[**ganapati.adimulam@gmail.com**](mailto:ganapati.adimulam@gmail.com)**, 9948412127**

**Module - 1**

----------

**1. R/3 Architecture**

ERP - is a way to integrate the data and processes of an organization into a Single System.

E - Enterprise , R - Resource(s), P - Planning

S - Systems A - Applications and P - Products in Data Processing (started in 1972)

A - Advanced B - Business A - Application P - Programming

**R/3 -> Real time 3 - Tier Client - Server Architecture**

Real time Integration means - For each change in one application causes, the automatic change

in other associated applications .

**Note** : Main purpose of R/3 is to provide a suite of tightly integrated , large-scale business

Applications.

**3 - Layers**

--

**1. Presentation Layer -** Where the GUI is installed

Collect User inputs via GUI

**2. Application Layer** - Where Business Logic is executed

It Consists of a dispatcher and multiple work processes

**Requests from PS -> Dispatcher Queue -> Dispatcher -> Work Porcess1, WP2, WP3, WP4**

**Note :** A Work Process Can handle only one request at a time.

**Note :** To Process any user request, work process need to address two special memory areas i.e

**User Context and Program Roll area.**

**User Context** : about the user i,e name, authorizations,currentprog executed.

is allocated when login and freed when they log off.

**Program Roll area :** about the program execution , allocated by Work Process instance for

. values of variables

. Dynamic Memory allocations

Each time a user starts the program execution, roll area starts and freed when the prog.execution ends.

**3. Database Layer**

Stores & Retrieves all Data.

**Centralistic** : All the three Layers resides on the same physical Computer and is known as 1-Tier

**Note :** Centralistic Client/Server Configurations **are not scalable** at all and this way is not used

in real time / production environments.

SAP Netweaver is SAP's platform for Composition and integration of loosely coupled applications.

**SAP Netweaver supports two languages i.e ABAP and Java.**

**Note :** A Table is Client Dependent , when it's first field is of type - CLNT(3)

By default, Each SAP system comes with three clients installed- 000,001,066.

**Note :** Client is a Unit in R/3 System , With separate set of Master Data and it's own

Set of Tables.

**SAP Open SQL** - is Database Independent SQL and the Database Interface will convert

Open SQL -> Native SQL, So SAP is Database Compatible/Independent.

Open SQL is Automatic Client Handling.

**SAP suports even Native SQL also , by keeing the Native SQL Statements with EXEC - ENDEXEC**

**Basic Navigation via Commands**

---------

/n<tcode>: To Call <tcode>

/n : Cancel the Current transaction/session

/o<tcode>: To Call the tcode in a new Session

/o : Display an Overview of Sessions

/nend : Logoff

/nex : Logoff

**ABAP Workbench Tools** : Collection of tools to Develop and test, run ABAP Programs.

. ABAP Editor,ABAPDictionary,ScreenPainter,MenuPainter,Function Builder ...

Q1. \_\_\_\_ is like an Operating System for SAP ( NetWeaver ? )

Q2. An \_\_\_\_\_\_ Interprets the ABAP/4 programs and manage the Input and Output for them.(Application Server)

Q3. \_\_\_\_ Memory Contains User Info that Logged On.(User Context)

Q4. \_\_\_\_ to Cancel the transaction (/n)

Q5. \_\_\_\_ is collection of tools you use to develop,test and run ABAP Progs(ABAP Workbench)

**2. Introduction to ABAP Programming**

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

**Program types**

-------

**1** - Executable/Report Programs - it's source code is stored in Table - DD010S.

**M** - Module pool/Dialog/transaction Program - can be executed via tcode only.

**I** - Include Program

**Program Attributes**

---

Editor Lock - if Set, Other users cannot modify ,rename,delete the program.

Fixed Point Arithmatic - if not set, system doesn't recognize decimals ,i.e 3.14 as 314

Start via variant - If set, user can start the program only via Variant.

**ABAP Syntax**

-----------

. Every statement must begin with a Keyword, end with a Period

. Commented lines begin with '\*' and comments from begin of line begins with "

**Note :** ABAP programs are Interpreted but not Compiled.

Only the active development object is visible system-wide.

During the activation, the system creates a runtime Object and Upon execution,

the system executes this Runtime Object.

**ABAP Language Elements**

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

. **Declarative Language Elements**

TYPES, DATA,TABLES,PARAMETERS,SELECT-OPTIONS

**. Time Language Elements** - Specify the Point in time(Event) When to execute

START-OF-SELECTION, END-OF-SELECTION,AT SELECTION-SCREEN, AT LINE-SELECTION ...

. **Control Language Elements** - Controls Processing Logic - IF-ELSE-ENDIF, CASE-ENDCASE.

. **Operational Language Element** -WRITE, MOVE ...

**Note :** ABAP is an **Event oriented Language** , it doesn't necessarily process statements in

Sequential Order.

**The Block of statements under each event is known as a Processing Block.**

There **is no particular** element to **mark the end of processing block** .

A New Event or Defintion of Subroutine(FORM-ENDFORM) ends the Preceding Processing Block.

**Note :** It's good practice, to list the processing blocks in their order of execution.

**ABAP Data Types**

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

. Non-Numeric Length Types Lenth in Bytes

C - Character 1 - 65535

N - Numeric Character 1 - 65535

X - Hexa Decimal Byte Field 1 - 65535

D - Date 8

T - Time 6

Numeric

I - Integer 4

F - Packed number 8

P - Floating Point number 1 - 16

. Variable Length

STRING, XSTRING

**Text Symbols** : are simple text literals and Can be translated to other languages. TEXT-<XXX>.

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

**WRITE and It's Formatting options For All Data types**

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

LEFT-JUSTIFIED, CENTERED,RIGHT-JUSTIFIED.

UNDER <g> - Output starts under variable <g>

NO-GAP

USING EDIT MASK <m> - Specifies format template <m> - generally used for

Date,Time

USING NO EDIT MASK - Deactivates a format template specified in Data Dictionary.

No-ZERO - Leading Zeros repalced with blanks - for C,N

**Formatting options Only For Numeric Fields**

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

NO-SIGN - Leading Sign is Not Displayed

DECIMALS <d>- <d> defines the no.of digits after decimal point

EXPONENT <e> - In type F Fields, the exponent is defined in <e>

ROUND <r> - Type P fields are Multiplied by 10 perwe(-r) then truncated

CURRENCY <c> - Format accoding to Currency <c> in table TCURX

UNIT <u> - No.of Decimal places is fixed according to unit <u> specified in Table

T006 for type P Fields.

**DATA Objects/Variables** : Whose Contents Can be Changed.

**Constants** : Whose Contents Cannot be Changed.

CONSTANTS C\_pie TYPE P DECIMALS 3 VALUE '3.14'.

**C\_pie = '3.14'**. "Even this throws an error as it is also

"overwriting/change irrespective of value

**Displaying Symbols & Icons**

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

INCLUDE <symbol>.

INCLUDE <icon>.

WRITE : / 'Phone Symbol', SYM\_PHONE AS SYMBOL.

WRITE : / 'Alarm Icon', ICON\_ALARM AS ICON.

**1st Statement of a Program is generally recommended to be**

REPORT <Name> LINE-COUNT 20 LINE-SIZE 150 NO STANDARD PAGE HEADING

MESSAGE-ID <Msg.Class>.

**NEW-PAGE - triggers a new page explictly.**

RESERVE 3 LINES - Same as printing 3 Blank Lines.

**3. Selection Screen**

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

PARAMETER(s) for Single Field -**Lenght of it's name cannot be more than 8.**

parameter p\_land1 type land1\_gp **lowercase.** "input case is as it is entered,

"without lower case, it is upper case by default

"beacause of it's data element

**Note :** **PARAMETER Cannot be used for type 'F'.**

**SELECT-OPTIONS** for Multiple/Range Of Inputs

With the name of Select-options, **an itab with header line is declared**

with**SIGN, OPTION,LOW,HIGH**.

**SELECTION-SCREEN for formatting the Selection Screen**

**Note :** Default Selection-Screen number is 1000 and we can go with custom number also i.e

SELECTION-SCREEN BEGIN OF SCREEN <number**> [AS WINDOW ]**

SELECTION-SCREEN END OF SCREEN <number> .

CALL SELECTION-SCREEN 500 STARTING AT 10 1.

AS WINDOW -> Screen is Displayed as Dialog Box

SELECTION-SCREEN SKIP [n]. "n Blank Lines in the Input Screen

SELECTION-SCREEN ULINE.

SELECTION-SCREEN COMMENT .

SELECTION-SCREEN BEGIN OF LINE.

To Keep more than one element in the same line as by default, each element is on a new line.

SELECTION-SCREEN END OF LINE.

**4. Data Dictionary(DDIC) - Objects (SE11)**

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

**.Domain**

**.Data Element**

**.Tables**

**.Structures**

**.Views**

**.Table Types**

**.Search Helps**

**.Lock Objects**

**DDIC - Virtual and is Database Independent.**

**Table Purpose**

DD02L List of all Tables

TSTC List of all Tcodes

TADIR R/3 Repository Objs

T000 All Clients

**DDIC(Database) Table Types**

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

. **Transparent Tables** - One to One Relationship between DDIC and Database

**. Pooled Tables**- Many to One , SAP Proprietary Construct, Stored in Table Pool

- Table Pool is collection of Pooled tables.

- For Many Pooled Tables(Table Pool) in DDIC, Only One Table Created in DB

i.e**for Table pool with Pooled Table details , so that it is Many -> One**.

