**SAP-BASIC ABAP**

**NAME : LAKSHMA REDDY G.**

**INSTITUTE : RELIENCE GLOBAL SERVICES**

**AUTHOR : RAJA SHEKAR**

**INDEX**

|  |  |  |
| --- | --- | --- |
| 1 | Introduction to ERP | 5 |
| 2 | Introduction to SAP R/3 | 6 |
| 3 | SAP R/3 Services | 8 |
| 4 | Introduction to ABAP/4 & ABAP Work Bench Tools | 10 |
| 5 | ABAP Data types | 12 |
| 6 | ABAP Programming Structures | 14 |
| 7 | Steps to Write & Create SAP Program | 16 |
| 8 | Tips for Writing a Program in Real Time | 19 |
| 9 | Internal Table & difference between database table & internal table | 20 |
| 10 | Reading Data from Body to Work Area (or) Header Line  (1. LOOP….ENDLOOP, 2. READ) | 22 |
| 11 | Selection Screen (1. Select-options, 2. Ranges, 3. Parameters ) | 26 |
| 12 | Other SELECT-OPTIONS keywords (1. Obligatory, 2. No Intervals,  3. No-Extension, 4. No-Display) | 28 |
| 13 | Other Important Keywords for Internal Table (1. Append, 2. Sort, 3. Delete Adjacent Duplicates, 4. Collect 5. Modify ) | 31 |
| 14 | 6. Deleting Data from Internal Table by (1. Fresh, 2. Refresh, 3. Clear, 4. Delete ) | 36 |
| 15 | 7. Describe Keyword | 37 |
| 16 | 8. Coping data b/w work areas (1. Move, 2. Move-Correspondig) | 38 |
| 17 | 9. Coping data b/w internal tables (1. Loop…….Endloop, 2. Append Lines) | 39 |
| 18 | Control Break Statements (1. At First, 2. At Last, 3. At New, 4. At Change,  5. At End ) | 41 |
| 19 | Insert keyword & Syntaxes for Internal Tables | 45 |
| 20 | System Fields for Internal Table | 46 |
| 21 | DML Operations on Database Table (insert, update, delete, modify keywords) | 48 |
| 22 | Types of Internal Tables | 50 |
| 23 | Data Dictionary (9 objects) 1. Database Table | 51 |
| 24 | 2. Structure (.include & Append Structure) | 55 |
| 25 | 3. View(Database view, Projection view) | 58 |
| 26 | 4. Search Help (Elementary search help, Collective search help) | 60 |
| 27 | Table Maintenance Generator (TMG) (TCode, Default values, Check boxes) | 63 |
| 28 | Text Table | 70 |
| 29 | Some Important Steps for Working with Standard Tables | 72 |
| 30 | Types of Tables (Transparent, Cluster, Pooled Tables) | 73 |
| 31 | Open SQL Statements & Native SQL Statements | 74 |
| 32 | Buffering (Single Record, Generic, Full Buffering) | 75 |
| 33 | MODULES(1. MM Module) | 76 |
| 34 | Inner Joins | 79 |
| 35 | Table Fields & Links (1. LFA1, 2. EKKO, 3. EKPO) | 84 |
| 36 | Message Handling | 86 |
| 37 | Modularization Techniques (5)  1. Subroutines (Internal Subroutine, External Subroutine) | 88 |
| 38 | Break Point (1. Static Break Point 2. Dynamic Break Point) | 90 |
| 39 | Pass by Value & Pass by Reference | 91 |
| 40 | Table Fields & Links (1. MKPF, 2. MSEG) | 94 |
| 41 | Function Module (1. Normal, 2. Remote Enabled, 3. ALv) | 96 |
| 42 | Include Program | 100 |
| 43 | Field Symbol | 101 |
| 44 | FOR ALL ENTRIES | 103 |
| 45 | Table Fields & Links (1. MARA, 2. MAKT, 3. TSPAT) | 108 |
| 46 | 2. SD Module | 109 |
| 47 | String Operations (shift, translate, replace, strlen, offset functionality, split, concatenate, condense) | 111 |
| 48 | Looping Statements (1. Conditional, 2. Unconditional) | 115 |
| 49 | CONTINUE, CHECK, EXIT keywords | 118 |
| 50 | Mathematical Operations (1. Frac, 2. Ceil 3. Floor, 4. Abs, 5. Rem, 6. Mod,  7. Sign, 8. Trunc, 9. Div ) | 120 |
| 51 | SAP Memory & ABAP Memory | 122 |
| 52 | Support Project | 125 |
| 53 | System Landscape | 126 |
| 54 | Version Management | 130 |
| 55 | Sub Objects in ABAP Editor (1. Source Code, 2. Variants, 3. Attributes,  4. Documentation, 5. Text Elements) | 131 |
| 56 | Interview Questions ABAP Editor | 133 |
| 57 | Interview Questions Internal Tables | 136 |
| 58 | Interview Questions Data Dictionary | 143 |
| 59 | LFA1 Table | 161 |
| 60 | EKKO Table | 162 |
| 61 | EKPO Table | 163 |
| 62 | MKPF Table | 164 |
| 63 | MSEG Table | 165 |
| 64 | MARA Table | 166 |
| 65 | MAKT & TSPAT Tables | 167 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**ERP**

**ERP:**

* Enterprise Resource Planning
* ERP is a package under which all the business resources are integrated in one system

Enterprise-------organization

Resource--------FICO, HR, MM, PP, SD, CRM

**Types of ERP:**

1. High End ERP
2. Mid Range ERP

**1. High End ERP:**

* These ERPs are implemented in large scale industries
* In these ERPs all the tables, programs & applications are predefined

**Example:**

1. SAP R/3 (60000+ applications, 400000+ tables, 55+ modules)
2. Oracle Apps (for Finance)
   1. Oracle Finance
   2. Oracle Manufactures
   3. Oracle HRMS
3. SIEBEL (for CRM)
4. People Soft (for HR)

**2. Mid Range ERP:**

These ERPs are implemented in small scale & medium scale industries

**Example:**

1. Baan (for Finance)
2. JD Ed Wards (for logistics)
3. Ramco (for Cement)
4. Microsoft ERP (for Finance)

**Need for ERP:**

1. Large Organization
2. Centralized Database
3. User Friendly
4. Speed
5. Security
6. Competitive Advantage
7. Reduced Errors

**SAP R/3**

**SAP R/3:**

Systems, Applications, Products in data processing

R/3------ Real time 3 tier architecture

**History:**

SAP AG---- developed by 5 IBM employees in 1973----in woldoff (Germany)

SAP R/1(Finance)------1973

SAP R/2 (Mainframes)------1978

SAP R/3 (Client Server Technology)----1992

**Systems:**

These are basic resources for implementing a project

**Examples:**

Servers, Hardware, Soft ware, Systems, Network, Database………etc

**BASIS Consultant:**

**Roles:**

1. Installation
2. Maintenance
3. Configuration
4. Customization (or) Administration

**Applications:**

* These are collection of Screens
* Each Screen is a collection of Fields

Use---Applications are used by Clients (or) End Users

Create---Applications are created by Abaprs (or) Technical Consultant (or) ABAP Consultant

Customized Data----Applications data customized by Functional Consultant

**Functional Consultant:**

**Roles:**

1. Customization of Data
2. Getting requirements from Clients
3. Preparing Functional Documents
4. Preparing End User Manuals (or) Snapshots
5. Providing end user Training

**Products:**

* These are called Objects
* These are created by Programming

**ABAP Consultant (or) Abapare (or) Technical Consultant:**

**Roles:**

1. Creates Technical Documents (brief logic)
2. Creates an Object from scratch implementation project
3. Modifies existing objects to support project

**14-08-2014**

**Features of SAP:**

1. Designed based on RDBMS
2. Designed based on R/3 Architecture
3. Database independent
4. Supports all types of industry specific solutions
5. International package available in 40 languages
6. It can be customized using ABAP language
7. Best ERP for FI, SD, MM, PP, HR, CR
8. Supports Client-Server Technology
9. Highly versatile (Operating System independent)
10. Supports web based application softwares

**SAP R/3 Architecture:**

External Systems (Printer, Fax……etc)

Spool Service

Maintains & Manages Database

Objects are Executed, Compilation……etc

Programs, Applications, Objects…….etc

Dialog Update

Service Service

Presentation Server Application Server Database Server

**Services in R/3 Architecture:**

1. **Dialog Service:**

It provides interface between Presentation Server & Application Server

1. **Update Service:**

It provides interface between Application Server & Database Server

1. **Spool Service:**

It provides interface between SAP R/3 & External Systems (Printers, Fax….etc)

1. **Background Service:**

It is for background scheduling

**Note:**

In real time once report is created, then it is scheduled in background by BASIS Consultants, they will set time & event related to it

1. **En-queue Service:**

It maintains data integrity (locking mechanism) in SAP

1. **Message Service:**

It is for Error & Exception handling

1. **Gateway Service:**

It is for distributed environment

**Note:**

The above 7 services are part of Application Server

1. **SAP GUI Front End Service:**

It handles all SAP GUI Operations in SAP Presentation Server

1. **Database Service:**

It Maintains & Manages data operations in Database Server

**Versions in SAP:**

**Version** **Year**

1.0 & 1.1 1991

2.0 & 2.1 1991

3.0 & 3.1 1992

4.0 1998

4.6A, 4.6B, 4.6C 2000

4.7EE 2003

ECC 5.0 2005 (ECC----Enterprise Core Component)

ECC 6.0 2006 (Present Working)

BI/BO 2008 (Business Independent)

CRM, SCM, PLM 2009 (Customer Relationship Management, Product Lifecycle Management)

**Soft wares in SAP:**

1. Production Software
2. IDES Software (or) Training Software

**1. Production Software:**

* It is Real time
* Every company should purchase this software from SAP
* Here you will find only live data
* Clients (End Users) are works with Production Software

**2. IDES Software:**

* International Demonstration & Education System Software
* It is also known as Training Software
* Here you can find data for practice & examples for practice

**16-08-2014**

**ABAP/4**

Advanced Business Application Programming 4th Generation Language

* All SAP applications are designed & developed using ABAP Language
* It is a high level language

**Features of ABAP:**

1. It was designed based on ‘C’ language
2. It is a platform independent language
3. It is a case insensitive language
4. It is database independent
5. It is truly business oriented language
6. It is rich in data types
7. It was designed based on Object Oriented Programming
8. It is an event driven programming language
9. It is highly user friendly
10. It supports web based applications also

**ABAP Work Bench:**

In real time as a Technical Consultant have to work with ABAP Work Bench tools

**ABAP Work Bench Tools:**

1. **ABAP Dictionary:**

* It works with Transaction Code (OR) T Code **SE11** (System Engineering)
* Here we can create & store Tables (or) Data

1. **ABAP Editor:**

* It works with Transaction Code (OR) T Code **SE38** (System Engineering)
* Here we can create & execute Programs

1. **Screen Painter:**

* It works with Transaction Code (OR) T Code **SE51** (System Engineering)
* Here we can design Applications (OR) Screens

1. **Class Builder:**

* It works with Transaction Code (OR) T Code **SE24** (System Engineering)
* It is for Object Oriented ABAP

1. **Function Builder:**

* It works with Transaction Code (OR) T Code **SE37** (System Engineering)
* Here we can work with Function Modules

1. **Object Navigator:**

* It works with Transaction Code (OR) T Code **SE80** (System Engineering)
* Here we can create & modify the Objects & store the Objects

**Note:**

* In Object Navigator you can develop all Objects in SAP
* It is also known as a True ABAP Development Work Bench

**Requirements for Installing SAP:**

1. 4GB RAM
2. 500GB Hard Disk
3. Core 2 Duo Processor

**18-08-2014**

**ABAP Programming**

**ABAP Data Types:**

ABAP data types are classified into 4 categories

They are:

1. Numeric
2. Character
3. String
4. Hexadecimal

**1. Numeric:**

These are classified into 3 categories

They are:

1. Integer (I)
2. Packed Decimal (P)
3. Floating point (F)

**i) Integer (I):**

It holds integer values

**Ex:** 102, 55, 74, …..

**ii) Packed Decimal (P):**

* It returns accurate values with decimal values
* It provides accuracy in ABAP programming

**Ex:** 102.55, 56.235, …..

**iii) Floating Point (F):**

It improves performance in ABAP programming by rounding of the value to nearest decimal

**Ex:** 22.63-----23, 58.232-----58

**2. Character:**

These are classified into 3 categories:

They are:

1. Character (C)
2. Date (D)
3. Time (T)

**i) Character (C):**

It holds character value provided in ‘ ’ (single quotes)

**Note:**

In ABAP programming the default data type is Character

**ii) Date:**

* Use Date data type to store date values
* The default SAP format for date is YYYYMMDD

**iii) Time:**

* Use Time data type to store time values
* The default SAP format for time is HHMMSS

**3. String (String):**

A group of characters combined to form a String

**4. Hexadecimal (HX):**

It is for SAP graphics management

**Note:**

QUAN, CURR, NUME are SAP provided business data types for storing Quantities, Currencies & other Numeric values

**Type Keyword:**

It specifies the type of data an Object (Variable) can hold

**Example:**

1. Data ch type String --------hi, hello, how are you,………
2. Data cnt type I ------54, 1025, 748,…….
3. Data count type P values 2 -------145.22, 52.69,……
4. Data chr(10) type C ------ ‘a’, ‘h’,………

**Data Keyword:**

* Using Data keyword Objects are defined in ABAP programming
* Using Data keyword memory is allocated in buffer by system

**Note:**

Every Object (OR) Variable should be define using Data keyword

**Programming Structures:**

These are 2 types

They are:

1. Pre defined Programming Structure
2. User defined Programming Structure

**Pre defined Programming Structure:**

* It is given by SAP
* It contains Tables, Structures, Type groups,……….etc

**User defined Programming Structure:**

These are 2 types

1. Work Area
2. Internal Table

**1. Work Area:**

* Work Area is a Structure (group of fields with different data types)
* It holds only one single record during run time

**Syntax:** Chain Operator Name of Work Area (any name)

Data: Begin of wa\_KNA1, { wa\_table name (OR) w\_table name is recommended }

Customer(10) type C,

Name(20) type C,

City(20) type C,

End of wa\_KAN1. Period Symbol

Field Name

**Note-1:**

With above syntax during run time a Work Area wa\_KNA1 is created in Application Server

**Note-2:**

* Work Area name can be any name
* In companies we have to follow Naming (Coding) standards while creating a program
* According to these standards Work Area should always begins with w\_ (OR) wa\_ followed by Table Name

**Note-3:**

Every Object (Work Area) should be define with Data keyword

**Note-4:**

Every Programming Structure should begins with Begin of & ends with End of followed by the Programming Structure name

**Note-5:**

* : is known as Chain Operator
* It provides the repetition of the Declarative Part

**Requirement:**

Transfer Customer Master Data from Database Server to Application Server & from Application Server to Presentation Server

**Solution:**

Select kunnr name1 ORT01 from KNA1 into wa\_KNA1. ----- Data transfer from

Database Server to Application Server (Work Area)

Write:/10 wa\_KNA1-Customer,

30 wa\_KNA1-Name, data transfer from Work Area to Virtual Page then

60 wa\_KNA1-City. Presentation Server

Endselect. ----------- it repeats the Select statement until records completed

Work Area

100 satya Hyderabad

m

KNA1

kunnr name1 ORT01

100 satya Hyderabad

200 ram Delhi

300 raj Bangalore

100 satya Hyderabad

200 ram Delhi

300 raj Bangalore

100 satya Hyderabad

200 ram Delhi

300 raj Bangalore

Presentation Server Application Server Database Server

Virtual Page

**20-08-2014**

**Steps to write & execute SAP program:**

* Double click on **SAP Logon** on Desktop
* You will find a Logon Pad
* Double click on **Development (121.241.50.175)**
* Provide **Client**: 800 (only for practice purpose)

**User**: user1 to user10

**Password**: Rgsabap

* You will find **SAP Easy Access screen** (Screen 0)
* On top of the screen you will find cursor blinking at **Command Prompt (or) Command Field**. Here you should type Transaction Codes (T Codes) in real time
* Next to Command Prompt you will find **Standard Icons (Standard Application Tool Bar)**. Here you cannot add Customized Icons
* Under Standard Application Tool Bar you will find **Text Area**. It specifies documentation of current application
* Under Text Area you will find **Customized Application Tool Bar** when you can add your own icons apart from Standard ones
* Under Customize Application Tool Bar you will find **Work Area** where user performs his tasks (or) navigations
* Type **SE38** in Command Prompt & press Enter
* Type program **ZSD\_Customer\_Report**

**Note:**

SAP is providing standard programs which starts with A, B …X. If you want to create ZPrograms (User Defined) start with Y (OR) Z

* Click on **Create** button
* Provide **Title**: Customer Details Report
* Provide **Type**: Executable Program

**Note:**

With Executable Program system will generate SAPGUI.EXE file

* Then press Enter you will find a pop-up here leave **Package**:\_\_\_\_\_\_\_\_\_\_\_ as blank
* Click on **Local Object** button

**Note-1:**

In real time Objects needs to be transported from Development client to Testing client & from testing client to Production client. For that Objects should always be saved in a package. If an Object is saved in Local Object it cannot be transported between clients (Non Transportable Objects)

**Note-2:**

Local Object is meant for practicing Objects in real time

**Note-3:**

All Executable Objects are Report Programs

**Note-4:**

On top of the program in Comment Section you should provide

1. Technical Consultant name
2. Functional Consultant name
3. Objective of the program
4. Transport Request number
5. Date of Creation (or) Change

* **Writing Program:**

Report ZSD\_Customer\_Report -----------title of program

Technical Consultant: Satya

Function Consultant: Raja Comment Section

Objective: To display details of the Customer

Transport Request Number: Not available because we not saved package

Date of Creation: 20-08-2014

\* provide Work Area --------- comment line

DATA: BEGIN OF WA\_KNA1,

CUSTOMER(10) TYPE C,

NAME(20) TYPE C,

CITY(20) TYPE C,

END OF WA\_KNA1.

\* provide execution logic

SELECT KUNNR NAME1 ORT01 FROM KNA1 INTO WA\_KNA1.

\* creating virtual page & printing

WRITE:/10 WA\_KNA1-CUSTOMER,

30 WA\_KNA1-NAME,

60 WA\_KNA1-CITY.

\* repeating SELECT statement

ENDSELECT.

**Note:**

Once program is created

1. Check it for errors by pressing **Ctrl+F2**
2. Save it for backup by pressing **Ctrl+S**
3. Activate it for to store in Database Server by pressing **Ctrl+F3**

**Activation of a Program:**

* Select Ctrl+F3 (or) Candle Icon to activate. You will find an Inactive Objects pop-up then press Enter
* With activation Objects are ultimately stored in Database Server
* With activation Objects can be integrated with each other in SAP
* Select **F8** to execute a program

**21-08-2014**

**Debugging:-** Work Area

* Using debugging you can check (OR) test internal execution of a program
* It is an important tool for testing programs in real time

**Note:**

/H is T Code for debugging

* Select Debugging button (shift+F5)
* By default you will find New Debugger Screen
* Select Debugger--🡪switch to Classical Debugger
* Under Field Names: type wa\_kna1-customer, wa\_kna1-name, wa\_kna1-city & press Enter
* Select F5 for single step debugging
* Select Display List option (ctrl+F12) to view Virtual page details.
* Select F3 again to switch from Virtual page view to Debugger
* Select F7 (or) F8 to execute

**New Debugger:**

* Currently in real time we are working with only New Debugger
* It is providing additional features compare to Old Debugger

**Working:**

* Provide Program:\_\_\_\_\_\_\_ name
* Select Debugging button
* Select Desktop1 tab
* Under Variable: type wa\_kna1-customer, wa\_kna1-name, wa\_kna1-city & press Enter
* Select F5 for single step debugging

**Tips for Writing a Program in Real Time:**

1. Always start a program with Y (or) Z followed by Module name

**Ex-1:** MM module

ZMM\_Material\_Details

**Ex-2:** SD module

ZSD\_Sales\_Register

**Ex-3:** HR module

ZHR\_Employee\_Details

1. Provide Apt Title for a program
2. In real time always save the Object in a Package. But for practicing save Object in Local Object
3. Provide Comments in a program wherever require
4. Try to write your program as Neat as possible
5. Try to write your program in Capital letters
6. Always select Pretty Printer (shift+F1) option after writing a program
7. Always Activate your program once it is created

**Note:**

Only activated programs can be debugged

Only activated programs can be integrated in SAP

Only activated programs can be transported between clients

1. Always Debug your program once it is created for testing
2. Never delete section of a program always Comment it. For commenting select that section with mouse pointer & press Ctrl+< similarly to document select the same section with mouse pointer & press Ctrl+>
3. For Searching a specific keyword (or) ABAP statement select Find option in Standard Tool Bar then provide fine & press Enter
4. For Replacing ABAP keywords (or) statements select Ctrl+F you will find a pop-up Find What: KNA1 select Replace button provide Replace With: LFA1 select Replace All
5. To open new application by closing existing one type /NTCode Ex: /NSE38
6. To open sessions simultaneously type /OTCode Ex: /OSE11 in Command prompt

**Note:**

In SAP you can open maximum 6 sessions simultaneously

1. To logoff SAP type /NEX in Command prompt

**INTERNAL TABLES**

**2. Internal Table:**

* Internal table is a temporary table created in RAM in Application Server
* It is created & filled with data during run time (execution time)
* Once execution is performed it is rolled out (or) discarded

**Note:**

Use internal table for storing multiple records during run time

**Difference between Database table & Internal table:**

**Database Table** **Internal Table**

1. It is created in Database Server ------- 1. It is created in Application Server
2. It holds data permanently ------- 2. It holds data temporarily
3. It holds only specific type of data ----- 3. It holds data of different tables of at one place

**Que:** What is exact purpose of internal table? Do you think should be created in ABAP

Programming?

