NAME: _____

SAP-ABAP

Notes

(EPIC Technologies)

ERP :- Enterprice Resource planning

Enterprice :- is a huge business organization

Resources :-
- 5M Methodology
  - Morely → Finance, Accounts, courting / controlling
  - Material → MM, Stock, inventory, purchasing.
  - Man power → HR
  - Mechanary → PP, PM, QA
  - Marketing → SD
  - Methods → CRM, SCM, SRM

Planning :- often optional citilization of Resources, for a lage business organization.

ERP products :
1. BAAN : Language is Tools
2. People SoFT : Language used is people code.
3. Oracle applications : OF, ON, OHRMS, Long is SQL.
4. SAP R/3 : Languge is ABAP and which have SAP Table
5. NaviGEN
6. JD Edwards : Language which is one word.
7. RAMCO : Have maximum No of tables and applications.

Four big org High range ERPs

Non-ERP's : 1. client / server
2. OR/VB

To implement ERP in on organization. it will take

Need of ERP : ERP is used to planning ow Resources to get goes Profiles with less efforts.

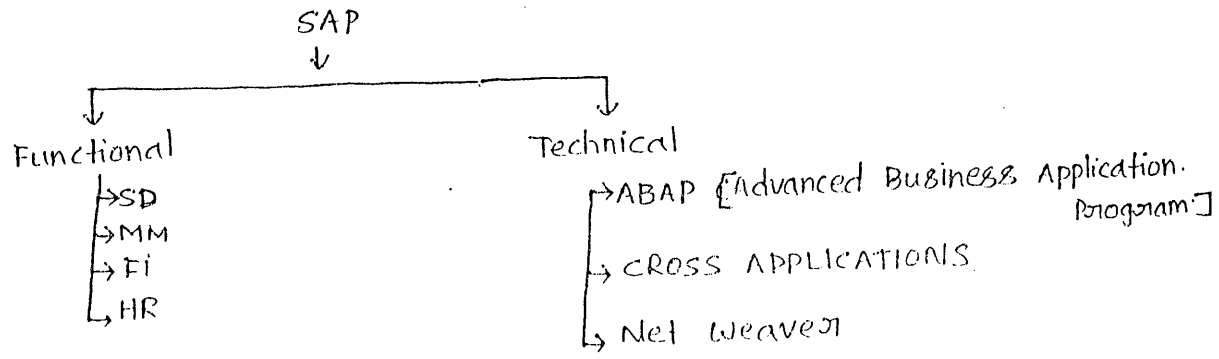
How to provide ERP Solutions :-

1. Business process analysis
2. Document use analysis
3. client Sign off
4. GAP analysis
5. Realization.
6. Testing
7. Q.A (Quality Assuarence)
8. Go-Live
9. End user training
10. Post production Supports.

1. Business processing :- analysis the SAP implemantation learn with study the project site and analice it i.c Resaurments of that site to implement an SAP.
2. Document the analysis :- After studying the site the team keep all the analysis of that site anto the document.
3. client Sign off :- client studies the document prepased by lean. and he will cheer weather the analysis is meet their, business.
4. GAP - Analysis :- GAP Between the client requirement and team. analysis i.e what is actually required and what anable.
5. Realization :- implementation starts on system for that arguments
6. Testing :- Diffrent types of testing will be done over implementation
7. Q.A :- Implementation will checked by Quality team.
8. Go-Live :- This implementation of SAP will deploys for that arguments.
9. End-user training :- Train the end users to work on that Implementation SAP for that organization.
10. Post production supports :-

SAP :- Systems applications and products for data processing. SAP devloped by five IBM X employees in 1972 and SAP AG ↓ means LTD company in Germany.

This ERP is completely devloped and we have to customizes and what ever we want.



Total SAP applications are developed

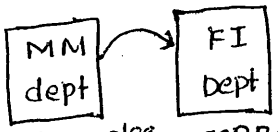
* ABAP Language developed based on 'c' Language.
* cross / applications are used to communicate the distribute business system.
* In case of other ERP's the devlopment was done upto 30% only the remaining will be done using the special tols provided.
* But incase of SAP is completely developed so we have to the predefined thing like applications is tables.
* To implement the ERP For a business organization. it takes to do
* we can implement SAP in all deportments of an organization. so it is most popular ERP in the market.
* In case of Non-ERP's every thing we have to devlop then scrates to do.
* JD Edwards is best for. vendor. because it controling maximum number of vendor tables and applications.

ABAP: Advanced Business Application programing.

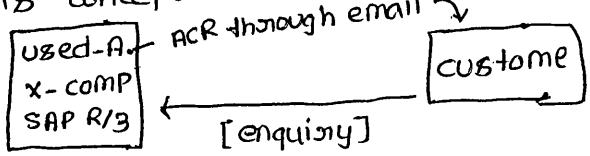
ABAP concepts:- 1. ABAP transactions
2. BDC concept (Batch data conversion)
3. SAP scripts
4. ABAP Reports.

ABAP transactions:- using this concept we can add a Helds to applications.

BDC concept:- using this concept we can transtes data from to SAP.



SAP Scripts:- using this concept SAP R/3 Communicate business Pasdness



ABAP Reports:- used for. reports for. SAP environments it is most imp Concepts.

SAP R/3 Hardware Requirements :-

1. P IV or P III Processor
2. 80 GE HRP
3. 512 ME RAM.

SAP Technical features :-

1. Plat form independent
2. Data base independent
3. Based on III tier Architeture
4. open System
5. Multi Language Support.

Plat form indipendent :- It can work in any types of platform indeper

Data base independent :- It can use any type of data base.

Based on III tier Architecture :- Because of III tier architechure is a more tastes and that will not effect the change in one layer.

open System :- open System is with out changing codes we devlop 70% of SAP remaining 30% we can. Customizes.

Multi Language Support :- It can Support more then 75

* open SQL : It can work for any RDEMS (SAP uses)
* Native SQL : It can work for Respective of Native.
* SAP has its own DBMS. in built.

Function of operating System :-

Resources

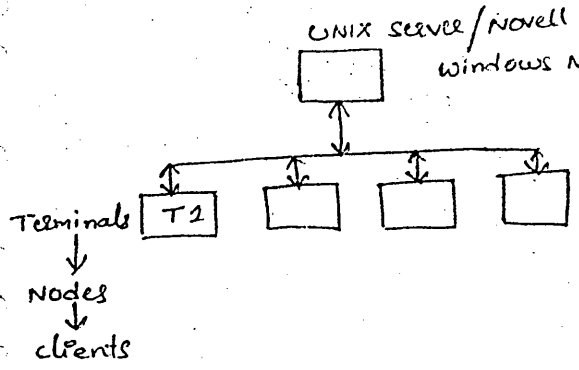
- Memory
- Processors
- Information.
- Devices

Management

- Keep track of Reasources
- Identify jobs that need Resource
- Inforce apolicy to allate.
- Allocate.
- Deallocate

There many functions allonted for one used means IS tier.

III tier Architecture:

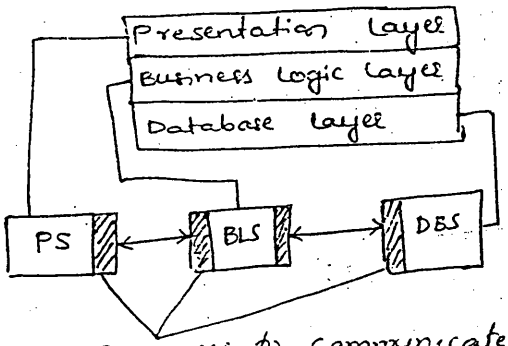


* Unix server uses Terminals because data execution will takes place in unix server it will based on time sharing so it give 20% accuracy.
* Novell netware uses Nodes because of load sharing it gives 50-55% accuracy
* windows NT uses clients because both server and client work as same and it gives 100% accuracy.

* client server architecture called as II tier architecture.

client is presentation layer/server.
server is Backend server

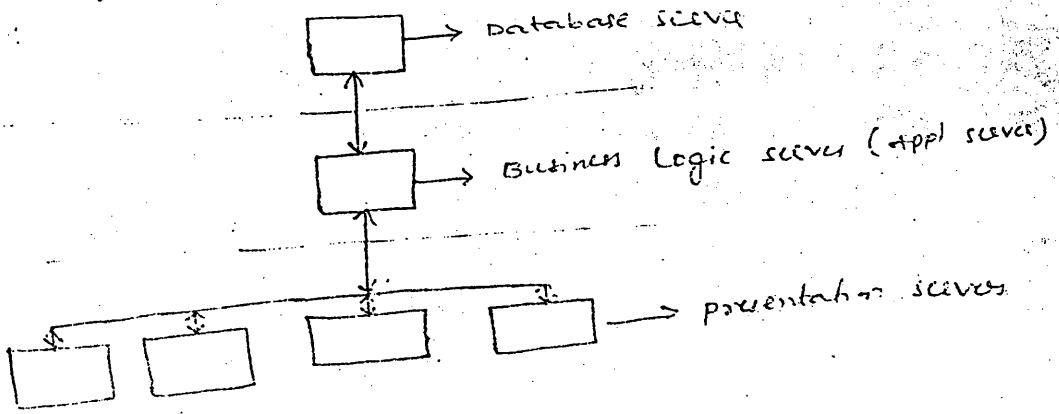
III tier :



Advantages of III tier technology:

1. Load balancing
2. Scalability (We can increase application servers according to our requirement)
3. Availability (If any problem in one AS we can remove that place it a new one without effecting ....)
4. Reliability.

Architecture:



SAP AG products:

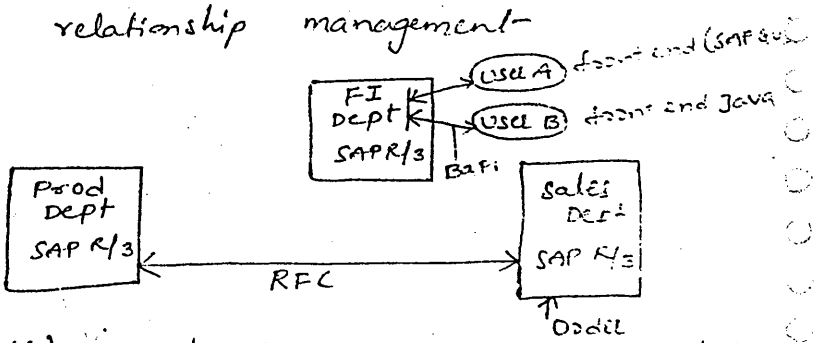
1. SAP R/2 First ERP product released in 1970, Language is cobol
2. SAP R/3 Second ERP product released in 1990, works on RDBMS and based on abap language this is based on 'c' language
3. MySAP.com : Used for e-business solutions language is ABAP
4. SAP mini/SMB : small scale or medium scale business, Language is ABAP
5. IS (Industries specific) : IS-oil, Textiles, sales.

* SAP R/2 works based on 'mainframes'

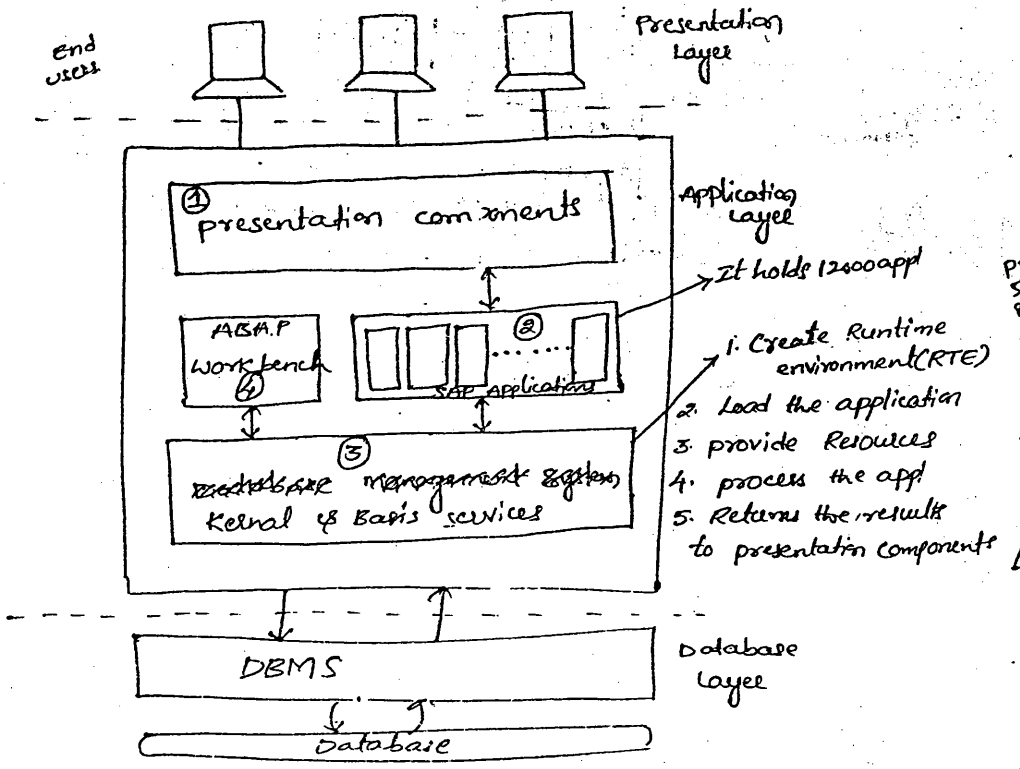
New dimensional products from SAP AG:

1. SAP-BW (or) BIW (Business Information Waiehousing)
   It is datawarehousing technology for forecasting analysis
2. SAP-APO's : Advanced planner and optimizers
   It is supply chain management technology (SCM)
3. SAP-SEM strategic enteeprice management
4. SAP-SRM Stratagic relationship management
5. SAP-CRM customer relationship management
6. SAP-Netweaver

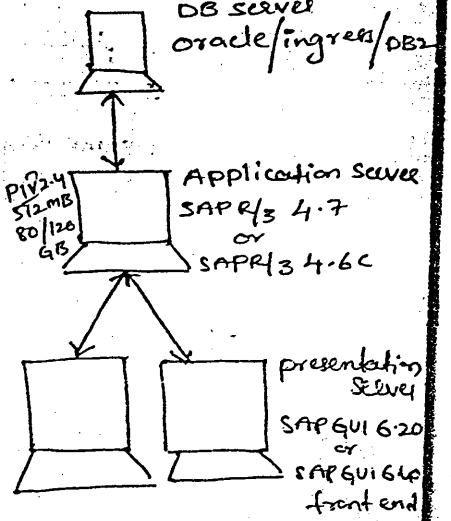
CROSS applications:



* RFC (Remote function call) provides communication b/w distributed SAP R/3 Systems'.
* BaPI (Business application programming interface) provides interface b/w SAP R/3 and third party front ends.
* Cross applications used for distributed business activities.



physical view



H/w Requirement
celeron/P IV
128/256 RAM, 10 GB HDD
(700-800 MB occupied)
called SAP green template

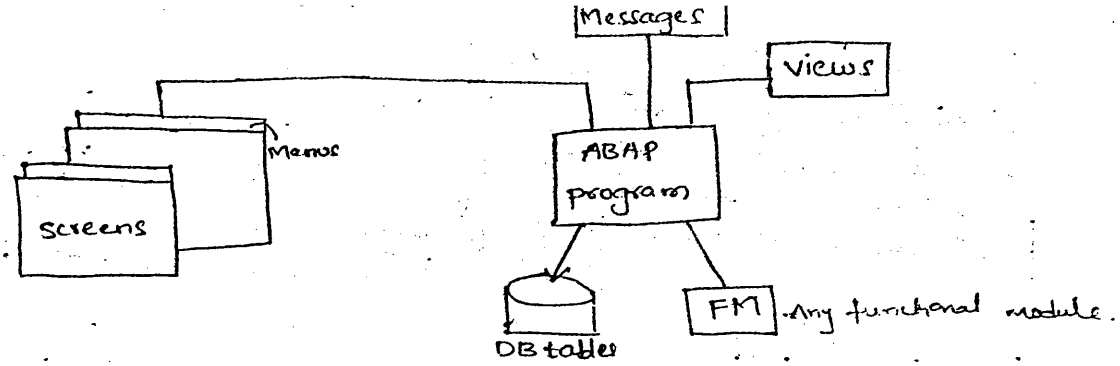
ABAP Workbench: is a Integrated development environment used to develop SAP applications.

* ABAP Workbench is a set of tools that can generate components of SAP application.

components of Workbench:-

1. ABAP EDITOR : To create ABAP programs.
2. ABAP DICTIONARY: Used to create global dictionary objects like database tables, views etc..
3. SCREEN PAINTER: Used to create GUI screens for SAP applications.
4. MENU PAINTER: Used to create Menus for SAP GUI screens.
5. FUNCTION BUILDER: Used to create function modules.
6. FORM PAINTER: Used to create SAP business documents.
7. MESSAGE CLASS BUILDER: Used to define 'ALERTS'

These components are technically called as 'ABAP OBJECTS'
These are stored in 'database'



* SAP application are build on component model.

* Each tool is responsible for creating a components. All these components are ABAP objects.

* All these components are directly stored in D/B.

* component model gives Reusability

ABAPER's Role:

1. Customizing → i) Modifications
   ii) Enhancements

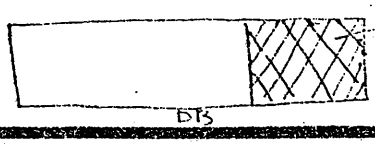
2. Reusability → Reuse the existing components

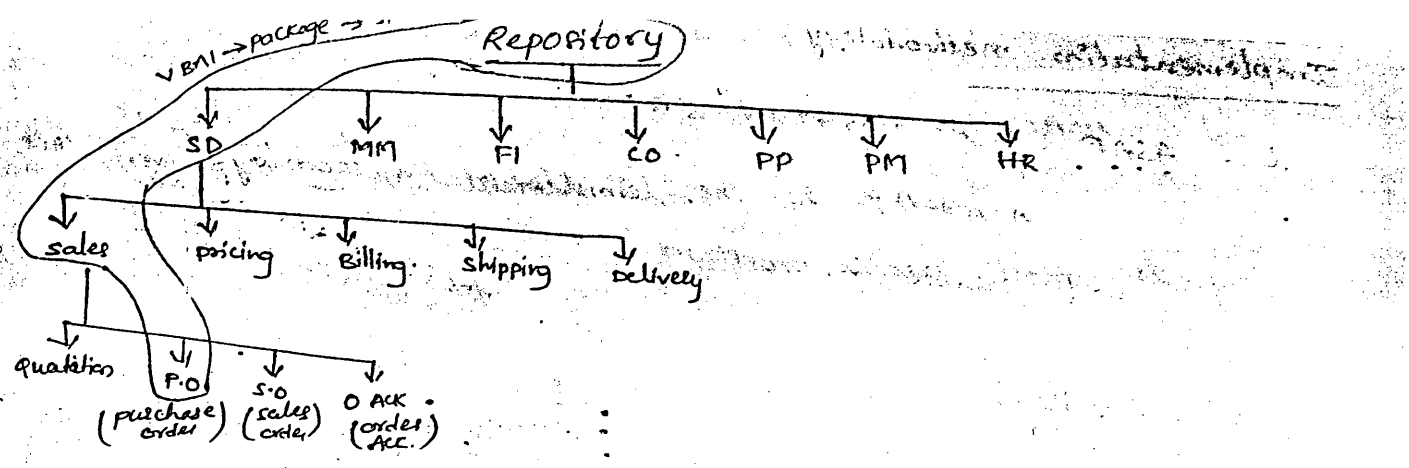
3. Develop new application :- We can build new appl using the components of existing appl

SAP application types:

1. Report application
2. GUI application
3. Business Document processing appl
4. Data transfer appl

Repository: A portion of database that is used to maintain SAP application components.



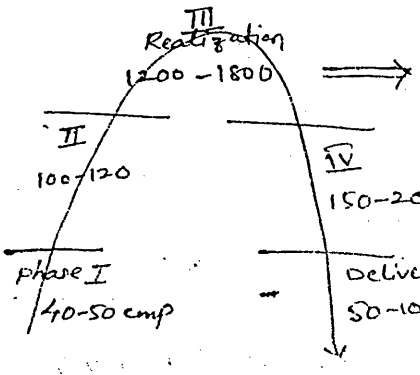


⟹ SAP provides a standard packages or user defined packages in Repository

SAP Implementation Life cycle: Life cycle or n-n is called SAP implementation from scratch to top

Development	Testing/QA	production
1. Project preparation 2. Business Requirement analysis documentation is called i) Business Road maps ii) Business blue prints iii) client sign off iv) FIT GAP analysis 3. Realigation phase i.e start doing on system.	4. Testing/QA	5. Go-live & Support → Deploy SAP → Configure/test → Endusr training → Finally gives ownership to organization → Limited post production support.

SAP implementation chart:



⟹ In this phase abapers will enter on contract basis.

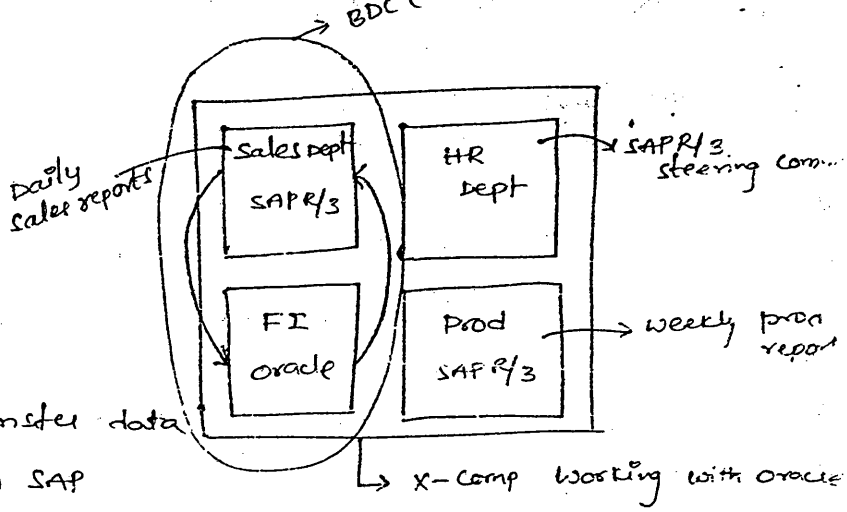
Implementation methodology :-

1. ASAP : Accelerated SAP

ASAP is the implementation methodology for SAP and this is most usable method.

ERP consultants:

1. PWC
2. KPMG
3. Deloitte
4. E&Y
:



BDC concept : used to transfer data from SAPR/3 to non SAP

Flow charts : Are nothing but EPC (enterprice process control diagr

EPC uses VISIO, ARIS tools for EPC diagrams.

* project team contains 13-15 employees

* Functional spek : Functional people will prepare it and forwarded to project leader of ABAP and in this area project leader convert function spe to technical spek and which can uses AB program.

SAP R/3 versions :-

SAPR/3

3.0A
3.0B
3.1A
From this version onwards ← 3.1F
SAP becomes object oriented technology
3.1H
4.0A
4.0B
4.5A
4.5B
4.6A
4.6B
4.6C
4.6D → Failure version
4.7 or 4.7EE (Enterprice Edition rele one yea
for experience purpose

ABAP Dont have any versions i.e "ABAP/4
we can write SAP R/3 4.6C/4.7EE

Websites for Real time Information:

1. WWW.Sap.com
   ↳ case studies
      ↳ Asian paints, Hero Honda, Videocon, BHEL } Best SAP implementation com

2. WWW.sappoint.com
3. WWW.sapGini.com
4. WWW.ABAP4.com
5. WWW.sapfans.com

SAP R/3 Installation :- IDES (International demo. eductional system) for training purpose, which provides examples for training.

production s/w : Dont Install because it will not provide examples

SAP GUI in 4.6C version:

Client : Group of user ID's

SAPAG → SAP R/3 provides client ID as '000' (Administrator will create this ID)

These are the following clients and Respective ID's

1. Development client : 100
2. quality client : 200
3. production client : 800

} These are the copy's of the client '000', Administrator will copy it

* As an ABAP programmer we can access development client and quality client
* As a functional people we can access any of three clients.
* If we are enduser of the company we can access production client

* Development client [illegible] Objects
* quality client for testing ABAP objects.

we can use any testing tool to test ABAP objects or use SAP provided tool - 'CATT' (computer aided testing tool)

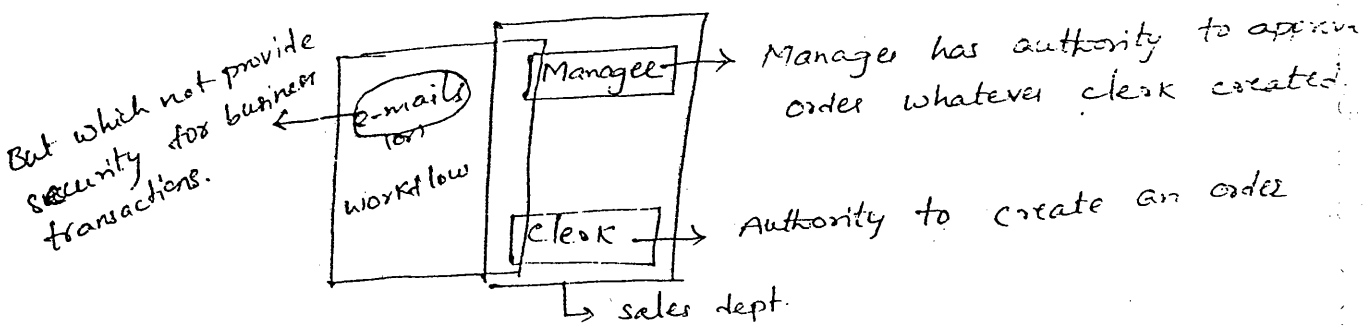
Logon details: client : 800
user : SAPUSER
password : ABAP
Language : En (English)

* SAP GUI Designed based on Java

Dimensions of SAP GUI: Height : 80 lines
Width : 255 characters.

SAP menu contains following:

1. Office : is for distributed environment
2. Workflow : Workflow works like a email and which also have Inbox and outbox in offices. The main advantage to go for workflow is provide security for business transactions. e-mails will not provide it



3. Logistics: Exclusively for functional people
   - sales & distribution (SD)
   - Material Management (MM)
   - Production & planning people

These people work under Logistics.

4. Accounting :- It is for Finance & controlling functional people

5. Human Resource: for HR functional people

Technical ABAP and HR ABAP are different

6. Information systems: Have predefined ABAP Reports.

SAP R/3 have 2000 predefined abap reports.

7. Tools : for abap programmers and administrators.

- ABAP workbench (which is called ABAP BOR ABAP Business Object Repository)
  - Repository browser (Like a directory for storing SAP Objects)
  - ABAP Dictionary (for designing database related activities) (or) Data Dictionary
  - ABAP Editor (for designing programs in SAP)
  - Function Builder (for creating functions in ABAP)
  - Screen painter & Menu painter (Designing applications in SAP R/3)

How to select ABAP Dictionary:

Tools → ABAP workbench → Development → ABAP Dictionary

Transaction codes: which are used to directly interact with a ABAP objects.

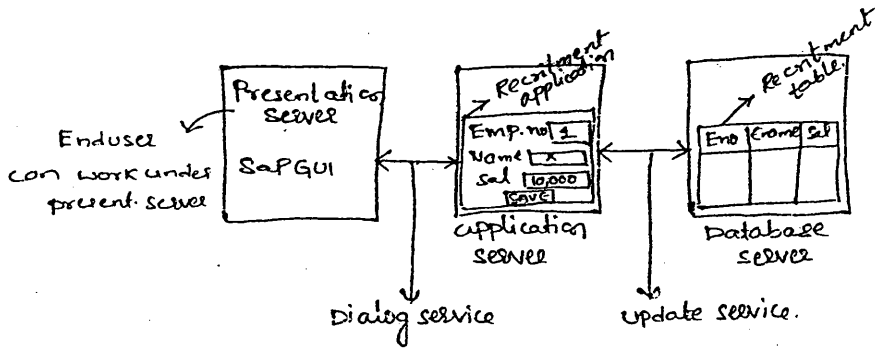
Repository Browser	:	SE 80 (SE : System Engineering)
ABAP Dictionary	:	SE 11
ABAP EDITOR	:	SE 38
Function Builder	:	SE 37
Screen painter	:	SE 51
Menu painter	:	SE 41

We can give this code in left side corner of the screen box then press enter to directly interact with them.

Naming conventions: SAP R/3 uses Y (or) Z letters to start name and remaining letters are reserved f (why Y or Z means it is predefined)

21/6/05

SAP R/3 Architecture:



Dialog service: is provides communication b/w presentation server and appl server. i.e SAPGUI will get the Recruitment application and user enter the values.

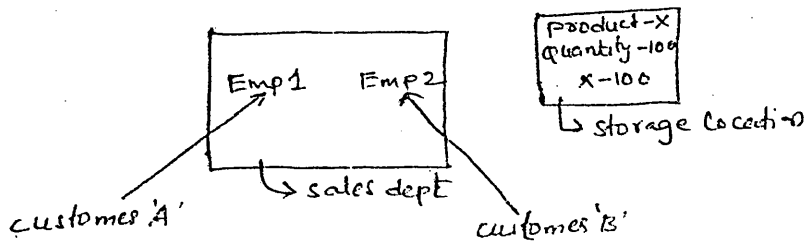
Update service: is provides interface b/w application server and database se i.e after entering entering the data in appl it can store i

These two services are very important in SAP R/3 Architecture.

SAP R/3 provides some more services:

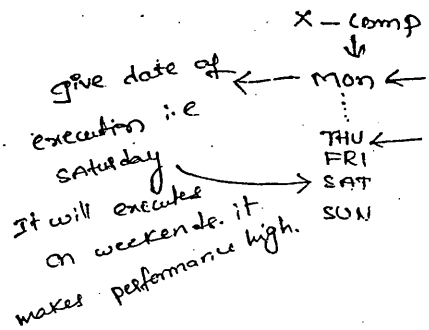
1. Gateway service: for distributed SAP R/3 environment. (Ex: BAPI RFC (urrent gatew
2. Message service: for exception management i.e error handling
3. Spool service: SAP R/3 can work with external systems i.e Fax, pr
4. Enque service: provides data integrity in SAP R/3 technologies.

Example for data integrity:



4. Enque service provides locking and unlocking of objects i.e Diagram shows there are two employees and two customers A, B If both customers order same time of product X of quantity 100 then using this service it will give 'A' unlocking facility to customer A while 'A' creating an order 'B' will be locked and viceversa. If product is avilable. Enque works based on Lock objects.

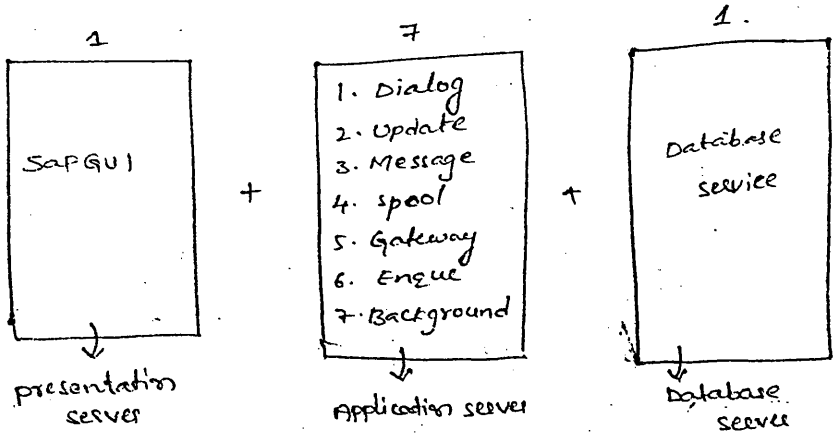
5. Background service:- Background service for background job scheduling. i.e first we select long running programs and give a particular time i.e weekends for execution in this time system will take care about that execution is called Background service.



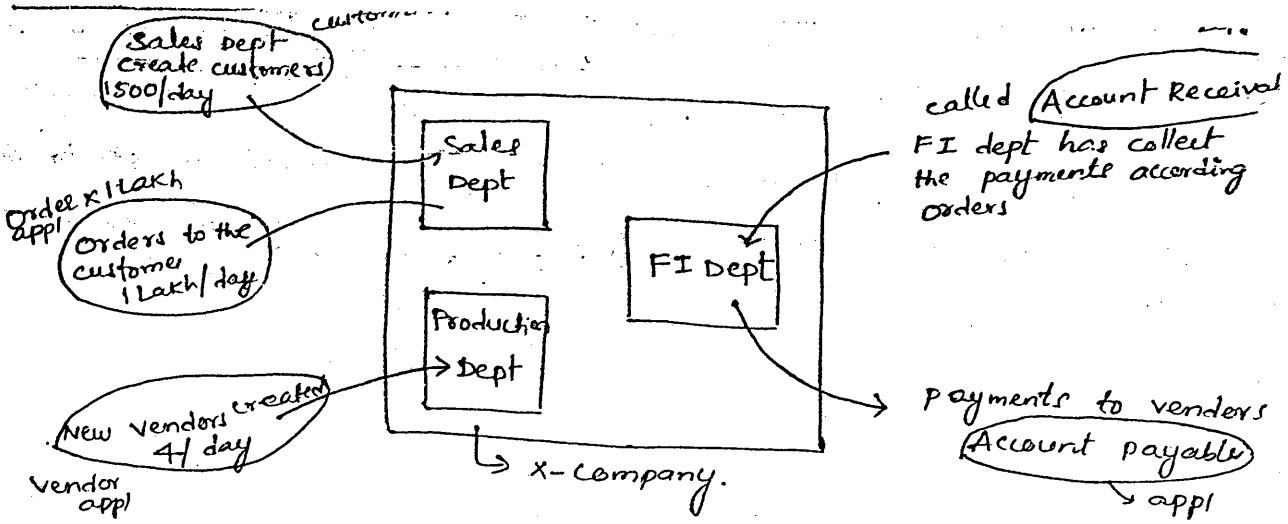
6. Database service:- It can store the data permanently in database.

* If any one of this service fails we not able to work with SAP R/3.

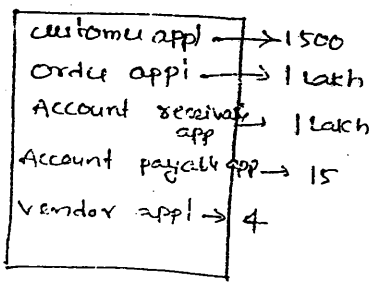
Services used in SAP R/3 architecture:



Application server uses more services.

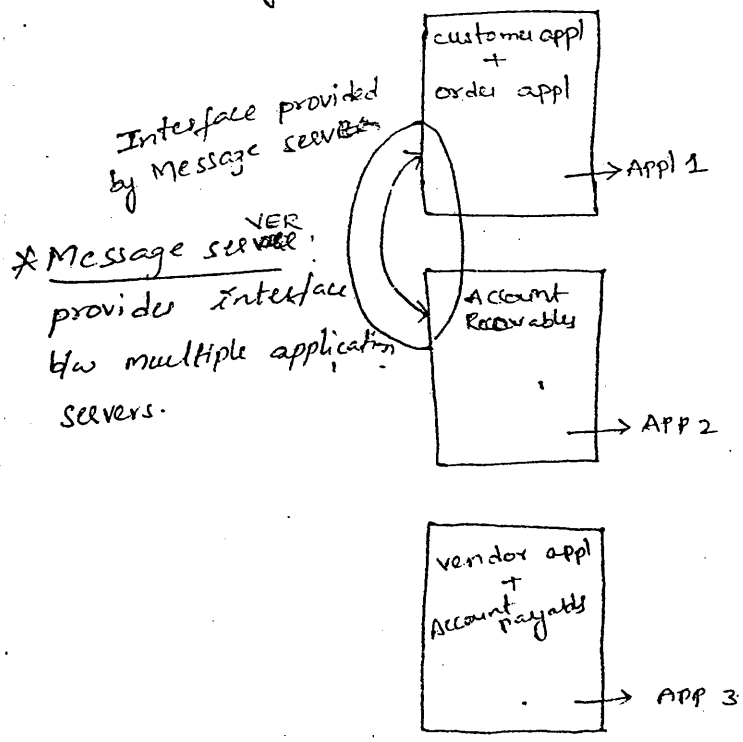


SAP R/3 architecture of above scenario:



These many applications will give low performance of applic. Server so we divide applicat. in different servers.

* For high performance we can add more application servers for sharing.



* Message server provide Interface b/w multiple application servers.

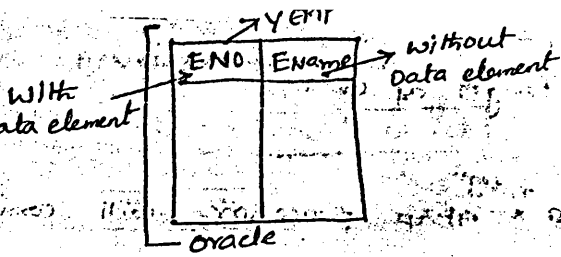


Table creation in oracle :

create table YEmp (ENO Integer ...)

* While working with SAP R/3 we cannot create tables directly by syntaxically like oracle it uses ABAP Dictionary to create it.

ABAP Dict:

→ Domain : provides properties of respective fields i.e 'Datatype' and 'Length' of the field.

→ Data element: provides field labels for respective fields in backend.

* without 'Data element' it is possible to create a field in SAPR/3.

Table creation steps:

SAPR/3 Menu
→ Tools
→ ABAP Workbench
→ Development
→ ABAP Dictionary (SE11) ↵

It will displays a initial screen in that select 'Domain' and give Domain name i.e

⊙ Domain : ZENODOMAIN These are user defined tables so start with 'Y' or 'Z'

select create button in that window

Displays a window and provide following in that-

Short text : Domain for DNO field

Mandatory fields: nothing but fields that compulsary enter in windows i.e

Datatype, Length of. char

* If we provide Data type as a Integer * Abap processor will convert it to character.

* If we provide character type directly processor uses directly

Data type : CHAR

Length of char : 15
Field or length

Save the details by pressing Ctrl + S it will display a window and provide following

Development class : [ ] Make it as blank

Select Local object on that window.

press Ctrl + F3 for activate that Domain. It will shows the Domain field in 'Active'.

* Press F3 for back to initial screen i.e abap dict in that

Select ⊙ Data type : ZENODATA

Select : Create it will display the window in that select

⊙ Data element

provide short text : Dataelement for ENO

provide Domain which we created just now

Domain : ZENODOMAIN

select field label option and provide following

Field label length	Provide label
Medium 20	EMPLOYEE NUMBER

Ctrl + S for save

Development class [ ] Blank

select [Local object]

Ctrl + F3 for Activate datatype.

F3 for back

Select ⊙ Database table [YEMP]

go for [create]

Delivery

Short discription : [EMP TABLE]

Delivery class : [A]

☑ Enable checkbox for maintenance allowed

Go to fields option:

Data element / Direct type

Fields	Key	Fieldtype	Datatype	Length
ENO	☑	ZENODATA (data element) i.e after enter the fieldtype click the datatype i.e datatype and length will automatically get it from domain.		

go for second field ENAME — select Direct type then field type will be disabled and datatype & length option will enabled so we can enter it manually.

Datatype : CHAR

Length : 25

Ctrl + S for save

Development class [ ] Leave it blank.

Go with local object then

select GOTO

→ Technical settings

→ Provide data class : "APPL0

Size category : [0]

Ctrl + S (save)

F3 for back and Activate it by

Ctrl + F3

CHECKING and Inserting of fields :-

select UTILITIES

→ Table contents

→ Go for Display for check weather created fields getting or not

→ Go for create to enter the data in that fields

By using Reset option we can enter more records.

Activation : By Activating the objects will transfer from application server to database server.

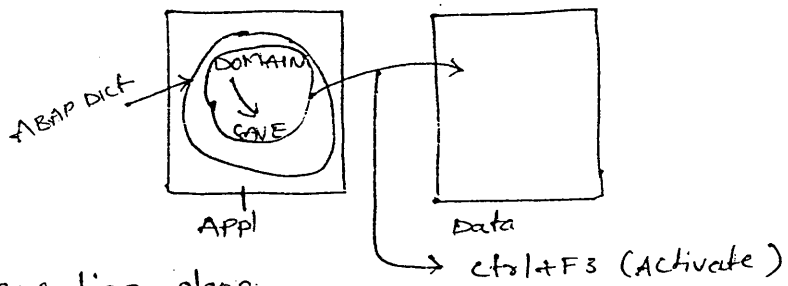


Table creation steps:

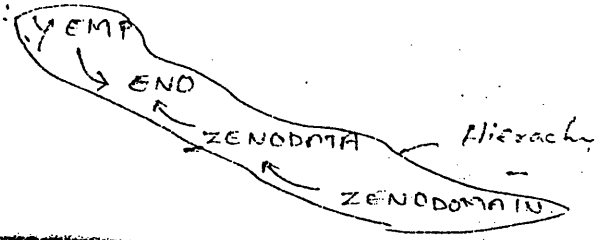
I step : DOMAIN : ZENODOMAIN

→ Data type : CHAR
→ Length : 15
→ short text

II step: Data element : ZENODATA

→ domainname : ZENODOMAIN
→ Field label
→ short text

III step : Database tabel :



→ Table contents
→ display

F8 for execute: it will display details for insertion items.

Field selection:

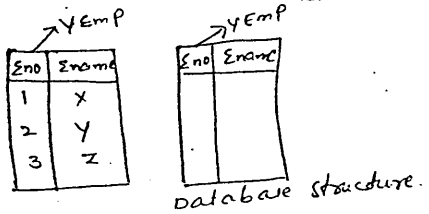
With Data element we get a field name
Without Data element we dint get label name we can enter it manually.

Domain: have Reusability concept : users can use domain within the table and across tables.

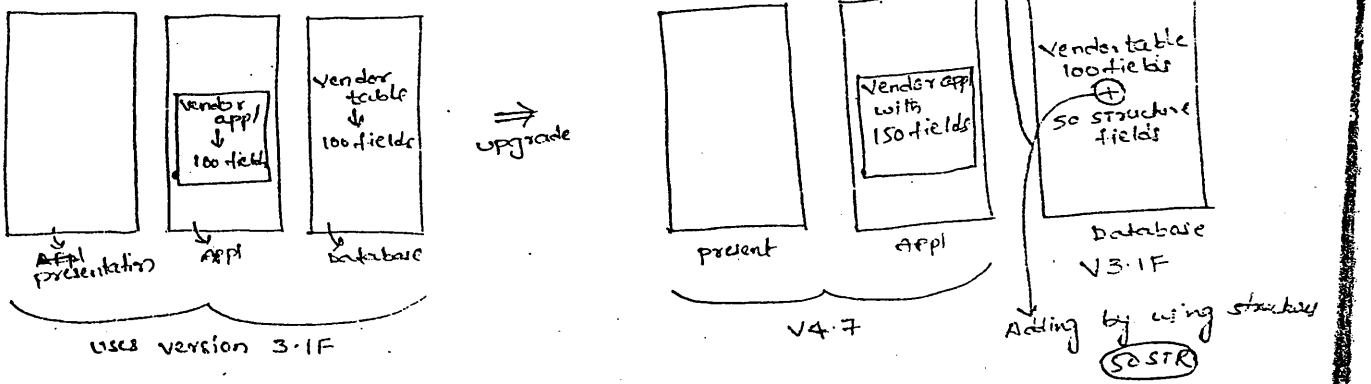
Data element: is restricted for that field only

25/6/05

ABAP Dictionary
→ Database structures : Database structures does not maintain the data by default.



Upgradation or Adding a fields:



* In SAP R/3 it is not possible for adding the fields directly to SAP tables. With database structures users can add a fields for SAP tables.

ex. previous upgradation

using 3.1F SAP version and have 100 vendory fields vendor table and vendor application. Upgradation takes place when new version coming into market then it will upgraded by using conversion presentation & applicat. server to new version i.e 4.7 and put it Database server as it is in any type of technology. If we want to add some fields to already existing database table i.e vendor table we can add it by using structure and following keywords.

* • INCLUDE (or) • APPEND or the keywords for adding a structure to the tables:

STRUCTURE CREATION STEPS:-

Go to ABAP Dictionary in which

Select ⊙ Datatype and give name of Structure. [YSTR]

go for [create] it will display a option window then

select ⊙ [Structure] then ↵ it display a window in which

provide short text for structure [Structure for YTEMP] → Table created + Developm class

provide component (Field in a structure) : [ADDRESS] what we want to add

provide component type (Data element) : [ZADDRESSDATA]

[ctrl+S] for saving

Development class [ ] Blank

go for [Local object] then

Double click on Dataelement i.e [ZADDRESSDATA] what we gave in previous step (component type)

select [Dataelement] ↵

provide short text for dataelement [Dataelement for ADD]

provide Domain [ZADDRESS DOMAIN]

↳ This Domain is not yet created we create it in coming steps so we create by using this Domain name

So, whatever domain name given in this box we remember for to create a domain in coming steps.

then select [Field label] option in that window then provide field label lengths i.e

Medium [20] [EMPLOYEE ADDRESS] ↳ Field label name

[ctrl + S]

Development class [ ] Blank

go with [Local Object] then

select [Definition] option in that window

Double click on Domain i.e [ZADDRESSDOMAIN] ↵
↳ Just we created in above

provide short text for Domain [Domain for ADD]

provide Datatype : [CHAR]

Length : [30]

[ctrl + S]

[ctrl + F3] then F3 for back

Activate Dataelement then

F3 for back

Activate Structure

go with ABAP Dictionary

select Database table [YEMP] → which we created in last class ie for which table we want to add a structure.

go to [Change] mode

go with [New rows]

Apply logic for including a field i.e

- INCLUDE YSSTR
  ↳ structure

[ctrl + S]

[ctrl + F3]

goto Utilities

→ Table contents

→ create

It will display a created fields and insert the Records in that window field and save it

For display of fields:

Utilities

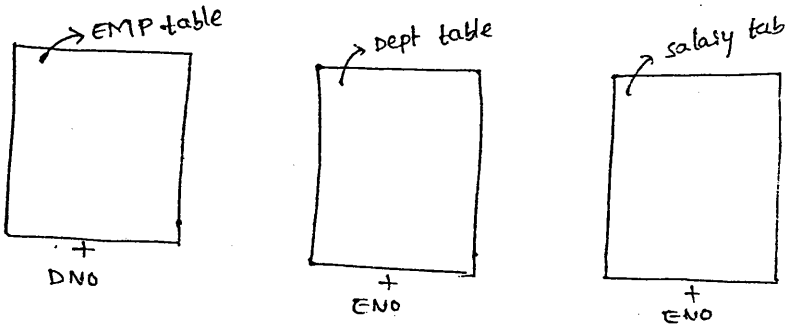
→ Table contents

→ display

Then press F8 for execute then it will the inserted records of that fields.

* Once we add a structure to the table it can have data

Difference b/w • Include and • Append:

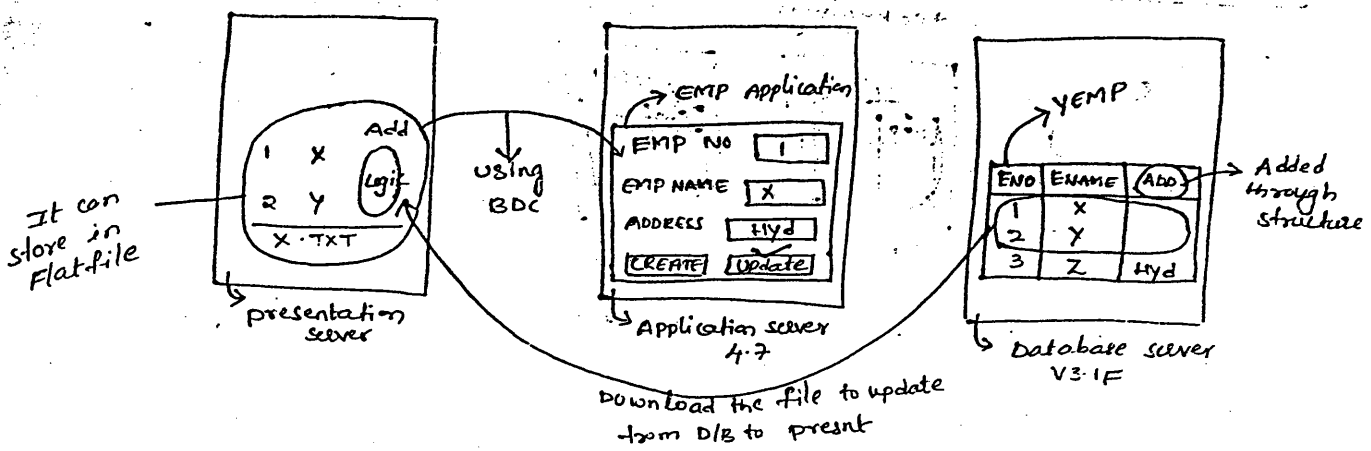


Here we have 3 tables and we can add a fields DNO, ENO, ENO respectively. Here we can use two structure for add that field bec ENO field is reusable so

With • APPEND structure will restrict for specific table only

With • INCLUDE structure can reuse different tables.

How to maintain previous data properly :-



we have to maintain all records in proper way after add a field through structure we can follow a above logical for that

* First we download the records which are we going to update
* SAP provides some programs for that logic in presentation server
* Downloaded record will store in flat file or textfile
* Flat file can be transferred to application server by using BDC
* Then we can update a record and transfer to database.

Flat file: is nothing but a text file or Excel sheet
After completion of logic flat file transfer to Appl server

9/1/1605

Database tables: Under database tables we have following

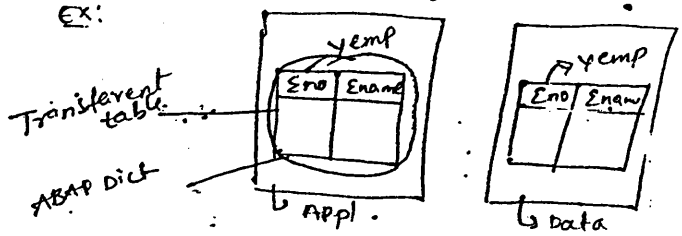
1. Data class
2. Size category
3. Buffering concept
4. Maintenance allow
5. Delivery class.