**- Holds Customizing Data**i.e Codes, number ranges,parameters,CountryCodes,Exchange Rates.

- Data in Customizing Table is Set by Functional Consultant during the initial

Implementation.

- **Table Pool Definition** Contains two Fields i.e

**Table Name ,**

**VarKey(Contains entry from all key fields)**

. **Cluster Tables**- Same as Pooled Tables **Except all the Cluster Tables in a Table Cluster**

**must have the Same Primary Key.**

**Restrictions On Pooled and Cluster Tables**

**------------**

. **Secondary Indexes Cannot be Created**

. Cannot use SELECT DISTINCT, GROUP BY

**. Cannot use Native SQL**

. ORDER BY Can be Used only for Primary Key

**Delivery Class**: Controls the transport of table data for installation,upgrade,client copy,

------- and When transporting between systems.

Possible Delivery Classes : A , C , L, G,E,S,W.

**Every table should have atleast one field as Primary Key field.**

A Table cannot have more than **16 fields combination as key fields.**

**Max. No of fields in table cannot be more than 249.**

**Note :Reference Fields required for fields with data type 'CURR', 'QUAN'.**

**Field Reference Field**

CURR CUKY

QUAN UNIT

**Reference Fields can be in the same table or from another table.**

**Technical Settings**

---------

**. Data Class** - Physical Area/Table space where the table need to be stored in DB

- APPL0(master Data), APPL1(Transaction data), APPL2(Organization Data) etc

. **Size Category** - Expected no. of records to be stored , space is allocated accordingly

If the table needs more space for data later, extents are added as per

size category.

**.Logging** - To Store the modifications to the entries of the table.

Along with this Check**, profile parameter rec/client need to be set**

**by admins.**

**rec/client = ALL (Applicable to all Clients)**

**rec/client = 000 (Only in client 000)**

**rec/client = OFF (Logging is not enabled on this system)**

All the Logs with transaction table history can be checked through tcode - **SCU3.**

**.Buffering** - Allows to access data quicker by accessing data from Application Server

instead of database.

**Buffer Synchronization**

------

If Program Changes table data , it is noted in the log table by **Database Interface**.

A Synchronization Mechanism runs at a fixed time Interval i.e**Log table is read and**

**Buffer contents are invalidated.**

**SO in the next access, data is read from database table and updated in buffer.**

**Points to be kept in mind While buffering**

**------------------**

. Only the tables frequently read and rarely changed

. The **Key Fields of the buffered table should be of Character data types C,N,D,T.**

**Note :** During the Selection/Reading data, **Buffering can be by passed**, if the table data

need to be read from Database.

**Buffering Type :** Determines Which records of the table are loaded into the buffer .

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

When When a record(s) accessed

**. Full Buffering** : **All the records of the table are loaded.**

For Smaller table Size, read frequently and write less frequently.

(Except table size, read frequently and write less frequently applicable for

all buffering types)

**. Single Record Buffering :Only that was accessed is loaded into the buffer.**

generally for SELECT SINGLE

If the record accessed in not in DB Table, this is recorded in buffer as non-existent.

**. Generic Buffering** : All the records whise**left-justified part of the Key**

**is the same are also loaded .**

**Data Elements & Domains**

**-------------**

**Field -> Data Element(Contains the Scematics/meaning i.e Field Labels and Online Documentation)**

**-> needs a Domain**

(For technical details i.e Data type,Length,Decimals,valuerange,value table)

**Note :Fixed values from Domain are checked via Screen Inputs but when inserted through program(WA/Itab).**

**Note**: **Value Table** at Domain level - **not acts as check table** but will **be proposed as a default check table during the Foreign key relation .**

**Note :** Like Fixed Values, **Value Table neither restrict the entries not proposed as F4.**

**Note** : Input Check valid for Few Data types

. Value Range For CHAR, NUMC

. Fixed Values For CHAR, NUMC, DEC, INT1, INT2,INT4

**Data Element**: Specifies the ScemanticCharacterstics of a Field

Refers an Elementary type i.e Data type,Length or Domain or

Reference typei.e for a Class or an Interface.

Short Description appears on Screen Field

**Note :**F1 help comes from Documentation,if no documentation then Short description itself.

**Foriegn Keys :** Defines Relationship between Tables i.e

. CHECK Table

. FORIGN Key Table

. Creates Values Checks for Input fields

**Note :Check Table Fields must be primary key fields and Both Check Table Field and Foreign Key field must refer the same Domain**.

**Indexes :** Always in Sorted Form.

Primary Index : is generated automatically during table activation .

Contains Only the Key Fields of a Table and pointer to non-key fields.

**Secondary Index:**Additional Indexes Could be Created explictly for the most frequently

accessed dimensions of the table , i.e for non-key fields.

**Note :The Indexes used depends on the optimizer used for database system,**

**but system doesn't use Suitable indexes for selection even if exist.**

**Structure** : Collection of Fields , generally the the common fields in many tables can be grouped as a structure, so the structure can be included in many tables.

**Structure Fields can refer**

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

. An Elementary Data Type - in Structures

. Another Structure - In Nested Structures

. Table Type - In Deep Structures

**Structure Types**

---------

**. Flat Structure** - Each Field refers an Elementary type

**. Nested** - A field may refer another structure

**. Deep** - A Field may refer a Table Type

**Modifying/Enhancing Standard Tables via**

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

1) **. APPEND Structures** - Each APPEN Str is For a Single Table Only

- With Customer name space - field names recommended to begin with YY/ZZ

**- Can be Created without SAP Preparation**

**Note**:**APPEND Str is not allowed for Pooled / Cluster Tables.**

**If table has Long field(LCHR/LRAW), APPEND Str is not allowed as the long should be always at the end**.

2) **. CI Include Structures**(Customizing Includes)

- Can be Included in multiple tables

- Already Integrated into SAP Tables by SAP with no fields

**Views**  : Data derived from One or Many Tables

----- Data in the View is not stored Physically .

**View Types**

----------

**. Projection** : View Data exactly from One Table. Pooled and Cluster Tables can be used.

To Project the required fields from all the fields of a Table.

No Corresponding object can be created in Database.

**. Database View** : Application Specific view and From One or Many Transparent Tables

and Implements Inner Join.

IF Contains Single Table , Even Data Can be Inserted else only for View.

it is Created in DataBase.

. **Maintenance View :** To Maintain the data of an application object

(Customer-General,Bank,Payments data together)

All the Tables in this view must be linked with Forign Keys.

. **Help View :** When an Outer join is required as Selection Method for a Search Help as

Selection Method of a Search Help is either a Table/View.

**Search Help** : To Display List of Possible Values When F4 is Pressed on Input Screen Field.

-----------

**Types :Elementary** - Only One Search Path

**Collective** - Combines Several Search Paths(Elementary Search Helps)

**Input For SeachHelp** :

. **Selection Method** - Table /View from Which Data need to be shown

. **Search Help Parameters** - List of Fields, Import/Export

**Import** - Information from the Screen is Copied to Help process

**Export** - Values from hitlist is returned to input template

**Lpos** - Input Fields order , if restriction is allowed

**SPos** - Display Order

**Note :Search help can be attached** to

. Data Element - Can be accessedby all the screen fields refer the data element

Export parameters , Search help name must be assigned

to Further Characterstics of the Data Element.

. Check Tables -

. Table Field / Structure Field

. Screen Fields - via Screen Paineri.e it's attribute's box and it is selection-screen

PARAMETERS/SELECT-OPTIONS AS SEARCH PATTERN

**Lock Objects**: To Synchronize Simultaenous access to same Data by several users.

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

**Lock Arguments** : Consists of Only key fields from the table.

Used as Inputs to Function Modules of Setting,Releasing Locks.

**Note** : Lock Object name must begin with 'EY/EZ' and after each lock object creation/activation,

Two Function Modules are created i.e ENQUEUE\_<LockObjName> -> To Set Lock

DEQUEUE\_<LockObjName> -> To Release Lock

**Tcode - SM12** can be used to Check, whether lock has been applied or not i.e after Enque Function.

**Lock Mode** : Defines how the users can acccess the locked records of the table

---------

**Write Lock** : Locked Data can be displayed or edited by a Single user.

Can be requested several times from the transaction and are processed successively.

**A Request for another Exclusive Lock or Shared Lock is Rejected.**

**Read Lock :** More than One user can access the Locked data at the same time

but only in Read/Display Mode.

A Request for another Shared is accepted.

**Exclusinve But not Cumulative** : Can be Called only once from the transaction

All Other Lock Requests are Rejected

**Summary**

To Maintain Database Object such as

. Domains,DataElements,Structures,Tables,Views,TableTypes,Search Helps, Lock Objects.

Q1. \_\_\_\_ in the Dictionary has one -> one Relation. (Transparent Table)

Q2. \_\_\_\_ Determines the Table Space that the table is assigned to.(Data Class)

Q3. An \_\_\_Can be used to speed up the selection of data records from a table.(Index)

**5. Common Control Statements**

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

IF - ENDIF , CASE - ENDCASE

Exit

DO - ENDDO , WHILE - ENDWHILE

CONTINUE and CHECK

**Note :** IF-ENDIF Can check for Expressions also and where as CASE-ENDCASE, always single value and also compared for equality.

**EXIT**  : Prevents further processing .

When it is With in FORM - ENDFORM, MODULE - ENDMODULE, DO - ENDDO or inside loop, it exits only from

the corresponding block. Else When it is outside the block, it exits from the Program.