**Ans:** In real time we are creating Internal table for storing different types of application data

from different tables at one place

Every Internal table is having 2 parts

They are:

1. Internal Table Body
2. Header Line

**1. Internal Table Body:**

The name Internal table itself specifies body of Internal table & it holds multiple records

**2. Header Line:**

It is a default Work Area & holds single record

**Note:**

Header Line is system defined Work Area & Work Area is user defined Header Line

**Syntax:**

DATA: BEGIN OF it\_kna1 OCCURS 0,

customer(20) TYPE C,

name(20) TYPE C,

city(20) TYPE C,

END OF it\_kna1.

**Note:**

1. With above syntax an Internal table it\_kna1 is crated in Application Server
2. You can provide any name. But in real time Internal table should always begins with it\_ (or) i\_ followed by table name
3. With BEGIN option Header Line is created & OCCURS option Body is created
4. 0, 1, 2, 3, …..9 is called Size Category
5. Both Header Line & Body name is same (it\_kna1)

**Flow of Data in Internal Table:**

1. Database Table ----🡪 Body---🡪 Header Line--🡪 Virtual Page ---🡪 Presentation Server
2. Database Table ---🡪 Body---🡪 Presentation Server

**22-08-2014**

**Reading Data from Body to Work Area (or) Header Line:**

1. LOOP ……….. ENDLOOP
2. READ keywords

**1. LOOP ……… ENDLOOP:**

It reads multiple records (record by record) from Body to Header Line (or) Work Area

**Syntax:**

LOOP AT it[INTO wa]

[FROM m][TO n]

[where codition].

…………

…………

ENDLOOP.

* **LOOP AT it:**

Reads multiple records (record by record) from Body to Header Line

**Note:**

Internally system converts LOOP AT it as LOOP AT it INTO it.

**Working:**

* Go to SE38
* Program: ZSD\_LOOPENDLOOP
* Select Create option
* Title; Internal Table
* Type: Executable Program
* Select Save option
* Leave Package:\_\_\_\_\_\_\_\_\_\_ blank
* Select Local Object option

**Pg)** REPORT ZSD\_LOOPENDLOOP.

\* provide internal table

DATA: BEGIN OF it\_kna1 OCCURS 0,

customer(20) TYPE C,

name(20) TYPE C,

city(20) TYPE C,

END OF it\_kna1.

\* provide extraction logic

SELECT kunnr name1 ort01 FROM kna1 INTO TABLE it\_kna1.

\* Appling processing logic

LOOP AT it\_kna1 INTO it\_kna1.

WRITE:/10 it\_kna1-customer,

30 it\_kna1-name,

60 it\_kna1-city.

ENDLOOP.

**Note;**

Activate the program then select F8 to execute the program

* Debugging: Internal Table
* Go to SE38
* Program: \_\_\_\_\_\_ which you want to debug
* Select Debugging button
* Select Tables tab (starting from 6th option)
* Table: it\_kna1 & press Enter

**Note:**

You will find a Cap symbol which indicates Header Line (holds only single record)

Select F5 for single step debugging

**Note:**

1. Table keyword will transfer the data directly to Internal Table to Body
2. Always use Table keyword in select statement if you are working with Internal Tables
3. As long as you are in loop system will read next record in-line in an Internal Table. Once the loop is terminated again the loop starts reading from 1st line

* **LOOP AT it INTO wa:**

Reads multiple records (record by record) from Body to Work Area

**Working:**

* Go to SE38
* Program: ZSD\_INTERNAL\_WORKAREA
* Select Create option
* Title: Internal Table & Work Area
* Type: Executable Program
* Select Save option
* Leave Package:\_\_\_\_\_\_\_\_\_\_ blank
* Select Local Object option

**Pg)** REPORT ZSD\_ INTERNAL\_WORKAREA.

\* provide internal table & work area

TYPES: BEGIN OF ty\_kna1,

customer (20) TYPE C,

name(20) TYPE C,

city(20) TYPE C,

END OF ty\_kna1.

DATA: wa\_kna1 TYPE ty\_kna1,

it\_kna1 TYPE TABLE OF ty\_kna1.

\* provide extraction logic

SELECT kunnr name1 ort01 FROM kna1 INTO TABLE it\_kna1.

\* Appling processing logic

LOOP AT it\_kna1 INTO wa\_kna1.

WRITE:/10 wa\_kna1-customer,

30 wa\_kna1-name,

60 wa\_kna1-city.

ENDLOOP.

**Note:**

1. Using TABLE OF you can create your own body
2. Using TYPES keyword Structures are created in ABAP programming
3. In ABAP programming always create Objet (Work Area & Body) based on Structures
4. Avoid creating Internal Table with OCCURS clause

* **LOOP AT it INTO wa FROM m TO n:**

Reads multiple records from Body to Work Area based on given condition

**Example:**

1. LOOP AT it INTO wa FROM 100.
2. LOOP AT it INTO wa FROM TO 50.
3. LOOP AT it INTO wa FROM 100 TO 200.

**23-08-2014**

**2. READ:**

It reads a single record from Body to Header Line (or) Work Area

**Syntax:**

READ TABLE it[INTO wa]

[INDEX n][WITH KEY keyexpression]

[TRANSPORTING fieldname]

[Binary Serach]

* **READ TABLE it:**

Reads a single record from Body to Header Line

**Ex:** READ TABLE it INTO it.

* **READ TABLE it INTO wa INDEX n:**

It reads a specific record (Line Number) from Body to Work Area

**Ex:** READ TABLE it INTO wa INDEX 5.

* **READ TABLE it INTO wa WITH KEY keyexpression:**

It reads a specific record which matches a given condition in Work Area

**Ex:** READ TABLE it INTO wa WITH KEY customer=wa1-customer

* **READ TABLE it INTO wa WITH KEY keyexpression TRANSPORTING <field1> <field2>……….:**

**Ex:** READ TABLE it INTO wa WITH KEY customer=wa1-customer TRANSPORTING city (it will shows only city field)

**Binary Search:**

It reads data from Body to Work Area based on binary search algorithm

**Note:**

1. Use binary search if Internal table is larze
2. Data should be sorted before Appling binary search algorithm
3. There should be no duplicate records in Internal Table
4. Binary search algorithm improves performance of programming

**SELECTION-SCREEN:**

Using Selection Screen you can create a selection screen based on given requirement (Input Values, Radio Buttons, Check Boxes…………..)

**Input Values:**

1. You can create with ------ Select-Options
2. You can create with ------ Parameters
3. You can create with ------ Ranges

**1. SELECT-OPTIONS:**

Using select-options you can create a selection screen with range of input values

**Syntax:**

SELECT-OPTIONS: s\_kunnr for kna1-kunnr.

With above syntax system will create a selection screen in the following manner

S\_kunnr 100 TO

200

100

With above syntax system will create an implicit Internal Table with following fields

|  |  |  |  |
| --- | --- | --- | --- |
| Low | High | Sign | Option |
| 100 | 200 | I (including) | BT (between) |

**Note:**

The default comparison operator for select-options in ‘IN’

**Pg)** \* provide tables work area

TABLES kna1.

\* provide selection screen

SELECTION-SCREEN: BEGIN OF BLOCK b1 WITH FRAME TITLE TEXT-009

SELECT-OPTIONS: s\_kunnr FOR kna1-kunnr.

SELECTION-SCREEN: END OF BLOCK b1.

\* provide objects

TYPES: BEGIN OF ty\_kna1,

customer(20) TYPE C,

name(20) TYPE C,

city(20) TYPE C,

END OF ty\_kna1.

DATA: wa\_kna1 TYPE ty\_kna1,

It\_kna1 TYPE TABLE OF ty\_kna1.

\* provide extraction logic

SELECT kunnr name1 ort01 FROM kna1 INTO TABLE it\_kna1 WHERE

kunnr IN s\_kunnr.

\* Appling processing logic

LOOP AT it\_kna1 INTO wa\_kna1.

WRITE:/10 wa\_kna1-customer,

30 wa\_kna1-name,

60 wa\_kna1-city.

ENDLOOP.

**Text Elements:**

Using text elements (text-000 to text-009) you can provide field labels in selection screen

* Double click on Text-000
* You will find a pop-up Yes
* Provide Text: Enter Customer Code
* Select Selection Texts tab
* Provide Text: Customer Code
* Activate the screen (Ctrl+F3)
* Go to back (F3)

**Note:**

Using Tables work area system will create a Work Area with all the fields of kna1 table. In programming it is mandatory to declare for declaring select-options

**Syntax for Creating Block:**

SELECTION-SCREEN: BEGIN OF BLOCK <block name> FRAME TITLE <title>.

……………..

……………….

SELECTION-SCREEN: END OF BLOCK <block name>.

**25-08-2014**

**Other SELECT-OPTIONS keywords:**

1. OBLIGATORY
2. NO INTERVALS
3. NO-EXTENSION
4. NO-DISPLAY

**i) OBLIGATORY:**

Using this addition you can make selection screen fields as Mandatory

**Syntax:**

SELECT-OPTIONS: s\_kunnr FOR kna1-kunnr OBLIGATORY.

**ii) NO INTERVALS:**

Using no intervals you can remove high option from selection screen input field with select-options

**Syntax:**

SELECT-OPTIONS: s\_kunnr FOR kna1-kunnr NO INTERVALS.

**iii) NO-EXTENSION:**

With this addition Multiple Selections option is removed from selection screen with select-options

**Syntax:**

SELECT-OPTIONS: s\_kunnr FOR kna1-kunnr NO-EXTENSION.

**iv) NO-DISPLAY:**

With this addition selection screen input field will be hidden mode

**Syntax:**

SELECT-OPTIONS: s\_kunnr FOR kna1-kunnr NO-DISPLAY.

**Working with Multiple Selection:**

**Step-1:** providing Random Values using Multiple Selection

* Execute your program
* Select Multiple Selection option
* Under Single Value provide values
* Select Clock symbol (F8)
* Execute (F8)

**Step-2:** Providing Multiple Ranges using Multiple Selections

* Execute your program
* Select Multiple Selection option
* Under Select Ranges provide multiple ranges
* Select Clock symbol (F8)
* Execute (F8)

**Step-3:** Providing Excluding Random Values

* Execute your program
* Select Multiple Selection option
* Under Exclude Single Value provide values
* Select Clock symbol (F8)
* Execute (F8)

**Step-4:** Providing Excluding Ranges

* Execute your program
* Select Multiple Selection option
* Under Exclude Ranges provide values
* Select Clock symbol (F8)
* Execute (F8)

**Step-5:** Coping the Data from Excel sheet to Multiple Seletion

* Copy the data from Excel sheet (Ctrl+C)
* Select Multiple Selections option
* Press Ctrl+V (or) select second last button

**2. PARAMETERS:**

Using parameters you can create a selection screen with single input value

**Syntax:**

PARAMETERS p\_kunnr TYPE kunnr. (or)

PARAMETERS p\_kunnr LIKE kna1-kunnr.

**Note:**

1. With above syntax system will define (or) create a selection screen in the following manner

P\_kunnr

1. With above syntax no Internal Table is created by system
2. The default comparison operator for PARAMETERS is ‘EQ’

**Pg)** \* provide tables work area

TABLES kna1.

\* provide selection screen

SELECTION-SCREEN: BEGIN OF BLOCK b1 WITH FRAME TITLE TEXT-009

PARAMETERS p\_kunnr TYPE kunnr.

SELECTION-SCREEN: END OF BLOCK b1.

\* provide objects

DATA: BEGIN OF wa\_kna1,

customer(20) TYPE C,

name(20) TYPE C,

city(20) TYPE C,

END OF wa\_kna1.

\* provide extraction logic

SELECT SINGLE kunnr name1 ort01 FROM kna1 INTO wa\_kna1

WHERE kunnr EQ p\_kunnr.

\* Appling processing logic

WRITE:/10 wa\_kna1-customer,

30 wa\_kna1-name,

60 wa\_kna1-city.

ENDSELECT.

**Note:**

1. If you are not providing SINGLE addition with PARAMETERS for fletching a single record then it is mandatory to provide ENDSELECT
2. In real time never use SELECT with ENDSELECT (it reduces performance of the program)

**3. RANGES:**

It work similar to SELECT-OPTIONS

**Syntax:**

RANGES s\_kunnr FOR kna1-kunnr.

**Note:**

In new versions RANGES keyword is not allowed (it is out dated)

**Other Important Keywords for Internal Table:**

**1. APPEND keyword:**

It appends record from Work Area to Body

**Syntax:**

APPEND wa TO it.

**Pg)** DATA: BEGIN OF wa,

material(10) TYPE C,

quan(10) TYPE C,

END OF wa.

DATA it LIKE TABLE OF wa.

\* fill data into it

wa-material=’F001’.

wa-quan=50.

APPEND wa TO it.

wa-material=’F002’.

wa-quan=70.

APPEND wa TO it.

wa-material=’F002’.

wa-quan=100.

APPEND wa TO it.

wa-material=’F001’.

wa-quan=80.

APPEND wa TO it.

LOOP AT it INTO wa.

WRITE:/10 wa-material,

30 wa-quan.

ENDLOOP.

**2. SORT keyword:**

It sorts Internal Table Body data based on ascending (or) descending

**Syntax:**

SORT <it> BY <field1> ASCENDING (or) DESCENDING

**Note:**

The default order of sorting is ASCENDING

**Pg)** DATA: BEGIN OF wa,

material(10) TYPE C,

quan(10) TYPE C,

END OF wa.

DATA it LIKE TABLE OF wa.

\* fill data into it

wa-material=’F001’.

wa-quan=50.

APPEND wa TO it.

wa-material=’F002’.

wa-quan=70.

APPEND wa TO it.

wa-material=’F002’.

wa-quan=100.

APPEND wa TO it.

wa-material=’F001’.

wa-quan=80.

APPEND wa TO it.

SORT it BY material ASCENDING.

LOOP AT it INTO wa.

WRITE:/10 wa-material,

30 wa-quan.

ENDLOOP.

**3. DELETE ADJACENT DUPLICATES keyword:**

Deletes adjacent duplicates from Internal Table

**Syntax:**

DELETE ADJACENT DUPLICATES FROM it COMPARING <filed name> WHERE codition.

**Note:**

Before Appling above keyword it is mandatory to sort Internal Table

**Pg)** DATA: BEGIN OF wa,

material(10) TYPE C,

quan(10) TYPE C,

END OF wa.

DATA it LIKE TABLE OF wa.

\* fill data into it

wa-material=’F001’.

wa-quan=50.

APPEND wa TO it.

wa-material=’F002’.

wa-quan=70.

APPEND wa TO it.

wa-material=’F002’.

wa-quan=100.

APPEND wa TO it.

wa-material=’F001’.

wa-quan=80.

APPEND wa TO it.

SORT it BY material.

DELETE ADJACENT DUPLICATES FROM it COMPARING material.

LOOP AT it INTO wa.

WRITE:/10 wa-material,

30 wa-quan.

ENDLOOP.

**26-08-2014**

**4. COOLECT Keyword:**

It provides summarized representation of data in an Internal Table

**Note:**

COLLECT checks the data based on Key Field (Character Field) in an internal table Body, if the record is not existing it will performs Append job, if it is already existing it will Add (Collects) Numeric Values based on Character Field inside the Body.

**Syntax:**

COLLECT wa INTO it.

**Pg)** DATA: BEGIN OF wa,

material(20) TYPE C,

quan TYPE I,

END OF wa.

DATA: it LIKE TABLE OF wa,

it1 LIKE TABLE OF wa.

wa-material = ‘F001’.

wa-quan = 50

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 150

APPEND wa TO it.

wa-material = ‘F001’.

wa-quan = 550

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 350

APPEND wa TO it.

LOOP AT it INTO wa.

COLLECT wa INTO it1.

ENDLOOP.

LOOP AT it1 INTO wa.

WRITE:/10 wa-material,

30 wa-quan.

ENDLOOP.

**Ques-1:** What is the difference between APPEND & COLLECT?

**Ans:** APPEND keyword appends the record from Work Area to Body (always to next line

in Internal Table). COLLECT keyword performs append job if the record fails to exist

in the Body. If already exists it adds all Numeric Values on Right Side of Character

Field inside the Body

**Ques-2:** Whenever you will use COLLECT keyword in real time?

**Ans:** Whenever I will get a requirement for providing consolidated totals based on

customers, vendors, materials……..etc I will use COLLECT keyword

**Ques-3:** Suppose there are more than one Character Fields then what is the rollback of

COLLECT keyword?

**Ans:** If there are two (or) more than two Character Fields it will Collect (add) the data only

when both the Character Fields are same

**5. MODIFY keyword:**

This keyword modifies Internal Table’s Body data based on given requirement

**Syntax:**

MODIFY it FROM wa [TRANSPORTING <field1> <filed2> ……..]

[INDEX n]

[WHERE condition]

**Pg)** DATA: BEGIN OF wa,

material(20) TYPE C,

quan TYPE I,

unit(2) TYPE C,

END OF wa.

DATA it LIKE TABLE OF wa.

wa-material = ‘F001’.

wa-quan = 50

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 150

APPEND wa TO it.

wa-material = ‘F001’.

wa-quan = 550

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 350

APPEND wa TO it.

LOOP AT it INTO wa.

wa-unit = ‘kg’.

MODIFY it FROM wa TRANSPORTING unit.

ENDLOOP.

LOOP AT it INTO wa.

WRITE:/10 wa-material,

30 wa-quan,

60 wa-unit.

ENDLOOP.

**Note:**

1. Always provide MODIFY with in LOOP………..ENDLOOP of the Internal Table which you want to modify
2. If you are providing MODIFY outside LOOP……..ENDLOOP then INDEX addition (or) WHERE condition is mandatory

**Ques-1:** What is the difference between APPEND & MODIFY?

**Ans:** APPEND always appends record to Internal Table (if record exists (or) not). MODIFY

modifies (adds field value) only if records exists in an Internal Table

**Ques-2:** When you will prefer MODIFY keyword in ABAP programming?

**Ans:** When all the data in an Internal Table is filled and a new filed value is required to be

added in Internal Table

**6. Deleting Data from Internal Table:**

1. FREE
2. REFRESH
3. CLEAR
4. DELETE

**i) FREE:**

It deletes Internal Table’s Body data along with memory

**Syntax:**

FREE it. (where it is Internal Table name)

**ii) REFRESH:**

It deletes Internal Table’s Body data but memory will not be affected

**Syntax:**

REFRESH it.

**iii) CLEAR:**

It deletes 1. Body data------- CLEAR it[ ].

2. Work Area data------- CLEAR wa.

3. Ordinary Variable data --------- CLEAR total.

**Note:**

CLEAR it[ ] is applicable only where Internal Table is created with OCCURS clause

**Example:**

1. CLEAR it[ ]. ------- deletes Header Line’s data
2. CLEAR it. -------- deletes Body’s data

**iv) DELETE:**

It deletes Body data based on requirement

**Syntax:**

DELETE it [INDEX n]

[FROM m] [TO n]

[WHERE condition]

**Example:**

1. LOOP AT it INTO wa.

DELETE it.

ENDLOOP.

1. DELETE it INDEX 5.
2. DELETE it FROM 10 TO 20.
3. DELETE it WHERE material = ‘F001’

**27-08-2014**

7. **DESCRIBE keyword:**

It returns the Count of number of records in an Internal Table

**Syntax:**

DATA n TYPE I.

DESCRIBE TABLE it LINES n.

**Example:**

DATA: BEGIN OF wa,

material(20) TYPE C,

quan TYPE I,

END OF wa.

DATA it LIKE TABLE OF wa.

DATA n TYPE I.

wa-material = ‘F001’.

wa-quan = 50.

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 150.

APPEND wa TO it.

wa-material = ‘F001’.

wa-quan = 250.

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 350.

APPEND wa TO it.

DESRIBE TABLE it LINES n.

WRITE n.

8. **Coping Data between Work Areas:**

i) **MOVE keyword:**

It transfers (or) moves data from one Work Area to another Work Area if the Structures are same.

**Example:**

MOVE wa TO wa1.

ii) wa1-material = wa-material.

wa1-quan = wa-quan.

iii) **MOVE-CORRESPONDING keyword:**

It transfers (or) moves data from one Work Area to another Work Area if the Structures of both the Work Areas are different

**Example:**

MOVE-CORRESPONDING wa TO wa1.

**Note:**

In real time avoid using MOVE-CORRESPONDING

instead use wa1-material = wa-material.

wa1-quan = wa-quan.

9. **Coping Data between Internal Tables:**

i) LOOP AT it INTO wa.

APPEND wa TO it1.

ENDLOOP.

ii) If Internal Tables are created with OCCURS clause

it1[ ] = it[ ].

iii) **APPEND LINES keyword:**

It performs,

* Copy the hold data from one Internal Table to another Internal Table (If there is no data in Target Internal Table)
* Appends the records at the end of another Internal Table (If Target Internal Table is already having some records)
* It adds the records from Source Internal Table to Target Internal Table based on given condition

**Syntax:**

APPEND LINES OF it [FROM m] [TO n] TO it1.

**Example:**

1. APPEND LINES OF it TO it1.

2. APPEND LINES OF it FROM 2 TO 3 TO it1.

**Pg)** \* creating work area

DATA: BEGIN OF wa,

material(20) TYPE C,

quan TYPE I,

END OF wa.

\* creating bodies

DATA: it LIKE TABLE OF wa,

it1 LIKE TABLE OF wa.

\* appending data from work area to bodies

wa-material = ‘F001’.

wa-quan = 50.

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 150.

APPEND wa TO it.

wa-material = ‘F001’.

wa-quan = 250.