Data class: we are providing it as APPL0 it means Transparent tables.

Types of tables:
1. Transparent tables
2. cluster tables
3. pooled tables.

technology.

Ex:



If it is transparent table structure in abap dictionary is same as structure in database.

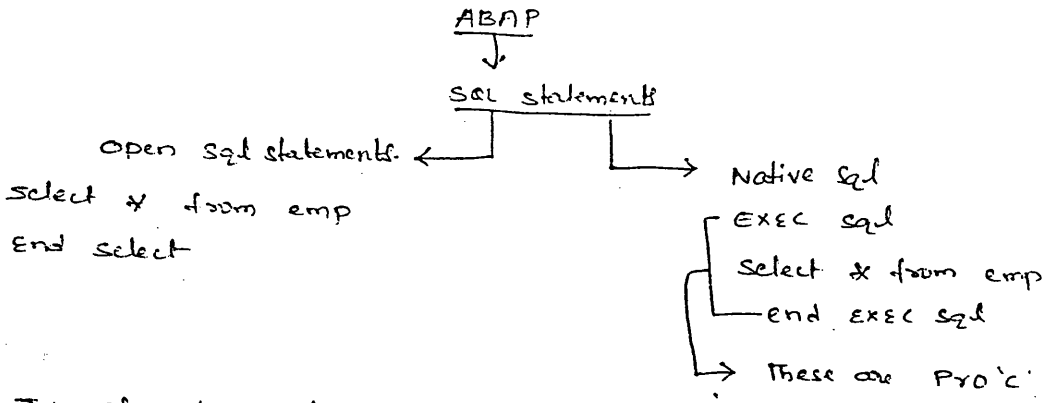
Cluster & pooled tables: These are used in HR-ABAP

* Mainly used for performance

* Clustering in nothing but group of fields from different transparent tables and it follows 'Binary search'

* pooled table also same as cluster but it follows linear search

The Main difference between ~~clusting~~ clustering table and pooled table is Binary search and linear search.



Transparent tables: Users can work with open sql and native sql

If it is cluster or pooled tables user can work with open sql only.

APPL0 indicates a transparent tables.

Size category : 1 9
(61000) (9x61000)

① Indicated user can maintain max upto 61000 records.

* Size category provides the size of the data which table maintains.

Buffering concept:

- Buffering switched on
- Buffering not allow
- conditional buffering.

Buffering switched on means all records stored parallel in application server and database server i.e when buffering is switched on that buffer maintains in application server.

In Real time conditional buffering is used for storing the records.

Ex:

Eno	Ename
1	x
2	y
3	z

* permanent records stored in buffer

permanent employee

→ contract emp

☑ Maintenance allowed: It provides Authorization to the tables.

Enable the checkbox allows directly store records in front end aswell as database.

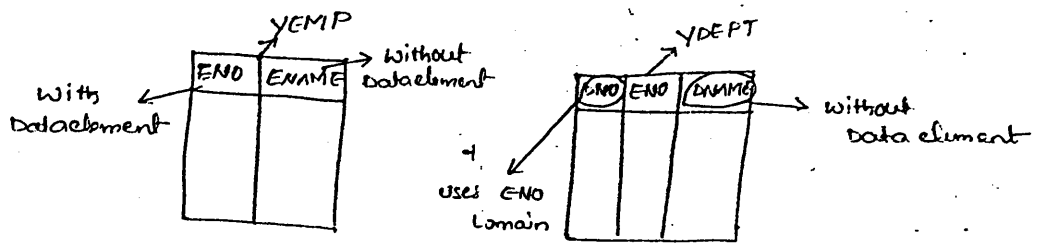
Disable checkbox allows only front end.

Delivery class: provides type of data which tables maintains data is of 3 types.

1. Master data (Eno, Ename etc)
2. Transactional data (salary)
3. Control data (Dept)

[A] Indicates master & transactional data.

— also creation of VIEW of this tables.



(Because both fields properties are same)

Go to ABAP Dictionary

⊙ Select Database table and provide name [YJEMP]

Go for [create]

provide description for table : [Employee details]

Select [Attributes] option in that window (Attributes are properties of table)

Under attributes

provide Delivery class [A] (indicates master & transactional data)

[✓] Enable checkbox for maintenance allowed.

Go to [Fields] Option, under this

Fields	P.K	Fieldtype
ENO	✓	YENOJDATA

Ctrl + S

Double click on Dataelement i.e YENOJDATA ↵

provide short text for Dataelement [Data element for ENO]

provide [Domain] name [YENOJDOMAIN] → we create it later i.e next step

go with [fields label]

provide length and label

Medium [20] [Employee Number]

Ctrl + S

go with [definition]

Double click on Domain which u provided i.e YENOJDOMAIN

provide short text for Domain [DOMAIN FOR ENO]

Data type : [CHAR]

Length : [10]

Ctrl + F3

F3

Activate Data element

F3

Go with new rows

provide second field [ENAME]

Go with Data element Direct type.

for second field provide the Datatype and length

Datatype : [CHAR]

Length : [25]

Ctrl + S

Select GOTO → Technical settings

under setting provide data class : [APPL①]

Size category : [①] it means 61000 records can be stored.

Select ⊙ Buffering not allowed.

Ctrl + S

F3

Activate the table Ctrl+F3

Inserting of Records:

Utilities → Table contents → create

Enter a record

END : 1

ENAME : Jagadeesh

Ctrl + S

F3

Creation of YDEPT table:

Go to ABAP Dictionary

provide table : [YJDEPT]

Go for [Create]

provide the Description [Dept Details.]

Delivery class [A]

☑ Enable checkbox for Maintenance allowed.

go with fields Option provide the first field

field	P.K	field type
DNO	☑	YDNOJDATA (Dataelement)

ctrl + S

Double click on Dataelement mentioned above (YDNOJDATA) ↵

provide short text [Dataelement for DNO]

Provide Domain : [YENOJDOMAIN]

↳ provide domain of ENO of YEMP table because these two tables properties are same.

go with field label and provide length & label

Medium [20] [Dept Number]

ctrl + S
ctrl + F3 } or we can use ctrl+F3 for save & activate

F3

go with new rows

provide second field [ENO]

and provide Dataelement which we already created for ENO in YEMP table because labels are same. i.e [YENOJDATA]

go with [Foreign Key Symbol] i.e (After new rows we can see this button)

provide short text : [Relation b/w YJEMP and YJOEFT]

provide check table (primary key table name) : [YJEMP] ↵

↳ provide Primary key table name.

↵

Accept the proposal ↵

(Here proposal means relation of two tables)

By seeing checkbox we can identify foreign key.

provide last field

DNAME without Dataelement direct type

provide Data type (char) and length (30) for last field

[Ctrl + S]

Goto → settings

provide data class [APPL0]

: Size category [0]

Select Buffering not allowed

Ctrl+S

F3

Ctrl+F3

Inserting a record: Utilities → contents → Create

Enter record

DNO : 10

ENO : 1

DNAME : HRDEPT

It will saved successfully.

Enter one more record

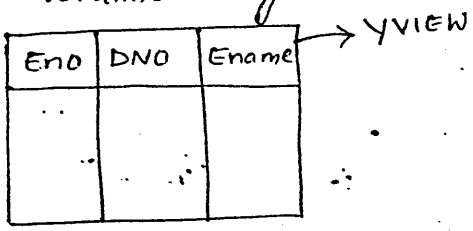
DNO : 11

ENO : 2

DNAME : FIDEPT

It will give error because second record is not available in YJEMP table relation will not exists.

VIEW CREATION: View is a logical tabel and it maintains the data at runtime only.



VIEW DEMO:

Go to ABAP Dictionary

select ⊙ View and provide view name [YJVIEW]

go for create

Select Data base View (Because we are using database tables)

provide short text: DEMO

provide tables which we are using for view creation

i.e YJEMP

YJDEPT

provide Joined condition Relation by selecting Joined

YJEMP ENO = YJDEPT ENO

Go with View fields

provide the fields from above view table (Diagram)

ENO	YJEMP	ENO
DNO	YJDEPT	DNO
ENAME	YJEMP	ENAME

Ctrl+S

Ctrl+F3

FOR execution:

Utilities

→ Contents.

press F8 for execute

It will show the records from both tables what we select.

28/6/05

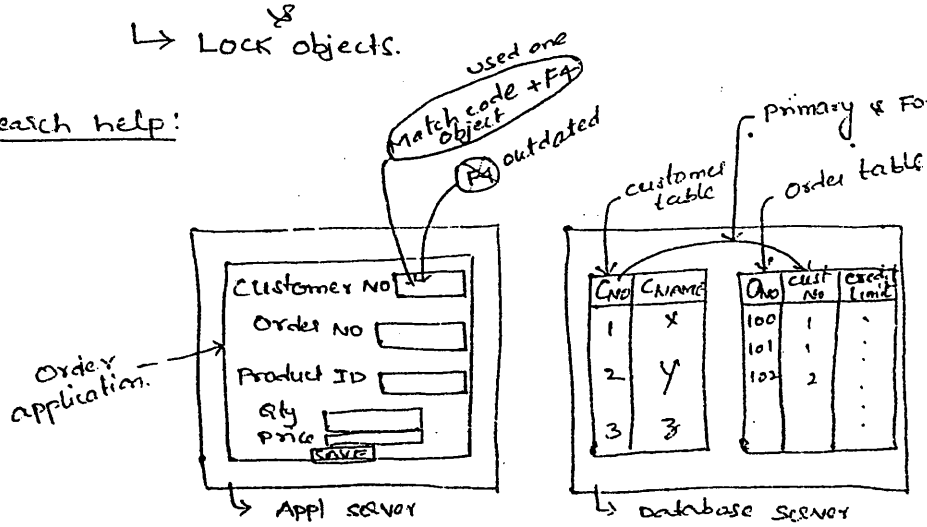
ABAP Dictionary:

→ Search help

&

→ Lock objects.

Search help:



Search help: provides a list of values from database

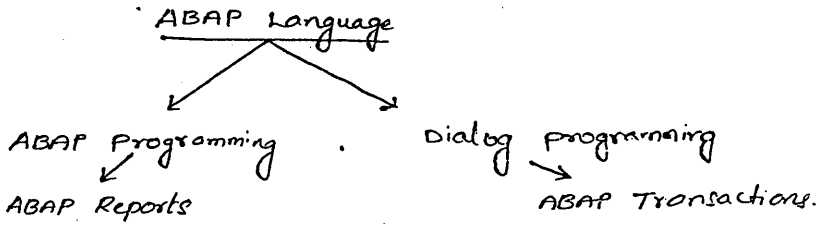
provide the cursor on respective field and press F4 for search he

[F4] it can check the tables independently but it doesn't check relation b/w the tables i.e primary key & Foreign key relations shown in above fields.

* If it is Matchcode objects first it can check the tables independently later it can check relation b/w the tables also.

* with Matchcode objects user can avoid 'Data redundancy'

Lock objects: Enque service works based on Lock object. it provides data integrity in SAP R/3.



* ABAP programming is for data extraction only

* Dialog programming is for Data extraction as well as data manipulation.

Pre-defined Data types in ABAP :- ABAP language have 8 predefined datatypes they are

Character data types →
1. CHARACTER — 'C'
2. DATE — 'D' By default it is 8 char length (YYYYMMDD). Datamask is a keyword for changing Date format.
3. TIME — 'T' - HHMMSS (6 char length)

Integer datatypes. →
4. INTEGER — 'I'
5. PACKED DECIMAL — 'P' } For decimal point calculations.
6. FLOATING POINT — 'F' }
7. NUMERIC — 'N' (Dont use for Airthmatic calculations it is for postal code & Telephone numbers.)
8. Hexa decimal — X → (It is for SAP Graphics)

Example for packed decimal & floating point

1.23456

* packed decimal : provides accuracy in ABAP language i.e it will display exact value as it is (1.23456)

* Floating point : provides performance in ABAP language i.e it will roundoff the values (1.2)

* Floating point designed based on machine code language

⇒ From SAP 4.5B onwards SAP added following datatypes:

9. String (collection of characters)
10. Xstring : For machine code language functionality.

User defined Datatypes:

types : NAME(15) type c

types colon statement is datatype.

type keywords to refer data type

Ex: types : Name1 type Name

* Datatypes are descriptive there is no memory for data types.

* Where as data objects have memory

Data : indicated statement is data object

Data : Name2 type Name1

Data : Name3 like Name2

like → Keyword to refer data objects

Data : Name(15) type c → SAP uses like direct statement.

* Datatypes will define properties for data objects

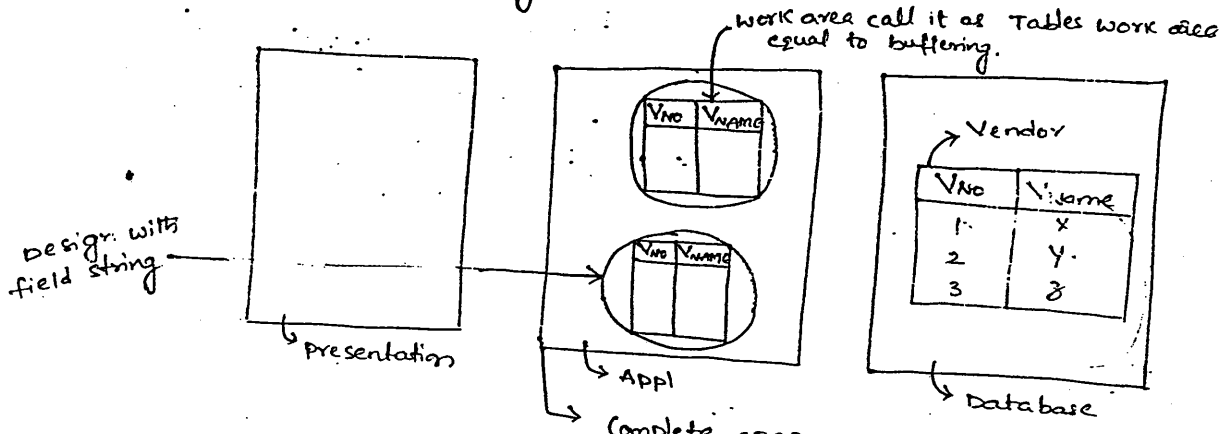
Programming structures

predefined prog structure → Tables

User defined prog. structure → Field string & Internal tables.

* User defined prog structures are runtime objects.

ABAP Prog with Field string :-



Using pre-defined prog. structure:

Tables : Vendor.

↓ It creates work area in appl server

↓ It can extract Vendor table to work area.

Design with field string:

Vijaynirlekhstring

Data : Begin of <fs>,

Data : Vendorno(15) type I,

Data : Vendorname(15) type C,

End of <fs>.

Instead of keeping data two times let it write before begin.

```
select * from vendor into fs
End select.
```

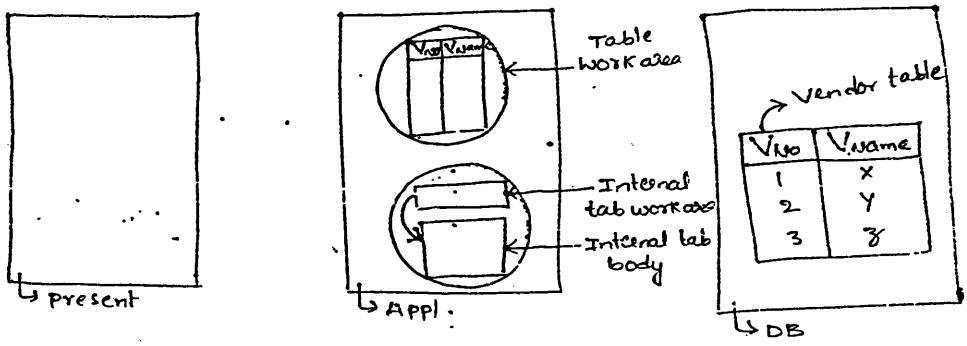
Extract data from vendor table and store in field string.

Write :/fs (it prints the o/p in presentation server)

* Field string can handle only one record at a time at last field string can hold last record (3-z)

Internal tables: are enhancement to field string concept. it can handle multiple records.

29/6/05



prog using Internal table:

Tables : Vendor (creates work area in appl server)

Int : table

Data : Begin of <ITAB> occurs 0 (syntax for Int table)
(It can be any name) → (It can be any no i.e 0, or 100, 200, that many records it can store after reaches that no then increase same no. of records 100 more in)

(or)

Data : ITAB like Vendor occurs 0 with Header line.
→ refers internal table work area it can design with all fields from database

Vendorno like Vendor-VNO
Vendername like Vendor-VName
End of <ITAB>
→ with these statements system create Internal tab workarea and body table

Open sql:

These '3' Select statements need append ITAB and end select

Select * from Vendor into ITAB (It will select all fields)
(or)
Select VNo VName from Vendor into ITAB (selected fields but mismatch occurs)
(or)
Select * from Vendor into corresponding fields of ITAB (low performance)
Append ITAB (Transfer data from workarea to body)
End select

(or)

It no needs Append ITAB End select ← Select VNo VName from Vendor into table ITAB (Best performance)
→ keyword

presenting multiple records in presentation server.

Loop at ITAB (or) Loop at ITAB from 1 to 5 (selected records transfer from workarea to presentation server we have to mention values)
Write : /ITAB (presenting records)
End loop :

* If statement is occurs 0 by default memory is 8KB
* If statement is occurs 0 system dynamically provides a memory for that internal table.
* For performance point of view declare internal table with occurs 0 instead of occurs others i.e 100 or 1000
* Internal table work area can hold only one record at a time where as body can hold multiple records.

APPend
Insert
collect
} which can transfer data from work area to body.

* With Open sql data transfers from database table to internal table work area.
* ITAB indicated workarea name
* ITAB [ ] indicates body name
* Internal table work area can hold a last record at last
* In Abap language Data will process record by record
* With Loop end Loop Data will " " " "

```
Data : ITAB like Vendor occurs 0 with Header line
```
→ refers internal table work area

* Using with Header line Internal table can design with all the fields from database table.
* Using without Header line Internal table can design with specific field from db table we are going to work frequently.

```
Select Vno Vname from Vendor into ITAB
Select * from Vendor into corresponding fields of ITAB
```
→ key word.

Whenever database table and internal tables mismatched for providing open sql with select * use 'into corresponding fields of' key word.

If we want to use above keywords it can follow a mandatory i e field names of internal table and vendor table names are same

* Into corresponding fields of doesn't provides performance in Abap. Because it takes time to search the fields.

```
Select VNO VName from Vendor into table ITAB
```
↳ keyword.

* By providing table keyword in select statement user can avoid Append and End select

* provide table keyword in select statement bec it provides best performance.

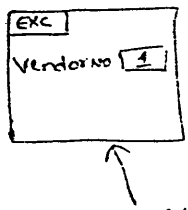
* `Loop at ITAB from 1 to 5` it will present only 5 records can be o/p into presentation server.

** In case of HR-abap data will process based on Validity periods

10/6/15 Lastclass program using selection screens:

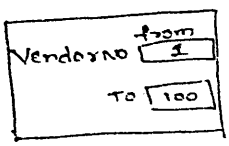
```
Tables : Vendor
parameters : Vendorno like Vendor-VNO.
                (or)
Select-options : VendorNO for Vendor-VNO.
                (or)
Ranges : VendorNO for Vendor-VNO.

Data : Begin of ITAB occurs 0,
        VNO like Vendor-VNO,
        VName like Vendor-VName,
        end of ITAB
```




```
Select VNO VName from Vendor into table ITAB where VNO = VendorNO
                (or)
Select VNO VName from vendor into table ITAB where VNO in VendorNO
                (or)
Select VNO VName from Vendor into table ITAB where VNO in VendorNO

Loop at ITAB
Write : /ITAB
endloop.
```



Selection screens Logic: Users can provide input values to the respective programs.

Selection Screens (Assign before internal table)

- → Parameters
- → select-options
- → Ranges

parameters: using parameters users can able to give Input values but it is not possible to provide range.

syntax: `parameters : vendorno like vendor-VNO`

(parameter name, keyword)

`=` (equal)

In where condition `=` is the operator.

* parameters doesn't create internal table.

select-options: with select options system implicitly create select-options internal table. which have sign, option, low, high fields.

Syntax:

`Select-options: vendorno for vendor-VNO`

in where condition `in` is operator

Sign	option	low	high
I	B/W	1	100
(or)	1-100		
E	Logical Expression		

'I' for including apart from the range

'E' for excluding apart from the range

By default sign is including

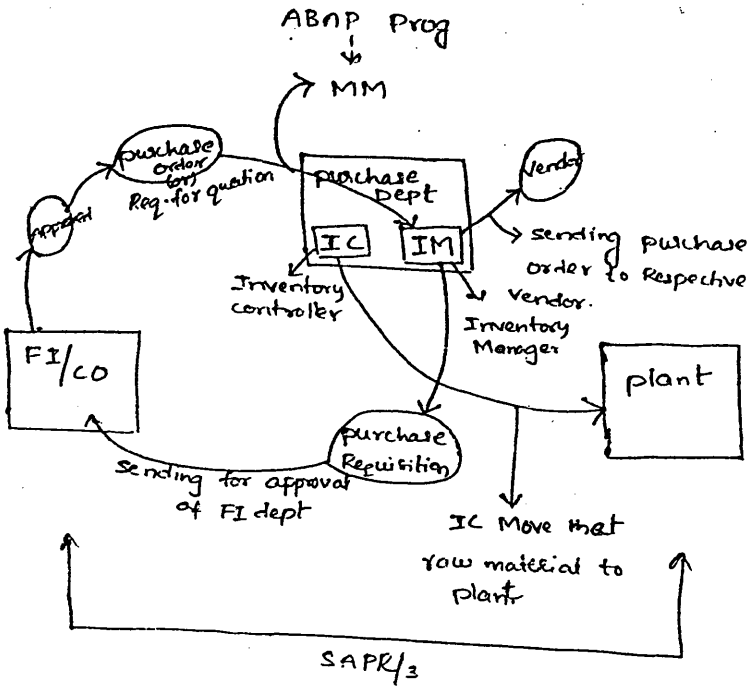
Ranges: user has to define an internal table with fields sign, option, low & high explicitly but it is outdated one now a days not using it.

syntax: `Ranges : vendorno for vendor-VNO`

in where condition `in` is operator.

Main Difference b/w **parameters** and **select-options** is:

1. Using parameters we are not possible to provide Input range but incase of select-options it is possible.
2. parameters will not create internal table but select-options will create internal table with sign, option, low, high fields.



LFA1 (Vendor master table)
- (P) → LIFNR (Vendor Account No)
- → LAND1 (country of the vendor)
- → NAME1 (Name of the vendor)
- → ORT01 (City of the vendor)

EKKO → purchase document header details table
- (P) → EBELN (P. order no)
- → AEDAT (Date on which P. order created)
- (F) → LIFNR (Vendor Ac no)

purchase order format:

purchase order no 100

purchase Req No 1000

Vendor acc no 1

Date . . .

Items	Material	Qty	price
1	X		
2	Y		
3	Z		

EKPO → purchase document item details table
- → EBELP (Item no)
- (F) → MATNR (Material no)
- (F) → EBELN (purchase order no)

MARA → Material Master table
- (P) → MATNR : Material No

* Without Masterdata there is no header data, without header data there is no item data.

DEMO:

Go with abap edition (SE38)

provide program name by following naming condition convention

ZVendor prog

Under subobjects select [Attributes] nothing but properties of a program.

Go for create it displays attribute screen provide

Title : [Vendor details program]

Type : Prog.type → ABAP → programming types

1. Executable prog types (stand alone programming)
2. Function Group
3. Subroutine pool
4. include

} for Reusability purpose but not possible for executing directly so call executable prog type.

5. Module pool → For abap transactions.

select type is : [Executable program]

Status : [SAP standard production program] provides program environment (post implementation)

Application : [Material Management] for which functional working

ctrl + s

Development class [ ] empty

If we are working with executable type it starts with Report

```
Report
Tables : LFA1. " vendor Master table  ( provides work area)
*  selection screen
   Parameters : vendor like LFA1-LIFNR.
*  Internal table
   Data : Begin of ITAB occurs 0,
   LIFNR like LFA1-LIFNR,
   NAME1 like LFA1-NAME1,
   ORT01 like LFA1-ORT01,
   LAND1 like LFA1-LAND1,
   End of ITAB.
```

* Open sql

Select LIFNR NAME1 ORT01 LAND1 from LFA1 into ITAB
where LIFNR = Vendor.

Append ITAB.

End select.

* processing logic

Loop at ITAB.

write : / ITAB-LIFNR , ITAB-NAME1,
ITAB-ORT01 , ITAB-LAND1.

Endloop.

ctrl+S

ctrl+F2 (syntax checking)

ctrl+F3 Activate

F8 for execute.

Enter parameter NO : 100

Then F8

Displays a record of 100 details in selection screen.

11/7/05 Matchcode objects: for search help i.e it provides permisable values as a table. F4 is used for search help

DEMO: Go with abap editor

provide the program name ZYVendorProg → what we designed in last class.

Go with chang mode

parameters : Vendor like LFA1-LIFNR Matchcode object YZOB
↳ Object name ar

Double click on this object YZOB ↵

Go with elementary search help ↵ (in this case only one field in selection screen

provide description for search help : Search help for DEMO

provide selection method : LFA1 LIFNR table where field exists.

provide search help parameter : LIFNR on which field we are implementing search help.

Enable checkbox for IMPORT ☑ & EXPORT ☑

provide LPos : 12 (Left position)

SPos : 12 (Sequence position)

ctrl+S

ctrl+F3

F3

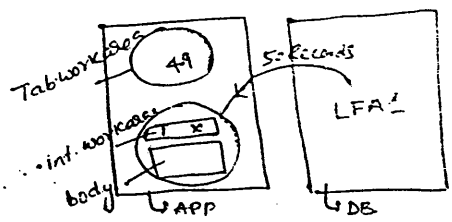
Activate the program (ctrl+F3)

F8

press F4 for check and import values i.e it will display a records list avilable in database table.

* In this program we comment the parameters statement and execute the program it will executes.

* If we use select-options statement and comment it and execute the program it will show error because internal table work area stores only one record at a time it will send it to body then another records comes to work area. This record before coming to the internal table it will store in Table work area so it needs work area.



Background execution: (under subobjects) ⊙ VARIENTS

Go to Abap editor menu

select ⊙ VARIENTS (it can provide an input values required for background process)

go to chang mode

provide varient name YVAR

Go for create

Keep input values in this program i.e window

Go with attributes button (second button in left side)

provide description DEMO

ctrl+S

Documentation useful for Abap programmers and end users.

Documentation is for abap prog help it is for post implementation

Text elements:

parameters, select-options length should be 8-character length.

Select Textelements

go to changmode

go with selection text

Vendor : Vendor Account number (Provide text)

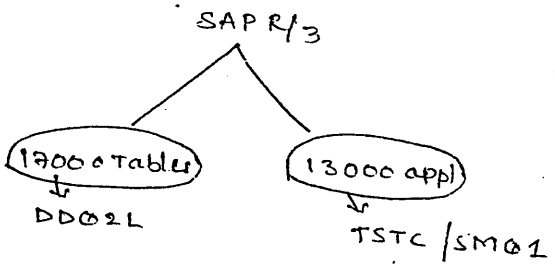
Ctrl+S

Ctrl+F3

F3

F8 (execute program selection screen display it)

* With the text elements user can provide description for selection screen fields.



Database table for SAP table names : DD02L

Database table for SAP Transaction codes : TSTC

SM01 is a transaction code for SAP transaction codes

↳ System Maintenance.

ABAP

↓

Database structure

↓

Total 172 variables

SY-DATUM : system variable for Date (Provides current date of system)

SY-UZEIT : " " " TIME (" " TIME " ")

SY-MANDT : " CLIENT (provides client number which we working)

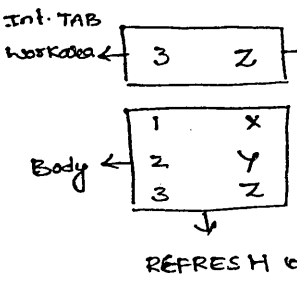
↳ Return code.

SY-SUBRC = 0 successfully processed

SY-SUBRC = 1 ⋮ 20 } for Respective errors.

13/05

Internal table Keywords:



CLEAR : It is a Keyword which refreshes memory of int table work area but it doesn't effect the body.

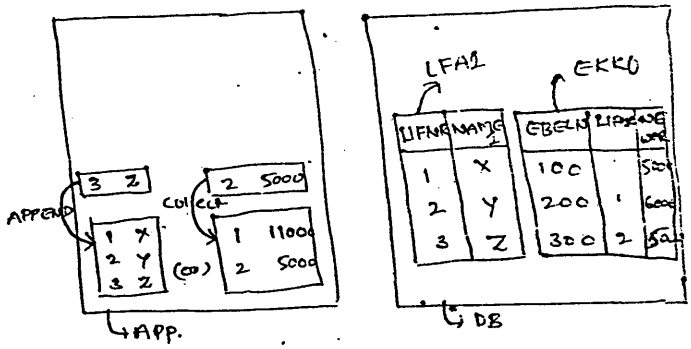
syntax: `CLEAR ITAB.`

REFRESH (or) FREE are the keywords which clears memory of an internal table body. But it doesn't effect on workarea.

Syntax: `REFRESH ITAB.`

`FREE ITAB.`

FEW MORE KEYWORDS: 1. APPEND.
2. INSERT.
3. COLLECT.



APPEND appends the records sequentially in internal table body.

With INSERT keyword new records can be inserted either before or after an existing records in internal table body. i.e we can insert records wherever we want in Int. tablebody.

COLLECT: It avoids duplicate records in internal table body i.e if above diagram shows one record repeats two times 1-5000, 1-6000 but using collect keyword it will collect records which are same.

* Abap language is an event driven language:

Here we are not providing any event in source code but system will provide by default event i.e START OF SELECTION

How to see that event:

Go with abap editor (SE 38)

select attributes it will display a window

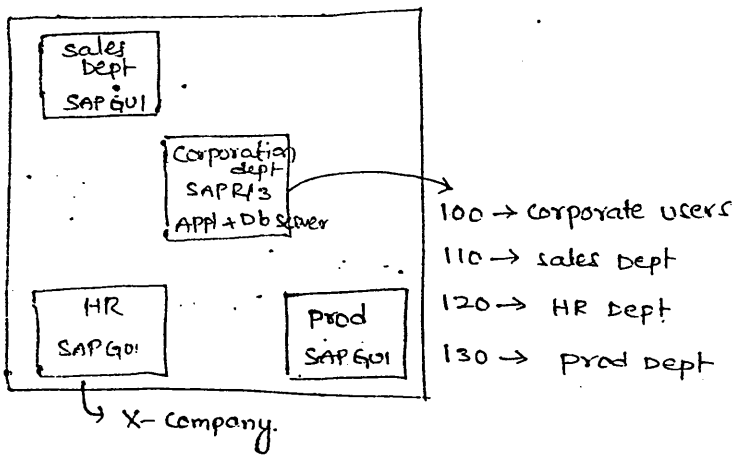
select Debugging option located at right side menu

select calls in that window (calls is events provided by system)

EVENT : START OF SELECTION is a default event which triggers in abap language.

* TRDIR Database table for SAP Programs.

*



If suppose sales dept have clients which are accessable only for sales dept so they are created und sales dept of 110 like

In SAP R/3 weather it is Master table data or Transactional data it is CLIENT DEPENDENT and Version dependent and these tables are start with first Record of MANDT i.e client

Joins program

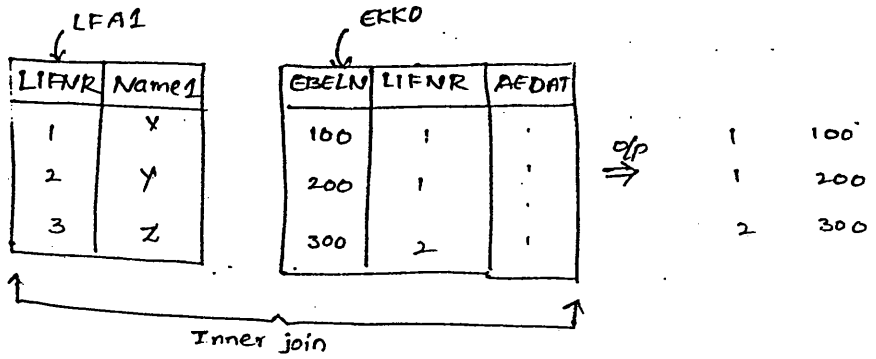
↓

joins concept

↓

Inner join

Example for Innerjoin :



Innerjoin can extract a common data from respective tables.

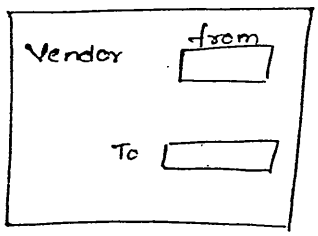
Syntax : Select LFA1 (~) LIFNR, ...

↓

Tilt

DEMO : LFA1 & EKKO using selection screen with select option & Inner join option.

I/P



⇒ Expected o/p

Vendors which provided	P.order for Respective vendor	Date of P.O created

Let us create executable program.

provide tables workarea i.e

Tables : LFA1, EKKO.

Define selection screen with select options.

Select-options : Vendor for LFA1-LIFNR.

Define an Internal table

Data : Begin of ITAB occurs 0,

LIFNR like LFA1-LIFNR,

EBELN like EKKO-~~EBELN~~ EBELN EKKO,

AEDAT like EKKO-AEDAT,

END OF ITAB.

following innerjoin.

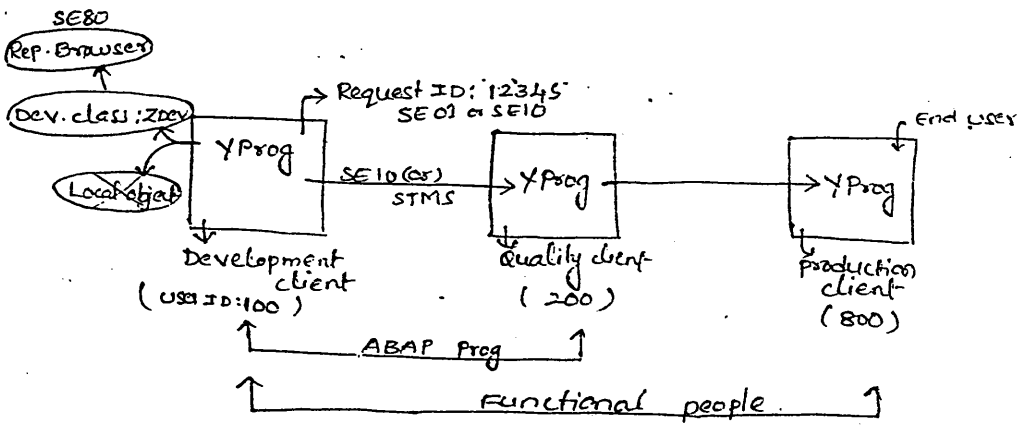
Select LFA1~LIFNR EKKO~EBELN EKKO~AEDAT into table ITA
from LFA1 innerjoin EKKO
provide relation b/w these two tables
ON LFA1~LIFNR = EKKO~LIFNR
where LFA1~LIFNR in vendor.

Loop at ITAB.
write: / ITAB-LIFNR, ITAB-EBELN, ITAB-AEDAT
Endloop.

Ctrl+S
Ctrl+F3

4/7/05

ABAP Programming
↓
client transportation system
(or)
correction and transportation system (CTS concept)



Development client is for designing SAP objects. Any program can be developed by ABAP'ers in development stage.

* If abap objects stores under local object that respective objects are not authorized for transportation. so we go for Development class.

* If abap objects stores under development class that respective objects are authorized for transportation.

If abap objects stores under development class for that respective objects system dynamically generates a request number i.e Request Identity.

Once Request ID is created and transportation is over we are not able to use that ID. for next transportation bcoz it expires so we go for another ID.

Using SE01 or SE10 respective object request number will be released i.e providing authorization to the system for transportation.

If Req. ID shows ✓ (Tick mark) Success full Release transportation

" " " ○ Not released

" " " LOCK Locked.

* ALL TRANSPORTATION JOB CAN DONE BY ADMINISTRATOR.

But before transportation of object administrator will need some information from ABAP programmer i.e objectname, request number, Development class, destination client. According to this specifications it will send to quality client.

* With SE10 or STMS transportation can be done.

* Once transportation is done a copy of SAP object will maintain in destination client.

* While working with SAPR/3 it is not possible to do reverse transportation whereas in other SAP technologies it is possible exa: BW, SCM.

* Once testing is over administrator can transmit program to production client by using SE10 or STMS.

* After transportation if user makes any changes in development with that changes again system generates a request number based new request number if transportation done those changes overwrites in destination client.

Development class creation:- for creating development class database t is V_TDEVC

Go with abap dictionary (SE11)

provide the tbp table for development class : (V_TDEVC)

Go for display

select Utilities

→ clients contents

go for create development class

provide development class name : YZDEV

Short text : Dev. class for MM

software component: HOME↵ (Indicates dev.class which we created restricted for this client only.)

It will display the Request identity number. If we want to change that request number then follow

Creation of Request number:- select the new button left side down

provide description : Request number for Demo↵

We get a new number ↵

Go with abap edition

provide program : YZDEVPG

go for create

Define attributes to the program i.e Title etc... ↵

provide development class which we created : YZDEV

go with save ↵

write : / 'REQNO'.

ctrl+s

ctrl+F3

* For releasing request number of respective object status should be active.

* Why we activate sourcecode is to store that code in sap programs table i.e 'TRDIR'

* If user designs multiple objects without releasing request number for all these objects same number can be assigned

Releasing Request number:

Go with SE10 (T.code for transport organizer)

go for display.

Select request.no from the display i.e what we created

Every object have two request numbers

First no : Main request number

Second no : Dev. correction request no.

While releasing request no use development correction req. no

select Dev. correction request no

go with release directly (Truck symbol button)

ctrl+s

F3

Select Main request number for transportation

go with release directly (Truck button)

We dint get transported number in displayed list so it is transported.

* Using dev. correction number abap objects will be released.

* Using Main req.no abap objects will be transported.

Development class : $TMP is also local object so dont select this.

Objects can be copied across development classes even objects can be copyied from local object to development class.

Predefined Dev. classes are:

1. SLIS
2. SDWA } Dev. classes with training examples.

Go with SE80

Provide SLIS

select display.

5/7/05

ABAP Prog

↓

Error handling (or) (Exception management)

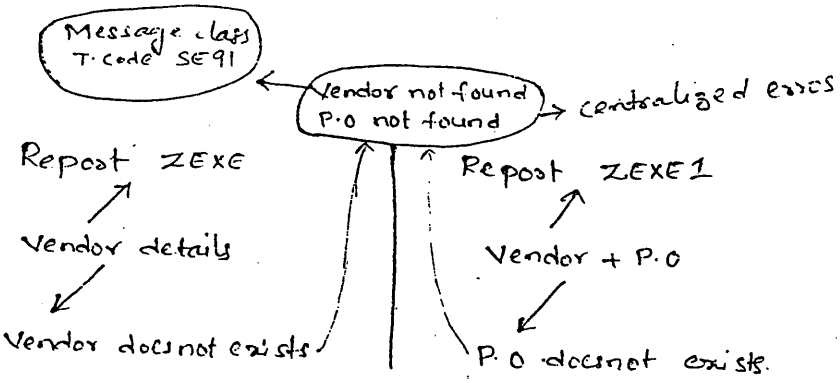
↓

SY-SUBRC → System code for error handling

↳ Return code.

SY-SUBRC = 0 successfully processed

= 1 ⋮ 20 } for respective errors.



Message class creation :-

Go with SE91

Provide Message class name : YZMCLS

↳ any name

Go for create

Ctrl + S

go with Messages

Message number start with 000 to end with 999.

Define Messages according to our requirement

000 – Vendor not found.

001 – P.O. not found.

002 – Vendor found.

Ctrl+S

If SY-SUBRC = 0

Message (I) 002 (YMCLS)

I → Prefixes

YMCLS → Message class name.

Prefixes avilable are:

I – Information.

W – Warning.

E – Error.

A – Abend.

X – exit

S – status.

* Difference b/w warning and error is it will allows you to move the cursor to next position in case of warnings but errors it will not allows you to move control to next position.

* Difference b/w error and abend is: In case of error after displaying error message the control remains within the program only. in case of abend after displaying error message control moves out of the program.

* With exit after displaying error message the control leads to error analysis eg) Dump analysis transaction code is ST22

Error handling syntax:

```
Report ZEXE
If SY-SUBRC = 0
Message I002(YMCLS)
else.
Message E000(YMCLS)
endif.
```

Program with error handling.

Create executable program : YJagadeesh2
Provide tables work area
Tables : LFA1.

Design selection screen with parameter

Parameters : Vendor like LFA1-LIFNR.

Data : Begin of ITAB occurs 0,
LIFNR LIKE LFA1-LIFNR,
LAND1 LIKE LFA1-LAND1,
NAME1 LIKE LFA1-NAME1,
END OF ITAB.

Provide select statement.

Select LIFNR LAND1 NAME1 from LFA1 into Table ITAB
where LIFNR = Vendor.

Error handling logic :

IF SY-SUBRC = 0
Message I002(YZMCLS).
↳ Message class name)
ELSE.
Message E000(YZMCLS) (it will display errors)
(or)
Message X000(YZMCLS) (it can display errors and analysis of errors
so always go for X.)
EndIF.

Loop at ITAB.
Write : / ITAB.
Endloop.

Ctrl+S
Ctrl+F3

Run the Program F8

Provide vendor number : 1000 (existing number)

Display a Message : Vendor found because 1000 is existing in LIFNR

S- status : information message displayed in status bar.

Provide vendor number that not exists in LIFNR

Display a Mesg : Vendor not found in case of E (Error)

: Vendor not found with analysis in case of X (exit

Database table for error handling : T100

Fields are : 1. ARBGB (Application area)
↳ Functional module.

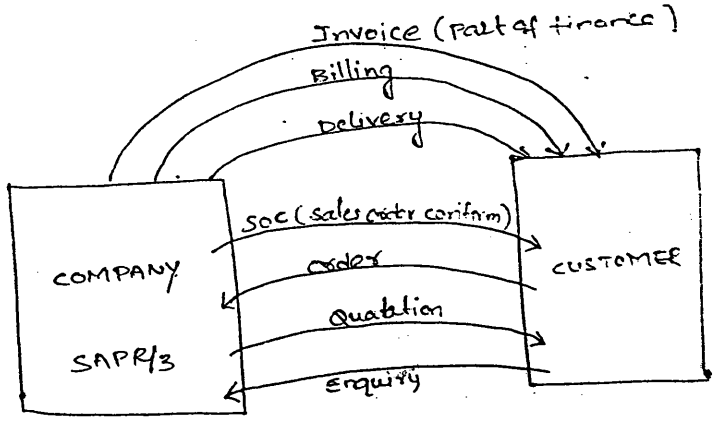
2. SPRSL (Language key)
↳ in which language you designed errors.

3. MSGNR (Message numbers)
↳ 000 to 999

4. Text (Vendor found, vendor not found etc.....

05

ABAP Programming with
↳
SD (sales & distribution)



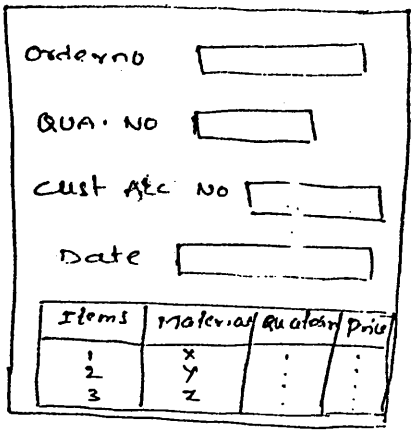
Sales start with enquiry then company can provide a quotation to the customer (price details) according to the enquiry.

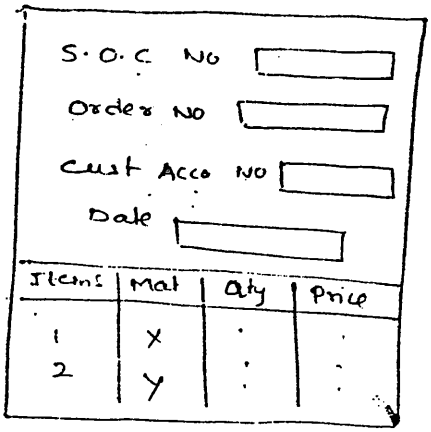
Based on quotation customer give an order to company and based on order company send sales order confirmation to the customer. Then company send raw material delivery and billing of that delivery and invoice of that.

KNA1 → customer master table.

(P) → KUNNR (customer account number)
→ LAND1 (country of the customer)
→ NAME1 (name of the customer)

Sales order confirmation format:





VBAK → Sales document header details table

(P) → VBELN (Sales document no)

→ ERDAT (Date on which the confirmation created)

→ NETWR (Amount of confirmation)

(F) → KUNNR (Customer account no)

VBAP → Sales document item details table.

→ POSNR (item number) EX: 1, 2, 3

→ MATNR (Material number)

(F) → VBELN (Sales document no)

LKIP : Delivery document header details table.

LIPS : Delivery document item details table

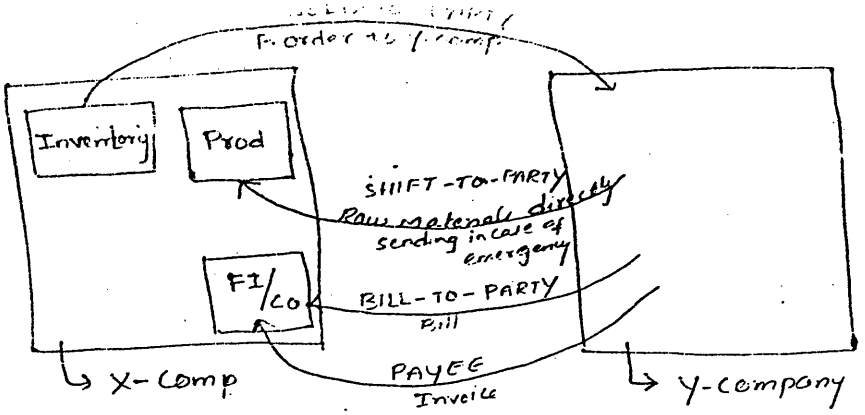
VBRK : Billing document header details table

VBRP : Billing document item details table.

BKPF : Account document header details table (Invoice table)

VBFA : Document flow table in SD

↳ Data flow.



The customer who place an order is SOLD-TO-PARTY

The customer who receive materials is SHIFT-TO-PARTY

The customer who receive bill is Bill-To-PARTY

The customer who receive Invoice is PAYEE.

Partner types:

1. VENDOR (LI)
2. CUSTOMER (KU)
3. Bank (B)

Partner function:

Vendor ——— VN(Vendor)

customer ——— shift-To-Party
Sold-to-Party
Bill-to-Party
Payee.

HOW TO REMEMBER TABLES : If table starts with following words

B - Related to FI/CO

P - " HR

M - " MM

V - " SD

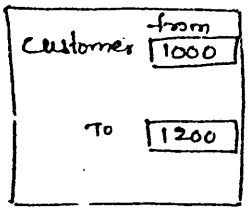
T - " ADMINISTRATION RELATION.

K - " CUSTOMER TABLE.

L - " VENDOR.

E - " PURCHASE ORDER DETAILS.

Ex: with selection screen



Exp. o/p ⟹

Title: customer sales details :

customer NO: 1001

Sales DOC	Date	Amount

Total amount =

Tables are : KNA1, VBAK

In this program we are going to use

Next customer: 1002

Sale DOC	Date	Amount

Total amount =

grand total =

CONTROL BRAKES (or) INTERNAL TAB EVENTS: we have total '5' but now discuss '4' only

1. AT FIRST (Provide the TITLE)
2. AT NEW (Display output)
3. AT END OF (calculate the total amount) } These two use sumkey 'E'
4. AT LAST (For grand total)

LINE TYPE and ROW TYPE concepts:

A — user defined datatype — B

* LINE TYPE is an internal table workarea.
* ROW TYPE is an internal table body.

9/2/05

DEMO:

Go with abap dictionary (SE11)

Select datatype and provide structure name ZVLTYPE

Go for create

Select structure ↵

Provide short text for structure : DEMO

Provide fields which we are using in this DEMO

COMPONET	COMPONENT TYPE
KUNNR	KUNNR
VBELN	VBELN
ERDAT	ERDAT
NETWR	NETWR

Here NETWR field is belongs to amount so apply currency for that

Go with currency and quantity fields

UNDER NETWR provide Reference table and field

NETWR : VBAK WAERK

↳ currency type

ctrl+S
ctrl+F3

** structure is nothing but a line type means an internal tab work area whatever created upto now is line type

ROW type creation:-

Go with abap dictionary

select datatype : ZVRTYPE

Go for create.

select ~~structure~~ TABLE TYPE ↵

↳ Nothing but row type i.e an int table body.

Under row type field provide line type which we declared just now in above

- ROW type : ZVLTYPE

ctrl+S
ctrl+F3

⇒ Here we are using TYPE GROUPS:

- Type groups can hold declaiations.
- It can hold only data types and constants.
- It doesn't holds data objects.
- Type groups are for reusable datatypes and contants.

Let us go with SE80 (T-code for repository browser)

go with edit object

select dictionary

select type group and provide group name ZTG

go for create

provide short text ZTG TYPE ↵

It will display a window starts with Type-pool

```
TYPE-POOL ZTG TYPE

TYPES : ZTG_Wa type ZVLTYPE,
        ZTG_ITAB type ZVRTYPE.
```

Ctrl+S
Ctrl+F3.

The main use of Type groups is to reusability as datatypes.

Let us create executable program (YLRTYPEJAGAN)

by default start with REPORT

```
REPORT JAGANPROG.
TABLES : KNA1, VBAK.
Select-Options : customer for KNA1-KUNNR.
```

* Call type group which we created in previous steps.

```
TYPE-POOLS : ZTG
```
↳ Type group name.

* convert datatypes into data objects.

```
DATA : Wa type ZTG~Wa.
```
↳ work area

```
DATA : ITAB type ZTG~ITAB.
```
↳ body.