SY-INDEX is the Current Loop Counter for DO - ENDDO, WHILE - ENDWHILE

SY-TABIX is the Current record Counter of Itabfor LOOP AT ITAB - ENDLOOP.

**Note** : After ENDWHILE, sy-index contains the value it had before entering into the loop.

**Terminating Loops Explictly**

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

Un-Conditionally via CONTINUE, EXIT

Conditionally via CHECK

**CONTINUE** : Should be always within a LOOP.

It Continues from the next loop pass from the place it appears, by ignoring the statements

below it.

Ex : DO 10 TIMES.

ifSy-index between 3 and 8.

continue.

endif.

writesy-index.

enddo.

**Output** : 1 2 9 10 (not between 3 and 8)

**CHECK** :Unlike CONTINUE, it accepts expression.

if it is false, jumps to end of loop or to next processing Block

else(true) execute the block the below it.

Q1. In a Case Statement , a Complex expression can be Compared . FALSE

Q2. The \_\_\_\_ Statement leaves the Current loop . CONTINUE

**6. String Operations**

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

SHIFT <str> [by n places] <mode> . "mode can be right, left(default),circular(begin->end).

"left + circular(default), right+circular

SHIFT <str1> up to <str2><mode>.

Ex : data alpha(10) value 'abcdef'.

shift alpha upto 'cd'. "left

write alpha. ->cdef

shift alpha upto 'cd' right. "cd ?

Ex : Shift '0000100100500' left deleting leading '0'.

output : 100100500.

Shift '100100500' right deleting trailing '0'.

output : 1001005

**TRANSLATE** : Converts Characters into upper/lower case or uses substituion rules to convert all

--------- occurrences of one character to another char.

**TRASLATE**<str> USING <pattern>. "pattern contains the substition rule

Pattern(variable) contains letter pairs where the 1st letter of each pair is replaced by 2nd.

DATA v1 type c VALUE 'AbCde12'.

DATA rule(10) type c VALUE 'AXbxCYdy12'.

data v2 like v1.

v2 = v1.

TRANSLATE V2 TOP LOWER CASE.

write / v2. "ABCDE12

**TRANSLATE** v1 **using rule**. "same as TRANSLATE v1 using 'AXbxCYdy'.

write / v1. "XxYye22

**FIND** : To find ouccrances of a pattern within a string.

---- can also be used for finding the string pattern within an internal table.

FIND <Str> for <pattern><option>.

FIND <FIRST OCCURRENCE/ALL OCCURRENCES >OF <str1> IN <str2><RESPECTING/IGNORING CASE>

<MATCH COUNT mcount>.

**Defaul**t - FIRST OCCURRENCE, RESPECTING CASE.

**Strlen** : is to find out the string length/no of chars in a given string.

**CONDENSE** : All the Sequential spaces into a single space.

--------

Ex : V1 = 'abc ltd'.

CONDENSE v1. "result is 'abc ltd'

**CONDENSE v1 no-gaps**. "No space at all through out the string

**CONCATENATE <**str1><str2> into <str3> [SEPARATED BY <Any Sep>].

**SPLIT**<str> at <delimiter/separator> INTO <str1><str2> ... <Str n>.

**Operators For Character Strings**

**Note :** CS,CP,NS,NP operators ignores trailing blanks and are NOT CASE Sensitive(ignore).

SY-FDPOS

Operator Meaning True When for TRUE For False

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

v1 CO v2 Contains Only v1 is Composed Solely

of v2

v1 CN v2 Not v1 CO v2 v1 Contains Chars that

are NOT in v2

v1 CA v2 Contains Any v1 Contains(Any of) i.e

at least one char of v2

v1 NA v2 Not v1 CA v2 v1 doesn't contain any

char of v2

v1 CS v2 Contains String v1 contains the

character string v2

v1 NS v2 Not v1 CS v2v1 does not contain

character string v2

Q1. \_\_\_ is used to find the length of string. (strlen)

Q2. CONDENSE and CONCATENATE perform the same function. (FLASE)

**7. Internal Tables**: Acts as Arrays. But Dynamic in Size

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

**Purpose** : For Retrieving and processing data from a Database table within a Program.

**Note** : Data type of an Itab is fully specified by it's

. Line type - Can be any data type but usally a structure

. Key - Identifies Table rows

. Standard/default Key

. Tables with Structured type - All Character type fields together

. Tables with Elementary line-type - Entire line

. User-Defined Key

Any Column can be defined

**Note :** Key Can be either UNIQUE or NON-UNIQUE.

Itab types : Indexed

- Standard(default),

- Sorted - records in sorted order always by it's key

Non-Indexed(Key)

- Hashed

**Key and Index Access**

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

**Standard Sorted Hashed**

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

**Index Access** Yes Yes No

**Key.Access** Yes Yes Yes

**Key.Values** Non Unique Unique/Non-Unique Unique

**Preferred Access** Maninly IndexMainly Key Always Unique

**Note :** Response time for both standard and Sorted tables is propotional to no.of records and where as for hashed tables, it is constant and it is via Hashing alogirthm.

**HEADER LINE** : Declares an addition Data Object(Variable) with the same name and line type as an Internal table .So Header line is used as Implict Work Area.

**CLEAR,REFRESH and FREE**

REFRESH Always can refresh Only Itab and where as CLEAR and FREE Works for any Variable(V1,WA,ITAB...)

**When I TAB is With Header Line**

CLEAR ITAB . "Header Line CLEAR ITAB[]. "Internal Table/Body

REFRESH ITAB. "Internal/Body as REFRESH Can refer only Itab

FREE ITAB. "Internal/Body , it is same as FREE Itab[].

IF ITAB -> Refers Header Line and IF ITAB[] -> Refers Itab

**Note :** FREE Will Free the Whole Memory along with Content and

Where as CLEAR and REFRESH for Only Content , the Initial Memory required remains allocated.

**Filling Internal Tables**

. **For Standard Itabs** - Lines are always Appended at the end of Itab

. **For Sorted,Hashed** - Inserted according to the Table Key

**READ** : Can always read only one record either with KEY or INDEX.

**Note :**Itab must be SORTed before READing with BINARY SEARCH.

READ TABLE <itab> INTO <wa> WITH KEY kunnr = '1001' BINARY SEARCH.

SY-TABIX after READ, return the index of record found in the Itab.

**Changing Line(s) of Itab**

MODIFY <itab> FROM <wa> [INDEX idx] [TRANSPORTING f1 f2 ....].

**Deleting Duplicates** via DELETE ADJACENT DUPLICATES and

for the same, input must be Itab with Duplicates Adjacent i.e Sorted Itab.

**Determining the Number of Rows in an Internal Table**

**DESCRIBE TABLE**<itab> LINES V\_Lines [OCCURS v\_no].

This fills v\_lines , sy-tfill with the no of rows from the itab.

sy-telng with the Length of Row in bytes

sy-toccu With the Current Value of Occurs Clause if exist in the Itab Declaration

**Creating Top 10 Records via append wa to itab sorted by [col1].**

**It has two un-usable Properities**

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

. It Only Sorts in Descending Order

. No.Of records will be limited to Occurs <n>i.e if it is Occurs 10, max.records - 10 Only

**COLLECT <WA> INTO<Itab>.**

If a record already exist with the same non-numeric fields, it adds the existing and current record's numeric fields else(no such record(s) exist), it appends.

**Control Break Processing/Events/Statements - Begins with AT, ends with ENDAT**

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

**Note :** AT - ENDAT Can be used only within LOOP - ENDLOOP , to process the Internal Table Data.

Within Loop-Endloop, the Same Statements can be placed any no.of times and Nested is not allowed.

Loop atitab into wa.

**at new <f3>.**

"Triggers not only by checking <f3>, it also checks it's left hand side fields <f2>,<f1> ..

"if any of <f3> to <f1> changed, then it is treated as different <f3>.

"Within the block <f3> , all the right side(<f3>+1, ...<fn> ) numeric fields turns into '0',

non-numeric fields turns into '\*'

"before and after the block, the original values remains and this behaviour is within the block

**Note** : Same is applicable for at end of <f3>. "to decide is it same <f3> or diffrent<f3>.

sum. "of the each <f3> block

"sum may cause - overflow with error - SUM\_OVEFLOW and need to be handled by increasing length of numeric field in the itab declarion

endat.

endloop.

**on Change of** : Similar to at new but this can be used even out side loop-endloop also.

------------ Only V1 value is checked but not the fields to it’s left hand side

on change of v1 [ or v2 ...]

endon.

Here it checks only the fields mentioned , here v1 and v2 need not to be the fields from itab.

Q1. \_\_\_\_is a field string with the same structure as row of body, but can hold single row.(Work Area)

Q2. \_\_\_\_ adds a single row to an Internal Table

Q3. \_\_\_\_\_ variable is used to determine the no.of rows in an Itab ( sy-tfill )

Q4. \_\_\_\_\_ Statement accumulates the contents of a structure into an Internal Table(COLLECt?)

**Module -2**

---------

1**. Report Events**

1. AT SELECTION-SCREEN Event is triggered only when there is a selscreen

or Logical DB is Used in the Prog's attribute.

LDB Can be used only by executable prog.

Other events make sense any exe.prog

Events : GET, GET LATE make sense only for LDB.

2**. Classical Events**

INITIALIZATION

AT SELECTION-SCREEN, ON <f1>, <Block>. "For Validation

AT SEELCTION-SCREEN OUTPUT.To Modify Sel.Screen before display.