APPEND wa TO it1.

wa-material = ‘F002’.

wa-quan = 350.

APPEND wa TO it1.

\* coping data one body to another body

APPEND LINES OF it TO it1.

\* displaying data of target body

LOOP AT it1 INTO wa.

------------- CLEAR wa.

WRITE:/10 wa-material,

30 wa-quan.

------------- CLEAR wa.

ENDLOOP.

**Ques:** What will happened if CLEAR is provided before WRITE & the same CLEAR is provided after WRITE?

**Ans:** If CLEAR is provided before WRITE statements it will clear Work Area data so nothing is transformed from Work Area to Virtual page (Hence no output)

If CLEAR is provided after WRITE statements data is transformed from Work Area to Virtual page then Work Area is cleared (Hence you will find output list)

**28-08-2014**

**Control Break Statements:-** (Events)

1. AT FIRST
2. AT LAST
3. AT NEW
4. AT END OF
5. ON CHANGE OF

Using these statements you can control the flow of ABAP programming statement with in the LOOP……………ENDLOOP

**AT FIRST:**

It triggers during first loop iteration (first loop pass)

**Syntax:**

AT FIRST.

------------

-----------

ENDAT.

**Note:**

Use AT FIRST for providing Headers in ABAP Report

**AT LAST:**

It triggers for last line of an Internal Table (or) It triggers during last loop iteration

**Syntax:**

AT LAST.

------------

-----------

ENDAT.

**Note:**

Use AT LAST for printing Grand Totals in a Report

**AT NEW:**

It triggers for a group of similar contents (based on Character Value) in an Internal Table. (or) It triggers whenever there is a change in field values in an Internal Table based on Character Value

**Syntax:**

AT NEW <field name>.

------------

-----------

ENDAT.

**Note:**

Use AT NEW for printing unique values for key fields in an Internal Table

**AT END OF:**

It will triggers the end of group of similar values in an Internal Table based on Character Field

**Syntax:**

AT END OF <field name>.

------------

-----------

ENDAT.

**Note:**

Use AT END OF for returning grand totals in an Internal Table

**ON CHANGE OF:**

It works similar to AT NEW

**Syntax:**

ON CHANGE OF <field name>.

------------

-----------

ENDON.

**Note:**

1. Due to some draw backs currently we are not using ON CHANGE OF
2. You should provide above Control Break Statements within the LOOP……..ENDLOOP
3. Don’t provide any conditional statements (IF, CASE,…….etc) with in Control Break Statements

**Requirement:**

Material Details Report

--------- AT FIRST

The Material Cod is: F001

50 5000

100 10000

The Total Quantity is: 150 AT NEW

The Total Amount is: 15000

The Material Cod is: F002

70 6000

100 12000

The Total Quantity is: 170 AT END OF

The Total Amount is: 18000

The Grand Total Quantity is: 320

The Grand Total Amount is: 33000 ---------AT LAST

**Solution:**

\* creating work area

DATA: BEGIN OF wa,

material(10) TYPE C,

quan TYPE I,

amount TYPE I,

END OF wa.

\* creating internal table

DATA it LIKE TABLE OF wa.

\* filling data into it

wa-material = ‘F001’.

wa-quan = 50.

wa-amount = 5000.

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 70.

wa-amount = 6000.

APPEND wa TO it.

wa-material = ‘F001’.

wa-quan = 100.

wa-amount = 10000.

APPEND wa TO it.

wa-material = ‘F002’.

wa-quan = 100.

wa-amount = 12000.

APPEND wa TO it.

\* sorting it by material

SORT it BY material.

\* appling Control Break Statements

LOOP AT it INTO wa.

AT FIRST.

WRITE:/50 ‘Materials Details Report’ COLOR 1.

SKIP.

ULINE.

ENDAT.

AT NEW material.

WRITE:/ ‘The Material Code is:’, wa-material.

SKIP.

ENDAT.

WRITE:/10 wa-quan,

30 wa-amount.

AT END OF material.

SUM.

WRITE:/ ‘The Total Quantity is:’, wa-quan,

/ ‘The Total Amount is:’, wa-amount.

ULINE.

ENDAT.

AT LAST.

SUM.

WRITE:/ ‘The Grand Total Quantity is:’, wa-quan,

/ ‘The Grand Total Amount is:’, wa-amount.

ULINE.

ENDAT.

ENDLOOP.

**30-08-2014**

**Ques:** What is the difference between SUM & COLLECT keywords?

**Ans:** COLLECT keyword is applicable LOOP……….ENDLOOP without usage of Control

Break Statements

COLLECT keyword performs totals inside internal table body based on one (or) more

than one Character Values

SUM keyword performs totals only one Single Character Value for Control Break

Statements places the values in Work Area

COLLECT keyword works outside LOOP………ENDLOOP also where as SUM

keyword can be applied only with in LOOP………ENDLOOP

**INSERT keyword:**

Inserts Work Area data at a specific index position inside Internal Table Body

**Syntax:**

INSERT <work area name> INTO <internal table name> INDEX <line number>.

**Syntaxes for Internal Tables:**

1. NO DATA: BEGIN OF <internal table name> OCCURS 0,

material(10) TYPE C,

quan TYPE I,

END OF <internal table name>.

**Note:**

Both Header Line & Body created

1. DATA <internal table name> TYPE <database table name> OCCURS 0.

**Note:**

Only Body created based on all fields of database table

1. DATA <internal table name> TYPE <database table name> OCCURS 0 WITH HEADERLINE.

**Note:**

Both Body & Header Line created

1. TYPES: BEGIN OF ty\_kna1,

customer(10) TYPE C,

name(10) TYPE C,

city(10) TYPE C,

END OF ty\_kna1.

DATA: <work area name> TYPE <structure name>,

<body name> TYPE TABLE OF <structure name>.

**Note:**

Work Area & Body created based on Sturcture

1. TYPES: BEGIN OF ty\_kna1,

customer(10) TYPE C,

name(10) TYPE C,

city(10) TYPE C,

END OF ty\_kna1.

DATA <body name> TYPE TABLE OF <structure name> INITIAL SIZE 0.

**Note:**

The above Body is created when you transfer whole Body data to another Body

**Example:** it1[ ] = it[ ].

**System Fields for Internal Table:**

SAP is providing system variables maintained in a structure SYST

There are 171 system variables in the structure & all system variables start with a prefix SY

**Example:**

1. SY-DATUM----returns current system data
2. SY-UNAME----returns current login user name
3. SY-UZEIT------returns current time
4. SY-DATUM----returns current system data
5. SY-LANGU----returns current login language
6. SY-TABIX-----returns current LOOP pass
7. SY-TOCCU----returns OCCURS value
8. SY-TLENG----returns size of internal table
9. SY-TFILL----returns count of number of records in an internal table (it works like

DESCRIBE keyword)

**Example:**

\* creating internal table

DATA: BEGIN OF it OCCURS 0,

material(10) TYPE C,

quan TYPE I,

amount TYPE I,

END OF wa.

\* creating variable

DATA n TYPE I.

\* filling data into it

wa-material = ‘F001’.

wa-quan = 50.

wa-amount = 5000.

APPEND it TO it.

wa-material = ‘F002’.

wa-quan = 70.

wa-amount = 6000.

APPEND it TO it.

wa-material = ‘F001’.

wa-quan = 100.

wa-amount = 10000.

APPEND it TO it.

wa-material = ‘F002’.

wa-quan = 100.

wa-amount = 12000.

APPEND it TO it.

\* counting number of records in internal table

DESCRIBE TABLE it LINES n.

WRITE:/ ‘size of it:’, SY-TLENG,

/ ‘number of records:’, SY-TFILL,

/ ‘OCCURS value:’, SY-TOCCU.

**DML Operations on Database Table:**

1. INSERT
2. UPDATE
3. DELETE

**Inserting Data into Database Table:**

**MODIFY keyword:**

It will insert records in database table if they are not existing & performs modify operation if they are already existing

**Syntax:**

1. MODIFY <database table name> FROM TABLE <internal table name>
2. MODIFY <database table name> FROM TABLE <work area name>

**Example-1:**

TABLES kna1.

DATA: wa TYPE kna1,

It TYPE TABLE OF kna1.

wa-kunnr = ‘0000009083’.

wa-name1 = ‘rahul’.

wa-ort01 = ‘mumbai’.

wa-pstlz = ‘400002’.

APPEND wa TO it.

MODIFY kna1 FROM TABLE it.

WRITE:/ ‘ number of records inserted:’, SY-DBCNT.

**Example-2:**

LOOP AT it INTO wa.

MODIFY kna1 FROM wa.

ENDLOOP.

**UPDATE keyword:**

Updates specific fields based on given requirement

**Syntax:**

UPDATE <database table name> SET field1 = wa-filed1, field2 = wa-field2……….

………….. WHERE <codition>.

**Example:**

LOOP AT it INTO wa.

UPDATE kna1 SET ort01 = wa-orto1, pstlz = wa-pstlz WHERE kunnr = wa-kunnr.

ENDLOOP.

**DELETE keyword:**

Deletes database tables data based on given requirement

**Syntax:**

1. DELETE <database table name> FROM TABLE <internal table name>.
2. DELETE FROM <database table name> WHERE <condition>

**Example-1:**

DELETE kna1 FROM TABLE it.

WRITE:/ ‘number of records deleted:’, SY-DBCNT.

**Example-2:**

LOOP AT it INTO wa.

DELETE FROM kna1 WHERE kunnr = wa-kunnr.

ENDLOOP.

**Example-3:**

DELETE FROM kna1 WHERE kunnr = ‘0000009083’.

**INSERT keyword:**

Inserts data into database table

**Syntax:**

INSERT INTO <database table name> VALUES <work area name>.

**Types of Internal Tables:**

1. Standard Table
2. Hashed Table
3. Sorted Table
4. Index Table

**Note:**

The default Internal Table is Standard

**Assignments-1:**

Students Details Report

Sno Sname Phy Chem Maths Total

100 satya 70 60 90 220

101 pruthvi 80 70 60 210

102 raj 90 80 80 250

240 210 230 680

**Assignment-2:**

Material Quan

|  |  |
| --- | --- |
| F001 | 50 |
| F001 | 100 |
| F002 | 70 |
| F002 | 150 |

F001 150

F002 220

**Hint:**

1. Create 3 Internal Tables
2. Use DELETE ADJACENT DUPLICATE keyword
3. Use CLEAR keyword

**01-09-2014**

**Data Dictionary**

It is also called ABAP Dictionary

It is a repository of data

T Code for Data Dictionary is SE11

**Objects in Data Dictionary:**

1. Database Table
2. Structure
3. View
4. Data Element
5. Domain
6. Table Type
7. Type Group
8. Lock Object
9. Search Help

**Database Table:**

It is a set of fields which holds data persistently

**Data Element:**

Using Data Element you can provide field documentation for a specific field

Conceptually Data Element = Field Lable + Domain

**Domain:**

It specifies the technical attributes of a field (Data Type & Field Size)

**Ques-1:** Have ever created a Zee table in real time?

**Ans:** Yes

**Ques-2:** SAP is providing so many tables then why you created a table?

**Ans:** SAP is not providing table for Allowances,

Similarly SAP is not providing a table for Transportation Details.

**Example:** Delivery challana number, date, truck number……….etc

Similarly in real time we have to create a table for Customer mail ids for sending invoice details…….etc

**Table Creation:**

* Go to SE11
* Provide Title: ZMM\_TR
* Select Create option
* Provide Short Description: Truck Entry Details

**Note:**

In real time we are working with application data.

Application data is 2 types

1. Master Data
2. Transactional Data

**Master Data:**

It is created only once in Client Database & modified very rarely

**Example:**

Bank Master Data, Customer Master Data, Vendor Master Data, Employee Master Data…etc

**Transactional Data:**

It is always created based on Master Data

It is modified frequently in SAP database

**Example:**

Purchase Order Data, Sales Order Data………..etc

**Delivery Class:**

It specifies the type of data a table can hold

A-------specifies Application Data

* Provide Table View Maintenance: Display Maintenance Allowed (both display & create entries is allowed)
* Select Fields tab

Field key Data Element

TRNO ZTR\_DAT

* Double click on ZTR\_DAT
* You will find a message pop-up select Yes option
* Select Local Object option
* Again you will find a warning: application table should be client specific the press Enter
* You will find a pop-up: create the data element then click Yes
* Provide Short Description: Truck Number
* Provide Domain: ZTR\_Domain
* Double click on Domain Name (ZTR\_Domain)
* Click Yes
* Select Local Object

**Note:**

$TMP is a default package for non-transportable object (local object)

* You will find a pop-up: create the domain click Yes
* Provide Short Description: Domain for Truck Field
* Data Type: CHAR
* No. Characters: 10
* Activate the domain (Ctrl+F3)
* Select Local Object
* Go to back (F3)
* Select Field Label tab

Length field lable

Short 10 trno

Medium 20 Truck Number

Lang 40 Truck Number

Heading 50 Truck Number

* Activate data element (Ctrl+F3)
* Go to back (F3)

Field key data element

Ddate LFDAT

Driver ZDR\_DAT

* Double click on ZDR\_DAT
* Save Yes
* You will find a warning pop-up: Enter
* You will find a pop-up: create data element Yes
* Provide Short Description: Driver Name
* Provide Domain: CHAR20
* Select Field Label tab
* Provide Short Description: Driver Name

Length field lable

Short 10 Dname

Medium 20 Driver Name

Lang 40 Driver Name

Heading 50 Driver Name

* Activate data element (Ctrl+F3)
* Go to back (F3)
* Select Technical Settings button
* Provide Data Class: APPL0 (APPL0---for master data & APPL1---for transactional data)
* Provide Size Category: 0 (it means table holds up to 8000 records)
* Select Save option (Ctrl+s)
* Select back (F3)
* Activate Table (Ctrl+F3)
* During activation you will find a warning pop-up select Yes
* Ignore the warnings & go back (F3)

**Step-1:**

**Navigation for Inserting Records into a Table:**

* Select Utilites---🡪Table Contents---🡪Create Entries

TRNO:\_\_\_\_\_\_\_

Ddate:\_\_\_\_\_\_\_\_\_

Driver:\_\_\_\_\_\_\_\_\_

* Save (Ctrl+S)
* Select Reset button for new record

**Step-2:**

**Navigation for Viewing Records in a Table:**

* Select Utilities---🡪Table Contents---🡪Display
* Select Clock symbol (F8)

**02-09-2014**

**Structure:**

It is a group of fields

Physically it will never hold any data

Structure can be reuse in other programs & other tables

**Differences between Structure & Table:**

**Structure** **Table**

1. It is a collection of fields & it 1. It is a Structure & hold data persistently

Never hold data

1. It can be reuse in other tables 2. It cannot be reuse

**Difference between .INCLUDE & APPEND STRUCTURE:**

Using .include you can add a Structure in N no. of tables

.include is best for adding Structure in ZEE tables

Using Append Structure you can add a Structure only in a single table

Append Structure is best for adding Structure in Standard Tables

**Structure Creation:**

* Go to SE11
* Select Data Type: ZMM\_Order
* Select Create option
* You will find a pop-up select Structure option & press Enter
* Provide Short Description: Order Fields
* Provide Component: EBELN

BEDAT

* Provide Component Type: EBELN

BEDAT

* Activate structure (Ctrl+F3)
* Select Local Object
* You will find warnings pop-up then select Yes
* Ignore warnings & go back (F3)

**Navigation for Including a Structure in a Table:**

* Go to SE11
* Provide Database Table: ZMM\_TR
* Select Change option
* Under Field: .include
* Under Data Element: ZMM\_Order
* Activate table (Ctrl+F3)
* Select Contents option
* Select Clock symbol (F8)
* Select SelectAll option (F9)
* Select Change option (F6)
* Add data & save (Ctrl+S)
* Select Next Entry option for next record

**Primary & Foreign key Relationship:**

SAP is a tightly integrated package. In SAP between all the tables there is a relationship called Primary & Foreign key Relationship.

**Condition (or) Pre represents for maintaining a Relationship between Two Tables:**

In SAP if you want to create relationship between two tables there should be Key Field with common data between tables even the fields are not common there should be common data between two tables

**Establishing Relationship between Two Tables:**

* Go to SE11
* Provide Database Table: ZMM\_DCTR
* Select Create option
* Provide Short Description: Delivery Challana Details
* Provide Delivery Class: A
* Provide Table View Maintenance: Display Maintenance Allowed
* Select Fields tab

Field Key Data Element

MANDAT MANDAT

DCNO ZDAT\_DC

* Double click on ZDAT\_DC
* Click on Yes, Local Object & Yes
* Provide Short Description: Data Element for DC
* Provide Domain: CHAR10
* Select Field Label option
* Medium: 20
* Field Label: Delivery Challana
* Activate Field (Ctrl+F3)
* Like add all required fields (DCdate, TRNO)
* Select TRNO field
* Select Foreign Keys option
* You will find a pop-up provide Short Text: Relationship

Check Table: ZMM\_TR

* Select Generate Proposal button
* Select Enter option
* Provide Field: .include
* Data Element: ZMM\_Order (it provides EBLEN, BEDAT fields)
* Select Technical Settings option
* Provide Data Class: APPL1 (for Transactional data)
* Size Category: 0 (for upto 8000 records)
* Select Save option (Ctrl+s)

**Data Class:**

It is a physical storage location (or) table space where tables are stored ultimately after activation. It is of 3 types

1. APPL0 (holds master data tables)
2. APPL1 (holds transactional data tables)
3. APPL2 (holds organizational & customizing data related tables)

**Size Category:**

It specifies the probable space requirement for a table in the database

* Go to back (F3)
* Activate the table (Ctrl+F3)
* Select Content option
* Apply F4 on Truck Number field
* You will find data related to Check Table
* Go to back (F3)
* Select Utilities--🡪Table Contents--🡪Create Entries

**Note:**

In real time we are not creating any relationships between tables

**Ques:** What is the difference between Client Dependant & Client Independent tables?

**Ans:** A table created without MANDT field is Client Independent table. Client Independent tables are available in other clients also with data

A table created with MANDT field is Client Dependant table. Client Dependent tables are available in other clients also without data

**Note:**

In real time it is always advisable to create Client Dependant tables

**View:**

* It is a logical table (virtual table) which is created based on other tables
* It hires Fields & Data from other tables

**Difference between Table & View:**

**Table** **View**

1. Having its own fields & data 1. Always created based on other table
2. It holds either master data (or) 2. It holds both master data & transactional data

Transactional data

**Types of Views:**

1. Database View
2. Projection View
3. Help View
4. Maintenance View

**Database View:**

It is always created based on two (or) more than two tables having relationship between them

**Database View Creation:**

* Go to SE11
* Select View option: ZMM\_DNTR
* Select Create option
* You will find a pop-up select Database View
* Provide Short Description: Database View

Tables: ZMM\_TR

ZMM\_DCTR

* Select Relationships button
* You will find a pop-up double click on Check Box under referenced table heading & press Enter
* Select View Flds option
* Select Tables Fields option
* Double click on ZMM\_TR
* Select all fields & press Enter
* Select Tables Fields option
* Double click on ZMM\_DCTR
* Select DCNO, DCdate & press Enter
* Activate view (Ctrl+F3)

**Testing Database View:**

* Select Contents option
* Select Clock symbol (F8)

**Note:**

View always holds common data (linked data between two (or) more than two tables)

**03-09-2014**

**Projection View:**

It created based on a single table with specific fields

**Projection View Creation:**

* Go to SE11
* Select View option: ZPROJ\_VIEW
* Select Creation option
* You will find a pop-up select Projection View option then press Enter
* Provide Short Description: Projection View
* Provide Basis Table: KNA1
* Select Table Fields tab
* You will find all fields of KNA1 table then select KUNNR, NAME1, ORT01, PSTLZ then press Enter
* Activate view (Ctrl+F3)

**Testing Projection View:**

* Select Contents option
* Click Clock symbol (F8)

**Search Help:**

Using search help you can search application data based on given search criteria

Search help works based on F4 function key

**Note:**

In a table (or) tables there is large volume of application data. Manual search of data will take lot of time. SAP is providing an option called Search Help for searching data as for requirement

**Types of Search Help:**

1. Elementary Search Help
2. Collective Search Help

**Elementary Search Help:**

It is always created based on a single table

**Creation of Elementary Search Help-1:**

* Go to SE11
* Select Search Help: ZELE\_TRSEARCH
* Select Create option
* You will find a pop-up then select Elementary Search Help then press Enter
* Provide Short Description: Elementary Search Help for Truck Details
* Selection Method: ZMM\_TR (table name)

Search Help Parameter Import Export LPOS SPOS

TRNO \_/ \_/ 1 1

DDATE \_/ \_/ 2 2

EBLEN \_/ \_/ 3 3

BEDAT \_/ \_/ 4 4

DRIVER \_/ \_/ 5 5

* Activate search help (Ctrl+F3)

**Testing Search Help:**

* Select F8
* You will find a pop-up then press Enter

**Creation of Elementary Search Help-2:**

* Go to SE11
* Select Search Help: ZELE\_DMSEARCH
* Select Create option
* You will find a pop-up then select Elementary Search Help then press Enter
* Provide Short Description: Elementary Search Help for Delivery Challana Details
* Selection Method: ZMM\_DCTR (table name)

Search Help Parameter Import Export LPOS SPOS

DCNO \_/ \_/ 1 1

DCDATE \_/ \_/ 2 2

* Activate search help (Ctrl+F3)

**Testing Search Help:**

* Select F8
* You will find a pop-up then press Enter

**Collective Search Help:**

It is a group of two (or) more than two Elementary Search Help contained to form a Collective Search Help

