
        where WERKS = 'HYD1'.
If SY-SUBRC = 0
    WRITE : / 'plant is succesfully updated', SY-DBCNT.
If SY-SUBRC ≠ 0
    WRITE : / 'No plant is updated', SY-DBCNT.
```

MODIFY

* modify for inserting New record

```
WA_T001W - WERKS = 'HYB3'
WA_T001W - NAME1 = 'HYDERABAD PLANT3'.
APPEND WA_T001W TO I_T001W.
WA_T001W - WERKS = 'HYD4'.
WA_T001W - NAME1 = 'HYDERABAD PLANT 4'.
APPEND WA_T001W TO I_T001W.
MODIFY T001W FROM TABLE I_T001W.
```

* MODIFY FOR updating Existing Record.

```
WA_T001W - WERKS = 'HYD4'.
WA_T001W - NAME1 = 'HYDERABAD P4'.
MODIFY T001W FROM TABLE I_T001W.
```

* DELETE THE RECORD

```
Delete T001W FROM Table I_T001W.
```

** → difference b/w Insert / del / modify

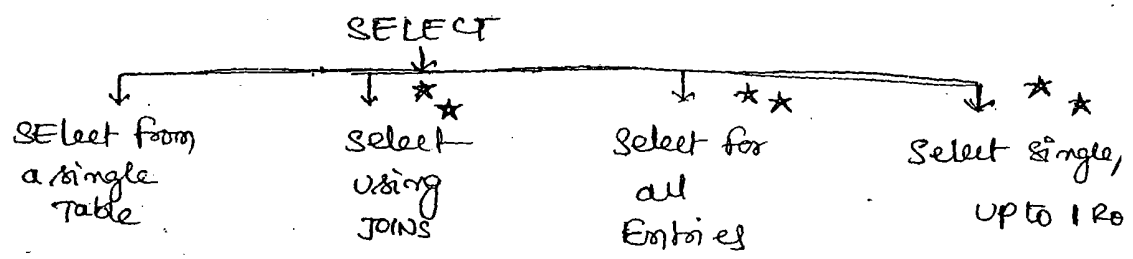
→ Accepting duplicate records ?

→ SY-SUBRC, SY-DBCNT ?

↓ ANS

* which holds the succesful Records processed.

SELECT STATEMENT



SELECT USING JOINS

- This statement is used to read data simultaneously from multiple Tables.
- There are two types of JOINS
  1. Inner Join
  2. Outer Join (or) Left Outer Join

INNER JOINS (95% use in Real time)

- ONLY the matching Records b/w the two Tables will be selected using Inner Joins
- matching or Common Records.

OUTER JOIN

- All the Records from left table are displayed
- If there is any matching Records in Right Table it is displayed.
- If there is no matching record it is displayed as Blank.

```
SELECT  <T1> ~ <F1>
        <T1> ~ <F2>
        <T2> ~ <F1>
        <T2> ~ <F2>
        INTO TABLE <ITAB>
        FROM <T1> AS <Alternative Name> Inner Join/Left Outer Join
             <T2> AS <Alternative Name>
        ON <T1> ~ <F1> = <T2> ~ <F1>
        where <CONDITION>
```

SELECT WITH JOINS

★ Business Requirements

Develop a material master Report which displays the following details

MAterial NO, mat type, Ind-Sector, Units, Leng, descr

Table Name : MARA, MAKT

Field Name : MATNR, MTART, MBRSH, MEINS, SPRAS, MAKTX

ASSIGNMENT

- Develop a Report which selects data from KNA1 & KNBK (XD01)
- Develop a Report which selects data from LFA1 & LFBK (XK01)
- Develop a Report which select data from VBAK & VBAP (VA01)
  EKKO & EKPO (ME21-N)

TYPES : Begin of TY_MARA_MAKT,
MATNR TYPE MARA-MATNR
:
End of- TY_MARA_MAKT

DATA : I_MARA_MAKT TYPE TABLE OF TY_MARA_MAKT.
DATA : WA

Select MARA~MATNR

Into Table I_MARA_MAKT
From MARA AS MARA INNER JOIN
MAKT AS MAKT
ON MARA~MATNR = MAKT~MATNR.

Loop at

"SELECT FOR ALL ENTRIES"

- This statement is used to avoid using INNER JOINS or Nested Select statements

★★ = NOTE =

- Select with Joins is the Best For Two Tables only
- If we are using more than Two Tables Then we should use SELECT FOR ALL ENTRIES

Disadvantages of Select with JOINS

- Select with Joins for more than Two Tables will put Heavy Load on the data base, as it has to take each record from 1st Table and Compare for the matching record in the Second or 3rd Table.
- Because, Comparision is happenning on more than Two Tables.

  It will put heavy Load or it will take a long time for Execution.

★★. So Always ↓ USE For all Entries.

SYNTAX FOR Select for all Entries

```
SELECT <F1>
       <F2>
       <F3>
        FROM <DBTABLE1>
        into Table <ITAB1>
If <ITAB1> is not initial
 Select * from <DBTab62>
  into Table <ITAB2>
     for all Entries in <ITAB1>
        where <F1> = <ITAB1-F1>
        and <F2> = <ITAB2-F2>
```

Example

```
SELECT matnr
       mtart
       mbrsh
       from MARA
       into Table I_mara.
If I_mara is not initial
  SELECT matnr
         spras
         maktx
         from makt
         into Table i_maKT
         for all Entries in I_mara
         where matnr = I_mara-matnr.
ENDIF.
```

PRE REQUISITES FOR Select ↑FOR ALL ENTRIES

- Before Using for all Entries, we need to check wether The First Internal Table or base Table is Empty or not.
- Suppose if base Table is not Empty i.e, Not initial, The matching Records will be fetch between The two internal Tables and The select statement is Executed Perfectly.
- Suppose if The base_Table is Empty or initial, The matching Records will not be fetch because The Select Statement is failed as There are No Records in The base Table.
- Due to This The Select Statement will fetch all The records From The database Table irrespective of matching Condition.

Example

```
REPORT  ZFOR_ALL_ENT.

TYPES : BEGIN OF TY_MARA,
          matnr  TYPE  mara-matnr,
          mtart  TYPE  mara-mtart,
          mbrsh  TYPE  mara-mbrsh,
          meins  TYPE  mara-meins,
        END OF TY_MARA.

TYPES : BEGIN OF TY_MAKT,
          MATNR  TYPE  makt-matnr,
          SPRAS  TYPE  makt-spras,
          maktx  TYPE  makt-maktx,
        END OF TY_MAKT.

TYPES : BEGIN OF TY_FINAL
          matnr  TYPE  mara-matnr,
          MTART  TYPE  mara-mtart,
          MBRSH  TYPE  mara-mbrsh,
          MEINS  TYPE  mara-MEINS,
          SPRAS  TYPE  MAKT-SPRAS,
          MAKTX  TYPE  MAKT-MAKTX,
        END OF TY_FINAL.

DATA : i_mara  TYPE TABLE OF TY_MARA.
DATA : i_MAKT  TYPE TABLE OF TY_MAKT.
DATA : i_FINAL TYPE TABLE OF TY_FINAL.

DATA : wA_MARA TYPE TY_MARA.
DATA : wA_MAKT TYPE TY_MAKT.
DATA : wA_FINAL TYPE TY_FINAL.

* BREAK POINT.
* TYPE DECLARATIONS DECLARED
*** SELECT FOR ALL ENTRIES
```

```
SELECT  MATNR
        MTART
        MBRSH
        MEINS
        FROM MARA
        INTO TABLE i_MARA.
IF i_MARA IS NOT INITIAL.
  SELECT  MATNR
          SPRAS
          MAKTX
          FROM MAKT
          INTO TABLE i_MAKT
          FOR ALL ENTRIES IN i_MARA
          WHERE MATNR = i_MARA-MATNR.
  ENDIF.
 SKIP.
 SKIP.
SORT i_MARA BY MATNR.
SORT i_MAKT BY MATNR.
LOOP AT i_MAKT INTO WA_MAKT
  READ TABLE i_MARA INTO WA_MARA
       WITH KEY MATNR = WA_MAKT-MATNR
         BINARY SEARCH.
  WA_FINAL-MATNR = WA_MARA-MATNR.
  WA_FINAL-MTART = WA_MARA-MTART.
  WA_FINAL-MBRSH = WA_MARA-MBRSH.
  WA_FINAL-MEINS = WA_MARA-MEINS.
  WA_FINAL-SPRAS = WA_MAKT-SPRAS.
  WA_FINAL-MAKTX = WA_MAKT-MAKTX.
  APPEND WA_FINAL TO i_FINAL.
ENDLOOP.
LOOP AT i_FINAL INTO WA_FINAL.
    WRITE: / WA_FINAL-MATNR,
             WA_FINAL-MTART,
             WA_FINAL-MBRSH,
             WA_FINAL-MEINS,
             WA_FINAL-SPRAS,
             WA_FINAL-MAKTX.
ENDLOOP.
```

★ SELECT SINGLE / UPTO 1 ROWS :

SELECT { Single / upto 1 R

- Both are Used to Read a single Record
- Both statements are used for Field Validation at REPC level
- The difference is,

SELECT SINGLE	SELECT ... UPTO 1 ROWS
1. Used to Read Exact record	1. Used to Read appropriate Record.
2. To Read Exact Record we need to specify all KEY fields	2. TO Read appropriate Record There is no need to specify Key fields.
3. Used for field validation when we have all the Key fields.	3. Used for field validation when we don't have all Key fields.
SELECT SINGLE * FROM <DBTAB> INTO <WA> where <specify All Key fields>	SELECT * FROM <DBTAB> INTO <WA> UPTO 1 ROWS WHERE <specify Part of KF> END SELECT.

Example on Select SINGLE

```
DATA : WA_KNA1 TYPE KNA.
Parameters : P_KUNNR TYPE KNA1-KUNNR.
             Select single * from KNA1
             into WA_KNA1
             where KUNNR = P_KUNNR
If SY-SUBRC <> 0
Message 'Please Enter a valid customer', TYPE 'E'.
```

Example on Select upto 1 Rows

Real time Example ⇒ CEPC / PRCT