START-OF-SELECTION. After Sel.Screen processing

**Note :Non Declarative Statements \_(like do-enddo) between REPORT and 1st Processing Block are**

**processed as part of start-of-selection.**

& Interactive (AT LINE-SELECTION-D.Click/F2, AT USER-COMMAND,

AT PF-<NN> ,”f5,f6.f7.

TOP-OF-PAGE DURING LINE SELECTION).

**END-OF-SELECTION**. Last Event Processed by the Sys.

After All the data has been read from program/Logical DB before the

list processor is being started.

**Exiting Events Blocks via STOP, CHECK and EXIT**

**STOP.** "Stops the Current block and triggers the next event accordingly.

**EXIT**. From Start-of-selection, Systems starts the list processor imm.

i.e control is transferred to end-of-selection ?

**CHECK**<Expr> . "if not satisified, exit the block and continue with nxt blk.(conditional)

**Similar to Stop but stop is un-conditional**.

**REJECT to exit from a GET.**

**Interactive**

-----------

**Basic List + 20 2nd ry = 21 Lists.**

Each List has it's own memory area called a list buffer.

**SY-LSIND.**No Standard page can be displayed for 2nd ry lists.

**GET and GET LATE Cannot be used for 2nd ry lists.**

Lists in Dialog **Box** : applicable under 2nd ry list events.

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

WINDOW STARTING AT <left><upper> [ENDING AT <right><lower>].

Dialog Box do not have their Menu/Std. toolbar

Application tool bar is seen at the bottom of the (any)dialog box.

**System Variables :** SY-LILLI, SY-LINNO

SET PF-STATUS <Name> [EXCLUDING <f>|<itab> [OF PROGRAM <prog> ]].

SET PF-STATUS SPACE . To Set the standard list status

SET TITLEBAR <ttl> ..valid for all the lists until we set another title.

**Menu Painter** - SE41 - to design user interfaces of ABAP Program.

. Menu Bar - 6 Menus + 2 Std.Menus(System ,Help)

. Application Toolbar

. Function key is mandatory and max 35 buttons ---- for app toolbar gui

. Function Keys(Standard Toolbar)

**2. Modularization Techs**

. Function Modules - Global - SE37, func. group, main prog, include progs.

. Subroutines - Local

**Subroutines**

**-------------**

Passing Arguments -**pass by ref (Default**) - USING/CHAING

**pass by value** - USING VALUE

**pass by value and result** - CHAINGING VALUE()

**.Stop**

**STATICS with in subroutine**.

Creates Local variables that are not freed when the subroutine ends.

**Note** : Using must come before Chainging when both exist.

Using, chaing can occur only once in each subroutine.

pass by valud and result - CHAINGING VALUE

.Stop inside this form, terminate the form and goes to

end-of-selection directly by discarding the changes to the actual arguments,if no end-of-selection, exits from the program.

**Leaving a subroutine**

EXIT.

CHECK. And STOP

-

**Check and Exit** Immediately leave the subroutine and process continue with

the next executable statement and where as

**STOP** leaves the subroutine and goes directly to the end-of-selection event.

**Stop, exit and stop do set value to sy-subrc.**

**External Subroutine**

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

perform A(z2). "where Z2 is the Prog name

Zprog1 Zprog2

TABLES LFA1. TABLES LFA1.

perform A(Zprog2). form A.

f1 is specific to each program

LFA1 is common in both the progs

endform.

data f1. data f1.

--

**LOCAL** Keyword to declare a local var only when there is a global var exist

with the same name.

**Passing Itabs as Parameters to subroutines**

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

Pass with Header Line

Pass Body Only

**form s1 tables it**.

Here it is with header line always irrespective of actual parameter has

Header line or not.

If Only itab need to be used within subroutine irrespective of ctual parameter has

Header line or not.

**form s1 using/using value/changing/changing value it[].**

**Functional Modules** - reusable - SE37 - type F - stored in central library.

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

Function group and it's main prog, includes.

Interface Parameters

---------

Importing

Cannot overwrite the content of imports with FM.

Exporting

FM -> to calling Prog

Chainging

Progm -> FM ->Change the value in FM then result to ->Prog

TABLES

Only for Itabs.

Exception -> Name of the Error.

Exceptions

---

raise<excp name>. "setsy-subrc with the excp's no in the list

message ..raise<excp name>.

**3. type M -Module Pool/Online/Dialog/Transactions Programming - do not produce lists**

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

**Transaction - user requests - a programs starts by a tcode**.

**type M(**name space SAPM(Y/Z) - can be executed via Tcode only.

Tools used - Menu, Screen Painter , DDIC, ABAP Editor, SE80, SE93(Tcode creation)

**Note : Screens Can be Created for**

. Executable - 1

. Module Pool - M

. Function Group/Function Pool - F

**Data passed between Data Objects in Program and Screen Fields Automatically**

**between the Identically named fields(tablename-de).**

**TABLES KNA1**."Importance of this st in UI prog.

Screen Elements ->correspongs to DDIC/Program.

Screen Types

. Normal

. Sub Screen

. Modal Dialog Box

Next Dynpro - by default, the same screen.

**Screen No - any <nnnn> , except 1000.**

Screen Group - SY-DYNGR for many screens to be modified similarly.

Screen Modes : Graphical(User friendly for design , Alpha numeric).

OK\_CODE -> Every Screen except Subscreen has a 20 char OK\_CODE field for function code.

**Flow Logic/Screen Logic/Screen Language**

**----------------------**

**Events**

PROCESS BEFORE OUTPUT.(PBO) - Generated by default

PROCESS AFTER INPUT .(PAI) - Generated by default

PROCESS ON HELP REQUEST(POH) - for F1 on the Input Field

PROCESS ON VALUE REQUEST(POV) - for F4 on the Input Field

PBO -> After PAI of Previous Screen and before displaying Current Screen.

PAI ->When a user chooses a function on the Screen.

Every Screen must contain PBO and PAI.

**Keywords allowed Within the Flow Logic/Flow Logic Editor**

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

MODULE

FIELD

ON -conjunction with FIELD

VALUES -conjunction with FIELD

CHAIN

ENDCHAIN

CALL SUBSCREEN

LOOP

ENDLOOP

POH - to display the Programmed Help for F1.

---

The Supplemental Documentation will be displayed, along with the Data Element's

Short text and Doc.

PROCESS ON HELP-REQUEST .

FIELD <name> WITH '0001'. "Supplement ID - 0001, corresponding doc is displayed.

FIELD <Name> MODULE <M1>. "is also allowed.

POV Programmed Help for F4 on the Input Field.

---

PROCESS ON VALUE-REQUEST.

FIELD <Name> MODULE <M1>.

**Calling Dialog Modules**

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

**. Simple Module Calls** : MODULE <M1>. Data transfer is Unconditional

. **Controlling the Data Transfer**

FIELD F1 MODULE M1. "Now Only F1 will be transferred to Prog

. **Calling Modules Un-Conditionally**

MODULE M1 AT EXIT-COMMAND.

Is Executed only when the user invokes a function code with func.type 'E'

Module M1 is executed before the automatic checks regardless of where it

occurs in the Screen Flow's PAI.

**. Calling Modules Conditionally**

For Single Screen Fields

.FIELD <f1> MODULE <M1> ON INPUT -> is called only when the <F1> value is

Other than initial Value(not exe when 0).

.FIELD <f1> MODULE <M1> ON REQUEST "called when **any value** is entered for <f1>

"even if initial value is entered explictly

.\*-INPUT

Ex: FIELD <f1> MODULE <M1> ON INPUT|REQUEST|\*-INPUT

**FIELD dynp\_field MODULE mod [ {ON INPUT}**

**| {ON REQUEST}**

**| {ON \*-INPUT}**

**| {ON {CHAIN-INPUT|CHAIN-REQUEST}}**

**| {AT CURSOR-SELECTION}.**

**ON \*-INPUT**

With this condition, module mod is called only if the user entered an asterisk ("\*")

as first character in the input field of screen field dynp\_field and

if the input field has the special attribute \*-Input.

The content of dynp\_field is passed to the ABAP program without the leading asterisk.

**ON {CHAIN-INPUT|CHAIN-REQUEST**}

The conditions ON CHAIN-INPUT and ON CHAIN-REQUEST have the same meaning as if used

after a stand-alone MODULE statement.

FIELD <f1> MODULE <M1> AT CURSOR-SELECTION. "When Cursor positioned on <f1>.

Also called when the fcode of user action is 'CS' with function type 'S'.

if you assign 'CS' to 'F2' then the module is called even for Double click.

**Message Types**

-------

S, I, A,W,E,X

I - Info on the Same Screen, Press Enter and Proceed Further

A - Stops , No changes are allowed on the Current Screen and Press Enter -> Terminates

W - On the bottom of the Same Screen, Press Enter and Proceed Further

E - On the bottom of the Same Screen, Correction Required.

X - Exit is Displayed on the next screen i.e terminated with runtime error.

MESSAGE <tnnn>[ WITH<var1><var2> ... <var4>].

**Validation can be done via**

PAI.

FIELD <F1> MODULE <M1> . "Validation Logic in the M1 or

FIELD <F1>VALUES( val1,val2,...).

**CURSOR Position**

-----

By default, cursor set on the first field and it can be set explictly via

SET CURSOR FIELD <Name> .or

from Screen attributes.

**LEAVE,SET,CALL SCREEN**

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

PAI.

MODULE M1.