* Provide select statement with Inner join

```
SELECT KNA1~KUNNR VBAK~VBELN VBAK~ERDAT VBAK~NETWR
       INTO TABLE (or WA) ITAB FROM KNA1 INNER JOIN VBAK ON
       KNA1~KUNNR = VBAK~KUNNR WHERE KNA1~KUNNR IN CUSTOMER.
```

By providing 'TABLE' keyword in select statement data transfers from database table to directly to internal table body.

By providing 'wa' keyword in select statement data transfers from database table to workarea.

Not required

APPEND wa to ITAB (It will send data from workarea to body)
ENDSELECT.

LOOP AT ITAB INTO WA (Take the data body to workarea)

AT FIRST. (Apply first control brake for title)

WRITE : / 'CUSTOMER SALES DETAILS'.

ENDAT. (control braces ends with ENDAT)

AT NEW KUNNR. (second control brake for ~~next~~ customer no i.e KUNNR)

WRITE : / 'CUSTOMER NUMBER', wa-KUNNR.

ENDAT.

* provide write statement for to display output.

WRITE : / wa-VBELN, wa-ERDAT, wa-NETWR.

AT END OF KUNNR. (control brake for to calculate total amount)

* Provide sum key word

SUM.

* Provide the write statement for calculated total amount.

WRITE : / 'TOTAL AMOUNT:', wa-NETWR.

ENDAT.

AT LAST. (It works based on at end of brace i.e based on total amount grand total will be calculated).

SUM.

WRITE : / 'GRAND TOTAL:' wa-NETWR.

ENDAT.

ENDLOOP.

Ctrl+S
Ctrl+F3
F8

Provide i/p : 1000 to 1200

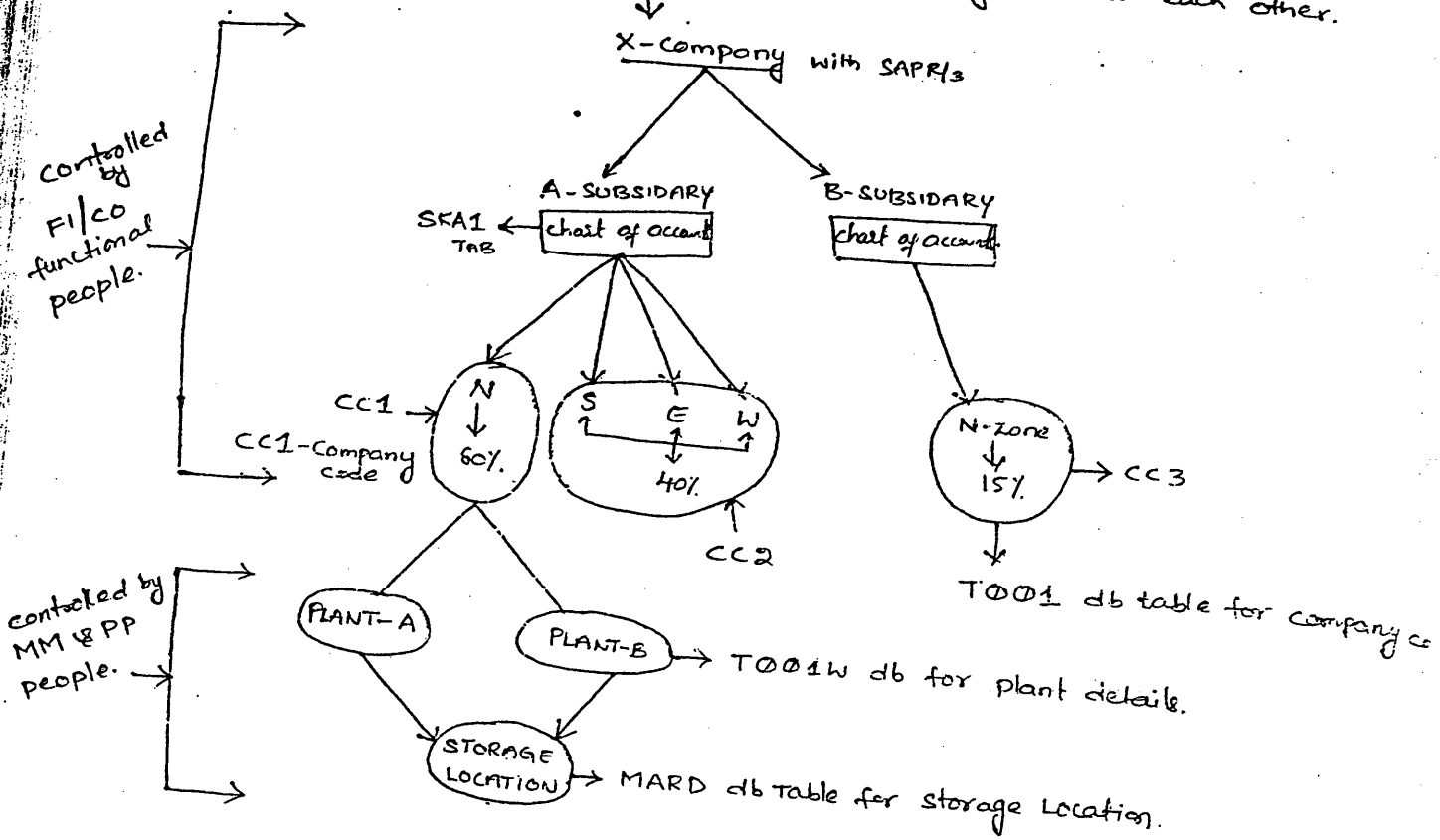
It will display customer wise sales details according an expected output.

* * From 4.6 version onwards ~~have~~ LINE TYPE & ROWTYPE features are added.

SAP R/3 integration:

→ SAP R/3 have Total 45 Functional modules. No has this much of functional modules.

→ These are Integrated to each other.



* Based on company codes accounting data can mantion in SAP R/3.
* T001 is a database table for company codes.
* Chart of accounts sheet is to maintain business charts.
* SKA1 is a db table for chart of accounts.
* T001W db table for plant details.
* MARD db table for storage Location details.
* MARC db table for plant data for material.
* MAKT Material description table
* MARM Unit of measure for material.

Based on above X-Comp flow the following programs can design by ABAP programmers.

1. STOCK OVERVIEW REPORT

TOOIW MARD MARC MAKT MARM

plant ID

↳ selection screen.

EXP. O/p ⇒

STORAGE Loc. for that plant	Material existing	qty	Material Discription

2.

Sales document

No

VBAP-VBELN

EXP. O/p ⇒

Items which contains in that document	Material contains each item	From which plant mat. manufactured	From which storage collected
VBAP-POSNR	VBAP-MATNR	MARD-WERKS	MARD-LGORT

3.

T001 - BUKRS

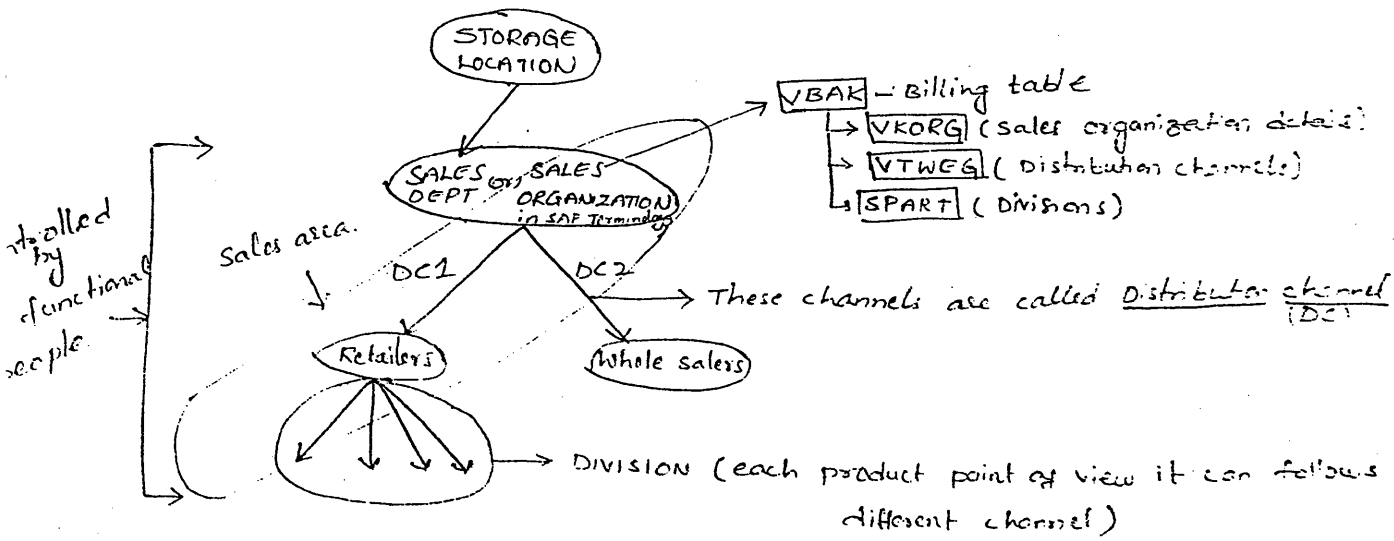
Comp Code

Date to

VBAP - ERDAT

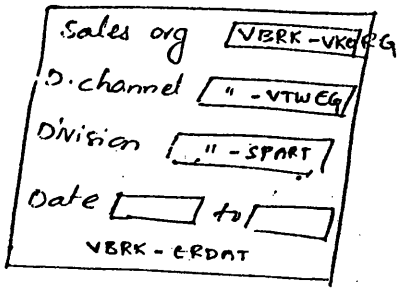
⇒

plants comes Under that code	Material	Revenue
VBAP-WERKS	VBAP-MATNR	VBAK-NETWR.



* Sales organization, distributed channels and divisions is called one sales area

4.



EXP o/p ...

Bills gen Doc. NO	Bill Doc ~~Date~~ Type	Date bill created	Amount of that bill	How many days Have to for next payment	Due date
VBRK-VBELN	VBRK-FKART	VBRK-ERDAT	VBRK-NETWR	TOS2-ZTAG1	TOS2-ZSTG1

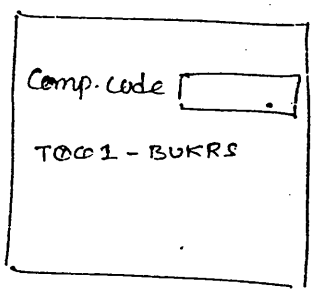
KNBK - customer bank details table

SKB1 - G/L accounts details (General Ledzus)

KNB1 - customer transaction figures table

KNC1 - customer company codes table.

5.



⇒

customers existing on that code	G/L cust of Rep cust	on which date Have cust. created
KNB1-KUNNR	SKB1-KUNNR	SKB1-ERDAT

11/7/05

ABAP Programming

↓

Performance techniques

↓

Report ZEXE

Tables workarea
Define Int. table → Executes in application server.

SQL command → sql command in database server.

Loop
Endloop. → Application server.

Performance tools: in abap language e.g.-

1. Runtime analysis.
2. Sql tracer.

⇒ with Runtime analysis user can check performance of abap programs.

⇒ with sql tracer user can check the performance apart from it user can search tables required for abap programming. -

Go with abap editor

Provide one existing program name

go with Utilities → More utilities → Runtime analysis.

Before Runtime analysis in application tool bar

Select Tips & tricks button. in this window have some sql statements for performance purpose select one sql statement and go with Measure Runtime it will display process time.

In Runtime analysis window select [program] and provide prog name which we want to check performance

Execute it (F8)

F3

In the same ~~window~~ screen go with [other file]

Under otherfile select subset-

and selection option [program]

provide program name which we check for performance ←'

Select the program and it will displays a performance graph

ABAP - abap process execution time

Database - database server execution time.

R/3 system - Application server " "

If any of this execution time is grater than 50% it will not acceptable in Realtime.

To improve performance techniques regarding database point of view:-

1. provide select statements based on index support.
2. provide select statements with table keyword.

3. while extracting large amount of data provide innerjoin.

Regarding application server :-

1. Define internal table with occurs 0 and without header line.
2. provide linetype & Rowtype concepts.
3. provide Reusability concept i.e subroutines, function modules..

Navigation for access all performance examples :-

Go with abap editor

select Environment → performance examples.

Reusability concept: In SAP terminology called as (OR) MODULARIZING TECHNIQUES

Types:
1. MACRO's concept.
2. SUBROUTINES.
3. FUNCTION MODULES.
4. INCLUDES.
5. FIELD SYMBOLS.

These are the programming types we are going to use

1. SUBROUTINE POOL
2. INCLUDES.
3. FUNCTION GROUP.

These programs are not possible to execute directly.

INCLUDES CONCEPT :- Navigation

Go with abap editor

provide program name YVLIN

Define attributes

Type : Include program

Status : SAP standard production

Appl : MM

ctrl+s

Syntax: start with comment include

provide table workarea.

Tables: LFA1.

Define select-options

Select-options : Vendor for LFA1-LIFNR.

Define an internal table

Data : Begin of ITAB OCCURS 0,

LIFNR LIKE LFA1-LIFNR,

LAND1 LIKE LFA1-LAND1,

NAME1 LIKE LFA1-NAME1,

END OF ITAB.

Ctrl+S

Ctrl+F3

Using above include program we can call it into any executable program according to our Requirement and it is called as Reusability.

With INCLUDE keyword user can call an include within an include as well as an include into EXECUTABLE

INCLUDE WITHIN INCLUDE PROGRAM:

Define INCLUDE Program type and provide Name YVLIN1

Let us call in include within include.

INCLUDE YVLIN.

Select LIFNR LAND1 NAME1 From LFA1 into table ITAB

where LIFNR IN VENDOR.

Loop at ITAB.

Write : / itab.

Endloop

Ctrl+S

Ctrl+F8.

Define Executable program type : YVLIN2

Report YVLIN2.

Let us call second include into executable

INCLUDE YVLIN1.

Ctrl+S

Ctrl+F3

Double click on Include program name i.e (YVLIN 1) control will display with Include program.

* INCLUDES are equivalent to header files in other languages.

FUNCTION MODULES:

Syntax:

```
Function Z_FUN
   :
   :
ENDFUNCTION.
```

Z_FUN → Function module name.

While working with function module we can use function group type.

Function group is a collection of function modules under one function group we can maintain maximum of 99 functional modules.

Syntax for calling function modules:-

```
call function Z_Fun
```

Z_Fun → Function module name.

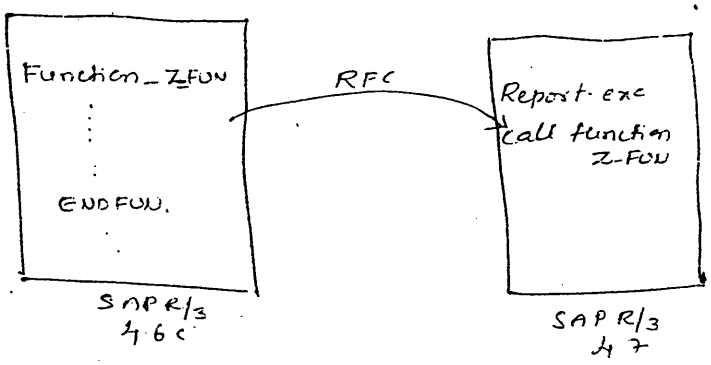
while calling function module programming type is Executable.

Function module types :-

1. Normal
2. RFC (Remote function call)
3. ALV's (Abap list viewer)

* Normal function modules are for abap programming.
* RFC's are distributed enviroment puipose (BDC)
* ALV's are for providing performance in abap language (Reports)

ALV's are part of Normal type. ~~dute~~ due to importance treat it as a different.

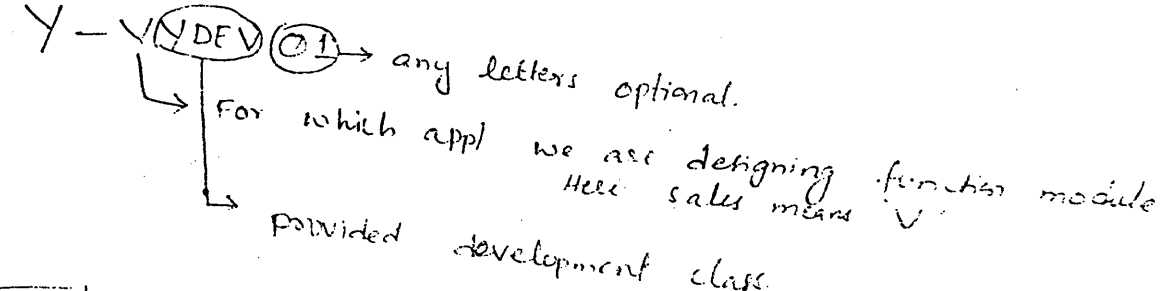


If it is normal function module user can call within the system only.

If it is RFC user can call within the system as well as across the systems.

Naming conventions in Function module :-

Start with Y or Z indicates user defined



SE37 T-code for function builder.

18/10 Function modules: program type is Function group

SE37 – T.code for function buildor.

Navigation for Function group:-

go with SE37

select GOTO
→ Function group
→ Go For creategroup.

provide function group name YVLNGROUP

provide short text : Group for MM

Ctrl+s

Navigation for dynomically systems genereated function group:

select GOTO
→ Function group
→ change group.

provide function group which we created above YVLNGROUP ←

It will display a window in that select MAINPROGRAM

It will display a system genereated programs

Double click on first Include see the status it is inactive then activate it but type time of activation it will display a message have syntax essors activate it anyway click yes.

Ctrl+s.

F3

Once again activate it

Go with SE37

Function Module Creation :- ↗

provide the function module name start with Y or Z and (_) under Square is mandatory Y_MYDEV001

→ Development class

→ MM (for application)

Go for create.

provide function group : YVLGROUP (under which group it exists)

short text : Module for DEMO ↵

select Attributes: (properties of a function module)

under attributes select

processing type is ⊙ Normal function module

select interface IMPORT : Importing vendorno

Vendor LIKE LFA1-LIFNR

select interface EXPORT : Exporting P.order.

Pordes LIKE EKKO-EBELN

Let us go with tables

provide table name which depends on o/p

KTAB LIKE EKKO

This is equivalent to with header line.

Go with Exception

* In Function modules exception management designed implicitly.
* where as in subroutines user has to design exception management explicitly.

Provide the exceptions :-

Vendor_NOT_Found

PO_NOT_FOUND.

GO with source code

write the logic after the comments.

Select EBELN AEDAT FROM EKKO INTO corresponding fields of TABLE KTAB where LIFNR = Vendor.

Vendor is import parameter.

ctrl+S
ctrl+F2

Define · executable program type (SE38)
Define properties

go with 4 line space from Report
go with [pattern] in application tool bar.
select call function
provide the function module name : [Y_MYDEV001] ↵

it will print function module statements in Report* executable program

Go back to free space (4 lines space)
provide tables workarea
Tables : LFA1, EKKO
Design selection screen for exporting vendor no
parameter : vendorno like LFA1-LIFNR
Define int. table
Data : ITAB like EKKO occurs 0 with header line

in funct. mod statements {
Vendor = vendorno
KTAB = ITAB

After endif
Loop at itab
write : / ITAB-EBELN, ITAB-AEDAT.
Endloop.

ctrl+S
ctrl+F3
F8.

---

* SAP has 11,000 predefined function modules.

[TFDIR] → db table for predefined function modules.

SUBROUTINES : syntax for defining subroutines.

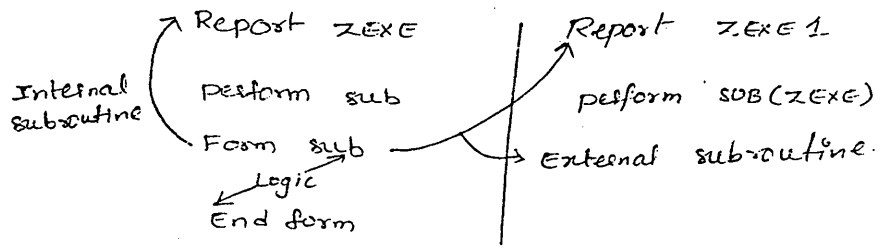
```
FORM <subroutinename>
.
.
.
ENDFORM.
```

`Perform <sub>` syntax for calling subroutines

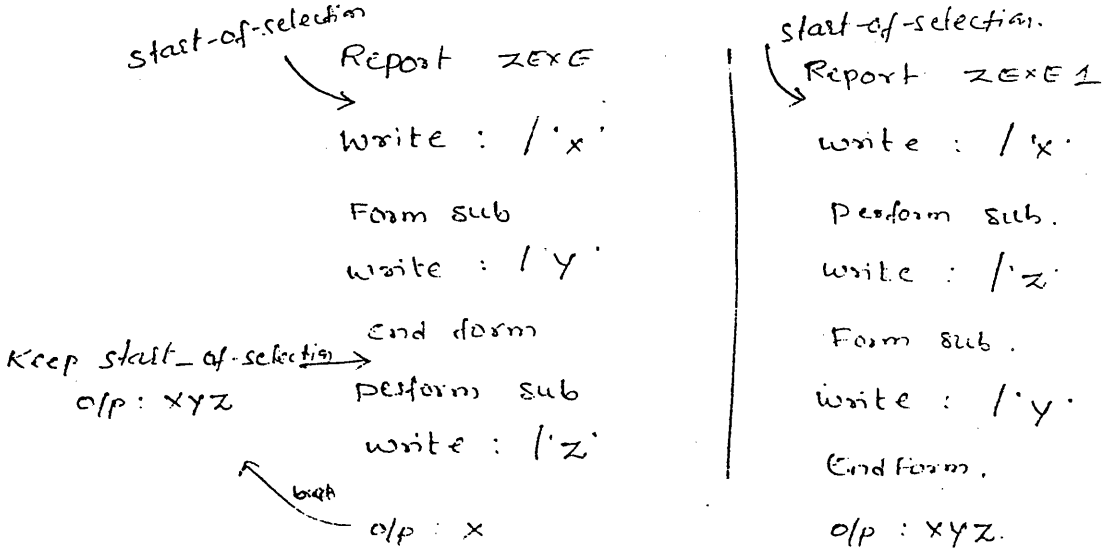
Subroutine types:

1. Internal subroutines.
2. External subroutines.

Example for Internal & external:



Abap programming is a collection of processing block processing block means logic between an event and subroutine



While defining subroutines user can define at the end of programming or else by providing start-of-selection explicitly user can define some where in b/w the program.

Passing Parameters in Subroutines :-

Keywords are Using / changing for passing parameters.

FORM
formal
:
:
Endform

Perform (actual)

If user pass parameters during definition of a subroutine those are formal parameters.

If User pass parameters during calling of a subroutine those are actual parameters.

13/7/05

SUBROUTINES: Example program for call back

```
Report ZEXE
Data : x type I value 1000.
       y type I value 3000.
       z type I.
* call subroutine
perform SUB using x y z.
z = x + y
write : / z.

* subroutine
Form SUB using value(x) value(y) value(z).
```

(in this statement if delete value and provide x y z. it is called call by reference then o/p is 4000/4000

Form → Subroutine name → syntax for call back value
using → keyword for passing parameters.

```
x = 500
y = 500
z = x+y
write : / z.
EndForm.
```

o/p : 1000 / 4000 (500+500 / 1000+3000)

passing internal table in subroutines :-

Report ZEXE

Tables : KNA1

Data : Begin of ITAB occurs 0,

KUNNR LIKE KNA1-KUNNR,

NAME1 LIKE KNA1-NAME1,

END OF ITAB.

* Let us call subroutine

Perform sub tables ITAB.

Loop at ITAB.

write : / ITAB.

Endloop.

* Define the subroutine at the end of the program

Form sub tables JTAB like ITAB[ ].

↳ Keyword for passing int table in subroutine.

* Provide select statement

~~Select LIFNR LAND1 NAME1 from LFA1 into table itab where~~

~~LIFNR between vendor-low and vendor-high.~~

* ~~Provide loop write endloop statements.~~

~~Loop at ITAB.~~

~~write : / ITAB.~~

~~Endloop~~

~~Endform.~~

~~clolif3~~

Define - executable program type ... YJAGANPROG

Tables : LFA1.

Select-options ... Vendor for LFA1-LIFNR.

let us call subroutine

Perform subr(

Select KUNNR NAME1 from KNA1 into table JTAB.

Endform.

* Bydefault Subroutines are call by reference. type.

Prog. type: Subroutine pool is collection of subroutines we can define N no. of subroutines under subroutine pool.

Demo: (External subroutine type)

go with abap editor.

Provide the program name YJAGAN

Define attributes

Type : Subroutine pool

Status : SAP standard prog

Appl : MM.

Syntax start with program.

Tables: LFA1.

Data: Begin of ITAB occurs 0,

LIFNR LIKE LFA1-LIFNR,

LAND1 LIKE LFA1-LAND1,

NAME1 LIKE LFA1-NAME1,

End of ITAB.

* Define subroutine

Form sub using value(vendor_Low) value(Vendor_High).

↳ keyword for passing parameters.

* provide select statements.

Select LIFNR LAND1 NAME1 from LFA1 into table itab where

LIFNR between vendor-Low and vendor-High.

* provide loop write endloop

Loop at ITAB.

write : / ITAB.

Endloop.

Endform.

Ctrl+F3

Define executable program type YJNGNN

Tables : LFA1.

select-options : vendor for LFA1-LIFNR.

Let us call subroutine.

perform sub(y.....) using vendor-low vendor-high.
↳ Subroutine pool name

Ctrl+F3

F8

MACRO'S :- Syntax

Define <macroname>
:
End-of-Definition.

Features of MACRO'S :-

- Macros has designed with place holders concept
- In one micro user can maintain maximum of '9' place holders
- within an macro is possible.

Demo : Tax application with macro

* Define executable program
  Report zexe
* Design selection screen with parameters
  Parameters : Income type P decimals 4.
  Data : Tax type P Decimals 3.
* Apply tax logic in macro
  Define macro
    ↳ macro name
  Tax = '0.1' * &1 → place holder
  End-of-definition.
* Let us call macro
  Macro Income.

Write : / Tax.

2 Demo:

```
Report ZERE
Data : A type I value 5000,
       B type I value 1000,
       C type I.

Define cal
       └→ Macro name
       C = &1 &2 &3.
       Output &1 &2 &3 C
       End-of-definition
       Define output
Write : / 'The result of &1 &2 &3 is :' &4.
End-of-Definition.
cal A+B
cal A-B
```

TRMAC : db table for predefined macros for HR

SAP provided 65 predefined macros for HR.

14/7/05

Function modules :-

Background Job scheduling

Background Job scheduling is for long running programs
ex: yearly sales report

which service provided by Background service.

Methods of Background Job scheduling :-

1. Full control method.
2. Job varient method (it is outdated one)

SAP provided Function modules for full control method :

1. Job_Open
2. Job_ submit
3. Job_ close.

BTCEVTJOB : Transparent table for background Job searching.

→ Jobname(32) (provided an identity required for background scheduling)
↳ length.

→ Jobcount (provides no.of programs involved in background.)

Demo with full control method :-

Go with abap editor

provide an existing program in abap editor YJaganprogram

select Varients and go with changemode.

provide varient name YJagan

go for create

provide the input value

plant id : 1000

go with attributes

provide description for varient

ctrl+s

let us create an executable program (SE38)

parameters : Job(32) (for providing Job name at runtime)

Define Data object for calculating count

Data : count like BTCEVTJOB-JOBCOUNT.

go with pattern

call function : JOB_OPEN ↵

Jobname = Job

Delete the comment for Jobpost and Jobcount

Jobcount = count

* JOB_OPEN will create an identity required for background process.

Provide the write statement with that system variable after endif

Write : / sy-subrc.

Go with pattern

call function : JOB_SUBMIT ↵

AUTHCKNAM (Authentication name) = 'SAPUSER' (provide uses id)

PROVIDE JOBCOUNT = COUNT.

JOBNAME = JOB.

Delete the comment for report and for varient

REPORT = 'YJAGANPROGSALES' (provide which program are going to scheduling background.)

VARIENT = 'YJAGAN' (provide varient name created above steps)

* Based on Jobname JOB_SUBMIT will submit the report in background.

Let us keep write statement

WRITE : / SY-SUBRC. (so that we find Job submitted successfully or not)

Go with pattern

call function : JOB_CLOSE ↵

JOBNAME : JOB.

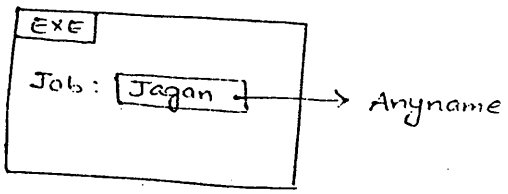
JOBCOUNT : COUNT.

WRITE : / SY-SUBRC. (TO find out Job closed successfully or not)

Ctrl+S

Ctrl+F3

Go for execute F8 it will display a window



go for execute then it display

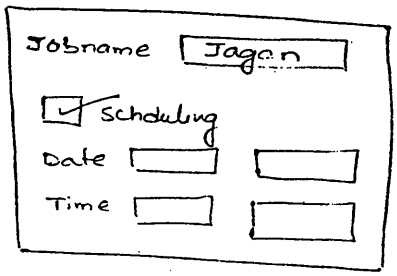
O/P : ⊘ (open successfully)

⊘ (submit ")

⊘ (closed ")

Go with SM37
↳ system maintenance

SM37 → T. code for background scheduling. it will display a wind like



(provide date for when we want to execute that program.

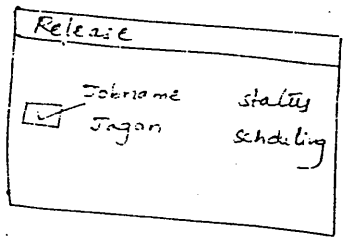
(provide time)

Above window provide Jobname : Jagan

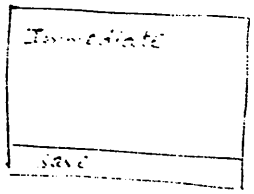
Enable checkbox for scheduling.

Provide date & time

Execute the program (F8) it will display a window like

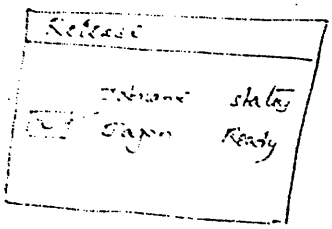


Select the entry and go for Release option then it display

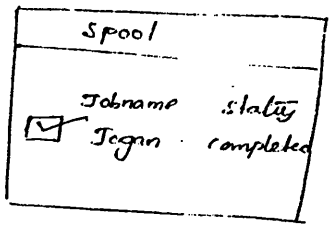


Select option Immediate and save.

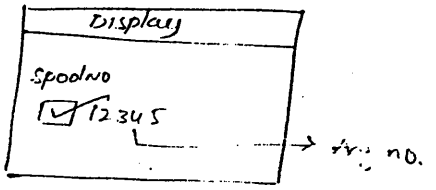
* RELEASE is nothing but providing authorization to the system for executing the program in background.



Select the entry and go with Release again



Select the entry go with spool and it displays



Select spoolno go for display it will display output.

15/7/05

DEBUGGING CONCEPT :- use (/H) indicates debugging in T-code call.

1. Single step debugging → F5 (Function key)
2. Execute → F6
3. Return → F7

* Single step can debug line by line
* With execute mode changes from debugging to normal mode
* With Return user can avoid external programs debugging.

DEMO WITH Single step :-

go with abap editor. SE38

Let us provide an existing program YJaganprogsales

provide /H in T-code field for debugging mode can be switched on.

(or)

Abap editor we have debugging option we can select that also

Under debugging :- FIELDS : nothing but an internal table fields.

With fields user can check data transferring from database table to internal table fields.

provide which internal table we want to check

ITAB - LINE

press F5

Table :- nothing but an Internal table.

provide internal table existing in program : ITAB.

Press F5

cap symbol shows workarea.

index shows body.

With this user can check data transfering from db table to workarea and WA to body.

go with Watch Points :- With watch points user can check data transfering based on specific range.

under watch points

go for create watchpoint

provide which field data we want to check : ITAB-LIFNR

provide relational operator [<] (less than)

provide value : 1000 ↵

go to [fields] option under fields

Let us provide fieldname which we defined in the watchpoint

ITAB - LIFNR

press F5

It will give a message from 1 to 1000 as watchpoint reached after it will show error in status bar.

Calls : provide list of events contains in a program.

start-of-selection is an syst. provided event

Overview :- provides abap program functionality basically i.e How many subroutines are there or Includes are there etc.

Settings : is for system defined programs which it is using for debugging.

Demo with execute debugging :-

Before select statement provide BREAK-POINT.

Break point is a keyword for debugging.

From select statement onwards statement in debug mode before statements are normal mode. i.e Data declarations.

Break-point doesn't provides user specific debugging. i.e if if user is SAPUSER, or SAP1 user whatever may be the user it is debug.

BREAK <USERID> is a keyword which provides user specific debugging.

With dynamic breakpoints debugging restricted for that select only. Select the statement which we want to debug (drag that) and click Stop symbol in application tool bar for dynamic debugging.

SAPS

M. Jagadesh BE
Meet me at Jagadeesh_6@yahoo.co.in

18/9/05

ABAP TRANSACTIONS

programming type is : Module pool (Type for abap tr)

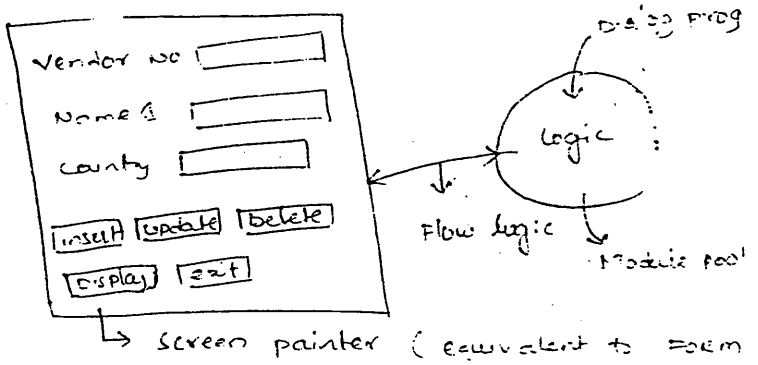
ABAP transaction program is called as dialog programming.

With dialog programming user can do data extraction as well as data manipulation. but in case of abap programs only for data extractions.

- with abap transactions user can modify SAP provided applications.
- with abap transactions a new application can be designed in SAP R/3 environment.

APPLICATION Designing :-

Using screen painter we can design applications in SAP R/3.



for logic designing prog type is Modulepool that logic is dialog prog

FLOW LOGIC provides communication between screen painter and module pool program. Flow logic is part of screen painter.

Events in ~~appl~~ abap to :

abap transaction events.

1. PAI (process after input)
2. PBO (process before output)
3. POV (process on value request)
4. POH (process on Help request)

PBO : PBO triggers before screen display Ex: Tr.code.

PAI : PAI triggers after providing input values Ex: insert, delete

POV : POV triggers with F4 Function key

POV is for search help.

POH : POH triggers with F1 function key.

POH is for user defined documentation.

Flow logic by default have PAI & PBO where as POV & it user has to provide explicitly in flow logic.

YJaganmodule

DEMO ON FLOW LOGIC : (Using command field) & without int. table

go with abap editor

provide program name define attributes : YJagenmodule

Type : Modulepool

Status : SAP st.

Appl : MM

Ctrl+S

go with SE51 T code for screen painter.

provide module program name which we design above step YJagenmodule

provide screen no. 0100. → it should be any

go for create.

provide discription : Vendor details appl

Ctrl+S

go with layout in appl. tool bar (under layout we are going to design application)

For selection FIELDS :-

select goto

→ secondary window

→ Dictionary prog fields.

Let us provide table name : LFA1 ↵

Select the fields which we need for application designing.

Select
LIFNR
LAND1
NAME1 ↵

Move the control which position or location we need that fields,

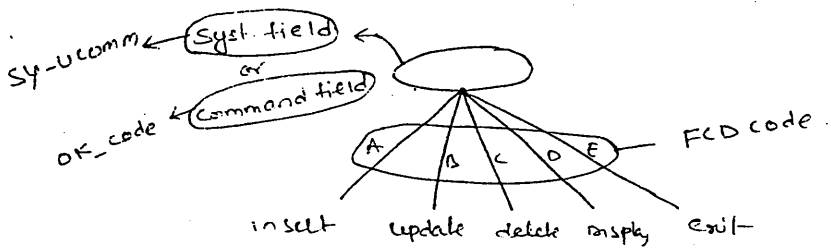
Select push button Drop it in layout-

Double click on push button

Provide pushbutton name : INSERT

Discription : INSERT

FCODE code : Function control code.



With Respective FCT codes system field or command field will control dialog programming.

while working with command fields FCT code lengths should be maximum of 4 character length. This definition is applicable for upto version 3.1H.

provide FCT code for insert : INSE (any one)

close the property sheet-

Repeat the same process for delete, display and exit etc.

go with element list in appl toolbar.

provide command field [OK_code] in OK field. (Here we u... command fie... syst fie...

ctrl+s.

go with layout in toolbar

select flow logic in toolbar

we see ... events PBO, PAI

Delete comments for PAI program because our appl. is insert, delet, disp appl type

Double click ... USER_command_0100 ↵

it will display a module pool window by using flow logic

start declarations in modulepool.

Tables : ...

Data ... code (4)

└→ Fct code with 4 character

provide ... b/w module & endmodule

CASE ... CODE.

WHEN ...

INSERT ....

WHEN ....

DELETE ....

WHEN ....

UPDATE ....

WHEN ....

Select * from LFA1 where LIFNR = LFA1-LIFNR

Endselect

WHEN 'EXIT'

LEAVE program. (is a keyword it can terminate the appl)

ENDCASE.

ctrl+s

ctrl+F3

F8

Activate flow logic events.

go with SE93 (T code for designing Transaction code for an appl)

provide T code : YJaganTcode ↵ or wish

go for create

short text : Demo

select dialog transaction option ↵

provide module pool program : YJaganmodule

screen no : 0100

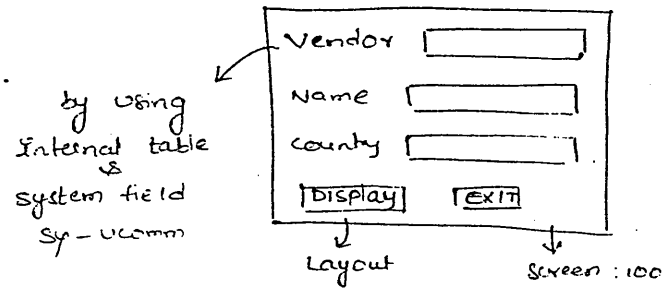
ctrl+s

provide T code which we designed above and work with appl.

19/7/05

YJaganmodule1

ABAP Transactions: DEMO by using system field.



Let us create Modulepool program YJaganModule1

Type Modulepool

go with screen painter SE51

provide Modulepool program which we design above : YJaganmodule1

provide screenno : D100

go for create

provide discription for screen 0100.

: vendor application with internal table

ctrl+s

go with layout

selecting screen fields:

select GOTO → secondary window → dictionary program fields.

LFA1 ↵

Select the fields which are required for application

LIFNR
LAND1
NAME1

Drop the fields in layout.

provide the options display & exit by using push buttons

ctrl+s

go with flow logic

Delete the comment for PAI program
Double click on that program
Control leads to module pool program ↵ ↵

Start declaration

Provide table work area.

Tables : LFA1

Data : Begin of ITAB occurs 0,
LIFNR LIKE LFA1-LIFNR,
LAND1 LIKE LFA1-LIFNR,
NAME1 LIKE LFA1-NAME1,
END OF ITAB.

Let's go with module & endmodule write logic in them

Write the logic for display and for exit

```
Case sy-ucomm.
WHEN 'DISP'.
Select LIFNR LAND1 NAME1 from LFA1 into ITAB where LIFNR =
                                                LFA1-LIFNR
APPEND ITAB.
ENDSELECT.
WHEN 'EXIT'.
LEAVE PROGRAM.
ENDCASE.
```

ctrl+s
ctrl+F3.

MOVE-CORRESPONDING ITAB TO LFA1

↳ screen fields

Statement for transfering data from internal table to screen fields. it is always under PBO.

F3

Delete comment for PBO program

Double click on this program ↵ ↵

Let us keep logic btw module & endmodule

Move-corresponding ITAB TO LFA1

Ctrl+ F3

F3

Ctrl+F3 (activate flowlogic keywords)

go with SE93 for T-code design.

provide the T-code name YJaganmodule1

↳ It is our wish

go for create

provide short text : Demo

Select dialog transaction ↵

provide the module pool program name : YJaganmodule1

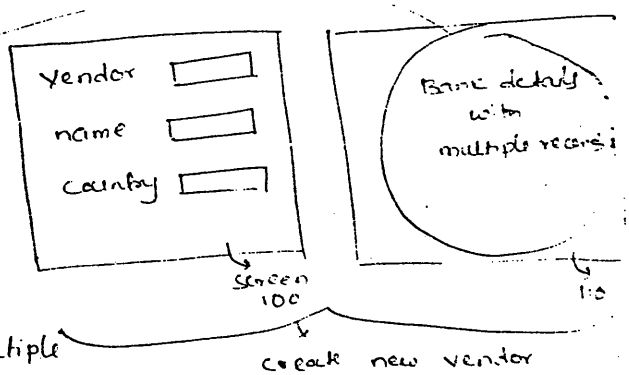
Screen no : 0100 ↵

Ctrl+S

go with designed T-code and use application

ABAP Tr:

1. Table controls
&
2. step loops

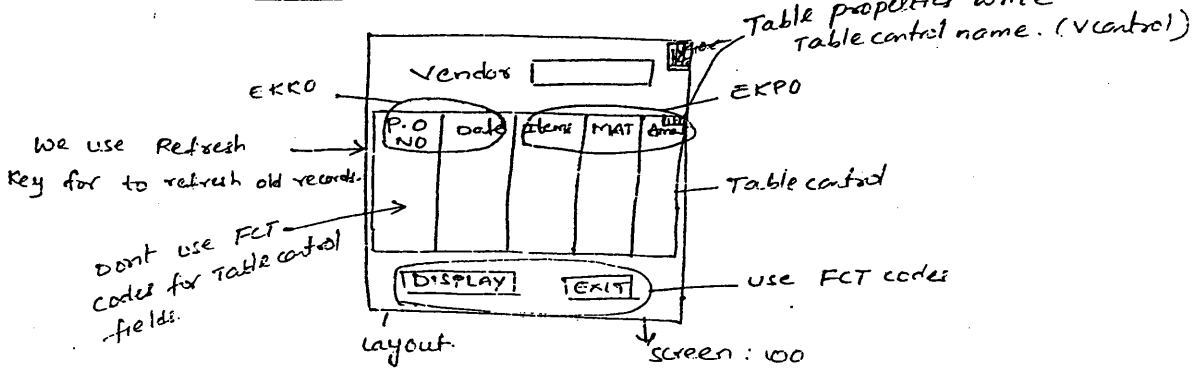


- With table controls & step loops user can display and insert multiple records.
- step loops is outdated concept
- Table controls are enhancement to stepdoops.
- Table controls we can used for DML & DDL operations

- In case of table controls user have horizontal & vertical scrollers.
- Where as In step loops user have vertical scrollers only.
- In case of table controls cursor controlling logic implicitly.
- Where as in step loops user has to design cursor controlling logic explicitly.

21/7/05

DEMO ON TABLE CONTROLS:



* FCT codes are required whenever user dialog is required.
  (or) → user interaction

* Avoid FCT code for table controls.

Let us create Module pool program: YJagadeeshTablecontrol.

go with Screen painter SE51

provide program name : YJagadeeshTablecontrol.

Screen no : 100

go with layout

provide the vendor field by same procedure in last demo

Select the Table control button (from last to fourth) drop it in layout

Double click on table control (Righ mat corner we have Table control symbol)

Define properties

Name . Vcontrol
↳ Any name

Select the fields from EKKO table. (EBELN, AEDAT)

Drop the fields in the table control.

Select the fields from EKPO (EBELP, MATNR, NETWR)

Drop it in table control with extension of last fields.

provide the options display & exit by using button.

go with flow logic

Delete comment for PAI program

Double click on that program

start the declarations.

provide table workarea

Tables : LFA1, EKKO, EKPO.

Data : Begin of ITAB occurs 0,
EBELN LIKE EKKO-EBELN,
AEDAT LIKE EKKO-AEDAT,
EBELP LIKE EKPO-EBELP,
MATNR LIKE EKPO-MATNR,
NETWR LIKE EKPO-MATNR,
END OF ITAB.

Provide the table controls declarations.

Controls : Vcontrol type tableview using screen '100'.
- Vcontrol → Name of a table control
- Controls → syntax for define table control

go with module, endmodule provide the logic for display & exit

case SY-UCOMM
when 'DISP'.
Refresh ITAB.
Select EKKO~EBELN EKKO~AEDAT EKPO~EBELP EKPO~MATNR
EKPO~NETWR into itab from EKKO inner join EKPO
on EKKO~EBELN = EKPO~EBELN where LIFNR = LFA1-LIFNR.
Append ITAB.
endselect.
when 'EXIT'.
Leave program.
endcase.

Apply loop & endloop for PAI program i.e next line to that

Loop at ITAB.
ENDloop. } going to process record by record (PAI)

Delete the comment for PBO program

Double click on that ↵

provide Move statement b/w module & endmodule

Move - corresponding ITAB to EKKO.
Move - corresponding ITAB to EKPO.

Activate (ctrl+F3)

F3

Apply loop & endloop for PBO i.e

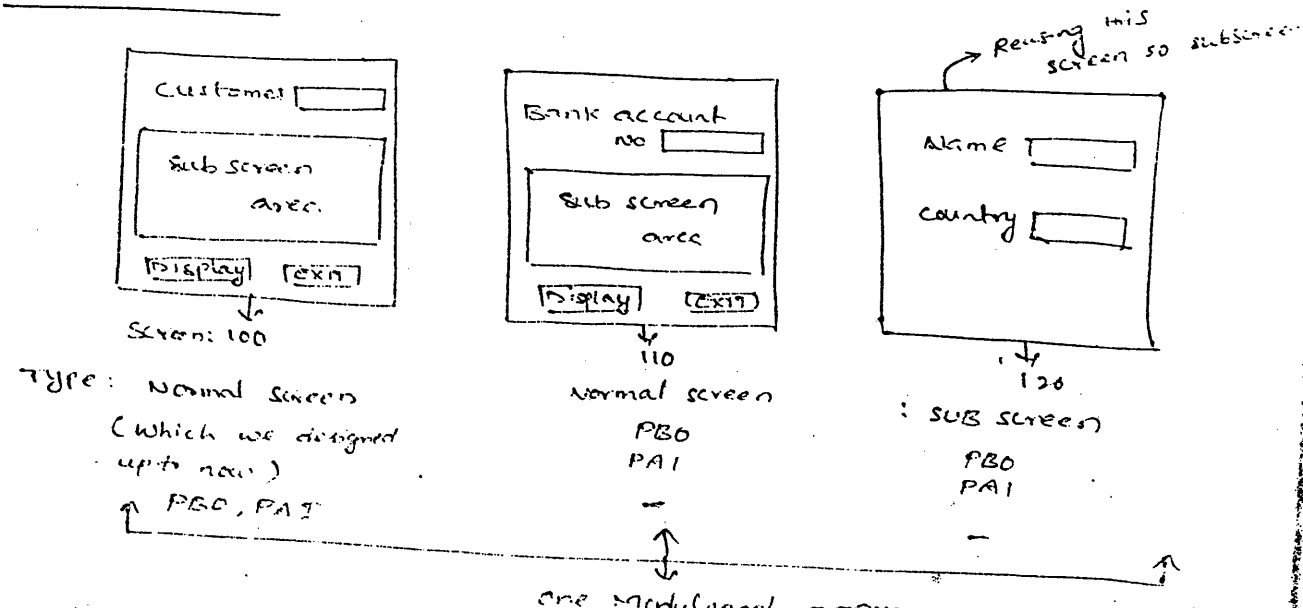
Loop at ITAB with control VCcontrol
cursor VCcontrol-current_line. } which transfers Multiple records to table control (PBO)
Module status -'100' (PBO statement)
Endloop.

ctrl+S

ctrl+F3

Design a Transaction code for this application by using SE93 and access with that application.

SUB-SCREENS:



* SUB-screens for Reusability. i.e which screen we going to dis In normal screen it is called sub screen.

* Where subscreen display in normal screen it is called as subscreen area.

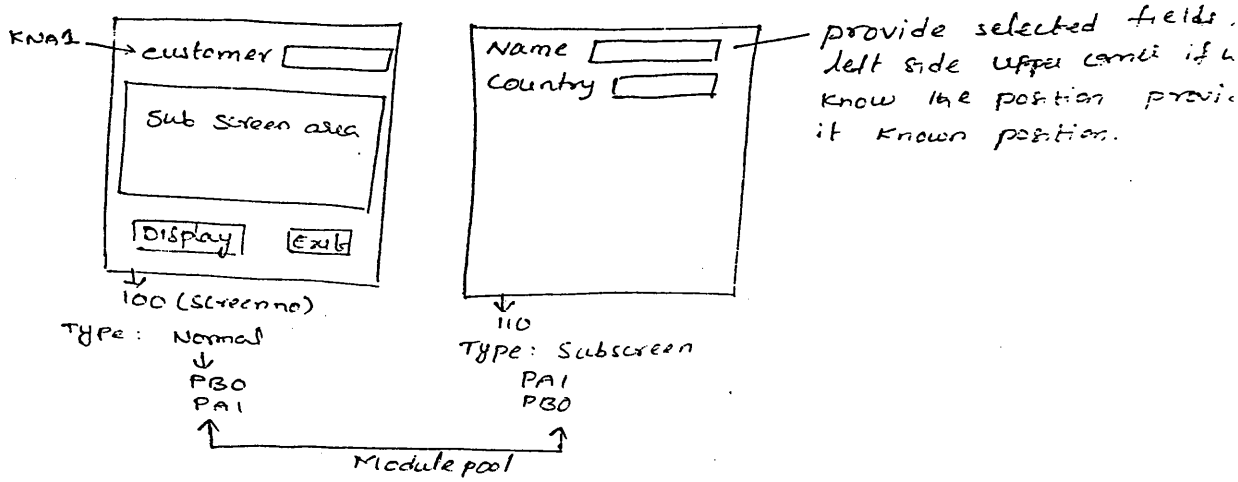
* In case of sub screen the command field is not allowed use system field.