```
Parameters : P_PRCTR TYPE CEPC-PRCTR.
Select * from CEPC
        INTO wa_cepc
         UP TO 1 Rows
        where prctr = P_prctr.
ENDSELECT.
IF SY-SUBRC <> 0
  message 'please Enter a valid profit cntl', TYPE 'E'.
ENDIF.
```

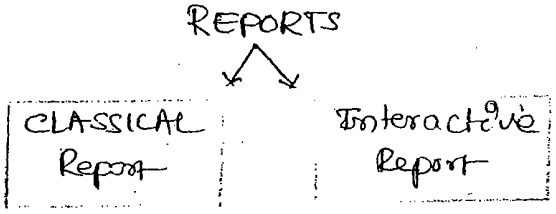
NOTE

* Here we are using up to 1 Rows Because the Table CEPC contains 3 Key fields, But our programming is Having only one field. i.e, PRCTR
* Since we don't have all the Key fields, we are using up to one Rows.

"REPORTS"

* It is an Executable program which is used to read the data from data base tables and display the data in a well defined format as per the business requirements are called Reports.

- There are Two different types of Reports available,



CLASSICAL REPORT

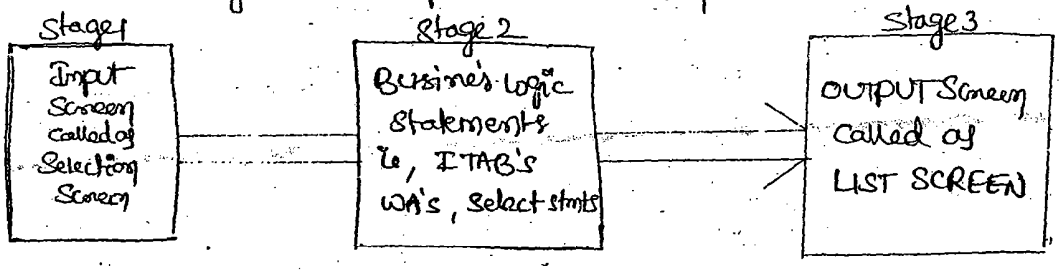
- Displaying the Entire information on a single Screen is called a classical report

INTERACTIVE REPORT

- Displaying the information in more than one output Screen is called an interactive Report.

Main functions or Stages of reports

- There are 3 Stages for Report Execution



SELECTION SCREEN

- The Input Screen is called as Selection Screen
- Below are the selection Screen Commands.

Selection Screen Command	DESCRIPTION
PARAMETERS	used to display a single Input field
SELECT-OPTIONS	Used to display a range for all field
Selection Screen formatting Statements	used to decorate/format Selection Screen

Parameters

- This statement is used to display a single input field, check boxes, Radio Button's on the Selection Screen.

Syn

```
Parameters : <P_Name> (length) TYPE <Datatype>
```

Ex:

```
Parameters : P_NAME (25) TYPE C.
Parameters : P_KUNNR TYPE KNA1-KUNNR.
Parameters : P_LAND1 TYPE KNA1-LAND1 default 'DE'.
Parameters : P_NAME1 TYPE KNA1-NAME1 obligatory.
```

OBLIGATORY is the statements which is used to make a field mandatory on Selection Screen

DISPLAYING CHECK BOX ON SELECTION SCREEN

- Below is the syntax used for displaying check Box.

Syn

```
Parameters : <check Box Name> AS checkBox
```

Ex

```
Parameters : P_CH1 AS checkbox
Parameters : P_CH2 AS checkbox
```

- Suppose if the check Box is selected the value 'X' is stored in the parameter variable.
- Based up on this parameter variable we write ABAP code.

```
If P_DOWNLOAD = 'X'
  . . .
  . . .
else if P_DOWNLOAD = ' '
  . . .
  . . .
ENDIF.
```

Example program

```
Parameters : P_CH1 AS CheckBox.
If P_CH1 = 'x',
    write : / 'check Box is selected'.
ElseIf P_CH1 = ' '.
    write : / 'check Box is not selected'
ENDIF.
```

Displaying RADIO Buttons on Selection Screen

- Radio Buttons are always displayed in the form of group so that we can select a single RADIO Button

Syn:

```
Parameters : <RB NAME> Relationship <Group Name>
```

Ex:

```
parameters : P_RB1 Radiobutton group ABC,
             P_RB2 Radiobutton group ABC. (or)
             P_RB2 Radiobutton group ABC default 'x'.
If P_RB1 = 'x'.
    write : / 'RB1 is selected'.
ELSEIF P_RB2 = 'x'.
    write : / 'RB2 is selected'.
ENDIF.
```

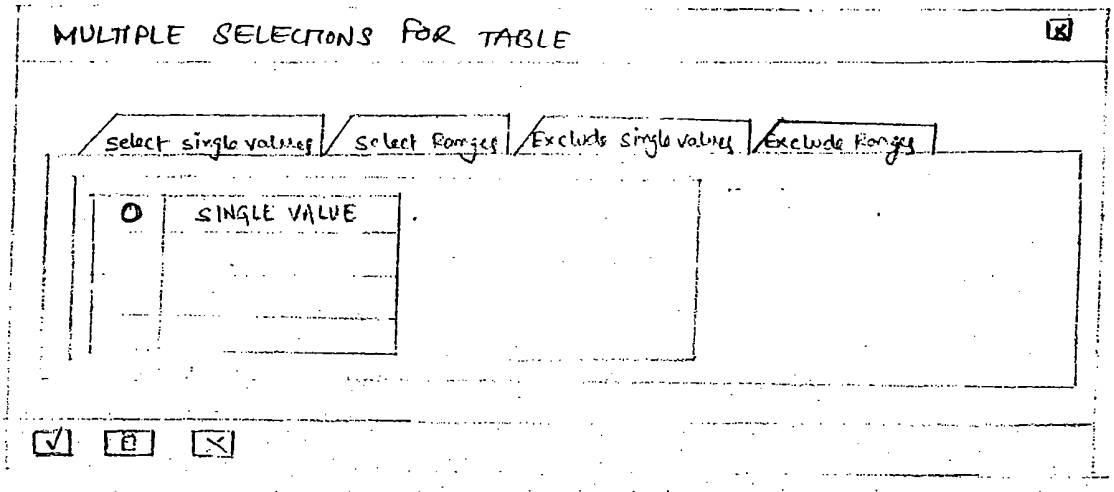
If we write this RB2 is selected defaultly

SELECT OPTIONS

* This statement is used to display a range, ie, From value, TO value for an Input field.
* By default an internal table is declared for SELECT OPTIONS with the fields SIGN, OPTIONS, LOW, AND, HIGH, FEEDS
* By using Select options we can define Not only the Range But also,

1. we can define multiple single values (10, 17, 20, 23, 90, 99)
2. we can define multiple single Ranges (10-15, 17-20, 25-29, 50-69)
3. we can Exclude multiple single values
4. we can Exclude multiple single Ranges

- All the above options are visible only when you click on Extension Box or multiple selection Box



Syn

Select-option : <SO_NAME> for <Table-FIELDNAME>

Ex :

Tables : KNA1

Select-options : S_KUNNR for KNA1-KUNNR.

- The Select statement for the Select options will be of below.

Select * From KNA1
into I_KNA1
where KUNNR IN S_KUNNR.

Example

Table : KNA1

Data : I_KNA1 TYPE TABLE OF KNA1

Data : WA_KNA1 TYPE KNA1

SELECT-OPTIONS : S_KUNNR FOR KNA1-KUNNR.

BREAK-POINT.

```
Select * from KNA1
       INTO Table I_KNA1
       where KUNNR IN S_KUNNR.
Loop at I_KNA1 into WA_KNA1.
   write:/ WA_KNA1-KUNNR.
ENDLoop.
```

SELECT OPTIONS INTERNAL TABLE

- When ever we use Select options on internal table wil b Created with the below fields,

SIGN - Stores either I or E
- I - Inclusive or Includes
- E - Exclusive or Excludes

OPTION - Stores the option like BT (Between)
- NB (Not Between)
- LT (Less Time)
- LE (Less Than or Equal to)
- GT (Greater Than)
- GE (Greater Than or Equal to
- NB (Not between)

Low - Stores Low value

HIGH - Stores High value

NO EXTENSION

- If you want to hide the Extension box Then use the below Syntax

Syn:

SELECT-OPTIONS : <SO-NAME> for <Table-FieldName> NO-	Extension
SELECT-OPTIONS : " "	No Interval
SELECT-OPTIONS : " "	No Extension No Interval

⁂ what is the difference between a parameter and select option with NO Interval and No Extension?

⁂ what happens if only low value is given in select option?

⁂ what happens if only high value is given in SELECT-OPTIONS?

VARIANT

- It is an object which is used to store Selection Screen field values, for future reference
- Generally we create variants for the Test Cases.
- Create a program with some parameter statements as Below.

```
Parameters : P_LAND1 TYPE KNA1-LAND1.
Parameters : P_MTART TYPE MARA-MTART.
```

- click on Execute
  - Enter the values as Below

    P_LAND1 [US]

    P_MTART [FERT]

  - click on Save Button
  - Give the Name, meaning as TEST1
  - This TEST1 is called a variant.
- Similarly Create variant 2 ie, TEST2
- An Icon will be visible on the Selection Screen to display List of variants.
- The Icon is [📋] (GET VARIANT)

Selection Screen Text

- By default, whenever we create a Selection Screen all the fields in the Selection Screen will be displayed with Technical Names.
- To avoid this, we need to maintain Selection Screen Text

STEP①: open SE38 program

②: click on Go To
↳ Text Elements
↳ Selection Texts

③ In this we can provide our own Text or we can copy the Text from data dictionary.

Name	Text	data dictionary
P_LAND1	Country	☐
P_MTART	material type	☑

④ Save, Activate and Test the program.

SELECTION SCREEN COMMANDS TO FORMAT SELECTION SCREEN:

1. Selection Screen ULINE
   - Used to display horizontal line on Selection Screen.
2. Selection Screen SKIP
   - Used to display blank line on Selection Screen.

★★
★★3. Selection Screen Block
   - used to display group of Selection Screen fields in a Block

SYNTAX

```
Selection Screen Begin of Block <Block Name> with Frame title
                                                <Text-001>
  :
Selection Screen End of Block <Block Name>.
```

Example (99% we use this in Real time) double click it and create

Selection-Screen Begin of Block B1 with frame Title Text-001.

Parameters: P_LAND1 TYPE KNA1-LAND1.

Parameters: P_MTART TYPE MARA-MTART.

Selection-Screen End of Block B2.

4. Selection Screen Begin of Line
   - This statement is used to display all the selection screen fields in a single line.

★ NOTE
- By default Every selection screen field is displayed in a seperate line.

Syntax

```
Selection Screen Begin of Line
    :
Selection Screen End of Line
```

5. Selection Screen COMMENT.
   - This statement is used to provide Custom label for a selection screen field.

SYNTAX:

```
Selection-Screen Comment <pos> (width) <Text Symbol>
```

Example:

Selection-Screen Begin of Line.

Selection-Screen COMMENT 2(10) Text-003.

Parameters: P_LAND1 TYPE KNA1-LAND1.

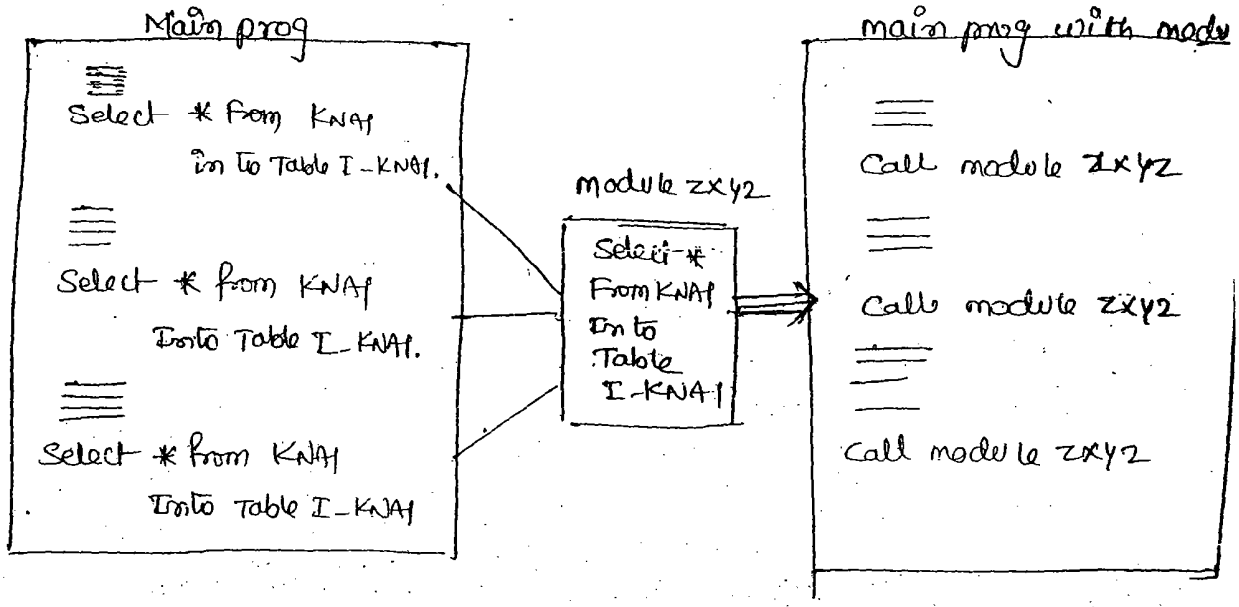
Selection-Screen comment 20(10) Text-004.

Parameters: P_MTART TYPE MARA-MTART.

Selection-SCreen End of Line.

"MODULARIZATION"

- It is a Technique of dividing a Main program into Sub programs for better readability and Re-usability is called modularization.



- modularization is achieved using the following Techniques

1. Include program's
2. Function modules
3. SubRoutines
4. MACRO's (Used in HR-ABAP)
5. Class methods (Used in OO-ABAP)

★ INCLUDE PROGRAMS

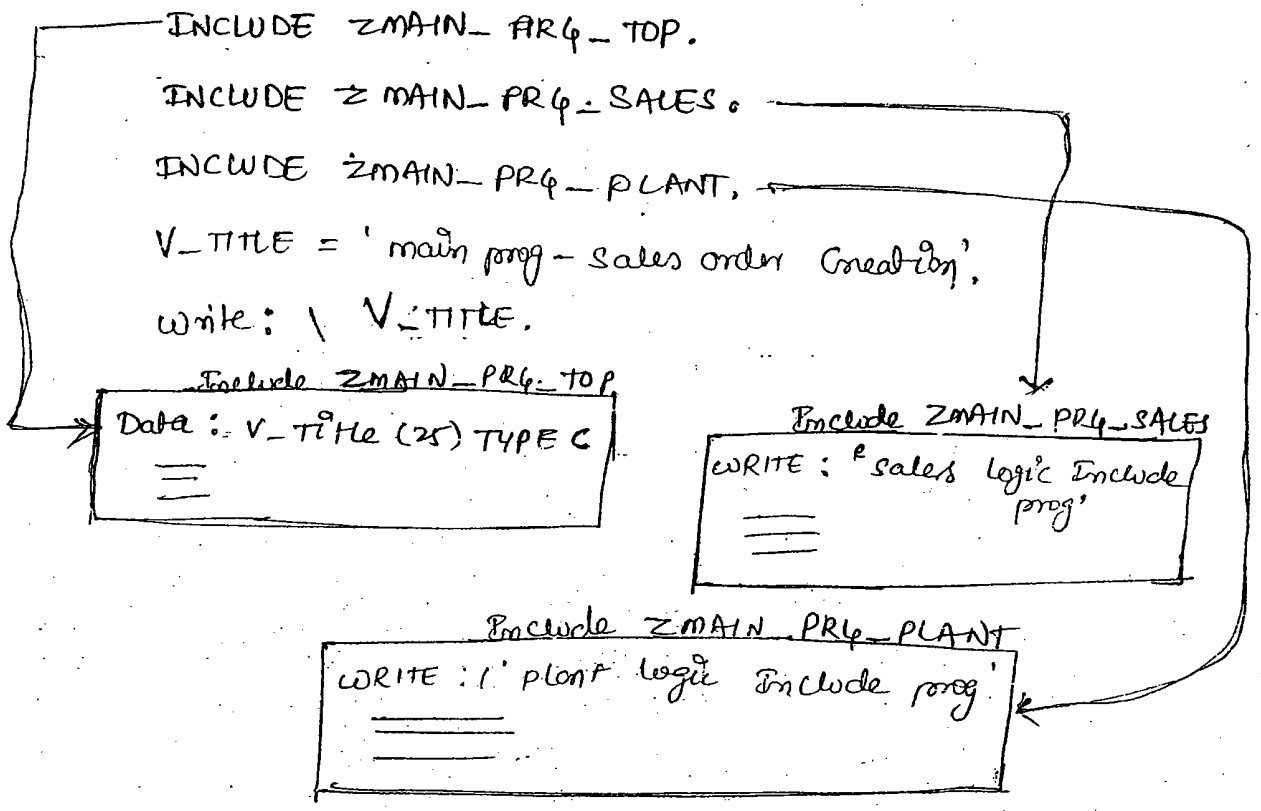
- These programs contains a set of Re-usable statements.
- These can not be Executed independently.
- These must be Inserted in the main program for Execution
- These programs are called as TYPE 'I' programs.

★★ It does not contain any parameter Interface
i.e, No Importing, Exporting parameters.

Example

** SAPMV45A is a standard program for SALE order which contains No of Include programs.

Example,



FUNCTION MODULES

- It is a repository objects which contains ABAP logic or statements for doing a specific Task or functionality.
- function modules contains PARAMETER Interface, ie, Import, Export, changing, Tables Etc
- These are mainly used for Global modularization
- function modules can be REMOTE Enable.
  ie, They can access Remote Server's to fetch the data.
- function modules Contains Exceptions to Catch Certain type of Errors.

FUNCTION GROUP

- It is a group or containes of function modules
- Every function module must be Inserted in a function gr

FUNCTION MODULE PARAMETER INTERFACE (SE37)/ SE80

IMPORT :

- Input variables To function modules are called Importing parameters.

EXPORT :

- Output variables From function module are called as Exporting parameters.

CHANGING :

- Variables which act's as Importing and Exporting are called as changing parameters.

TABLES :

- Internal Tables which act's as Importing and Exporting are called as Tables parameter's.

EXCEPTIONS :

- These are used to Catch Certain types of Errors.
- Exceptions are written in SY-SUBRC Variable

SOURCE CODE

- It Consists of ABAP code for doing a specific Task.

STEPS TO CREATE A FUNCTION GROUP :

We Can Create function group either in SE37 / SE80.

- GO TO SE 80
- Select Function group
- Give The Name as ZFUNGRP
- PRESS ENTER

- click on YES
- provide some short text
- click on Save
- Activate the function module Immediately.

Steps to Create a Function module

★ Business Requirement

develop a function module which displays customer details

- Go to SE37
- Give the Name as ZCUST_GET_DET
- Click on Create
- Give function group, description, click on Save
- Provide the Importing parameter as Below.

  IM_KUNNR — TYPE — KNA1-KUNNR
- provide the Exporting parameter as Below

  EX_KNA1 — TYPE — KNA1
- Provide the Exceptions As Below

  NO_KUNNR — NO_KUNNR

  INVALID-KUNNR — INVALID-KUNNR
- write the Below Source Code

```
If IM_KUNNR is INITIAL.
   RAISE NO KUNNR.
else
select SINGLE * from KNA1
    into EX_KNA1
    where KUNNR = IM_KUNNR.
If SY-SUBRC <> 0.
   RAISE INVALID-KUNNR.
  ENDIF.
ENDIF.
```

- Save, Activate, Test it Immediately.

USING FUNCTION MODULE IN PROGRAM

```
DATA : WA_KNA1 TYPE KNA1.
Parameters : P_KUNNR TYPE KNA1-KUNNR.
CALL FUNCTION 'ZCUST_GET_DET'
  EXPORTING
    IM_KUNNR = P_KUNNR
  IMPORTING
    ER_KNA1 = WA_KNA1
  EXCEPTIONS
    NO_KUNNR = 1
    INVALID_KUNNR = 2
    OTHERS = 3
If SY-SUBRC = 1.
  MESSAGE 'please Enter Customer NO', TYPE 'I'.
else if SY-SUBRC = 2.
  MESSAGE 'Please Enter VALID CUSTNO', TYPE 'I'.
else
  message 'UNPREDICTABLE ERROR OCCURED', TYPE 'I'.
Endif.
WRITE : / WA_KNA1-KUNNR,
          WA_KNA1-NAME1.
```

ASSIGNMENT

- Develop a function module which displays SALE ORDER details. VBAK -VBELN
- Develop a function module which displays PURCHASE ORDER details. EKKO -EBELN

USING TABLES IN FUNCTION MODULES:

- Develop a function module which displays list of the customers for a country

① Create function module by name ZCUST_GET_LIST

② Define the Importing parameter as Below,

IM_LANDI TYPE KNAI-LANDI

③ Define the Tables parameters as Below

EX_KNAI LIKE KNAI

④ write the Below Source Code,

```
Select * FROM KNAI
        INTO TABLE EX_KNAI
        WHERE LANDI = IM_LANDI.
```

⑤ Save, Activate and Test it.

⑥ use the above function module in a program.

USING TABLE TYPES IN FUNCTION MODULES

- Tables are obselete in SAP
  Ie, we should not use Tables parameter in function modules.
- Instead, we should Create Table types in data dictionary

Steps to Create Table Types

- Go to SE 11
- Give the data type as ZKNA1_T
- click on Create
- Select Table TYPE
- Give description as Table Type for KNA1
- Give the Line type as KNA1
- Save and activate

Go to SE 37

- Create a function module by Name YCUST_GET_LIST
- provide the Importing parameter as Below.

IM_LAND1 TYPE KNA1-LAND1

- provide the Exporting parameter as Below.

EX_KNA1 TYPE ZKNA1_T

- WRITE THE SOURCE CODE
- Save activate and Test it.

'SUB ROUTINES'

⊙ Sub routines Contains piece of ABAP Code which one mainly used for readability and Re-usability.

⊙ These one also Used for Local modularization.

SYNTAX

The Syntax Contains Two steps

1. Sub routine definition
2. Sub routine Implementation.

Syntax for Subroutine definition

```
Perform <Subroutine> Using <V1> <V2>
                     Changing <CH1> <CH2>
                     Tables <IT1> <IT2>
```

} ACTUAL PARAMETERS

SYNTAX FOR SUB ROUTINE IMPLEMENTATION

```
FORM <Subroutine>  USING     <V1> <V2>
                   CHANGING  <CH1> <CH2>
                   TABLES    <IT1> <IT2>   } Formal parameters

- - -
- - -
- -
- - -

END FORM.
```

TYPES OF SUB ROUTINES

- There are Two types of Sub routines

1. Local Sub routines
2. GLOBAL Sub routines

LOCAL SUB ROUTINES

- If Sub routine definition and implementation
  i.e, PERFORM And FORM .... ENDFORM, are available in Same program Then it is called as Local Sub routines

GLOBAL SUB ROUTINES

- If Sub routine definition is available in one program and Sub routine Implementation is available in another program Then it is called as Global Sub routines
  (OR) External Sub routines.

Syntax:

Available in ONE PROGRAM:

```
PERFORM <SUB ROUTINE NAME> in program <program Name>
                           USING <V1> <V2>
                           . . .
                           . . .
```

Available in another program of TYPE SUBROUTINE pool:

```
FORM <SUB ROUTINE NAME> USING <V1> <V2>
. . .
. . .
. . .
END FORM.
```

PARAMETER INTERFACE OF SUBROUTINES

USING : Importing variables to Sub routines are defined with USING

CHANGING : Exporting variables are defined with CHANGING

TABLES : Internal Tables for importing and Exporting are defin with TABLES

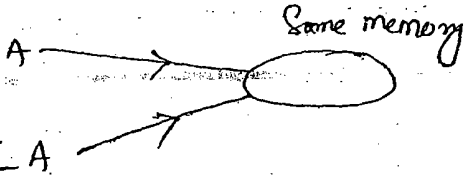
TYPES OF PASSING VALUES

★★
★ • There are 3 types to send the values from Subroutine definition to sub routine implementation

1. pass By value (or) Call by value
2. pass By Reference
3. pass By value and Return.

PASS BY REFERENCE

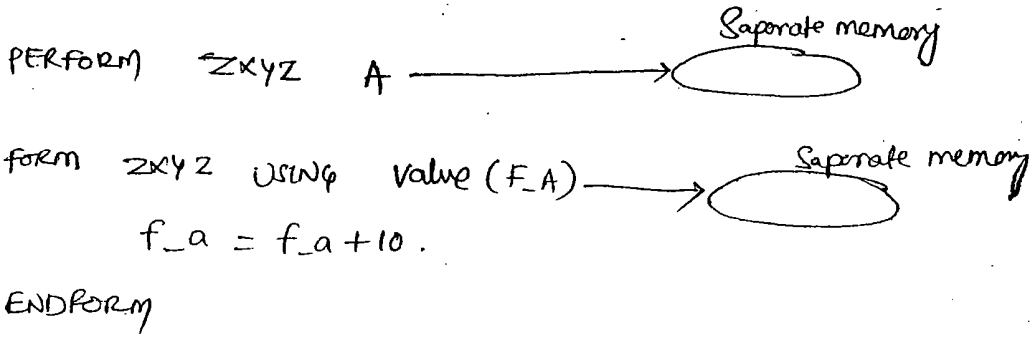
- In this type, the actual and formal parameter will be refering to the same memory location
- If any changes are made to the formal parameter's the actual parameters will also be effected.




```
PERFORM ZXYZ USING A
FORM ZXYZ USING F_A
  f_a = f_a + 10.
ENDFORM.
```

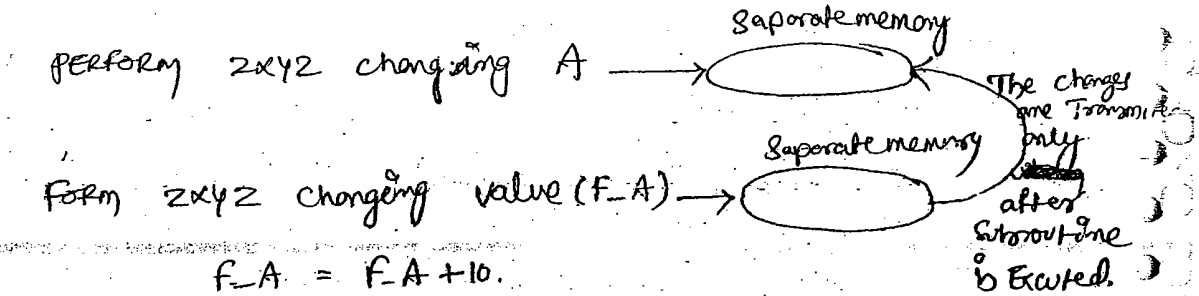
PASS BY VALUE

- In this type the actual and formal parameters will be refering to Separate Memory locations.
- If any changes are made to the formal parameters it will not effect the actual parameters.



PASS BY VALUE AND RETURN

- In this type the actual and formal parameters will be refering to Saparate memory locations.
- If any changes are made to the formal parameters, the actual parameters will be changed only when the last statement in the Subroutine Implementation (or) the Entire Subroutine is Executed.



EXAMPLE ON SUBROUTINES"

```
Data : A TYPE I.
       A = 10.
WRITE : / 'Before Subroutine call', A
PERFORM pass_by_Ref Using A
PERFORM pass_By_value Using A
PERFORM pass_By_value_Return changing A
WRITE : / 'After Subroutine call', A

FORM PASS_BY_REF using F_A.
     F_A = F_A + 10
END FORM.

FORM PASS_BY_VALUE Using value(F_A)
     F_A = F_A + 10
ENDFORM.

form PASS By_value return changing value(F_A)
BREAK POINT
SKIP.
SKIP.
f_A = f_A + 10
SKIP.
SKIP
END FORM.
```

REALTIME EXAMPLE ON SUB ROUTINES

```
REPORT ZSUBROUTINES.

Data : WA_MARA TYPE MARA.

parameters : P_MATNR1 TYPE mara-matnr.
parameters : P_matnr2 TYPE mara-matnr.
parameters : p_matnr3 TYPE mara-matnr.

PERFORM get_data_FROM_MARA USing P_MATNR1.
  WRITE : / WA_MARA-MATNR,
            WA_MARA-MTART,
            WA_MARA-MEINS,
PERFORM get_data_from_MARA Using P_MATNR2
  WRITE : /
PERFORM get_data_from_MARA Using P_MATNR3
  WRITE : /

form get_data_from_mara Using F_MATNR.
  SELECT SINGLE *FROM MARA
    INTO WA_MARA
    where MATNR = F_MATNR.
  ENDFORM.
```

"MESSAGES"

- All the messages are stored in a class called as Message class
- SE91 is the TCODE for creating message class.
- Generally there will be a single message class for Entire project

MESSAGE-ID

- It is a Key word which is used to Assign the message class to our Report so that, our report can Access all the messages in the Message class.

TYPES OF MESSAGES

TYPE	DESCRIPTION
E	Error message
S	Screen
W	warning
I	Information
A	Abort

SUBSTITUTIONAL PARAMETERS

- '&' is the Symbol use to substitute a variable or parameter in to the Message
- we can substitute a maximum of 4 substitutional parameters.

SYNTAX

1. MESSAGE <MSG TXT> TYPE <MSG TYPE>
2. MESSAGE <MSG TYPE> <MSG NO>
3. MESSAGE <MSG TYPE> <MSG NO> WITH <SV1>
   <SV2>
   <SV3>...

STEPS To Create a message class

- Go TO SE91
- Give The Name as ZSONY_MSG_CLASS
- click on Create
- Give description
- click on messages TAB
- define the messages as Below

message	message short text
000	material does not Exist
001	material & does not Exist (we generally use this Real time)
003	& & & &

- click on Save.

Using Message class in a REPORT

```
REPORT ZSUBROUTINES MESSAGE-ID ZSONY_MESSAGECLASS.
data: wa_mara TYPE mara.
Parameters: P_matnr1 TYPE MARA-MATNR.
    Select single * FROM MARA
        INTO WA_MARA
        where MATNR = P_MATNR1.
If SY-SUBRC = 0.
    WRITE :/ WA_MARA-MATNR,
             WA-MARA-MEINS.
ELSE.
* MESSAGE 'MATERIAL DOES NOT EXIST' TYPE 'E'.
* MESSAGE I000.
* MESSAGE I001 with P_MATNR1.
MESSAGE I002 with 'material' 'DOES' 'NOT' 'EXIST'.
ENDIF.
```

"EVENTS"

- ABAP is called as Event driven programming language Because, The ABAP PROGram is Executed Based on Even Not line by line.
- C language is called as structured programming langua where The Execution will be LINE BY LINE.

LIST OF THE EVENTS

1. Load of progrom
2. Initialization
3. AT Selection Screen output
4. AT Selection Screen on field
5. AT Selection Screen on value-Required
6. AT Selection Screen on Help Required
7. AT Selection Screen

} Selection Screen Event

8. START of Selection
9. End of Selection
10. TOP-of-page
11. END of page

} List Events.

Load of program

; This Event is used to load the program in to the memory for Execution

INITIALIZATION

- This Event is used to initialize default values to Variables and Selection Screen Variables.
- This Event is Triggered Before Selection Screen is displayed

AT SELECTION SCREEN OUTPUT

- It is used to modify selection screen dynamically based on user Actions.
- It is Triggered Before selection Screen is displayed and for Every action MADE on The Selection Screen by the User.

AT SELECTION SCREEN ON FIELD

- It is used to validate a single field on Selection Screen
- It is Triggered after making an action on Selection Screen

At Selection Screen on Value Request

- It is used for displaying Search help values
- It is Triggered when The User click's on Search help button

At Selection Screen on Help Request

- It is used to display Help Information for a field.
- It is Triggered when The User click's on F1 Button.

At Selection Screen

- It is used for validating multiple Selection Screen fields and also used for Containing all The above Events under ONE Block.

START OF SELECTION

- It is used for writing or Fetching The data from The database i.e, All our select Statements must be defined under This Event.

★★★ This is The default Event which is Triggered automatically

END OF SELECTION

- This Event is used to Recognize All The Select Statements have been fetched The data from data base
- ★ It does not have Any Importance in ABAP
- It is mainly used when we work with logical data base
- ★ Logical data bases are Not used in ABAP, But they are used in HR ABAP
- ✓ I didn't work on HR ABAP

TOP OF PAGE

- It is used to display constant Heading for all The pages
- ★ It is Triggered after The first WRITE statement (or) Any output statement like SKIP, SY-ULINE, SY-VLINE

END OF PAGE

- It is used to display Constant footer information for all The pages.

NOTE

We Need to Set LINE-COUNT TO display End of page

Example

```
REPORT <Report_NAME> LINE-COUNT TotalLines(FOOTERLINES)

Report ZKYZ LINE-COUNT 37(3)
                       ↓
                  Reserved for Footer Info
```

SALE ORDER TABLES

VBAK - Header Table
VBAP - Item details

✓ File Name → RLGRAP FILENAME

EKKO - Purchase order Header
EKPO -

DIFFERENCE BETWEEN INCLUDE prog/FUNmodule/ subroutine

INCLUDE	FUNMODULE	SUBROUTINE
Can't not be Executed	Can be Executed	Can not be Executed
No parameter Interface	parameter Interface	parameter Interface
No Exception	Exception	No Exception
No RFC	RFC	NO RFC

Example on AT Selection Screen output

- SCREEN
  - It is a structure in data dictionary which holds the properties of Each Selection Screen field.
  - The properties are listed below,

PROPERTY	DESCRIPTION
NAME	Name of the Selection field
GROUP1	modification ID of Selection Screen field
INPUT	Input property { 0 - mean's Input disabled, 1 - mean's Input Enabled
ACTIVE	Active property { 0 - mean's field is Invisible, 1 - mean's Field is visible

- MODIFICATION ID
  - It is an ID which is assign to a Selection Screen field or group of Selection Screen field to modify their properties dynamically.
- USER COMMAND
  - It is a Key word which is used to identify that some action is raise on Selection Screen.
  - Based up on the User Command at Selection Screen output is Triggered.

Business Requirement

* Create a REPORT with Selection Screen fields LIKE
  CNO, CNAME, VNO, VNAME

Create 2 Radio Buttons on Selection Screen to display Either Customer Information or vendor Information based on Radio button Selection.

- If Customer Radio Button is selected, Vendor Number, Vendor Name should be disabled.
- If Vendor Radio Button is Selected Customer Number, Customer Name should be deleted.