MODULE M2.<- LEAVE SCREEN / TO <NNNN> : Leaves Current Screen <nnnn>. i.e without processing further modules M3,.. and go to next screen if any

<- SET SCREEN <nnnn> : Executes the rest of the Modules in PAI then go

to the screen i.e SET

<- CALL SCREEN <nnnn>: Go to the Called Screen and after coming back if, rest of the modules will be executed.

Coming back make sense for modal Dialog box.

MODULE M3.

**LUW(Logical Unit Of Work)**

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

The time Span in which one consistent data state is transferred to another consitent state is a LUW.

ABAP has two types

. Database LUWs realized by DB System <- Database Locks

. SAP LUWs realized by Special ABAP Prog.Techniques<- SAP Locks

LUW Refers collection of actions performed at DB Level as a Complete Unit.

SELECT FROM TABLE A, B

UPDATE TABLE A.

DELETE TABLE B.

All the Statements are recommended to be done as part of one LUW i.e SAP LUW to achieve all or none.

Since DB Changes from DB LUW cannot be reversed in the subsequent LUW, it is recommended make all the DB LUWs as a Single SAP LUW.

UPDATE A.

COMMIT WORK. "DB LUW

UPDATE B.

COMMIT WORK. "DB LUW

DELETE C.

COMMIT WORK. "DB LUW

If commit or rollback work is required at the end of the 3rd DB Operation, Instead of Imm, then SAP LUW.

**Table Control**

------

CONTROLS <Name> TYPE TABLE VIEW USING SCREEN <Screen No>.

LOOP - ENDLOOP must be part of both PBO,PAI.

**TabStrip Control**

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

By Default - 2 tabs and Tab title - for the no of tabs

For Each TAB, Subscreen Area is required to call a SubScreen into it.

**CONTROLS <TS> TYPE TABSTRIP.**

PBO.

<ctrl>-activetab = "Fcode of the TAB to be activated by default"

CALL SUBSCREEN <SA> INCLUDING <Prog><SubScreen No>.

PAI.

CALL SUBSCREEN <SA>.

**SubScreen Restrictions**

**----------------------**

SET PF-STATUS

SET TITLEBAR

SET SCREEN

LEAVE TO SCREEN

CALL SCREEN

These Statements in SubScreen Flow Logic passes through syntax check but throws runtime error.

**Module 3**

--------

1. OO-ABAP

Benefits of OO

. Multiple Instances

. Inheritance

. Encapsulation : Implementation is hidden/separated from Def(Other Comps).

. Polymorphism : ability to have multiple forms and via Interfaces

ABAP allows Single but not multiple inheritance

. Compatibility : OO and Non-OO can be Mixed and only useful OO things can be included

. Maintainability

Classes can be Local(Prog.Specific)/Global(SE24)

CLASS Def. cannot be nested.

CLASS is the abstact description/template of the Obj.

CLASS Defines state and behaviour of Obj.

Every Component of a CLASS can be either

. STATIC via CLASS-

exist only once irrespective of no.instances

. Instance Specific

Exist for each instance once.

Note : Static Methods can access only static attributes and where as

Instance Methods can access both static and instance specific.

Note: Each Component of any class must be part of a visibility section i.e

PUBLIC, PROTECTED, PRIVATE(Default) and in the same order.

Note : Static Components can be accessed from the class itself i.e<CLASS>=> and

where as instance specific things via instance only i.e<obj>->

Note : ME is the reference var, refers the current obj inside the instance methods.

classlcl\_vehicle definition.

public section.

methods : get.

private section.

data make type i.

endclass.

classlcl\_vehicle implementation.

method get.

data make type i.

make = 10. "local variable

me->make = 100. "Instance specfic make

endmethod.

endclass.

--

Functional Method : A method with a returning(pass by Val always)parameter and can have only IMPORTING,

Exceptions but not EXPORTING, CHANGING .

-

Constructor : is a Public Instance method with name constructor, with only Importing , exceptions and is

is called Implictly during the object creation and cannot be called

explictly.

-

Note : One Class can have only one constructor as in ABAP, more than one method is not allowed

with the same Name in the same class eventhough the no.of, type of arguments are same.

-

CLASS-METHODS : CLASS\_CONSTRUCTOR is the static constructor can have only exceptions.

is called only once i.e either during 1st obj. creation or while accessing first

static method/attribute .

Inheritance : is known as "is a" relationship.

A truck is a vehicle.

Allows only single inheritance in ABAP i.e can have only one direct super class.

Note : The empty class "OBJECT" is the Root node for every inheritance tree.

Subclasses are extremly dependent on super class and is also known as "White Box Reuse" i.e

Subclass must posses detailed knowledge of the implementation of super class.

Note : In Subclass , "you can never some thing away from a super class". All the Components

of super class except private section are automatically present in subclass.

Note : Super class methods can be reimplemented(REDEFINITION) in the Subclass but under the

same visibility section as it is in the super class definition.

In Subclass, super class method's interface cannot be changed(Method Overloading not possible)

except in CONSTRUCTOR as the subclass, we cannot have two constructors i.e Super Class's and

Sub Class's . So that the Subclass constructor has to call the super class constructor explictly.

So that the Subclass constructor can have it'ssingature along with super class's.

Note : SUPER can be used Only within the re-defined/re-implemented(is the right word) to

access the direct super class's components in the subclass.

Casting

-------

Upcast/Widening Cast

-----

Super Ref = SubClassInstance .

So that the target var can access more dynamic types in comparision to source var. Hence it is known

as Widening Cast.

Down/Narrowing Cast

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

Cannot be Checked statically and checked at runtime .so the cast operator(?=).

Subclass ref ?= super class .

As the target var can accept less dynamic types after assignment and is known as Narrowing cast.

if the assignment is not possible, an exception CX\_SY\_MOVE\_CAST\_ERROR is raised.

Interfaces : Defines only the Definitions/Protocols but not the implementations.

----------

No Implementations and Implemented in the Public Section of the Class.

Do not have visibility section.

A Class Can implement any of interfaces and an interface can be implemented by any no of classes.

Interfaces can be nested and known as compound interface.

Exception : is a situation where we cannot continue in the normal way.

---------

CX\_ROOT

CX\_NO\_CHECK CX\_DYNAMIC\_CHECK CX\_STATIC\_CHECK

Methods : GET\_SOURCE\_POSITION

GET\_TEXT.

--

Handling Exceptions

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

TRY.

TRY.

CATCH <Ex1> INTO <Obj1>.

CATCH <Ex1> INTO <Obj2>.

CATCH <Ex1> INTO <Obj3>.

CLEAN UP.

"Executed When Exception this block is not caught inside the same block .

ENDTRY.

ENDTRY.

EVENTs : Can be Static or Instance and Can have only exporting parameters and which can be

------ imported by the Event Handler methods.

1) Event Definition - with in the Class Definition

2) Event triggering - with in any method of the class implementation- RAISE EVENT

3) Event Handling - Either from method of the same class or from program/other class which

access the event's obj.

4) Event Handler Method Registration via SET HANDLER.

ALV

---

Non-OO ALV via Function Modules and it's related types are defined in type pool SLIS.

FMs used : REUSE\_ALV\_LIST\_DISPLAY or REUSE\_ALV\_GRID\_DISPLAY

Field Catalog :collection of Display Characterstics and can be prepared in 3 ways.

i) Automatic -> via DDIC Str

ii) Manual -> prepare it\_fieldcat manually i.e everything on our own.

iii) Semi Auto -> via DDIC for FM 'REUSE\_ALV\_FIELDCATALOG\_MERGE'->it\_fieldcat(modify it)

OO-ALV

------

via Container class (CL\_GUI\_CUSTOM\_CONTAINER) and Grid Class(CL\_GUI\_ALV\_GRID->SET\_TABLE\_FOR\_FIRST\_DISPLAY).

CL\_GUI\_ALV\_GRID from CL\_GUI\_ALV\_GRID\_BASE from CL\_GUI\_CONTROL From CL\_GUI\_OBJECT.

Is\_layout type LVC\_S\_LAYO

Is\_layout-zebra = 'X'.

Is\_layout-colwidth\_optimize = 'X'.

If Grid Obj is Initial , Call method : SET\_TABLE\_FOR\_FIRST\_DISPLAY Else.

REFRESH\_TABLE\_DISPLAY, so that no need to create container,

grid instance again, it's enough to refresh the data.

For Non-Event Based Functionality

it\_sort type lvc\_t\_sort.

pt\_fit typelvc\_t\_filt.

3. File Handling

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

Presentation/Frontend files :

DOWNLOAD

--------

FM 'GUI\_DOWNLOAD' fromItab -> File

Exporting

BIN\_FileSIZE =

File\_Name =

type = 'ASC' "default

WRITE\_SEPARATOR = for Download

CONFIRM\_OVERWRITE = for Download

HAS\_SEPARATOR = for upload

APPEND = 'X' "Appends while Download else over writes

FILELENGTH = NO OF bytes transferred

TABLES

DATA\_TAB = <itab>

FILETYPE valid for DOWNLOAD

----

'ASC' : ASCII Format , the table is stored with rows

'DAT' : Same as ASC but with column separation TAB

'BIN' : Binary format(BIN\_FILESIZE has to be passed)

'DBF' : Stored as Database File(always with DOS code page)

'IBM' : as ASC with IBM code page conversion(DOS)

'WK1' : Data is downloaded in Lotus 1-2-3 format

On Presentation Server : UPLOAD

via Function Modules 'GUI\_UPLOAD' from File ->Itab

FILETYPE : Source File type

'BIN' -> Binary Files

'ASC' -> ASCII files, text files with end of line markers

'DAT' -> File is loaded line by line into the transferred table.