* sub-screens doesn't hold input values.

* Minimum one normal screen is required for design applic we are not able to design application with all subscreens.

22/7/05

Demo on SUB-SCREENS :-



Let us create Modulepool Program YJagsub

go with screen painter SE51

provide Name : YJagsub

screen no : 100

go with layout-

provide the customer field in layout from KNA1 table select subscreen area button (from last to fifth one) Drop it in layout-

Double click on subscreen area and define properties

Name : SUB

↳ Subscreen name it should be any name

... copying to exit in layout and provide properties of that i.e name, description, FCT code

go with screen painter SE51

provide screen no : 110

go for create

provide the description for screen 110

select screen type

⊙ Sub screen.

go with layout

provide Name & country field in layout at left side upper most corner

go with screen painter

provide screen no : 100

select flow logic

go to the changemode.

Delete the comment for PAI program

Double click on that program

start declarations

TABLES : KNA1

Data : Begin of ITAB occurs 0,
Name1 like Kna1-Name1,
Land1 like Kna1-Land1,
End of itab.

go with Module & endmodule

Case sy-ucomm.
When 'DISP'.
Select Name1 Land1 from Kna1 into itab where Kunnr = Kna1-Kunnr.
Append itab.
Endselect.
When 'Exit'.
Leave program.
Endcase.

Ctrl+S
Ctrl+F3

F3.

provide following call statement in PBO statement

call subscreen SUB including 'YJagsub' '110'.
↳ subscreen name ↳ Modulepool program name

ctrl + =3.

go with screen painter

provide screen no : 110

select flow logic

go to the changemode

Delete the comment for PBO program

Double click on that program

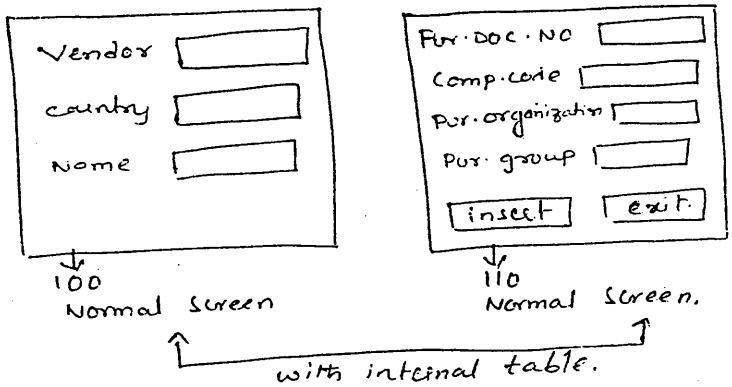
let us go b/w module & endmodule provide following

MOVE-corresponding ITAB to kna1.

ctrl+F3.

design T code for this application. and use it. (SE93).

Assignment:



Flow logic : PBO + PAI + PBO + PAI

we can provide Next screen : 110 in screen 100 attributes for screen connection.

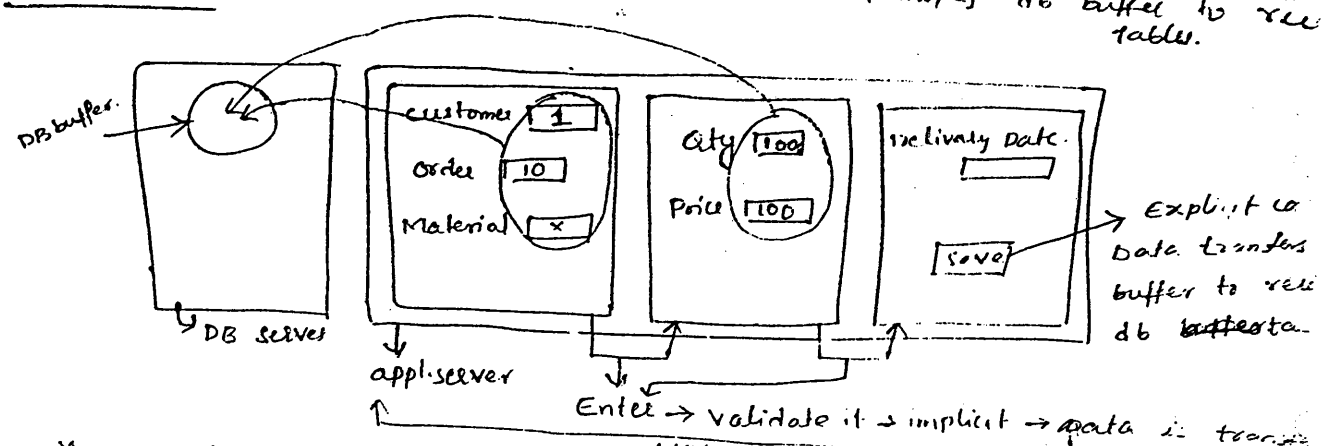
provide Tables workarea and internal table and insert, exit statements.

Tables : LFA1, EKKO

The above statements should be in PAI Program

Validations:

with explicit comm it data transfers db buffer to scr table.



* Enter is a keyword operation for appearing next screen.
* Between two screens Implicit exists and last screen explicit exists.
* By default SAP R/3 is screen level validator.

Implementation methods of fieldlevel validations:

1. Using chain ... Endchain field level validations can be implemented in SAP R/3.

2. Enhancement concept :- Under this Fieldexits

* With Fieldexists user can implement field level validation. SAP doesn't supports field exists. from 4.6C onwards.
* BTs Enhancement concept can be replaced in future but now BADI's are alternative to enhancement

BADI's → Business ADD IN's

Transaction code is SE18, SE19

Field level validation can be used for City fields in Realtime.

* Whenever application terminated in b/w data will rollback from db buffer automatically.

The above validation logic is called LUW

LUW - Logical Unit of work

Implicit come out called it as db LUW.

Explicit come out called it as SAP LUW.

SAP R/3 Implement by default db LUW.

23/7/05

LUW disadvantage: Network traffic is high b/w application server and database server. i.e if we have 15 screens in a application we have to connect 15 times db buffer so traffic is high.

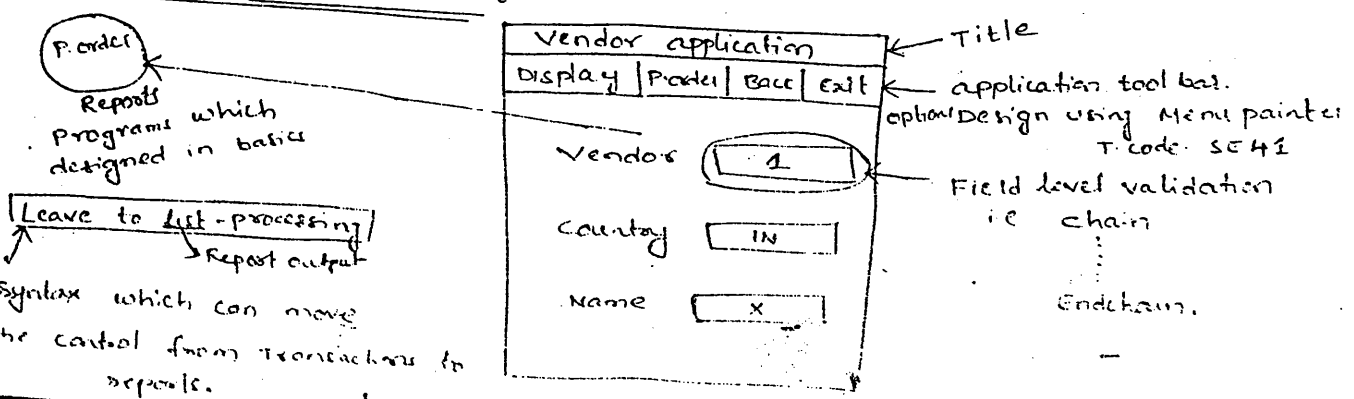
To overcome LUW disadvantage:

* Update Bundling ~~Bundling~~ can maintain buffer in a application server so every screen initially store it in appl buffer at last once store it in db buffer so traffic malls low.

* By default SAP follows LUW concept whereas update bundling we have to maintain explicitly in SAP.

* Why SAP by default follows LUW concept is it provides security because only one buffer that is located in db but in case of Update Bundling have two buffers one in appl server and one in db so it not provide that much security.

DEMO ON VALIDATION:-



by processing P.order control will move to respective Vendor

Let us create Modulepool program

go with screen painter SE51

screen no : 100

go with layout

provide the fields in layout

go with flow logic

Delete comment for FA1 program

Double click on that program

Start declarations

Tables : LFA1 , EKKO.

Define internal table ITAB , JTAB.
ITAB → For Transaction
JTAB → for Report

Data : Begin of ITAB OCCURS 0,
LIFNR like LFA1-LIFNR,
LAND1 like LFA1-LAND1,
NAME1 like LFA1-NAME1,
End of ITAB.

Data : Begin of JTAB OCCURS 0,
EBELN like EKKO-EBELN,
AEDAT like EKKO-AEDAT,
END of JTAB.

go b/w Module & endmodule

Write the logic for display purchase order for exit

```
case SY-UCOMM.
when 'DISPLAY'
Select LIFNR LAND1 NAME1 from LFA1 into Itab
                    where LIFNR = LFA1-LIFNR.
Append Itab.
Endselect.
when 'PORDER'.
Select EBELN AEDAT from EKKO into JTAB
                    where LIFNR = LFA1-LIFNR.
Append Itab.
Endselect.
```

Loop at JTAB.

Write : / JTAB.

Endloop.

Move control transactions to Reports by providing

Leave to list-processing.

When 'exit'.

Leave program.

Endcase.

Back will work automatically No need to provide any statements.

Ctrl+F3

F3.

Delete the comment for PBO program

Double click on that program

Go with Module & endmodule

Move-corresponding ITAB to LFA1.

Delete the comment for SET PF-STATUS 'xxx' in Modulepool of PBO replace that by provide following statement

SET PF-STATUS 'YJagan'.

→ status name.

Ctrl+S

go with SE41

provide Modulepool program name : YJaganmenu

provide status name : 'YJagan'.

go for create

provide short text : Demo

select application tool bar

provide the first option DISPLAY

Double click on Display

provide function text (or) FCT code : FCT code which we defined for display → DISPLAY

Select function F2 → select anyone from that list provided by SAP.

↵

process for remaining three P.order, back,

ctrl+S

ctrl+F3

go back to the program.

Delete the comment for SET TITLE.BAR 'XXX'.
pro Replace it by ↳ Title name

SET TITLEBAR 'ZTITLE'.

Double click on that ↵

provide Title : Vender application

go with all title

Ctrl+F3

F3.

F3

Under
~~Before~~ PAI module provide chain & endchain (under PAI
For field level validation

CHAIN

Field LFA1-LIFNR.

Module Value. ↳ validation name

Endchain.

Double click on value ↵

go with Module & endmodule

IF LFA1-LIFNR <1000 OR LFA1-LIFNR > 2000.

Message E000(0) with 'Enter correct input'.

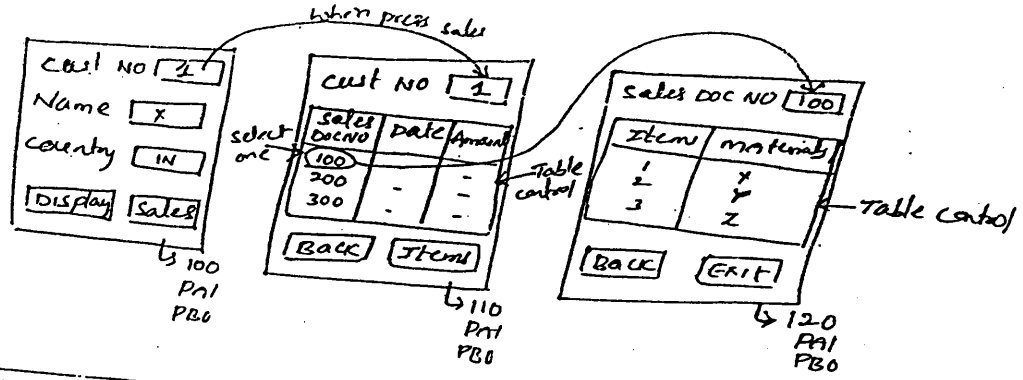
Endif.

Ctrl+F3

F3

Ctrl+F3

Design T. code for application and use it.



Get cursor field FNAM value FVAL.

Variables

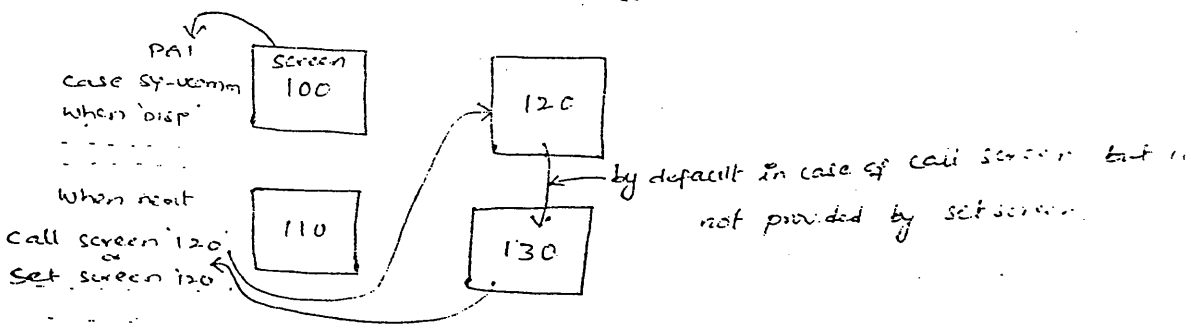
which is a syntax for system can identify the record i.e we can able to select one sales DOC NO from screen 110 it is transfered to screen 120.

FNAM : Field Name (sales DOC NO)

FVAL : value which u selected from that value (100)

call screen & set screen are the keywords which can move the control between multiple normal screens.

Ex:



* In case of call screen the control comes back by default whereas in set screen it doesn't comes back by default.

* In case of call screen maximum calling modes are '9'.

* By using set screen control moves to particular screen but it not comes back by default.

Let us create Module pool program YJMultiplescreens

go with screen painter SE51

screen no : 100

go with layout

Provide the fields in layout from KNA1

KUNNR, LAND1, NAME1

Provide options Display, sales and provide properties

go with screen painter SE51

screen no : 110

go with layout

Provide the customer field in layout

Provide the table control define properties

Provide the fields in a table control from VBAP

VBELN. ERDAT, NETWR

Provide options BACK, Items and provide properties of it

go with screen painter SE51

Screen no : 120

go with layout

provide Sales DOC field in layout

Provide the table ~~field~~ control and define properties

Select the fields for table control from VBAP

POSNR, MATNR.

Provide options BACK, Exit and define properties

go with Screen 100 flowlogic

Delete the comment for PAI program and Double click on that

* Provide Tables workarea

Tables : KNA1, VBAP. VBAK

* Define Int. tables Itab, Jtab, Ktab. for scree 100, 110, 120 PAI

Data : Begin of itab occurs 0,
KUNNR like Kna1-KUNNR,
LAND1 like Kna1-Land1,
Name1 like Kna1-Name1,
End of itab.

Begin of Itab occurs 0,
VBELN like VBAK-VBELN,
ERDAT like VBAK-ERDAT,
NETWR like VBAK-NETWR,
End of itab.

Data : Begin of Ktab occurs 0,
POSNR like VBAP-POSNR,
MATNR like VBAP-MATNR,
END of Ktab.

* Declare table control statements.

Controls : Vcontrol1 type Tableview using screen '110'.
(Table control name 110)

controls : Vcontrol2 type Tableview using screen '120'.
(120)

* Declare variables FNAM, FVAL for Get cursor statement-

Data : FNAM (10), FVAL(10) Type N.

* Go thru Module & endmodule of 100

Case sy-ucomm.
When 'DISP'.
Select- KUNNR LAND1 NAME1 from KNA1 into Itab where
KUNNR = KNA1-KUNNR.
Append Itab.
Endselect.
When 'sale'.
Refresh Itab.
Select VBELN ERDAT NETWR from VBAP into Itab where
KUNNR = KNA1-KUNNR.
Append Itab.
Endselect.
Call screen '110'.
Endcase.

* Maintain call screen statement after select statement

Ctrl+F3

F5

the comment for PBO program.
Double click on that
provide Move-corresponding
Move-corresponding Itab to KNA1.
ctrl+F3
go with screen 110 flow logic
Delete the comment for PAI program double click on that
go in module & endmodule
case sy-ucomm.
when 'BACK'
leave to screen '100'. (which can move control back)
when 'Item'.
Refresh KTAB.
Get cursor field FNAm value FVAL.
Select POSNR MATNR from VBAP into itab where VBELN = FVAL.
APPend KTAB.
Endselect.
call screen '120'.
Endcase.

ctrl+F3
F3

* provide Loop & endloop under PAI of 110
Loop at Itab.
Endloop.

Delete the comment for PBO program double click on that
Move-Corresponding Itab to VBAK.

F3

* provide move-corresponding b/w Loop & endloop under PBO of 110
Loop at Itab with control Vcontrol 1.
Modul status-110
Endloop.

go with screen 120 flow logic

Delete the comment for PAI program Double click on that-
go thw module & endmodule

```
case sy-ucomm.
When 'BACK'.
Leave to screen '110'.
When 'exit'.
Leave program.
Endcase
```

F3

* provide Loop & endloop und PAI of 120

```
Loop at Ktab.
Endloop.
```

Delete the comment for PBO and Double click on that-

```
MOVE-corresponding KTab to VBAP.
VBAP-VBELN = FVAL.
```

F3.

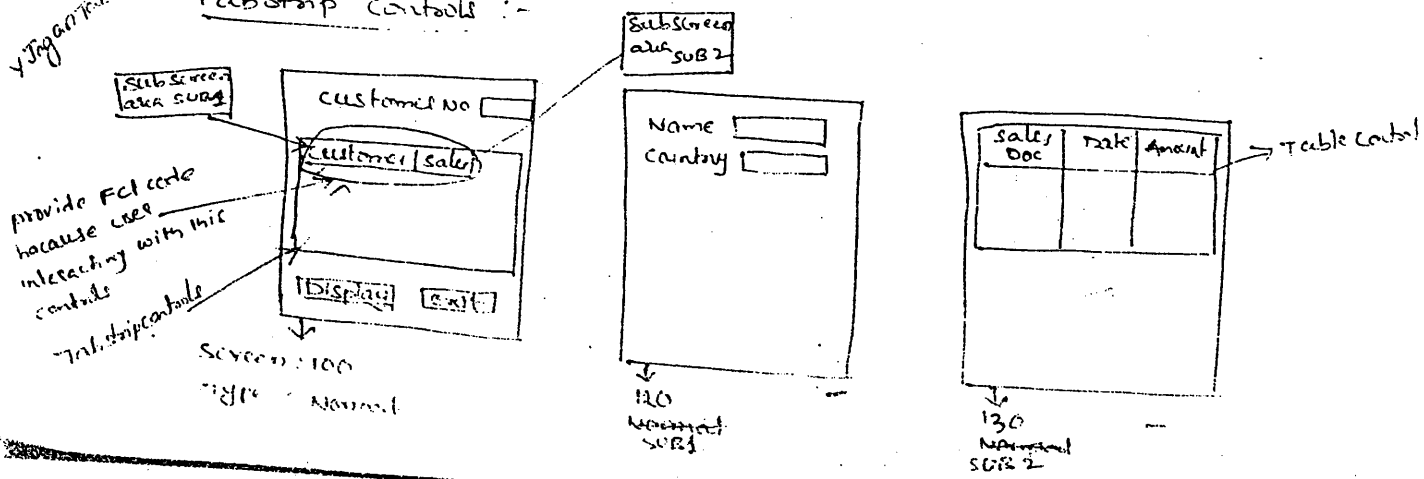
* Provide Move-corresponding in loop & endloop under PBO

```
Loop at Ktab with control Vcontrol2.
Module status_120.
Endloop.
```

Design a T.code and use application.

SG/7/05

Tabstrip controls :-



U ... & Sales properties define

FCT type : Local (P)

↳ Local.

DEMO:

Let us create Modulepool program

go with screen painter SE51

Screen No : 100

Screen Type : Normal

go with layout

provide customer field in layout from KNA1

Select the Tabstrip control (After push button) Drop it in layout

Double click on Tabstrip control

Define properties

Name : Strip

↳ go with anyname.

Double click Tab1

provide Name : customer

Fctcode : cust

Fcttype : P (Local).

Select Subscreen area Drop it in customer tab1

Define properties

Name : SUB1

Double click on Tab2

Provide Name : Sales

Fctcode : sale

Fcttype : P (Local)

Select subscreen area drop it in sales control

Define properties

Name : SUB2

Provide the options Display & Exit

go with screen painter

Screenno : 110

Type : SUBSCREEN

go with layout.

go with screen painter

screen no : 120

Type : subscreen

go with layout-

provide the Table control and define properties

provide the fields in a Table control

go with screen 100 flowlogic

Delete the comment for PAI program double click on that

Tables : KNA1, VBAK.

Data : Begin of itab occurs 0,
Land1 like Kna1-Land1,
Name1 like Kna1-Name1,
End of itab.

Data : Begin of Jtab occurs 0,
VBELN like VBAK-VBELN,
ERDAT like VBAK-ERDAT,
NETWR like VBAK-NETWR,
End of Jtab.

* provide Table controls declaration

Controls : Vcontrol Type Tableview using screen '120'.

* provide Tablestrip declaration

Controls : strip Type Tablestrip.
↳ Tablestrip name.

go with module endmodule

Case SY-UCOMM.
When 'DISP'
Refresh ITAB
Select Land1 Name1 from KNA1 into itab where Kunnr = Kna1-Kunnr.
Append itab.
Endselect.

Refresh Itab.

Select- VBELN ERDAT NETWR from VBAK into Itab
where Kunnr = kna1-Kunnr.

Append Itab.
Endselect.
When 'exit'.
Leave program.
Endcase.

Ctrl+F3
F3

* call subscreen logic twice under PBO of 100

call subscreen SUB1 Including 'YJagantablestrip' '110'.
call subscreen SUB2 Including 'YJagantablestrip' '120'.

Ctrl+F3

go with screen 110 flow logic

Delete the comment for PBO Double click on that-

Move-corresponding Itab to kna1.

go with screen 120 flow logic.

provide Loop & endloop under PAI

Loop at itab
Endloop.

Delete the comment for PBO Double click on that-

Move-corresponding Itab to VBAK.

F3

provide Move-corresponding b/w Loop & endloop under PBO

Loop at Itab with control vcontrol.
Modul ...
Endloop.

Ctrl+F3

Design a T-code and use application.

provide PAI module b/w loop & endloop because for performing multiple records.

Loop with control vcontrol
Module .....
Endloop.

Let us maintain one module program explicitly under PAI for exit

MODULE USER_exit → any name

If exit provides under loop every time it comes all so for create one records explicitly

Double click on that (user_exit) write logic for exit

Case sy-ucomm.
when 'exit'.
Leave program.
Endcase.

ctrl+F3
F3

Provide loop & endloop for PBO

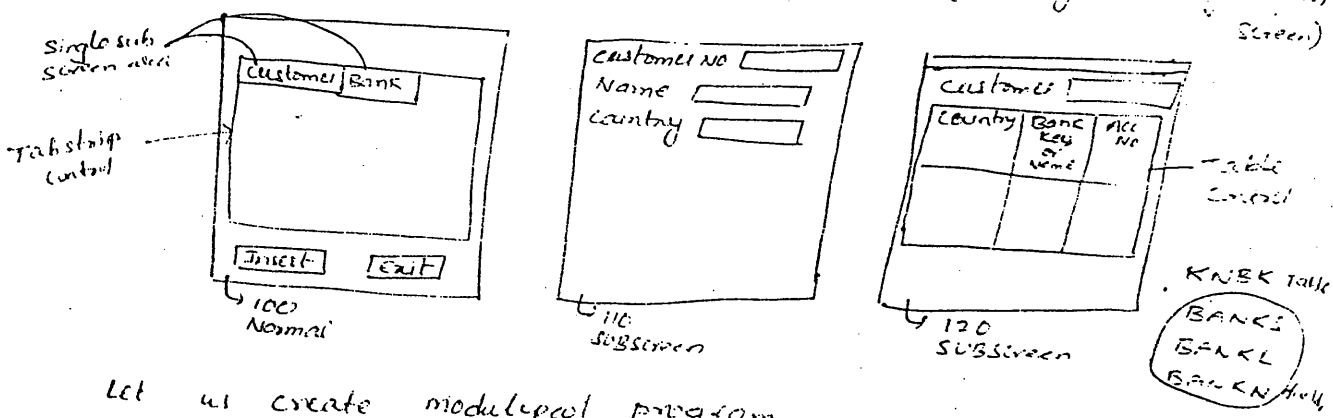
Loop with control vcontrol.
Endloop.

It is for process record while transfering screen fields to itab.

ctrl+F3

Design T. code and use application.

TABSTRIP CONTROL WITH DML OPERATION :- (working with single sub screen)



Let us create modulepool program

go with screen painter

Screen no : 100

go with layout

provide tabstrip control in layout

provide options Insert, Update, Delete, exit.

go with flow logic

Delete the comment for PAI program. Double click on that- start declarations.

```
Tables : LFA1
Data : Itab like LFA1 occurs 0 with Header line.
Controls : Vcontrol Type tableview Using screen 100.
```

go with module & endmodule

```
case SY-UCOMM.
when 'INSE'.
Move LFA1 to Itab.
Append Itab.
Insert Into LFA1 values Itab.
IF SY-SUBRC = 0
Message I000(0) with 'Inserted'.
Else.
Message E001(0) with 'Notinserted'.
End IF.
When 'UPDA'.
Move LFA1 to ITAB.
Append Itab.
Update LFA1 from Table Itab.
IF SY-SUBRC = 0.
Message I000(0) with 'Updated'.
Else.
message E001(0) with 'Not updated'.
EndIF.
When 'Dele'.
MOVE LFA1 to ITAB
Delete from LFA1 where LIFNR = ITAB-LIFNR.
IF SY-SUBRC = 0.
Message I000(0) with 'Deleted'.
Else.
Message E001(0) with 'Not Deleted'.
EndIF.
Endcase.
```

Ctrl+S

Ctrl+F3

F3

Ctrl + F3

F3

provide calling statement under PBO at 100

call subscreen SUB Including 'MPool' Dynnr.

go with before PAI module

call subscreen SUB.

↳ subscreen area name.

Define one module program explicitly under PAI

Module user_exit.

Double click on that user_exit.

case sy-ucomm.

when 'exit'.

Leave program.

Endcase.

Ctrl + F3

go with 110 flow logic

Ctrl + F3.

go with 120 flow logic

Delete the comment for PAI program

Double click on that

case sy-ucomm.

when 'INSE'.

Insert- KNA1, KNBK.

IF sy-subrc = 0

Message I000(0) with 'Inserted'.

else

Message E001(0) with 'Not inserted'.

EndIF.

Endcase.

Ctrl + F3

F3

Define properties

Double click on Tab1 and define properties

select subscreen area drop it in customer control and define properties

Double click on Tab2 and define properties

Name :

FCtcode :

Ref. field : SUB (subscreen area name)

provide options Insert, Exit.

go with screen painter

Screen : 110

Type : subscreen

go with layout and provide the fields

go with screen painter

Screen no : 120

Type : subscreen.

go with layout

provide Table control and define properties.

provide fields in a Table control from KNBK (BANKS, BANKL BANKN)

go with screen 100 flow logic

delete the comment for PAI program

Double click on that

Tables : KNA1, KNBK.

controls : Vcontrol Type Tableview Using screen '120'.

controls : Strip type Tabstrip.

Data : Dynnr Like sy-dynnr value '110'.

go b/w module & endmodule

case sy-ucomm.

when 'Tab1'.

Dynnr = 110.

strip-activetab = 'TAB1'.

when 'TAB2'.

Dynnr = 120.

strip-activetab = 'Tab2'.

endcase.

provide account group : sold-to-party

company code : 0001

↳ IDES customizing by 4 char length

sales organization : 0001

Distribution channel : 01

Division : 01 ↵

provide Title : Mr.

Name : Jagadeesh

search term 1/2 : J (It is for search help)

postal code : 507122

City : NKFURT

country : DE.

go with payment transaction control

go with comp code data in appl tool bar

Reconcellation account : 12000

* Reconcellation account is nothing but G/L acc maintain parallel with comp. account.

go with sales area data in appl tool bar

go with customer pricing procedure : 1 (one time customer no discount)

↳ standard pricing procedure

go with shipping control

shipping condition : 01 (Make delivery as early as possible)

go with Billing document control option

Tax : 0

click s.

customer was created

provide PAI b/w Loop & endloop

Loop with control vcontrol.
Module ....
Endloop.

Provide PBO b/w loop & endloop

Loop with control vcontrol
Module.
Endloop

Ctrl+F3

Double click on PBO program (120)

KNBK-KUNNR = KNA1-KUNNR.

Design a T.code.

28/7/05

ABAP Transactions with predefined applications:-

1. Vendor master application → T.codes: XK01 / XK02 / XK03
   - XK01 ↓ For create new Vendor
   - XK02 ↓ changing existing vendr
   - XK03 ↓ Displaying existing vendr
2. purchase order application → ME21 / ME22 / ME23
3. purchase Requisition application → ME51 / ME52 / ME53.
4. customer master application → XD01 / XD02 / XD03
5. sales order application → VA01 / VA02 / VA03
6. Delivery application → VL01 / VL02 / VL03
7. Material master application → MM01 / MM02 / MM03

Customer master application : XD01

Go with a T code SPRO for customizing which can be interacted by Functional people which can show How integrate FICO/ and SD applications

Go with SAP Reference img
- Enterprise structure → Definition
- → implementation guide

select all check boxes ↵

Ctrl+S

4. create a T. code for copyed appl

5. go with screen painter

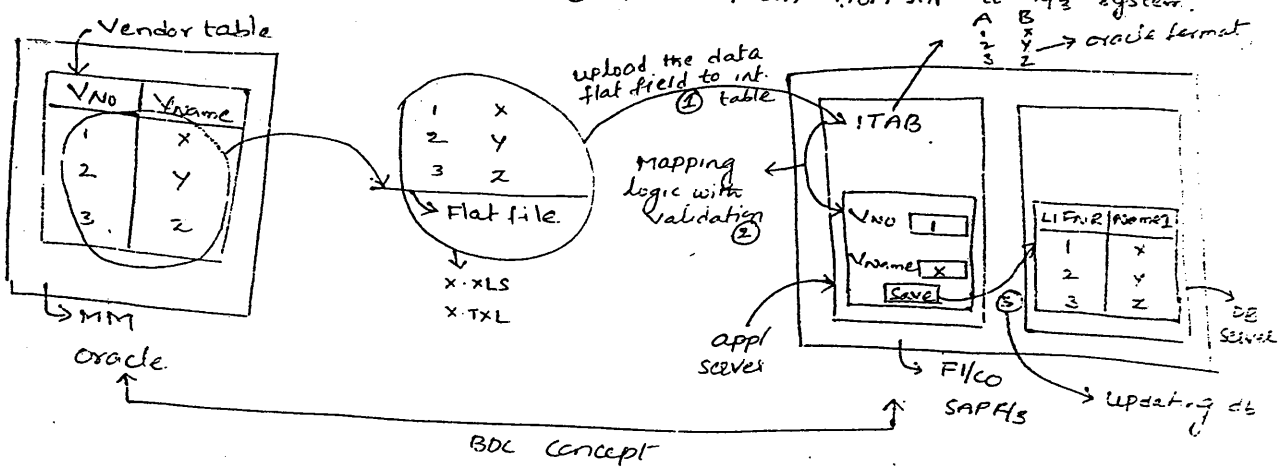
program : ZJagansapmf02k

Add a field to respective screen no.

29/7/05

BDC - Batch data conversion

With BDC user can transfer the data from non SAP to R/3 system.



BDC steps

1. upload the data from flat file to internal table.
2. Implementing validations on the data upload to the int table. using mapping.
3. Updating the db server.

Function modules (for upto V4.6c)

1. WS_upload
   ↓ WS- work station.
2. Upload.

These are the SAP provided function modules which upload the data from flat file to int. table

From V4.7 onwards WS_upload becomes Gui_upload

go with XK01 appl (Vendor master application)

Navigation for to check modulepool of an appl

go with System
→ status.

It will show a window in that

Program : SAPMF02K
→ Module pool of an appl

Double click on that program

Control leads to that module pool directly

in that have include programs

Include .... O0O (screen 100 PBO include)

Include .... I0O (screen 100 PAI include)

Include....TOP : Declaration of this modulepool

Double click on TOP include

Double click on first include

Whenever user wants to add a fields concepts are Modification and Enhancements.

Whenever appl is modified that modified appl doesn't support in upgradation.

When appl is enhanced that enhanced appl even supports in upgradation also.

Steps for modifications :-

1. Create a Structure with a field 'Vendor business address'
2. Append a Structure to the LFA1 table
3. copy XK01 appl

go with SE38

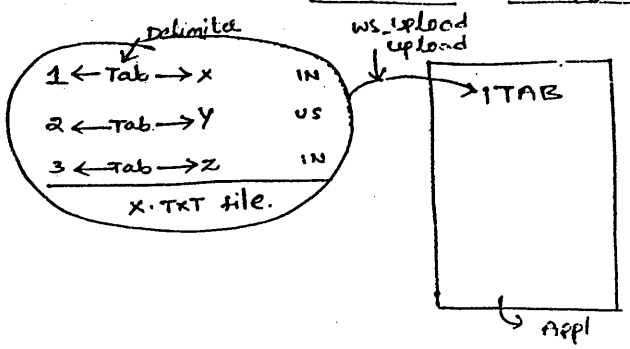
SAPMF02K (Respective modulepool program)

go with copy option in test bar

provide Target : ZJagan_SAPMF02K ↵

2/8/05

DEMO: Data Conversion from Non SAP R/3 to SAP R/3. (Uploading)



Let us create executable program YJagandataupload.

Start declaration

* Provide Tables workarea

```
Tables : KNA1
```

* Define internal table with 3 fields.

```
Data : Begin of itab occurs 0,
       KUNNR like Kna1 - Kunnr,
       Name1 like Kna1 - Name1,
       Land1 like Kna1 - Land1,
       End of itab.
```

go with pattern in toolbar and under pattern provide

call function : ws_upload. ↵ (or) upload ↵

Let us go for these parameter Remove comments for following.

Provide which file from data is uploaded

```
FILENAME = 'C:\Jagan.TXT'
FILETYPE = 'DAT' (or) 'ASC'
```

↳ Later we can use this

provide into which internal table the data has to be uploaded

```
DATA_TAB = ITAB.
Loop at ITAB.
Write : / ITAB.
Endloop.
```

ctrl+S
ctrl+F3.

With mapping data can transfer from internal table the application.

```
Loop
LFA1-LIFNR   ITAB-A
LFA1-Name1   ITAB-B
Endloop.
```

Flat file is a source for BDC concept

* BDC programs as interface programs also source is called outbound system
Target is nothing but inbound system.

DB structures in BDC :-

1. BDC DATA : It provides a mapping logic
   - → Program (Module pool program of an appl)
   - → Dynpro
   - → Dynbegin
   - → FNAM (Field Values ~~LFA1~~ ITAB-A , ITAB-B.)
   - → FVAL (Application fields LFA1-LIFNR, LFA1-Name1)

DYNPRO :- Dynpro is nothing but a current screen no

Dynbegin :- always first screen of an appl

2. BDC MSGCOLL : Database structure for error handling.

Methods in BDC :-

1. Session Method (or) Batch input method.
2. call transaction method
3. Recording (it is a part of session and call transaction)
4. Direct i/p method.
5. LSMW (Legacy system migration workbench)

* Provide select statement so that data can be transferred to the it.

```
select KUNNR Name1 Land1 from KNA1 into Table itab.
```

go with pattern

call function : WS_Download. ↵ (or) Download.

Remove the comments for all following.

Provide into which flat file data has to be download.

FILENAME : 'C:\Jagan\D.TXT'

FILETYPE : 'DAT'.

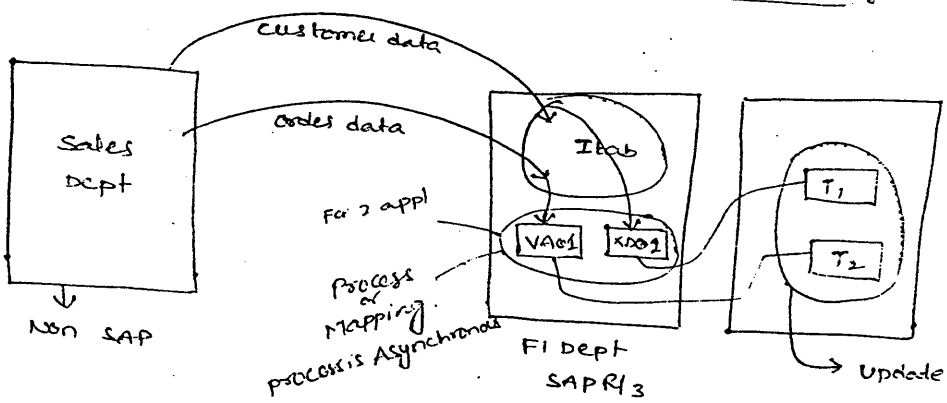
DATA_TAB : ITAB. from which int. table data has to be download.

Ctrl+S

Ctrl+F3.

go to FE and check Text file for downloaded data.

SESSION METHOD :- FEATURES OF SESSION METHOD :-



1. It is compatible for small amount of data as well as for large amount of data.
2. It is compatible for four ground as well as background executor.
3. It's have a log file concept by default.

   Log file is to store error records i.e which are not in SAP format.
4. It can work for multiple applications (RD01, VA01)
5. It process the data Asynchronously updates the database synchronously.

Design flat file Jagan.TXT file using notepad.

Delimiter is gap b/w fields (TAB)

KUNNR ← Tab →	Name ← Tab →	Country
75777	X	IN
75778	Y	US
75779	Z	IN

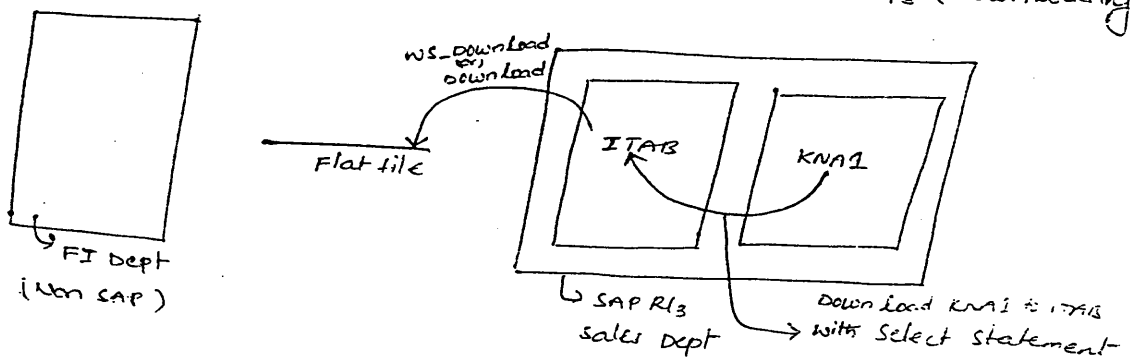
* Provide Delimiter is 'TAB' i.e gap b/w fields

Save the file as Jagan.TXT

go for F8 and close

* If we use call function is ws_upload user dialog is not requ i.e after pressing F8 it will upload. but in case of Upload user dialog is required.

DEMO : Conversion of Data from SAP R/3 to Non SAP R/3 (Downloading)



1. ws_Download
   or
2. Download

} Function modules provided by SAP for Downloading data from ITAB to flat file.

Let us create executable program YJagandownload.

Provide tables work area

Tables : KNA1

Data : Begin of itab occurs 0,
KUNNR like Kna1-Kunnr,
Name1 like Kna1-Name1,
Land1 like Kna1-Land1,
End of itab.

Let us conclude open_group with close_group.

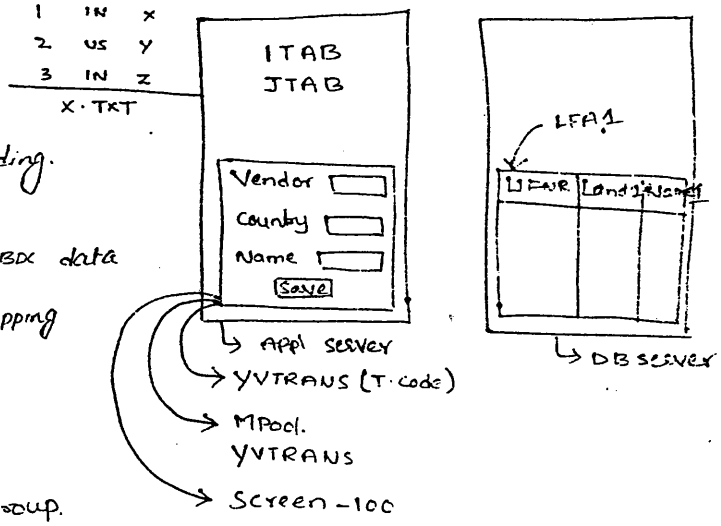
BDC_insert : It maintains mapping logic with BDC data structure.

parameters :

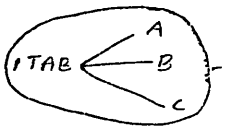
1. TCode : Tr.code of an appl which we are using for mapping

2. DYNPROTAB : 'JTAB'
   L> is an int. table design based on BDCDATA structure.

SESSION METHOD PROGRAMMING steps :-



- Define ITAB with the fields A,B,C



→ data uploading.

- Define an int. table based on BDC data

Itab based on BDCDATA → Mapping

call function : WS_upload

call function : BDC_Open_group.

provide pre requisities req for data transferring

Loop at itab

perform sub using 'YVTRANS' 100.

form

call function BDC_insert provide

T.code : YVTRANS

Dynprotab: Itab

perform sub1 using LFA1-LIFNR ITAB-A. (control move to field LIFNR

perform sub1 using LFA1-Land1 ITAB-B. (" " " Land1

perform sub1 using LFA1-Name1 ITAB-C. (" " " Name

End loop.

BDC_close_group.

Define subroutines SUB, SUB1.

Update than synchronous i.e Asynchronous it first updated the basic table then give a message c table updated automatically. but in case of synchronous all table and give a message so it is slow.

process of mapping is Asynchronous and updatation is Synchronous.

SYNTAX FOR SESSION Method :- Function modules used by sess exclusively :-

1. BDC_open_group.
2. BDC_Insert
3. BDC_close_group.

1. BDC_open_group : It maintains pre requisites required for data transfering.

parameters which come under BDC_open_group

1. CLIENT = production ID (Real data maintain by produc client
   = SY-MANDT system variable for client
2. USER = USER ID Under that client
   = SY-UNAME.
3. HOLDDATE = provide system variable for date
   Hold date maintains data on which data will transfer.
   = SY-Datum.
4. Keep = Keep is required for multiple applicatons.
   = 'X' default in abab language.
5. Group : Group provides an identity while transfering the data.
   = 'XYZ' (same identity)

Suppose we dealing with multiple applications for first appl we provide group and for second app provide keep.

Jtab : functionality is for mapping

* int. table for mapping

Data : Jtab like BDCDATA occurs 0 with Header line.

* Up loading data from Flat file into int. table

Call function : WS_Upload ↵

Delete comment for Exporting and following.

Filename = 'c:\Jagan.TXT'

Filetype = 'DAT'

Tables

Data_TAB = 'ITAB'.

* Basic information Required for Data transmission

Call function : BDC_Open_group ↵

Delete comment for following

Exporting

client = SY-MANDT

group = 'Jagan'

Holddate = SY-Datum

Keep = 'X' (default bec we are using single appl)

User = SY-UNAME.

* Read the data from itab

Loop at itab.

gc b/w 9-10 lines space and define subroutines. space for perform

Refresh Jtab.

* Mapping logic

* Call first subroutine (or) → YVTRANS (Vendor appl)

Perform sub using 'YJagannappl' '100'.

* call second subroutine 3 times because application have 3 fields.

Perform sub1 using 'LFA1-LIFNR' ITAB-LIFNR.

Perform sub1 using 'LFA1-Land1' ITAB-Land1.

Perform sub1 using 'LFA1-Name1' ITAB-Name1.

Call function : BDC_insert ↵

Delete comments for following

Define first subroutine.

```
Form sub using A B.
                  |  |
                  |  parameters
                  it can pass parameters.
clear Jtab.
Jtab-program = A
     Dynpro = B
     Dynbegin = 'X' - default in abap
Append Jtab.
Endform.
```

Define second subroutine

```
Form SUB1 using C D
clear Jtab.
Jtab - FNAM = C
Jtab - FVAL = D
Append Jtab.
EndForm.
```

1/8/05

DEMO : SESSION METHOD :-

Use Vendor application T code : YVTRANS

i.e For which application we are using this method

Use modulepool prog name of that appl : YVTRANS

screen : 100

Let us create executable program

* provide Tables workarea

Tables : LFA1.

* Define int. table with 3 fields (application dependent)
* Data Handling int. table

```
Data : Begin of itab occurs 0,
       LIFNR like LFA1-LIFNR,
       Land1 like LFA1-Land1,
       Name1 like LFA1-Name1,
       End of itab.
```

But group was locked so we cannot process the group we have unlock the group by select the group and

UNLOCK :

go for session option
↳ unlock

Deselect the entry and

go with process

select fourground option

go with process again.

It will display a vendor appl with flat file value then click Insert button and select exit button for second value so

CALL TRANSACTION METHOD :- FEATURES :-

1. It is compatible for small amount of data only.
2. It is compatible for foreground process only.
3. It process the data synchronously and update the d. synchronously.
4. It can handle only one appl at a time.
5. It doesn't have log file concept.
6. Under this method user has to design a log file explicitly using BDCMSGCOLL.

SYNTAX :-

Call Transaction 'YVTRANS' Using Itab
↳ T.code ↳ int. table for mapping using BDCDATA.

* Replace BDC_insert in session method by above syntax it becames call transaction method.

TCode = 'YJagonapp' (or) 'YVTRANS' (application Tr. code)

Tables

Dynprotab = Jtab.

Endloop.

Call function : BDC_close_group ↵

Delete all parameters under close_group.

* Definile first subroutine which moves control to the screen.

```
Form SUB using A B.
Clear Jtab.
Jtab-program = A.
Jtab-Dynpro  = B.
Jtab-Dynbegin = 'X'. (Default)
Append Jtab.
Endform.
```

* Define secand subroutine which moves the control to the field level.

```
Form SUB1 using C D.
clear Jtab.
Jtab-FNAM = C.
Jtab-FVAL = D.
Append Jtab.
EndForm.
```

Ctrl+S

Ctrl+F3

Design flat file Jagan.TXT by using notepad.

Vendano	Country	Name
70190	IN	Jagan
70191	US	XYZ
70192	IN	ABC

Execute logic (F8) then

go with SM35 T. code for Batch input session.

Provide session : Jagan (group name what u provided in program).

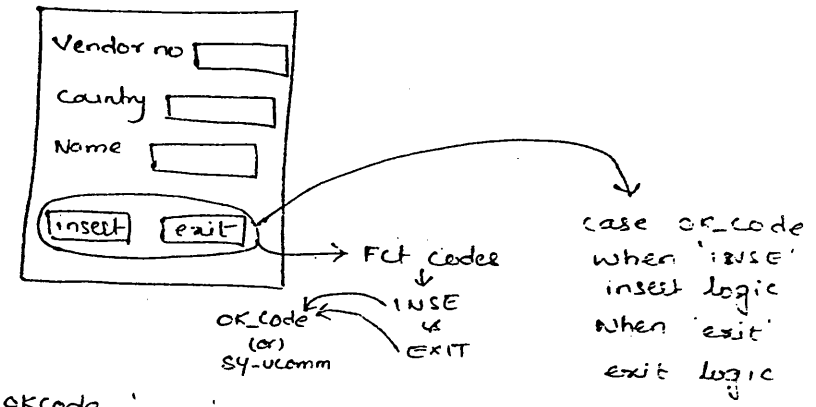
Display is mandatory parameter.

Complete syntax for call tr. method :-

Call transaction 'YVTRANS' using Itab mode 'N'
↳ No-display mode.

5/8/05

YVTRANS application :-



Perform sub using BDC-OKcode 'INSE' it will provide insert operation internally

DEMO : CALL TRANSACTION METHOD :-

Let us create executable program

provide Tables workarea

Tables : LFA1.

Define Itab for uploading data

Data : Begin of itab occurs 0,
LIFNR like LFA1-LIFNR,
Land1 like LFA1-Land1,
Name1 like LFA1-Name1,
End of itab.

* Define Itab for mapping

Data : Itab like BDCDATA occurs 0 with Header line.

Go for pattern:

call function : WS_upload

Delete the 'comment for following

Exporting

Filename : C:\Jagan CT.TXT

Filetype : 'DAT'

Table

Data_tab = Itab.

PARAMETERS IN CALL TR. METHOD :-

These parameters are for exclusively for call Tr. method

1. Display
   - Append Mode 'A'
   - NO-display mode 'N'
   - Error Mode 'E'

2. Update
   - Synchronous – 'S'
   - Asynchronous – 'A'
   - Local – L (out dated)

3. Messages : For designing log file explicitly using BDCMSGCOLL.

* With APPEND mode user dialog is required for every screen contains in application. Ex: For go to next screen perform entry.
* With NO-DISPLAY user can avoid the dialog with appl. it will effect on immediately. i.e internally system will perform entry act
* It is not a background process because memory of fore ground and background are different.
* If it is ERROR mode user dialog is required whenever error screen comes.
* In case of error mode messages parameter is mandatory.
* Where as in case of Append & no-display messages is optional.

UPDATE PARAMETERS :- Call transaction is fastest than session method because of in this method user can change the update mode synchronously to Asynchronously this possibility not these in session method.