```
Parameters : P_KUNNR TYPE KNA1-KUNNR MODIF ID ABC.
Parameters : P_CNAME TYPE KNA1-NAME1 MODIF ID ABC.
Parameters : P_LIFNR TYPE LFA1-LIFNR MODIF ID DEF.
Parameters : P_VNAME TYPE LFA1-NAME1 MODIF ID DEF.

Parameters : RB_CUST Radiobutton Group AAA DEFAULT 'X'
             RB_VEND Radiobutton Group AAA.
                                        USER-COMMAND UCOM

AT Selection Screen Output.
  If RB_CUST = 'X'.
    Loop at SCREEN.
      If SCREEN-GROUP1 = 'ABC'.
        SCREEN-INPUT = '1'.
        MODIFY SCREEN.
      ELSEIF SCREEN-GROUP1 = 'DEF'.
        SCREEN-INPUT = '0'.
        MODIFY SCREEN.
      ENDIF.
    ENDLOOP.
```

```
ELSEIF RB_VEND = 'X'
    Loop at screen.
      If screen-group1 = 'ABC'.
        screen-input = '0'.
        modify screen.
      Elseif screen-group1 = 'DEF'.
        screen-input = '1'
        modify screen.
      ENDIF.
    ENDLOOP.
ENDIF.
```

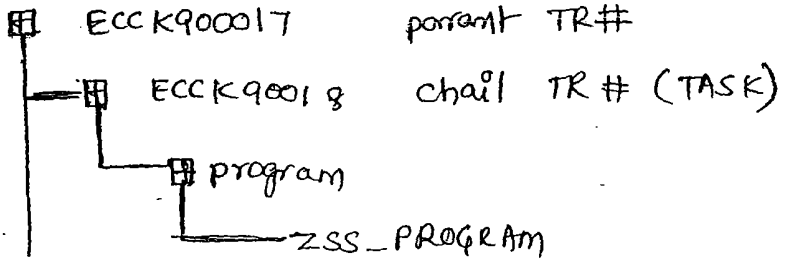
CREATING TCODE FOR REPORTS : (SE93)

- Go to SE93
- Give the Transaction Code Name as ZSS
- click on Create
- Give the description
- Select Second Radio button for Report Transaction
- Press ENTER
- Give the program Name -
- ** Give the Selection Screen No (By default it is 1000)
- Select all the check Boxes (GUI)
- click on Save
- Test the Transaction by Entering the TCode in the Command bar.

RELEASING Transport Request Number : (SE09)

- Go to SE09
- Select the check box modifiable
- click on display.

- Select the Transport Request No and Expand it



- put the cursor on chaild TR# (TASK) and click on the TRUCK Symbol.
- Again put the cursor on the parent TR# and click on TRUCK Symbol again
- The TR# is Released from the User Name.

– Send an Mail to BASIS Consultant to Move the TR# from development Server to Quality Server.

CONTROL / BREAK STATEMENTS / EVENTS

- The statements which are used to Break the Control with in Loop.. EndLoop are called as Control Break Statements.
- These statements will work only between Loop... Endloop.

Below are the statements,

AT FIRST ... ENDAT:

– This statement is Executed when Ever the loop Executes the first record of the internal Table. (or)
when Ever the loop Executes the First iteration
ie, SY-INDEX = 1

– This statement is generally used for to display Heading

AT LAST .... ENDAT

- This statement is Executed whenever the loop Executes the last record of internal table. (or)
  when Ever the loop Executes the last iteration.
  ie, SY-INDEX = 100/1000 etc.
- This statement is generally used to calculate grand totals.

AT NEW <FIELDNAME> .... ENDAT

- This statement is Executed when Ever there is a New value on a particular field.
- It is generally used for displaying Sub-Headings.

AT END of <FIELDNAME> ... END AT

- This statement is Executed when Ever the New value Ends on a particular field.
- Generally used for calculating sub totals.

ON CHAGE OF <FIELDNAME>

- It is same as at New
- This statement was used in the older version and it is obsolete in ECC6.0
- So always us AT NEW instead of on change of

SUM

- It is a Keyword which is used to calculate totals and Sub totals on currency and quantity fields with in Loop.. Endloop.

** ISSUE WITH THE CONTROL BREAK STATEMENTS *

- When Ever we are working with Control break statement *** will be available in the work area, when Ever any control break statement Executed.
- To avoid this problem, move the data from the origina work area to Temporary work area for doing Calculatio

— INTERACTIVE REPORTS —

Displaying the Basic information in the first list and Basic list and detailed information in the Secondary list are called as Interactive reports.

★ In Real time max Secondary list are 3 or 4

- There are Totally 21 lists are available
  - The first list is called BASIC LIST (LIST NO = 0)
  - The Remaining list's are called Secondary List (LIST NO = 1, 2 ... 20)
  - The List No is stored in a System variable called SY-LSIND (LIST INDEX)

★ SY-LSIND

It is a System variable which stores the List no.

- There are 4 Events used to Create Interactive Reports
  1. AT LINE SELECTION
  2. USER COMMAND
  3. AT PF
  4. TOP-OF-PAGE DURING LINE SELECTION

Interactive Report Events

1. AT LINE SELECTION

This Event is triggered whenever the User double click's on any list line

2. AT USER COMMAND

- This Event is Triggered when Ever The User click's on custom GUI Buttons.

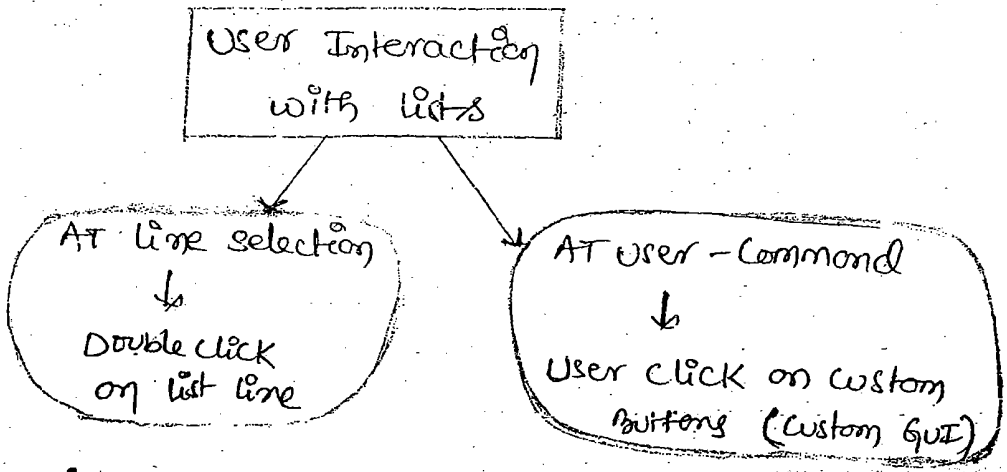
3. AT PF

- This Event is Triggered when Ever The User Hits The Function Key

4. TOP OF PAGE DURING LINE SELECTION

- This Event is used to generate Constent page Heading For all The Secondary List. where as The TOP of Page Event is used to generate page Heading only for The Basic list.

USER INTERACTION




```
WRITE: 'BASIC LIST NO IS', SY-LSIND.
TOP-OF-PAGE.
   WRITE: ' INTERACTIVE Report Basic List Heading.'
AT LINE-SELECTION.
   WRITE: / 'Secondary List NO is', SY-LSIND.
TOP-OP-PAGE DURING LINE-SELECTION.
   WRITE: 'INTERACTIVE REPORT SECONDARY LIST
                Heading'.
```

FINDING THE SELECTED LINE CONTENTS

- There are 3 Techniques to find the line contents on which the user has raised the double click Events.

1. SY-LISEL (we never use this on Real time) ★
2. ★ HIDE
3. GET CURSOR

OFF SET FUNCTIONALITY

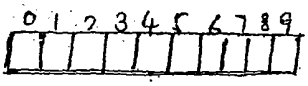
- This functionality is used to read a single character or group of characters from a source variable in to Target variable

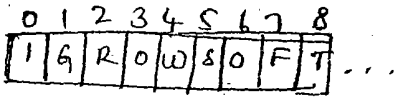
SYNTAX

Target variable = Source Variable + Starting position (width)

Ex:

```
Data : V_NAME1 (15) o TYPE C.
Data : V_NAME2 (10) TYPE C.
Break-point.
V_NAME1 = 'IGROWSOFT'.
V_NAME2 = V_NAME1 + 0(5).
WRITE :/ V_NAME2.
```





Example program on SY-LISEL

- Business Requirement

- Develop a material master Report which displays basic details in the Basic List and description details on the secondary list.

ASSIGNMENT

- Develop a SALE order Reports which display's Header details in the Basic list and Item details in the Secondary list

  Header details one ; VBELN , VKORG, VTWEG, SPART

  Item details one ; VBLN, POSNR, MATNR, NETWR

- Develop a purchase order Report which displays Header details in the Basic list and item details in the Secondary list. EKKO

" Interactive Reports Using Hide Statement "

HIDE :

- It is a Key word which is used to store a variable or work area in to a Temporary memory called as HIDE AREA.

- HIDE AREA

- It is a Temporary memory

FUNCTIONALITY OF HIDE :

- When Ever we are displaying the data in the output, we need to store a copy of the output in to HIDEAREA memory using the Below SYNTAX

SYNTAX

```
Loop at I_KNA1 into wa_KNA1.
  WRITE: wa_KNA1-KUNNR, wa_KNA1-NAME1....
  HIDE WA_KNA1_KUNNR
ENDLOOP.
```

↳ A copy of o/p is stored in HIDE AREA.

- When Ever the User double click's on any LIST LINE, the System will take the selected line and it will check wether the Selected line is available in HIDE AREA

- If the line is available the data is re-trive from Hide area and it is stored back in to HIDE variable.
- Using this Hide variable we can generate the interact Report.

Example program

* NOTE: Same program as above with Below changes

```
Loop AT I_MARA INTO WA_MARA.
  write : /wa_MARA-MANR, WA_MARA-MTART,
                                WA_MARA-MEIN
  HIDE : wa_MARA-MATNR.   " Stores variable in HIDE AREA
ENDLoop.

AT LINE SELECTION.
  Sy-LSIND = 1
  Select *FROM MAKT
      INTO TABLE I_MAKT
        WHERE MATNR = wa_MARA-MTNR.
Loop AT I_MAKT INTO WA_MAKT.
  WRITE : /WA_MAKT-MATNR, WA_MAKT-SPRAS,
                WA_MAKT-MAKTX.
ENDLoop.
```

USE HIDE variable Her

System will Search Hide area If found

Hide area with more than One Secondary list

```
Select
=
  Loop at I_mara into wa_mara
    write : /wa_mara-matnr, ...
  HIDE  WA_MARA-MATNR.
  ENDLOOP

AT LINE SELECTION
  If SY-LSIND = 1
      select * from MAKT
      =
      where matnr = wa_MARA-MTNR
  Loop at E_MAKT INTO wa_MAKT
    write : / wa_MAKT-MATNR, ....
  HIDE  WA_MAKT-MATNR.
  ENDLOOP.

ELSE IF SY-LSIND = 2
      Select * from MARC
      =
      wher matnr = wa_maKT-matnr.
  Loop at I_MARC into wa_MARC.
  write : / wa_MARC-matnr, wa_marc-werks.
    * HIDE ..
  ENDLOOP.