Tab in the file mean a change of field.

FILELENGTH

Note : SEPARATOR = SPACE => means no separator and

<> SPACE => TAB Separator

Exceptions

----------

CONVERSION\_ERROR

FILE\_OPEN\_ERROR

FILE\_READ\_ERROR

INVALID\_TABLE\_WIDTH - Invalid table str

INVALID\_TYPE - Invalid File type

Application Server Files/Data Sets/Sequential Files/Global Files

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

Tcode - AL11 - to see all the SAP Directories.

Note : Reading or writing Datasets is always Line by Line.

OPEN DATASET <Additions> "Binary Mode by default ->

FOR INPUT "for reading -> default

FOR OUTPUT "Write/Overwrite

FOR APPENDING "Add to the end of existing File

IN BINARY MODE "Default and used for un-structurred data - bytes by bytes

IN TEXT MODE "Line by Line for Structurred Data

AT POSITION <p> "while reading from a particular pos IN TEXT MODE generally

"Doesn't work for the files where <p>>= 2 Gigabytes

MESSAGE <msg> "collects the status msg after reading/writing

FILTER <f> "Windows NT or UNIX Command eg: COMPRESS from UNIX

CLOSE DATASET

DELETE DATASET

Possible Errors :

. File doesn't exist

. File is a Directory

. File is a Program - currently runnig

READ DATASET <Name> INTO <Var> [LENGTH <v\_len>].

Note : The Actual length of the data object read is placed in the field <v\_len> after read.

LENGTH must be passed as variable else it throws syntax error.

TRANSFER <var> TO <DatasetName> [LENGTH <len>].

<len> can be a variable/value , the length of the data object to be written.

Note : Always a file can be opened only in one mode at a time i.e read/write/append

4. BDC - Batch Data Communication

---

Is a programming technique to load data into SAP.

is a combination of ABAP Programming and built in SAP functionality.

Simulates the act of user entering data into SAP transaction.

Data transfer can be in two - way

SAP -> is Outbound(from SAP)

-> SAP is Inbound(Into SAP)

Methods of transfer Data into SAP(Inbound)

---------

. Batch Input

-> Call Transaction

-> Session Method

Needs an Itab of BDCDATA(PROGRAM,DYNPRO,DYNBEGIN,FNAM,FVAL)

BDCMSGCOLL(tcode,program,msgtyp,msgspra,msgid,msgnr,msgv1,msgv2,msgv3,msgv4)

. Direct Input : via standard programs to post data into SAP.

CALL TRANSACTION

----

CALL TRANSACTION <tcode> USING <IT\_BDCDATA>

MODE 'A/E/N' "A-All Screens default

UPDATE 'A/S/L' "A-Asynchronous default

MESSAGES INTO <IT\_BDMSGCOLL>.

A-Asynchronous - doesn't wait for any updates it produces, to be completed.

- faster in execution

- Not recomonneded for large amount of data because the called transaction

receives no completion message from the update module.

S-Synchronous - reverse of Asynch.

is slower than Asynch but eaiser to analyze errors and recover errors as it

waits for acknowledgement.

L-Local Update - update of DB is NOT processed in a separate process but in the process of

calling program itself.

Note : In Call transaction, errors need to be handled explictly via IT\_BDCMSGCOLL.

Return codes : 0 - Successful

<= 1000 - Error in Dialog program(Called tcode)

>1000 - Error in Batch Input(BDC) Program

SESSION Method

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

. Several transactions can be processed through the same session and where a CALL TRANSACTION

can process only ONE Tcode at a time.

. Unlike Call Trans, Data is not processed immediately and it is only when the session

is processed explictly via SM35 or in background via program 'RSBDCSUB'.

Input for the Program RSBDCSUB

. Session Name -> to be processed

. Date, Time

. Status -> New Sessions/Errors Sessions to be Processed

Function Modules specific to Session

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

FM 'BDC\_OPEN\_GROUP' -> To Open a New Session/Group

Client -> in Which session need to be Processed

GROUP -> Name of the Session to be Created

HOLDDATE -> Locked until that date

KEEP -> Retain the session even after processing successfully

USER -> Authorized user to run the session in background

FM 'BDC\_INSERT' -> To Insert <Tcode> and IT\_BDCDATA into the Session.

TCODE -> The transaction code to be run

POST\_LOCAL -> update data locally i.e in the same session

DYNPROTAB -> IT\_BDCDATA

FM 'BDC\_CLOSE\_GROUP' -> Needs no parameters as it closes the currently opened session.

Session is Asynchrnous in Processing and

Synchrnous in Database update.

Sessions Cannot be generated parallellyi.eBefore closing one session, other session cannot be opened.

The Session cannot be re-opened once it is closed.

Each New call to 'BDC\_OPEN\_GROUP' creates a new session even with the same session name.

Explict Session Processing(SM35)

------

Run/Processing/Display Modes

. Process/Foreground -> Same as 'A'(All Screens)

. Display Errors Only (E)

. Background (N-No Screens)

Note : Batch Input Methods(Both Call Trans and Session Method) runs on the

Same Principle i.e Simulating/Transferring Data via Screens .

Direct Input Method : Data is Not transferred via Screens(No Online Processing)

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

is via Standard Programs.

Incase Of errors, this method provides restart mechanism by executing prog in background only.

To Start these programs - use prog - RBMVSHOW or tcode - BMV0

Direct Input Prog Application

RMDATIND MM

RVAFSS00 SD

RFBIBLOO FI

RAALTDII AM

RKEVEXT0 CO-PA

05. LSMW - Legacy System Migration Workbench

----

Is an SAP tool to transfer data from(Legacy)non-SAP Systems to SAP Systems once or

Periodically.

Data Can be Imported via All fields available Njoytcode support

. Batch Input(Session with Recording) - Yes No

. Direct Input - No No

. BAPIs - No Yes

. IDocs - No Yes

06. SMARTFORMS - TCODE

----------

Improved SAP SRIPTS (SE71)

Note : In Smartforms, a FM(FM Interface) is generated automatically for each

activesmartform and this FM is unique within the system.

Environment -> FM Name to get it

FM Interface is used to communicate between driver program <-> Form

Note : FM 'SSF\_FUNCTION\_MODULE\_NAME' is used to get the FM associated with SForm.

Input ->Smartform Name

Output -> FM Name TYPE RS38L\_FNAM ?

SE78 -> to upload graphic objs from .BMP, .TIF images only

SE73 -> For Creating New Fonts and Bar codes

SMARTSTYELS -> for Smartstyels(collection of paragraph & character formats) and

default style for custom smartfirms is 'SYSTEM'

Navigation Panel

----------

Global Settings

---

. Form Attributes

. Form Interface

IMPORT,EXPORT,TABLES,Exceptions

. Global Definitions

TYPES, Global Data, Forms, INITIALIZATION,FieldSymbols

Pages & Windows

---

Pages

Windows : MAIN(Only One), 2nd ry(default type),Copies,Final

Grphic

Address : from Address number and in Sendorctry's Postal Format

Address Types

. Company Addresses( Type 1)

. Personal Addresses( Type 2)

. Workplaces Addresses( Type 3)

**Main Window**

----

Display data that runs into several pages(Main Window's )

Can trigger page break either implictly , explictly(via NEW-PAGE Command)

Whole Form Can have only on MAIN and the same Can be placed in many Pages

MAIN Window on all the Pages must have the Same Width and Height can differ.

**Note** : A page without main window must not call itself as next page as it

trggers an endless loop,system terminates after processing 3 pages(?).

Except in MAIN Window, the text that do not fit into 2nd ry window are

truncated and not displayed.

Copies Window : Content Can be Printed either in Orginal(1st Copy) and Copy(2nd copy onwards)

----

**Printing of texts**

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

All the texts in the form are displayed using text nodese except address(node).

Note : FM 'SSF\_FIELD\_LIST' - to list all form interface parameters .

**Types Of Texts**

----

.Text Element (default type) -> print from either variable data or static

.Include Text ( Standard text created for SAPSCRIPTs in SO10)

.Text Module( Same as standard text but exclusively for Smartforms)

.Dynamic text for variable texts

Both standard texts and text modules are form independent i.e reusable across forms.

**Control Parameters of Form's FM : Helpful to control Print Dialog .**

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

no\_dialog -> No dialog box for output

preview -> Print Preview

Startpage -> explictly any page can be started

device -> 'PRINTER'/'TELEFAX'/'MAIL'

**Tables & Templates**

-----

Tables for variable size data i.e from Itabs so that Tables make sense in MAIN Windows Only.

Cannot be nested and Each table has 3 Components by default i.e

.Header

.MAIN Area(Executed for each record of Itab looped)

.Footer

Templates for fixed size texts where each row structure is different, not from Itab

**SmartForm Specific System Variables**

---------

&SFSY-PAGE& - Current page of Print job

&SFSY-JOBPAGE& - Total no.of pages of print job

&SFSY-FORMPAGES& - Total no.of Form Pages

**Summary**

**-------**

. Smartforms Architecture

. SAP Form Builder

. Texts, Addresses, Graphics

. Tables & Templates

. Flow Control

. Integrating Smartforms into Application Programs

. Fonts and Barcodes

Adobe Forms - SFP / SE80 . SE80 Can be used for almost all ABAP Workbench tools.