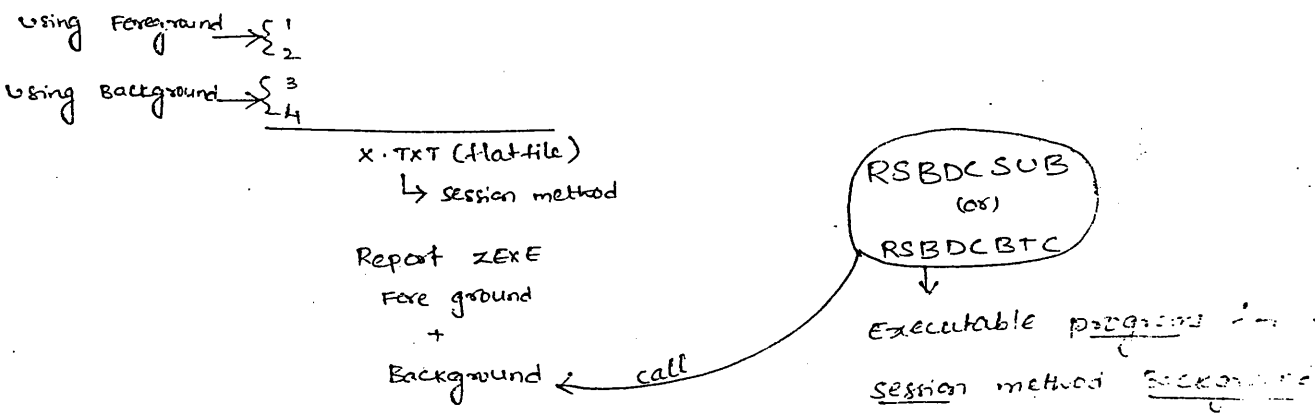
Design flat file C:\Jagan CT.TXT

22221	IN	Jagan
22222	US	XYZ
22223	DE	YZ

go for execute F8. It will inserted directly no need of user dialog. check the table for inserted fields.

If we take mode 'A' in call tr. syntax then user dialog is required for every insertion.

DEMO :- HALF RECORDS FROM FOREGROUD and HALF RECORDS FROM BACKGROUND By using SESSION METHOD :-



SUBMIT is a Keyword for calling an executable with in executable

With UNLOCKING system is authorized for processing the work with respective group name. But in case of Background process will not provide directly unlocking concept bydefault. For that we can avoid HOLDDATE parameter in BDC_open_group then system will provide unlocking by default.

Loop at itab.

Refresh Jtab.

* Call subroutines SUB1 for screen level & SUB2 for field leve

Perform SUB1 Using 'YVTRANS' 100.

perform SUB2 Using 'LFA1-LIFNR' ITAB-LIFNR.

perform SUB2 Using 'LFA1-Land1' ITAB-Land1.

perform SUB2 Using 'LFA1-Name1' ITAB-Name1.

* Call SUB2 for performing insert operation

Perform SUB2 Using 'BDC_OKcode' 'INSE'.

* Instead of BDC_insert provide call tr. logic

Call transaction 'YVTRANS' Using Jtab Mode 'N' (or) 'A'

↓ ↓

No-Display mode. Append mode

Endloop.

* Define First subroutine SUB1 for screen level control

Form SUB1 Using A B.

Clear Jtab.

Jtab-Program = A.

Jtab-Dynpro = B.

Jtab-Dynbegin = 'X'.

Append Jtab.

EndForm.

* Define second subroutine SUB2 for Field level control

Form SUB2 Using C D.

clear Jtab.

Jtab-FNAm = C.

Jtab-FVAL = D.

Append Jtab.

EndForm.

ctrl+S

ctrl+F2

Demo :- Let us create executable program

* provide Tables work area

Tables : LFA1.

* Define Itab for uploading

```
Data: Begin of itab occurs 0,
      LIFNR like LFA1-LIFNR,
      Land1 like LFA1-Land1,
      Name1 like LFA1-Name1,
      End of itab.
```

* Define Itab for mapping.

Data : Jtab like BDCDATA occurs 0 with Header line.

* Define variables for to make half of records.

Data: X, Y, N Type I.

* Upload the data

call function : ws_upload. ↵

Delete comments for following parameters

Filename : C:\ : Jagan~~forground~~Half.TXT

Filetype : 'DAT'

Tables

Data_TAB = Itab.

Apply logic to read no.of records contains in ~~the~~ itab

```
N = 0
Loop at itab
  N = N+1
Endloop.
```

Apply logic to make half of records.

X = N / 2.

Y = X + 1.

* Apply foreground process

call function : BDC_open_group.

Delete comments for following

Exporting

client = SY-MANDT.

Group = 'Jaganforeground'

Holddate = SY-datum

Keep = 'X'

User = SY-uname.

Let us read first half of records

Loop at Itab from 1 to X.

Refresh Jtab.

Perform sub1 using 'YVTRANS' 100.

Perform sub2 using 'LFA1-LIFNR' ITAB-LIFNR.

Perform sub2 using 'LFA1-Land1' ITAB-Land1.

Perform sub2 using 'LFA1-Name1' ITAB-Name1.

call function : BDC_insert ↵

Delete comments for following

T.code = 'YVTRANS'

DynproTAB = Jtab.

Endloop.

call function : BDC_close_group.

Delete all parameters.

* Apply the logic for Background process.

call function : BDC_open_group ↵

Exporting

client : SY-MANDT

Group = 'Jagan Background'

Keep : 'X'

User = SY-Uname.

Here Delete Holddate for to unlock the group by default.

Vendor table with three records

663340	IN	Jagan
663341	US	x
663342	IN	y
663343	US	z

go for execute (F8)

First background process executes because group wa unlocked by default then fore ground execute.

FOR Background after pressing F8 it will display a win.

Provide session : JaganBackground (Groupname for Backgrou

again press F8. Records are inserted.

Then go with SM35 for fore ground execution

select group and unlock it and deselect and

go will process

select Foreground again process.

user dialog is required for inserting remaining records.

6/8/05

RECORDING :- Transaction code is : SHDB.

It is a part of session method and call transaction method

* We can find the field and Table names by manually using following steps.

Open any application ex: XD01 (Vendor appl)

place a cursor in Field which we need

and press F1 and go with Technical Information option in tool bar - it will display details but in case of large amount of data we can't use it.

Loop at ... y ... IN.

Refresh Itab.

Perform Sub1 using 'YVTRANS' 100.

Perform Sub2 using 'LFA1-LIFNR' ITAB-LIFNR.

Perform Sub2 using 'LFA1-Land1' ITAB-Land1.

Perform Sub2 using 'LFA1-Name1' ITAB-Name1.

* Call subroutines for Background process

Perform Sub2 using 'BDC_OKcode' 'INSE'. (For inserting)

Perform Sub1 using 'YVTRANS' 100. (For display screen 100)

Perform Sub2 using 'BDC_OKcode' 'Exit'. (for exit)

Call function : BDC_insert ↵
exporting
T. code : 'YVTRANS'

DynproTAB : Itab.

Endloop.

Call function : BDC_close_group.

* apply SUBMIT keyword to call background executable program RSBDCSUB into executable

SUBMIT RSBDCSUB VIA selection-screen.
(or) RSBDCBTC

Define subroutines: Form Sub1 using A B.

Clear Itab.

Itab-program = A.

Itab-Dynpro = B.

Itab-Dynbegin = 'X'.

Append Itab.

Endform.

Form Sub2 using C D.

Clear Itab.

Itab-FNAM = C.

Itab-FVAL = D.

Append Itab.

Endform.

* NOTE:

If we use program RSBDCBTC then it will ask QUEUE-ID so we execute the program then go to SM35 and there select our foreground for that group have last colum Queue ID then copy that ID and provide in selection screen.

ctrl+1 S
ctrl+F3

provide declarations and uploading and loop, endloop

i.e

* provide Tables work area

Tables : LFA1.

Data : Begin of itab occurs 0,
LIFNR like LFA1-LIFNR,
LAND1 like LFA1-LAND1,
NAME1 like LFA1-NAME1,
END OF ITAB.

After start of selection statement provide

call function : WS_Upload ↵

Delete comments for following

Exporting

Filename : C:\JaganRec.TXT

FileType : 'DAT'

Tables

Data_TAB : Itab.

After perform opengroup

Loop at itab.

Refresh BDCDATA.

Delete the dummy value provide itab fields.

Itab-LIFNR.

Itab-Land1.

Itab-name1.

After perform bdc-transaction statements

Endloop.

Design flat file

Ctrl+F3

F8

at runtime we can select session (or) call transaction methe

Select call transaction

Runmode : N (NO-Display)

Update : A (ASynchronous)

F8

Log file is will display code = 1 (Error record)

* With RECORDING user can search fields & Tables requir for mapping. we can know all fields at a time.

DEMO: FOR SEARCHING FIELDS & TABLES AND RECORDING

go with SHDB T-code

go for create recording option in toolbar

provide recording name : YJaganRec

provide on which appl we have to be Recording : YVTRANS
↳ T-code of one,

select ® start Recording

provide the dummy data in this application.

perform insert and exit.

ctrl+s

go for F3

It will display information about Tables, fields which we used in recording then

select Recording name

go for create program in toolbar

provide program name : YJaganRecording

select option : Transfer from recording. ↵

provide attributes to the program

go with sourcecode it will display a predefined program for appl.

ctrl+s.

Double click on include program : BDCRECX1 which provides by default 1. Subroutines → BDC_DYNPRO (TO move control to screen level)
BDC_Field (TO " " . Field ")

2. Internal tables → BDCDATA (For mapping)
MESSTAB (For log file)

3. provided session method logic with
BDC_open_group
BDC_Insert
BDC_close_group

4. call transaction logic with all parameters.

Vendor no : 11331

Comp code : 0001 (IDES customizing)

Pur. organization : 0001

Account group : 0001 ↵

provide the title : Mr.

Name : Jagadeesh

search term : J

City : Hyd

postal code : 50712

country : DE

Language : EN ↵

↵

provide in which country vendor having acc : DE

" " " Bank " " " : X

" " " Accountno " " " : 11

provide reconcellation account : 170000.

provide the cash management group : A1. ↵

↳ payment duration (60 days)

↵

↵

provide the currency : INR ↵

Ctrl + S

/00 is a predefined FCT code for ENTER.

/01 is for save

Imp structures { RF02K : is a db structure for vendor application
RF02D : is a " " " customer. "

Ctrl + S

F3

select start Recording

go for create program

provide program name -

Select session method

Session name : Jagangroup (groupname)

User : SAPUSER

Lockdate : 5.08.2005 (Holddate)

F8

go with SM35

select group name i.e Jagangroup unlock it (Tool bar unlock option)

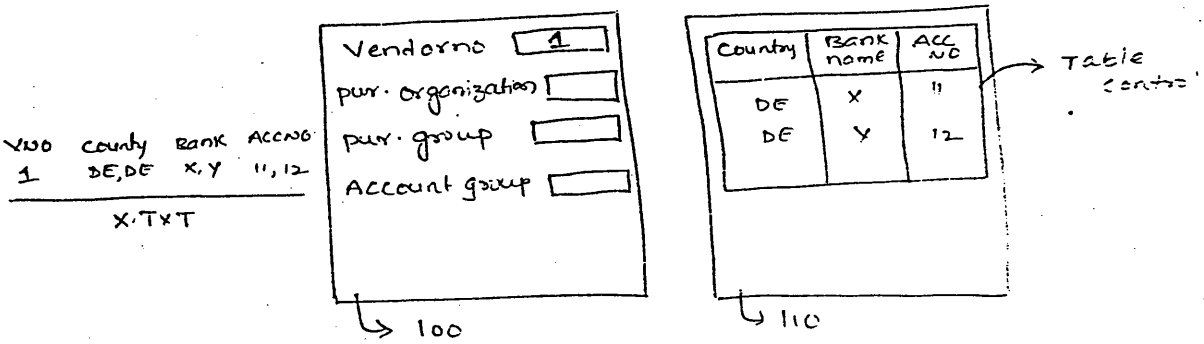
go with process

select foreground

process.

9/8/05

CALL TRASACTION : with XK01 Vendor master create



'SPLIT' is a keyword which can transfer the data through table controls.

i.e DE,DE → DE
DE

comma (,) will be split by using keyword 'SPLIT'

DEMO:

go with Trasaction code SHDB

go for create recording

provide the recording name

provide T.code for which we have to do recording.

: XK01

Start recording

provide the dummy data in application.

Data : Begin of BANKL OCCURS 0,
BANKL like LFBK-BANKL,
END OF BANKL.

Data : Begin of BANKN OCCURS 0,
BANKN like LFBK-BANKN.
END OF BANKN.

Data : FLD (20) Type C.

Data. : CNT (2) Type C.

After start of selection.

Using pattern Ctrl+F6 Call function : W.S_Upload

Exporting

Filename : 'C:\Jagan.TXT'

Filetype : 'DAT'

Int table : ITAB.

1 DE,DE X,Y 11,12

Jagan.TXT

ITAB

↓

1 - LIFNR

DE,DE - BANKS

X,Y - BANKL

11,12 - BANKN

Int. tables of BANKS L, N

↓

BANKS
↳ DE
DE

BANKL
↳ X
Y

BANKN
↳ 11
12

splited b
SPLIT KE
WAS

After patern open_group

LOOP at itab.

Refresh BDCDATA

Refresh : BANKS, BANKL, BANKN.

SPLIT ITAB-BANKS at ',' INTO Table BANKS.

SPLIT ITAB-BANKL at ',' INTO Table BANKL

SPLIT ITAB-BANKN at ',' INTO Table BANKN.

Delete the dummy values what we provided in the recording and provide ITAB fields.

ITAB-LIFNR

" - BUKRS

" - EKORG

" - KTOKK

" - SPRAS

" - ANRED

" - SORTL

Select option ⊙ Select Transfer from recording. ↵

provide attributes to the program

Ctrl+S

provide Tables workarea . . .

Tables : RFO2K , LFA1 , LFBK, LFB1 , LFM1.

Define Int. table for Data handling

Data : Begin of itab occurs 0,

LIFNR like RFO2K-LIFNR, } screen 100 fields

Comp. code ← BUKRS like RFO2K-BUKRC,

Pur. organization ← EKORG " " - EKORG,

Acc. group ← KTOKK " "

Title ← ANRED LFA1-ANRED, } screen 110 fields

← NAME1 " ,

Search term ← SORTL " ,

City ← ORT01 "

postal code ← PSTLZ "

Land 1 ← LAND1 "

Language ← SPRAS "

* Extend Internal table

B. County ← BANKS(50) TYPE C, } screen 130

B. Name ← BANKL(50) TYPE C,

B. Acc. No ← BANKN(50) TYPE C,

Reconciliation acc ← AKONT like LFB1-AKONT, } screen 150

Extra cash management group ← FDGRV like " - FDGRV,

Amount ← WAERS like LFM1-WAERS, --- screen 170

END OF ITAB.

Based on each field contains in the table control define an Int. table

Data: Begin of BANKS OCCURS 0,

BANKS LIKE LFBK-BANKS,

END OF BANKS.

OPEN DATASET CONCEPT :-

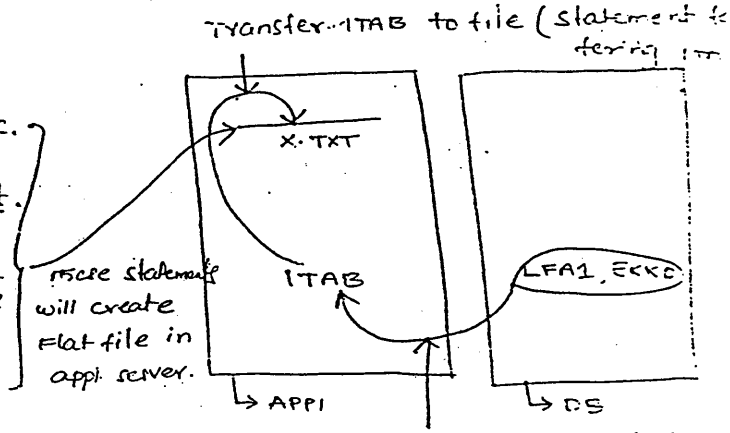
* Whenever Flat file comes under presentation server it comes Physical file handling technique this technique is not compatible Unix platform.

* Whenever Flat file comes under application server it is c to file handling technique. This technique is compatible for all platf Logical file handling even we call it as open dataset (or) sequ dataset.

* Flat files under presentation server will not provide security Compared to application server Flat files.

Ex:

```
Report ZEXE
Parameters : file(200) type c.
Open dataset file for output.
:
:
Create a flat file
:
close dataset file.
```



DEMO: DATA RETRIEVAL FROM DB INTO APPL SERVER PLATFILE BY USING OPEN Data set (Data Downlo

Let us create executable program

Provide Tables workarea

Tables : LFA1, EKKO.

Design selection screen with parameters.

Parameters : File(200) Type C.

Define an Int. table :

provide the comments for perform statements related to the t

Control field. i.e

BANKS, BANKL,

* perform BDC_

provide following statements for table control fields.

MOVE 1 TO CNT.

↳ control

LOOP at BANKS. (or) BANKL (or) BANKN

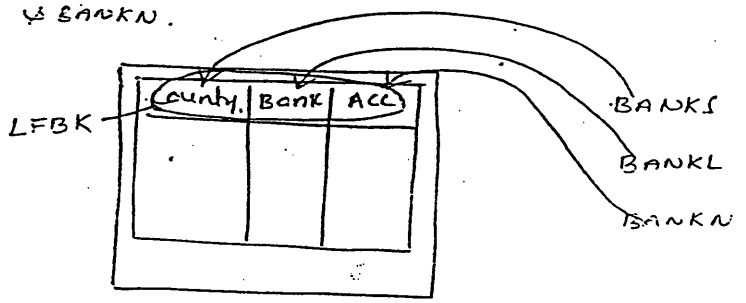
CONCATENATE 'LFBK-BANKS(' CNT ')' INTO FLD.

PERFORM BDC_FIELD USING FLD BANKS-BANKS.

CNT = CNT + 1

ENDLOOP.

Repeat same move logic for BANKL & BANKN replace BA... by BANKL & BANKN.



delete the dummy values and provide ITAB fields.

Endloop.

Ctol + F3

Design flat file (sequence is according to Recor... fields)

F8

Select call tr. method

F8 we can the appl with notepad the

ASSINGMENT:

1. XD01 Application (customer Master create) by Recording.

UPLOADING FROM APPLICATION LAYER TO DB.

Let us create executable program.

Provide Tables, workarea.

Tables: LFA1, EKKO.

Parameters: File(200) Type C.

Design int. table for data uploading.

```
Data : Begin of ITAB occurs 0,
       LIFNR
       LAND1
       NAME1
       EBELN
       BUKRS
       EKORG
       EKGRP
```

Define int. table for Mapping

```
Data : Begin of Itab occurs 0.
       Include structure BDCDATA.
Data : End of Itab.
```

Syntax for defining int. table based on struc

Define int. table for log file.

```
Data : Begin of MESSTAB occurs 0.
       Include structure BDCMSGCOLL.
Data : End of Messtab.
```

Open Dataset file for Input.

```
DO.
Read Dataset file into ITAB.
IF SY-SUBRC <> 0.
EXIT.
ENDIF.
```

* Apply Mapping logic.

```
perform Sub using 'YVENDORMPOOL' 100.
  "     Sub1  "   'LFA1-LIFNR'  ITAB-LIFNR.
  "     Sub1  "   'LFA1-LAND1'  ITAB-LAND1.
  "     "     "   'LFA1-NAME1'  ITAB-NAME1.
perform Sub using 'YVENDORMPOOL' 110.
```

```
Data : Begin of ITAB occurs 0,
       LIFNR  like' LFA1-LIFNR,
       Land1  like' LFA1-Land1,
       Name1
       EBELN  like' EKKO-EBELN,
       BUKRS
       EKORG
       EKGRP
       END OF ITAB.
```

provide the select statement and transfer data to ITAB.

```
SELECT LFA1~LIFNR LFA1~LAND1 LFA1~NAME1 EKKO~EBELN
       EKKO~BUKRS EKKO~EKORG EKKO~EKGRP INTO table ITAB
       FROM LFA1 INNER JOIN EKKO ON LFA1~LIFNR = EKKO~LIFNR.
```

Open Dataset file for output.

* Read the data from itab.

```
Loop at itab from 1 to 3.  (It is for first 3 records)
Transfer ITAB to file.
Endloop.
close dataset file.
```

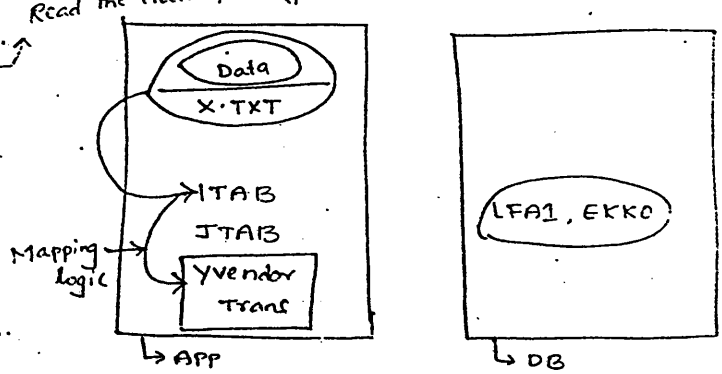
Ctrl+F3.

Provide the filename in selection screen. → C:\BDC\Jagan.TXT
↳ Appl. server path

F8.

Check the down loaded file in appl server

Read the data from flat file.



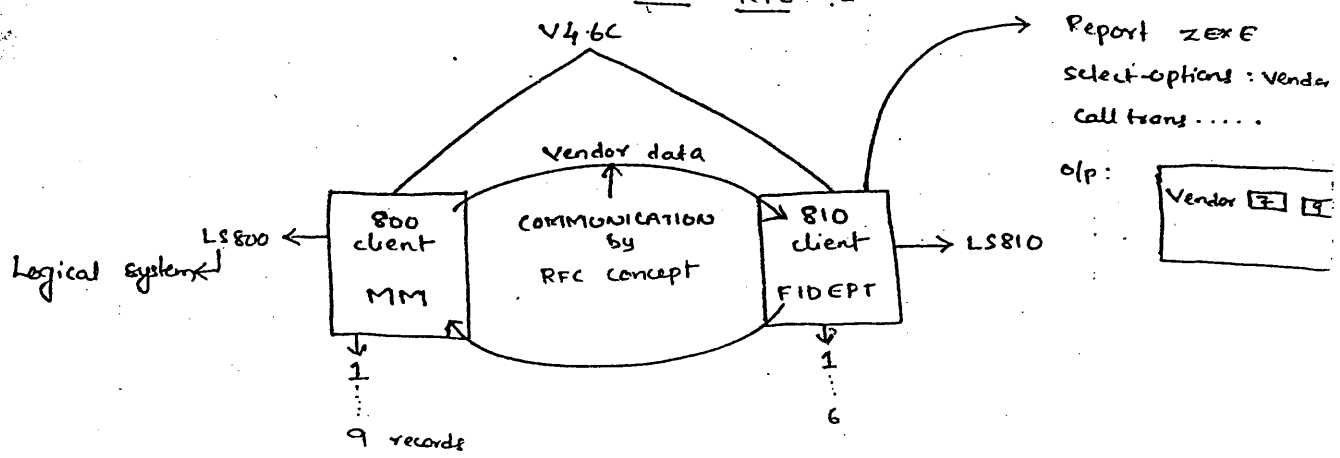
* Open Dataset file for input.
* apply cobol statements for read multiple records from flat file. i.e

```
DO.
READ DATASET FILE INTO ITAB.
```

* apply the logic to avoid infinite logic.

```
IF SY-SUBRC <> 0.
EXIT.
ENDIF.
Perform statements. ....
.....
.....
Call tr. logic....
Enddo.
Close Dataset file.
```

CALL TRASACTION METHOD WITH RFC :-



Logical systems : are replacement to the flat files.

* Logical systems are client identities.
* Logical systems are client independent and version dependent.

RFC STEPS:

1. Define Logical systems.
2. Assign Logical systems to clients.
3. Maintain communication with RFC.

DEMO: RFC WITH TWO DIFFERENT CLIENTS IN SINGLE SERVER.
(46CIDESLAB)

Go with client 800 and

go with T-code : SALE

go with sending & Receiving systems.

→ Logical systems.

→ Define logical system.

F8.

cross client is nothing but client independent ↵

go with new entries

If logical system is client independent we define both LS in client 800. (i.e both are in same server) –

perform sub1 using 'EKKO-EBELN' ITAB-EBELN.
" " " " BUKRS' ITAB-BUKRS.
" " " " EKORG' ITAB-EKORG.
" " " " EKGRP' ITAB-EKGRP.

perform sub1 using 'BDC_OKcode' 'INSE'.

call transaction 'YVendorTrans' using Itab mode 'N'
Messages into Messtab.

apply log file logic.

Loop at MESSTAB.
IF Messtab-MSGTYP = 'I' (Information)
OR MESSTAB-MSGNR = 000.
WRITE : /1 'vendor', 15 ITAB-LIFNR, 25 'INSERTED'.
ELSEIF Messtab-msgtyp = 'E'
OR messtab-msgnr = 001.
Write : /1 'vendor', 15 ITAB-LIFNR, 25 'NOT inserted'.
Endif.
Refresh messtab.
Endloop.
Enddo.
close Dataset file.

Define Subroutines SUB, SUB1

Form Sub using A B.
clear Itab.
Itab-program = A.
Itab-Dynpro = B.
Itab-Dynbegin = 'x'.
Append Itab.
EndForm.

Form SUB1 using C D.
clear Itab.
Itab-FNAM = C.
Itab-FVAL = D.
Append Itab.
Endform.

Ctrl+F8.
provide flat file in select option : C:\BDC\Jagan.TXT.
↓
Appl server path

go with client 810 and

go with SM59 for RFC Maintainence.

go for create

provide the destination : LLS800.

Logon details of application

client : 800

User : SAPUSER

Pw : abab ↵

provide the Target host : classroom (servername)

ctrl+s

go with client 800

go with SE11 For create structure using appl fields

select datatype : YJaganstr.

provide fields according to appl i.e

LIFNR	LIFNR
LAND1	LAND1
NAME1	NAME1.

ctrl+s

ctrl+F3

go with SE37 T.code for function builder.

provide function modulename : Y_MJaganFM

go for create

provide function group : YJagangroup

↵

select attributes :

go with processing type is

⊙ Remote enabled model.

go with Import :

Define import parameters			Passing	Pass by value
VENDOR_LOW	LIKE	LFM1-LIFNR		☑
VENDOR_HIGH	"	"	—	☑

For both parameters enable checkbox for Pass by values.

LLS800 Logical system for 800.

LLS810 " " " 810.

. Ctrl+S

F3

F3

go with Assign client to logical system.

F8 ↵

Double click on client 800.

Assign the logical system which we defined for that client 800

Logical syst : LLS800.

Ctrl+S.

F3

Double click on client 810

Assign the logical system which we defined for 810 client

Logical system: LLS810.

Ctrl+S

go with SM59 (T.code for RFC Destination Maintenance)

go for create

provide 810 logical system : LLS810.

provide connection type : 3 (communication b/w R/3 Systems)

provide description : RFC for 800.

provide Logon details (Destination data)

Language : EN

client : 810

User : SAPUSER

Password : abab ↵

provide the Target host : classroom

↳ system id (Ip address)

Here we dont know the Ip address so we keep server name in Target host i.e classroom. status bar shows servername.

go to Tables option

Define internal table based on structure.

KTAB LIKE . . YJaganstr.

↳ structure which we designed.

go to the sourcecode.

Write the logic in sourcecode after comments.

```
SELECT LIFNR LAND1 NAME1 FROM LFA1 INTO TABLE KTAB .
                    Where LIFNR between Vendor_Low and Vendor_.
```

Ctrl+S

Ctrl+F3

go with client 810

Define executable program : YJagonRFC

Tables: LFA1.

Select-option : Vendor for LFA1-LIFNR.

Define ITAB for Data uploading.

```
Data: Begin of ITAB OCCURS 0,
      LIFNR like LFA1-LIFNR,
      Land1 like LFA1-Land1,
      Name1 like LFA1-Name1,
      End of Itab.
```

* Define Jtab for Mapping.

Data: Jtab like BDCDATA OCCURS 0 with Header line.

Call function : Y_MJaganFM ↵

↳ RFC function module name

* provide the following statement which we created in ' under call function statements.

```
Destination 'LLS800'
Exporting
Vendor_low = Vendor-low
Vendor_high = Vendor-High.
    KTAB = ITAB.
Loop at ITAB.
Refresh Jtab.
```

" Sub1 " 'IFN1-LIFNR' ITAB-LIFNR.

" " " " Land1' ITAB-Land1.

" " " " Name1' ITAB-Name1.

Call transaction 'YVTRANS' using Itab mode 'A'
↳ Append mode.

Endloop.

Define subroutines same as previous programe SUB, SUB1.

Ctrl+F3.

go to client 800 and provide, application Tr.code and insert some records. YVTRANS (T.code)

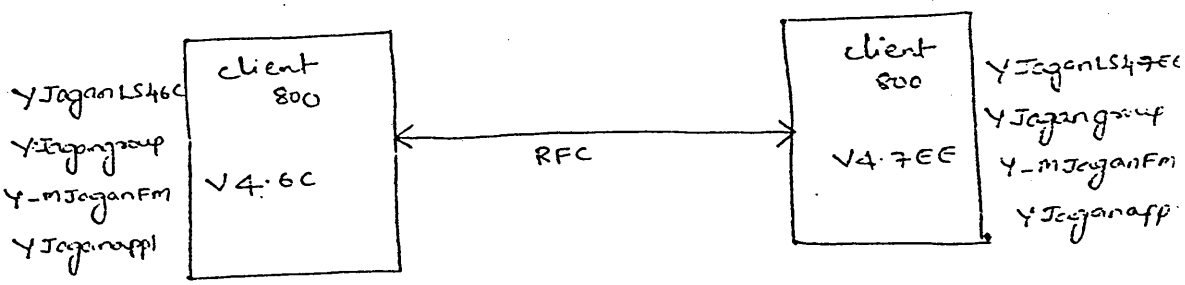
go to client 810 and F8

provide the records which we want to Transfer.

Vendor : [2] [6] ↵

Same records data will be transfered to 810 and inserts to db.

DEMO: RFC WITH TWO DIFFERENT SERVER CLIENTS i.e (or) TWO DIFFERENT VERSIONS V4.6C TO V4.7EE



Steps:

1. First define logical system in V4.6C client 800 and assign to logicalsystem. (YJagan46C) using T.code SALE

2. Define function module using application fields structure, Here structure we have to create using appl fields

3. Go to V4.7EE and define logical system and assign it to client 800

4. Define function module using structure fields.

5. Define executable program and same function module in it

6. Insert fields in tabc and some fields are inserted after execution of executable program.

procedure is same as previous program and include above steps.

THANKS FOR GIVING NOTEBOOK BACK

M. Jagadresh BE
C.no: 78552-55087 (M)
08745-246416 (R)
Jagadresh_G@yahoo.co.in

12/8/05

BDC :

↳ LSMW (Legacy system migration workbench)

Transaction code is : LSMW

LSMW is Data migration tool provided by SAP
↳ Data Transfering.

Advantage : LSMW can implement mapping logic automatically.

Versions in LSMW

V1.1
1.2 → SAP R/3 V3.0, V3.1
1.3
1.6
1.7 → SAP R/3 V4.x ↳ 6c 7EE

Demo is based on V1.7 LSMW.

DEMO : LSMW WITH BATCH INPUT METHOD.

go for Tr. code LSMW

provide the project name : YJaganPr

provide the sub project name : YJaganPr1

provide the object name : YJaganPr11 ↵

* project is nothing but an interface program name.
* sub project is attributes to the project
* object is an identity while transfering the data.

go for create option in tool bar button

provide description for project : project for LSMW ↵

provide description for sub project : sub project for LSMW ↵

description for object : object for LSMW. ↵

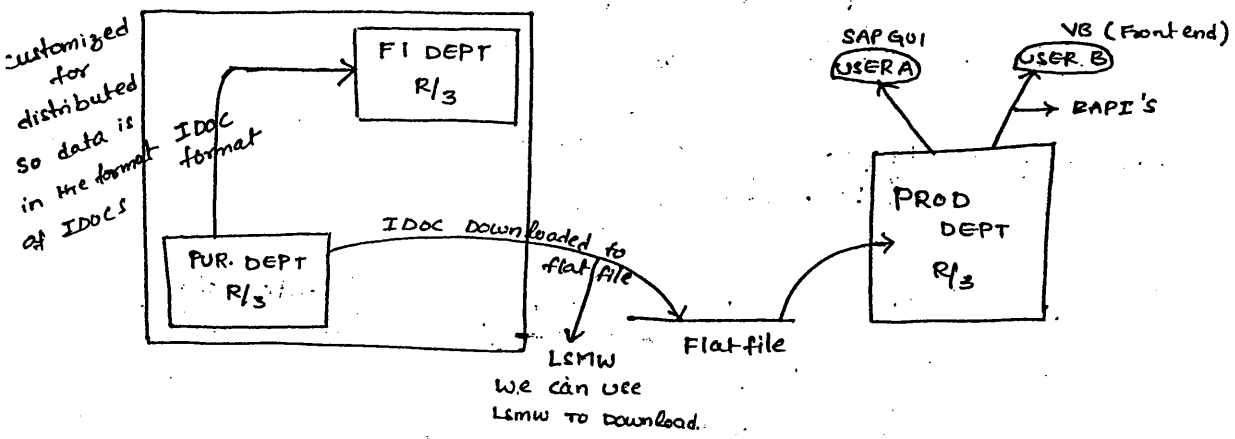
F8 (execute)

It will display a window having 14 LSMW steps.

1. go with first step ⊙ Maintain object attributes and press F8

go to the change mode

* LSMW with BAPI's and IDOC's for distributed R/3 systems.



* IDOC is Intermediate document it is a data carrier across distributed R/3 systems.
* With LSMW Data can download from IDOC to flat file.
* Diagram FI & PUR.DEPT are integrated so which can use IDOC format for data transfering.
* With BAPI's user can carrier data from third party front ends ex VB to SAP R/3 system.
* With LSMW user can design interface with BAPI's.

From first step select the option Batch input Recording.

provide Recording name : YJaganRec

go with overview i.e right side button or)

go to system → Recording overview.

go for create recording option in tool bar

provide Recording name : YJaganRec

provide description : Recording for appl

provide which Tr.code we are doing recording : YJaganapp. → vendor appl

provide dummy data in application.

perform insert operation and exit operation

ctrl+s

Double click on LIFNR field

provide the field at an int table . LIFNR

Description : Vendor acc number

Delete the Dummy value ↵

Repeat the same process for remaining 2 fields Land1 & Na.

Ctrl+S
F3
F3
Ctrl+S
F3

* With object attributes user can select a method required for Data transfering.

* LSMW Doesn't supports call transaction method.

Go for second step :

② Maintain source structure.

Source structure means an Int. table for data uploading.

F8.

go for create

provide source structure : KTAB (provide the int table)

Description : Int. table for LSMW. ↵

Ctrl+S
F3

③ Maintain source fields :

→ Int. table fields

F8

place the control on int. table KTAB

go for create option in toolbar

provide first field : LIFNR

Description : Vendor acc no. ↵

Select LIFNR go for create

provide next field : Land1

Description Land↵

select Land1 go for create

provide last field Name1

Description : Name 1 ↵

④ Maintain structure relations.

F8

* With structure relations int table can be assigned with db tables.
(<<<<) symbol shows already assigned.

ctrl+s

F3.

⑤ Maintain field mapping and conversion rules.

F8

select the first field LIFNR

go with assign source field in toolbar

Double click on LIFNR.

select next field Land1

go with assign source field

Double click on Land1.

If all Records Kaaar want to have same country then.
we go for constant data

select Land1

go with rule option (RULE) in toolbar

select is constant ↵

provide the constant value [IN] ↵

(or)

Under Rule option.
select abap editor ↵
provide following st for some records one country and remaining another country.

```
Loop at ktab.
IF KTAB-LIFNR <= 50000.        ↳ LIFNR NO
YJJREC-LAND1 = 'IN'.
ELSE.
YJJREC-LAND1 = 'US'.
ENDIF.
Endloop.
```

select Name1 assign sourcefield in toolbar

Double click on Name1

ctrl+s

F3.

⑥ Maintain fixed values, Translations user-defined routines.
↳ subroutines.

* It is for reusability (Ex: Subroutines)

⑦ Specify files

F8

select the first option

Specify the filename : C:\RLS.TXT

Description : File for LSMW

provide Tabulator is Delimiter.
↳ TAB.

Select file structure

☑ Field names at the begining of the file.

which provides the notepad file is in any sequence but having fieldnames on Top of the records.

ctrl+s

Design that sheet file

NAME1	LIFNR	LAND1
x	1	
y	2	

Here Land is constant 'IN' which can provide system by default.

F3

⑧ go with assign files.

F8

* with assign files ws_upload will be implemented.

ctrl+s

F3.

⑨ Read data :

ws_upload will be executed

F8
F8
F3

we can see the records which readed by syst

⑩ Display read data :

with display read data Loop & endloop will be executed.

F8 ↵

⑪ go with convert data :

with

(12) Display converted data (Display purpose)

F8 ↵

F3.

(13) Create batch Input session

F8

F8

With this step session method program can execute.

(14) Run batch Input session.

F8

With this step control leads to SM35

select the groupname

go with process

select Foreground

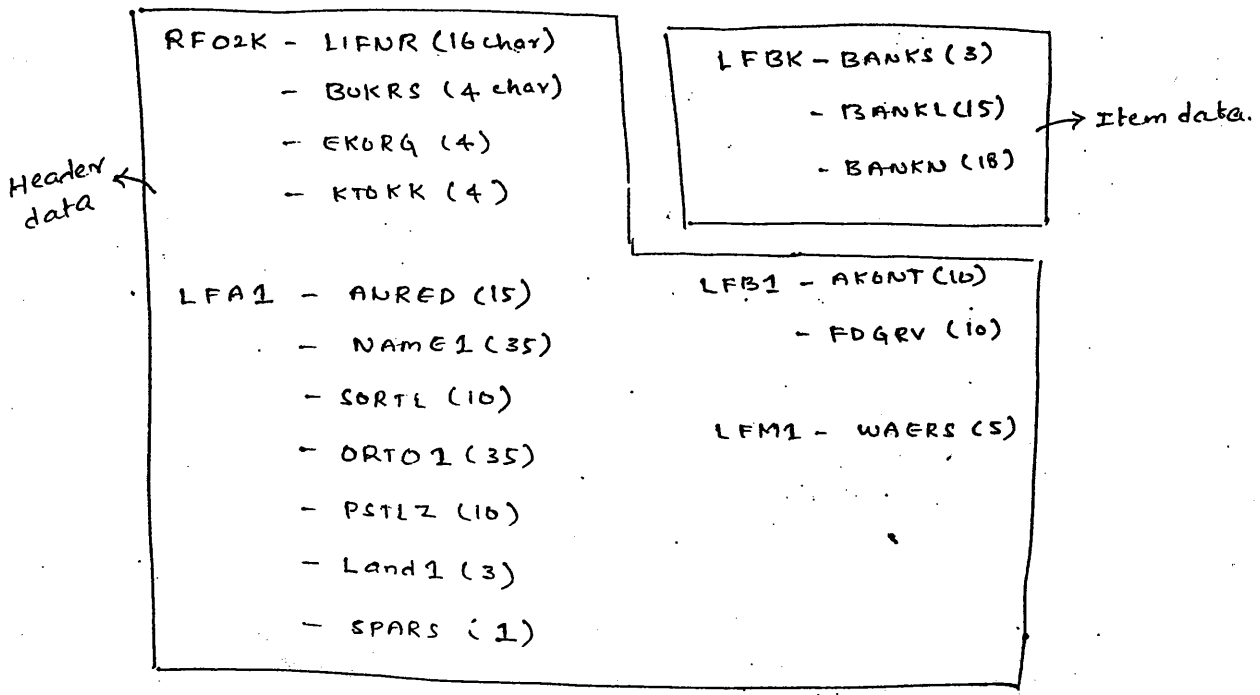
process.

Notepad records will appear in application.

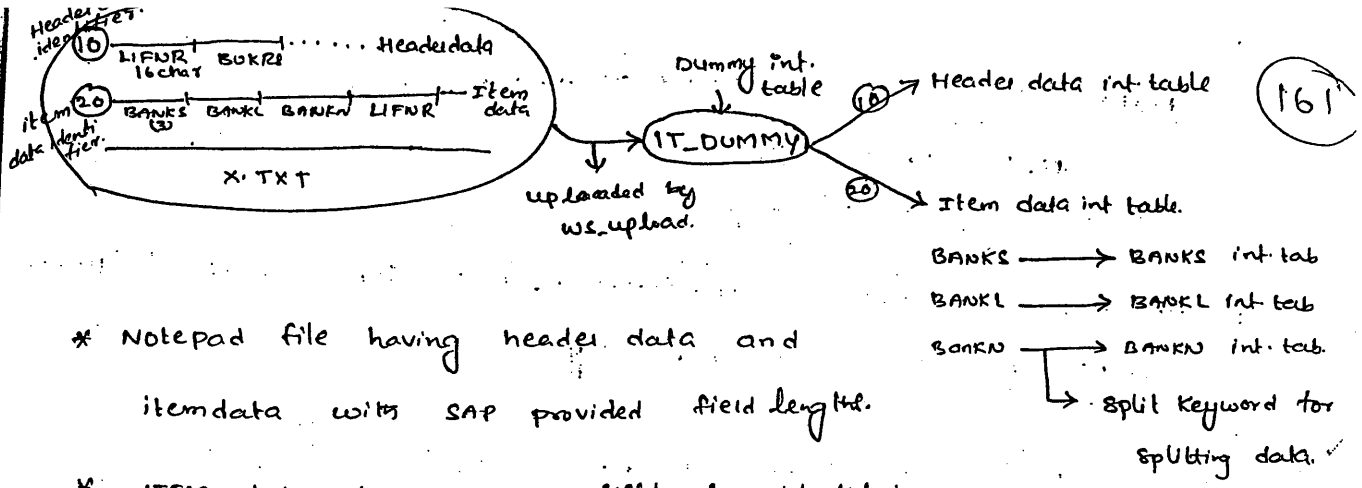
3/8/05

CALL TRASACTION METHOD WITH XK01 (Vendor master create)

Fields of XK01 appl.



Field lengths



* Notepad file having header data and itemdata with SAP provided field length.
* ITEM data have LIFNR field for identifying header data in item data.
* 10, 20 are Header data and item data identifiers.
* Segregate the data by using identifiers in dummy data and transfer to Header data & item data int. tables.

DEMO:

Go with T.code SHDB

go for create recording

Rec. Name : YJaganRec

Tr.code : XK01 (For which appl we doing recording)

start recording.

provide dummy data in application

after complete recording

ctrl s

F3

Select recording go for create program.

provide program name :

Select Transfer from recording ↵

provide attributes to the program

go with source code

provide Tables Workarea

Tables : RF02K , LFA1 , LFBK .

Define Dummy Int table. 'IT_DUMMY' for uploading all field to

```
Data : Begin of IT_DUMMY OCCURS 0,
       DUMMY(200) Type C,
       End of IT_Dummy.
```

* Define two Int. tables IT_Header, IT_Items for transfering Header data and item data respectively.

```
Data : Begin of IT_Header OCCURS 0,
       LIFNR  like  RF02K-LIFNR,
       BUKRS        "
       EKORG        "
       KTOKK        "
       ANRED  like  LFA1-ANRED,
       NAME1        "
       SORTL        "
       ORT01        "
       PSTLZ        "
       LAND1        "
       SPRAS        "
       AKONT        LFB1
       FDGRV        "
       WAERS.       LFM1
       End of IT_Header.

Data : Begin of IT_ITEMS OCCURS 0,
       BANKS  like  LFBK-BANKS,
       BANKL   "     "  -BANKL,
       BANKN   "     "  -BANKN,
       LIFNR   "     RF02K-LIFNR, (For identifying header data)
       End of IT_ITEMS.
```

Define int. tables with item fields.

```
Data : Begin of Banks OCCURS 0,
       Banks like KNBK-Banks,
       End of Banks.

Data : Begin of BankL OCCURS 0,
       BankL like KNBK-BankL,
       End of BankL.

Data : Begin of BankN OCCURS 0,
       BankN like KNBK-BankN,
       End of BankN.
```

Define variables FLD , CNT

Data : FLD(20) Type C.

CNT(2) Type N.

Header

After start of selection

Call function : WS_Upload

exporting

filename : 'C:\Jagan.TXT'.

filetype : 'ASC'

tab : IT_Dummy.

Read the data from Dummy Int table

Loop at IT_Dummy.

with identifiers segregate the data

IF IT_Dummy-Dummy+0(2) = '10'.

MOVE IT-Dummy-Dummy+2 TO IT_Header.

Append IT_Header.

Elseif IT_Dummy-Dummy+0(2) = '20'.

MOVE IT_Dummy-Dummy+2 TO IT_ITEMS.

Append IT_ITEMS.

EndIF.

Endloop.

After perform open-group.

Loop at IT_Header.

Refresh BDCData.

Instead of Dummy values provide IT_Header files

IT_Header-LIFNR.

provide the comments for perform statements related to item data fields.

After perform bdc_dynpro

Loop at IT_ITEMS when LIFNR = IT_Header-LIFNR.

Refresh : BankS , BankL , BankN.

apply split logic.

```
Split IT_ITEMS-BANKS AT ' ' INTO table BankL.
Split IT_ITEMS-BANKL AT ' ' INTO " BankL.
Split   "    -BankN  "  "  "  "  " BankN.

MOVE 1 TO CNT.
Loop at Banks.
Concatenate 'LFBK-Banks(' CNT ')' INTO FLD.
Perform BDC_field Using FLD Banks-Banks.
Endloop.

Repeat same Move statements for BankL, BankN.
Instead of Dummy values provide Header data field for remaining
                                                    Header files.
Endloop.
```

Ctrl+F3
F8

Work with call tr. method.

DIRECT INPUT METHOD :

FEATURES:

* This method can work for large amount of data only.
* Direct input is faster than batch Input method (session method)
* In case of Direct input method Validation will be done based on pre-defined function modules whereas in session method Validations will be done based on applications so direct method is faster than session method.
* PRE-DEFINED DIRECT INPUT PROGRAMS:

1. RMDATIND → Direct input program on Material master appl.
2. RFBIDE00 → " " " " customer Master appl.
3. RFBIKR00 → " " " " Vendor master appl.

By using above programs, user can download the records throu flat file into R/3 system.

* This method can work in foreground as well as in background but by default is foreground.

* With Restart Mechanism direct input programs can execute in background.

RBMVSHOW (Prog. name)
(or)
BMVO (T. code)

By executing RBMVSHOW (or) BMVO Restart mechanism can be implemented i.e providing authorization to syst to executes in background.

* Its have log file concept by default.

DEMO: BASED ON MATERIAL MASTER APPL. (RMDATIND)

RMDATGEN is the program to download data from material maste appl. to flat file.
↓
Pre defined program.

go with abap editor.

Provide program name : RMDATGEN.

F8

Let us provide which material no data we want to download.

: 628 (go for any existing)

select option

⊙ select write file to presentation server i.e ws_download executes.

* which have a option write file to appl. server i.e open dataset execs.

provide filename : C:\Jagan.TXT

F8

Flat file can create

go with abap editor.

LOADING: provide the prog. name: RMDATIND

F8

select the option ⊙ using physical filename

provide filename: c:\Jagan.TXT (what we downloaded in last step)

provide option LOCK mode: E (exclusively)

E indicates unlocking automatically.

F8 ↵ ↵

System will provide mat. no implicitly.

17/8/05

BDC REVIEW:

* While processing the data if user come across errors in session method process will be continued and error records goes to log file because process in Asynchronous. In call Transaction method process will be stopped because process is Synchronous.

* If user come across errors while updating the data in session method updation stops because updation is Synchronous where as in call transaction method updation continues because Asynchronous updation and error record goes to log file.

* User can rectify the error in that instance without checking log file we can use call transaction method with error mode (E) in statement. i.e error records will display on that instance user can rectify that on that instance only.

* LSMW is for one time requirement only and it contains in a production client. so if any requirement occurs with one time then we go for LSMW.

(167)

* LSMW is for small amount of data only because it executes the each record through 14 steps so large data we are not able to executes.

* WHY LSMW DOESN'T SUPPORTS CALL TRANSACTION is :

1. LSMW is for one time requirement whereas call transaction for frequent usage.

2. LSMW doesn't supports because call transaction log file also should design explicitly.

17/8/05

BDC — PREDEFINED PROGRAMS :

Why we go for user defined behind of pre-defined is whenever application is enhanced or modified for these applications predefined BDC programs are not compatible so we create programs

go with SPRO T. code for customizing.

go with SAP Reference img.

go with an option search

provide the search term : Data transfer ←
↳ i.e BDC programs.

select the program Data transfer workbench : Fixed assets.

Double click on this.

BDC

* Call Transaction Method : 100% objects are user defined.
* Session method : 20% are pre-defined, 80% user defined.
* Direct input : 100% predefined.

while creating a program we are using function module

call function : ws_upload i.e providing filename, file type and int. table but it is restricted to end user to use same file name what programmer given so we go for another option parameters for to not restrict end user i.e

parameters : file(200) like RLGRAP-fieldname.

⇒ RLGRAP is DB structure for provide selection screen in BDC interface programe.

ASSIGNMENT! Call tr & session methods

1. ME21 – Purchase order create.
2. VA01 – sales order create.
3. Records with errors should download automatically and successfull records update db.

RESUME POINT OF VIEW:

Projects :

1. Support (post imple)

BDC objects with changemode : XK02, XD02, MM02, VA02.

2. upgradation :

BDC objects with create mode : XK01, XD01, MM01, VA01.