Elsse if SY-LSIND = 3
      SY-LSIND = 2
ENDIF.
```

Business Requirement

- Develop a Sale order Report which displays Sale order Header details in the Basic list and item details in the Secondary LIST.
- Display the materials details also, once the User double clicks on material Number on the Secondary LIST 1

Tables used : VBAK, VBAP, MARA

"GET CURSOR"

- This statement is used to read the details of Field Name and field value on which the double click Event is Raised.
- Get Cursor is mainly used to generate various Secondary list's depending on Selected Field.

SYNTAX :

```
GET CURSOR   FIELD  <FNAME>
             VALUE  <FVALUE>
```

Example Report

```
Select ---

[Loop at I_marc in to wa_marc.
   write:/ wa_marc-matnr, wa_marc-mtart...
ENDLoop

AT LINE SELECTION
    SY-LISIND = 1
      Get cursor  Field  V_FNAME
                  Value  V_FValue
If V_FNAME = 'wa_marc-MTNR'
      = SELECT *
                  where matnr = V_FVALUE.
Else If  V_fname = 'wa_marc-MTART' .... Endif.
```

MENU PAINTER (CUSTOM GUI)

- Custom GUI is used to create our own GUI with custom buttons, menu's.
- SE41 is the Transaction code to create Custom GUI
- AT USER COMMAND is the Event which is Triggered when ever we raise an action on custom GUI
- SET PF STATUS

  It is the key word which is used to assign the Custom GUI to the program.

  - we can create custom button's or Menu's for the
    - MENU BAR
    - APPLICATION TOOL BAR
    - FUNCTION KEY (Standard Tool bar)

STEPS TO CREATE CUSTOM GUI

- Business Requirement

  Generate an output list which contains custom GUI with a custom menu option, Back Button as defined below.

MY MENU SYSTEM HELP
↓
CREATE CUSTOMER
CALL DATA DICTIONARY
CALL VENDOR ▶ create vendor
change vendor
Display vendor.

- Create a program with SET PF STATUS Statement

  SET PF-status ' MENU ' → double click and create

  write : ' A sample prog on Custom GUI '

- Double click on MENU and Create the custom GUI
- The Transaction Code SE41 will be displayed as below

  MENU BAR ⊞

  ★ APPLICATION TOOLBAR ⊞ (In Real time we use only

  FUNCTION KEYS ⊞ this Remaining we never use at all)

- Expand the menu bar and define a Menu option

  MYMENU

- Double click on MYMENU
- Define the function Code, function Text as Below,

CODE	TEXT
SE11	CALL DATA DICT
X001	CREATE CUSTOMER
MM01	CREATE MATERIAL
	VENDOR

- Double click on vendor and define function Code and function Text as Below.

XK01	CREATE VENDOR
XK02	CHANGE VENDOR
XK03	DISPLAY VENDOR

- Save and activate it.
- click on Back Button
- And write the Below Code in the program,

```
AT USER-COMMAND.
  If sy-ucomm = 'SE11'.
   CALL TRANSACTION 'SE11'.
  Else If sy-ucomm= 'XD01'.
   CALL TRANSACTION 'XD01'.
  Elsif sy-ucomm = 'mm01',
   CALL TRANSACTION 'mm01'.
 Else if
   :
 ENDIF
```

- Save, Activate and Test it.

USING STANDARD TOOL BAR

- When Ever we Create Custom GUI, The standard toolbar button's are automatically disabled.
- To Enable Them we need to give the function code for Each button
  - Double click on SET PF STATUS
  - Expand the function Keys
  - Provide the standard FCT code (function code) for each button.

BACK EXIT CANCEL

- Save Activate and Test it.

USING APPLICATION TOOL BAR

- Double click on the SET PF Status.
- Expand application tool bar
- we can Create a maximum of 35 custom buttons on application tool bar

STEPS

- Give a function code in the first Box
  Ex, HELP
- Double click on it.
- press ENTER.
- provide the function TEXT, ICON NAME, Info Text,
- press Enter
- Assign a short cut function Key
  Ex, Shift +F5
- provide the ICON TEXT Also.
- Save and activate it.
- write the Below ABAP code.

```
If SY-UCOMM = 'HELP'.
  MESSAGE 'we are working App tool bar button'
                                    TYPE 'I'.
```

Example on Custom GUI Buttons

- Develop a material master Report to download only the Selected Records

→ Display a check box to Select Each record
→ Create Custom GUI with 3 Custom buttons by Name Select all, deselect all, download.

"ALV REPORTS" ABAP LIST VIEWER

Displaying the data in ALV format are called as ALV REPORTS.

– The main advantage of ALV Reports are

1. Better Look and feel
2. 'n' number of pre-defined options are already available.

Like,

- SORT IN ASCENDING ORDER
- SORT IN DESCENDING ORDER
- SET FILTERS
- TOTALS AND SUB TOTALS
- CHANGING LAY OUT STRUCTURE
- DOWLOADING DATA into Files.
- PRINT PREVIEW, MAIL RECIPIENTS

Function modules for developing ALV Reports

If u don't Know
SE37
REUSE_ALV *

1. REUSE_ALV_GRID_DISPLAY ⇒ Displays ALV data in GRID format
2. REUSE_ALV_LIST_DISPLAY ⇒ Displays ALV data in LIST format
3. REUSE_ALV_COMMENTARY_WRITE ⇒ Used to display TOP-OF-PAGE, Logo, END-OF-LIST.
4. REUSE_ALV_FIELDCATELOG_MERGE ⇒ Used to generate Field Catelog automatically.
5. REUSE_ALV_EVENTS_GET ⇒ Used to display all ALV EVENTS
6. REUSE_ALV-HIERSEQ_LIST_DISPLAY ⇒ To display Hierarchical ALV
7. REUSE_ALV_BLOCKED_LIST_DISPLAY ⇒ To display Blocked ALV

LIST OF THE ALV'S TO BE DISCUSSED

1. ALV with structure = we may not use in Real time
2. ALV with field Catelog options
3. ALV with Layout options
4. ALV with field Catelog – Merge
5. ALV with Totals and Sub totals
6. ALV with LOGO / TOP OF PAGE / END OF LIST
7. Interactive ALV
8. Interactive ALV By Calling a Transaction
9. Heirarchical ALV
10. Blocked ALV's.

EXAMPLE ON ALV WITH STRUCTURE

Busines Requirement

Develop a material Master details REPORT which displays all the fields

STEPS

- Declare an Internal Table and work area for the MARA Table
- Write a Select Statement to fetch the data
- Call the function module Reuse_ALV_GRID_DISPLAY And Specify the Below parameters,

1. STRUCTURE NAME =
2. PROGRAM NAME =
3. ITAB NAME =

```
data : I_mara type table of mara.
data : wa_mara type mara.

START OF SELECTION.
    PERFORM GET_DATA.
END OF SELECTION.
```

```
PERFORM display_data.
FORM get_data.
    Select * from mara
        Into Table I_mara
        Up to 100 Rows.
ENDFORM.
FORM DISPLAY_data.
  CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
    EXPORTING
      I_CALLBACK_PROGRAM = SY-REPID
      I_STRUCTURE_NAME = 'MARA'
    TABLES
      T_OUTTAB = i_mara.
ENDFORM.
```

SY-REPID → *SAP STANDARD to give prog Name

'MARA' → what fields to be displayed.

i_mara → what data to be displayed.

ALV WITH FIELD CATELOG

Field Catelog

- It is an internal table which contains the list of the Fields to be displayed in the ALV Report
- The field Catelog also contains various options to set the properties for Each Field.
- The properties are defined Below.

1. COL_POS :- Specify the Column position of a field.
2. FIELD NAME :- Specify the Name of the field to be displayed Under the Column position.

3. TABLE NAME :- Specifys the Name of the internal Tay where the Fields are available.

4. SELTEXT_S, SELTEXT_M, SELTEXT_L :-

Specifys the short, medium, Long Column Heading Text.

★★ 5. DO_Sum = 'X' :- Specifys wether Grand totals to b Calculated on the field.

If DO_Sum = 'X', Grand total will be Calculated

If DO_Sum = ' ', Grand totals are NOT Calculated

6. SUBTOT = 'X' :- Specifys whether Subtotals to be Calculated or not on a field.

★★ 7. EDIT = 'X' :- Specifys whether the field can be Editable or not.

★★ 8. EMPHASIZE :- Specifys whether we can apply some Colors to Highlight the Fields.

SYNTAX :

EMPHASIZE = CXYZ

C - Color

X - Color Number

Y - BOLD < 1 = ON Background Color, 0 = OFF Background Color

Z - INVERSE

1 → FONT COLOR =

0 → NO FONT COLOR

EX : EMPHASIZE = C610

9. REFERENCE TABLENAME, REFERENCE FIELD NAME :-

Specifys to copy all the properties from data dictionary to a field Catelog field.

★★ 10. KEY :- Specify whether the field is Key field or Not.

- All the Key fields will be Blue in Color and they will Remain in FIXED position
  i, NON SCROLLABLE.

  If KEY = 'X'

★ NOTE

- We need to Specify the column position, Field Name, SelTEXT to display a field in the field Catelog.
- The Remaining properties are optional.

' TYPE-GROUP '

① It is a dictionary object which Contains all the Re-Usable User defined TYPES.

Example : SLIS

- It is a TYPE GROUP which Contains all the User defined types for developing ALV Reports

TYPE-POOLS

- It is a statement or Key word which is used to assign the type group to a ALV Report.

SYNTAX FOR FIELD CATELOG

```
DATA : <IT_FCAT> TYPE SLIS_T_FIELDCAT_ALV.
DATA : <WA_FCAT> TYPE SLIS_FIELDCA_ALV.
DATA : <WA_FCAT> ~~TYPE~~ LIKE line of IT_FCAT.
```

★ Business Requirement

- Develop a material master Report which displays MATNR, MTART, Industry Sector, and Units.

EXAMPLE ALV with field catalog

Same program as above with the below changes.

1. Declare an Internal table and work area for the field catelog.
2. Create a sub routine to set the properties for Each Field
3. Call the Function module and send the field catelog in Internal table as an Exporting parameter to IT_FCAT

Ex:

```
Call function 'REUSE_ALV_GRID_DISPLAY'
EXPORTING
    I_CALLBACK_PROGRAM = SY-REPID
    IT_FIELDCAT = I_FCAT
Tables
    T_OUTTAB = I_marq.
```

ALV WITH LAYOUT

LAYOUT

- It is a structure or work area which is used to decorate or Embellish the output of ALV Report.
- This Lay out Contains few properties to decorate the output
- The properties are

1. **ZEEBRA = 'X'**

   The output will be displayed with alternative colors.

2. **COL_WIDTH_OPTIMIZE = 'X'**

   Each Column in the ALV output will be displayed with maximum length, to display the Entire data.

3. EDIT = 'X'

All the columns in the ALV outputs will be displayed in Editable mode.

4. NO_VLINE = 'X'

Vertical lines will not be displayed.

*NO_OUT = 'X' ✓

5. NO_HLINE = 'X'

Horizontal lines will not be displayed.

Syntax to declare work area for Layouts

DATA: wa_layout TYPE SLIS_LAYOUT_ALV.

Ex:

```
wa_layout = Zebra = 'X'.
wa_layout - colwidth_optimize = 'X'
wa_layout - Edit = 'X'.
wa_layout - NO_VLINE = 'X'.
wa_layout - NO_HLINE = 'X'.
```

Call function REUSE_ALV_GRID_DISPLAY

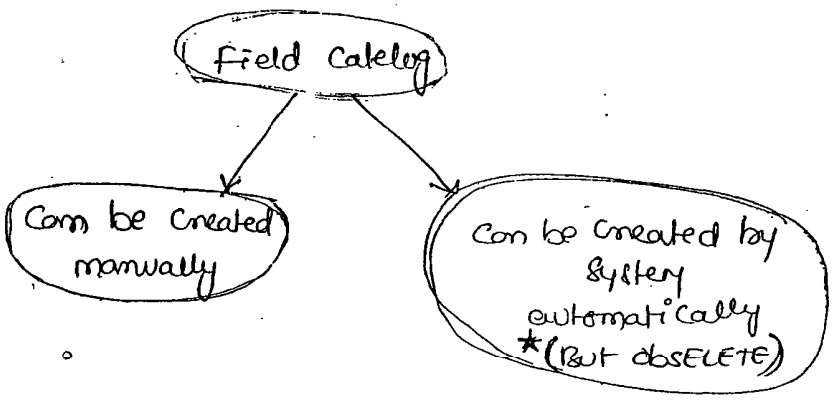
Exporting

```
I_CALLBACK_PROG = SY-REPID
IT_FIELDCATELOG = I_FACT
IS_LAYOUT = wa_LAYOUT
```

TABLES

```
T_OUTTAB = I_MARA.
```

"ALV with Field Catelog – Merge"



- We can Create a field Catelog either manually or by the System automatically.
- REUSE_ALV_FIELD CATELOG_MERGE is the function module which is used to create field catelog automatically.
- But this is obselete Because, the function module uses the old syntax internal Tables.
  ie, The internal Table should be Created as Below,

  And, All the fields in the internal Table must be declared using LIKE statement, Not the type statement.

Example

```
DATA : Begin of I_MARA OCCURS 0,
         MATNR LIKE MARA-MATNR,
         MTART LIKE MARA-MTART,
         MBRSH LIKE MARA-MBRSH,
         MEINS LIKE MARA-MEINS,
       END OF I_MARA
```

I_mara should be declared using old System and All fields should use 'LIKE'

```
Form Create_FCAT_MERGE.
  Call Function 'Reuse_ALV_FieldCatelog_merge'
    Exporting
      I_Program_Name = SY-REPID
      I_Internal_TABNAME = 'I_MARA'
      I_INCLNAME = SY-REPID
```

changing

CT_FIELDCAT = I_FCAT.

* Additional changes to Generated fieldcatelog.

Loop at I_FCAT Into wa_FCat.

If wa_fcat = FieldName = 'meins'.

wa_fcat-NO_out = 'x'.

modify I_FCAT from wa_FCat INDEX SY-TABIX.

ENDIF.

If wa_FCat-FieldName = 'MTART'.

wa_feat-KEY = 'x'.

modify I_FCAT From wa_feat INDEX SY-TABIX.

ENDIF.

"ALV WITH TOTALS"

- The totals can be calculated in two ways.

1. By clicking on [Σ] on ALV output field.
2. By programatically setting the option DO_Sum = 'x' for a Currency/Quantity field.

Ex:

wa_FCAT-FIELDNAME = 'NETWR'.

wa_FCAT-SELTEXT_M = 'NETPRICE'.

wa_FCAT-DO_SUM = 'x'.

APPPEND wa_FCAT TO I_FCAT.

CLEAR WA_FCAT.

ALV WITH SUBTOTALS

- To Calculate Sub totals, we need to find out on what basis i,e, field Name,
  The Sub totals need to be calculated.
- we need to Sort the field in the Ascending order
- Now we need to set the property,
  SUBTOT = 'X'.

Ex:

```
DATA: I_SORT TYPE SLIS_T_SORTINFO_ALV.
DATA : wa_SORT TYPE SLIS_SORTINFO_ALV.

  wa_SORT-FIELDNAME = 'VBELN'
  wa_SORT-UP = 'X'.          { UP - means Ascending order
  wa_SORT-SUBTOT = 'X'.      { DOWN - descending order
  Append wa_SORT TO I_SORT.

Call function REUSE_ALV_GRID_DISPLAY
EXPORTING
        =
    IT_SORT = I_SORT
TABLES
    T_OUTTAB = I_VBAP.
```

ALV WITH TOP of Page / EVENTS

1. There are totally 17 Events available for ALV Reporting
2. The function module REUSE_ALV_EVENTS_GET will display all the list of ALV Events.
3. Below are some Events,

1. TOP-OF-PAGE
2. END-OF-LIST
3. AT USER-COMMAND
4. SET PF-STATUS

TOP-OF-PAGE EVENT

- Follow the Below steps to generate TOP-OF-PAGE Event in ALV.

1. Call the function module REUSE_ALV_EVENTS GET
2. provide the Sub routine Name for the Top of page Event.
3. write down the Code in the Sub routine.
4. Call the function module REUSE_ALV_COMMENTARY_WRI. to display top-of page in ALV.
5. Finally Export the Events internal Table to the REUSE_ALV_GRID_DISPLAY

STEP 1

- Call the function module REUSE_ALV_EVENTS_GET

Importing

E T_EVENTS = (I_EVENTS.) → It will contain all 17 ALV Events.

STEP 2

- provide the Sub routine Name for top of page Event.

```
READ TABLE I_EVENTS INTO WA_EVENTS
                    WITH EKY NAME = 'TOP_OF_PAGE'
WA_EVENTS-FORM = 'FORM_TOP'
MODIFY_I_EVENTS FROM WA_EVENTS INDEX SY_TABIX.
```

Step 3

- write down the Below Code in the Subroutine.

```
FORM FORM_TOP.
    WA_HEADING-TYP = 'H'.
    WA_HEADING-INFO = 'material master Report'.
    APPEND WA_HEADING TO I_HEADING.
  CALL FUNCTION 'REUSE_ALV_COMMENTARY_WRITE'
    EXPORTING
      IT_LIST_COMMENTARY = I_HEADING.
ENDFORM.
```

Step 4

- Call the function module REUSE_ALV_GRID_DISPLAY and provide I_EVENTS internal Table Name.

```
FORM DISPLAY_ALV.
  call function 'Reuse_ALV_Grid_display'
    Exporting
      =
      IT_EVENTS = I_EVENTS.
    TABLES
      T_OUTTAB = I_mara.
ENDFORM.
```

ALV WITH END OF LIST EVENT :

- Call the function module REUSE_ALV_EVENTS_GET
- provide the Subroutine Name

```
Read table I_EVENTS into wa_Events
                    with Key Name = 'END_OF_LIST'.
wa_EVENTS-FORM = 'FORM_END'.
MODIFY I_EVENTS FROM WA_EVENTS INDEX SY-TABIX.
```

- Define the Subroutine and write the Below Code.

```
FORM FORM_END.
PERFORM I_HEADING.
    WA_HEADING-TYP = 'S'.
    WA_HEADING-INFO = 'IBM INDIA'.
APPEND WA_HEADING TO I_HEADING.
CALL FUNCTION 'REUSE_ALV_COMMENTARY_WRITE'
    EXPORTING
        IT_LIST_COMMENTARY = I_HEADING
    I_END_OF_LIST_GRID = 'X'  → Indicates to display Info at the END-OF-LIST
ENDFORM.
```

- Call the function module and provide I_EVENTS internal Table.

```
FORM DISP_ALV.
  Call function 'REUSE_ALV_GRID_DISPLAY'
    Exporting
      IT_EVENTS = I_EVENTS
    Tables
      T_OUTTAB = I_MARA.
ENDFORM.
```

" LOGO IN ALV REPORTS "

- OAER is the Transaction code to upload the image from presentation Server in to SAP downyent Server.
- Call the function module REUSE_ALV_COMMENTARY_WRITE and set the parameter I_LOGO with Logo Name

STEPS

1. Go to OAER Tcode
2. Specify the parameter as Below.

   Class Name : pictures
   Class type : OT
   Object Key : ZImage (Any name for image)
3. Click on Execute
4. Go to Create Double Tab
5. Expand standard document types folder.
6. double click on Screen
7. Select the Image Name
8. click on Back Button
9. The Image will be successfully stored in SAP Server.
10. Now call the Function module and give the Logo Name,

```
CALL FUNCTION 'REUSE_ALV_COMMENTARY_WRITE'
  EXPORTING
    IT_LIST_COMMENTARY = I_HEADING
    I_LOGO = 'ZIMAGE'
```

↳ Name of Image in OAER

INTERACTIVE ALV's

- USER_COMMAND is the Event Triggered when Ever the User double click's on Basic Alv out put list.
- The System variable sy-ucomm will store the function code (FCT code) of the double click action

  ie, sy-ucomm = '&IC1'
  ***
- The System Variable SELFIELD type SLIS_SELFIELD will Contain the Content of Selected line.
- SELFIELD-TABINDEX will give the Line Number

- SELFIELD-FIELDNAME will give the Name of the field.
- SELFIELD-VALUE will give the value where double click action is raised by the user.

STEPS

1. Call the function module REUSE_ALV_EVENTS_GET to get the list of the Events.
2. provide a subroutine Name for the Event USER_COMMAND

Ex: form_UC USING Ucomm TYPE SY-Ucomm
SELFIELD TYPE SLIS_SELFIELD.

3. write the Below code in the subroutine.

```
CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
  EXPORTING
    INPUT = SELFIELD-VALUE
  IMPORTING
    OUTPUT = V_MATNR.
  Select MATNR SPRAS MAKTX FROM MAKT
    INTO TABLE I_MAKT
    WHERE MATNR = V_MATNR.
  PERFORM CREATE_RCAT2.
  PERFORM DISP_ALV2.
```

** * SAP MEMORY / ABAP MEMORY

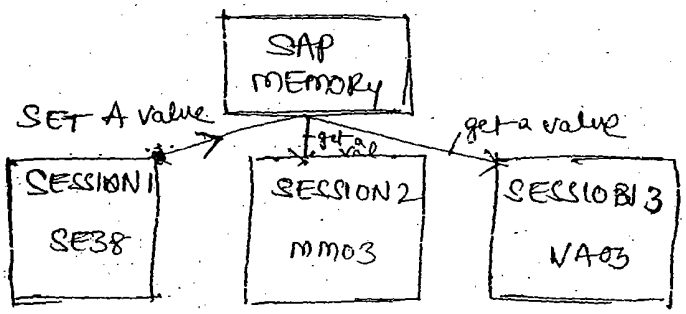
- SAP memory is a global memory which is used by Extern sessions
- SET, GET are the statements which are used to set a val and Read a value from global memory.

SYNTAX

SET PARAMETER <PARAMETER ID> FIELD <FIELD VALUE>

GET PARAMETER <PID> FIELD <FIELD VALUE>

- The Main Use of SAP memory is to SET the value in global memory so that they can be used by standard Transactions to Fill the field values



** How do you find out PARAMETER ID ?

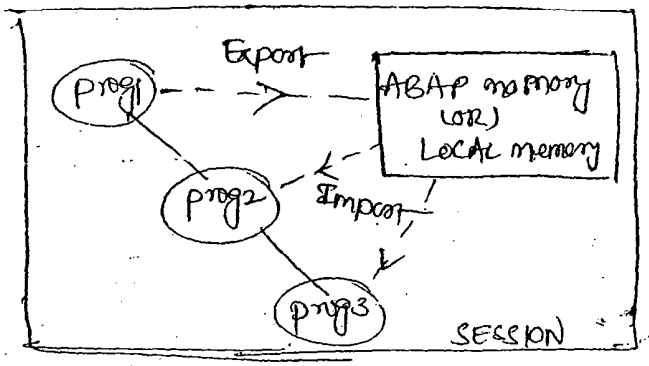
1. Go to any Transaction, Ex MM03 or XD03
2. Click on F1 on any field Ex: material field
3. Click on Technical information Icon
4. It will display the program Name, Screen No, TableName, field Name, Data element Name, PARAMETER ID

Example program on SET PARAMETER