-----------

Along with printer as medium, even Internet(HTML output),fax and an email can be used .

.SE71 - SAP Scripts

.SmartForms - Introduces in SAP 4.6C

.Adobe Forms - SAP Netweaver'04 .

Adobe Forms -> PDF as output by default.

Usage of PDF, will retain their appearance regardless of the environment they are used in.

**Advantages over Smartforms**

----------

Graphics(BMP,JPEG,GIF,PNG,EXIF) also can be included.

Objects( including texts ) can be rotated.

Existing PDF ,WordDocs can be Imported.

Mailing and Faxing is easier.

Scenarios and Integration into browser based applications are possible

(Web Dynpro for java or ABAP).

Every Adobe needs an Interface and the Link between the Driver Program <--> Form.

Interface type can be

. ABAP Dictionary based - for Print Scenario

. Smart Forms Compatible - for Print Scenario

. XML Schema based (mainly Web Dynpro - from Netweaver 2004)

For XML Schema based Interface

----------

Suitable XSD file need to be uploaded.

Cannot have explict parameter and it is the existing importing /BCDWB/DOCXML, i.e All the

business data is transferred to the form at runtime must be passed over with this XSTRING.

It's not possible to add further parameters to XML Interface.

**Context**

------- From Which the Form can access the Data.

So Interface -> Context mapping has to be done .so that form can access Context.

For XML based Interface, there will bo no context and the Interface is available to the

form directly.

**Adobe Designer's Components**

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

1. Master Page

2. Body Page - Topmost subform , wrapper around content area/Dynamic content

3. Content Area - Space provided for Dynamic Content i.e for Itab

4. Subforms - to group the relavantfields .

**Subforms can be repeatable as table header and table rows**.

**Note** :Evenry Form Contains atleast one master page and is created automatically.

biolerplat Objects remains same on all the pages i.e acts as 2nd ry windows.

One Content area is also created by default for master page and is used for dynamic content.

Content area Can be included only on master pages.

Master Page ->Contnet Area -> Body page(top most subform) for dynamic content.

If the subform is part of content area , it is for dynamic content and if it is outside , for static.

Multiple Master Pages

-------

FIRST -> NEXT -> TERMS

Occurance = 1 Occurance(no restriction-loop)

**Note :** If Max.Occurance is restricted to 1, without the next master page, this setting will be

ignored at runtime, if more data needs to be displayed.

**Body Pages and When to Include them**

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

To Force a Page break, possibly using a different master page.

**Pagination of Bodypage/Subform**

**------------------------------**

for 1st body page :

Place - Following Previous

After - Continue Filling Parent

Note : Body Page will use the space provided by a content area . At runtime, if the space is not

enough, the body page will look for the next content area/body page(as MAIN Window),

which involves an automatic page break.

Master Page1(Body Page1) -> Master Page2(Body Page2 ) -> Master Page 3( TERMS )

**Occurance- 1 Occurance - no restriction**

So Page2/Body2 will be repeated

after completion of repetion,

PAGE3 need to be triggered

Explictyly like New-PAGE command

in SCRIPTS/SMARTFORMS.

Body Page2/Subform

Content - Flowed

Allow Page Breaks within Content

Pagination

Place - Top of PAGE(TERMS)(Explict Page triggering)

After - Continue Filling Parent

**When to use Subforms explictly**

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

Subforms can be thought of as folders , containing several objects.

**Placing multiple objs.in the same subform make sense**

When those objs. need to be grouped visually

protect all those objs against page break

To Output all those Objs repeatedly as Table Headers/Rows

To hide all those objs at once

**Types of Subforms**

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

1. Positioned -Obects laid down at their exact position at runtime, relative to subform.

2. Flowed - Objects follow each other, depends on the space they require at runtime

A body page(Top most subform ) is typically of this type.

**Layout**

------

Inserting Fields from Data Preview , So that automatic Link/Binding from Content -> Layout.

**Intergration from ABAP Program**

**------------------------------**

1. CALL FUNCTION 'FP\_FUNCTION\_MODULE\_NAME'

->Input : Adobe Form Name

->Output : FM Generated for Form i.e into V\_FM

2. CALL FUNCTION 'FP\_JOB\_OPEN'

CALL the Generated FM from V\_FM

CALL FUNCTION 'FP\_JOB\_CLOSE'

**Parameters of 'FP\_JOB\_OPEN'**

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

nodialog

noprint

nopdf

copies

**Summary**

**-------**

1. The 1st Master Page is created automatically - TRUE

2. \_\_\_\_ Helps to rearrange several Objects at a different place in the Form . (Subform)

8. Enhancement and Modification

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

Note :Enhancment acts as hook to hand the additional functionality. It will gurantee that the

std.functionality is safe and the additional functionality can be transported , so that

no need to repeat the same amount of work explictly in Quality and Production Servers.

.Changing the Standard System

.Personalization - User Sepecific Menus, Transaction,Screen Variants.(Tcode - SHD0)

.Enhancements to Dictionary Elements

Program /Source CoudeEnhancement Techniques

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

Adjust the SAP System to meet the Enterprize requirements

Customization - Via Implementation guide - tcode - SPRO - done by functional consultants in general

personalization - What you see is , what you need . ,

Simplifying navigation(user specific menus),

Simplifying transactions(transaction variants,screen variants)

.Hide Fields,Menus ...

Modification

Enahance the SAP System to meet the Enterprize requirements

Exits/BADIs - Enhancments "ABAP Work Bench

When there is no standard solution then

Customer Develoments - New Developments "ABAP Work Bench

1. Exits (Non - OO )

. User Exits - via Subroutine calls - for SD Module initially but needs access key to implemnt

Only for Program Enhancment

. Customer Exits - Via Function Calls

CALL CUSTOMER-FUNCTION '001' =>Same as 'EXIT\_<progName>\_001'

FUNCTION 'EXIT\_<progName>\_001'

INCLUDE ZX.... "Custom Include not yet created and we create it

"to implement the Additional Custom Logic

ENDFUNCTION. "No Access key required as it is our include .

Customer Exits Can be used to Enhance

.Menus via Menu Exits via fcodes begins '+'

.Screens Via Screen Exits via Custom Subcreens into subscreen Areas

CALL CUSTOMER-SUBSCREEN '0110'

.Programs Via Function Module Exits

Note : Along with Menu Exit, Screen Exit also , a FM exit is associated to implement the respective

functionality.

Note : Before Implementing any exit, the right exit need to be searched/located.

Tcode - SMOD For Locating/Searching the Customer Exits

- CMOD For Implementing the Customer Exits and can be used for Searching/Locating also.

Searching the exits for any tcode is at package level i.eTcode ->Program -> Package -> SMOD/CMOD

**Note :Implenentation of an exit is known as a Project.**

**BADIs(Business ADD Ins) - OO**

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

Same as Customer Exits but Object Oriented.

Tcode - SE18 For Search/Locate BADI Definition

- SE19 For Implementing the BADI

Note : Both SE18,SE19 Can be done through SE80.

Note : BADI Definition is Interface Definition i.e Empty Methods, Since the Same Interface can have

Multiple Implementations, BADI Can be Multiple use i.e Can have multiple implementations and

where as the Same Exit can have Only one Imeplemntation/Project.

BADIs

-----

Can be Multiple use except Menu BADI(Single Use)

If Multiple use, can be filter dependent also.

**Note** : If the BADI is Multiple use, all the Implementations are called one by one,if filtered, as per

the condition but the Order of Calling the Implementations can be any.

Enhancements to ABAP Dictionary

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

. APPEND Structure

. CI Include Structures - already integrated into Standard Tables but not yet created.

**Note** : APPEND Str. is table specific and where as CI Include Str , can be included in any no.of tables.

**Note** : During the upgrades, the append str fields are available, when the std.tables are

activated for the 1st time.

**Note :** APPEND Str cannot be created for Pooled and Cluster Tables.

Always a Long field(LCHAR,LRAW) should be at the end of the table, if any long field exist

in the std.table, Append Str is not allowed as nothing can be added after long field.

Field Names in the APPEND Str/CI Include are recommended to begin with 'YY' or 'ZZ'.

**Note :** Overwriting SAP Field Labels/Texts(Short,med,Long) can be done through tcode - CMOD

Goto ->Glob.Enhancments -> Keywords -> Change

**Note** : Enhance 2nd ry Index from SE11 itself is allowed from SAP Netweaver 7.0,

Enhance Domain's Fixed values via Fixed Value appends

1. \_\_\_\_ allows you attach your functions to menu options in SAP Menus(Menu Exit/Menu BADI)

2. \_\_\_\_ allow to implement additional logic in application functions/progs (Function Module Exit/BADI)

BADIs

----- Introduced in 4.6 and is the OO way of Enhancing std.Menus,Screens and Programs.

BADI is technically , an interface i.e collection of method definitions.

BADI has an adapter class , which calls all the active Implementations

i.e CL\_EXIT\_HANDLER=>GET\_INSTANCE "Can search for this statemenet also to locate BADI

BADIs are same exits i.e it is for

. Program Enhancement

. Screen Enhancement

. Menu Enhancement

Note: No need to register either BADIs/Exits in SSCR and the registration is required for Modifications.

BADIs Can be Multiple use and Filter Dependent except Menu BADI and where any exit is Single use only.

Finding Standard BADI Definition Can be through

--------

1. CL\_EXIT\_HANDLER=>GET\_INSTANCE "Searching this directly in the source code

2. Via SE24 , by keeping break point in the above GET\_INSTANCE Method

3. Via SQL Trace - ST05

4. Via SE80/SE84 - Repository Info from the Package

Custom BADI Can be Created from SE18 but ideally has to be done by SAP ,

as SAP Defines the BADI(SE18) and We Implement it(SE19).