ABAP REPORTS

↳ ABAP PROG. (Data extraction)

- Reporting types:

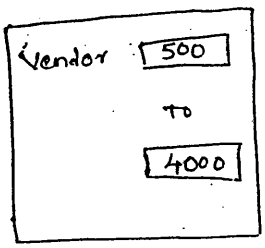
1. classical reporting. (95% us
2. Interactive reporting.
3. Logical database reporting (LDB concept)
4. ABAP query.

* classical reporting is equivalent to standalone programming.

EVENTS IN ABAP REPORTS:

1. Initialization: which triggers before selection screen display.

```
Ex:   Report ZEXE.
      Tables: LFA1
      select-options: Vendor for LFA1-LIFNR.
      Initialization.
      Vendor-Low = 500.
      Vendor-High = 4000.
      Vendor-option = 'BT', (Between)
      Vendor-sign = 'I'. (Including)
      Append vendor.
```



once initialization triggers Input will display on selection-screen.

2. AT selection-screen: with this user can implement validations in abap reports. It triggers after processing user Input still selection screen in active mode.

```
AT selection-screen on vendor.
IF vendor-low < 500. or
   vendor-High > 4000.
Message E000(CO) with 'Enter proper input'.
EndIF.
```

3. Start-of-selection: it triggers after processing selection screen. under this event following statements can be executed.

```
select ......
```

4. TOP-OF-PAGE: It provides header for abap reports

5. END-OF-PAGE: It provides footer for abap reports i.e page nos.

170

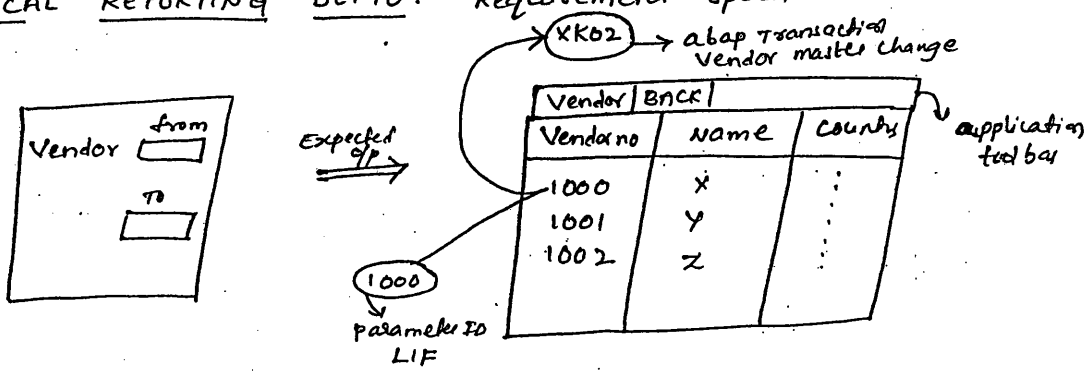
6. AT PF : PF is predefined function keys.

Which triggers behind predefined function keys.

7. AT user-command: for user defined function keys.

sfelos

CLASSICAL REPORTING DEMO: Requirement specification is



parameter ID can supply memory to field values.

After displaying output cursor place on any vendor no and perform vendor option in appl tool bar control will move to XK02 transaction application.

DEMO:

Let us create executable program.

* program start with Report program name then extent this by

Report YJaganprog NO Standard page heading Line-count 30(3).

* Here NO standard page heading will suppress the title defined in program attributes.

* 30(3) : No. of lines in page is 30 and gap b/w pages is 3 lines.

* provide Tables workarea.

Tables: LFA1.

* Design selection screen with select options.

select-option : Vendor for LFA1-LIFNR.

e.

Define internal table according to output requirement.

```
Data : Begin of ITAB occurs 0,
       LIFNR like LFA1-LIFNR,
       Land1 like LFA1-Land1,
       Name1 like LFA1-Name1,
       End of itab.
```


Define variable FNAM, FVAL.

```
Data : FNAM(10), FVAL(10).
```

* provide the event Initialization.

Initialization. (with this apply logic for providing input values).

```
Vendor-Low = 1000.
Vendor-High = 3000.
Vendor-Sign = 'I'.
Vendor-Option = 'BT'. (Between)
Append Vendor.
```

* Apply the event at selection screen for validating input values which we defined in Initialization.

```
AT selection-screen ON vendor.
IF Vendor-Low < 1000 OR
   Vendor-High > 3000.
Message E000(0) with 'Enter Between 1000 and 3000'.
EndIF.
```

* provide the event start-of-selection for display output.

```
start-of-selection.
Select LIFNR Land1 Name1 from LFA1 into Table itab where
                        LIFNR IN vendor.
Loop at itab.
Write : / ITAB-LIFNR, ITAB-Land1, ITAB-Name1.
Endloop.
```

* For providing vendor and Back options in application toolbar Define following statements.

```
Set PF-status 'yJagan status'.
              ↳ status name.
```

Select appl -

provide the first option : Vendor.

(172)

Double click on vendor ↵

provide function text ie Description : Vendor ↵

select one of the function key F2 ↵ ↵

Repeat same process for Back.

Ctrl+F3

F3.

* Apply the event AT USER-COMMAND for control vendor, Back options because these function keys are user defined so at user-comand.

19/8/05

AT USER-COMMOND.
CASE SY-UCOMM.
WHEN 'VENDOR'.
GET CURSOR FIELD FNAM VALUE FVAL.

* The above statement for identify the record which user selected.

* Using F1 is Technical information user can identify parameter ID for respective fields (XKO2) → LIFNR using parameter ID memory can be supplied for respective values.

FOR any field First three letters are parameter ID.

* Write the logic to supply the memory for respective value.

SET parameter ID 'LIF' field FVAL.

* Apply call transaction logic to move cursor to abap transaction.

CALL transaction 'XKO2' and SKIP first screen.

↳ it is going to skip first screen of XKO2 appl.

ENDCASE.

* BACK will work automatically.

* Provide the Header for this report use TOP-OF-PAGE.

TOP-OF-PAGE.

* provide footers for this report use END-OF-PAGE.

```
END-OF-PAGE.
write :/ 'pageno:', sy-pageno.
```


ctrl+S

ctrl+F3.

F8.

Select one of the vendor.no from output list.

perform option 'Vendor'

cursor will move to abap transactions.

19/8/05

Assignment:

1. provide Difference with and without Initialization in abap programs.

Initialization provides default values but without initialization we have to provide input values.

2. provide the sequence of events in classical reporting.

Top-of-page

Initialization

AT selection-screen

Start-of-selection

AT user-command.

End-of-page.

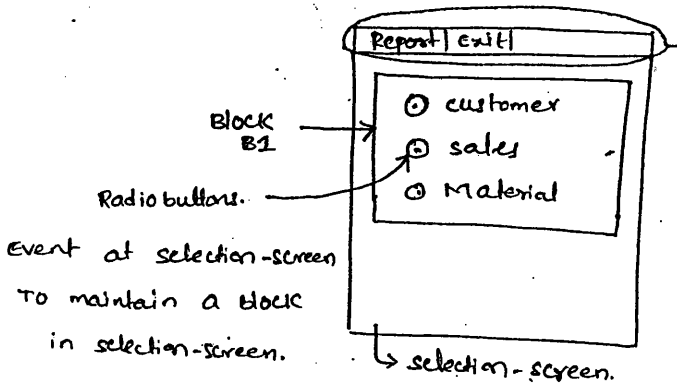
3. Why Top-of-page trigger first while start-of-selection.

Top-of-page provides Title so it triggers first.

4. Identify the first event triggers in abap program.

Top-of-page.

classicReporting: DEMO with RADIO BUTTONS and BLOCKS.



→ use db structure SSCRFIELDS for selection screen options. Maximum we can keep 5 options in tool bar.

FunctionText_01
FunText_02
" 03
" 04
" 05
} 5 options.

DEMO: Let us create executable program..

provide Tables workarea.

Tables : LFA1, VBAK, MARA, SSCRFIELDS.

Define Int. table itab for customer Radio button.

```
Data: Begin of itab occurs 0,
      KUNNR like Kna1-Kunnr,
      Land1 like Kna1-Land1,
      Name1 like Kna1-name1,
      End of itab.
```

* Itab for sales radio button.

```
Data : Begin of Itab occurs 0,
       VBELN like VBAK-VBELN,
       NETWR like VBAK-NETWR,
       END of Itab.
```

* Ktab for material radio button.

```
Data : Begin of Ktab occurs 0,
       MATNR like MARA-matnr,
       MEINS like MARA-MEINS,
       End of Ktab.
```

Let us design selection screen with a block.

selection-screen Begin of block B1 with frame title Text-001.
↳ For provide Title.

Parameters : customer Radiobutton group RG1,
sales Radiobutton group RG1,

... requirement.

Data : Begin of ITAB occurs 0,
LIFNR like LFA1-LIFNR,
Land1 like LFA1-Land1,
Name1 like LFA1-Name1,
End of itab.

Define variable FNAM, FVAL.

Data : FNAM(10), FVAL(10).

* provide the event Initialization.

Initialization. (with this apply logic for providing Input values).

Vendor-Low = 1000.
Vendor-High = 3000.
Vendor-Sign = 'I'.
Vendor-Option = 'BT'. (Between)
Append Vendor.

* Apply the event at selection screen for validating Input values which we defined in Initialization.

AT selection-Screen ON Vendor.
IF Vendor-Low < 1000 or
Vendor-High > 3000.
Message E000(0) with 'Enter Between 1000 and 3000'.
EndIF.

* provide the event start-of-selection for display output.

start-of-selection.
Select LIFNR Land1 Name1 from LFA1 into Table itab where LIFNR IN Vendor.

Loop at itab.
Write : / ITAB-LIFNR, ITAB-Land1, ITAB-Name1.
Endloop.

* For providing vendor and Back options in application toolbar Define following statements.

set PF-status 'yJagan status'.
↳ Status name.

Double click on that status name ↵

Select application toolbar

provide the first option : Vendor.

Double click on Vendor ↵

provide function text i.e Description : Vendor ↵

select one of the function key F2 ↵ ↵

Repeat same process for Back.

Ctrl+F3

F3.

* Apply the event AT USER-COMMAND for control Vendor, Back options because these function keys are user defined so at user-comand.

AT user-command.
case sy-ucomm.
When 'Vendor'.
Get cursor field FNAM value FVAL.

* The above statement for identify the record which user selected.

* Using F1 is Technical information user can identify parameter ID for respective fields (XKO2) ↳LIFNR using parameter ID memory can be supplied for respective values.

For any field First three letters are parameter ID.

* Write the logic to supply the memory for respective value.

Set parameter ID 'LIF' field FVAL.

* Apply call transaction logic to move cursor to abap transaction.

call transaction 'XKO2' and SKIP first screen.

↳ it is going to skip first screen of XKO2 appl.

Endcase.

* BACK will work automatically.

* Provide the Header for this report use TOP-OF-PAGE.

TOP-OF-PAGE.
WRITE : / 'Vendor Details'.

19/8/0

* provide footes for this report use END-OF-PAGE.

```
END-OF-PAGE.
write : / 'pageno:', sy-pageno.
```

Ctrl+S
Ctrl+F3.
F8.

Select one of the vendor no from output list
perform option 'Vendor'
cursor will move to abap transaction.

19/8/05

Assignment:

1. provide Difference with and without Initialization in abap programs.

Initialization provides default values but without initialization we have to provide input values.

2. provide the sequence of events in classical reporting.

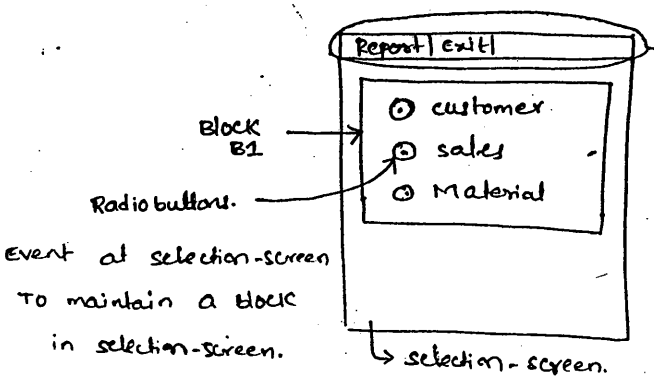
Top-of-page
Initialization
AT selection-screen
Start-of-selection
AT user-command.
End-of-page.

3. Why TOP-of-page trigger first while start-of-selection.

TOP-of-page provides Title so it triggers first.

4. Identify the first event triggers in abap program.

TOP-of-page.



→ use db structure SSCRFIELDS for selection screen options. Maximum we can keep 5 options in tool bar.

FunctionText_01
FunText_02
" 03
" 04
" 05
} 5 options.

DEMO: Let us create executable program..

provide Tables workarea.

Tables : LFA1, VBAK, MARA, SSCRFIELDS.

Define Int table itab for customer Radio button.

```
Data: Begin of itab occurs 0,
      KUNNR like Kna1-Kunnr,
      Land1 like Kna1-Land1,
      Name1 like Kna1-name1,
      End of itab.
```

* Jtab for sales radio button.

```
Data : Begin of Jtab occurs 0,
       VBELN like VBAK-VBELN,
       NETWR like VBAK-NETWR,
       END of Jtab.
```

* Ktab for material radio button.

```
Data : Begin of Ktab occurs 0,
       MATNR like MARA-matnr,
       MEINS like MARA-MEINS,
       End of Ktab.
```

Let us design selection screen with a block.

```
selection-screen Begin of block B1 with frame title Text-001.
```
↳ For provide Title.

```
Parameters : customer Radiobutton group RG1,
             sales Radiobutton group RG1,
             material Radiobutton group RG1,
```

Double click on Text-001 ↵

provide the title for that block

: classic Report

ctrl+S

ctrl+F3

F3.

Selection-screen Function Key 1. (Report)

Selection-screen Function Key 2. (Exit)

Initialization.

SSCRFIELDS-FUNCTXT_01 = 'REPORT'.

SSCRFIELDS-FUNCTXT_02 = 'Exit'.

provide the logic for Report and exit.

AT selection-screen.

IF SSCRFIELDS-Ucomm = 'FC01'

↳ User command.

SSCRFIELDS-Ucomm = 'ONLI'.

↳ is a keyword which can move the control within selection screen for displaying output.

ELSEIF SSCRFIELDS-Ucomm = 'FC02'.

LEAVE. (Move the control out of the selection screen).

EndIF.

* Logic for displaying output.

start-of-selection.

IF customer = 'x'.

↳ Default in abap language.

select KUNNR Land1 Name1 into table itab.

Loop at itab.

write: / itab-Kunnr, itab-Land1, itab-Name1.

Endloop.

ELSEIF sales = 'x'.

select VBELN NETWR from VBAK into table Jtab.

Loop at Jtab.

write: / Jtab-VBELN, Jtab-NETWR.

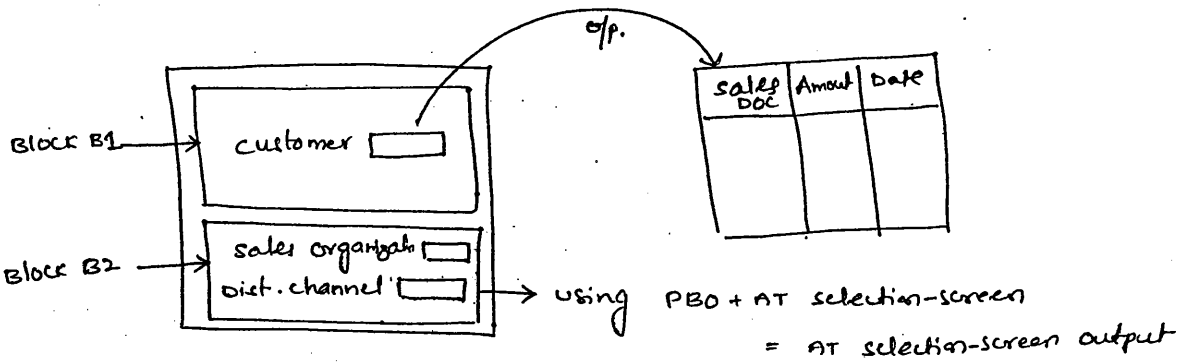
```
ELSE
select MATNR MEINS from MARA into table Ktab.
Loop at Ktab.
Write : / Ktab-MATNR, Ktab-MEINS.
EndLoop.
EndIF.
```

Ctrl+F3.

F8.

select one Radiobutton and perform Report option it will disply output for respective input.

DEMO: classical Report on Validation:



Requirement is by default block B2 is in disable mode and after entering input in customer field if it is correct then enable block B2 otherwise disable. for this at selection-screen output is event.

DEMO: Let us create executable program,

provide tables workarea.

```
Tables : KNA1, VBAK.
Define Int table acording to output Requirement.
Data : Begin of itab occurs 0,
       VBELN like VBAK-VBELN,
       ERDAT like VBAK-ERDAT,
       NETWR like VBAK-NETWR,
       End of itab.
```

* Define selection-screen with blocks B1, B2.

selection-screen Begin of block B1 with frame Title Text-001.

parameters : cust like kna1-kunnr.

(181)

selection-screen End of block B1.

selection-screen Begin of block B2 with frame Title Text-002.

parameters : SORG like VBAK-VKORG MODIF ID ABC,
DCH like VBAK-VTWEG MODIF ID ABC.
↳ group name.

selection-screen end of block B2.

* MODIF ID can maintain fields under one group so we can able to validate on that group.

Double click on Text-001 and Text-002

provide Titles to blocks i.e

1. Customer Details
2. Sales Details ↵

Ctrl+F3

F3

AT selection-screen output.

Loop at screen.

IF Screen-group1 = 'ABC'.

IF VAL = 1. (i.e (False) NO input)

Screen-input = 0 (input is wrong)

Modify screen

ELSE.

VAL = 2.

Screen-input = 1

Modify screen.

Endif.

Endif.

* Group1 is a component provided by SAP in Screen.

* Input is a component " " " " " .

182

Logic for to findout Input is Right or wrong.

```
AT selection-screen on cust.
select single * from kna1 where kunnr = cust.
IF sy-subrc <> 0.
  message E000(0) with 'Enter correct input'.
  exit.
else.
  val = 2
endIF.
```

* provide logic for display output.

```
start-of-selection.
Select VBELN ERDAT NETWR from VBAK into Table itab
        where  kunnr = cust OR
               VKORG = SORG OR
               VTWEG = DCH.

Loop at itab.
write : / ITAB-VBELN , ITAB-ERDAT , ITAB-NETWR.
Endloop.
```

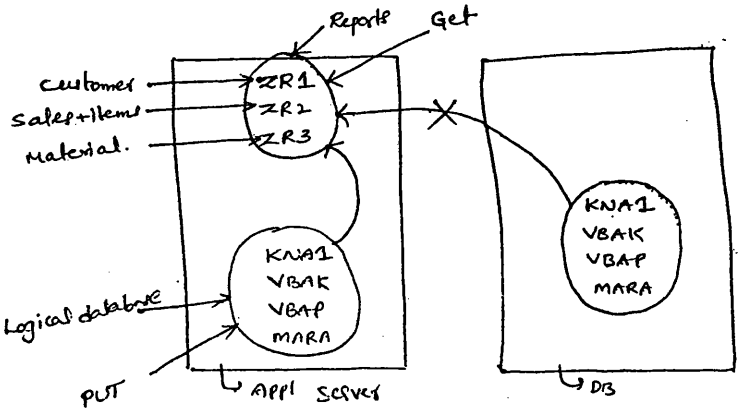
Ctrl+F8

F8.

20/8/05 : ABAP Reports:

↳ Logical database reporting (LDB concept)

T-code : SE36. (or) SLDB.



* With LDB's user can create database logically in application servers.

* Using LDB's user can improve the performance of abap reports.

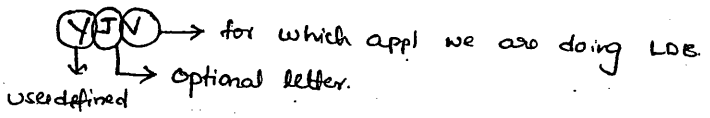
EVENTS of LDB :

1. Get
2. PUT
3. Get late
4. Get late reject
5. Get late reject table
6. End-of-selection.

* put writes the data in logical database. put keyword get the data from database to LDB.

* Get reads the data from logical database.

Naming conventions: Name of LDB should be 3 letters:



DEMO:

go with SE36 or SHDB.

provide LDB. name : YJV

go for create.

provide short text for LDB. : LDB for Demo.

Ctrl+S

provide Name of root node (i.e which report is starting first there customer report is rootnode)

: KNA1

provide short text : customer node.

provide database table name : KNA1

go for create.

select root node KNA1

go for create

Short text : sales node.

provide database table : VBAK.

go for create.

Select VBAK node

go for create

provide next node : VBAP

Short text : ITEMS

db table : VBAP.

go for create.

select VBAP node go for create

Nodename : MARA

Description : material

db table : MARA

go for create.

Ctrl+s

go with F3

Under Subobjects : go with selections.

* Selections generate selection-screen automatically

go to the changemode ↵

go with NO option ↵ (i.e we no need create matchcode object for this because SAP already providing this option)

Delete the comment for KNA1 and select-options and instead of ? mark provide select option name.

Select-options : customer for KNA1-KUNNR.

Repeat the same steps for VBAK

select-options : sales for VBAK-VBELN.

Delete the comments for VBAP select-options.

Here VBAP-VBELN is already provided so go with

Repeat the last for Material for MARA-MATNR.

Ctrl+S

Ctrl+F3.

go with F3.

select database program in subobjects.

* Database program is nothing but logical db driver program.

go to the changemode ↵

Drives program name : SAPDB YJV

SAPDB ← common. YJV → LDB name.

For any drives program SAPDB is common, LDB name will add to this.

Let us Double click second Include.

Let us Double click on first Include

Delete the comments for select statements according to our requirement.

i.e select * from KNA1

Wher Kunnr in customer

Delete the comment for Endselect.

Ctrl+F3

F3

Double click on second Include.

Delete the comment for select statements whichever we required.

F3

Repeat the same steps for remaining 2 Includes.

Activate Driver program.

go with abap editor.

provide the program name : YJVProg

Define attributes under attributes

provide logical database which we created : YJV

Save the attributes.

provide table work area

Tables : KNA1

provide Get event

Get KNA1

Ctrl+S
Ctrl+F3
F8.

Repeat the same executable program steps for remaining tables VBAK, VBAP, MARA.

---

* 197 pre-defined LDB's are provided by SAP.
We dont create LDB's in real time we use pre-defined LDB's.

Pre-defined LDB's :-

1. AKV - sales document logical db.
2. MRM - Material master logical db.
3. PAP - Applicant data logical db.
4. PNP - LDB for HR Master data, travel Management, Time Management.

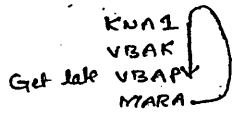
SE36 and use searchhelp for predefined LDB's

23/8/05

LDB events:

Get late.
Get late Reject.
Get late Reject table.
End-of-selection.

* With Get event always execution starts from Root node onwards.
* With Get late all subordinate nodes will be executes first later on execution starts from root node onwards.



* With Get late Reject can execute subordinate node only.
* With Get late Reject table can executes the node where user specifies that event (particular node only)
* End-of-selection triggers after processing db program.

ABAP Reports

→ Misleniou concepts

→ prog. analysis

→ T.code SE49.

* with program analysis user can search the tables required for abap report

go with SE49

select Transaction

provide T.code : XDO1

go for display it will display tables.

* with Recording also user can search the tables required for reporting (SHDB).

* SQL Tracer: T.code ST05.

with sql tracer user can check the performance apart from it user can search the tables required for reporting.

go with ST05.

go for option ~~Trace~~ Trace ON.

go with XKC1.

provide the dummy date in addi

go back to ST05

~~Disable~~ & go for Trace off. so that sql tracer became inactive.

go for list trace. ↵

It will display the object name i.e Tables which are searched.

* SQL Tracer will try to search for primary key tables.
* Recording will give particular field
* SQL Tracer will give total fields & tables.

ABAP Reports:

→ MIS. concept

→ POH (process on Help request).

POH Triggers with F1 function Key and it maintains user defined documentation. (abap prog 'document).

let us create executable program.

Tables : LFA1.

select-options : vendor for LFA1-LIFNR.

use the event

Design the screen.

Call Screen 100 starting at 5 5 ending at 55 10.
↳ screen size.

(188)

Double click on screen 100.

control leads to screen painter.

provide description for screen 100.

provide screen type is ⊙ modal dialog box

Next screen : 0 (Initial screen from where control started).

go with layout.

select second documentation field.

provide documentation (line by line)

go for Utility
↳ Settings

Under settings let us delete graphical layout editor.

(For multiple lines)

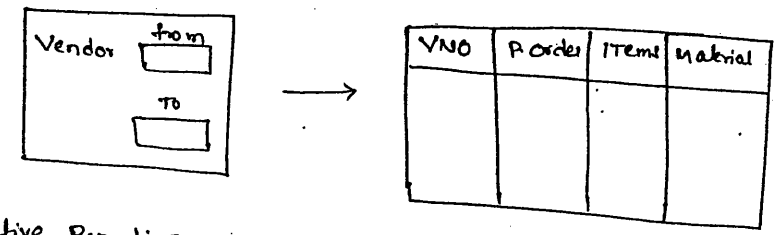
Ctrl+F3
F3
F3
F8.

24/8/05

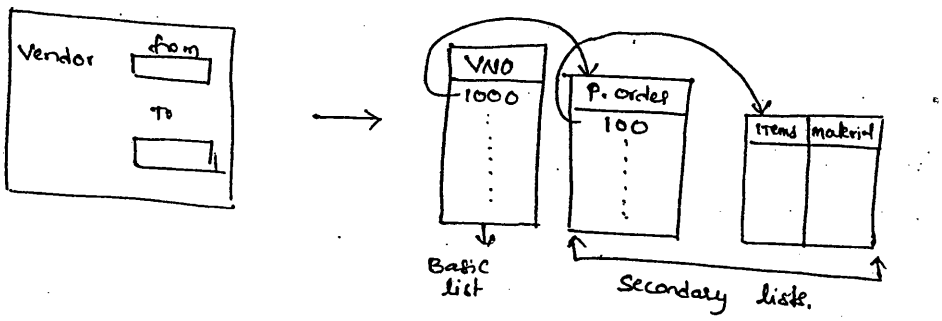
Interactive Reporting:

Difference b/w classical & Interactive reporting :-

Classic Reporting :-



Interactive Reporting :



* Here classical reporting output have extensive records i.e large no. of records it is difficult to check particular record so classic report is recommended to use for small amount of data. while large amount of data go for Interactive.

189

* With Interactive reporting user can generate a report with multiple lists.

* Under Interactive reporting user can generate maximum upto 20 secondary lists.

* even if user provides a logic for generate 21st secondary list instead of displaying that list the control leads to error analysis.

* It is possible for displaying 21st secondary list but in sequence. It is possible by moving back cntrol to Basic screen.

AT line-selection: Triggers whenever user clicks on a Record.

HIDE is a keyword and it can hold the record which user selected.

AT selection-screen, HIDE exclusively for Interactive reporting.

System variables for interactive reporting :-

1. SY-LSIND → List index.

It provides the current list no of interactive reporting.

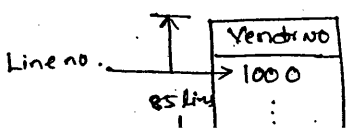
SY-LSIND = 0 - indicates Basic lists.

= 1 ... 20 → secondary lists.

Mandatory system field. (SY-LSIND).

(5). SY-LILLI : (Optional)

it provides line no. where user clicked. i.e GUI line no.



3. TOP-of-page is the event [illegible], report here Basic list is equal to classic Report output so TOP-of-page is used for Basic list. (190)

4. TOP-of-page during line-selection: (for secondary lists)

5. End-of-page.

6. AT PF

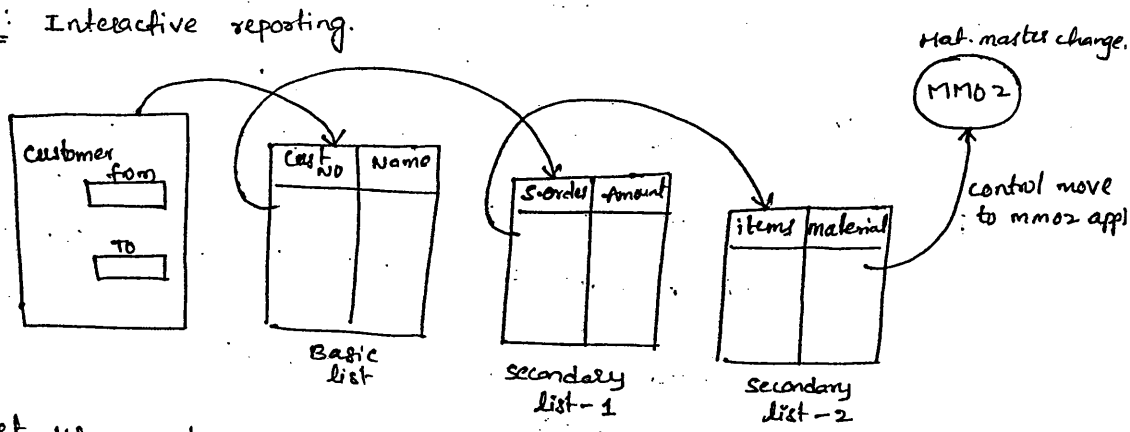
7. AT User command.

8.

Sequence of events in interactive reporting :-

1. Initialization.
2. AT selection-screen.
3. Start-of-selection.
4. AT line-selection.
5. TOP-of-page during line-selection.
6. End-of-page.
7. AT PF
8. AT User command.

DEMO: Interactive reporting.



Let us create executable programe.

Provide Tables work area.

Tables : KNA1, VBAK, VBAP, MARA.

Design selection-screen with select-options.

Select-options : cust for kna1-kunnr.

Define Int table itab for basic list, Jtab for First secondary list and Ktab for second secondary list.

Data : Begin of ITAB OCCURS 0,
kunnr like kna1-kunnr,
name1 like kna1-name1,
End of [illegible]

```
Data : Begin of Itab occurs 0,
       VBELN like VBAK-VBELN,
       NETWR like VBAK-NETWR,
       End of Itab.

Data : Begin of Ktab occurs 0,
       POSNR like VBAP-POSNR,
       MATNR like MARA-MATNR,
       End of Ktab.
```

Define variables required for Get cursor logic.

```
Data : FNAM(10) , FVAL(10).
```

Provide logic required for basic list.

```
Start-of-selection.
Select kunnr name1 from KNA1 into table itab where kunnr in cust.
Loop at itab.
Write : / itab-kunnr Hotspot , itab-name1.
HIDE ITAB-KUNNR.
Endloop.
```

Apply the logic for first secondary list.

```
AT line-selection.
  case SY-LSIND.
  When 1.
  Select VBELN NETWR from VBAK into table Itab where kunnr = ITAB-kunnr.
  Loop at Itab.
  Write : / Itab-VBELN HOTSPOT, Itab-NETWR.
  HIDE Itab-VBELN.
  Endloop.
```

HOTSPOT is a keyword for hand symbol for selection of record in output screen.

Apply logic for 2nd secondary list.

```
  when 2.
  Select VBAP~POSNR MARA~MATNR into Ktab from VBAP inner join
         MARA on VBAP~MATNR = MARA~MATNR where
                                          VBELN = Itab-VBELN.
  Append Ktab.
  Endselect
  Loop at Ktab.
  Write : / Ktab-POSNR , Ktab-MATNR HOTSPOT.
  HIDE Ktab-MATNR.
  Endloop.
```

* provide the logic to move the control from report to Transaction.

When 3.

Get cursor field FNAM value FVAL.

Set parameter ID 'MAT' field FVAL.

↳ Material parameter ID. (First 3 letters)

call transaction 'MM02' and SKip first screen.

Endcase.

* provide the logic to maintain Headers.

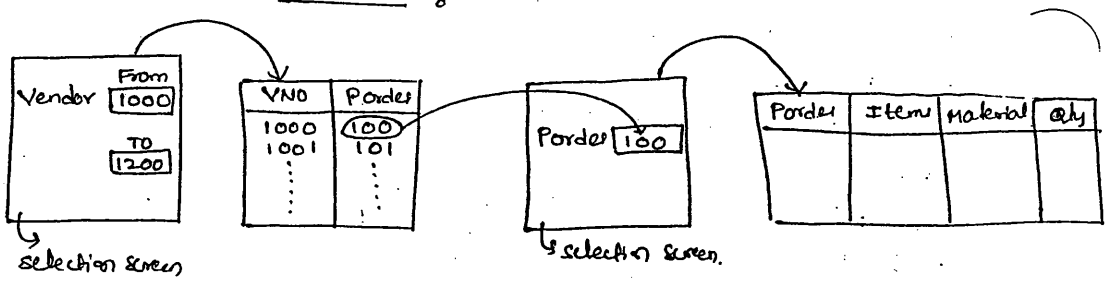
```
TOP-of-page.
Write : / 'customer details'.

TOP-of-page during line-selection.
Case SY-LSIND.
When 1.
Write : / 'sales details'.
When 2.
Write : / 'item details'.
Endcase.
```

Ctrl+F3

F8.

8/8/05 REPORT WITHIN A REPORT :-



In Interactive reporting user can call Report within a report.

* Using SUBMIT keyword user can call a report within a report.

DEMO:

Let us Create executable program

provide Tables workarea.

Tables : EKKO, EKPO, MARA.

Design selection screen with parameters.

parameters : pordes like EKKO-EBELN.

Define Int. table acording to dp requirement.

Data : Begin of itab occurs 0,

EBELN LIKE

EBELP LIKE EKPO-EBELP,
MENGE LIKE EKPO-EKPO,
MATNR LIKE MARA-MATNR,
End of itab.

* provide select statement.

select EKKO~EBELN EKPO~EBELP EKPO~MENGE MARA~MATNR
into table itab from EKKO inner join EKPO on
EKKO~EBELN = EKPO~EBELN inner join MARA on
EKPO~MATNR = MARA~MATNR where EKKO~EBELN = Porder.

Loop at itab.
write: / itab-EBELN, itab-EBELP, ITAB-MENGE, ITAB-MATNR.
Endloop.

ctrl+F3
F3

Let us create executable program.
provide tables workarea.
Tables : LFA1, EKKO.
Design selection screen with select-options.
select-options : vendor for LFA1-LIFNR.
Define an int. table.
Data : Begin of itab occurs 10,
LIFNR like LFA1-LIFNR,
EBELN like EKKO-EBELN,
End of itab.
Define variable required for get cursor.
Data : FNAM(10), FVAL(10).
provide select statement.
select LFA1~LIFNR EKKO~EBELN into table itab from LFA1 inner
join EKKO on LFA1~LIFNR = EKKO~LIFNR where
LFA1~LIFNR in vendor.
Loop at itab.
write: / itab-LIFNR, itab-EBELN.
Endloop.
Using an event at line-selection. use a submit keyword to call Report within a report.

AT line-selection.
case sy-LSIND.
when 1.
[illegible] FNAM value FVAL

Transaction codes are

SQ01 → Query

SQ02 → Infoset (or) Functional area.

SQ03 → user group.

ABAP Query is a reporting tool provided by SAP it can generate logic automatically. i.e 90% logic will generated by system.

Navigation for ABAP QUERY :-

go with T. code SQ02

provide infoset name : Y Jaganinfoset (previous versions it is Functional area).

go for create.

provide the description for info set : Infoset for Demo.

Under datasource :

* Abap query with LDB maintains selection-screens in LDB itself.
* Abap query with Join by basis table doesn't maintains selection-screens.

select the option.

⊙ Table join by Basis table.

provide the first table : KNA1. ↵

go with insert table in toolbar

provide the next table : VBAK ↵

Select VBAK table go with insert table option

provide next table : VBAP ↵

Select VBAP table go with insert table

provide last table : MARA ↵

go with F3

we get a window with 3 options in which

select create empty field group ↵

select field group 01 go for Delete field group option in toolbar.

" " " 02 " " " " " "

" " " 03 " " " " " "

" " " 04 " " " " " "

Here for 4 tables 4 field groups are generated so we delete it

then create one group for dlp req

go for create field group.

provide field group name & description.

FG — Field group for demo ↵

↳ Field group name.

Select the fields which we display in output.

go with KNA1 table

select KUNNR

place the control on PG (field group)

go for insert field in field group option in toolbar.

go with VBAK table

select VBELN

go for insert field option in toolbar.

go with VBAP table.

select POSNR

go for insert field in field group option

go with MARA table select MATNR

go for insert field in field group.

Ctrl+S

generate it (by using Toolbar option)

generate is nothing but activate.

* Infoset provides Declarations required for reporting.

* With Infoset tables [illegible] int. table

29/8/07

* Usergroup is a collection of user ID's

provide usergroup name : YJagngrap

go for create

Description : DEMO

ctrl+s.

go for Assign users and Infosets option.

go with assign infoset option in toolbar.

☑ select the infoset which we created in list

ctrl+s

go with SQ01 T-code for abap query

provide query name : YJaganQuery.

go for create

select infoset and double click on that

provide title for a report : Customer sales details Report.

DEMO:

go with basic list option in toolbar.

Let us double click on respective fields inside the tablenodes

System provides select statements automatically.

go with Query painter menu option

→ Execute (F8)

→ Executed (F8)

System provides loop and write endloop automatically.

F8 ↵ (or) F8.

If we want check the automatically generated report statements then go with same session only

go with abap editor

go with changemode

System will provide report.

29/8/05

Reports

→ ALV's → Abap list viewer.

* With ALV's user can provide performance for abap reports.

ALV Types:
1. Simple ALV's
2. Hierarchical ALV's
3. Blocked ALV's

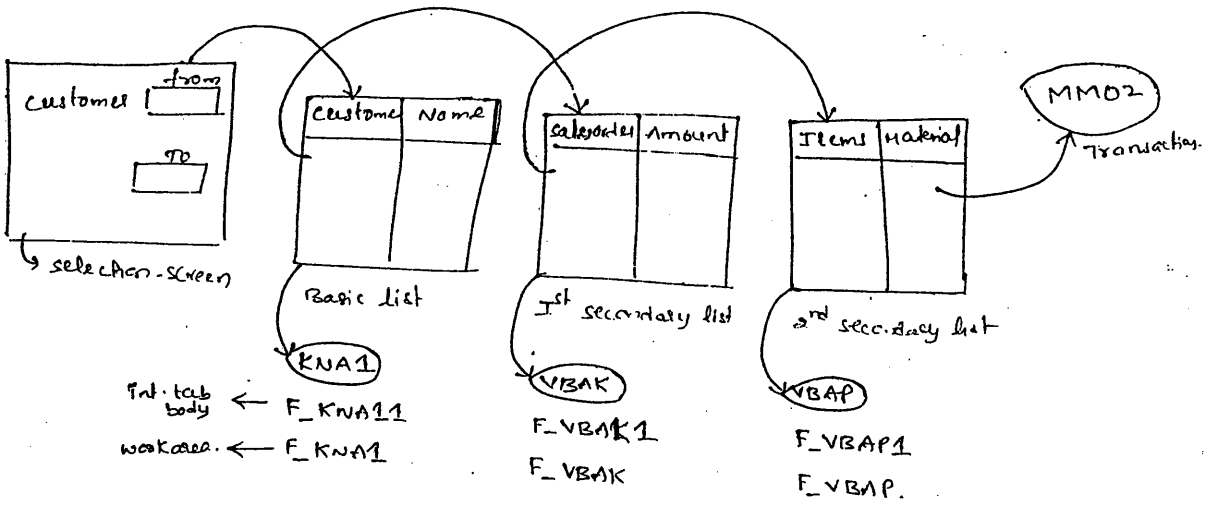
Working with Line Type & Row Type concepts in ALV's while executing

Implement ALV's using classes is called object oriented abap.

Type groups is used by ALV's and Type group can hold reusable datatypes and constants.

SLIS is Type group regarding ALV's.

DEMO:-Interactive reporting with ALV's :-



Let us create executable program.

Tables: KNA1, VBAK, VBAP.

Design selection screen

Select-option: cust for KNA1-KUNNR.

Define Internal tables itab, itab1, ... secondary lists Respectively.

Data: Begin of itab occurs 0,
KUNNR like kna1-kunnr,
Name1 like kna1-Name1,
End of itab.

Data: Begin of itab occurs 0,
VBELN like VBAK-VBELN,
NETWR like VBAK-NETWR,
End of Itab.

Data: Begin of Ktab occurs 0,
POSNR like VBAP-POSNR,
MATNR like VBAP-MATNR,
End of Ktab.

* working with type group SLIS
Type pools : SLIS.
Data : Repid like sy-repid.
↳ is a system field for program name i.e report id which we are working currently.

F_KNA11 type SLIS_T_FIELDCAT_ALV,
↳ Int. table body ↳ Datatype.

F_KNA1 type SLIS_FIELDCAT_ALV,
↳ Int. workarea. ↳ Field string.

* Declare same way for all fields required for output

F_VBAK1 type SLIS_T_FIELDCAT_ALV,
F_VBAK type SLIS_FIELDCAT_ALV,
F_VBAP1 type SLIS_T_FIELDCAT_ALV,
F_VBAP type SLIS_FIELDCAT_ALV,

* provide statements for events.

I_Events type SLIC_T_Event,
↳ Contains type groups and events required for reporting.
S_Events type SLIS_ALV_Event.

* call subroutine
Perform Get_VAL.
REPID = SY-REPID.
* provide the select statement for basic list.

Select KUNNR Name1 from KNA1 into table itab where kunnr in cust.

* In ALV's no need to provide Loop, endloop statements i.e
By avoiding Loop endloop statements ALV's can improve performance.

call function : REUSE_ALV_LIST_DISPLAY ↵

which can displays output in presentation server i.e equal to Loop, write, endloop statements.

Delete comments for following statements.

Exporting.

I_callback_program = REPID

IT_Fieldcat = F_KNA11

IT_Events = I_Events.

Tables

T_outtab : Itab.

Define subroutine which we call before select statement.

Form Get_val.

F_KNA1-fieldname = 'KUNNR'.

F_KNA1-REF_Tabname = 'KNA1'.

F_KNA1-REF_Fieldname = 'KUNNR'.

Append F_KNA1 TO F_KNA11. (Workarea to body).

* For providing colour headings we can define workarea and body for respective fields.
* Repeat same logic for all fields in output.

F_KNA1-fieldname = 'Name1'.

F_KNA1-REF_Tabname = 'KNA1'.

F_KNA1-REF_Fieldname = 'Name1'.

Append F_KNA1 TO F_KNA11.

F_VBAK-Fieldname = 'VBELN'.

F_VBAK-REF_Tabname = 'VBAK'.

F_VBAK-REF_Fieldname = 'VBELN'.

Append F_VBAK TO F_VBAK1.

F_VBAK-Fieldname = 'NETWR'.

F_VBAK-REF_Tabname = 'VBAK'.

F_VBAK-REF_

```
F_VBAP-REF_Tabname  = 'VBAP'.
F_VBAP-REF_Fieldname = 'POSNR'.
Append F_VBAP TO F_VBAP1.

F_VBAP-Fieldname  = 'MATNR'.
F_VBAP-REF_Tabname = 'VBAP'.
F_VBAP-REF_Fieldname = 'MATNR'.
Append F_VBAP TO F_VBAP1.

S_Events-Name  = 'User_command'.
S_Events-Form  = 'VAL'.
   ↳ Subroutine   ↳ Subroutine name
Append S_Events TO I_Events.
EndForm.
```

* Define Subroutine 'VAL'.

(@ equal to at line-selection)

```
Form VAL using User-Command like SY-UCOMM.
             SEL type SLIS-SELFIELD.
               ↳ equal to Hide.
```

Depends on fieldvalue define data objects accordingly.

```
Data : CUS(10) type N,
       SAL(10) type N,
       MAT(10) type C.

IF SEL-Fieldname = 'KUNNR'.
   CUS = SEL-VALUE.
```

* provide select st. for secondary list-1.

```
Select VBELN NETWR from VBAK into table itab where kunnr = cust.
Call function : Reuse_ALV_list_Display.
   Delete comments for following.
   I_Callback_program : REPID.
   IT_Fieldcat        : F_VBAK1
   IT_Events          : I_Events
   Tab                : Itab.
Endif.
```

```
IF SEL-Fieldname = 'VBELN'.
   SAL = SEL-VALUE.
* Select statement for second secondary list.
  Select POSNR MATNR from VBAP into table Ktab where VBELN = SAL.
Call function : Reuse_ALE_POPUP_TO_Select
                                  ↳ Function module which can display
                                    o/p in popup window.
Delete * comments for following statements.
   exporting.
    I_Title : 'Item details'
    I_screen_start_coloumn = 20
      "      "   line   = 5
      "    end  coloum  = 60
      "      "   line   = 40
    I-tabname = 'VBAP'.
   IT_fieldcat = F_VBAP1
   I_callback-program = PEPID
    ES_Selfield = SEL
    Table
      outtab = Ktab
 EndIF.
    IF SCL-Fieldname = 'MATNR'.
      matt = SEL-value.
*   Move the control from report to TR.
   Set parameter ID 'MAT' Field MAT.
   Call transaction 'MM02' and Skip first screen.
    EndIF.
   Endform.
```

Simple ALV's

↓

Object oriented abap

↓

using grid control

* With ALV grid control user can display and can manuplate multiple records.

ALV grid control is Enhancement to table controls concept.

While working with ALV grid control classes are used i.e

1. CL_GUI_ALV_GRID → it can display a grid control in o/p
2. CL_GUI_custom_container → it can identify the location for output display.

T.code SE24 for check classes provided by SAP.

classes for logo is :

CL_GUI_ALV_TREE_SIMPLE → it can display logo.

used type group SDYDO

DEMO:

Let us create executable program

provide tables workarea.

Tables : KNA1, VBAK.

go b/w 19 to 20 line space for call screen 100 st.

Declarations of Logo:

Data : itab type table of VBAK,

↳ working with object oriented prog

container type SCRFNAME value 'ALVcontrol',

↳ screen field name.

* container is a data object.

cust TYPE REF TO CL_GUI_custom_container,

↳ Keyword to refer classes.

Grid type REF TO CL_GUI_ALV_GRID,

Declarations for logo.

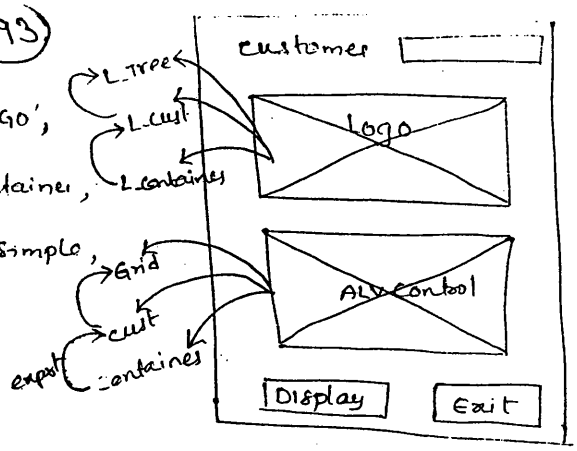
L_container type SCRFNAME value 'LOGO',

L_cust type REF TO CL_GUI_Custom_Container,

L_tree type Ref to CL_GUI_ALV_tree_Simple,

L_list type SLIS_T_listheader,

L_Logo type SDYDO_value.



Call screen 100.

Let us double click on screen 100. (control move to screen painter)

provide description : ALV grid control

go with layout

provide the customer field in layout

select custom control button from toolbar (last to second button) drop it

in layout define properties

Name : Logo.

once again select custom control drop it in layout and define properties

Name : ALVcontrol

provide the options display, exit (push buttons)

go with flow logic

Delete the comment for PBO program

double click on that

go with module x.extmodule

logic for display

Case sy_ucomm.

When 'DISP'.

Select * from VBAK into table itab where kunnr = kunnr-kunnr.

IF cust is Initial.

create object cust exporting container_name = container.

Create object grid exporting T_parent = cust.

```
call method grid -> set_table_for_first_display exporting
                    I_structure_name = 'VBAK'
                    changing IT_outtab = ITAB.
Endif.
IF L_cust is initial.
create object L_cust exporting container_name = L_container.
create object [illegible] L_Tree exporting I_Parent = L_cust.
perform logosub using. L_Logo.
call method l_tree -> create_Report_Header exporting
                    IT_list_commentary = L_List
                    I_logo = L_Logo.
EndIF.
When 'exit'.
leave program.
Endcase.
Endmodule.
```

* Define subroutine.

```
Form logosub using P_logo type SDYDO_value.
P_Logo = 'ENJOYSAP_Logo'.
                  ↳ Logo name.
EndForm.
```

cfolts
cbutts
F6.

3/8/05

ALV's

↓

Hierarchical ALV's → List view control (Demo is based on this)

↓

Object oriented abap

↓

classes

↓

CL_GUI_ALV_TREE_SIMPLE

CL_GUI_custom_container

Using
Line type
&
Row type

Let us create executable program

provide tables workarea.

Tables : KNA1, VBAK.

Let us go with 9 to line space and design screen then come back to declarations

* Define Int table based on VBAK table.

Data : itab type Vbak occurs 0,

Tree type Ref to CL_GUI_ALV_Tree_Simple,

Fcat Type LVC_T_Fcat,

Data object i.e ↵

Field catalog i.e for list of fields display in list view control

SORT_B Type LVC_T_SORT.

↓

For sorting while display output

Call scree 100.

Double click on screen 100 and provide description.

: ALV Hierarchical

go with layout

provide customes field in layout

select custom control button drop it in layout

Define properties for custom control

Name : LVcontrol

provide option display and exit.

go with flow logic

Let us delete comment for PBO program

Let us Double click on that

* DEMO is based on List view control

with List view control user can display multiple records but we are not able to manuplate records.

* go with after call screen 100 statement provide PBO subroutine for colum Headings.

Form COL_Head.

Call function (ctl+F6) : LVC_FieldCatalog_Merge ↵

exporting ↳ which can provide colum Heading.

I_Structure_Name : 'VBAK' [illegible]

list view control.