**Creation of Collective Search Help:**

* Go to SE11
* Select Search Help: ZCOLL\_TRDM
* Select Create option
* You will find a pop-up then select Collective Search Help option then press Enter
* Provide Short Description: Collective Search Help for DC details
* Select Included Search Help tab
* Provide Search Help: ZELE\_TRSEARCH

ZELE\_DMSEARCH

* Activate search help

**Testing Collective Search Help:**

* Select F8
* You will find a pop-up then press Enter

**Calling a Search Help in an ABAP Program:**

**Syntax:**

PARAMETERS <parameter name> MATCHCODE OBJECT <search help name>

**Example:**

PARAMETERS p\_tr MATCHCODE OBJECT ZELE\_TRSEARCH

**Note:**

In real time calling a search help using MATCHCODE OBJECT is outdated

**Adding a Search Help in a Table:**

* Go to SE11
* Provide Database Table: ZMM\_TR
* Select Change option
* Place cursor on TRNO field & select Srch Help option
* Provide Search Help Name: ZELE\_TRSEARCH
* Select Yes & then Yes
* Activate table (Ctrl+F3)

**Testing:**

* Select Contents option
* Apply F4 on TRNO field
* You will find an error then
* Go to SE38
* Provide Program: ZMM\_TR
* Select Create option
* Write PARAMETERS p\_tr TYPE ZMM\_TR-TRNO.
* Activate program then go back & test the table

**Table Maintenance Generator (TMG):**

Using TMG you can directly Insert, Update, Delete data from a database table

**Note:**

1. In real time for certain requirements when data is not linked with other tables. Data can be inserted (or) manipulated from table level directly. This can be done by creating a TMG on that table
2. TMG is allowed only for ZEE tables

**Creation of Table Maintenance Generator:**

**Create Table First:**

* Go to SE11
* Provide Database Table: ZSD\_MAIL
* Select Create option
* Provide Short Description: Customer Mail Table
* Provide Delivery Class: A
* Provide Table View Maintenance: Display Maintenance Allowed (both display & create entries is allowed)
* Select Fields tab

Field key Data Element

MANDT \_/ MANDT

KUNNR \_/ KUNNR

MAILID \_/ ZMAIL\_DAT

* Double click on ZMAIL\_DAT
* You will find a message pop-up select Yes option
* Select Local Object option
* Again you will find a warning: application table should be client specific the press Enter
* You will find a pop-up: create the data element then click Yes
* Provide Short Description: Customer Mail Id
* Provide Domain: CHAR
* Select Field Label tab

Length field lable

Medium 20 Mail Id

* Activate data element (Ctrl+F3)
* Go to back (F3)

Field key data element

INDICATOR \_/ Z\_INDICATOR

* Double click on Z\_INDICATOR
* Save Yes
* You will find a warning pop-up: Enter
* You will find a pop-up: create data element Yes
* Provide Short Description: Indicator
* Provide Domain: CHAR20
* Select Value Range tab

FIX VAR Short Desciption

1. TO
2. CC
3. BCC

* Activate the domain (Ctrl+F3)
* Go to back (F3)
* Select Field Label tab
* Provide Short Description: Indicator

Length field lable

Medium 20 Indicator

* Activate data element (Ctrl+F3)
* Go to back (F3)
* Select Technical Settings button
* Provide Data Class: APPL0 (APPL0---for master data & APPL1---for transactional data)
* Provide Size Category: 0 (it means table holds up to 8000 records)
* Select Save option (Ctrl+s)
* Select back (F3)
* Activate Table (Ctrl+F3)
* During activation you will find a warning pop-up select Yes

Ignore the warnings & go back (F3)

**Creating TMG:**

* Go to SE80
* Under Test Repository select Function Group option
* Provide Function Group: ZMAIL\_GR then press Enter
* You will find a pop-up create object press Yes
* Again you will find a pop-up Short Text: Mail Group then press Enter
* Select Local Object
* Go to SE11
* Provide Database Table: ZSD\_MAIL
* Select Change option
* Select Utilities---🡪Table Maintenance Generator
* Provide Authorization Group: &NC&
* Provide Function Group: ZMAIL\_GR
* Select One Step option
* Select Scr Number(s) button
* You will find a pop-up select Propose Screen Number option then press Enter
* Select Create option (F6)
* Select Save option (Enter)
* Select Local Object (Enter)

**Creating TCode:**

* Go to SE93
* Provide Transaction Code: ZMAIL
* Select Create option
* Provide Short Text: Customer Mail Details
* Select Transaction with Parameters (Parameter Transaction) option then press Enter
* Provide Transaction: SM30
* Scroll down to bottom then provide
* Name of Screen Field Value

ViewName SZD\_MAIL

Update X

* Save (Ctrl+s)
* Select Favorites---🡪Inset Transaction
* Provide Transaction Code: ZMAIL then press Enter

**Testing TMG:**

* Provide Run TCode: ZMAIL then press Enter
* Provide Table: ZSD\_MAIL
* Select Maintain option
* Select New Entries button
* Insert records & save

**Navigation for maintain Default Values using TMG:**

Sometimes Date, User name, Time needs to be entered in a table. These values will not be inserted by client (inserted automatically by system)

**Create Table First:**

* Go to SE11
* Provide Database Table: ZFI\_LIMITS
* Select Create option
* Provide Short Description: Bank Limits
* Provide Delivery Class: A
* Provide Table View Maintenance: Display Maintenance Allowed (both display & create entries is allowed)
* Select Fields tab

Field key Data Element

MANDT \_/ MANDT

COUNTER \_/ INT1

Bank Name \_/ BANKA

Amount DMBTR

ERDAT ERDAT

Uname ZUSER

ERTIM ZTIMEDAT

* Select Currency Quantity Fields tab

Provide Amount: Reference Table Reference Field

T001 WAERS

* Select Technical Settings button
* Provide Data Class: APPL0 (APPL0---for master data & APPL1---for transactional data)
* Provide Size Category: 0 (it means table holds up to 8000 records)
* Select Save option (Ctrl+s)
* Select back (F3)
* Activate Table (Ctrl+F3)
* During activation you will find a warning pop-up select Yes

Ignore the warnings & go back (F3)

**Providing default values using TMG:**

* Select Utilities---🡪Table Maintenance Generator
* Provide Authorization Group: &NC&
* Provide Function Group: ZMAIL\_GR
* Select Find Scr Number option then press Enter
* Select Create option (F6) then press Enter
* Select Environment option
* Select Modification------🡪Events
* You will find a message- do not make any changes on SAP data then press Enter
* Select New Entries option provide

T Form Routine

05 Create\_Entry

* Press Enter
* Under Editor double click
* You will find a pop-up then press Enter
* You will find a warnings pop-up then press Enter
* Provide code for default values

FORM CREATE\_ENTRY.

ZFI\_LIMITS-ERDAT = SY-DATUM.

ZFI\_LIMITS-UNAME = SY-UNAME.

ZFI\_LIMITS-ERTIM = SY-UZETT.

ENDFORM.

* Activate table & go back

**Testing:**

* Go to SM30
* Provide Table: ZFI\_LIMITS
* Select Maintain option
* Select New Entries option
* Insert records & Save
* You will find a dump note: SAPLZMAIL\_GR
* Go to SE38
* Provide Program: SAPLZMAIL\_GR
* Select Change option
* Activate the program (Ctrl+F3)
* Go to SM30
* Provide Table: ZFI\_LIMITS
* Select Maintain option
* Select New Entries option
* Insert records & Save

**04-09-2014**

**Adding Check boxes in Table in TMG:**

**Create Table First:**

* Go to SE11
* Provide Database Table: ZPP\_MACHINE
* Select Create option
* Provide Short Description: Machine Down Time
* Provide Delivery Class: A
* Provide Table View Maintenance: Display Maintenance Allowed (both display & create entries is allowed)
* Select Fields tab

Field key Data Element

MANDT \_/ MANDT

MACHNO \_/ ZMACH\_DAT

BUDAT \_/ BUDAT

STIME ZSTART\_DAT

ETIME ZEND\_DAT

DTIME ZDOWN\_DAT

CHK ZCHK\_DAT

* Double click on ZCHK\_DAT
* Save Yes
* You will find a warning pop-up: Enter
* You will find a pop-up: create data element Yes
* Provide Short Description: Check Box
* Provide Domain: ZCHK\_DOMAIN
* Provide Data type: CHAR
* Provide No.of Characters: 1
* Select Value Range tab

FIX VAR Short Desciption

X ON

OFF

* Activate the domain (Ctrl+F3)
* Go to back (F3)
* Select Field Label tab
* Provide Short Description: Check Box

Length field lable

Medium 20 Check

* Activate data element (Ctrl+F3)
* Go to back (F3)
* Select Technical Settings button
* Provide Data Class: APPL0 (APPL0---for master data & APPL1---for transactional data)
* Provide Size Category: 0 (it means table holds up to 8000 records)
* Select Save option (Ctrl+s)
* Select back (F3)
* Activate Table (Ctrl+F3)
* During activation you will find a warning pop-up select Yes

Ignore the warnings & go back (F3)

**Providing Check boxes using TMG:**

* Select Utilities---🡪Table Maintenance Generator
* Provide Authorization Group: &NC&
* Provide Function Group: ZMAIL\_GR
* Select Find Scr Number option then press Enter
* Select Create option (F6) then press Enter
* Select Environment option
* Select Modification------🡪Events
* You will find a message- do not make any changes on SAP data then press Enter
* Select New Entries option provide

T Form Routine

05 Create\_Entry

* Press Enter
* Under Editor double click
* You will find a pop-up then press Enter
* You will find a warnings pop-up then press Enter
* Provide code for default values

FORM CREATE\_ENTRY.

ZDOWN\_DAT = ZEND\_DAT - ZSTART\_DAT.

ENDFORM.

* Activate table & go back

**Creating TCode:**

* Go to SE93
* Provide Transaction Code: ZDOWN
* Select Create option
* Provide Short Text: Machine Down Time
* Select Transaction with Parameters (Parameter Transaction) option then press Enter
* Provide Transaction: SM30
* Scroll down to bottom then provide
* Name of Screen Field Value

ViewName SPP\_MACHINE

Update X

* Save (Ctrl+s)
* Select Favorites---🡪Inset Transaction

Provide Transaction Code: ZDOWN then press Enter

**Testing:**

* Go to SE38
* Provide Program: SAPLZDOWN
* Select Change option
* Activate the program (Ctrl+F3)
* Go to SM30
* Provide Table: ZPP\_MACHINE
* Select Maintain option
* Select New Entries option
* Insert records & Save

**Text Table:**

Use text table for storing explanatory texts. For storing such texts it is not advisable to store in Primary Table

**Note:**

Text Tables works based on Primary & Foreign key relationship

**Creation of Text Table:**

**First Create Primary Table:**

* Go to SE11
* Provide Database Table: ZTEXT\_TAB
* Select Create option
* Provide Short Description: Department Table
* Provide Delivery Class: A
* Provide Table View Maintenance: Display Maintenance Allowed (both display & create entries is allowed)
* Select Fields tab

Field key Data Element

MANDT \_/ MANDT

DEPT \_/ ZDEPT\_DAT

* Select Technical Settings button
* Provide Data Class: APPL0 (APPL0---for master data & APPL1---for transactional data)
* Provide Size Category: 0 (it means table holds up to 8000 records)
* Select Save option (Ctrl+s)
* Select back (F3)
* Activate Table (Ctrl+F3)
* During activation you will find a warning pop-up select Yes

Ignore the warnings & go back (F3)

**Creating Text Table:**

* Go to SE11
* Provide Database Table: ZTEXT\_TAB1
* Select Create option
* Provide Short Description: Department Text
* Provide Delivery Class: A
* Provide Table View Maintenance: Display Maintenance Allowed (both display & create entries is allowed)
* Select Fields tab

Field key Data Element

MANDT \_/ MANDT (for client dependant table)

SPRAS \_/ SPRAS (for language selection)

DEPT \_/ ZDEPT\_DAT

DESC1 MAKTX

* Select Technical Settings button
* Provide Data Class: APPL0 (APPL0---for master data & APPL1---for transactional data)
* Provide Size Category: 0 (it means table holds up to 8000 records)
* Select Save option (Ctrl+s)
* Select back (F3)
* Activate Table (Ctrl+F3)
* During activation you will find a warning pop-up select Yes

Ignore the warnings & go back (F3)

* Select SPRAS
* Select Foreign keys option
* You will find a pop-up then press Yes & then press Enter
* Select DEPT
* Select Foreign keys option
* Provide Check Table: ZTEXT\_TAB
* Select Generate Proposal button
* At bottom select Keys Fields of a Text Table option
* Provide Cardinality 1:N then press Enter
* Select MANDT
* Select Foreign keys option
* You will find a pop-up then press Yes & then press Enter
* Activate table (Ctrl+F3)

**Testing Text Table:**

* Go to SE11
* Provide Database Table: ZTEXT\_TAB (primary table name)
* Select Change option
* Select Utilities----🡪Table Contents----🡪Create Enteries
* Here you will find text table options also then provide text & save

**Some Important Steps for Working with Standard Tables:**

* Go to SE11
* Provide Database Table: KNA1
* Select Display option
* Select Contents option

**Step-1:**

**Navigation for Getting Selected Fields in Output List:**

* Select Settings---🡪Format List----🡪Choose Fields option
* By default all fields are selected then select Deselect option (Shift+F2)
* Select required fields KUNNR, NAME1, ORT01, PSTLZ then Enter
* Execute (F8)

**Step-2:**

**Adding (or) Removing Fields from Selection Screen:**

* Select Settings---🡪Fields for Selection option
* Select & deselect fields based on requirement then Enter

**Step-3:**

**Transforming Data from SAP to Local File (XL Sheet):**

* Select F8
* Select Local File option (last from 5th option) (Ctrl+Shift+F9)
* You will find a pop-up select Spread Sheet option then Enter
* You will find a pop-up apply F4 on Directory field
* You will find a pop-up then select Desktop
* Provide File Name: Customer then select Save option
* Select Generate option

**05-09-2014**

**Types of Tables:**

1. Transparent Table
2. Cluster Table
3. Pooled Table

**Transparent Table:**

* It is a default table in SAP
* It holds application data (master data & transactional data)
* It follows one-to-one relationship (a table created in Presentation Server, if with the same Structure & Fields another table created in Database Server)
* These tables can be accessed even outside SAP
* This table allows both Open SQL & Native SQL statements
* In real time we are mostly working with Transparent Table

**Pooled Table & Clustered Table:**

* These tables holds SAP proprietary format data (system data, statistical data, control data……)
* These tables allows (or) follows many-to-one relationship (many tables data combined to form a Pooled Table & Clustered Table)
* These tables cannot be accessed outside SAP
* These tables allows only Open SAL statements
* Very rarely we are using these tables (most common tables used one BSEG, KONU)

**Note:**

You cannot apply Inner-Join statement on Cluster Table

**Open SQL & Native SQL Statements:**

**Native SQL Statements:**

* These are traditional SQL statements
* With Native SQL Statement request is directly forwarded to Database Server which processes the request & returns the response
* With Native SQL statement performance of the system is highly degraded

**Syntax:**

EXEC SQL.

SELECT………………

ENDEXECSQL.

**Open SQL Statements:**

These are normal SQL statements which are converted to embedded SQL using database interface which is further passed to Buffer, if data is available in Buffer you will get the response

**Buffering Concept:**

Buffering is a way of improving performance which accessing data from database. Whenever you run a program it will make a request to Buffer for data, if it is not available it will make a request to Database Server which returns response to Buffer & simultaneously to Program. Next time whenever you run the same program this time it will collect the data directly from Buffer

**Note:**

Buffering is important for frequent & common accessing of application data

**Types of Buffering:**

1. Single Record Buffering
2. Generic Buffering
3. Full Buffering

**Single Record Buffering:**(SELECT SINGLE)

This buffering is used for frequent access of data from database. Here volume of data is very large

**Generic Buffering:**(WHERE <condition>)

Use generic buffering where records are accessed based on condition

**Full Buffering:**(SELECT \*)

This buffer is allocated when you are fletching all the fields from database table

**Check Table:**

It works based on Primary & Foreign key relationship. Data is always validated at Field level

**Value Table:**

It is a default Check Table & it always maintained under Domain under Value Range option

**09-09-2014**

**MODULES**

**MM Module:**

Materials Management Module

**Departments in MM Module:**

1. Purchase
2. Warehouse
3. General Stores
4. Finance in MM

**Process Flow in MM Module:**

IT Department

Purchase Requisition (Indent)

PR Approval (HOD, VP, President)

Purchasing Department Vendor Analysis

Request for Quotation Best Vendor

Vendor1 Vendor2 Vendor3 Purchase Order

Quotation PO Approval

Delivery (Manifacture+Pickup+Packing) Vendor1

Delivery Challana (Delivery Note)

Row Material Finished Goods

General Store (GRS)

Invoice Verification (Bill Processing)

Vendor Payment

**Purchase Order:**

Vendor: PO Date:

Vendor Details:

Payment Terms:

Item No.of Pieces Unit price Amount

10 2 100 200

-- - - -

- - - -

**Brief Flow:**

PR

PR Purchase

RFQ

PO

PO Release

GRN

IV

Vendor Payment

**Applications & Tables in MM Module:**

1. **Purchase Requisition:**

**Tcodes:** ME51N, ME52N, ME53N

**Tables:** EBAN

**Note:** The key field is BANFN (PR number)

1. **Purchase Order:**

**Tcodes:** ME21N, ME22N, ME23N

**Tables:** EKKO, EKPO, EKBE, EKKN, EKAN, EKET

**Note:** The key field is EBELN (PO number)

1. **Goods Receipt:**

**Tcodes:** MIGO

**Tables:** MKPF, MIEG

**Note:** The key field is MBLNR (Material Document Number)

1. **Invoice Verification:**

**Tcodes:** MIR0, MIR7

**Tables:** RBKP, RSEG

**Note:** The key field is BELNR (Accounting Document Number)

1. **Vendor Master Data:**

**Tcodes:** XK01, XK02, XK03

**Tables:** LFA1, LFB1, LFM1, LFBK, LFC1

**Note:** The key field is LIFNR (Vendor Account Number)

1. **Material Master Data:**

**Tcodes:** MM01, MM02, MM03

**Tables:** MARA, MARC, MARM, MARD, MAKT, MVKE, MARC, MAST, MBEW, MSLB

**Note:** The key field is MATNR (Material Code)

1. **Service Entry Sheet:**

**Tcodes:** ML81N

**Tables:** ESSR, ESSL

**Note:** The key field is LBLNI (Entry Sheet Number)

**Inner Joins:**

Use inner joins for fletching common data from two (or) more than two tables

**Syntax-1:**

SELECT LFA1~lifnr LFA1~name1

EKKO~ebeln EKKO~bedat

EKPO~menge EKPO~netpr

INTO TABLE <it> FROM LFA1

INNER JOIN EKKO ON LFA1~lifnr=EKKO~lifnr

INNER JOIN EKPO ON EKKO~ebeln=EKPO~ebeln

WHERE <codition>

**Syntax-2:**

SELECT A~lifnr A~name1

B~ebeln B~bedat

C~menge C~netpr

INTO TABLE <it> FROM LFA1 AS A

INNER JOIN EKKO AS B ON A~lifnr=B~lifnr

INNER JOIN KKPO AS C ON B~ebeln=C~ebeln

WHERE <codition>

**10-09-2014**

**Example:**

Lifnr name1 ebeln lifnr bedat

|  |  |
| --- | --- |
| 1 | X |
| 2 | Y |
| 3 | Z |
| 4 | M |

|  |  |  |
| --- | --- | --- |
| 100 | 1 | 1.6.2014 |
| 200 | 1 | 2.3.2014 |
| 300 | 2 | 3.10.2014 |
| 400 | 2 | 5.6.2014 |
| 500 | 3 | 10.9.2014 |

Inner Join Left Outer Join

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | X | 100 | 1.6.2014 |
| 1 | X | 200 | 2.3.2014 |
| 2 | Y | 300 | 3.10.2014 |
| 2 | Y | 400 | 5.6.2014 |
| 3 | Z | 500 | 10.9.2014 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | X | 100 | 1.6.2014 |
| 1 | X | 200 | 2.3.2014 |
| 2 | Y | 300 | 3.10.2014 |
| 2 | Y | 400 | 5.6.2014 |
| 3 | Z | 500 | 10.9.2014 |
| 4 | M |

**Requirement:**

Purchase Order Details based on Vendors

**Input:**

Vendor code ------- s\_lifnr to

Company code---- s\_bukrs to

Plant ------- s\_werks to

Material type------ s\_matnr to

Material group---- s\_matkl to

Po date-------------- s\_bedat to

Summary

Details

**Output:** LFA1-lifnr EKKO-ebeln EKKO-bedat EKPO-matnr EKPO-menge EKPO-netpr

1000 EKPO-meins

3000 2.6.2.14 100-100 50 2000 PC

3001 15.6.2014 100-101 70 3000 PC

Totals 120 5000

1001

5001 3.6.201 F001 2 80 PC

5002 9.6.2014 F002 1 90 PC

5003 15.6.2014 F003 3 100 PC

Totals 6 270

Grand totals 126 5270

**Solution:**

Go to SE38

Provide Program: ZMM\_Purchase\_Order\_Details

Select Create

Title: Purchase Order Details based on Vendors

Type: Executable Program

Select Save, Local Object

\* provide tables work area

TABLES: LFA1, EKKO, EKPO

\* provide selection scree

SELECTION-SCREE: BEGIN OF BLOCK b1 WITH FRAME TITLE text-001.

SELECT-OPTIONS: s\_lifnr FOR LFA1-lifnr

s\_bukrs FOR EKKO- bukrs

s\_mtart FOR EKPO-mtart

s\_matkl FOR EKPO-matkl

s\_bedat FOR EKKO-bedat

SELECTION-SCREEN: END OF BLOCK b1.