Enhancement Spots From ECC 6.0 for Source Code Enahancements

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

Enhancement Spot Can contain one or more BADIs. can be Defined in SE18 and Implemented in SE19.

Enhancement Spot can be Implict, Explict.

Implict spots given by SAP, at the beginning, end of Standard every program,

FORM - ENDFORM,

FUNCTION-ENDFUNCTION,

METHOD - ENDMETHOD.

Explict spot can be via BADI or via statement called ENHANCEMENT-POINT <Name>.

Note : Menu BADI is same as Menu Exit even it is also single use , so no filter make sense.

Note : To Display, the already existing Implementations for any BADI,

SE18 -- BADI Name --- Then Implmentations Overview.

Note : Multiple use BADI should contain only Importing FLT\_VAL but not Export, Returning parameters.

Q1. \_\_\_ Tcode is used to Define BADI (SE18)

Q2. \_\_\_ Tcode is used to Implement BADI(SE19)

10. ALE IDocs - ALE for Explict Integration and IDoc acts as carrier for the Data.

ALE is via tRFC(Transactional/Asynchronous RFC ), which always gurantees the delivery by

monitoring receiver sys. always and it tRFC port means via Memory Buffers internally.

Same as an SMS to mobile, Even the mobile is switched of that time, later it will be delivered.

IDoc is an instance of IDoc type.

Each IDoc has a unique Idoc number with 3 types of records. i.e

. Control Records - Always 1 for each IDoc - in table EDIDC

. Data Records - <n>> 0 - in table EDID4

. Status Records - <n>> 0 - in table EDIDS

IDoc type is Collection of Segment types and Segment type size cannot be more than 1000 Chars.

ALE Comprises of 3 Layers

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

1. Application Layer - up to generating application docs like MM01 for Materials,ME21N- POs

and Master IDocs

2. Distribution/ALE Service Layer - Both master and Comm.IDocs by reading Receivers from Distrubution Model

and Filters in the DM.

3. Communication Layer - Dispatch IDoc as per port/RFC Desition type i.etRFC,TCP/IP .

ALE uses -tRFC between SAP <-> and TCP/IP is generally used between SAP <-> Non-SAP.

List of frequently used Tcodes

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

SNO Purpose Tcode

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

1. Define Logical Systems SALE

2. Assign Log.Systems to Client SALE

3. RFC Connections SM59/SALE

4. Distribution Model BD64

5. Message type WE81

6. Partners & Partner Profiles WE20

(Both Outbound , Inbound )

7. Ports WE21

8. IDoc type WE30

9. IDoc segment/Segment type WE31

10. Message type WE81

11. IDoc type + Message type WE82

12. IDOc Monitoring/IDoc List WE02/05/07/09

13. ALE Monitoring BD87

(Reprocess failed IDoc)

14. IDoc test tool WE19

15. IDoc Documentation WE60

16. Area Menu WEDI

**EDI - Electronic Data Interchange**

---

Is Non-SAP's technology to exchange the Business Documents among companies using ElectrnicComm.Sys

Note : ALE is within the Company and EDI is across the Company.

Note : Between ALE and EDI , Main difference occurs at communication level. i.e

ALE transmits IDocs to an SAP System via Memory Buffers using Asynch.Comm(tRFC port) and

No SubSystem is required ,

Where EDI transmits IDocs to an EDI Subsytem using Flatfile Format.

Q1. \_\_\_\_ is used for IDOc Documentation (WE60)

Q2. \_\_\_\_ is used for ALE Monitoring/reprocessing (BD87)

Q3. \_\_\_\_\_ is used for IDoc Monitoring/IDoc list - WE02/05/07/09

Q4. \_\_\_\_\_ is IDoc test tool (WE19)

Note : Master IDoc is an Internal table , so that not saved in any DB Table.

**11. Performance Tuning**

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

It is optimizing the performance of your application through various techniques

to increase the productivity of the user.

Most systems respond to increased load with some degree of decreasing performance.

A System's ability to accept higher load is called Scalability/Performance tuning.

Note : Both Presentation, Application layers are scalable and where as the Databse Layer,

as the central data repository , IS NOT SCALABLE with improved hardware , so

appilication program level the improvements must be done.

Goals

-----

To reduce the runtime of progs on applications server there by reducing the CPU load.

To reduce the database load, which is particulary importance as it is NOT SCALABLE.

Refer the defintionsof . Response time, load time, wait time ...

----

Follow the Golden Rules like

--------

1. Avoid nested SELECTs

2. use SELECT Single , if all the primary key fields can be supplied

3. use buffered tables if possible

4. Select all the required records in a single shot , instead of SELECT - APPEND - ENDSELECT

5. Use aggregate functions i.e MIN,MAX,AVG,SUM,COUNT ..

6. Use Explict work ares, no header lines, no occurs, read with BINARY SEARCH

7. Modify ITAB Only with required fields via TRANSPORTING

8. use DELETE ADJACENT DUPLICATES instead of LOOP - DELETE - ENDLOOP.

9. Copy Itabs APPEND LINES, INSERT LINES OF , ITAB1 = ITAB2 , instead of LOOP - APPEND - ENDLOOP

10. For Signle read acccess , HASHED Tables are better than SORTED Itabs

11. For Partial Sequential access ,SORTed Tables are more Optimized compared to HASHed.

12. TYPEed formal parameters are Faster than Un-TYPEed.

13. CASE is bit better than IF-ELSE-ENDIF

14. Use of WHILE , instead of DO+EXIT - Construction

15. Use Of I instead of F, when really need Integers

16. Use of Constants instead of Literals

17. Use N , when pure digit strings are required but intended for calculations

18. String operations are faster by specifying char length than strings

19. Calling Global Methods Faster then Calling Function modules

20. Calling Local Methods is Faster Global Methods

Tools For Performance Analysis

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

1. SCI - Code Inspector - Checks for Quality of Code

2. SLIN - EPC(Extended Program/Syntax Check) - verfiyobselte statements,(vars,forms declared but not used)

Both Code Inspector and EPC can do only Static Code Check i.e can recommend only problematic statements.

3. ST05/SAT - SQL /Performance Trace - Can give the run for Only SQL Statements but for all

4. SE30 - Runtime Analysis - Can give over all time taken at APP.Server, DB Server ...

**FOR ALL ENTRIES**

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

Advantages : Large amount of data

mixing processing and reading data

Fast Internal processing of data

Fast

Disadvantages:

Difficult to program or understand

Memory could be critical i.e use FREE or PACKAGE SIZE

IF <source\_itab> is NOT INITIAL.

select from <DB Table> INTO TABLE <target\_itab>

FOR ALL ENTRIES IN <source\_itab>

WHERE <Condition on Common field between DB Table and <Source\_Itab>.

ENDIF.

Things to keep in mind with FOR ALL ENTRIES

-------

1. IF <source\_itab> is NOT INITIAL. "is mandatory , else for the enpty<source\_itab>,

it will fetch ALL the RECORDS from DB Table instead of selecting NO RECORDs.

2. SELECT with FOR ALL ENTRIES as SELECT DISTINCT

SELECT = SELECT DISTINCT , only for Primary key selection.

So, it is mandatory to include at least the PRIMARY KEY Fields in the Selecition List.

**12. Transports**

**----------**

. Basics of Transport Control

. Transport with Import Queue

. Procedure of Mass or Single Import

. Transport Strategy

. Transport Organizer

. Types Of Request

SAP's Transport management system represents the centralized Change& Transport System(CTS)

for all R/3 Systems.

CTS is a tool , that helps to organize development projects in the ABAP Workbench and in

Customization, then transport the changes between the SAP Systems in the Landscape.

CTS records, all changes in the change requests.

Changes in Change requests can be linked together logically or can be completely independnent of

each other.

Once the development is finished, you can release the task/correction(first) and request respectively.

3 - System Landscape : Development -> Quality -> Production

2 - System Landscape : Development -> production

**Transport Layer** : is at the package/development class level, which decides the target system.

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

If a package has no transport layer, the objects in it are local i.e cannot be transported .

Coming back make sense for modal Dialog box. ---------------------------

Any Changes to Customization / Repository obj in the development sys. do not automatically

transported to target sys.

Two Procedure to use Import Queue by BASIS team

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

1. Mass Import 2. Single Import

Transport Organizer(SE09/SE10) : Tool to organize the transport request releases.

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

**Types of Requests**

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

1. Workbench Requests

for all the modifications made to repository and customizing settings during development.

2. Customizing Requests

for all the modifications made to Customizing/System settings from Client specific tables.

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

3**. Transport Copies**

Transport Objects to a Specified SAP System

original Location of the object will remain

No Delivery to another SAP System

**4. Relocations**

**Note : Status of the request can be**

.Modifiable - Not yet released, so can be changed without any new request/task

.Release - Already released, so changes need to be collected via new request/task.

**usefultcodes**

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

STMS - Transport Mgt.Systemconfig. - done by BASIS Team in general

SE01 - Transport Organizer(Extended View)

SE09/SE10 - Transport Organizer

SE03 - Transport Organizer Tools (To Merge More than One Request into a Single)

Q1.

Client Can be devided into 3 categories , namely

. Development , Quality and Production

. \_\_\_\_ is the tcode for Transport Organizer(SE01/SE09/SE10)