* Define second subroutine for output display.

```
Form output.
Select * from VBAK into table itab where kunnr = KNA1-kunnr.
Endform.
```

* Define subroutine for sorting the fields while displaying output.

```
Form sort.
Data : sort_w type LVC_S_SORT.
          └→ work area.
SORT_W_SPOS = 1.
          └→ sequence position is 1
SORT_W-fieldname = 'VBELN'.
Append sort_w to SORT_B.
                  └→ body.
SORT_W-SPOS = 2.
SORT_W-fieldname = 'ERDAT'.
Append SORT_W TO SORT_B.

SORT_W-SPOS = 3.
SORT_W-fieldname = 'ERNAM'.
Append SORT_W TO SORT_B.

SORT_W-SPOS = 4.
SORT_W-fieldname = 'NETWR'.
Append SORT_W TO SORT_B.
Endform.
```

* o with module & endmodule write logic for display & exit.

```
Case sy-ucomm.
When 'DISP'.
Perform callsub.
When 'exit'.
Leave program.
Endcase.
```

After endmodule define subroutine which we call for display.

11/8/05

Form callsub.

Call 3 subroutines which we defined earlier.

Perform COL_Head.

Perform output.

perform sort.

Data : container type SCRFNname value 'LVcontrol',
cust type Ref to CL_GUI_custom_container,

IF cust is initial.

Create object cust exporting container_name = container.

Create object tree exporting I_parent = cust.

call method tree-> set_Table_for_first_display

changing
IT_outtab = itab
IT_fieldcatalog = Fcat
IT_sort = SORT_B.

EndIF.

Endform.

ctrl+S
ctrl+F3
F8.

11/8/05

ALV's
↓
Simple ALV's
↓
Matchcode objects / POV concept (process on value request)

POV is stands for process on value request it triggers with F4 function key it is for search help.

* checktable is primary key table
* value table can work based on Domain

go with creating Domain under that we have option value range i.e it can insert btw that range.

* If it is matchcode objects it can work based on check table level.

based on check table as well as value table.

DEMO: SIMPLE ALV'S WITH POV CONCEPT:

Let us create executable program

provide Tables workarea.

Tables: KNA1, VBAK.

Let us Design selection screen

Parameters: cust like KNA1-KUNNR Default 1001.

select-option: sales for VBAK-VBELN.

Define Int. table itab with VBELN field for POV function.

Data: Begin of itab occurs 0,
VBELN like VBAK-VBELN,
End of itab.

Data: Itab like VBAK occurs 0 with header line (output display)

Data: IDEX Type I.

Type-pools: SLIS.

Data: Repid like sy-Repid,

VBAK_B Type SLIS_T_Fieldcat_ALV,

Int. body ← For providing colum headings.

Events_B Type SLIS_T_Event. (it is for events)

Call function: Reuse_ALV_Fieldcatalog_merge ←
↳ Function module which provides colum headings.

Exporting

I_Program_Name : REPID

I_Structure_Name : 'VBAK'

Changing

CT_fieldcat : VBAK_B.

Repid = sy-Repid.

AT selection-screen on value-Request for sales-Low.

Call function: Conversion_exit_Alpha_Input ←
↳ which can add leading zero's to Input and deletes leading zero's for

exporting

input = cust

Importing

output = cust.

* provide select statement for POV

Select VBELN from VBAK into table itab where kunnr = cust.

Call function : POPUP_WITH_Table_DISPLAY ↵

↳ Display list of values in popup menu.

exporting

EndPos_COL = 20

EndPos_ROW = 20

StartPos_Row = 5

startpos_ROW = 5

TitleText = 'Item details'

Importing

Choise = IDEX

Table

ValueTab = ITAB

Exceptions

Break_Off = 1.

* Logic to select one of the value from POPUP menu

Read Table itab index idex.

Sales_LOW = ITAB-VBELN.

IF SY-SUBRC <> 0.

Leave program.

EndIF.

Start-Of-Selection.

Select * from vbak into table Itab where vbeln in sales.

Call function : Reuse_ALV_list_display ↵ (For display output)

exporting

I_callback_program = RepId

IT_fieldcat : VBAK_B

IT_Events : Events_B

Tables

T_outtab : Itab.

Blocked ALV's :-

* With blocked ALV's different reports can be designed within single executable program.

* Layout is a Location where we displaying blocks.

DEMO:

Let us create executable program

Provide tables workarea.

Tables : LFA1, KNA1.

Design selection-screens

select-options : cust for Kna1-Kunnr,

select-options : vendor for LFA1-Lifnr.

Define itab for customer req, Jtab for vendor requirement

```
Data : Begin of itab occurs 0,
       Kunnr like kna1-kunnr,
       Land1 like Kna1-Land1,
       Name1 like Kna1-Name1,
       End of itab.

Data : Begin of Jtab occurs 0,
       LIFNR like LFA1-LIFNR,
       Land1 like LFA1-Land1,
       Name1 like LFA1-Name1,
       End of Jtab.

Type-pools : SLIS.

Data : Repid like sy-Repid,
       KNA1_B Type SLIS_T_Fieldcat_ALV,
       Layout_B Type SLIS_Layout_ALV, (For Location while displaying o/p)
       Events_B Type SLIS_T_Event.

Repid = sy-Repid.

select Kunnr Land1 Name1 from KNA1 into table itab
select Lifnr Land1 Name1 from LFA1 into table Jtab where
                                    LIFNR in vendor.
```

Call function : REU_ALV_BLOCK_list_init ↵

↳ which activates blocks. (20)

exporting

I_call'back_program = Repid.

Perform Appendblock Tables itab Using 'ITAB'.

perform Appendblock Tables Jtab Using 'Jtab'.

* Tables - keyword is for passing Int. tables in subroutines.

~~perform~~ ~~Appendblock~~

Call function : Reuse_ALV_block_list_Display ↵

↳ which displays the blocks in layout.

Delete all parameters.

* Define subroutine which we called before.

Form Appendblock tables Jtab Using itab.

Refresh KNA1_B.

Call function : Reuse_ALV_fieldcatalog_merge ↵

exporting

I_Program_Name = Repid

I_Internal_Tabname = ITAB

I_Inclname = Repid

changing

CT_fieldcat = KNA1_B.

Call function : Reuse_ALV_Block_list_Append ↵

↳ which can append 2nd block to the first block.

exporting

IS_Layout = Layout

IT_fieldcat = KNA1_B

I_tabname = ITAB

IT_Events = Events_B

Tables

T_outtab = Jtab.

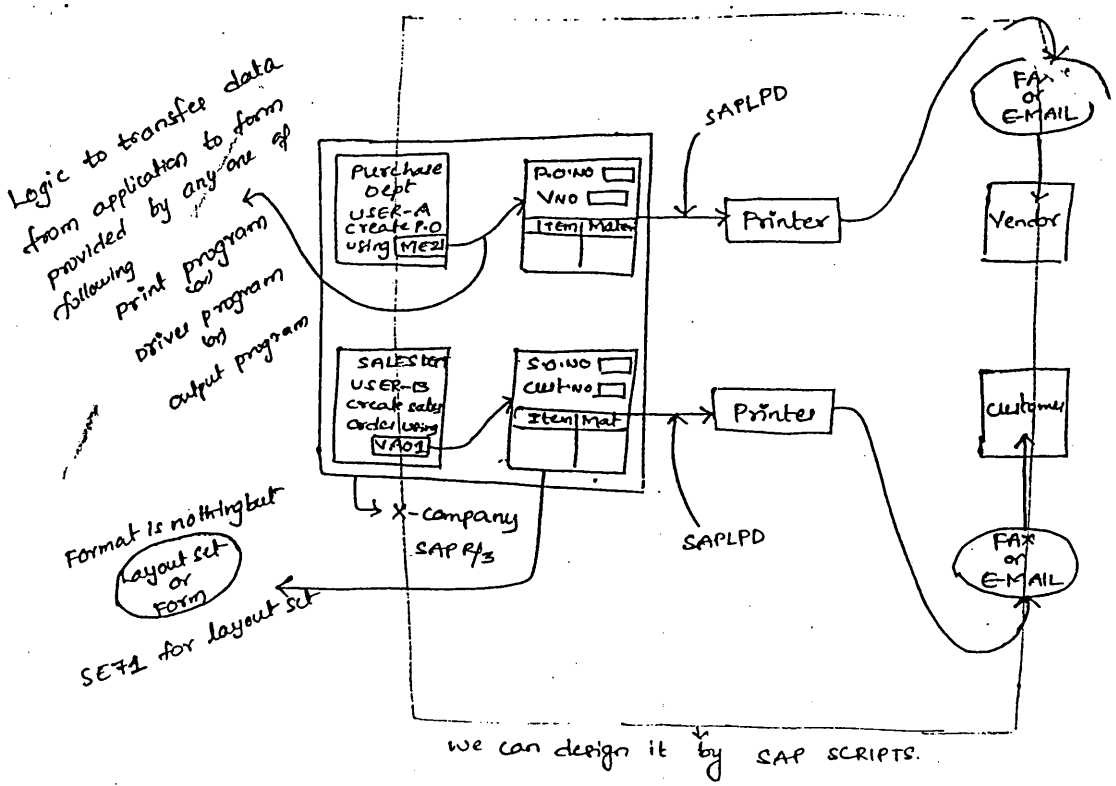
Endform.

Ctrl+F3

F8.

2/9/16.

SAP SCRIPTS



* SAPLPD provides Interface between SAPR/3 and printers.
* Using SAP scripts R/3 can communicate with business partners.
* Printers are the destination for SAP scripts.

Printers compatible for SAP scripts :

1. HPLJ4
2. KORIAN

DEMO: FORM DESIGNING :

go with SE71 T.Code

provide the form name : YJagan_Form

go for create ↵

Provide description : payment Remainders.

Select the option Translate under that

select ⊙ To all languages. (It will translate form into all languages in SAP)

go with Basic settings

Select page format : DIN A4 (Equal to A4 size page)

Select orientation ⊙ portrait format

203

Select Font family : COURIER

Font size : 12.0

go with pages option in appl. toolbar

page (i.e page no) : PAGE 1

Description : First page

If Form has multiple pages then go for following steps :

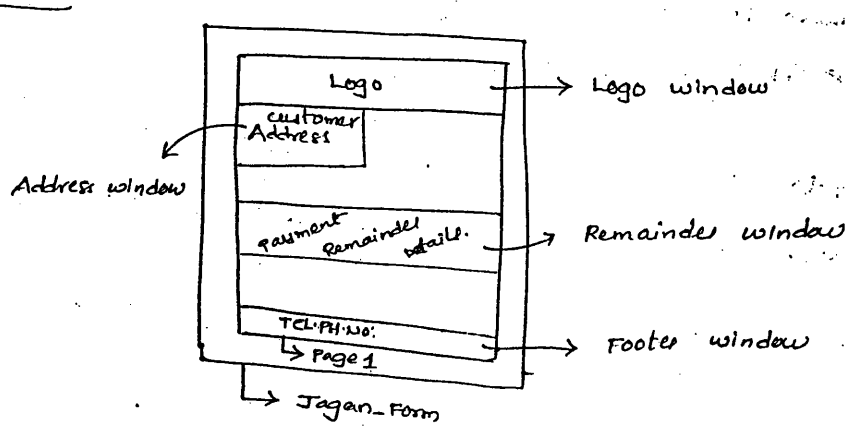
go with Edit → Create element

Provide Page no : Page 2

Description : second page

* Repeat above steps for as many pages.

Form Format:



In SAP scripts Windows are of 3 types :

1. Constant windows
2. Variable windows
3. Main windows.

* If it is constant windows those windows can reflect in all pages contains in a form

Ex. Letter pad contains company address with logo in top and

* Where as variable windows are restricted for multi page ....
i.e details contain in a letter pad. (Address etc....)

* From V4.6B onwards there is no between Constant and Variable windows.

* Main windows are for business information. Under one page we can maintain maximum of 99 main windows.

* 99 main windows for business labels as well as address labels in realtime.

* Without main window it is not possible to design a form.

* If there is a page breake in SAP scripts we can call it as Unprotected scripting i.e. After completing of first page control move to second page.

* If there is no page breake we can call it as protected scripting.

* Up-to V4.6B it is not possible to see a logo in print previews where as from V4.6C onwards it is possible.

* From V4.6C onwards. Scripts is compatible for E-MAIL

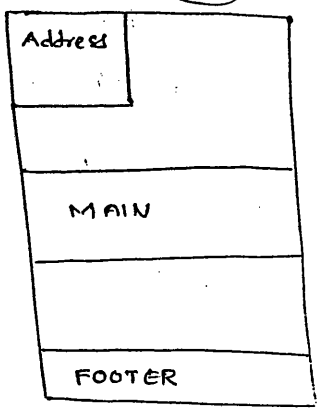
* In SAP scripts logo file format should be .TIF
From V4.6C onwards apart from .TIF format remaining formats also will be allowed.

RSTXLDMC is an executable program for logos

How forms can be transports for checcing is:

RSTXR3TR is an executable program for transporting forms.

go with windows option in toolbar



Under this option by default provides MAIN window so we can go for remaining two windows (ADD, Footer)

go with Edit
↳ go for create element

window name : Address

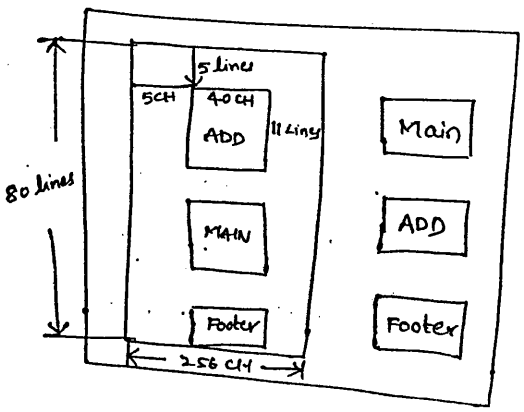
Description : Address window ↵

Once again Edit
↳ go for create element

window name : Footer

Description : ~~Address~~ Footer window. ↵

go with page windows. ~~windows~~ option in toolbar



go for Edit
↳ Create element

Let us Double click on Address window

Provide left margin : 5 CH (characters)

Upper margin : 5 LN (Lines)

window width : 40 CH

" length : 11 LN

Once again

* where as variable windows are restricted for that page only i.e details contain in a letter pad. (Address etc...)

* From V4.6B onwards there is no between constant and variable windows.

* Main windows are for business information. Under one page we can maintain maximum of 99 main windows.

* 99 main windows for business labels as well as address labels in realtime.

* Without main window it is not possible to design a form.

* If there is a page breake in SAP scripts we can call it as Unprotected scripting i.e After completing of first page control move to second page.

* If there is no page breake we can call it as protected scripting. i.e it is for single page.

* Up-to V4.6B it is not possible to see a logo in print previews where as from V4.6C onwards it is possible.

* From V4.6C onwards scripts is compatible for E-MAIL

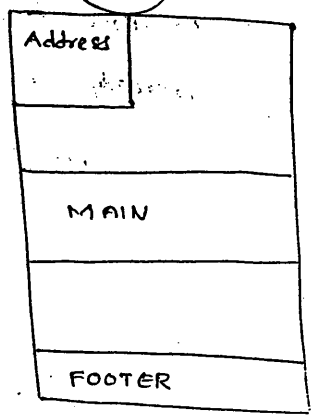
* In SAP scripts logo file format should be .TIF
From V4.6C onwards apart from .TIF format remaining formats also will be allowed.

RSTXLDMC is an executable program for logos

How forms can be transports for checcing is:

RSTXR3TR is an executable program for transporting forms.

go with windows option in toolbar



Under this option by default provides MAIN window so we can go for remaining two windows (ADD, Footer)

go with Edit
↳ go for create element

window name : Address

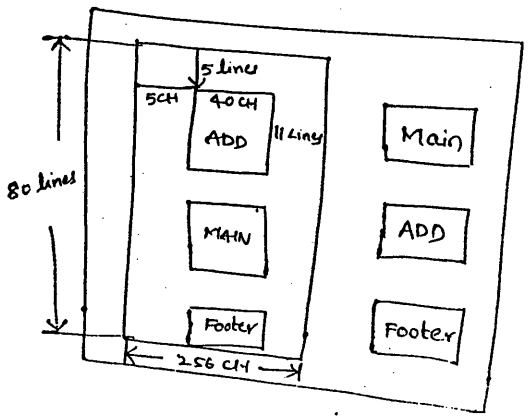
Description : Address window ↵

Once again Edit
↳ go for create element

window name : Footer

Description : ~~Address~~ Footer window. ↵

go with page windows. ~~windows~~ option in toolbar



go for Edit
↳ Create element

Let us Double click on Address window

provide left margin : 5 CH (characters)

Upper margin : 5 LN (Lines)

window width : 40 CH

" length : 11 LN

Once

In this window it will not display Add window -1 already we selected that go for main window

Let us double click on main window.

provide left margin : 5 CH

Upper margin : 15 LN (Add window occupy 11 lines 11+5(upper margin) = 16)

window width : 40 CH

" height : 11 LN

once again Edit

↳ create element

Here we can see the option Main window because form can use upto 99 main windows.

Let us double click on Footer window.

provide left margin : 5 CH

Upper margin : 30 LN

window width = 40 CH

" height : 11 LN

Ctrl+S.

go with paragraph formats

provide paragraph identity : P1 (we can go with any identity)

Description : Default paragraph.

Same window we have option FONTS.

With paragraph font format user can maintain different font apart from default.

go with character format option (Toolbar)

provide identity : C1 (go with any).

Description : Default char.

Using character format we can maintain different format for particular paragraph apart from default.

Ctrl+S

F3

select subobjects Header

go with basic settings

provide Default paragraph : P1 (which we defined)

provide first page : page 1

Ctrl+S

Ctrl+F3

go for utilities
↳ print preview

provide output device : LP01 (at this moment we dont have printer so use screen as o/p device)

go with printpreview

go with F3

select page windows

place select the control an Address window (use pageup, page down)

go with Edit
↳ text elements

go with goto
↳ change

select * in left side Tag window (small window)

Here * means Default setting if we create any paragraph setting we get that setting here we can select that by search help(F4)

Keep variable in window

customer number &ITAB-KUNNR& ↳ any field place b/w place holders means it is variables.

customer name &Itab-name1&

country &Itab-land1&

F3

select main window

go for Edit
↳ text elements

go for goto
↳ change

provide Tag is /E (indicates element) go for F4 and select
provide element : ELE1 (Element 1)

DEAR &ITAB-NAME1&

clear all your Dues before

&SY-DATUM&

In next page we can use same element.

F3

place control on Footer window

go for Edit
↳ Text elements

go with goto
↳ change

provide telephone no of a company

TEL: 08745-24644

F3

Ctrl+S

Ctrl+F3

* With Text elements user can define variables in a form.

PRINT PROGRAM DESIGNING: Function modules provided by SAP for print program are

1. Open_form : It initialises a form & Activates a form
   ↳ close_form.

2. start_form : It initialises form and it can open a form from specific page onwards.
   ↳ End_form.

3. write_form : For elements (Depends on elements that many times we have to call write_form.

Let us create executable program

Provide Tables workarea

Tables : KNA1.

Design selection screen with select options

Select-options : cust for KNA1-KUNNR.

Data : Begin of itab occurs 0,
Kunnr like KNA1-Kunnr,
Name1 like Kna1-Name1,
Land1 like Kna1-Land1,

*

need

Call function : open_Form ↵

exporting

Form : 'YJagan_Form' (which we created)

Language : SY-LANGU. (System field for language)

* Extract customer list

select kunnr name1 land1 from kna1 into table itab

where kunnr in cust.

Loop at itab.

Call function : start_form ↵

exporting

Form : 'YJagan_Form'.

Language : SY-LANGU

Start page 'PAGE1'.

Call function : write_Form

exporting

element : 'ELE1'

Function : 'SET'

Type : 'BODY'

window : 'MAIN' (element existing in main window)

Call function: End_Form ↵

Delete all Parameters

Endloop.

Callfunction : close_form ↵

Delete all Parameters.

Ctrl+S

Ctrl+F3

SAP SCRIPT MODIFICATIONS : Scripts is applicable for transactional data applications only.

T.code		Formname	printprogram
VA01	← Sales order appl	RVORDER01	RVADOR01
ME21	← purchase order appl	MEDRUCK	SAPFM06P
VL01	← Delivery appl	RVDELNOTE	RVADDN01
	Invoice appl	RVINVOICE	..

P.order application →	Form name	Print program
	MEDRUCK	SAFFM06P

DEMO ON P.order:

1. Copying a predefined form.

go with SE71

provide user defined form : YJMEDRUCK

go for create

go with Form
↳ Copy from

provide form : MEDRUCK (from which form we copying)

Ctrl+S

Ctrl+F3

2. Adding a logo in a form

File format is : TIF

program : RSTXLDMC

Let us create tif file by using painter and it can save in imaging with TIF format.

go with abap editor

provide prog ~~type~~ name : RSTXLDMC (For logos)

F8

provide logo file path : C:\Jagan.tif

Type : BCOL (Logo with multicolours)

Textname : ZHEX-MACRO-HEADE

↓ path for window

Delete * and provide Header i.e in which block we need logo

Text ID : ST (Identity for text name)

go with SE71

provide form name which we copied : YJMEDRUCK

select pagewindows under subobjects

go to the change mode

select Header window

edit
└→ Text elements

go with Insert Command option in toolbar

select ⊙ Command option

provide logic

Include ZHEX-MACRO- Header. Object Text Id ST↵

F3
Ctrl+S
Ctrl+F3 i.e Form
└→ Activate

go with print preview

3. Adding a field in a form

1. plant ID
2. material group

go with SE71

provide the form which we copied : YJmedruck

select page windows

go with change

under that select Info1 window use page up & page down

edit
└→ Text elements

goto
└→ change

left justify ← AS <S> Modifying layout </>

control stmnts ← /: IF &EKKO-LIFNR& > 1000

start with /: / plant ID is 100

close with /: /: EndIF

F3

Ctrl+S

Ctrl+F3

go with print preview

---

go with T-code NALE for modify SAP scripts

select appl EF (Purchase Order)

go with output types: Rn · twlb a1

select output type NEU (o/p type for P.order)

go with processing routines (letside)

* with processing routine user can assign a form to the print program.

go with change mode

select medium 1 indicate printer

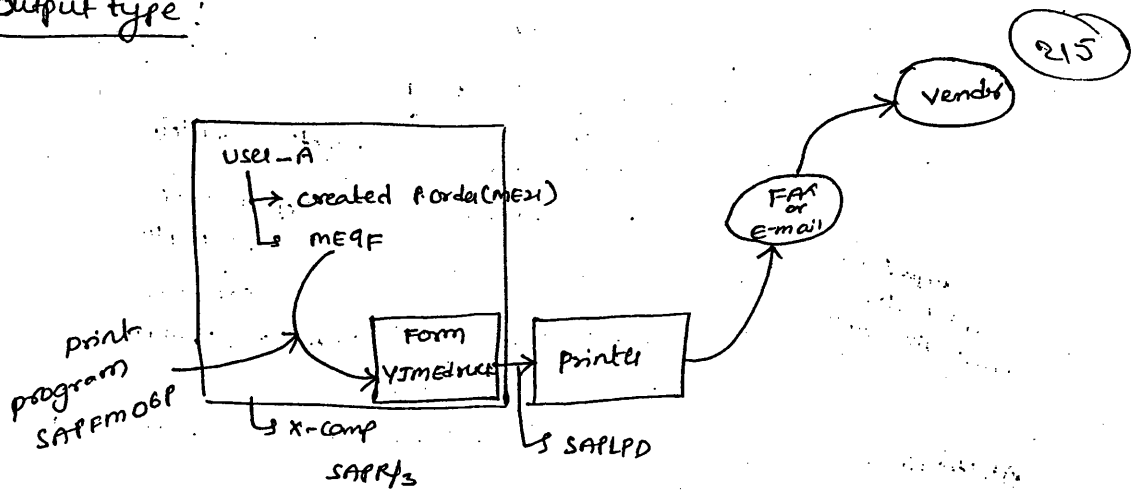
provide modified form name · YJmedruct

Ctrl+S

go for view

go for transport

Output type :



* With o/p types R/3 can identify the document which is forwarding to external system i.e printer.

o/p types provided by SAP are

No company use predefined o/p types Functional people will create o/p type

- NEU - P.Orders
- AF00 - Enquiry appl
- BA00 - Sales orders
- AN00 - Quation appl

TNAPR — db table for predefined o/p types.

go with MEQF t-code

provide P.order no which we created 4500004865

F8

select the entry ☑ NEU

go for display message (Toolbar)

ABAP Report

* Object oriented prog

1. ABSTRACTION → Function modules (Function group)
2. Encapsulation → Report ZERE
3. Inheritence → Report ZERE Int. table. ITAB Itab like ITAB access with header line
4. polymorphism → Int. table + SQL command → FWAM(10), FVAL(10)

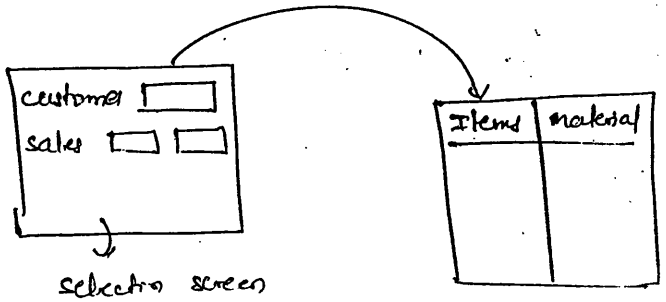
* Abstraction can restrict implementation to the real-world
* Encapsulation combines data and functionality together.
* Inherentence with new objects can be ~~deffo~~ derived based on existing objects.
* polymorphism with identical objects can behave different manner in different locations.
* class is a collection of objects

class
↳ Global
(or)
Local ~~(class builder)~~

* while creating class builder globally user has to work with class builder T-code for class builder SE24 classes which defined in class builder will store centrally in a class pool
* Local classes can be design within abap program
* Object is a runtime entity

Public class : it can access within the class across the class
private : " " " " .
protected : " " " " specific user

DEMO: Global class



Line type and row type creation

go with abap dictionary

select datatype : YJ linetype

go for create

select structure ↵

provide short text : Line type

provide o/p fields

POSNR	POSNR_VA
MATNR	MATNR.

Ctrl+S
Ctrl+F3

go with abap dictionary select datatype

provide Rowtype name: YJ Rowtype

go for create

select table type ↵

Short text : Row type

Row type : YJ Linetype. (provide linetype name)

Ctrl+S
Ctrl+F3

go with SE80 (Repository browser)

go with edit object appl toolbar

select Dictnry and Typegroup : ZVCTG ↵

go for create

provide shorttext : Type group ↵

Types: ZVCTG_WA type YJlinetype,
ZVCTG_ITAB type YJrowtype.

Ctrl+S
Ctrl+F3

go with class builder SE24

provide class name : ZVCLASS1

go for Create

Select the class option ↵

provide description : class program

Select public ↵

Ctrl+S

go with properties

provide type group / object type

ZVCTG

go with attributes

attributes	level	visibility	Reference type	
CUST	Instance	public	KNA1-KUNNR	
sales_low	"	"	VBAK-VBELN	
sales_high	"	"	"	
WA	"	"	ZVCTG_WA	workarea
ITAB	"	"	ZVCTG_ITAB	body.

go with Methods

	level	visi	
Constructor	Instan	public	Des Constructor (cbar)
Select_data	"	"	Select data
Display_data	"	"	Display data

it can initialize the objects

for select statement

Loop write endloop

Select Constructor method go with parameters

Import parameters

	Ref type
I_CUST	KNA1-KUNNR
I sales_low	VBAK-VBELN
I_Sales_high	"

Double click on constructor ↵

which ever parameters define in the method initialise those params.

go with method and endmethod

cust = I_cust.

Sales_low = I_sales_low,

sales_high = I sales_High.

Endmethod.

P3

Double click on display data

Keep loop & endloop

Loop at itab into wa.

write: / WA-POSNR, WA-MATNR.

Endloop.

F3

Double click on select data

provide select statement

Select VBAP~POSNR VBAP~MATNR into table itab from VBAK

Inner join VBAP on VBAK~VBELN = VBAP~VBELN where

VBAK~KUNNR = cust and VBAK~VBELN between sales_low and sales_high.

P3

Ctrl+F3

go with abap editor

create executable program.

tables: KNA1, VBAK.

parameters: cust like VBAK-KUNNR.

Select-options: sales for VBAK-VBELN.

Create object with reference of class which we define

Data: ITEMS type Ref to zvcl_a1.

↳ keyword to refer a class.

Provide start-of-selection.

Select abap objects patterns ↵

Select the method create object

Instance : ITEMS

object type : zvclass1 ↵

I_cust = cust

I_sales_low = sales-low

I_sales_high = sales-High.

go with pattern

Select abap object pattern ↵

Select call method

Instance : ITEMS

class : zvclass1

method : select_data ↵

Repeat same navigation for display_data.

Ctrl+S

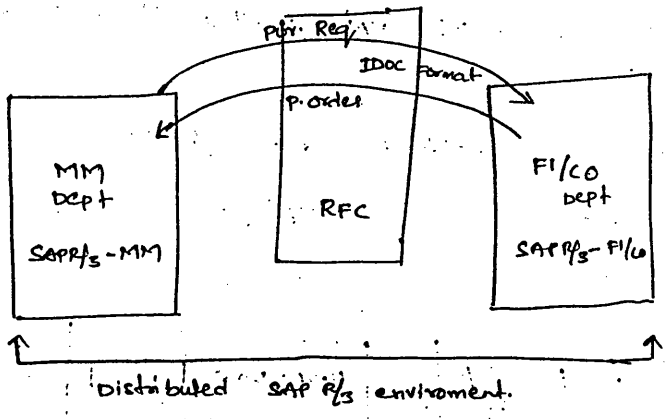
Ctrl+F3

22/8/05

CROSS APPLICATIONS

Cross applications also called as Advanced ABAP.

Cross applications for distributed environment.



IDOC is Intermediate Document.

IDOC is data carrier across distributed SAP R/3 systems. IDOC's provides better security than flat files.

* Communication layer is designed with RFC concept

1. Distribution layer: MM Module some where and FI/CO some where.
2. Application layer: For distributed application
3. Communication layer: RFC layer.

1. ALE : Application Linking and Enable. (All systems are SAP R/3)
2. EDI : Electronic data interchange. (R/3 to non SAP R/3)

With ALE Data can transfer across distributed SAP R/3 systems using IDOC format.

With EDI data can transfer across distributed R/3 to non SAP systems.

ALE : IDOC Architecture:

Control Record.
Data. Record.

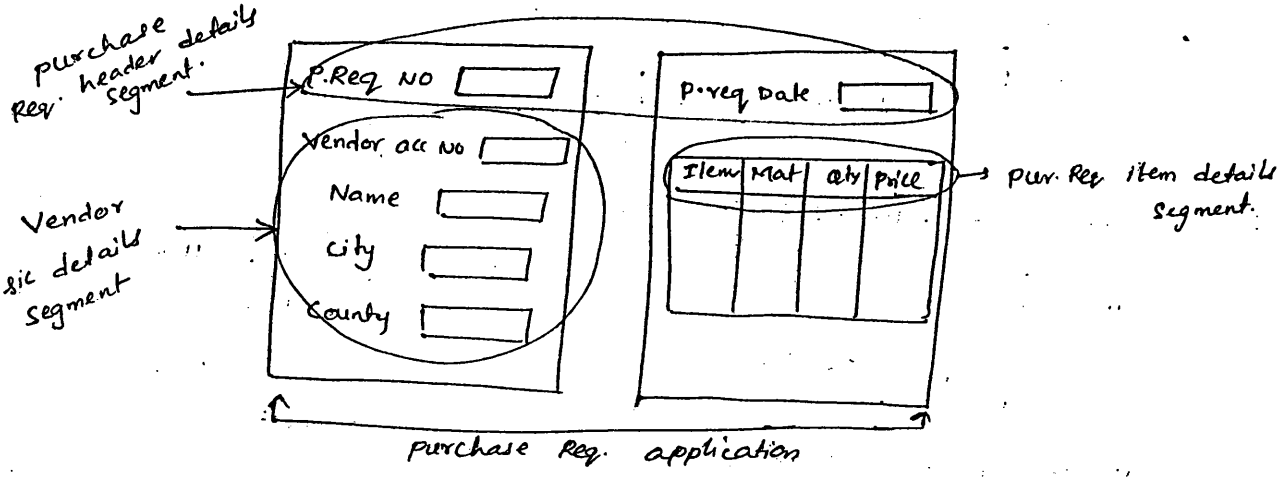
Receiver is inbound system

Sender is outbound system.

Control Record: Control Record have IP address of Inbound R/3 system.

Control Record can hold application document details also i.e application belongs to which dept weather it is Vendor or customer etc.

Data Record: IDOC will hold the data based on data Record. i.e which data it can carrying.



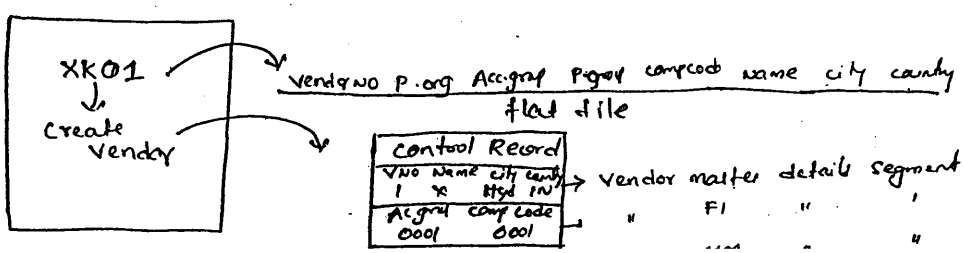
* Data Record is collection of segments.
* Segment is a collection of fields.
* If segments start with E or E1 are predefined segments.
* If segments start with E are client independent Version independent.
* If segments start with E1 are " dependent " dependent.
* If segments start with Z1 are user defined segments.

T.code for segments is : WE31.

Status Record: provides the status of IDOC i.e weather IDOC have any errors or not.

105

Difference b/w Flat file and IDOC.



* Flat file data any body can change or manupulate so it is not providing security.

223

* IDoc will not allow you to manupulate the records even super user also can't change, so IDoc provides security.

IDoc customizing :-

① Message types: are the identities for sap applications.

* with Message types message can be flow across distributed R/3 systems.
* Message types is the identity for application.
* control Record can hold the message types with IP address.

Message Types for Material Master application – MATMAS

" Vendor " " – CREMAS

" customer " " – DEBMAS

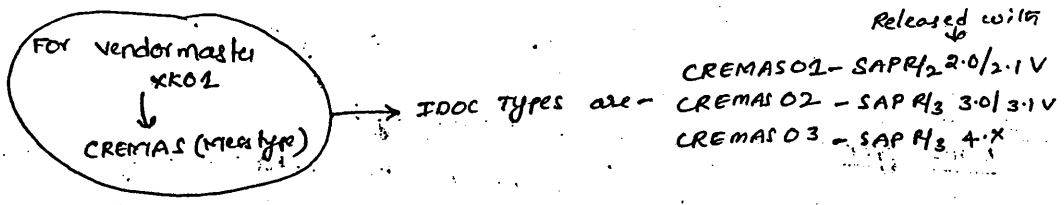
" Purchase order " " – ORDERS

" sales order " " – ORDRSP

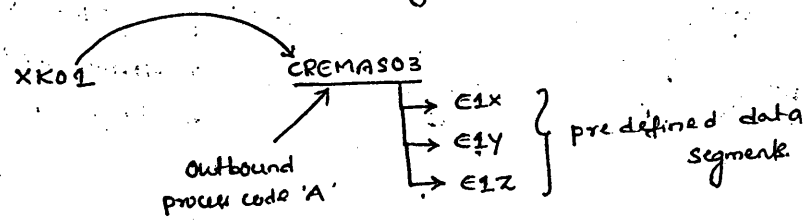
" G/L account " " – GLMAST

For creating new message types and checking message types transaction code is WE81.

⑤ IDoc Type: IDoc is an instance of IDoc type.



IDoc type is collection of segments.



process codes: process code will process the data by using function modules and the final output is IDOC.

WE41 is the T.code for outbound process code.

Both the sides we have process codes:

Outbound process code (outbound syst as sender)

Inbound process code (Inbound syst as Receiver)

Under process code SAP defined logic for segments avilable in IDOC Type (A)

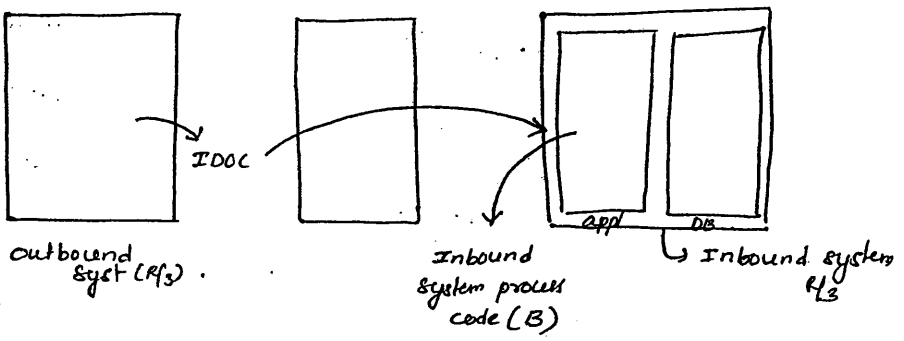
↳ For example.

Logic is in the form of function modules Function module Type is

IDOC_OUTPUT_<IDOCTYPE>

↳ CREMAS03 (for vendor)

* Inbound system process code:

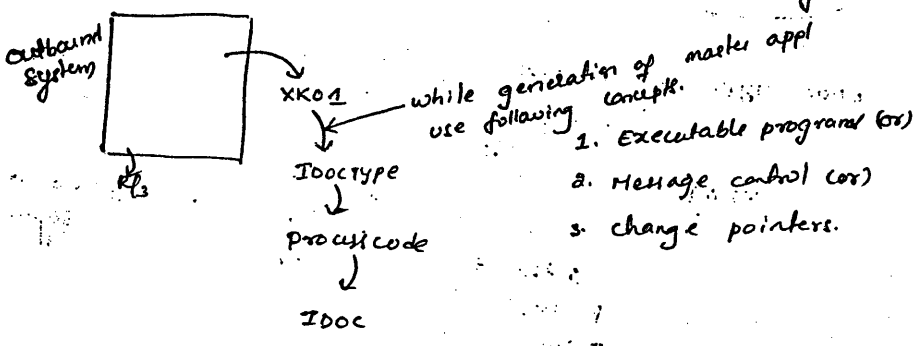


Inbound system process code will process the IDOC with function module. i.e Transfered IDOC is not able to move directly to db process code will process that.

WE42 T.code for Inbound system process code.

IDOC generation steps in outbound systems:-

process code will process the data and genereater the IDOC.



* While generating IDOC's for master data applications its concepts are Executable programs (or) change pointers. (or) message control

Ex: XK01.

225

* Using message control IDOC's can be generated for transactional data applications.

Executable programs: T-codes for executable programs is

BD10 – send Material
BD11 – get "
BD12 – send Customer
BD13 – get "
BD14 – send vendor
BD15 – get "

Change pointers: Transaction codes are

BD50 → Activating Message Type.
BD52 → To check for the fields which changes will be effected.
BD61 → For activating change pointers.

RBDMIDOC is executable program for change pointers.

* By using change pointers Any change in outbound system IDOC will effect on Inbound System automatically but the changing fields will must contains in BD52 then only inbound syst will effected.

* By using change pointers IDOC can be transfered and process will be automatically. i.e no need of user Interaction in Inbound syst.

Disadvantage of executable programs:

Both the sides i.e outbound and Inbound sides we have to execute the programs for respective applications by using T-codes of it. for example vendor (XK01) in outbound system BD14 will be executed and in Inbound BD15 will be executed

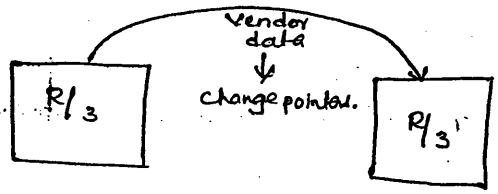
Master data application

↓

Executable / change pointers.

↓ BD14 (outbound)
BD15 (inbound)

↓ → SMD Tool
BD50
BD52
BD61
RBDMIDOC.



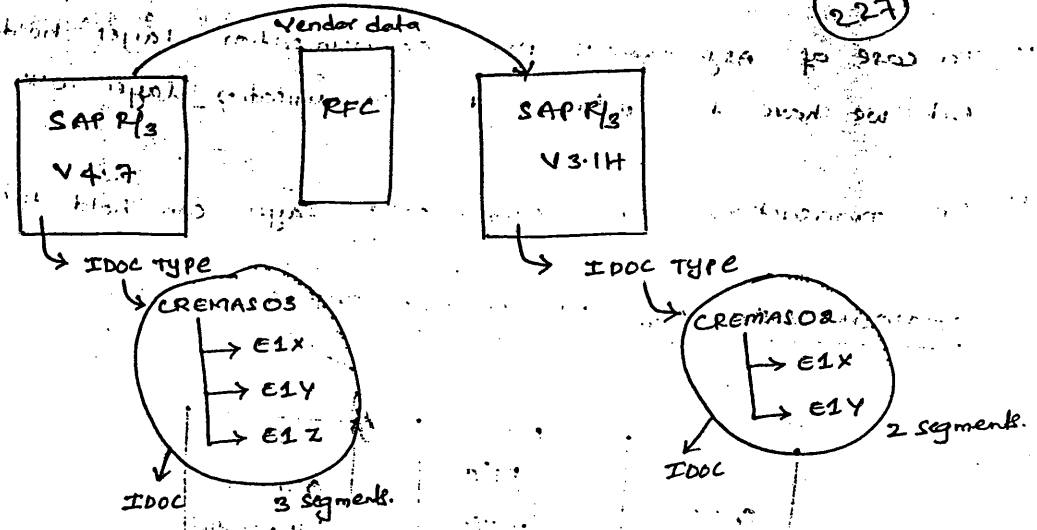
RBDMIDOC can identify data which distributed and which was not distributed. i.e outbound system have 100 records on which some of records already sended now we have to send remaining records so RBDMIDOC can identify it which was not sended.

SMD TOOL : SHARED MASTER DATA TOOL which can work behind the change pointers.

Difference b/w Executable & change pointers:

Executable	change pointers
1. In Inbound system we have to execute the program.	1. NO user Interaction is required in Inbound system.
2. Any change in IDOC will not effect automatically on Inbound again we have to process the IDOC	2. changes will effect automatically.
3. Which cannot Identify the Records which was distributed and which was not distributed.	3. which can find which was distributed and then send which was not distributed.
4. NOT automatical.	4. IDOC process and transfer automatically.

Communication with two different versions:



If communication b/w two different versions then first check the IDOC type and segments. If both version segments are same then we can transfer if different i.e 3 segments in V4.7 and 2 segments in V3.1H then user have to drop or add a segment to IDOC types and make it same no. of segments.

Whenever user wants to drop a segment from existing IDOC type the concept type is IDOC TYPE REDUCTION (or) IDOC TYPE VIEWS.

With segment filtering segments can be droped from existing IDOC type.

Whenever user wants to add a ~~field~~ segments to the existing IDOC types concept is IDOC TYPE EXTENSION.

COMMUNICATION LAYER: With RFC concept communication layer will be designed.

RFC Types : 1. Synchronous RFC. } outdated.
2. Asynchronous RFC. }
3. Transactional RFC.

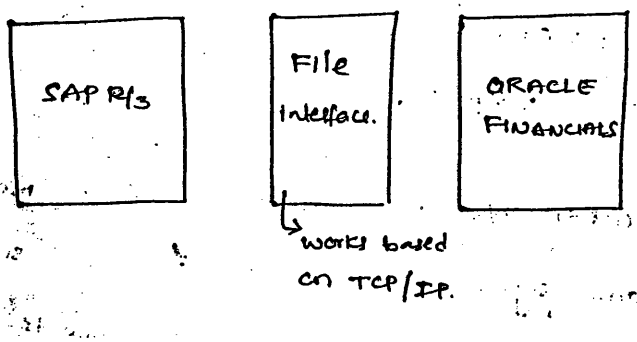
* With synchronous RFC IDOC forwarded to inbound system while it is in maintenance or not ready to accept IDOC then it will comes

Synchronous RFC doesn't hold the IDOC temporarily.

* In case of Asynchronous RFC communication layer holds the IDOC temporarily but we have to customize the communication layer with Asyn. RFC.

* In transactional RFC communication layer can hold the IDOC by default.

COMMUNICATION B/W R/3 TO NON SAP :-



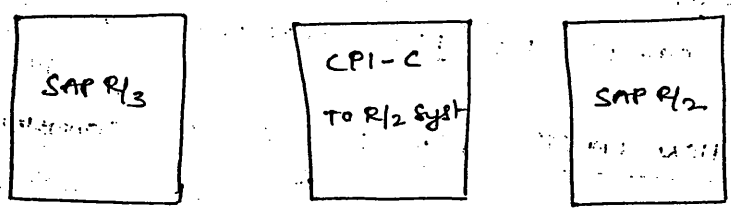
* Using file interface communication can be design across R/3 to NON SAP systems.

* In case of ALE communication can be design either by using File interface or by using transactional RFC.

* In case of EDI communication can be design with file interface only.

25/8/10-

Communication b/w SAP R/3 to SAP R/2 :-



1. CPI-C : can provide communication b/w R/3 to R/2 systems.

CPI-C : communication programming interfaces - Communication.

UNDER IDOC customizing :-

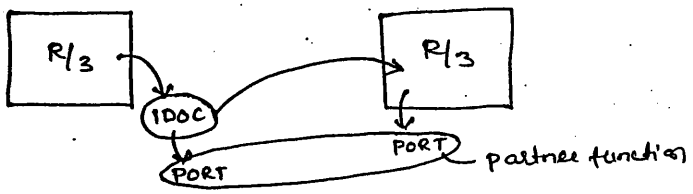
1. customer distribution model (CDM)

T. code : BD64

From port to IDOC will transfer to inbound.

CDM can distribute the data across distributed R/3 systems. using Respective message type.

229



CDM can hold sender IP address

Receiver IP address

Message type.

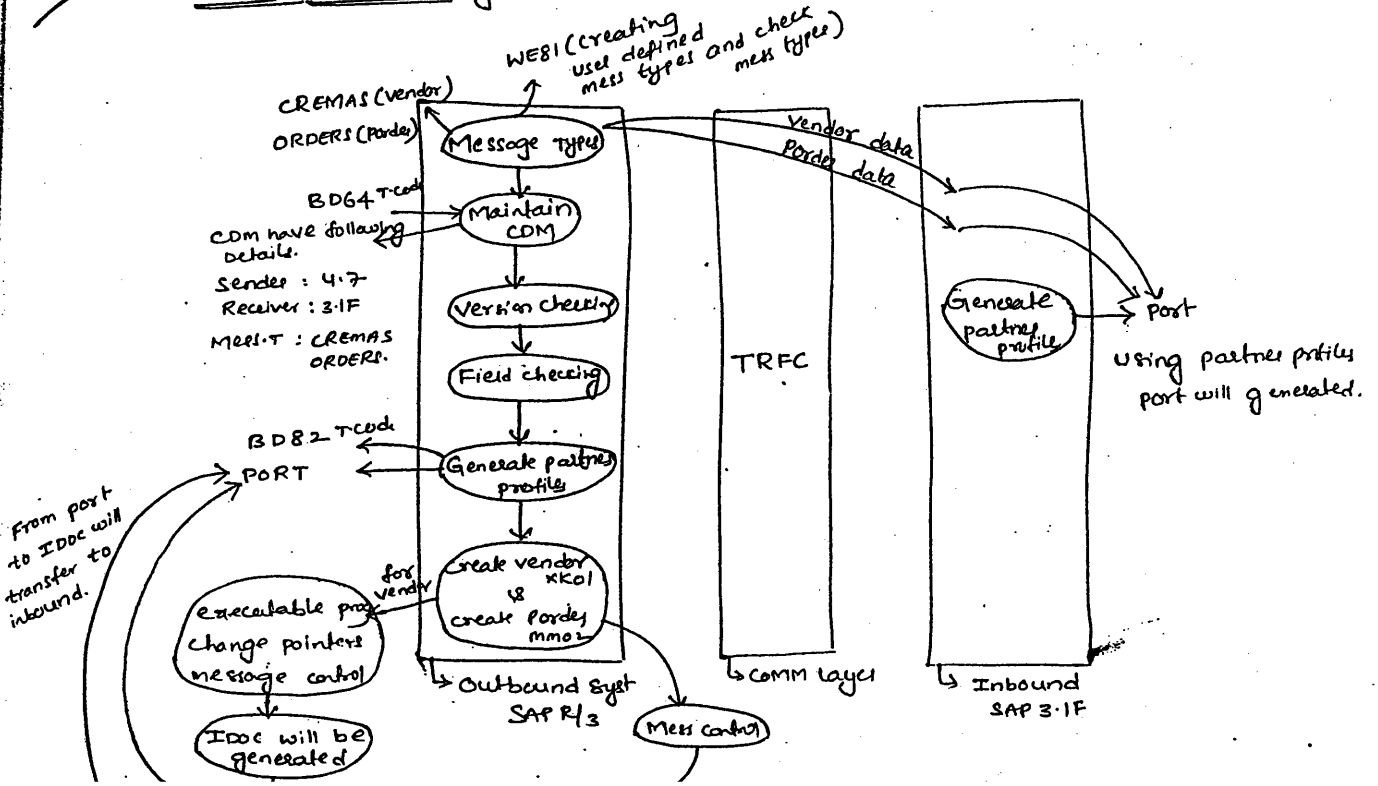
Based on CDM control Record can maintain in IDOC.

By default CDM is client dependent and version dependent by distributing CDM it becomes client independent and version Independent.

* With partner profiles ports can be generated for distributed R/3 Systems.
* Based on IP address we can generate ports.
* partner profiles execute based on CDM.