SELECTION-SCREE: BEGIN OF BLOCK b2 WITH FRAME TITLE text-001.

PARAMETERS: p\_rb1 RADIOBUTTON GROUP RG1

p\_rb2 RADIOBUTTON GROUP RG1

SELECTION-SCREEN: END OF BLOCK b2.

TYPES: BEGIN OF ty\_po,

lifnr TYPE LIFNR,

ebeln TYPE EBELN,

bedat TYPE BEDAT,

matnr TYPE MATNR,

meins TYPE MEINS,

mengeTYPE MENGE\_D,

netpr TYPE NETPR,

END OF ty\_po.

DATA: wa\_po TYPE ty\_po,

it\_po TYPE TABLE OF ty\_po.

SELECT LFA1~lifnr EKKO~ebeln EKKO~bedat EKPO~matnr EKPO~meins

EKPO~menge EKPO~netpr INTO TABLE it\_po FROM LFA1 INNER JOIN EKKO

ON LFA1~lifnr = EKKO~lifnr

INNER JOIN EKPO

ON EKKO~ebeln = EKPO~ebeln

WHERE LFA1~lifnr IN s\_lifnr

AND EKKO~bukrs IN s\_bukrs

AND EKPO~werks IN s\_werks

AND EKPO~mtart IN s\_mtart

AND EKPO~matkl IN s\_matkl

AND EKKO~bedat IN s\_bedat.

IF p\_rb1 EQ ‘X’.

SORT it\_po BY lifnr.

LOOP AT it\_po INTO wa\_po.

AT NEW lifnr.

WRITE:/10 wa\_po-lifnr COLOR 1.

SKIP.

ENDAT.

FORMAT COLOR COL\_GROUP INTENSIFIED OFF.

NEW-LINE.ULINE AT 1(135).

WRITE:/ SY-VLINE,

10 wa\_po-ebeln,

30 wa\_po-bedat,

50 wa\_po-matnr,

70 wa\_po-meins,

90 wa\_po-menge,

100 wa\_po-netpr,

135 SY-VLINE.

AT END OF lifnr.

SUM.

FORMAT COLOR COL\_TOTAL INTENSIFIED OFF.

NEW-LINE.ULINE AT 1(135).

WRITE:/ SY-VLINE,

10 ‘Totals’,

70 wa\_po-menge,

90 wa\_po-netpr,

135 SY-VLINE.

ENDAT.

AT LAST.

SUM.

FORMAT COLOR COL\_TOTAL INTENSIFIED OFF.

NEW-LINE.ULINE AT 1(135).

WRITE:/10 ‘Grand Totals’,

70 wa\_po-menge,

90 wa\_po-netpr,

135 SY-VLINE.

ENDAT.

ENDLOOP.

ELSE

LOOP AT it\_po INTO wa\_po.

AT FIRST.

FORMAT COLOR COL\_HEADING INTENSIFIED ON.

NEW-LINE.ULINE AT 1(135).

WRITE:/ SY-VLINE,

10 ‘Vendor Code’,

25 ‘PO Number’,

35 ‘PO Date’,

45 ‘Material Code’,

60 ‘Unit of Measure’,

70 ‘Quantity’,

80 ‘Unit Price’,

135 SY-VLINE.

NEW-LINE.ULINE AT 1(135).

ENDAT.

FORMAT COLOR COL\_GROUP INTENSIFIED OFF.

NEW-LINE.ULINE AT 1(135).

WRITE:/ SY-VLINE,

5 wa\_po-lifnr,

10 wa\_po-ebeln,

30 wa\_po-bedat,

50 wa\_po-matnr,

70 wa\_po-meins,

90 wa\_po-menge,

100 wa\_po-netpr,

135 SY-VLINE.

ENDLOOP.

ENDIF.

**LFA1 Table:**

This table holds Vendor Master Data

**Fields:**

1. Lifnr-------Vendor Number
2. Land1-----Country of Vendor
3. Name1----Name of Vendor
4. Ort01-----City of Vendor
5. Pstlz------Postal Code of Vendor
6. Adrnr-----Address Number

**Links:**

LFA1~adrnr

ADRC~addrnumber

1. Brsch-----Industry Key

**EKKO Table:**

This table holds Purchase Order Header Data

**Fields:**

1. Ebeln-------Purchase Order Number
2. Bukrs------Company Code
3. Bsart-------Purchasing Document Type
4. Statu-------Status of Purchase Order
5. Lifnr-------Vendor Number

**Links:**

LFA1~lifnr

EKKO~lifnr

1. Zterm------Payment Terms
2. Zbd1p-----Cash Discount Percentage
3. Ekorg------Purchase Organization (Purchase Department)
4. Ekgrp------Purchasing Group
5. Waers------Currency
6. Wkurs------Exchange Rate
7. Bedat-------Purchase Order Date
8. Inco1, Inco2-------International Terms for Carrying Bussiness
9. Knumv------Tax Number

**Links:**

EKKO~knumv

KONU~knumv

1. Kalsm------Pricing Procedure
2. Frgke-------Release Indicator

**EKPO Table:**

This table holds Purchase Document Item Data

**Fields:**

1. Ebeln-------Purchase Order Number

**Links:**

EKKO~ebeln

EKPO~ebeln

1. Ebelp------Item Number
2. Txzo1-----Item Text
3. Matnr-----Material Code
4. Werks-----Plant
5. Lgort------Storage Location
6. Matkl-----Material Group
7. Menge------Purchase Order Quantity
8. Meins------Unit of Measurement
9. Netpr------Unit Price
10. Netwr------Total Price
11. Mwskz-----Sales Tax Code
12. Elikz--------Delivery Completed Indicator

**11-09-2014**

**Message Handling:**

Using MESSAGE keyword you can display your own message

**Note:**

SY-SUBRC

* It returns values after ABAP statement
* It returns 0 if an ABAP statement is processed successfully otherwise it returns Exception (other than 0 value)

**Example:**

SELECT…………………

…………………………..

IF SY-SUBRC EQ 0.

MESSAGE ‘Data Found’ TYPE ‘I’.

ELSE.

MESSAGE ‘Enter Valid Input’ TYPE ‘E’.

**Message Class:**

* You can store all the messages in Message Class also
* T-code for Message Class is SE91

**Example:**

IF SY-SUBRC EQ 0.

MESSAGE I000(zmm\_mess1).

ELSE.

MESSAGE E001(zmm\_mess1).

* Double click on zmm\_mess1 (OR)
* Go to SE91
* Provide Message Class: zmm\_mess1
* Select Create option
* Provide Short Text: Message Handling
* Select Save option
* Provide Messages

000---------Data Found

001---------Enter Valid Input

* Select Save option
* Activate Message Handling

**Message Options:**

I-----Information------Information pop-up is displayed

S----Status------Message is displayed in Status Bar

W----Warning-----A yellow color warning message is displayed & Control will switch to next

Screen

E-----Error------Red color message is displayed & Control will stay in same Screen

A----Amend-----Control will switch to Screen 0

X-----Exit-------Control will switch to ABAP Runtime Screen (ST22)

**12-09-2014**

**Modularization Techniques:**

There are 5 modularization techniques in SAP.

They are

1. Subroutines
2. Function Modules
3. Include
4. Macro
5. Field Symbol

* In real time you should ensure that every program should be highly readable & reusable
* For achieving readability & reusability we have to work with modularization techniques

**Subroutines:**

* Subroutine is like a mini program which can be called within the same program (or) from other programs
* Using Subroutines you can perform Calculations, Call other function modules, Write statements……….etc

**Types of Subroutines:**

1. Internal Subroutines
2. External Subroutines

**Internal Subroutines:**

In this both calling part & definition part are in same program

**Syntax:**

PERFORM <subroutine name> USING <param1> <param2> ………….

FORM <subroutine name> USING <param1> <param2> ………….

-----------------

-----------------

ENDFORM.

**Note:**

1. In above syntax PERFORM will call its FORM & simultaneously pass Parameters to its FORM
2. Subroutine name can be any name. The same name should be used both in PERFORM & FORM
3. The Parameters defined in PERFORM are called Actual Parameters. These parameters have global visibility & must be defined in a program
4. The parameters maintained in FORM are called Formal Parameters. These parameters have local visibility & are automatically created based on Actual Parameters
5. Actual Parameters will pass values to Formal Parameters by using USING option
6. The number of Actual Parameters should be same that of Formal Parameters

**Example:**

WRITE:/ ‘Welcome to Subroutine’.

PERFORM sub.

WRITE:/ ‘First Call’.

PERFORM sub.

WRITE:/ ‘Second Call’.

FORM sub.

WRITE:/ ‘Inside Sub’.

ENDFORM.

**Output:**

Welcome to Subroutine

Inside Sub

First Call

Inside Sub

Second Call

**Debugging:**

**Break Point:**

Using break point the execution of a program can be terminated & until the break point is reached

**Note:**

In real time for large programs debugging each & every part of a program is time consuming. If you know where the problem arises you can directly keep a break point & test the program

**Types of Break Points:**

1. Dynamic Break Point
2. Static Break Point

**Dynamic Break Point:**

* Dynamic break point is kept automatically & can be removed during runtime
* It is automatically discarded whenever system is logged off

**Note:**

Dynamic break point is used if you are not aware of the exact problem in your program. During debugging you can set dynamic break point at different, different places & you can remove the same dynamically

**Static Break Point:**

It is kept by using a keyword BREAK-POINT (or) BREAK <user name>

**Note:**

Static break point is used if you know the exact problem in your program & you want to debug the same code N number of times

**Pass by Value & Pass by Reference:**

**Pass by Value:**

* In pass by value, value (or) values is passed from Actual Parameters to Formal Parameters
* Both calling part & definition part will share different memory locations
* In call by value nothing is reflected back to calling part

**Example:**

DATA v1 TYPE C VALUE ‘a’.

PERFORM sub USING v1.

WRITE:/ v1.

FORM sub USING VALUE(p1).

p1 = ‘b’.

WRITE:/ p1.

ENDFORM.

**Output:**

b

a

**Pass by Reference:**

* In pass by reference value is not passed to Formal Parameters instead a Pointer (or) Reference (or) Address is passed to Formal Parameters
* Any changes done in Formal Parameters will be effected to Actual Parameters also

**Example:**

DATA v1 TYPE C VALUE ‘a’.

PERFORM sub USING (or) CHANGING v1.

WRITE:/ v1.

FORM sub USING (or) CHANGING p1.

p1 = ‘b’.

WRITE:/ p1.

ENDFORM.

**Output:**

b

b

**External Subroutines:**

In external subroutines both PERFORM & FORM maintained in separate programs

**Example:**

* Go to SE38
* Program: ZMM\_GRM\_DETAILS
* Create
* Title: GRM Details
* Type: Executable Program
* Save
* Local Object

TABLES: MKPF, MSEG.

SELECTION-SCREEN: BEGIN OF BLOCK b1 WITH FRAME TITLE text-000.

SELECT-OPTIONS: s\_mblnr FOR MKPF-mblnr,

s\_werks FOR MSEG-werks,

s\_budat FOR MKPF-budat.

SELECTION-SCREEN: END OF BLOCK b1.

PERFORM extsub (ZMM\_GRM\_FORM) USING s\_mblnr-low s\_mblnr-high s\_werks-low s\_werks-high s\_budat-low s\_budat-high.

* Double click on ZMM\_GRM\_FORM
* You will find a pop-up select Yes
* Save & Yes
* You will find a pop-up deselect With Top Incl check box the press Enter

TYPES: BEGIN OF ty\_mseg,

mblnr TYPE mblnr,

budat TYPE budat,

menge TYPE menge\_D,

matnr TYPE matnr,

dmbtr TYPE dmbtr,

END OF ty\_mseg.

DATA: wa\_mseg TYPE ty\_mseg,

It\_mseg TYPE TABLE OF ty\_mseg.

FORM extsub USING VALUE(grm\_low) VALUE(grm\_high) VALUE(plnt\_low) VALUE(plnt\_high) VALUE(date\_low) VALUE(date\_high)

SELECT MKPF~mblnr MKPF~budat MSEG~menge MSEG~matnr MSEG~dmbtr INTO TABLE it\_mseg FROM MKPF

INNER JOIN MSEG

ON MKPF~mblnr = MSEG~mblnr

WHERE MKPF~mblnr BETWEEN grm\_low AND grm\_high

AND MSEG~werks BETWEEN plnt\_low AND plnt\_high

AND MKPF~budat BETWEEN date\_low AND date\_high

AND MSEG~bwart EQ ‘101’.

LOOP AT it\_mseg INTO wa\_mseg.

WRITE:/10 wa\_mseg-mblnr,

30 wa\_mseg-budat,

50 wa\_mseg-menge,

70 wa\_mseg-matnr,

90 wa\_mseg-dmbtr.

ENDLOOP.

ENDFORM.

**Output:**

s\_mblnr : 49000000 to 490005000

s\_werks: 1000 to 1200

s\_budat: 15.11.1994 to 01.06.2009

Execute (F8)

**13-09-2014**

**MKPF Table:**

It holds Material Document Header Data

**Fields:**

1. Mblnr ---------- Material Document Number
2. Mjahr ---------- Material Document Year
3. Blart ---------- Material Document Type
4. Budat ---------- Material Document Date
5. Xblrn ---------- Reference Document (Purchase Order Document) Number

**MSEG Table:**

It holds Material Document Item Data

**Fields:**

1. Mblnr ------ Material Document Number
2. Mjahr ------ Material Document Year
3. Bwart ------ Moment Type
4. Werks ------ Plant
5. Lgort ------- Storage Location
6. Charg ------ Batch Number
7. Lifnr ------- Vendor Account Number

**Link:**

LFA1-lifnr

MSEG-lifnr

1. Shkzg ------- Debit / Credit Indicator
2. Dmbtr ------ Amount in Local Currency
3. Menge ------ Quantity
4. Meins ------- Unit of Measurement
5. Ebeln ------- Purchase Order Number

**Link:**

EKPO-ebeln

MSEG-ebeln

1. Ebelp -------- Item Number

**Link:**

EKPO-ebelp

MSEG-ebelp

1. Smbln ------- Number of Material Document Number
2. Kostl -------- Department
3. Aufnf ------- Production Order Number

**Function Module:**

* Function Module is a piece of code which performs a specific task based on given requirement
* Function Module is a responsible program which always accepts Parameters & returns a Value

**Types of Function Modules:**

1. Normal Function Module
2. Remote Enabled Function Module
3. ALV Function Module (ABAP List Viewer)

**Normal Function Module:**

It is a piece of code which performs a specific task based on given Parameters

**Remote Enabled Function Module:**

* It is for distributed environment.
* You can call this function module within the system & across the system

**ALV Function Module:**

* These are part of Normal Function Module
* Has they improves performance of a programming so they are separated from Normal Function Modules

**Function Group:**

* It is a collection of identical objects
* Without Function Group you cannot create a Function Module
* In one Function Group we can maintain 99 Function Modules
* Apart from this SAP providing more than 120000 Function Modules which are surrounding more than 10000 Function Modules

**Syntax:**

CALL FUNCTION ‘<FUNCTION MODULE NAME>’. ------ calling part (SE38)

FUNCTION <FUNCTION MODULE NAME>.

-------------

------------- ------- definition part (SE37)

ENDFUNCTION.

**Function Module Interface:**

These are Parameters which are pass & return from a Function Module

**Types of Function Module Interfaces:**

1. Exporting
2. Importing
3. Tables
4. Changing

**Exporting:**

These are Variables (or) Field Groups which are pass to a Function Module in order to perform a task

**Importing:**

Function Modules always returns a value via Importing Parameters

**Tables:**

These are Internal Tables

**Changing:**

In latest versions Table Parameters are replaced by Changing

**Note:**

1. Exporting & Importing Parameters works based on Pass by Value
2. Tables & Changing Parameters works based on Pass by Reference

**Steps for Create Function Module:**

1. Work with SE80 (create Function Group)
2. Work with SE37 (create Function Module based on Function Group)
3. Work with SE38 (call Function Module)

**Step-1:**

* Go to SE80
* Under Test Repository select Function Group
* Provide Function Group name: ZCAL\_GR
* Enter, Yes
* Provide Short Text: Function Group for Calculations
* Select Save, Local Object

**Step-2:**

* Go to SE37
* Function Module: ZCALC\_MOD
* Create
* Function Group: ZCAL\_GR
* Short Text: Function Module for Calculations
* Enter, Enter
* Import tab

Parameter Name Typing Associated Type

X TYPE I

Y TYPE I

* Export tab

Parameter Name Typing Associated Type

Z TYPE I

* Exception tab

Exception Short Text

No\_Data Wrong Calculation

* Source Code tab

z = x + y.

IF SY-SUBRC NE 0.

RAISE No\_Data.

ENDIF.

* Activate Function Module (F8)

**Step-3:**

* Go to SE38
* Program: ZCALC\_CALL\_SUM
* Create
* Title: Function Module for Calculations
* Type: Executable Program
* Save, Local Object

PARAMETERS: a TYPE I,

b TYPE I.

DATA c TYPE I.

\* calling function module

CALL FUNCTION ‘ZCALC\_MOD’.

EXPORTING.

x = a.

y = b.

(or)

IMPORTING.

z = c.

* Select Pattern option
* Call Function: ZCALC\_MOD

WRITE:/ ‘The Addition is:’, c.

**Working with Standard Function Module:**

* Go to SE37
* Function Module: HR\_RU\_AGE\_YEARS (standard function module name)
* Select F8

PERNR: 1000 --------- employee number

BSDTE: 18.9.2014 -------- To days date

* Select F8

**Assignment:**

Accept a number as input say 3900 & the Function Module should return the same number in figures say Three Thousand and Nine Hundred

**Include Program:**

Include Program’s memory is available to any ABAP work bench tool

**Example:**

INCLUDE ZNC1

WRITE:/ ‘work’.

INCLUDE ZNC2

INCLUDE ZNC1.

WRITE:/ ‘hard’.

REPORT ZNC3

INCLUDE ZNC2.

**Note:**

Include Program is a program (or) reusable program without any Parameters

**Field Symbol:**

* Field Symbol is analog to pointer concept in ‘C’ language
* It holds Reference of other Variables & returns Value stored in the Reference

**Note:**

Using Field Symbol you can improve performance of a program

**Example:**

* Go to SE38
* Program: ZMM\_FIELDSYMBOL
* Create
* Title: Field Symbols
* Type: Executable Program
* Save, Local Object

TABLES: LFA1.

INCLUDE ZVEND\_DECC.

* Double click on ZVEND\_DECC
* Yes, Yes, Enter

SELECTION-SCREEN: BEGIN OF BLOCK b1 WITH FRAM TITLE text-000.

SELECT-OPTIONS: s\_lifnr FOR LFA1-lifnr,

s\_land1 FOR LFA1-land1.

SELECTION-SCREEN: END OF BLOCK b1.

TYPES: BEGIN OF ty\_lfa1,

lifnr TYPE lifnr,

land1 TYPE land1,

name1 TYPE name1,

ort01 TYPE ort01,

pstlz TYPE pstlz,

stras TYPE stras,

END OF ty\_lfa1.

FIELD-SYMBLOS <l\_fs> TYPE ty\_lfa1.

DATA it\_lfa1 TYPE TABLE OF ty\_lfa1.

PERFORM get\_data.

PERFORM display\_data.

FORM get\_data.

SELECT lifnr land1 name1 ort01 pstlz stras INTO TABLE it\_lfa1

FROM LFA1

WHERE lifnr IN s\_lifnr

AND land1 IN s\_land1.

ENDFORM.

FORM display\_data.

LOOP AT it\_lfa1 ASSIGNING <l\_fs>.

WRITE:/10 <l\_fs>-lifnr,

30 <l\_fs>-land1,

50 <l\_fs>-name1,

70 <l\_fs>-ort01,

90 <l\_fs>-pstlz,

110 <l\_fs>-stras,

ENDLOOP.

ENDFORM.

**Note:**

Nowadays most of the companies are using Field Symbols instead of Work area

**19-09-2014**

**FOR ALL ENTRIES keyword:**

Using FOR ALL ENTRIES you can fetch common data between two tables

**Syntax:**

SELECT <field1> <field2>……… FROM <database table> INTO TABLE

<internal table> FOR ALL ENTRIES IN <database internal table> WHERE <condition>

**Example:**

Segment wise Materials Details

**Input:**

1. Material Code
2. Material Type
3. Material Group
4. Division

**Output:**

Material Code Material Description Segment Description

**Processing Logic:**

1. Get Matnr Spart fields from MARA table based on input
2. Pass Spart field of MARA table to TSPAT table & get Vtext field where Spras field EQ to ‘EN’
3. Pass Matnr field of MARA table to MAKT table & get Maktx field where Spars field EQ to ‘EN’

**Solution:**

* Go to SE38
* Program: ZMM\_SEGMENTWISE\_MATERIALS
* Create
* Title: Segment wise Materials Details
* Save, Local Object

\* providing tables work area

TABLES: MARA.

\* providing input selection screen

SELECTION-SCREEN: BEGIN OF BLOCK b1 WITH FRAME TITLE text-000.

SELECT-OPTIONS: s\_mtart FOR MARA-mtart,

s\_matnr FOR MARA-matnr,

s\_matkl FOR MARA-matkl,

s\_spars FOR MARA-spars.

SELECTION-SCREEN: END OF BLOCK b1.

\* creating structure for MARA table

TYPES: BEGIN OF ty\_mara,

matnr TYPE matnr,

spart TYPE spart,

END OF ty\_mara.

\* creating structure for MAKT table

TYPES: BEGIN OF ty\_makt,

matnr TYPE matnr,

maktx TYPE maktx,

END OF ty\_makt.