```
Parameters : P_matnr TYPE mara-matnr.
data: V_matnr TYPE mara-matnr.
V_matnr = P_matnr.
Set parameter ID 'mat' FIELD V_matnr.
Call Transaction 'MM02' AND SKIP FIRST SCREEN.
```

ABAP MEMORY

- ABAP memory area that all ABAP programs with in the same internal session can access using the EXPORT and IMPORT statements.
- ABAP programs can Export parameters using the Export statement
- Other ABAP programs can receive parameters using the IMPORT statement.



EXPORT
IMPORT

"SUBMIT"

- It is a Key word which is used to call Another ABAP program from the current program.

SUBMIT And Returning

- It is a Key word which is used to call Another ABAP program from the current program and the returning will be automatic from the second program to first program.

EXPORT AND IMPORT

- These are the statements or Key words which are used to Export and Import the values Between different ABAP programs via ABAP memory.

SYNTAX

EXPORT <VALUE> TO MEMORY ID <memory ID/ Name>

IMPORT <VALUE> TO <LOCAL VAR> FROM memory ID <MID/NAME>

Syntax

SUBMIT <PROG_NAME>

SUBMIT <PROG_NAME> and Return.

Example

- Create a first ABAP program By NAME ZEXPORT and write the below code,

```
Parameters : P_MTART TYPE MARA-MTART.
EXPORT P_MATART TO MEMOMRY ID 'ZMTART'.
SUBMIT ZIMPORT AND RETURN.
write :/ ' This is ZEXPORT program'.
```

- Create a Second ABAP program by Name ZIMPORT and write the Below code.

```
DATA : LV_MTART TYPE MARA-MTART.
DATA : I_MARA TYPE TABLE OF MARA.
DATA : WA_MARA TYPE MARA.
Import P_MTART TO LV_MTART from memory ID
                                    'ZMTART'.
  Select * From Mara
        into Table I_MARA
        where MTART = LV_MTART.
Loop at I_MARA INTO WA_MARA
  WRITE : / wa_MARA-MATNR, WA_MARA-MTART.
ENDLoop.
```

- Now Execute the first program, which internally Executes the Second program and displays the output list of Second program as well as first program.

Interactive ALV's to CALL Standard Transactions

```
FORM FORM_USER_COMMAND USING UCOMM TYPE SY-UCOMM
                             SELFIELD TYPE SLIS SELFIELD.
  SET PARAMETER ID 'AUN' FIELD SELFIELD-VALUE.
  CALL TRANSACTION 'VA03' AND SKIP FIRST SCREEN.
ENDFORM.
```

'AUN' → parameter ID for VA03 (Sale doc field)

"HEIRARCHIAL ALV'S"

- Displaying the Basic and Secondary Information on the Basic list in the form of Heirarchi is called Heirarchial ALV Reports.
- The Function module used is REUSE_ALV_HEIRSEQLIST_DISPLAY
- We Need to pass the Below parameters to this function module

  Exp
  - Program Name
  - Header Table Name
  - Item Table Name
  - Field Catelog
  - Key Information

- KEY INFORMATION
  - It is a work area which consists of common field Names. ie, KEY FIELDS Between Header Table and Item Table
  - We can provide a maximum of 5 common fields Between Two Tables.
  - The main Use of the Key Information is to Create the Heirarchi
    ie, Link Between Header and Item Tables.

NOTE

when ever we generate the field catelog for Heirarchy ALV Report we need to set the Table Name also in the field catelog.

Example for field catelog

```
FORM CREATE_FCAT.
    WA_FCAT-COL_POS = '1'
                FIELDNAME = 'VBELN'.
                TABNAME   = 'I_VBAK'.
                SELTEXT_M = 'DOCNO'.
APPEND WA_FCAT TO I_FCAT.
CLEAR WA_FCAT.
```

EXAMPLE FOR KEY INFORMATION

```
DATA : WA_KEY TYPE SLIS_KEYINFO_ALV
FORM CREATE_KEYINFO.
    WAR_KEY-HEADER01 = 'VBELN'.
    WA_KEY-ITEMNO01  = 'VBELN'.
```

EXAMPLE FOR HEIRARCHIAL DISPLAY

```
CALL FONCTION 'REUSE_ALV_HIERSEQ_LIST_DISPLAY'.
EXPORTING
    I_CALLBACK_PROGRAM = SY-REPID.
  IT_FIELDCAT = I_FCAT
  I_TABNAME_HEADER = 'I_VBAK'
   I_TABNAME_ITEM = 'I_VBAP'
  IS_KEYINFO = WA_KEY
TABLES
   T_OUTTAB_HEADER = I_VBAK
   T_OUTTAB_ITEM = I_VBAP.
```

"BLOCKED ALV" "(OBSELE)

- Displaying the information on a single screen in the form of Blocks are called BLOCKED ALV.
- The function modules used are
  1. REUSE_ALV_BLOCK_LIST_APPEND → Add a Block to Blocked ALV
  2. REUSE_ALV_BLOCK_LIST_DISPLAY → Display Blocked ALV
  3. REUSE_ALV_BLOCK_LIST_INIT → Initialize Blocked ALV

★ RUNTIME ANALYSIS

★ • Analyzing or checking the efficiency of a program, TCODE Function module is called Runtime Analysis.

- The efficiency is checked in terms of,
  - what is the load on database Server, Application Server, presentation Server.
  - The efficiency will be displayed in a graph mode representing the load in percentages.
  - There are 3 colors which represent the efficiency
  - Color RED will Indicate the program need's to be optimize with Best programming Standard.

    Color GREEN will Indicate the program efficiency is good.

    Color YELLOW Indicates warnings.
- SE30 is Transaction code for Runtime Analysis.

★ NOTE

The load on the data base Server should be always 40%.

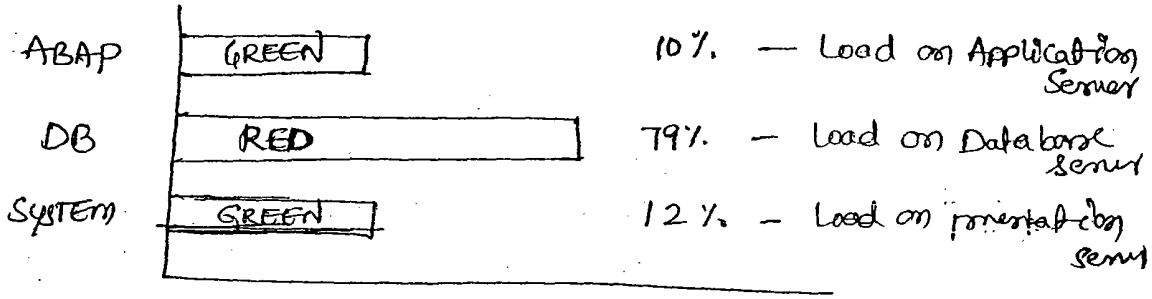
EXAMPLE PROGRAM ON RUNTIME ANALYSIS

```
DATA : I_BKPF TYPE TABLE OF BKPF.
DATA : WA_BKPF TYPE BKPF.
SELECT * FROM BKPF
    INTO CORRESPONDING FIELDS OF TABLE I_BKPF
    WHERE XBLNR <> '',   " don't specify Key fields
LOOP AT I_BKPF INTO WA_BKPF.
  WRITE : / WA_BKPF-BUKRS, WA_BKPF-BELNR,
                           WA_BKPF-GJAHR.
ENDLOOP.
```

- Save
- Activate, Test it

Steps

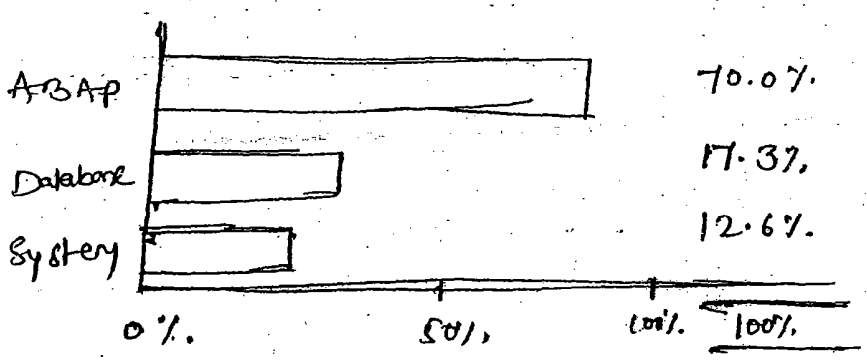
- Go to SE 30
- Give the program name
- click on the Button [Execute]
- click on [Evaluate]
- It will display a graph



- Now modify the Above program By Including the KEY FIELDS in the where Condition of the Select statements.

```
Select * FROM BKPF
  INTO Corresponding fields of Table I_BKPF
  WHERE BUKRS = '0001'.
```

- Now check the efficiency in SE 30
- It will display the Below graph.



PERFORMANCE TUNING (SQL TRACE)

- It displays the time taken for each open SQL statement in a program
  ie, SELECT, INSERT, UPDATE, MODIFY, DELETE
- ST05 is the Transaction code for performance Tuning.
- Once the performance tuning is done we should find out wether any statement is displayed with **RED** in color.
- Suppose if it is available we need to optimize that statement By following the Best programming standards.

EXAMPLE

- Create the same program as above with non-key fields in the where condition.
- Now check the performance Tuning By following Below steps.
  1. Go to ST05
  2. Click on activate Trace button
  3. Now open another session
  4. Execute the program
  5. Click on de-activate Trace button.
  6. Click on display Trace button.
  7. Performance Tuning will be displayed with time Taken for each Select statement.
  8. Identify wether there is any statement with **RED** Color
  9. If it is available, optimize the SELECT statement By Including the Key fields or following BEST PROGRAMMING STANDARDS.

Duration	obj NAME	OP	REC	RC	Statement
7	BKPF	REOPEN		0	SELECT WHERE 'MANDT = '800' AN 'BUKRS' = '0001'
532	BKPF	FETCH	2	1403	

★ "BEST PROGRAMMING STANDARDS"

1. Always Create internal Tables and work area's refering to TYPES statement.
2. Never Use SELECT *
3. Never User INTO CORRESPONDING FIELDS
4. Always Use Key Fields in The WHERE Condition
5. If NON-Key fields are available in The where condition Create Secondary Index
6. Always Use joins for 2 Tables
7. USE FOR ALL ENTRIES FOR MORE THAN TWO TABLES
8. check wether The first internal Table is initial or not when using for all Entries.
9. Always Use Binary Search To read a Record.
10. SORT The internal Tables Before using BINARY SEARCH
11. Never Use Nested loops.
12. Never Use Select statement Inside The Loop.
13. Always Use SELECT SINGLE * By providing KEY FIELDS.

→ Methodology :

Chowdary.a.b@gmail.com

RBOMOIND

BOTS

TED52 - Sbroscody ✓