25/8/05

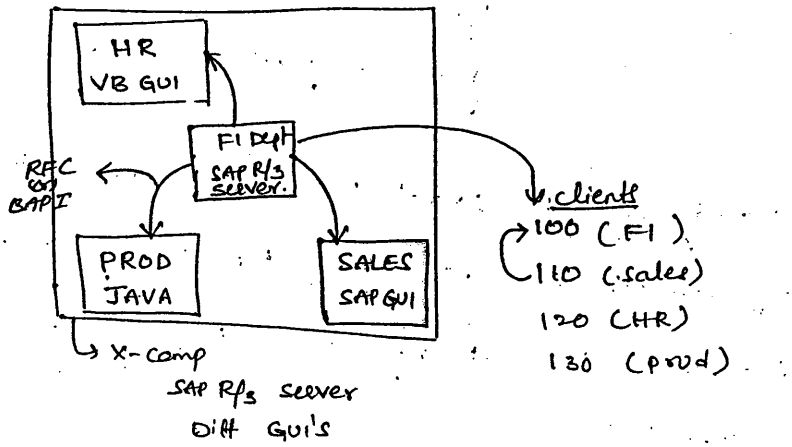
ALE Customizing flow :



* Based on version checking and field checking idoc type reduction extention will be customized.

* outbound system last stage for IDOC is port and in inbound system first stage is port.

Business scenario:



Centralized distributed environment : with in the system across the clients i.e single server with different GUI's

Decentralized distributed environment: with different depatment with different systems or severs.

ALE customizing steps:

SALE is a T-code for ALE customizing

ALE
↓
SALE
↓

① Define logical systems.

Logical systems are client identities.

Logical systems are client independent & version dependent.

Define logical systems are LS100, LS110.

② Assign logical system to respective clients.

100 – LS100.

110 – LS110.

⑧ Maintain RFC destination

provides communication b/w 100 & 110.

⑨ Maintain customer distribution model (CDM)

⑤ Generate partner profiles.

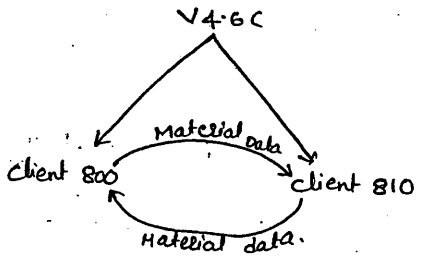
26/8/05

ALE DEMO : Based on centralized distribution (In single server)

↓

on executable programs

↓



STEPS: 1. Define logical systems. (Define both the clients in client-800 side because we are working under centralized means client independent.)

LS 800
LS 810

2. Assign clients to logical systems. (In client 800)

800 → LS800.
810 → LS810.

3. Maintain RFC Destination. (In client 800)

From LS800 ⟶ LS810 (Forwarding data)

Maintain RFC Destination for Receiving data from 810 (In clien 810)

From LS 810 ⟶ LS800 (Reverse data)

④ Maintain customer Distribution model (CDM) In client 800 side

Name of CDM : LSModel

provide sender : LS800

Receiver : LS 810

Message type : MATMAS

Maintain sender, Receiver details in Reverse so it makes distributed

sender : LS810

in client 800.

⑥ Distribute CDM for making client independent then only client 810 can be able to generate port.

client 810 generate partner profiles.

⑦ create material by using T.code MM01 in client 800.

⑧ Based on material execute BD10 for send material T.code in 800.

⑨ Let us check IDOC status by

go with IDOC lists in WE05 T.code.

⑩ Execute BD11 in client 810 for get material T.code.

After executing BD11 T.code Inbound syst will Receive the material IDOC.

⑪ Let us check IDOC status in client 810

go with IDOC lists → WE05 T.code.

⑬ If we find any errors in IDOC weather Inbound side or outbound side for rectifying errors we can use ALE TESTING TOOL

T.code is WE19.

DEMO :

start with client 800 then

go with T.code 'SALE' for ALE customizing.

go with Sending & Receiving systems

→ Logical systems

→ Define logical systems.

F8

it will display a message cross client i.e client independent.

↵

Let us go for New Entries

Under New entries define Logical systems for client 800, 810.

JAGANLS800 : Logical System for client 800

JAGANLS810 : " " " " 810

F3
F3

go with Assign client to logical system option

F8 ↵

Let us Double click on client 800

Assign Logical system which we defined for this client

Logical system : JAGANLS800

Ctrl+S

F3

Double click on client 810

Assign Logical system which we defined for this client.

Logical system : JaganLS810.

Ctrl+S

go for SM59 T.code in client 800 for maintain RFC Destination.

go for create.

provide the Destination : JaganLS810 (Logical syst of inbound)

Connection Type : 3 (Communication b/w R/3 Systems)

provide Description : RFC b/w 800 and 810.

provide Logon details in same window i.e

client 810 Logon details :

Language : EN
client : 810
User : SAPUSER
Password : ABAP. ↵

provide the target host : CLASSROOM (i.e system id or server name

Ctrl+S

go with client 810

go with T.code SM59 for RFC Maintain in client 810.

go for create

provide Destination : JaganLS800.

connection Type : 3 (B/w R/3 Systems)

Provide Description : RFC b/w 810 and 800.

provide logon details ... client ...

Language : EN

client : 800

user : SAPUSER

PW : ABAP. ↵

provide Target host : CLASSROOM (server name i.e or system id)

Ctrl+S

go with client 800

go with BD64 T-code to maintain CDM.

go for change mode option in Toolbar.

go for create model view option in toolbar.

provide short text : Model for executable program.

provide Technicalname : JaganModel

↳ CDM name ↳ go with anyname.

↵

select CDM in that list provided by window i.e window displays all CDM's then select our CDM in that list (get in last position)

go with Add message type option in toolbar

provide sender : JaganLS800

Receiver : JaganLS810

messagetype : MATMAS. ↵

Add Message type once again for client 810 make client independ.

Sender : JaganLS810

Receiver : JaganLS800

messagetype : MATMAS ↵

Ctrl+S

go with environment option in menubar

↳ Generate partner profiles.

provide partner system i.e Inbound logical system

: JaganLS810

F3
F3

go for edit option in menubar
→ Model view
→ Distributed ↵

In this Demo we are defining CDM for both the clients in client 800 so we can distribute the CDM to client 810 then we can generate partner profiles in client 810.

go for client 810

go with T.cod BD82 for generate partner profiles.

provide Model view (i.e CDM name) : Jaganmodel.

provide partner system : JaganLS800 (Inbound syst i.e from client 810 client 800 is inbound

F8

go for client 800.

go with T.code MM01 for create Material which we going to transfer.

provide Material : Jaganmat

Industry sector : Mechanical Eng

Material type : Finished product ↵

select Basic data 1

Basic data 2 ↵

Here we can select acording to our requirement

provide Description for material : Material for exe programs

Units of measure : Kg.

gross weight : 75

Net weight : 75

weight unit : Kg.

ctrl+s.

go for BD10 T.code for send material

provide Material name which we created just now

* Master IDOC will exists in application server.
* communication IDOC will exists in DB server.

go for WE05 for checking the IDOC list i.e status checking

F8

Let us double click on message type

it will display IDOC Architecture, there we can check the status

Under IDOC status Record status code from 01 to 49 related to outbound system.

status code 50-75 related to Inbound syst.

If IDOC generated successfully. then

go for client 810.

go with BD11 T.code for get material

provide material name : Jagan mat.

provide message type : MATMAS.

F8

go for WE05 for status checking of IDOC

F8

Let us Double click on message type 59 — MATMAS
↳ status code.

check the status here status is error because Function module is in in active mode so

Note the IDOC No : 77001

go for WE19 for ALE Testing tool

provide IDOC : 77001 (IDOC no which we noted)

F8

go with IDOC option in menubar

→ Inbound Idoc

→ Inbound function module.

provide the function module :

IDOC_INPUT_MATMAS01. ↵

Go for MM02 for checcing weather material received or not

provide Material name : Jaganmat. ↵

Select views which selected in entering material ↵

It will display a material which we ~~cost~~ created in 800.

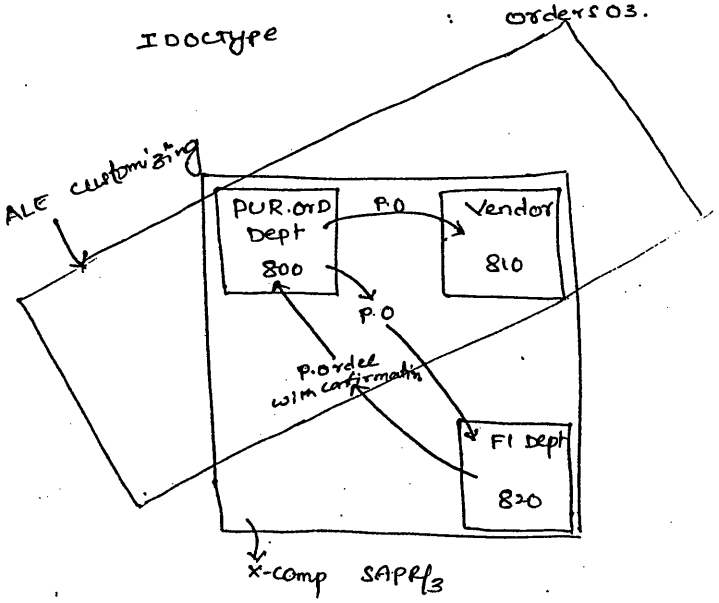
and also we can create material in client 810 and check weather transferred or not.

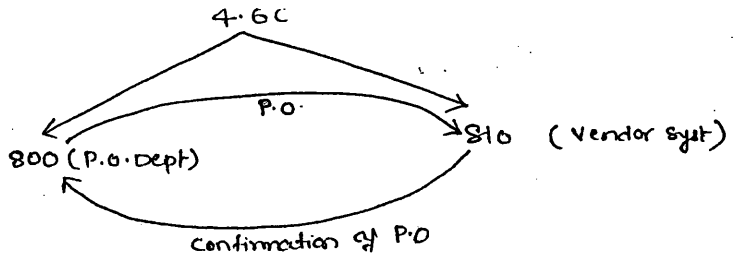
28/8/05

ALE DEMO WITH MESSAGE CONTROL : Centralized distributed and based on Transactional data i.e purchase order data

Message Type is : orders

IDOCtype : orders03.





Under 800

1. Define logical systems. LS800, LS810.
2. Assign logical systems to clients. 800 – LS800, 810 → LS810.
3. RFC maintanence for destination. LS800 → LS810.

Under 810 :

4. Maintain RFC for destination i.e LS810 → LS800.

Under 800 :

5. Maintain CDM (Customer distribution model)

   Sender : LS800
   Receiver : LS810
   Messtype : Orders ↵
   Sender : LS810
   Receiver : LS800
   messtype : Orders ↵

6. Generate partner profiles. (partner profiles dependent on CDM)
7. Distribute CDM.

Under 810 :

8. Generate partner profile.

essage control
keys :→ 9. Define partner profiles (is independent from CDM)

T.code : WE20

* Generate partner profiles is dependent on CDM where as Define partner profiles is Independent from CDM.
* Under define partner profiles SAP can list out business partners working under distributed.

We can go with any one of above. either Generate

Under 800 :

10. Maintain condition record.

T. code : MN05

Condition record provides medium of Transaction.

11. ME22 T. code P. order change.

After this step with proper customizing IDOC will be generated.

Check IDOC list for status using WE05

Distribute CDM

go for WE19 for Error correction using ALE testing tool.

DEMO :

Under client 800 :-

Go for SALE T. code

go with Sending & Receiving systems

→ Logical systems

→ Define logical systems.

F8

go with new entries

Define logical systems. Jagan800, Jagan810.

Ctrl+S

F3

F3

go with assign client to logical system

F8

Double click on client 800

Assign the logical system which we defined for that client

Ctrl+S

F3

Double click on client 810

Assign client which we define for that client

Ctrl+S

go for create

provide destination : Jagan810

connection type : 3

Description : RFC b/w 800 & 810.

Logon details of destination.

Language : EN
client : 810
user : SAPUSER
PW : ABAP. ↵

provide the target host : CLASSROOM (system id or server name)
Ctrl+S.

under 810:

go for SM59 for RFC destination maintanence.

go for create

provide destination : Jagan800.

Connection Type : 3

Description : RFC b/w 810 & 800.

Logon details on destination.

Language : EN
client : 800
user : SAPUSER
PW : ABAP.. ↵

provide Target host : classroom (system id)
Ctrl+S

go with client 800 :-

go with BD64 T-code for CDM maintanence

go to changmode

go for create modelview option toolbar
↳ CDM name

provide description : model for Purdue appl

select the CDM go with add message type option in toolbar.

provide sender : Jagan800

Receiver : Jagan810

messagetype : orders ↵

(241)

once again add message type

sender : Jagan810

Receiver : Jagan800

messagetype : orders ↵

Ctrl+S

F3

go with environment and generate partner profiles

provide partner system : Jagan810

F8

go for edit

↳ Modelview

↳ distribute ↵

go with client 810 :

go for BD82 generate partner profiles

provide CDM : Jaganmodel

partner system : Jagan800

F8

go with client 800 :

Define partner profiles go for WE20

Create partner: select partner type : LI

go for create

provide partner no : 100 (vendor no. which we are keeping distributed.

partner type : LI (vendor).

provide details of following

Type : US (USER)

Agent : SAPUSER (USERID)

Language : EN.

go for Create outbound parameters i.e

control under that we have option create outbound parameter

provide partner function : VN (vendor)

message type : ORDERS (Resp message type)

provide Receiver port : A000000025 (Keep SID port ID BD82 have this ID)

under outputmode:

Select option : Transfer IDOC Immediately

i.e whenever IDOC is genereated it will transfer immediately and we have one more option in this i.e collect IDOC and transfer i.e it will wait for IDOC packets NO which provided for IDOC limit then IDOC will reach that no then it will transfer to inbound EX: Daily P.O.

provide Basic type : ORDERS03 (IDOC Type)

Ctrl+S

go with message control option

go for Insert line option in button of window

provide application : EF (purchase order)

message type : NEU (O/p message type for purchase order)

* Using O/p types R/3 can identify the document which it is forwarding.

process code : ME10 (outbound process code)

go with WE41 / WE42 for check process codes.

Ctrl+S

go with MN05 for Maintain Condition Record.

provide output type : NEU ↵

↵

Let us provide under which Purchase organization

F8

provide vender NO : 100

partner function : VN (Vender)

partner no : 100 (VNO)

Medium of transmission : A (Distribute ALE)

Time : 4 (send immediately)

Language : EN.

Ctrl+S

go with ME22

provide one existing p.ordes which we created for vendor : 4500005016

go with Header (Menubar)
↳ Messages

provide the details

output : NEU

Medium of transmission : Distribution ALE

partner function : VN

partner NO : 100

Language : EN

Ctrl+S

IDOC will be generated

go for WE05 for IDOC status
F8

if status is OK

go for client 810.

check IDOC list WE05
F8

go for ALE Testing tool for rectifying error.

29/8/10

ALE DEMO

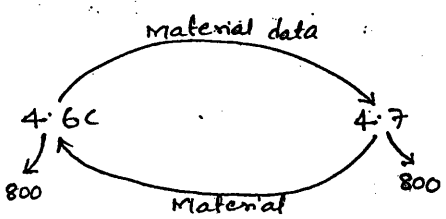
↳ Decentralized

→ BD50 - Activating message type

→ BD52 - To check the fields

→ BD61 - Activate change pointers.

→ Exe. prog : RBDMIDOC.



1. Define logical system LS46C800
   LS47800.

2. Define Logical system LS47800
   LS46C800.

* while working with different versions define all logical systems in both the sides.

3. Assign client to logical system
   800 → LS46C800

4. Assign client to logical systems.
   800 → LS47800.

5. Maintain RFC destination

6. RFC destination.

7. Maintain CDM

8. Generate partner profiles.

9. Distribute CDM

10. Generate partner profiles

11. Let us customize BD50
    BD52
    BD61

12. Create material MM01

13. Execute RBDMIDOC

14. Check IDOC status WE05

15. Check IDOC status WE05

16. WE19 for error correction.

DEMO:

Under 4.6C:

go with SALE T-code

go for define logical systems under sending & Receiving Systems.

F8

go with new entries

Define logical systems : Jagan46c for 4.6 version

Jagan47 for 4.7 version (Reference purpose)

ctrl+s

F3

F3

go with assign client to logical system

800 → Jagan46c

go with SM59

go for create

provide destination : Jagan47

connection type : 3

Description : Jagan46c to Jagan47

Logon details of destination (Jagan47 details)

Language : EN

client : 800

user : SAPUSER

pw : abap ↵

Target system = 471DES (Destination system id)

ctrl+s

* Repeat same steps in 4.7 side using LS : Jagan47

go with 4.6C:

go with BD64 T-code

go for create modelview

provide short text : Model view for Jagan

provide Technical name : Jaganmodel ↵

select model and go with message type

provide sender : Jagan46c

Receiver : Jagan47

mess type : matmas ↵

again go with message type for 47

sender : Jagan47

Receiver : Jagan46c

mes type : matmas ↵

Ctrl+S

go for environment and generate partner profile

provide partner system : Jagan47

F8

F3

F3

go with edit for distribute com

→ Modelview

→ distribute com ↵

go with V4.7 :

go with BD82 T-code

select message type provide modelview : Jaganmodel

partner system : Jagan46c

F8

go with 4.6V :

go with BD50 T-code

select message type MATMAS (in that list)

Enable checkbox for activate MATMAS.

Ctrl+S

go with BD52

provide message type which we are working for change

: MATMAS ↵

any fields then first check the Respective fields are existing in BD52 otherwise go for New entries and create a fields which we want to change

247

go with New entries for create fields.

go with BD61

Enable the checkbox

ctrl+s

* Before creating Material execute RBDMIDOC by using SE38 which can generate IDOCs which was not distributed before. Because change pointers identify which records are not distributed and which was distributed.

go with MM01 T-code for create material

provide material: Jagan material.

Industry sector: Mech. Eng

Material type: Finished goods ↵

Select views Basic data 1
Basic data 2 ↵

Description: Material for Demo

Unit of measure: kg

Gross weight: 100

Net weight: 100

weight unit: kg

ctrl+s

With same Navigation create 5 more Materials.

go with abab editer SE38

provide RBDMIDOC

F8

provide message type: MATMAS

F8

go with WE05 for IDoc status

go with 4.7V:

go with WE05 for check IDoc status

If any error in IDoc, then go for

WE19 for ALE Testing tool.

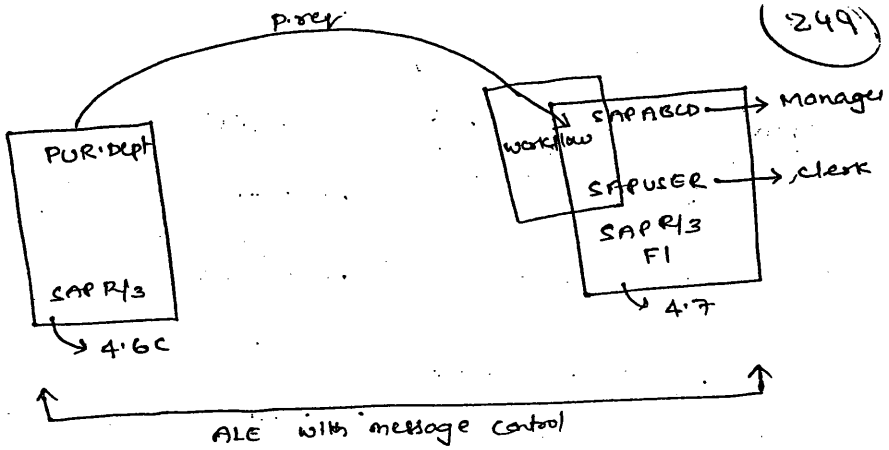
go with MM02 and check the material which we created.

Always go for change pointers, when we transfering master data.

go with WE21 for Define port manually.

30/8/05

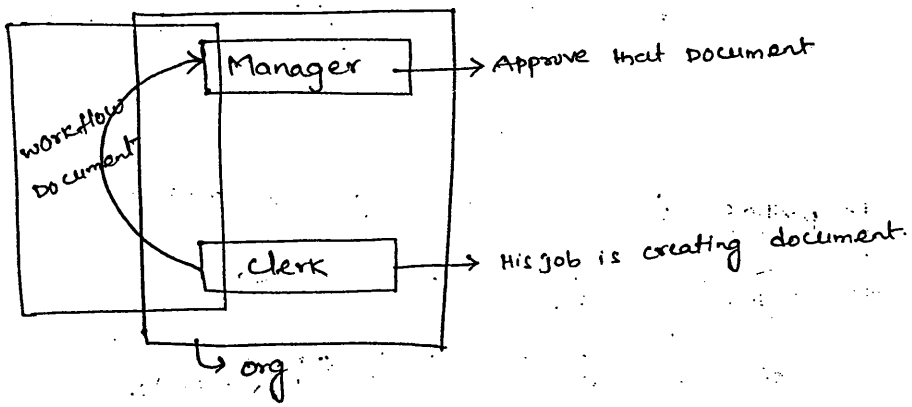
Workflow:



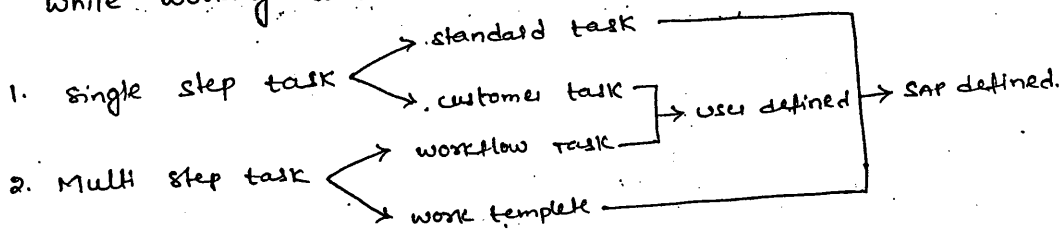
* Work flow is for error handling.
* Work flow is for Business Tracking.

In case of ALE if user come across errors it will be processed by using function modules.

In case of ALE Integration with workflow errors can be processed using workflow instead of function modules.



Tasks: while working with workflow tasks can be used



Multi step task is nothing but a collection of single step task.

Single step task is nothing but task perform by position (i.e clerk or manager)

User defined task is user dependent, version dependent

SAP defined task is user independent, version independent.

Create organization plan : PPOC_OLD

Create a task : PFTC_INS

change a task : PFTC_CHG

Business work place
↓
work flow inbox
&
work flow outbox
} : SBWP

Start work flow : SWUS.

Integration workflow with ALE & EDI → WE57

DEMO: go with PPOC_OLD T.code for create org plan

provide org. name : org for Jagan

provide description : Demo org

go for create option (Toolbar)

Create position: go with staff assignment (Toolbar)

go for create position.

go for create jobs
↳ activity.

Define jobs acording to org requirement

J_create clerk

J_approve manager

ctrl+s

ctrl+s

go for create position once again

provide abbrivation which we defined for clerk

: J_create

ctrl+s.

Assign user ids :

select clerk position

go with assign holder option in toolbar

holder type : US (user)

Name : SAPUSER (user id existing in system)

Ctrl+S

select manager

go with assign holder

Name : SAPABCD (existing in system)

Ctrl+S

select manager go with Edit → create chief position → create.

Ctrl+S

Manager position will get a cap symbol i.e chief position.

go for PFTC_INS T.code for create a task

select task type is : standard task (SAP defined single step task)

go for create option in toolbar.

provide abbrivation : CREATEDOC (workitem) → Runtime object

* work item generated at runtime whenever the task is performed.

* Based on workitem basis people can check for no. of times work is done by respective system for backup.

provide Name (Description) : Creating document for jagan.

work item text : Creating document for jagan

Release status : Released (always keep release then only task will be performed)

go with object method (same window)

* SAP is object oriented from V3.1F first technology after object oriented is BAPI

BAPI is a T.code for check business objects.

BOR for all business objects.

Object type : FORMABSENC (For leave appl. form)

Method : Create

Ctrl+S

go with additional data

→ Agent assignment

→ Maintain

select position

go for Agen assignment

→ Create.

provide abbrivation defined for clerk : J_create. ↵

F8

F3

Note the TASK ID : 97200611 (identity)

go for PFTC_INS for create task for Manager.

go for create

provide abbrivation : ApproveDoc

Name : Approving Document

work item text : Approving Document

Status : Released.

Object type : FormabsENC.

method : Approve

Ctrl+S

go with additional data

→ Agent assignment

→ Maintain

go for Agent Assignment

→ Create

select position ↵

provide abbrivation which we defined for manages

J_Approve ↵

go with SWUS T.code for start workflow.

provide task : TS97200611 (which we noted before)

F8

Let us fill the leave form

Name : Jagadeesh

Dept : HR dept

NO : 92844

Chk+s

Log on to manager user id

User : SAPABCD (existing user id)

PW : abap

go for T.code SBWP for business workplace.

go with workflow Inbox

select the document double click on that

select Approve option in toolbar

Approved document is forwarded to clerk inbox automatically

check the clerk inbox.

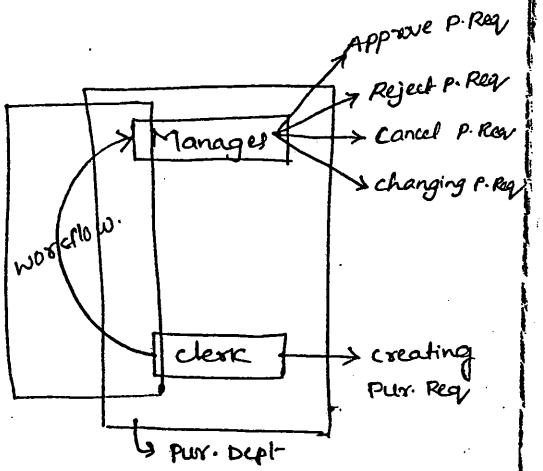
31/8/05

Work flow

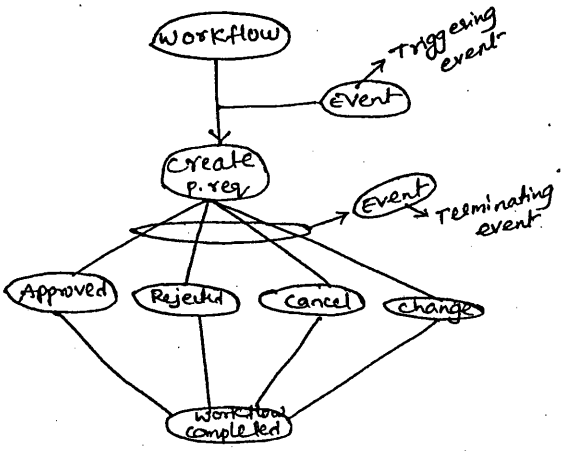
↓

Multi step task

Business object for pur. Req : BUS2009.



Multistep task :-



Events: Whenever events executes then tasks will be performed whenever tasks performed workitem will be generated.

* Triggering events starts the workflow and terminating event ends the workflow.

Let us go with PPOC_OLD. Code for create org.

provide org. name : Purchase org.

Description : purchase dept

go for create

go with staff assignments

go for create positions

go for create jobs

provide abbr. for jobs

M (Create) clerk

M (E+R+C+A) Manager

E - edit, R - Reject

C - cancel, A - Approve

ctrl+s

ctrl+s

go for create position again

provide abbr for which we defined for clerk : M (Create)

ctrl+s

select clerk position go with assign holder option in tool bar

Type : US

Name : SAPUSER

ctrl+s

select manager go with assign holders

Type : US

Name : SAPABCD (Provide existing userid)

ctrl+s

select manager edit → chief position → create.

Ctrl+s

* Under one organization user can provide maximum '9' chief positions.

go with PFTC_INS

Select standard task go for create.

Abbr : PREQ. cre.

Description: purchase Req create

work item text : Pur. Req create

Name : REQNO

Description : REQNO

Make it ⊙ Mandatory element option

Enable checkbox ☑ Import
☑ Export

Select object type : BUS2009 ↵

go with F3

Select Basic data

go with workflow builder

Select undefined block go with activity (First Middle bl

TASKNO : ~~TS97200613~~ TS97200613
↳ Standard task

Stepname : Req created (confirmation for that task)

go with generate binding ↵ (small red button bottom of that wi.

* Binding provides relation between tasks.

go with F3

select next undefined block (second middle blocks

go with activity

Task ID : TS97200614 (this id is belongs to approve task)

Stepname : Req approved

generate binding ↵

F3

Repeat same steps for remaining blocks for Reject, Cancel and change tasks. provide Respective ID's

After defining blocks we have to ~~define~~ delete undefin blocks from that window

go with Edit
↳ Delete
↳ Delete undefined blo

Same screen go with workflow container (left side bottom

Click on Release code (if it is mandatory) different windo

Ctrl+S
Ctrl+F3

go with SWUS for start workflow

provide workflow template no: WS97200085
↳ workflow template.

go with Input data.

* Input data can hold Input values using elements.

Let us provide existing reqno: 1000709 (go for search help and provide it)
item no: 0010 (F4) ↵

Ctrl+S
F8
Cancel the screen

Logon to manager userid: SAPABCD

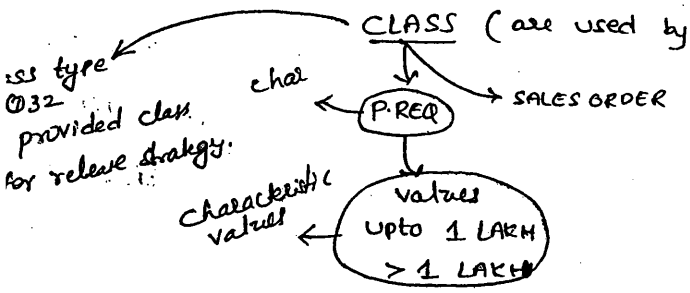
SBWP T.code for Business workplace

check workflow Inbox. for document.

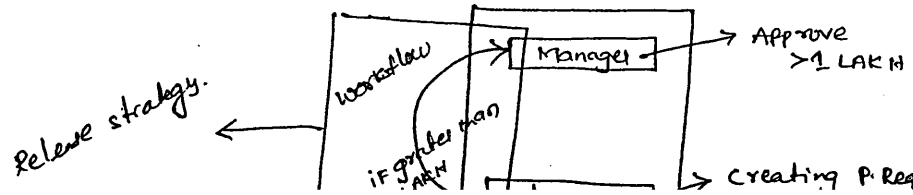
WORKFLOW
↓
RELEASE STRATEGY
↳ RELEASE CODES

* With Release strategy user can implement conditions on workflow.

Under one Release code we can able to define one condition



CLASS is a collection of appl under class each appl is a characteristic.



DEMO: Under SAPMENU go with Logistics
↳ central function
↳ classification
↳ master data
↳ characteristic

Double click on that characteristic

provide characteristic name : JaganPREQ

go for characteristic (in menu)
↳ create

provide description for characteristic : char for PREQ.

Select data type : character format

provide No. of Characters : 30

Under value assignment :

Select ⊙ multiple value

Let us go with values option (under same window)

Define the values

UP to 1 LAKH

> 1 LAKH

Ctrl+S

Class creation: go with F3

under same navigation select classes double click on that

provide classname : Jagan_class

class type : 032

go for class
↳ create

Description for class : PREQ for Demo

go with char. option in same window

provide characteristic which we defined
: JaganPREQ

Ctrl+S

go with SPRO T-code for customizing

go with Material management
→ purchasing
→ purchase Requisition
→ Release procedure
→ procedure with classification
→ set up procedure with classification

F8

I st step go with Release group:

Delete entries if there in window

go for new entries in toolbar

Define Release group : RG (go with any name)

jg Provide class which we define : Jaganclass

Ctrl+S

F3

II nd step go with Release code :

Delete entries if there in window

go for new entries in toolbar

provide Release group which we created : RG

Define Release code R1 (any name)

Specify workflow : 1 (indicates plant based workflow)

provide Description : DEMO

provide second entry for Manager

Rel.group	Rel.code	workflow	Description
RG	R2	1	DEMO

* Release group is a collection of Release codes.

Ctrl+S

F3

III rd go with Release indicator :

go for new entries

provide indicator : [A] [create]

once again go for new entries

provide indicator [B] [Approve]

Ctrl+S
F3
F3

go with Release strategy

Delete if there old entries

go for new entries

provide Release group which we defined : RG

provide Release strategy RS

provide Rel. codes which we defined

[R1]

[R2]

select Release code R1

go with Release prerequisities ↵ ↵

go with Release status ↵ ↵

go with classification

under that go with searchhelp select

0 upto 1 Lakh ↵

F3

go with Release simulation ↵

Repeat same steps for Release Code R2

Ctrl+S
F3
F3

Let us go with workflow

Lets go for new entries

provide Rel. group which we defined	Rel. codes	plant ID	ob	Agent ID
RG	R1	1000	US ↳ user id.	SAPUSER
RG	R2	1000	US	SAPUSER

Ctrl+S

go with workflow builder create task

go for container and check Release code is Mandatory or not if it is mandatory ok otherwise double click on that change it to mandatory so under input we have to provide

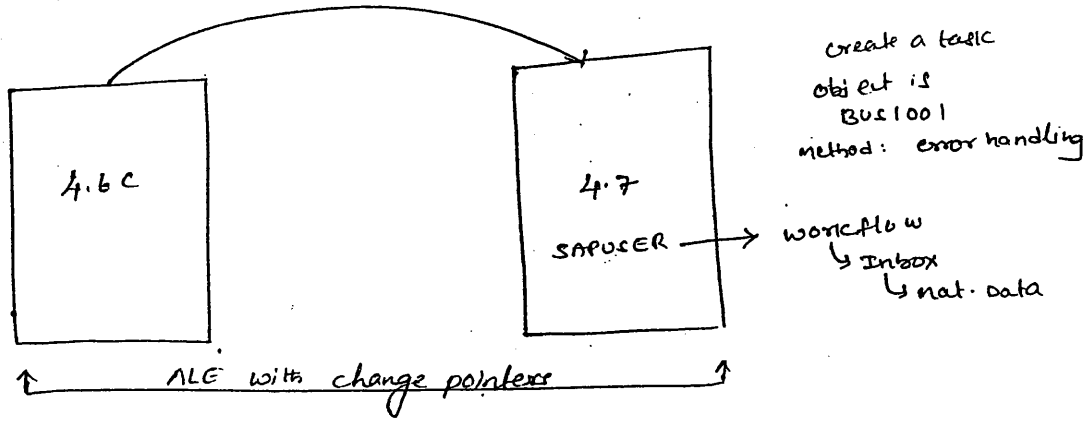
Release code : R1

REQNO :

If the order value is < 1 LAKH it is avilable in clerk inbox if grater than lakh it is avilable in manager inbox.

ASSIGNMENT:

1.



Let us customize workflow to rectify IDOC errors.

2. Difference b/w Events / Tasks / workitem.
3. provide Event linkage functionality
4. " Binding "
5. Diff b/w single step & multistep tasks
6. Explain container functionality & Elements.
7. provide Business scenario on workflow.

4/9/05

ALE ENHANCEMENTS:

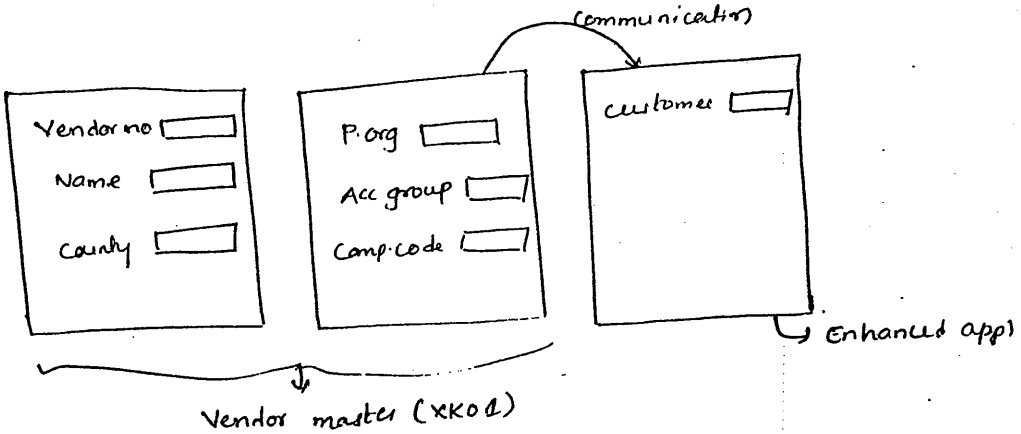
1. USER_EXITS.
2. MENU_EXITS.
3. FIELD_EXITS.
4. FUNCTION_EXITS.

USER_EXITS:

* Either with modifications or with enhancements user can add a fields for SAP predefined appl.

* Modified appl doesn't supports for upgradations where as enhanced appl can support for upgradations.

* With the user_exits user can add a fields to SAP provided appl.



* Whenever appl is enhanced by default SAP implements LUW concept on enhanced appl

* Whenever appl is enhanced user can find enhanced screen always in the last.

BADI's are alternative to the enhancements.

↳ Business ADDIN's

T. codes are SE18, SE19 } BADI's

* In case of BADI's user can call enhanced appl screen

SMOD is a T.code for creating new enhancements codes and to check existing enhancement codes without enhancement code it is not possible to enhance field. Otherwise go for SAP AG provided codes

SAPMF02K → Enhancement code for Vendor appl

SAPMF02H → " " G/L account

SAPMF02D → " " Customer appl

Under code we can maintain can two predefined user defined.

MENU_EXITS: With menu_exits user can design an user defined option in SAP menu using this we can run programs directly.

Field_EXITS: With field_exits user can implement field level validations in SAP R/3.

From V4.6C onwards SAP doesn't supports field_exits concept.

Function_Exits: Function_exits is a part of user_exits and it holds the logic required for enhancements.

User_exits:

go for CMOD T.code for SAP enhancements

provide projectname : YJPR

go for create

provide short text for a project : project for vendor appl

ctrl+s

go with Enhancement assignments in toolbar

provide enhancement code

SAPMF02K (For vendor appl)

go with components in toolbar ↵ ↵

Function_exit nothing but Function

F3

Ctrl+F3

go with SE37 T.code for function builder

provide function_exit : Exit_SAPMF02K_001

go for display

we find a user defined include provided by SAP

copy down that ZXF05U01

go with SE38

provide that include : ZXF05U01

go for change mode

provide logic

call transaction 'YTRANS'

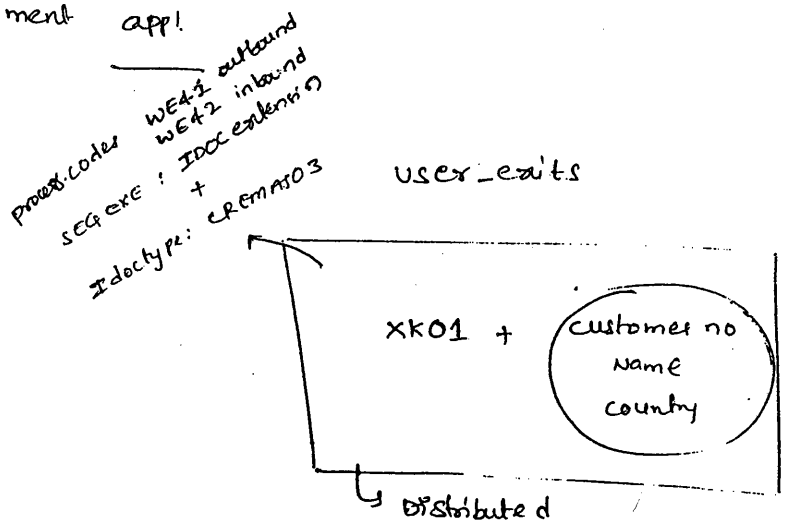
↳ T.code for user defined appl

Ctrl+S

Ctrl+F3

Check the T.code for vendor we can find with enhance

ment appl



* Whenever appl is enhanced regarding distributed even user has to enhance respective IDOC type also

* With IDOC type extension

Creating Segments.

go for WE31 T.code for creating segment

provide segment name : Z1 segment

go for create in toolbar ↵

provide short text : segment for vendor

provide the fields in a segment which we enhanced

KUNNR	KUNNR
LAND1	LAND1
NAME1	NAME1

Ctrl+S ↵

go with WE30 T.code for IDOC type

Select the extension under subobjects

provide extension name : segexe
↳ go for any

go for create

Select linked. basic type : Cremas03

Description : Demo ↵

Select one of the segment

go for create in toolbar ↵

Add the segment which we created : z1segment

make it mandatory also ☑ mand key.

provide minimum number : 1

max. " : 1 ↵

For one transaction if it item details go for more than 1

Ctrl+S

go with WE82 T.code for assign basic type to the extension and assign extension to the message type.

go for new entries

provide mess. type : CREMAS Basic type CREMAS03 extension SEGEXE version 4.6C

Ctrl+S

(267)

go with function module

↓

IDOC programming

MENU_EXITS: T. code is SE43

go with SE43

provide menu_exit name : YJaganexit

go for create

Description : Menu for Demo ↵

go for Edit

↳ insert menu entry

↳ insert as subnode

go for add report

provide existing report : Yvendorprog ↵

Keep some applications

Vendor master application : XK01

Material " " : MM01 ↵

Ctrl+S

go with SU01 T. code for user maintenance i.e user creation

USER CREATION: provide userid : .USERSNP in SU01

.go for create

provide Title : Mr.

Lastname : Jagan

Firstname : Jagan

go with logan data

provide password to user : Jagan

Jagan

provide menu_exit name : yJagan exit

go with profiles

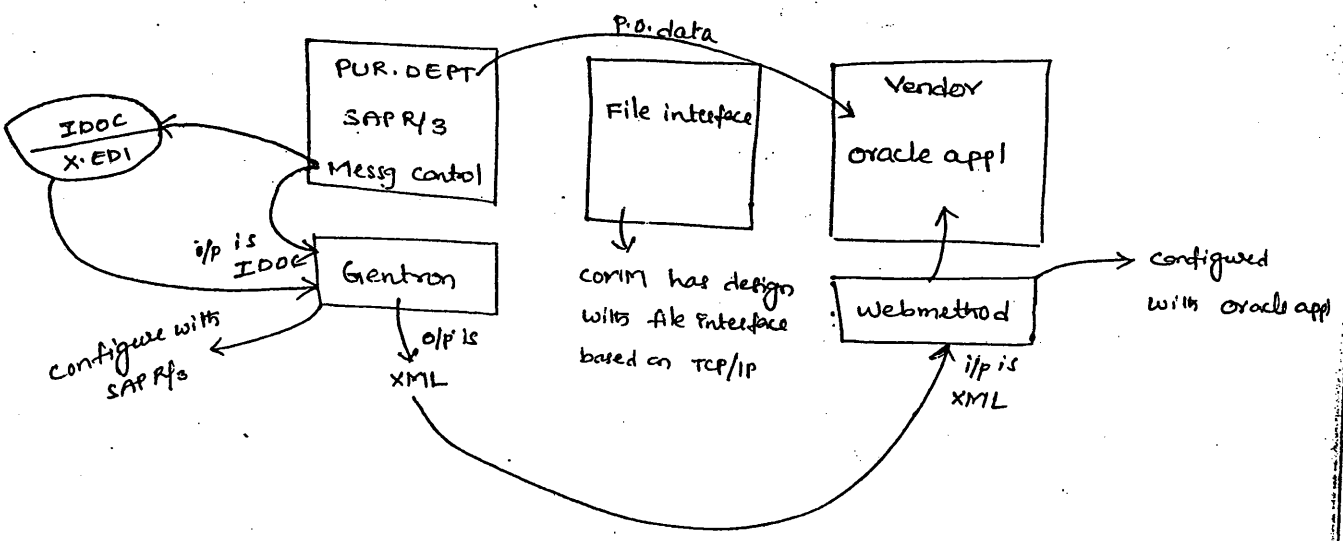
provide SAP_ALL (all appl can access by users)

Ctrl+S

Logon to that user id

EDI : ELECTRONIC DATA INTERCHANGE

* With EDI users can transfer data across R/3 to non SAP systems under distributed.



3rd party systems:

1. EDI subsystems
   - (i) Gentran
   - (ii) Mercator

   } products

2. EAI Technologies (Enterprice appl integration)
   - (i) Web methods
   - (ii) MQ series
   - (iii) IBM cross world

EDI or EAI respective technologies can speak by using XML.

provide RFC destination : Server_EXEC (TCP/IP)

RFC EXEC is executable program it can execute form for forwarding XML document to communication layer.

go with Inbound file (For receiving Acknowledgment)

provide funct. mod : EDI_Path_Create_Date_Time

ctrl+s

go for WE20 Define partner profiles

Select partner type : LI

go for create

provide partner no : 100

provide details of Vendor

ctrl+s

go for Create outbound parameters

partner function : VN

Messg type : orders

Receiver port : VJPORT (same port for sending & Receiving)

Transfer IDOC Immediately

select start subsystem

Basic type : ORDERS03

ctrl+s

go with messg control

go for create

EF NEU ME10

ctrl+s

go with MN05 maintain condition record

op type is = NEU ↵ ↵

P.org : 1000

Function modules which can generate files dynamically. (220)

EDI_path_create_username
EDI_path_create_Date_Time } we have 7 F.M
⋮

These function modules will allows the system to generate files dynamically it is used for take backup.

Once IDOC transfer to subsystem, subsystem can map fields contains in IDOC with data element contains in subsystem with this mapping finally IDOC convert into XML.

EDI STEPS:

1. Define port T.code is WE21
2. Define partner profiles T.code WE20
3. Maintain condition record T.code MN05
4. purchase order change T.code ME22

DEMO:

go for WE21

select file option (i.e file interface)

go for create

provide port name : VJPORT

Description : port for EDI

go for outbound file

for

P.No Vendor	P.Func	P.NO	med	Time	Lang
100	VN	100	6 ↓ EDI	4	EN

Ctrl+S

go with ME22

select one existing P.order for vendor 100

go with Header → Messages

Keep details

o/p type	Med	P.F	P.NO	Lang
NEU	EDI	VN	100	EN

Ctrl+S

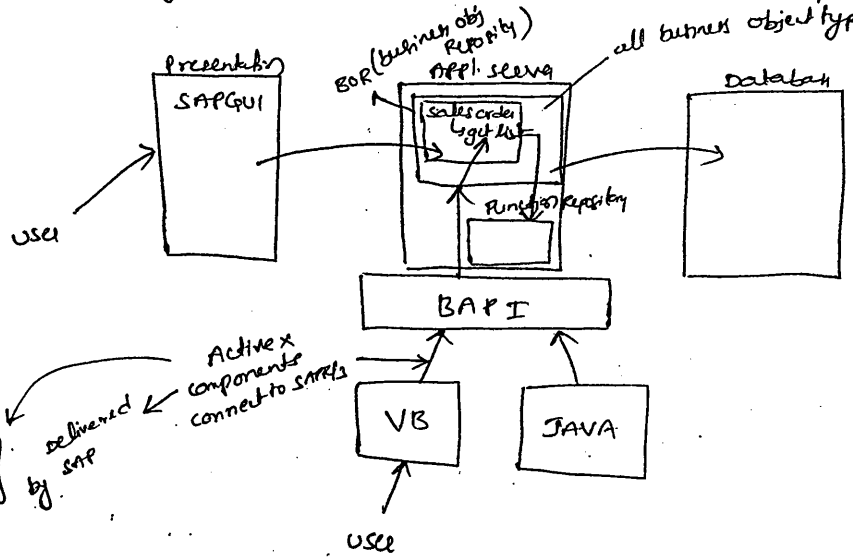
go for WE05

F8

10/9/05

BAPI

* Business appl programming interface
* collection of methods and functions
* using BAPI we can provide 3rd party front end interface.



* BOR contains all business objects

Business object type : Collection of characteristic of a business object
- → properties
- → Functions or methods.

Business object : an instance of a business object type

BAPI T-code for bapi explorer to access BOR

go with sales vs dish
- sales
  - salesorder (collection of methods & attributes)
    - getlist- (it will return all saly orders)

right side we get a function module.

BAPI_salesorder_getlist-

BAPI function modules are remote function module ie RFC

Activex components:

SAPlogon : we can logon to R/3 system

SAPBAPI : we can create instance of a business object type in BOR.

SAPTableview: equivalent to Table control in Dans.

Create appl by using VB

go to project
- components

Select ✓ SAPBAPI
✓ SAP Logon
✓ SAP Tableview ↵

Drop it in . layout

F5 (execute)

For logon : Private Sub Command_Click()
SAPBAPIControl1.Connection = saplogoncontrol1.NewConnection
If sapbapicontrol1.Connection.Logon = True then

```
Else
  msgbox " connected to R/3 system not"
End if
End sub
```

For Getlist: private sub command1_click()

```
set BAPI =                                  → object type
SAPbapicontrol1.getsapobject ("BUS2032")
BapI.Getlist customernumber := text1, _
             salesorganization := text2, _
             salesorder := orders, _
             Return := BAPIRet
MSG = Bapiret.value ("message")
If not MSG = Vbnullstring then
   msgbox "no data found"
Else
   Orders.views.add saptableview1.object
   orders.Refresh
   end if
   end sub
```

Advantage BAPI against BDC :

1. BAPI's are always backward compatible i.e which can work on upgradations also.
2. BDC will not work for upgradations.
3. BAPI provides versions for each and every upgradations.

BAPI Function modules:

BAPI_material_availability (returns material stock avicable) from MARD

BAPI_PO_create (create Purchase requisition)

1. BAPI Function module as remote enabled

2. all parameters should be passed by value only not reference

3. we can use exception in BAPI's

BAPI Return is structure is used for exception

BAPIS

Data : Begin of Itab occurs 0,
MATNR like mara-matnr,
WERKS like mard-werks,
Meins like mara-meins,
Labst like mard-labst,
End of Itab.

Data : Begin of Itab occurs 0.
include structure ITTAB.

call function . WS_upload.
exporting.
filename : c:\Jagan.txt.
filetype . 'dat'.
tables
Data_tab : ITTAB.

Data : AVAIL like mard-LABST
end of ITAB1.

Data : V_BAPIWMDVS like BAPIWMDVS occurs 0
V_BapiWmDVE . BAPIWMDVE occurs 0.

Loop at itab. into itab1
write : ITAB1

call function BAPI_Material_Availability

plant : ITAB-WERKS
mat : ITAB-MEINS
UNIT : ITAB-LABST

Importing
AV_QTY_PLT : ITAB1-AVAIL

tables
WMDVSX : V_BAPIWMDVS
WMDVEX : V_BAPIWMDVE

Append ITAB1.

endloop.

call function : WS_download

filename :