\* creating structure for TSPAT table

TYPES: BEGIN OF ty\_tspat,

spart TYPE spart,

vtext TYPE vtext,

END OF ty\_tspat.

\* creating structure for FINAL output

TYPES: BEGIN OF ty\_final,

matnr TYPE matnr,

maktx TYPE maktx,

vtext TYPE vtext,

END OF ty\_makt.

\* creating work areas & internal tables

DATA: wa\_mara TYPE ty\_mara,

wa\_makt TYPE ty\_makt,

wa\_tspat TYPE ty\_tspat,

wa\_final TYPE ty\_final,

it\_mara TYPE TABLE OF ty\_mara,

it\_makt TYPE TABLE OF ty\_makt,

it\_tspat TYPE TABLE OF ty\_tspat,

it\_final TYPE TABLE OF ty\_final.

\* calling Subroutines

PERFORM get\_mara.

PERFORM get\_makt.

PERFORM get\_tspat.

PERFORM combine\_data.

PERFORM display\_data.

\* getting matnr & spart from MARA table by comparing with input

FORM get\_mara.

SELECT matnr spart FROM MARA INTO TABLE it\_mara

WHERE matnr IN s\_matnr

AND matkl IN s\_matkl

AND spart IN s\_spart.

IF SY-SUBRC NE 0.

MESSAGE ‘Enter Valid Input’ TYPE ‘E’ DISPLAY LIKE ‘I’.

LEAVE LIST-PROCESSING.

ENDIF.

ENDFORM.

\* getting spart & vtext from TSPAT table by comparing spart of TSPAT with spart of MARA

FORM get\_tspat.

IF NOT it\_mara IS INITIAL.

SELECT spart vtext FROM TSPAT INTO TABLE it\_tspat

FOR ALL ENTRIES IN it\_mara

WHERE spart = it\_mara-spart

AND spars EQ ‘EN’.

ENDIF.

ENDFORM.

\* getting matnr & maktx from MAKT table by comparing matnr of MAKT with matnr of MARA

FORM get\_makt.

IF NOT it\_mara IS INITIAL.

SELECT matnr maktx FROM MAKT INTO TABLE it\_makt

FOR ALL ENTRIES IN it\_mara

WHERE matnr = it\_mara-matnr

AND spars EQ ‘EN’.

ENDIF.

ENDFORM.

\* combining required fields i.e. matnr, maktx & vtext

FORM combine\_data.

LOOP AT it\_mara INTO wa\_mara.

wa\_final-matnr = wa\_mara-matnr.

READ TABLE it\_makt INTO wa\_makt WITH KEY matnr = wa\_mara-matnr.

wa\_final-maktx = wa\_makt-maktx.

READ TABLE it\_tspat INTO wa\_tspat WITH KEY spart = wa\_mara-spart.

wa\_final-vtext = wa\_tspat-vtext.

APPEND wa\_final TO it\_final.

ENDLOOP.

ENDFORM.

\* displaying output

FORM display-data.

LOOP AT it\_final INTO wa\_final.

AT FIRST.

NEW-LINE.ULINE AT 1(135).

WRITE:/ SY-VLINE,

10 ‘Material Code’,

30 ‘Material Description’,

60 ‘Segment Description’,

135 SY-VLINE.

NEW-LINE.ULINE AT 1(135).

ENDAT.

FORMAT COLOR COL\_GROUP INTENSIFIED OFF.

NEW-LINE.ULINE AT 1(135).

WRITE:/ SY-VLINE,

10 wa\_final-matnr,

30 wa\_final-maktx,

60 wa\_final-vtext,

135 SY-VLINE.

NEW-LINE.ULINE AT 1(135).

ENDLOOP.

ENDFORM.

**20-09-2014**

**Pitfalls of FOR ALL ENTRIES:** (Drawbacks)

1. Before applying FOR ALL EMTRIES it is mandatory to check whether base internal table is having data (or) not

If you are not checking,

Assume if base internal table is completely empty then it will fetch all records from database table

1. It act as a SELECT DISTINCT (It removes duplicate set of records from an internal table)

**Example:**

BSEG Table

belnr hkgnt

1000 301011

301012

301012 ---- These records will be removed

301012

301013

301014

1. If base internal table is too large there will be performance degration

**Note:**

In real time always use

1. INNER JOINS for extracting common data between two tables
2. FOR ALL ENTRIES for extracting & combining multiple tables records

**MARA Table:**

It holds Material Master Data

**Fields:**

1. Matnr -------- Material Number
2. Ersda --------- Date of Creation
3. Ernam -------- Person who Created that Material
4. Mtart --------- Material Type
5. Mbrsg -------- Industry Key
6. Matkl --------- Material Group
7. Bismt --------- Old Material Number
8. Miens --------- Unit of Measurement
9. Brgew -------- Gross Weight
10. Ntgew -------- Net Weight
11. Spart ---------- Division Code

**MAKT Table:**

It holds Material Description Details

**Fields:**

1. Matnr -------- Material Number
2. Spras -------- Language Key
3. Maktx ------- Material Description

**TSPAT Table:**

It holds Division Description Details

**Fields:**

1. Spras --------- Language Key
2. Spart --------- Division Code
3. Vtext --------- Division Description

**SD Module:**

Sales & Distribution Module

**Departments in SD Module:**

1. Marketing
2. Excise
3. Billing
4. Finance part of SD

**Process Flow:**

Inquiry

Quotation

Sales Order

Delivery

Excise Invoice

Invoice

FI Module

**T Codes & Tables:**

**Inquiry:**

**Tcodes:** VA11, VA12, VA13

**Tables:** VBAK, VBAP

**Note:** The key field is vbeln

**Quotation:**

**Tcodes:** VA21, VA22, VA23

**Tables:** VBAK, VBAP

**Note:** The key field is vbeln

**Sales Order:**

**Tcodes:** VA01, VA02, VA03

**Tables:** VBAK, VBAP

**Note:** The key field is vbeln

**Delivery:**

**Tcodes:** Vl01N, VL02N, VL03N

**Tables:** LIKP, LIPS

**Note:** The key field is vbeln (Delivery Number)

**Excise Invoice:**

**Tcodes:** J1IS, J1IEX, J1ID

**Tables:** J\_1IEXCHDR, J\_1IEXCDTL, J\_1IMOCUST

**Note:** The key field is exnum

**Invoice:**

**Tcodes:** VF01, VF02, VF03

**Tables:** VBRK, VBRP

**Note:** The key field is vbeln (Invoice Number)

**Other Tables:**

VBKD, VBFA, VBPA, VBUK, VBUV, KONV, KONR

**String Operations:**

1. SHIFT
2. TRANSLATE
3. REPLACE
4. STRLEN
5. OFFSET FUNCTIONALITY
6. SPLIT
7. CONCATINATE
8. CONDENSE

**SHIFT keyword:**

It shifts a specific string to left (or) right side by deleting required options in a particular string

**Syntax-1:**

SHIFT <string> LEFT DELETING LEADING <option>

**Syntax-2:**

SHIFT <string> RIGHT DELETING TRAILING <option>

**Example:** ‘000001000’----------- 1000

DATA: str(10) TYPE C VALUE ‘000001000’.

SHIFT str LEFT DELETING LEADING ‘0’ .

WRITE:/ str.

**Output:** 1000

**Note:**

Size of the string variable must be equal to given string

**TRANSLATE keyword:**

It converts the case of a particular string from lower to upper and vice-versa

**Syntax-1:**

TRANSLATE str TO UPPER CASE

**Syntax-2:**

TRANSLATE str TO LOWER CASE

**Example:** SATYA--------satya

DATA: str(10) TYPE C VALUE ‘SATYA’.

TRANSLATE str TO LOWER CASE.

WRITE:/ str

**Output:**

satya

**REPLACE keyword:**

It replaces a string with another string based on requirement

**Syntax:**

REPLACE <string1> WITH <string2> INTO <final string>

**Example:** rupees-------dollars

DATA: str(30) TYPE C VALUE ‘one thousand rupees’.

REPLACE ‘rupees’ WITH ‘dollars’ INTO str.

WRITE:/ str.

**Output:**

one thousand dollars

**STRLEN keyword:**

It returns length of a string

**Syntax:**

n = STRLEN ( str ).

Where n is Integer Variable

**Example:**

DATA: str(10) TYPE C VALUE ‘satya’,

n TYPE I.

n = STRLEN ( str ).

WRITE:/ n.

**Output:**

5

**OFFSET FUNCTIONALITY keyword:**

0 1 2 3 4 5 6 7 --------offset

A B C D E F G H

1. st = str + 2 ( 1 ). --------output------ C
2. st = str + 4 ( 2 ). --------output------ EF

**SPLIT keyword:**

It splits a single string based on special charectors

**Syntax:**

SPLIT <string> AT <special charector> INTO <stirng1> <string2> <string3>…..

**Example:**

DATA: str(10) TYPE C VALUE ‘cool-drink’,

str1(10) TYPE C,

str2(10) TYPE C.

SPLIT str AT ‘-’ INTO str1 str2.

WRITE:/ str1,

str2.

**Output:**

cool

drink

**CONCATENATE keyword:**

It concatenates two (or) more than two strings into a final string

**Syntax:**

CONCATENATE <str1> <str2> <str3> INTO <final string> SAPARATED BY <special character>.

**Example-1:**

DATA: data(10) TYPE C.

CONCATENATE SY-DATUM + 6(2) SY-DATUM + 4(2) SY-DATUM + 0(4) INTO date SAPARATED BY ‘.’.

WRITE:/ date.

**Output:**

20.09.2014

**Example-2:**

DATA: year(4),

mon(2),

day(2),

date(10).

year = SY-DATUM + 0(4).

mon = SY-DATUM + 4(2).

day = SY-DATUM + 6(2).

CONCATENATE day mon year INTO date SAPARATED BY ‘.’.

WRITE:/ date.

**Output:**

20.09.2014

**CONDENSE keyword:**

It combines a specific string by removing gaps between them

**Syntax:**

CONDENSE <string> NO-GAPS.

**Example:**

DATA: str(20) TYPE C VALUE ‘satya narayana’.

CONDENSE str NO-GAPS.

WRITE:/ str.

**Output:**

satyanarayana

**Assignment:**

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

**Looping Statements:**

**Branching:**

1. Conditional Statements
2. Unconditional Statements

**Conditional Statements:**

1. **IF…………ENDIF.**

**Sytax-1:**

IF <condition>.

Do this

ENDIF.

**Sytax-2:**

IF <condition>.

Do this

ELSE.

Do this

ENDIF.

**Sytax-3:**

IF <condition>.

Do this

ELSE IF <condition>

Do this

ELSE.

Do this

ENDIF.

**Note:**

Use ELSE IF condition only a single condition is true

1. **CASE…………..ENDCASE**

**Syntax:**

CASE <expression>.

WHEN <exp1>.

Do this

WHEN <exp2>.

Do this

……….

……….

WHEN OTHERS.

Do this

ENDCASE.

1. **WHILE statement:**

**Syntax:**

WHILE <condition>

----------

----------

ENDWHILE.

**Unconditional Statements:**

1. **DO…………….ENDDO.**

**Syntax-1:**

DO.

---------

---------

ENDDO.

**Syntax-2:**

DO <n> TIMES.

---------

---------

ENDDO.

**Syntax-3:**

DO <n> TIMES VARYING <expression> FROM <m> NEXT <n>.

---------

---------

ENDDO.

**Operators:**

1. Logical Operators
2. Relational Operators

**Logical Operators:**

1. AND
2. OR
3. NOT

**Relational Operators:**

1. GT----- >
2. LT ----- <
3. GE ---- >=
4. LE ---- <=
5. NE ------ <>
6. EQ ------ =

**CONTINUE keyword:**

Whenever CONTINUE is executed it terminates the current LOOP pass & returns the control to next LOOP pass

**Example:**

DATA rem TYPE I.

DO 20 TIMES.

rem = SY-INDEX MOD 2.

IF rem NE 0.

CONTINUE.

ENDIF.

WRITE:/ SY-INDEX.--------- current LOOP pass number

ENDDO.

**Output:**

2 4 6 8 …………………… 20

**CHECK keyword:**

If check expression is true it will allows the remaining LOOP pass else it will terminates the current loop pass

**Syntax:**

CHECK <expression>.

**Example:**

DATA rem TYPE I.

DO 20 TIMES.

rem = SY-INDEX MOD 2.

CHECK rem = 2.

WRITE:/ SY-INDEX.--------- current LOOP pass number

ENDDO.

**22-09-2014**

**EXIT keyword:**

When the keyword EXIT is executed the whole loop will be terminated

**Example:**

DATA: n TYPE I VALUE 10.

DO n TIMES.

IF SY-INDEX >= n.

EXIT.

ENDIF.

WRITE:/ SY-INDEX.

ENDDO.

**Mathematical Operations:**

**Ex:** 5.55

1. FRAC -------- 0.55
2. CEIL --------- 6.0
3. FLOOR ----- 5.0
4. SIGN -------- 1 (+)
5. ABS --------- 5.55 (always returns only positive value)
6. TRUNC ---- 5.0

**Ex:** 6 / 2

1. REM -------- 0
2. DIV --------- 3
3. MOD ------- 0

**FRAC:**

It returns fraction part from a decimal value

**CEIL:**

It returns highest value in existing decimal value

**FLOOR:**

It returns lowest value from a particular decimal value

**SIGN:**

It returns the signature (positive (or) negative) from a particular numeric value

**ABS:**

It returns absolute value from a numeric value

**TRUNC:**

It truncates (removes) decimal part from a numeric value

**REM (or) MOD:**

These returns remainder

**DIV:**

It returns devisor

**Example:**

DATA: n TYPE P DECIMALS 2 VALUE ‘5.55’,

m TYPE P DECIMALS 2.

m = FRAC(n). WRITE:/ m

m = TRUNC(n). WRITE:/ m

m = CEIL(n). WRITE:/ m

m = FLOOR(n). WRITE:/ m

m = ABS(n). WRITE:/ m

m = SIGN(n). WRITE:/ m

**Question:** What is the difference between SY-TABIX & SY-INDEX?

**Answer:**

SY-TABIX returns current loop pass for Internal Tables

i.e. LOOP………..ENDLOOP

SY-INDEX returns current loop pass other than LOOP……….ENDLOOP

i.e. DO…………..ENDDO.

WHILE……………ENDWHILE.

**Assignments:**

1. Formats
2. 1
3. 2

1 2 3

1 2 3 4

1 2 3 4 5

1. \*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

1. \*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

1. Write a program to print prime numbers between 1 to 100?
2. Write a program to accept two numbers and returns HCF & LCM of that numbers?
3. Accept a string and check whether it is palindrome (or) not?

**SAP and ABAP Memory:**

**ABAP Memory:**

* Transferring data between two programs is done using ABAP memory
* Transferring data from ABAP report to other Standard Application is done by using SAP memory

**Export Import Functionality:**

**ZExport** **ZImport**

S\_matnr to S\_matnr to

S\_werks to S\_werks to

S\_budat to SUBMIT ZExport

SELECT……………. WITH matnr IN S\_matnr

……………………. WITH werks IN S\_werks

Exporting list to memory and return

EXPORT it TO MEMORY IMPORT it FROM MEMORY ID ‘material’.

ID ‘material’.

**Question:** Explain why you written Export Import Functionality in your real time?

**Answer:**

* The above functionality is for reusability
* Sometimes we needs to write a logic which already provided by a program
* Instead of writing the same program from beginning you can use the above functionality for getting the data from other programs
* I written a program for calculating Ageing for Materials
* In that I need to get Closing Stack of a material which is already provided by a standard report MB5B
* I simply passed input parameters from my Z-program to MB5B using SUBMIT keyword
* In MB5B I written logic for Exporting data to my Z-program & I Imported data using IMPORT keyword to my Z-program

**Example:**

* Go to SE38
* Program: YIMPORT\_PROG
* Create
* Title: Import Program
* Type: Executable Program
* Save, Local Object

TABLES: EKPO.

SELECTION-SCREEN: BEGIN OF BLOCK b1 WITH FRAME TITLE text-000.

SELECT-OPTIONS: s\_matnr FOR EKPO-matnr,

s\_werks FOR EKPO-werks.

SELECTION-SCREEN: END OF BLOCK b1.

TYPES: BEGIN OF ty\_ekpo,

ebeln TYPE ebeln,

matnr TYPE matnr,

menge TYPE menge\_D,

netpr TYPE netpr,

END OF ty\_ekpo.

DATA: wa\_ekpo TYPE ty\_ekpo,

It\_ekpo TYPE TABLE OF ty\_ekpo.

SUBMIT YEXPORT\_PROG WITH matnr IN s\_matnr

WITH werks IN s\_werks

EXPORTING LIST TO MEMORY AND RETURN.

IMPORT it\_ekpo FROM MEMORY ID ‘matt’.

LOOP AT it\_ekpo INTO wa\_ekpo.

WRITE:/10 wa\_ekpo-ebeln,

30 wa\_ekpo-matnr,

60 wa\_ekpo-menge,

90 wa\_ekpo-netpr.

ENDLOOP.

* Double click on YEXPORT\_PROG
* Yes
* Remove With Top INCL check box
* Enter, Enter, Enter

TABLES: EKPO.

SELECTION-SCREEN: BEGIN OF BLOCK b1 WITH FRAME TITLE text-000.

SELECT-OPTIONS: s\_matnr FOR EKPO-matnr,

s\_werks FOR EKPO-werks.

SELECTION-SCREEN: END OF BLOCK b1.

TYPES: BEGIN OF ty\_ekpo,

ebeln TYPE ebeln,

matnr TYPE matnr,

menge TYPE menge\_D,

netpr TYPE netpr,

END OF ty\_ekpo.

DATA it\_ekpo TYPE TABLE OF ty\_ekpo.

SELECT ebeln matnr menge netpr FROM EKPO INTO TABLE it\_ekpo

WHERE matnr IN s\_matnr

AND werks IN s\_werks.

EXPORT it\_ekpo TO MEMORY ID ‘matt’.

**Output:**

Execute IMPORT\_PROG

s\_matnr 100-100 to 100-500

s\_werks 1000 to 1100

**23-09-2014**

**Support Project:**

* In Support Project End user will raise Tickets (Issues) based on priority
* Issues are directly raised by End user in Production Client which are automatically updated to Solution Manager
* Solution Manager is a free Ticketing Tool given by SAP for raising tickets
* Help Desk people from Consulting Company will receive the Tickets & forward it to respective Functional Consultants based on priority
* If the issue is Functional (related to Application, Customizing Issue……….etc) then Functional Consultant will resolve it
* If the issue is Technical (related to Programming, Enhancement…….etc) then Functional Consultant will prepare Functional Document & forward it to Technical Consultant (ABAPER)
* Based on Functional Document ABAPER will create Technical Document & forward it to Team Leader for approval
* If everything is fine ABAPER will develop the Object in Development Client
* Once the Object is developed then series of Testing will done by ABAPER, Functional Consultant & End user
* Finally after final testing by End user the Object is transported to Production System with the help of Basis Consultant

**System Landscape:**

* It is also known as Correction & Transportation System (CTS)
* In real time End user will raise issues which are received by Functional Consultant
* For Technical Issues Functional Consultant will prepare Functional Document & based on that Technical Consultant will prepare Technical Document
* Technical Consultant will forward Technical Document to Team Leader who will approve it
* Objects are developed in Development Client
* While creating Objects system will prompt you for to save the object
* In real time Objects are always saved in a Package
* Once Objects are saved system will generate Request Numbers (Main Request Number & Sub Request Number)
* Objects are always stored in Sub Request Number
* Once Object is developed based on coding standards you should release Request Numbers in SE10
* Then you should forward a mail to Basis Consultant for transporting Transport Request (TR) numbers from Development Client to Quality Client
* In Quality Server testing is done by ABAPER, Functional Consultant, End user
* If everything is fine again you will forward a mail to Basis Consultant to transport the TR Numbers from Quality Server to Production Server

**Important Points to Remember:**

1. In real time you will receive tickets from FC. If you are working in Client Place you will get Tickets directly from End user
2. Before creating TD you gather the same from your TL
3. In Development Client Objects are created & modified. Testing is not possible here as recent data not available here (only test data is available)

**Note:**

Never discus with an interviewer that your Development Client is 800. It is always other than 800 i.e. 100, 200, 300,………….

1. Quality Server is meant for testing. Here recent data (10 days back) is available
2. Production Server is Live Server in real time. It holds live data of a company. Day to day activities of a company are maintained (or) stored in Production Server
3. Only Development Client is in change mode. Quality & Production Clients are always in display mode
4. In real time as an ABAPER you can access Development & Quality Clients
5. End user can access Quality & Production Clients
6. FC can access any of these Clients
7. Package is a collection of Objects. In real time we are not creating any package. Throughout the life time in a company we are working with a single package
8. Objects are always stored in Sub Request Number is transported from one client to another
9. In real time Basis Consultant is responsible for transporting Request Numbers from one client to another
10. Once a Request Number is released (or) transported system will prompt you for a new Request Number
11. In real time as a TC we have to work with Work Bench Request. FC will work with Customizing Request
12. Request Numbers are always forwarded from Development to Quality & Quality to Production Clients

**Note:**

Reverse transportation of a Request Number is not possible under any circumstances in SAP

1. In real time all the issues are posted in Production System & Threads are updated in Solution Manager
2. E071 is a standard table which holds all Objects & their Request Numbers history
3. In real time our job to create Object & release the Request Number in SE10
4. In real time different companies are having different System Landscapes. In some companies there will 1 Development & 3 Testing Clients & in some 1 Development & 1 Testing Client & in some companies Development & Quality Clients are Cross Clients
5. In Quality Server only 85% to 90% of testing is done as it holds recent data. If you want to test the live data you have to work with Production Server

**24-09-2014**

**Steps in System Landscape:**

1. Work with SE80 (create a Package)
2. Work with SE38 (create a Program)
3. Work with SE10 (release Sub Request Number, release Main Request Number)

**Step-1:**

* Go to SE80
* Select Package option
* Provide Package name: ZABAP
* Yes
* Short Description: New Package
* Save

(OR)

* Go to SE11
* Provide Database Table name: V\_TDEVC
* Select Display option
* Select Contents option
* Select Package option
* Provide Package name: ZABAP
* Short Description: New Package
* Save

**Step-2:**

* Go to SE38
* Program: ZDEMO\_PACKAGE
* Create
* Title: Package Program
* Type: Executable Program
* Save
* Package: ZABAP
* Save (Enter)

**Note:**

1. By default you may find a Request Number. The Request Number belongs to some other ABAPER
2. Never select Enter option on that pop-up

* Select Create Request (F8)
* Short Description: HR/ABAP: Employee Details Report (24.09.2014)
* Enter
* Enter

WRITE:/ ‘Welcome to ABAP’.

* Activate program (Ctrl+F3)

**Step-3:**

* Go to SE10
* Select Work Bench Request
* Under Request Status, select Modifiable check box & deselect Released check box
* Select Display option
* Expand Main Request Number
* Select Sub Request Number
* Select Truck Symbol (F9)
* Select Main Request Number
* Select Truck Symbol (F9)
* Go to Back (F3)
* Select Released check box & deselect Modifiable check box
* Select Display option

**Note:**

Up to here you have to perform in real time, from here you have to send a mail to your Basis Consultant for transporting TR-Number from Development Client to Quality Client

* Once the Request Number is transported from Development Client to Quality Client as a TC now you have to login to Quality Server & test the Object
* If it is fine now you will send a mail to your FC to test the same
* If it is fine FC will update the Tread in Solution Manager to test the Object
* Similarly End user will perform final testing in Quality Server

**Note:**

Always store identical Objects in one Request Number & different Objects in different Request Numbers

**Version Management:**

* Using Version Management you can get the history of a program with Request Numbers, User name, Date of creation & change
* It is highly useful to programmer if requires to retrieve old version instead of current version

**Example:**

* Go to SE38
* Program: ZDEMO
* Select Utilities -----------🡪 Versions ------🡪 Version Management

**Ques:** How you can store an Object from Local Object to existing Package?

**Ans:**

* Go to SE38
* Program: name which is stored in Local object
* Select Goto ----🡪 Object Directory Entry
* Select Change option
* Package: ZABAP
* Save

**Ques:** How you can delete an Object from existing Request Number?

**Ans:**

* Go to SE11
* Database Table: E071
* Display
* Contents option
* Object Name; program name (ZDEMO)
* Execute (F8)
* Go to SE10
* Select Display
* Expand Main Request Number
* Expand Sub Request Number
* Expand Program option
* Select your program
* Select Delete option
* You will find a message, Enter
* Yes

**Sub Objects in ABAP Editor:**

1. Source Code
2. Variants
3. Attributes
4. Documentation
5. Text Elements

**Source Code:**

It is nothing but source code of a program

**Attributes:**

It specifies attributes of a program (Title, Type, Date of creation & Change,………..etc)

**Documentation:**

Using this option you can provide documentation for a program

**Example:**

* Go to SE38
* Program: ZDEMO
* Documentation
* Change
* Provide Purpose: The report is displaying segment wise materials details
* Provide Integration: No
* Provide Prerequisites: s\_matnr

s\_werks

* Save

**Text Elements:**

Using Text Elements you can maintain Labels (or) Selection Screen Fields

**Example:**

* Go to SE38
* Program: ZDEMO
* Text Elements
* Change
* Selection Text

Name Text

s\_matnr Material Code

s\_werks Plant

* Activate

**Variants:**

* Using Variants you can provide default input values for a Selection Screen
* In background scheduling report, the reports are executed automatically by system in background. For such reports the input values are supplied by using Variants

**Example:**

* Go to SE38
* Program: ZDEMO
* Variants
* Change
* Variant: ZVAR
* Create
* Enter input values

i.e. 100-100 to 100-500

fart

* Select Attribute option (F6)
* Meaning: Variant Creation
* Save
* Run the report
* Select Get Variant option
* Double click on variant name (ZVAR)

**ABAP-EDITOR**

**Interview Questions & Answers:**

**1Q:** What is ERP?

**Ans:** ERP is a package under which all the business resources are integrated in one system

**2Q:** When did ABAP Consultant meets Clients?

**Ans:**

* ABAP Consultant will not meet Clients
* Only Functional Consultant will meets Clients
* ABAP Consultant only meets Functional Consultant

**3Q:** What are the Services in R/3 Architecture?

**Ans:** There are 9 services in R/3 Architecture

1. Dialog service
2. Update service
3. Spool service
4. Background service
5. En-Queue service
6. Message service
7. Gateway service
8. SAP GUI front end service
9. Database service

**4Q:** Where you can also execute programs other than ABAP Editor (SE38)?

**Ans:** We can also execute programs in Object Navigator (SE80)

**5Q:** What is the default Data type in ABAP Programming?

**Ans:** Character (C)

**6Q:** What are the SAP provided Business Data types?

**Ans:** QUAN, CURR, NUME

**7Q:** What is : operator in ABAP programming?

**Ans:** It is called Chain Operator. It is for repeating Declarative Part

**8Q:** Which file is generated by system when you execute the program?

**Ans:** SAPGUI.EXE file

**9Q:** What is purpose of Activating the Objects?

**Ans:**

* To store the Objects in Database Server
* To integrate the Objects with each other
* To debug & transport the objects

**10Q:** What is the T Code for Debugging?

**Ans:** /H

**11Q:** How many sessions you can open simultaneously in SAP?

**Ans:** 6 sessions

**12Q:** What is the use of Internal Table in SAP?

**Ans:** For storing different types of application data from different tables at one place

**13Q:** What is the difference between Database table & Internal table?

**Ans:** **Database Table** **Internal Table**

1. It is created in Database Server ------- 1. It is created in Application Server
2. It holds data permanently ------- 2. It holds data temporarily
3. It holds only specific type of data ----- 3. It holds data of different tables of at one place

**14Q:** What is the default comparison operator for SELECT-OPTIONS?

**Ans:** ‘IN’

**15Q:** What is the default comparison operator for PARAMETERS?

**Ans:** ‘EQ’

**16Q:** What is the default order of SORT?

**Ans:** ASCENDING

**17Q:** What is difference between APPEND & COLLECT?

**Ans:**

* APPEND keyword appends the record from Work Area to Body (always to next line in an Internal Table)
* COLLECT keyword performs append job if the record fails to exist in the Body
* If already records exists it adds all Numeric Values on right side of Character Field inside the Body

**18Q:** When you will use COLLECT keyword in real time?

**Ans:** Whenever I will get a requirement for providing consolidated totals based on Customers,

Vendors, Materials,………etc

**19Q:** Suppose there are more than two Character Fields then what is the roll of COLLECT

keyword?

**Ans:** If there are more than two Character Fields it will collect the data (adds) only when both

the Character Fields are same

**20Q:** What is difference between APPEND & MODIFY keywords?

**Ans:**

* APPEND always appends record to Internal Table (if record exists (or) not)
* MODIFY always modifies (adds field value) only if record exists in an Internal Table

**21Q:** When you will prefer MODIFY keyword in ABAP programming?

**Ans:** When all the data in an Internal Table is filled and a new field value is required to be

added in an Internal Table

**22Q:** What will happen if CLEAR is providing before Work Area & the same CLEAR is

provided in after Work Area?

**Ans:**

* If CLEAR is provided before Work Area data so nothing is transferred from Work Area to Virtual Page (hence no output)
* If CLEAR is provided after Work Area, data is transferred from Work Area to Virtual Page then Work Area is cleared (hence you will find output)

**23Q:** What is difference between SUM & COLLECT keyword?

**Ans:**

* COLLECT keyword is applicable with in LOOP………ENDLOOP without usage of Control Break Statements
* COLLECT keyword performs totals inside internal body based on one (or) more than one Character Fields
* SUM keyword performs totals only on Single Character Field for Control Break Statements & places the values in Work Area
* COLLECT keyword works outside LOOP………ENDLOOP also where as SUM can be applied only within LOOP…….ENDLOOP

**24Q:** What is default Internal Table in SAP?

**Ans:** Standard Table

**25Q:** What is the prefix for System Variables?

**Ans:** SY

**INTERNAL TABLES**

**1Q** : How to delete duplicates from an internal table?

**Ans**:

* To delete all duplicate entries from a sorted internal table (e.g. just after SORT), you can use the DELETE ADJACENT DUPLICATES FROM i\_tab statement.
* You can use the COMPARING addition to limit the fields that are used to test for duplicate entries

**Ex:** SORT i\_tab by matnr werks logort.

DELETE ADJACENT DUPLICATES FROM i\_tab COMPARING matnr werks.

All duplicates with same combination of matnr and werks will be deleted

**2Q** : What are types of internal tables?

**Ans**:

There are three types of Internal Tables exist

1. **Standard Internal Tables:**

* These tables have a linear index and can be accessed using the index or the key.
* The response time is in linear relationship with number of table entries.
* These tables are useful when user wants to address individual table entries using the index.

1. **Sorted Internal Tables:**

* These tables also have an index and the key.
* But, the response time is in logarithmic relationship with number of table entries, since it uses binary search algorithm instead of linear search.
* These tables are useful when user wants the table to be sorted while additional entries have to be added.

1. **Hashed Internal Tables:**

* These tables have no index, but have the key.
* The response time is constant irrespective of number of table entries, since it uses a Hash algorithm.
* These tables are useful when user wants to access the entries with key only.

**3Q** : What is the size of internal table?

**Ans**:

* The size of the internal tables is set using the 'OCCURS n' clause.
* Here ‘n’ refers to a integer number that specifies the size.
* Usually it given as 'OCCURS 0' which creates an internal table with the memory space of 8kb.
* The size of the internal table is the product of width and number of lines of the internal table. We can set the estimated number of lines of internal table in the OCCURS parameter.
* When first table entry is filled the runtime system allocates suitable memory to hold the whole table
* If the memory allocated is not sufficient then again a block of memory is allocated (in case of number of lines more than estimated lines).
* Therefore the OCCURS parameter does not affect the actual size of the internal table but the performance of the internal table operations.
* So if you are not able to estimate the lines, then give occurs parameter to zero.
* Then system automatically does it for you.

**4Q** : What is the effective way of using internal table record?

**Ans**:

* Create Internal table without Header.
* Do all the internal table operations by Creating Work area.
* This would be the efficient way of coding.
* Don't forget to CLEAR Work area whenever it is required.

**5Q** : Explain Sorted Tables in Sap internal tables?

**Ans**:

* This is the most appropriate type if you need a table which is sorted as you fill it.
* You fill sorted tables using the INSERT statement.
* Entries are inserted according to the sort sequence defined through the table key.
* Any illegal entries are recognized as soon as you try to add them to the table.
* The response time for key access is logarithmically proportional to the number of table entries, since the system always uses a binary search.
* Sorted tables are particularly useful for partially sequential processing in a LOOP if you specify the beginning of the table key in the WHERE condition.

**6Q** : What is difference between internal table and structure?

**Ans**: Internal table holds multiple records and structure holds single record during runtime.

**7Q** : What are field groups and internal tables?

**Ans**:

* Field groups are nothing but a similar type of fields grouped under single name (group) or grouping of the fields is called field groups.
* Internal tables are stored temp memory/in buffer for the further processing of the fields.

**8Q** : What is sap internal tables key?

**Ans**:

* The key identifies table rows.

There are two kinds of key for internal table keys

1. Standard key
2. User- defined key.

* You can specify whether the key should be UNIQUE or NON-UNIQUE.
* Internal tables with a unique key cannot contain duplicate entries.
* The uniqueness depends on the table access method.  
  If a table has a structured line type, its default key consists of all of its non-numerical columns that are not references or themselves internal tables.
* If a table has an elementary line type, the default key is the entire line. The default key of an internal table whose line type is an internal table, the default key is empty.  
  The user-defined key can contain any columns of the internal table that are not references or themselves internal tables.
* Internal tables with a user-defined key are called key tables. When you define the key, the sequence of the key fields is significant.
* You should remember this, for example, if you intend to sort the table according to the key.

**9Q** : Explain Hashed tables in sap?

**Ans**:

* This is the most appropriate type for any table where the main operation is key access.
* You cannot access a hashed table using its index.
* The response time for key access remains constant, regardless of the number of table entries.
* Like database tables, hashed tables always have a unique key.
* Hashed tables are useful if you want to construct and use an internal table which resembles a database table or for processing large amounts of data.

**10Q**: What do mean by table types.

**Ans**:

* The table type determines how ABAP will access individual table entries.

Internal tables can be divided into three types:

1. Standard tables

* It has an internal linear index.
* From a particular size upwards, the indexes of internal tables are administered as trees.
* In this case, the index administration overhead increases in logarithmic and not linear relation to the number of lines.
* The system can access records either by using the table index or the key.
* The response time for key access is proportional to the number of entries in the table.
* The key of a standard table is always non-unique.
* You cannot specify a unique key.
* This means that standard tables can always be filled very quickly, since the system does not have to check whether there are already existing entries.

**11Q**: What are control levels in internal tables?

**Ans**:

The control breaks in an Internal Table areControl Levels are of 5 types

1) AT FIRST

2) AT NEW

3) AT END OF

4) AT LAST

5) ON CHANGE OF

**12Q**: How can you specify internal tables as data objects?

**Ans**:

* Data objects that are defined either with the data type of an internal table (or) directly as an internal table, are always fully defined in respect of their line type, key and access method.
* However, the number of lines is not fixed.
* Thus internal tables are dynamic data objects, since they can contain any number of lines of a particular type.
* The only restrictions on the number of lines an internal table may contain are the limits of your system installation.
* The maximum memory that can be occupied by an internal table (including its internal administration) is 2 GB.
* A more realistic figure is up to 500 MB.
* An additional restriction for hashed tables is that they may not contain more than 2 million entries.
* The line types of internal tables can be any ABAP data types - elementary, structured, or internal tables.
* The individual lines of an internal table are called table lines or table entries.
* Each component of a structured line is called a column in the internal table.

**13Q**: What is generic internal table?

**Ans**:

* Unlike other local data types in programs, we do not have to specify the data type of an internal table fully.
* Instead, we can specify a generic construction, that is, the key or key and line type of an internal table data type may remain unspecified.
* We can use generic internal tables to specify the types of field symbols and the interface parameters of procedures.
* We cannot use them to declare data objects.

**14Q**: Explain standard internal tables?

**Ans**:

* This is the most appropriate type if you are going to address the individual table entries using the index.
* Index access is the quickest possible access.
* You should fill a standard table by appending lines (ABAP APPEND statement), and read, modify and delete entries by specifying the index (INDEX option with the relevant ABAP command).
* The access time for a standard table increases in a linear relationship with the number of table entries.
* If you need key access, standard tables are particularly useful if you can fill and process the table in separate steps.
* For example, you could fill the table by appending entries, and then sort it.
* If you use the binary search option with key access, the response time is logarithmically proportional to the number of table entries.

**15Q**: What are internal tables, value tables, check tables, transparent tables?

**Ans**:

**Internal table:**

It is a standard data type object, which exists only during the runtime of the program.

**Check table:**

Check table will be at field level checking.

**Value table:**

Value table will be at domain level checking

**Ex:** scarr table is check table for carrid.

**Transparent table:**

Exists with the same structure both in dictionary as well as in database exactly with the same data and fields

**16Q**: Explain choosing a table type in internal table.

**Ans** : The table type (and particularly the access method) that you will use depends on how the typical internal table operations will be most frequently executed.

**17Q**: Explain row type and line type concept?

**Ans**:

* Line type refers to the structure of an internal table
* Whereas row type is the actual part that contains the data and it refers to the table body
* Creating internal table using line type and row type concept is for reusability purpose.
* Line type and Row type are defined at DDIC LEVEL.

**18Q**: What is the difference between internal table and work area?

**Ans** :

* An internal table is a run time instance.
* It get created when program starts execution.
* \*It get destroyed when program terminates.
* It has two different parts:
* Header Line (optional)
* Body (Compulsory).
* \*Any value that comes to (or) goes from internal table, that travels through header line.

**19Q**: Difference between database tables and internal tables?

**Ans** :

* The basic difference is database tables are stored in DB server
* Internal tables are virtual tables these are created run time only
* Internal tables are created dynamically
* The memory of internal tables is not permanent memory
* For internal tables the memory will be created in the application server and it is external memory and terminates after the program termination.

**20Q**: What is the difference between collect and append?

**Ans**:

* Collect holds summarized data (collect the sums based on name etc)
* Append will appends the data from header line (work area) to body

**DATA DICTIONARY**

**Interview Questions & Answers:**

**1Q:** Have you ever created a Z-Table? Why explain the requirement?

**Ans:** Yes, SAP is not providing table for Allowances. Simultaneously SAP is not providing a table for Transportation Details. Simultaneously in real time we have to create a table for Customer Mail Ids for sending invoice details………….etc

**2Q:** Explain difference between Standard Table & Z-Table?

**Ans:**

* Standard Table is system defined table. It starts with A, B,C,………………….X
* Z-Table is user defined table. It starts with Y (or) Z

**3Q:** What is the difference between Structure & Table?

**Ans:** **Structure** **Table**

1. It is a collection of fields & it 1. It is a Structure & hold data persistently

Never hold data

1. It can be reuse in other tables 2. It cannot be reuse

**4Q:** What is the difference between View & Table?

**Ans:** **Table** **View**

1. Having its own fields & data 1. Always created based on other table
2. It holds either master data (or) 2. It holds both master data & transactional data

Transactional data

**5 Q:** What is the difference between Top-Bottom & Bottom-Top approach of crating tables?

**Ans: Top-Bottom:**

Table---🡪Data Element----🡪Domain

**Bottom-Top:**

Domain-----🡪Data Element----🡪Table

**6 Q:** What is a View? Explain the importance of View?

**Ans:**

* It is a logical table (virtual table) which is created based on other tables
* It hires Fields & Data from other tables
* It is used to generate reports based on user required fields

**7 Q:** What is the difference between Database View & Projection View?

**Ans:**

* Database View is always created based on two (or) more than two tables having relationship between them
* Projection View is created based on a single table with specific fields

**8 Q:** What is the difference between Open SQL statements & Native SQL statements?

**Ans:** **Native SQL Statements:**

* These are traditional SQL statements
* With Native SQL Statement request is directly forwarded to Database Server which processes the request & returns the response
* With Native SQL statement performance of the system is highly degraded

**Syntax:**

EXEC SQL.

SELECT………………

ENDEXECSQL.

**Open SQL Statements:**

These are normal SQL statements which are converted to embedded SQL using database interface which is further passed to Buffer, if data is available in Buffer you will get the response

**9 Q:** Explain Buffering & its advantages?

**Ans:** Buffering is a way of improving performance which accessing data from database. Whenever you run a program it will make a request to Buffer for data, if it is not available it will make a request to Database Server which returns response to Buffer & simultaneously to Program. Next time whenever you run the same program this time it will collect the data directly from Buffer

**10 Q:** What are the different types of Tables?

**Ans:** There are 3 types of tables in SAP

1. Transparent Table
2. Cluster Table
3. Pooled Table

**11Q:** Mention some Cluster Tables that you are using in real time?

**Ans:** These tables hold SAP proprietary format data (system data, statistical data, control

data……)

**Example:** BSEG, KONU

**12Q:** What is the difference between .INCLUDE & APPEND STRUCTURE?

**Ans:**

* Using .INCLUDE you can add a Structure in N no. of tables
* .INCLUDE is best for adding Structure in ZEE tables
* Using APPEND STRUCTURE you can add a Structure only in a single table
* APPEND STRUCTURE is best for adding Structure in Standard Tables

**13 Q:** How many tables you can reuse a Structure in other tables?

**Ans:** N number of tables

**14 Q:** How many times APPEND STRUCTURE can be used by other tables?

**Ans:** only One time

**15 Q:** What is Database Utility?

**Ans:**

* Using Database Utility you can reactivate a table (adjust a table in database)
* The Tcode is SE14
* Sometimes you may find errors when you change the Primary key (or) Domain name, it Data type…….etc you will find an error Structure Level Changes
* To adjust such errors select Utilities----🡪Database Object---🡪Database Utility
* Scroll down to bottom select Activate and Adjust Database

**16 Q:** What is the difference between Data Element & Domain?

**Ans: Data Element:**

Using Data Element you can provide field documentation for a specific field

Conceptually Data Element = Field Lable + Domain

**Domain:**

It specifies the technical attributes of a field (Data Type & Field Size)

**17 Q:** What are main functionalities of Domain?

**Ans:**

* Maintains Data type
* Technical attributes of Fields
* Value Range

**18Q:** What is Value Range under Domain option?

**Ans:** Using Value Range you can maintain fixed values and can be called in other programs as a Search Help

**19Q:** What is the Search Help? How you can maintain a search to specific field in a table?

**Ans:** Using search help you can search application data based on given search criteria

Search help works based on F4 function key

**20Q:** It is possible to create a field without Data Element?

**Ans:** Yes, using Predefined Type option

But it is always advisable to create with Data Element only

**21Q:** What is the difference between Client Dependent Table & Client Independent Table?

**Ans:**

* A table created without MANDT field is Client Independent table. Client Independent tables are available in other clients also with data
* A table created with MANDT field is Client Dependant table. Client Dependent tables are available in other clients also without data

**22Q:** It is possible to create a table without Primary key?

**Ans:** No

**23Q:** What is the difference between Master Data & Transactional Data?

**Ans:** **Master Data:**

It is created only once in Client Database & modified very rarely

**Example:**

Bank Master Data, Customer Master Data, Vendor Master Data, Employee Master Data…etc

**Transactional Data:**

* It is always created based on Master Data
* It is modified frequently in SAP database

**Example:**

Purchase Order Data, Sales Order Data………..etc

**24Q:** Define Data Class & Size Category?

**Ans:** **Data Class:**

It is a physical storage location (or) table space where tables are stored ultimately after activation. It is of 3 types

1. APPL0 (holds master data tables)
2. APPL1 (holds transactional data tables)
3. APPL2 (holds organizational & customizing data related tables)

**Size Category:**

It specifies the probable space requirement for a table in the database

**25Q:** What is TMG? What is the need of creating TMG in real time?

**Ans:**

* Using TMG you can directly Insert, Update, Delete data from a database table
* In real time for certain requirements when data is not linked with other tables. Data can be inserted (or) manipulated from table level directly. This can be done by creating a TMG on that table

**26Q:** It is possible to create TMG on Standard Tables?

**Ans:** No, TMG is allowed only for Z-tables

**27Q:** How you can create List Boxes & Check Boxes in TMG?

**Ans:** CHK ZCHK\_DAT

* Double click on ZCHK\_DAT
* Save Yes
* You will find a warning pop-up: Enter
* You will find a pop-up: create data element Yes
* Provide Short Description: Check Box
* Provide Domain: ZCHK\_DOMAIN
* Provide Data type: CHAR
* Provide No.of Characters: 1
* Select Value Range tab

FIX VAR Short Desciption

X ON

OFF

* Activate the domain (Ctrl+F3)

**28Q:** How you can provide Default Values such as Date & Time automatically in a table whenever user select Save option?

**Ans: Providing default values using TMG:**

* Select Utilities---🡪Table Maintenance Generator
* Provide Authorization Group: &NC&
* Provide Function Group: ZMAIL\_GR
* Select Find Scr Number option then press Enter
* Select Create option (F6) then press Enter
* Select Environment option
* Select Modification------🡪Events
* You will find a message- do not make any changes on SAP data then press Enter
* Select New Entries option provide

T Form Routine

05 Create\_Entry

* Press Enter
* Under Editor double click
* You will find a pop-up then press Enter
* You will find a warnings pop-up then press Enter
* Provide code for default values

FORM CREATE\_ENTRY.

ZFI\_LIMITS-ERDAT = SY-DATUM.

ZFI\_LIMITS-UNAME = SY-UNAME.

ZFI\_LIMITS-ERTIM = SY-UZETT.

ENDFORM.

* Activate table & go back

**29Q:** Mention some important Events that you created with TMG?

**Ans:** CREATE\_ENTRY

**30Q:** What is One Step & Two Step option in TMG?

**Ans:**

* One Step---By these option we can insert all the records at a time
* Two Step---By these option we can insert one by one record

**31Q:** Why Function Group is added while creating TMG?

**Ans:**

**32Q:** What is Authorization Group & Authorization Object in TMG?

**Ans:** Authorization Group is for which group of clients can access table

**Ex:** &NC&

**33Q:** How will add table values to a request number in ABAP?

**Ans:**

**34Q:** What is the difference between SE11 & SE16?

**Ans:**

* SE11 for creating all Data Dictionary Objects in SAP
* SE16 for just viewing data of specific table. Here manipulation of data is not possible

**35Q:** Why it is mandatory to Activate a table in SAP?

**Ans:**

* To store the Objects in Database Server
* To integrate the Objects with each other
* To debug & transport the objects

**36Q:** What is Indexing in SAP? What are types of Indexes?

**Ans:**

**37Q:** What is Lock Object? Have you created Lock Object?

**Ans:** No

**38Q:** What is the difference between Structure & Type Group?

**Ans:** **Structure:**

* It is a group of fields
* Physically it will never hold any data
* Structure can be reuse in other programs & other tables

**Type Group:**

It is a group of Structures

**39Q:** What is Text Table? What is its significance?

**Ans:**

* Use text table for storing explanatory texts. For storing such texts it is not advisable to store in Primary Table
* Text Tables works based on Primary & Foreign key relationship

**40Q:** How you can add a Field in Standard Table?

**Ans:** Using Append Structure option

**41Q:** What are various Dictionary Objects?

**Ans:**

1. Database Table
2. Structure
3. View
4. Data Element
5. Domain
6. Table Type
7. Type Group
8. Lock Object
9. Search Help

**42Q:** What is the significance of Data Dictionary in SAP?

**Ans:**

* It is also called ABAP Dictionary
* It is a repository of data
* T Code for Data Dictionary is SE11

**43Q:** Is it possible to create DDL operations directly in SAP?

**Ans:** No

**1Q :** What are the layers of data description in R/3?  
**Ans:**

• The external layer.  
• The ABAP/4 layer.  
• The database layer.

**2Q** : Define external layer?

**Ans**:

* The external layer is the plane, at which the user sees and interacts with the data,
* That is, the data format in the user interface.
* This data format is independent of the database system used.

**3Q** : Define ABAP/4 layer?  
**Ans**: The ABAP/4 layer describes the data formats used by the ABAP/4 processor.

**4Q** : Define Database layer?  
**Ans**: The database layer describes the data formats used in the database.

**5Q** : What is a Data Class?  
**Ans**: The Data class determines in which table space the table is stored when it is created in the database.

**6Q** : What is a Size Category?  
**Ans**: The Size category describes the probable space requirement of the table in the database.

**7Q** : How many types of size categories and data classes are there?  
**Ans**:

* There are five size categories (0-4) and
* 11 data classes

Only three of size categories are appropriate for application tables:  
1. APPL0- Master data (data frequently accessed but rarely updated)  
2. APPL1- Transaction data (data that is changed frequently)  
3. APPL2- Organizational data (customizing data that is entered when system is configured and then rarely changed)

The other two types are:  
4. USR  
5. USR1 - Intended for customer's own developments.

**8Q** : What are control tables?  
**Ans**: The values specified for the size category and data class are mapped to database-specific values via control tables.

**9Q** : What is the function of the transport system and workbench organizer?  
**Ans**:

The function of the transport system and the Workbench Organizer is to manage any changes made to objects of the ABAP/4 Development Workbench and to transport these changes between different SAP systems.

**10Q**: What is a table pool?  
**Ans** :

* A table pool (or pool) is used to combine several logical tables in the ABAP/4 Dictionary.
* The definition of a pool consists of at least two key fields and a long argument field (VARDATA).

**11Q**: What are pooled tables?  
**Ans** :

* These are logical tables, which must be assigned to a table pool when they are defined.
* Pooled tables can be used to store control data (such as screen sequences or program parameters).

**12Q**: What is a table cluster?  
**Ans** :

* A table cluster combines several logical tables in the ABAP/4 Dictionary.
* Several logical rows from different cluster tables are brought together in a single physical record.
* The records from the cluster tables assigned to a cluster are thus stored in a single common table in the database.

**13Q**: How can we access the correction and transport system?  
**Ans** :

Each time you create a new object or change an existing object in the ABAP/4 Dictionary, you branch automatically to the Workbench Organizer or correction and transport system.

**14Q**: Which objects are independent transport objects?  
**Ans** :

* Domains,
* Data elements,
* Tables,
* Technical settings for tables,
* Secondary indexes for transparent tables,
* Structures,
* Views,
* Match code objects,
* Match code Ids,
* Lock objects.

**15Q**: How is conversion of data types done between ABAP/4 & DB layer?  
**Ans** : Conversion between ABAP/4 data types and the database layer is done within the database interface.

**16Q**: How is conversion of data types done between ABAP/4 & external level?  
**Ans** : Conversion between the external layer and the ABAP/4 layer is done in the SAP dialog manager DYNP.

**17Q**: What are the Data types of the external layer?  
**Ans** : ACCP, Char, CLNT, CUKY, CURR, DATS, DESC, FLTP, INT1, INT2, INT4, LANG, LCHR, LRAW, NUMC, PREC, QUAN, RAW, TIMS, UNIT,VARC.

**18Q**: What are the Data types of the ABAP/4 layer?  
**Ans** : Possible ABAP/4 data types:  
C: Character.  
D: Date, format YYYYMMDD.  
F: Floating-point number in DOUBLE PRECISION (8 bytes).  
I: Integer.  
N: Numerical character string of arbitrary length.  
P: Amount of counter field (packed; implementation depends on h/w platform).  
S: Time Stamp YYYYMMDDHHMMSS.  
V: Character string of variable length, length is given in the first two bytes.  
X: Hexadecimal (binary) storage.

**19Q**: How can we set the table spaces and extent sizes?  
**Ans** : You can specify the extent sizes and the table space (physical storage area in the database) in which a transparent table is to be stored by setting the size category and data class.

**20Q**: What is the function of the correction system?  
**Ans** : The correction system manages changes to internal system components. Such as objects of the ABAP/4 Dictionary

**21Q**: What are local objects?  
**Ans** : Local objects (Dev class$TMP) are independent of correction and transport system.

**22Q**: What is a Development class?  
**Ans** : Related objects from the ABAP/4 repository are assigned to the same development class. This enables you to correct and transport related objects as a unit.

**23Q**: What is a data dictionary?  
**Ans** :

* Data Dictionary is a central source of data in a data management system.
* Its main function is to support the creation and management of data definitions.

It has details about  
• What data is contained?  
• What are the attributes of the data?  
• What is the relationship existing between the various data elements?

**24Q**: What functions does a data dictionary perform?  
**Ans** :

In a data management system, the principal functions performed by the data dictionary are  
• Management of data definitions.  
• Provision of information for evaluation.  
• Support for s/w development.  
• Support form documentation.  
• Ensuring that the data definitions are flexible and up-to-date.

**25Q**: What are the features of ABAP/4 Dictionary?  
**Ans** :

The most important features are:  
• Integrated to aABAP/4 Development Workbench.  
• Active in the runtime environment.

**26Q**: What are the uses of the information in the Data dictionary?  
**Ans** :

The following information is directly taken from the Data dictionary:  
• Information on fields displayed with F1 help.  
• Possible entries for fields displayed with F4 help.  
• Matchcode and help views search utilities.

**27Q**: What are the basic objects of the data dictionary?

**Ans**:  
• Tables  
• Domains  
• Data elements  
• Structures  
• Foreign Keys

**28Q**: What are the aggregate objects in the data dictionary?

**Ans**:  
• Views  
• Match codes  
• Lock objects.

**29Q**: In the ABAP/4 Dictionary Tables can be defined independent of the underlying database (T/F).  
**Ans**: True.

**30Q**: ABAP/4 Dictionary contains the Logical definition of the table.  
**Ans** :

**31Q**: A field containing currency amounts (data type CURR) must be assigned to a reference table and a reference field. Explain.  
**Ans**:

* As a reference table, a system containing all the valid currencies is assigned or any other table, which contains a field with the currency key format.
* This field is called as reference field.
* The assignment of the field containing currency amounts to the reference field is made at runtime.
* The value in the reference field determines the currency of the amount.

**32Q**: A field containing quantity amounts (data type QUAN) must be assigned to a reference table and a reference field. Explain?  
**Ans** :

* As a reference table, a system table containing all the valid quantity units is assigned or any other table, which contains a field with the format or quantity units (data type UNIT).
* This field is called as reference field.  
  The assignment of the field containing quantity amounts to the reference field is made at runtime.
* The value in the reference field determines the quantity unit of the amount.

**33Q**: What is the significance of Technical settings (specified while creating a table in the data dictionary)?

**Ans**:

By specifying technical settings we can control how database tables are created in the database. The technical settings allows us to  
• Optimize storage space requirements.  
• Table access behavior.  
• Buffering required.  
• Changes to entries logged.

**34Q**: What is a Table attribute?  
**Ans**:

The table's attributes determine who is responsible for maintaining a table and which types of access are allowed for the table.

The most important table attributes are:  
• Delivery class.  
• Table maintenance allowed.  
• Activation type.

**35Q**: What is the significance of Delivery Class?

**Ans**:  
• The delivery class controls the degree to which the SAP or the customer is responsible for table maintenance.  
• Whether SAP provides the table with or without contents.  
• Determines the table type.  
• Determines how the table behaves when it is first installed, at upgrade, when it is transported, and when a client copy is performed.

**36Q**: What is the max. no. Of structures that can be included in a table or structure.  
**Ans**: Nine.

**37Q**: What are two methods of modifying SAP standard tables?

**Ans**:  
• Append Structures and  
• Customizing Includes.

**38Q**: What is the difference between a Substructure and an Append Structure?

**Ans**:  
• In case of a substructure, the reference originates in the table itself, in the form of a statement include....  
• In case of an append structure, the table itself remains unchanged and the reference originates in the append structure.

**39Q**: To how many tables can an append structure be assigned.  
**Ans**: One.

**40Q**: If a table that is to be extended contains a long field, we cannot use append structures why?  
**Ans**:

* Long fields in a table must always be located in the end, as the last field of the table.
* If a table has an append structure the append line must also be on the last field of the table.

**41Q**: Can we include customizing include or an append structure with Pooled or Cluster tables?  
**Ans**: No.

**42Q**: What are the two ways for restricting the value range for a domain?

**Ans**:  
• By specifying fixed values.  
• By stipulating a value table.

**43Q**: Structures can contain data only during the runtime of a program (T/F)  
**Ans**: True.

**44Q**: What are the aggregate objects in the Dictionary?

**Ans**:  
• Views  
• Match Code.  
• Lock Object.

**45Q**: What are base tables of an aggregate object?  
**Ans** : The tables making up an aggregate object (primary and secondary) are called aggregate object.

**46Q**: The data of a view is not physically stored, but derived from one or more tables (t/f)  
**Ans**: True.

**47Q**: What are the 2 other types of Views, which are not allowed in Release 3.0?

**Ans**:  
• Structure Views.  
• Entity Views.

**48Q**: What is a Match Code?  
**Ans**:

* Match code is a tool to help us to search for data records in the system.
* Match Codes are an efficient and user-friendly search aid where key of a record is unknown.

**49Q**: What are the two levels in defining a Match Code?

**Ans**:  
• Match Code Object.  
• Match CodeId.

\***50Q**: What is the max no of match code Id's that can be defined for one Match code object?\*  
**Ans**: A match code Id is a one character ID that can be a letter or a number.

\***51Q**: Can we define our own Match Code ID's for SAP Matchcodes?\*  
**Ans** : Yes, the number 0 to 9 are reserved for us to create our own Match Code Ids for a SAP defined Match code object.

**52Q**: What is an Update type with reference to a Match code ID?  
**Ans** :

* If the data in one of the base tables of a match code ID changes, the match code data has to be updated.
* The update type stipulates when the match code is to be updated and how it is to be done.
* The update type also specifies which method is to be used for Building match codes.
* You must specify the update type when you define a match code ID.

**53Q**: Can matchcode object contain Ids with different update types?  
**Ans**: Yes.

**54Q**: What are the update types possible?  
**Ans** :

The following update types are possible:  
• Update type A: The matchcode data is updated asynchronously to database changes.  
• Update type S: The matchcode data is updated synchronously to database changes.  
• Update type P: The matchcode data is updated by the application program.  
• Update type I: Access to the matchcode data is managed using a database view.  
• Update type L: Access to the matchcode is achieved by calling a function module.

**55Q**: What are the two different ways of building a match code object?  
**Ans** :

A match code can be built in two different ways:  
1. Logical structure:

The match code data is set up temporarily at the moment when the match code is accessed. (Update type I, k).

2. Physical Structure:

The match code data is physically stored in a separate table in the database. (Update type A, S, P).

**56Q**: What are the differences between a Database index and a match code?

**Ans**:  
• Match code can contain fields from several tables whereas an index can contain fields from only one table.  
• Match code objects can be built on transparent tables and pooled and cluster tables.

**57Q**: What is the function of a Domain?

**Ans**:  
• A domain describes the technical settings of a table field.  
• A domain defines a value range, which sets the permissible data values for the fields, which refers to this domain.  
• A single domain can be used as basis for any number of fields that are identical in structure.

**58Q**: Can you delete a domain, which is being used by data elements?  
**Ans**: No.

**59Q**: What are conversion routines?

**Ans**:  
• Non-standard conversions from display format to sap internal format and vice-versa are implemented with so called conversion routines.

**60Q**: What is the function of a data element?  
**Ans**: A data element describes the role played by a domain in a technical context. A data element contains semantic information.

**61Q**: Can a domain, assigned to a data element be changed?  
**Ans** : Yes. We can do so by just overwriting the entry in the field domain.

**62Q**: Can you delete data element, which is being used by table fields.  
**Ans** : No.

**63Q**: Can you define a field without a data element?  
**Ans** : Yes. If you want to specify no data element and therefore no domain for a field, you can enter data type and field length and a short text directly in the table maintenance.

**64Q**: What are null values?  
**Ans** : If the value of a field in a table is undefined or unknown, it is called a null value.

**65Q**: What is the difference between a structure and a table?  
**Ans** :

Structures are constructed the almost the same way as tables, the only difference using that no database table is generated from them.

**66Q**: What is a view?  
**Ans** :

* A view is a logical view on one or more tables.
* A view on one or more tables

i.e., the data from a view is not actually physically stored instead being derived from one or more tables.

**67Q**: How many types of Views are there?

**Ans**:  
• Database View  
• Help View  
• Projection View  
• Maintenance View

**68Q**: What is Locking?  
**Ans** : When two users simultaneously attempt to access the same data record, this is synchronized by a lock mechanism.

**69Q**: What is database utility?  
**Ans** : Database utility is the interface between the ABAP/4 Dictionary and the underlying the SAP system.

**70Q**: What are the basic functions of Database utility?  
**Ans** :

The basic functions of database utility are:  
• Create database objects.  
• Delete database objects.  
• Adjust database objects to changed ABAP/4 dictionary definition.

**71Q**: What is Repository Info Systems?  
**Ans** : It is a tool with which you can make data stored in the ABAP/4 Dictionary available

**LFA1 Table:**

This table holds Vendor Master Data

**Fields:**

1. Lifnr-------Vendor Number
2. Land1-----Country of Vendor
3. Name1----Name of Vendor
4. Ort01-----City of Vendor
5. Pstlz------Postal Code of Vendor
6. Adrnr-----Address Number

**Links:**

LFA1~adrnr

ADRC~addrnumber

1. Brsch-----Industry Key

**EKKO Table:**

This table holds Purchase Order Header Data

**Fields:**

1. Ebeln-------Purchase Order Number
2. Bukrs------Company Code
3. Bsart-------Purchasing Document Type
4. Statu-------Status of Purchase Order
5. Lifnr-------Vendor Number

**Links:**

LFA1~lifnr

EKKO~lifnr

1. Zterm------Payment Terms
2. Zbd1p-----Cash Discount Percentage
3. Ekorg------Purchase Organization (Purchase Department)
4. Ekgrp------Purchasing Group
5. Waers------Currency
6. Wkurs------Exchange Rate
7. Bedat-------Purchase Order Date
8. Inco1, Inco2-------International Terms for Carrying Bussiness
9. Knumv------Tax Number

**Links:**

EKKO~knumv

KONU~knumv

1. Kalsm------Pricing Procedure
2. Frgke-------Release Indicator

**EKPO Table:**

This table holds Purchase Document Item Data

**Fields:**

1. Ebeln-------Purchase Order Number

**Links:**

EKKO~ebeln

EKPO~ebeln

1. Ebelp------Item Number
2. Txzo1-----Item Text
3. Matnr-----Material Code
4. Werks-----Plant
5. Lgort------Storage Location
6. Matkl-----Material Group
7. Menge------Purchase Order Quantity
8. Meins------Unit of Measurement
9. Netpr------Unit Price
10. Netwr------Total Price
11. Mwskz-----Sales Tax Code
12. Elikz--------Delivery Completed Indicator

**MKPF Table:**

It holds Material Document Header Data

**Fields:**

1. Mblnr ---------- Material Document Number
2. Mjahr ---------- Material Document Year
3. Blart ---------- Material Document Type
4. Budat ---------- Material Document Date
5. Xblrn ---------- Reference Document (Purchase Order Document) Number

**MSEG Table:**

It holds Material Document Item Data

**Fields:**

1. Mblnr ------ Material Document Number
2. Mjahr ------ Material Document Year
3. Bwart ------ Moment Type
4. Werks ------ Plant
5. Lgort ------- Storage Location
6. Charg ------ Batch Number
7. Lifnr ------- Vendor Account Number

**Link:**

LFA1-lifnr

MSEG-lifnr

1. Shkzg ------- Debit / Credit Indicator
2. Dmbtr ------ Amount in Local Currency
3. Menge ------ Quantity
4. Meins ------- Unit of Measurement
5. Ebeln ------- Purchase Order Number

**Link:**

EKPO-ebeln

MSEG-ebeln

1. Ebelp -------- Item Number

**Link:**

EKPO-ebelp

MSEG-ebelp

1. Smbln ------- Number of Material Document Number
2. Kostl -------- Department
3. Aufnf ------- Production Order Number

**MARA Table:**

It holds Material Master Data

**Fields:**

1. Matnr -------- Material Number
2. Ersda --------- Date of Creation
3. Ernam -------- Person who Created that Material
4. Mtart --------- Material Type
5. Mbrsg -------- Industry Key
6. Matkl --------- Material Group
7. Bismt --------- Old Material Number
8. Miens --------- Unit of Measurement
9. Brgew -------- Gross Weight
10. Ntgew -------- Net Weight
11. Spart ---------- Division Code

**MAKT Table:**

It holds Material Description Details

**Fields:**

1. Matnr -------- Material Number
2. Spras -------- Language Key
3. Maktx ------- Material Description

**TSPAT Table:**

It holds Division Description Details

**Fields:**

1. Spras --------- Language Key
2. Spart --------- Division Code
3. Vtext --------- Division Description